DEMOCRITUS UNIVERSITY OF THRACE

DOCTORAL THESIS

Extending Named Data Networking with delay-/disruption-tolerant and reputation-based trust mechanisms

---

Author:
Christos-Alexandros Sarros

Supervisor:
Dr. Vassilis Tsaoussidis

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

in the

Department of Electrical and Computer Engineering
Democritus University of Thrace

This research was partly funded by the Greek State Scholarships Foundation (IKY) action "Supporting human research resources through doctoral research", Operational Programme "Developing Human Resources, Education and Lifelong Learning", 2014–2020.

"I don't know anything, but I do know that everything is interesting if you go into it deeply enough."

Richard Feynman

Abstract

Nowadays, the Internet constitutes a vital part of our everyday lives and an increasing number of social functions and activities are being performed online. However, the underlying architecture supporting our interconnected digital world was designed several decades ago, before its commercial explosion.

At the core of the Internet lies the TCP/IP architecture, a protocol stack which supports this "network of networks" and the variety of applications that are being built on top of it. The way that devices located in different networks communicate with each other is by implementing a common layer based on the Internet Protocol (IP). This design has enabled protocol innovation above and below this common layer, with multiple new protocols being proposed and adopted over the years. But it has also resulted in an ossification of the middle layers - Layers 3/4 of the OSI model - which have remained largely unchanged during those years. The resulting architecture is often visualized as having an hourglass shape, with several protocols in the upper and lower layers and only a handful in the middle.

It can be argued that this approach was largely successful, as it has proven highly scalable and capable of supporting a large variety of applications. However, the fact that TCP/IP originated in the 1970s means that its design was inherently limited by the capabilities of that time. In particular, the focal points of the architecture's thin waist - IP addresses - conceptually reflect the endpoints of that day's fixed telephone networks, whose infrastructure provided the backbone for the early Internet.

Yet, Internet usage has largely changed those years. Nowadays, connecting to a remote host is not the primary aim of an Internet user. To a large and increasing extent, this goal has been substituted by the need for content retrieval. Moreover, the hosts are no longer stationary as mobile computing devices have been the norm for more than a decade. To accommodate needs such as host mobility and content retrieval, elaborate mechanisms have been designed on top of the TCP/IP Internet infrastructure. Still, the fact that the underlying fabric was designed for a different purpose, introduces additional complexity to the architecture.

This insight has given rise to proposals to replace the current Internet hourglass with a design that is more aligned with its current use, removing the unnecessary complexity and potentially increasing its performance. In this context, Information-Centric Networking architectures emerged. In Information-Centric Networking the focus is shifted to "what" is being transferred by the network, instead of "where" the data is being transferred to.

Named Data Networking is the most prominent Information-Centric Networking architecture. In Named Data Networking, the network layer operates directly on named content objects, instead of host addresses. Thus, it adopts a clean-slate approach designed to replace TCP/IP with content-oriented communication primitives.

In this dissertation we enhance the Named Data Networking architecture with various mechanisms, aiming to address some of its shortcomings. In particular, we

focus on two main issues. The first one is its lack of native support for communication in intermittently-connected wireless networks, due to its original focus was on well-connected fixed networks. The second one is its trust and security framework, which is open to specific attacks due to its sole reliance on cryptographic signatures and digital certificates.

In the first part of this dissertation, we propose the use of a layered design which leverages the existing work on Delay Tolerant Networking to achieve NDN's operation in challenged environments. This design, NDN-over-DTN (NoD), uses the Bundle Protocol as a transport option in intermittently-connected networks. Compatibility with the original NDN architecture is maintained, while the design can flexibly support intermittently-connected devices with various levels of resource capabilities.

Our research shows that this design can be effectively used to support various use cases, such as service deployment in a disrupted Mobile Edge Computing environment and content retrieval from intermittently-connected Internet of Things devices. It is capable of handling asymmetric routing paths and opportunistic connectivity, including scenarios with short network contacts. Its performance outperforms current host-centric DTN solutions and shows further improvement when leveraging cross-layer information.

In the second part of this dissertation, we propose the use of reputation-based trust mechanisms to complement NDN's default trust framework. To this end, we place our focus on two specific use cases: Dynamic Adaptive Streaming and Peer-to-Peer content distribution. In the first one, we assume a scenario in which Bitrate Oscillation Attacks take place and design a reputation-based mechanism which leverages adaptive caching by the routers to mitigate the attack. In the second one, we assume a scenario in which the network is plagued by Content Poisoning Attacks. In this case, we design a blockchain-based reputation system which is able to identify malicious behavior and reward honest users.

Our evaluation results demonstrate that in both scenarios, reputation can serve as an effective means for attack mitigation. In the first case, we show that our approach can potentially achieve similar levels of performance with a no-attack scenario. In the second case, we show that our mechanism is able to correctly identify malevolent actors and mitigate both full-on and on-off attacks, while consistently providing financial rewards to honest users.

Περίληψη

Το Διαδίκτυο αποτελεί σήμερα ένα ζωτικό κομμάτι της καθημερινής μας ζωής, καθώς ολοένα και περισσότερες κοινωνικές δραστηριότητες διεξάγονται μέσω αυτού. Ωστόσο, η αρχιτεκτονική που υποστηρίζει αυτό τον διασυνδεδεμένο ψηφιακό κόσμο σχεδιάστηκε αρκετές δεκαετίες νωρίτερα, πολύ πριν την ευρεία εξάπλωσή του.

Στην καρδιά του Διαδικτύου βρίσκεται η αρχιτεκτονική TCP/IP, μια στοίβα πρωτοκόλλων η οποία υποστηρίζει το λεγόμενο ΄δίκτυο των δικτύων' και την πληθώρα των εφαρμογών που αναπτύσσονται πάνω από αυτό. Οι υπολογιστικές συσκευές που βρίσκονται στα διάφορα δίκτυα επικοινωνούν μεταξύ τους υλοποιώντας ένα κοινό πρωτόκολλο, το πρωτόκολλο IP, το οποίο αποτελεί και τη βάση ενός κοινού επιπέδου επικοινωνίας για όλες τις διαδικτυακές συσκευές (το λεγόμενο επίπεδο 3 του μοντέλου OSI).

Ο τρόπος με τον οποίο σχεδιάστηκε αυτή η αρχιτεκτονική επέτρεψε την καινοτομία σε ανώτερα και κατώτερα επίπεδα της στοίβας, με αρκετά πρωτόκολλα να αναπτύσσονται και να υιοθετούνται ως πρότυπα επικοινωνίας ανά τα χρόνια. Ωστόσο έχει ταυτόχρονα οδηγήσει και σε μία ¨αποστέωση' των μεσαίων επιπέδων (3 και 4 στο μοντέλο OSI), καθώς αυτά έχουν παραμείνει πρακτικά απαράλλακτα εδώ και δεκαετίες. Η αρχιτεκτονική που προέκυψε από αυτή τη διαδικασία περιγράφεται συχνά ως μια αρχιτεκτονική με σχήμα κλεψύδρας, η οποία περιλαμβάνει αρκετά πρωτόκολλα στα ανώτερα και κατώτερα επίπεδα αλλά ελάχιστα στα μεσαία.

Αυτή η προσέγγιση μπορεί να θεωρηθεί επιτυχημένη, καθώς έχει αποδειχθεί ικανή να επεκταθεί σε δισεκατομμύρια συσκευές και να υποστηρίξει μια τεράστια πληθώρα εφαρμογών. Ωστόσο, από το γεγονός ότι δημιουργήθηκε τη δεκαετία του 1970 μπορεί εύκολα να γίνει κατανοητό ότι ο σχεδιασμός της αντικειμενικά περιοριζόταν από τις δυνατότητες της εποχής. Συγκεκριμένα, στην έννοια των διευθύνσεων IP - οι οποίες αποτελούν το κεντρικό σημείο όλης της αρχιτεκτονικής - αντανακλούνται τα σταθερά άκρα των τηλεφωνικών δικτύων εκείνων των ημερών, τα οποία αποτέλεσαν και την υποδομή για το πρώιμο Διαδίκτυο.

Όμως αρκετά έχουν αλλάξει στο Διαδίκτυο από εκείνη την περίοδο. Σήμερα, ο στόχος ενός τελικού χρήστη δεν είναι η σύνδεση σε κάποιο απομακρυσμένο μηχάνημα, αλλά η πρόσβαση σε περιεχόμενο ανεξάρτητα από του που βρίσκεται. Επιπλέον, τα δικτυωμένα μηχανήματα δεν είναι πλέον αποκλειστικά σταθερά, καθώς οι φορητές συσκευές αποτελούν μέρος του Διαδικτύου εδώ και πάνω από μία δεκαετία. Για να υποστηριχθεί η φορητότητα των διαδικτυωμένων συσκευών καθώς και η ανάκτηση περιεχομένου, σύνθετοι μηχανισμοί έχουν αναπτυχθεί πάνω από την αρχιτετονική TCP/IP. Ωστόσο, το γεγονός ότι η καρδιά της αρχιτεκτονικής σχεδιάστηκε για να καλύψει διαφορετικές ανάγκες, σημαίνει ότι εισάγεται αχρείαστη πολυπλοκότητα για να λυθούν τα προβλήματα που προκύπτουν.

Αυτή η συνειδητοποίηση οδήγησε σε αρκετές προτάσεις να αντικατασταθεί η σημερινή αρχιτεκτονική του Διαδικτύου. Αυτές οι προτάσεις καλούν να πάρει τη θέση της μπορεί μια αρχιτεκτονική η οποία θα ήταν καλύτερα ευθυγραμμισμένη με την τωρινή χρήση του Διαδικτύου. Μέσα από αυτή τη διαδικασία προέκυψαν οι αρχιτεκτονικές Πληροφοριοκεντρικής Δικτύωσης (Information-Centric Networking). Η Πληροφοριοκεντρική

Δικτύωση εστιάζει περισσότερο στο ¨τι' είναι αυτό που μεταφέρεται από το δίκτυο, παρά στο ¨που' πρέπει να μεταφερθεί.

Η πιο διαδεδομένη αρχιτεκτονική Πληροφοριοκεντρικής Δικτύωσης είναι η αρχιτεκτονική Named Data Networking (NDN). Στο NDN, το επίπεδο δικτύου χρησιμοποιεί απευθείας ονομασμένα αντικείμενα περιεχομένου, αντί για διευθύνσεις μηχανημάτων. Το κάθε πακέτο έχει ένα συγκεκριμένο όνομα και οι λειτουργίες προώθησης και δρομολόγησης λειτουργούν απευθείας με βάση τα ονόματα των αντικειμένων. Αυτό δίνει τη δυνατότητα το περιεχόμενο να μπορεί να ανακτάται όχι μόνο από τους αρχικούς διακομιστές, αλλά και από ενδιάμεσους δρομολογητές οι οποίοι μπορούν να αποθηκεύουν τα αντικείμενα περιεχομένου στην κρυφή τους μνήμη. Άλλα βασικά χαρακτηριστικά της αρχιτεκτονική αποτελούν η έναρξη της επικοινωνίας από τους παραλήπτες των δεδομένων, καθώς και η πληροφοριοκεντρική προσέγγιση στη δικτυακή ασφάλεια (η οποία επικεντρώνει στην ασφάλεια των ίδιων των δεδομένων, αντί του καναλιού που τα μεταδίδει).

Σε αυτή τη διατριβή επεκτείνουμε την αρχιτεκτονική Ναμεδ Δατα Νετωορκινγ χρησιμοποιώντας διάφορους μηχανισμούς, προκειμένου να αντιμετωπίσουμε συγκεκριμένες αδυναμίες της. Εν προκειμένω, εστιάζουμε σε δύο βασικά ζητήματα. Το πρώτο αφορά την έλλειψη εγγενούς υποστήριξης για επικοινωνίες σε συνθήκες διαλείπουσας συνδεσιμότητας η οποία συναντάται συνήθως σε ασύρματα περιβάλλοντα, καθώς η αρχιτεκτονική αναπτύχθηκε κυρίως με βάση τις απαιτήσεις καλά συνδεδεμένων δικτύων. Το δεύτερο αφορά το πλαίσιο εμπιστοσύνης και ασφάλειας που έχει αναπτυχθεί για την αρχιτεκτονική, το οποίο βασίζεται αποκλειστικά σε κρυπτογραφικές υπογραφές και ψηφιακά πιστοποιητικά και παραμένει ευάλωτο σε συγκεκριμένες επιθέσεις

Ορισμός Προβλήματος:

Το πρώτο πρόβλημα με το οποίο καταπιανόμαστε στη συγκεκριμένη διατριβή αφορά την επέκταση της αρχιτεκτονικής NDN προκειμένου να καταστεί ανεκτική σε καθυστερήσεις και διακοπές στη συνδεσιμότητα. Τα περιβάλλοντα τα οποία απαιτούν το συγκεκριμένο χαρακτηριστικό είναι πολυάριθμα. Για παράδειγμα, οι διακοπές στη συνδεσιμότητα είναι δύσκολο να αποφευχθούν σε απομακρυσμένες τοποθεσίες όπως βουνά ή δάση, στις οποίες πλέον αναπτύσσονται τακτικά αισθητήρες (π.χ. για εφαρμογές συλλογής περιβαλλοντικών δεδομένων). Επιπλέον, σε φυσικές καταστροφές μπορεί να συμβεί τόσο η κατάτμηση του δικτύου, όσο και έντονη συμφόρηση, τα οποία δύναται να οδηγήσουν σε αδυναμία παροχής συνεχούς, αδιάλειπτης συνδεσιμότητας. Ωστόσο, σε τέτοια σενάρια η ικανότητα επικοινωνίας καθίσταται εξαιρετικά απαραίτητη. Τέλος, τέτοιες συνθήκες απαντώνται ακόμη από τους κατοίκους αραιοκατοικημένων περιοχών και αναπτυσσόμενων χωρών, στις οποίες δεν έχουν αναπτυχθεί ακόμη οι απαιτούμενες υποδομές πρόσβασης στο Διαδίκτυο καθώς η εγκατάστασή τους υπόκειται στη σχέση κόστους-οφέλους και κρίνεται ασύμφορη από τους παρόχους.

Τέτοιες περιπτώσεις, στις οποίες είναι εξαιρετικά δύσκολη - έως και αδύνατη - η επίτευξη αδιάλειπτης συνδεσιμότητας, δεν προβλέφθηκαν στον αρχικό σχεδιασμό της αρχιτεκτονικής NDN. Έτσι, αυτή αδυνατεί να ανταπεξέλθει στις απαιτήσεις τους με βάση τον αρχικό της σχεδιασμό. Αυτό φαίνεται κι από συγκεκριμένους μηχανισμούς που ενσωματώθηκαν σε αυτή, όπως η χρήση συμμετρικών μονοπατιών για δρομολόγηση κατά την ανάκτηση αντικειμένων περιεχομένου. Αυτό σημαίνει ότι τα πακέτα αιτημάτων για κάποιο αντικείμενο περιεχομένου σχηματίζουν ένα συγκεκριμένο μονοπάτι, το οποίο ακολουθείται αντίστροφα για τη δρομολόγηση των απαντήσεων με τα πακέτα δεδομένων. Αυτή η προσέγγιση όμως απαιτεί μια σχετική σταθερότητα στις τοπολογίες, η οποία συχνά δεν υπάρχει σε συγκεκριμένους τύπους δικτύων ανεκτικών στην καθυστέρηση (όπως τα ευκαιριακά δίκτυα που σχηματίζονται από φορητές συσκευές).

Η υποστήριξη αυτών των περιβάλλοντων, διατηρώντας ταυτόχρονα τη συμβατότητα με την αρχική αρχιτεκτονική NDN, αποτελεί ένα ζητούμενο στο οποίο δεν έχει βρεθεί

ακόμη κάποια κοινά αποδεκτή λύση. Η παρούσα διατριβή προτείνει έναν τρόπο επίλυσής του, χρησιμοποιώντας μια προσέγγιση πολλαπλών επιπέδων.

Το δεύτερο πρόβλημα με το οποίο καταπιάνεται η παρούσα διατριβή, αφορά την επέκταση του πλαισίου εμπιστοσύνης του NDN με μηχανισμούς βασισμένους στην αξιοπιστία (reputation). Στόχος είναι η ενίσχυση των τρωτών σημείων του παρόντος πλαισίου, το οποίο βασίζεται αποκλειστικά στα ψηφιακά πιστοποιητικά και ιδιαίτερα στην επαλήθευση κρυπτογραφικών υπογραφών, παραμένοντας ευάλωτο σε συγκεκριμένες επιθέσεις.

Συγκεκριμένα, οι υπάρχοντες μηχανισμοί ασφαλείας του NDN βασίζονται στην απαίτηση κάθε πακέτο Δεδομένων να μεταφέρει μια ψηφιακή υπογραφή, η οποία δημιουργείται από τον ¨παραγωγό' του τη στιγμή που δημιουργείται και το πακέτο των δεδομένων χρησιμοποιώντας ένα κρυπτογραφικό κλειδί που του ανήκει. Ο παραλήπτης αυτού του πακέτου επαληθεύοντας το κλειδί, μπορεί να επαληθεύσει την ακεραιότητα και τη γνησιότητα των δεδομένων. Η εξασφάλιση της εμπιστευτικότητας είναι επίσης δυνατή, με την (προαιρετική) κρυπτογράφηση του περιεχομένου. Πρακτικά, η προσέγγιση εμπιστοσύνης που ακολουθείται συνδέει το όνομα ενός πακέτου με το περιεχόμενό του, καθώς και με τον παραγωγό του, επιτρέποντας στο NDN να αποσυνδέσει την εμπιστοσύνη σε ένα αντικείμενο δεδομένων από το πώς και από πού έχει ανακτηθεί αυτό.

Ωστόσο, κάποια από τα χαρακτηριστικά του NDN το καθιστούν ευάλωτο σε νέου τύπου επιθέσεις. Ένα τέτοιο χαρακτηριστικό είναι η δυνατότητα αποθήκευσης δεδομένων σε κρυφή μνήμη εντός του δικτύου (in-network caching). Αυτό το χαρακτηριστικό υλοποιείται από όλους τους δρομολογητές NDN ώστε να αυξήσει την απόδοση της αρχιτεκτονικής. Καθώς όμως το υπολογιστικό κόστος κρυπτογραφικών ενεργειών είναι σημαντικό, η επαλήθευση των υπογραφών από τους δρομολογητές δεν είναι υποχρεωτική για κάθε πακέτο, καθώς μια τέτοια απαίτηση θα επηρέαζε την ικανότητά του να προωθούν πακέτα με την ταχύτητα της γραμμής. Συνδυασμένα, τα δύο αυτά χαρακτηριστικά δίνουν τη δυνατότητα σε επιτιθέμενους να σχεδιάσουν επιθέσεις κατά τις οποίες εισάγουν κακόβουλα πακέτα στις λανθάνουσες μνήμες των δρομολογητών, γεγονός που τις καθιστά ιδιαίτερα ευάλωτες από άποψη ασφαλείας.

Στην παρούσα διατριβή επικεντρώνουμε σε δύο νέα είδη επιθέσεων που αξιοποιούν τα κενά ασφαλείας στο υπάρχον πλαίσιο του NDN και δείχνουμε ότι μπορούν να αντιμετωπιστούν με τη χρήση μοντέλων αξιοπιστίας (reputation models). Το πρώτο είδος επίθεσης αφορά τις Επιθέσεις Ταλάντωσης του Ρυθμού Μετάδοσης (Bitrate Oscillation Attacks / BOAs).

Οι συγκεκριμένες επιθέσεις εκμεταλλεύονται τη συμπεριφορά του πρωτοκόλλου μετάδοσης πολυμέσων Dynamic Adaptive Streaming (DAS) σε δίκτυα NDN. Το πρωτόκολλο DAS έχει σχεδιαστεί ώστε να προσαρμόζει την ποιότητα των μεμονωμένων κομματιών ενός αρχείου πολυμέσων με βάση τη διαθέσιμη ταχύτητα μετάδοσης. Στις επιθέσεις BOA ο επιτιθέμενος ζητά εκ των προτέρων κάποια μη-συνεχόμενα κομμάτια ενός βίντεο, έτσι ώστε να αποθηκευθούν στην κρυφή μνήμη των δρομολογητών οι οποίοι βρίσκονται στη διαδρομή τους προς τον παραλήπτη. Όταν ο έντιμος χρήστης ζητήσει μετέπειτα τα κομμάτια ενός βίντεο, κάποια από αυτά θα παραληφθούν από τους δρομολογητές καθώς θα βρίσκονται ήδη εκεί. Αυτό θα δημιουργήσει ανακρίβειες στην εκτίμηση του ρυθμού μετάδοσης από το πρωτόκολλο DAS και ως συνέπεια ο παραλήπτης θα καταλήξει να λαμβάνει διαδοχικά κομμάτια υψηλής και χαμηλής ανάλυσης, δημιουργώντας αυξομοιώσεις στην ανάλυση του τελικού προβαλλόμενου βίντεο. Ως αποτέλεσμα, επηρεάζεται αρνητικά η Ποιότητα Εμπειρίας (Quality of Experience / QoE) του τελικού χρήστη.

Το δεύτερο είδος επίθεσης το οποίο εξετάζουμε, είναι οι Επιθέσεις Δηλητηρίασης Περιεχομένου (Content Poisoning Attacks) και οι Επιθέσεις Δηλητηρίασης Κρυφής Μνήμης (Cache Poisoning Attacks). Παρόμοιες επιθέσεις ήταν δυνατές και σε προηγούμενα δίκτυα προσανατολισμένα στο περιεχόμενο (π.χ. P2P δίκτυα διαμοιρασμού αρχείων). Ωστόσο, δεν αφορούσαν την ίδια την αρχιτεκτονική TCP/IP και επηρέαζαν μόνο το επικαθήμενο

δίκτυο (overlay network) που αναπτυσσόταν πάνω από αυτή. Σε αυτού του είδους τις επιθέσεις, οι επιτιθέμενοι στοχεύουν στη διάδοση ψευδούς ή/και κακόβουλου περιεχομένου στο δίκτυο. Στην αρχιτεκτονική NDN οι Επιθέσεις Δηλητηρίασης Περιεχομένου (Content Poisoning Attacks) οδηγούν και σε Επιθέσεις Δηλητηρίασης Κρυφής Μνήμης, καθώς το κακόβουλο περιεχόμενο κατά την αποστολή του στους παραλήπτες θα αποθηκευθεί και από τις κρυφές μνήμες των ενδιάμεσων δρομολογητών. Τέτοιες επιθέσεις συχνά πραγματοποιούνται από ένα ζεύγος επιτιθέμενων που συνεργάζονται, με τον έναν να αναλαμβάνει το ρόλο του παραλήπτη και τον άλλο του αποστολέα. Ο παραλήπτης ζητά το κακόβουλο περιεχόμενο από τον αποστολέα.

Ως αποτέλεσμα, τα μετέπειτα αιτήματα έντιμων χρηστών για το ίδιο περιεχόμενο μπορεί να ικανοποιηθούν από το κακόβουλο περιεχόμενο που βρίσκεται αποθηκευμένο στις κρυφές μνήμες των δρομολογητών. Έτσι, ακόμη κι αν δε δημιουργηθούν ζητήματα ασφαλείας στους ίδιους τους τελικούς χρήστες, η επίθεση επιδρά στη δικτυακή απόδοση καθώς επηρεάζεται η ικανότητα αποθήκευσης ¨νόμιμου' περιεχομένου στις λανθάνουσες μνήμες των δρομολογητών. Μια παραλλαγή αυτής της επίθεσης χρησιμοποιούν ¨νόμιμο' αλλά μη-δημοφιλές περιεχόμενο ώστε να γεμίσουν τις μνήμες των δρομολογητών, επιτυγχάνοντας πρακτικά το ίδιο αποτέλεσμα. Οι επιθέσεις της τελευταίας μορφής ονομάζονται Επιθέσεις Μόλυνσης Λανθάνουσας Μνήμης (Cache Pollution Attacks).

Και στις δύο παραπάνω περιπτώσεις, οι επιτιθέμενοι αξιοποιούν για τους σκοπούς τους κάποια ιδιαίτερα χαρακτηριστικά του NDN όπως η ανάκτηση περιεχομένου με βάση το όνομά του και η δυνατότητα αποθήκευσης στην κρυφή μνήμη των δρομολογητών. Ωστόσο, είναι δύσκολο να αντιμετωπιστούν χρησιμοποιώντας αποκλειστικά το υπάρχον πλαίσιο ασφαλείας του NDN. Έτσι, η παρούσα διατριβή εξετάζει τη δυνατότητα επέκτασης του πλαισίου με μηχανισμούς εμπιστοσύνης βασισμένους στην αξιοπιστία (reputation).

Συνεισφορά:

Η διατριβή μας στοχεύει στην αντιμετώπιση των προαναφερθέντων ερευνητικών προβλημάτων που αφορούν την αρχιτεκτονική NDN. Όσον αφορα την ικανότητα ανοχής σε καθυστερήσεις και διακοπές, η παρούσα διατριβή προτείνει μια προσέγγιση δύο επιπέδων αξιοποιώντας πρωτόκολλα που έχουν προηγουμένως αναπτυχθεί για Δίκτυα Ανεκτικά σε Καθυστερήσεις και Διακοπές (Delay- and Disruption-Tolerant Networks / DTN). Συγκεκριμένα, προτείνουμε τη χρήση μιας στοίβας πρωτοκόλλων NDN over DTN και έναν συγκεκριμένο τρόπο ανάπτυξής της. Η συνολική μας λύση ονομάζεται NoD και αξιοποιεί την αρχιτεκτονική DTN (και συγκεκριμένα το Bundle Protocol) ως μέσο για τη μεταφορά των αντικειμένων περιεχομένου του NDN σε δίκτυα με διαλείπουσα συνδεσιμότητα. Το DTN παραμένει μία από αρκετές επιλογές μεταφοράς που μπορεί γενικότερα να χρησιμοποιήσει το NDN, με τη χρήση του να ενδείκνειται για τα συγκεκριμένα περιβάλλοντα. Με αυτό τον τρόπο διατηρείται η συμβατότητα με την αρχική αρχιτεκτονική NDN, ενώ η λύση που προτείνουμε μπορεί να συμπεριλάβει συσκευές που διαθέτουν διάφορα επίπεδα υπολογιστικών πόρων.

Δείχνουμε πως μια τέτοια λύση μπορεί να χρησιμοποιηθεί αποτελεσματικά για να υποστηρίξει αρκετές περιπτώσεις χρήσης σε σενάρια διαλείπουσας συνδεσιμότητας, όπως η μεταφορά και ανάπτυξη υπηρεσιών Mobile Edge Computing σε απομακρυσμένες περιοχές και η ανάκτηση περιεχομένου από απομακρυσμένους αισθητήρες στα πλαίσια του Διαδικτύου των Πραγμάτων. Τα πειράματά μας υποδεικνύουν ότι η λύση μας μπορεί να λειτουργήσει με ασύμμετρα μονοπάτια δρομολόγησης και σε συνθήκες ευκαιριακής συνδεσιμότητας, συμπεριλαμβανομένων σεναρίων με σύντομες δικτυακές επαφές. Επιπλέον, από πειραματικά μας αποτελέσματα αναδεικνύεται η βελτίωση της απόδοσης σε σχέση με τις υπάρχουσες μη-πληροφοριοκεντρικές λύσεις DTN καθώς και η δυνατότητα περαιτέρω βελτίωσης με την ανταλλαγή πληροφοριών μεταξύ των επιπέδων.

Στο δεύτερο μέρος της διατριβής μας, προτείνουμε τη χρήση μηχανισμών εμπιστοσύνης βασισμένων στη φήμη ως συμπληρωματικούς στο υπάρχον πλαίσιο εμπιστοσύνης

του NDN. Σε αυτά τα πλαίσια, επικεντρώνουμε σε δύο περιπτώσεις χρήσης: Τη ροή βίντεο με βάση το πρωτόκολλο DAS και τον διαμοιρασμό περιεχομένου σε δίκτυα Πεερ-το-Πεερ.

Στην πρώτη περίπτωση, προτείνουμε έναν μηχανισμό βασισμένο στη φήμη για την αντιμετώπιση επιθέσεων τύπου Bitrate Oscillation Attacks ο οποίος προσαρμόζει τις πολιτικές αποθήκευσης στην κρυφή μνήμη των δρομολογητών για να μετριάσει τις επιπτώσεις των επιθέσεων. Τα αποτελέσματά μας δείχνουν ότι ο μηχανισμός μας μπορεί να βελτιώσει αρκετά την Ποιότητα Εμπειρίας των χρηστών, φτάνοντας σε ορισμένες περιπτώσεις σε επίπεδα συγκρίσιμα με σενάρια στα οποία δε λαμβάνει χώρα επίθεση.

Στη δεύτερη περίπτωση προτείνουμε έναν μηχανισμό εμπιστοσύνης με βάση το blockchain, ο οποίος μπορεί να αναγνωρίσει την κακόβουλη συμπεριφορά και να ανταμείψει τους έντιμους χρήστες. Τα αποτελέσματά μας δείχνουν ότι ο μηχανισμός μας είναι ικανός να αναγνωρίσει ορθά τους επιτιθέμενους, παρέχοντας με συνέπεια χρηματικές ανταμοιβές στους έντιμους χρήστες. Δείχνουμε επίσης ότι μπορεί να δράσει τόσο σε περιπτώσεις επιθέσεων Full-on, όσο και όταν ο επιτιθέμενος υιοθετεί μια συμπεριφορά On-off με σκοπό να κάνει πιο αποτελεσματική την επίθεσή του.

Τα αποτελέσματα των πειραμάτων και για τις δύο περιπτώσεις χρήσης αποδεικνύουν ότι τα συστήματα αξιοπιστίας με βάση τη φήμη (reputation systems) μπορούν όντως να αποτελέσουν μια εναλλακτική μέθοδο επίτευξης εμπιστοσύνης στην αρχιτεκτονική NDN. Έτσι, αποδεικνύουμε οτι είναι δυνατή αλλά και ωφέλιμη η χρήση τους συμπληρωματικά στο υπάρχον πλαίσιο εμπιστοσύνης και ασφάλειας του NDN με σκοπό την ενίσχυσή του.

Acknowledgements

It seems that I finally arrived at the end of this PhD. Who would have thought! It has been a long road to get here, so I would like to dedicate this space to everyone who was - one way or another - a part of this journey.

First of all, I would like to thank my advisor, Professor Vassilis Tsaoussidis, for presenting me with this unique opportunity. He has supported me throughout this process and gave me a lot of freedom to explore ideas and directions on this interesting subject. I would also like to thank my advisory committee, Professors Pavlos Efraimidis and George Polyzos, for their valuable feedback, as well as all the members of my examining committee for the precious time they devoted to evaluate this dissertation.

I would also wish to thank the members of the Internetworked Systems laboratory (a.k.a. "Programming and Information Processing Lab" - long story), with which I had the pleasure of sharing the same space for several years. Without the collaboration and unique perspectives of Sotiris Diamantopoulos, Ioanna Kapetanidou and Vassilis Demiroglou, this dissertation would be very different. The same holds true for my collaboration with several other partners from various institutions during our shared participation in international research projects.

At this point, it would be hard not to mention all my friends and family. Even though nobody really understood what is the actual process for obtaining a PhD (in hindsight, most of them probably thought I was doing a very long Master's or something), their emotional support was invaluable. A special note is dedicated to my mother, Eleni, and my partner, Sissy, for patiently listening to my regular whining on various mishappenings. Their ability to withstand that for long periods of time is definitely a factor that contributed to the eventual completion of this thesis.

Contents

List of Figures

List of Tables

List of Abbreviations

ADU	**A**pplication **D**ata **U**nit
API	**A**pplication **P**rogramming **I**nterface
AP	**A**ccess **P**oint
AS	**A**utonomous **S**ystem
AVC	**A**dvanced **V**ideo **C**oding
BGP	**B**order **G**ateway **P**rotocol
BIBE	**B**undle-**I**n-**B**undle **E**ncapsulation
BOA	**B**itrate **O**scillation **A**ttack
BP	**B**undle **P**rotocol
CCN	**C**ontent **C**entric **N**etworking
CDF	**C**umulative **D**istribution **F**unction
CDN	**C**ontent **D**elivery **N**etwork
CGR	**C**ontact **G**raph **R**outing
CID	**C**ontent **ID**entifier
CLA	**C**onvergence **L**ayer **A**dapter
CoAP	**Co**nstrained **A**pplication **P**rotocol
CPA	**C**ontent **P**oisoning **A**ttacks
CPU	**C**entral **P**rocessing **U**nit
CR	**C**ontent **R**outer
CS	**C**ontent **S**tore
CSMA	**C**arrier-**S**ense **M**ultiple **A**ccess
DAG	**D**irected **A**cyclic **G**raph
DAS	**D**ynamic **A**daptive **S**treaming
DASH	**D**ynamic **A**daptive **S**treaming over **H**ttp
DHT	**D**istributed **H**ash **T**ables
DLT	**D**istributed **L**edger **T**echnology
DNS	**D**omain **N**ame **S**ystem
DONA	**D**ata **O**riented **N**etwork **A**rchitecture
DoS	**D**enial-**o**f-**S**ervice attack
DDoS	**D**istributed **D**enial-**o**f-**S**ervice attack
DTN	**D**elay-/**D**isruption-**T**olerant **N**etworking
ECDSA	**E**lliptic **C**urve **D**igital **S**ignature **A**lgorithm
EID	**E**ndpoint **ID**entifier
FIB	**F**orwarding **I**nformation **B**ase
FIFO	**F**irst **I**n **F**irst **O**ut
FQDN	**F**ully **Q**ualified **D**omain **N**ame
Gbps	**G**iga**b**it **p**er **s**econd
GCR	**G**ained **C**oins **R**atio
Ghz	**G**iga**h**er**z**
GNRS	**G**lobal **N**ame **R**esolution **S**ervice
GPS	**G**lobal **P**ositioning **S**ystem
GUID	**G**lobally **U**nique **ID**entifier

hICN	**h**ybrid **ICN**
HS	Wi-Fi **H**ot**S**pot
HTTP	**H**ypertext **T**ransfer **P**rotocol
i3P	**I**nternet **I**ndirection **I**nfrastructure
ICDTN	**I**nformation **C**entric **D**elay **T**olerant **N**etworks
ICN	**I**nformation **C**entric **N**etworking
ID	**ID**entifier
IETF	**I**nternet **E**ngineering **T**ask **F**orce
IFA	**I**nterest **F**looding **A**ttack
IoT	**I**nternet **o**f **T**hings
IP(v4/v6)	**I**nternet **P**rotocol (version 4/version 6)
IPC	**I**nter-**P**rocess **C**ommunication
IPFS	**I**nter**P**lanetary **F**ile **S**ystem
IPND	**IP** **N**eighbor **D**iscovery
IPNL	**IP** **N**ext **L**ayer
IRTF	**I**nternet **R**esearch **T**ask **F**orce
ISP	**I**nternet **S**ervice **P**rovider
IPC	**I**nter-**P**rocess **C**ommunication
L3	**L**ayer **3**
LFU	**L**east **F**requently **U**sed
LRU	**L**east **R**ecently **U**sed
LTE	**L**ong **T**erm **E**volution
MANETs	**M**obile **A**d-hoc **NET**works
MB	**M**ega**B**ytes
MPD	**M**edia **P**resentation **D**escription
NB-IoT	**N**arrow**B**and **I**nternet **o**f **T**hings
NDN	**N**amed **D**ata **N**etworking
ndnSIM	**n**amed **d**ata **n**etworking **SIM**ulator
NFD	**N**amed Data Networking **F**orwarding **D**aemon
NLSR	**N**amed-data **L**ink **S**tate **R**outing protocol
NoD	**N**DN **o**ver **D**TN
ns3-DCE	**n**etwork **s**imulator **3** - **D**irect **C**ode **E**xecution
OLSR	**O**ptimized **L**ink **S**tate **R**outing protocol
ONE	**O**pportunistic **N**work **E**nvironment simulator
OppNets	**Opp**ortunistic **Net**works
P2P	**P**eer to **P**eer
PDU	**P**rotocol **D**ata **U**nit
PIT	**P**ending **I**nterest **T**able
PKI	**P**ublic **K**ey **I**nfrastructure
PSN	**P**ocket **S**witched **N**etworks
QoE	**Q**uality **o**f **E**xperience
QoS	**Q**uality **o**f **S**ervice
PARC	**P**alo **A**lto **R**esearch **C**enter
PoP	**P**roof **o**f **P**restige
PoS	**P**roof **o**f **S**take
PoW	**P**roof **o**f **W**ork
PSIRP	**P**ublish-**S**ubscribe **I**nternet **R**outing **P**aradigm
PSN	**P**ocket **S**witched **N**etworks
Pub/Sub	**P**ublish-**S**ubscribe
QoE	**Q**uality **o**f **E**xperience
RAM	**R**andon **A**ccess **M**emory

RENE	**RE**ndezvous **NE**twork
RFC	**R**equest **F**or **C**omments
RH	**R**esolution **H**andlers
RIFE	architectu**R**e for an **I**nternet **F**or **E**verybody
RIB	**R**outing **I**nformation **B**ase
Rid	**R**endezvous **id**entifier
RINA	**R**ecursive **I**nter**N**etwork **A**rchitecture
Rivest	**S**hamir **A**dleman cryptosystem
RTT	**R**ound-**T**rip **T**ime
ROFL	**R**outing **O**n **F**lat **L**abels
SDN	**S**oftware **D**efined **N**etworking
SDSI/SPKI	**S**imple **D**istributed **S**ecurity **I**nfrastructure/**S**imple **P**ublic **K**ey **I**nfrastructure
SHA	**S**ecure **H**ash **A**lgorithm
Sid	**S**cope **id**entifier
SRAM	**S**tatic **R**andom **A**ccess **M**emory
SSP	**S**cheme **S**pecific **P**art
TCP	**T**ransport **C**ontrol **P**rotocol
TTL	**T**ime **T**o **L**ive
UaV	**U**unmanned **A**erial **V**ehicle
UCLA	**U**niversity of **C**alifornia, **L**os **A**ngeles
UDP	**U**ser **D**atagram **P**rotocol
UMOBILE	**U**niversal **MOBILE**-centric Opportunistic Communications Architecture
URI	**U**niform **R**esource **I**dentifier
URl	**U**niform **R**esource **L**ocator
V1/V2	**V**ersion **1** / **V**ersion **2**
XIA	e**X**pressive **I**nternet **A**rchitecture
XML	e**X**tensible **M**arkup **L**anguage
WSN	e**W**ireless **S**ensor **N**etwork

Dedicated to my mother, who
raised me
all by herself…

Chapter 1

Introduction

1.1 Context

The Internet has undergone several changes during its existence. One of the most significant is its main use: from an original network of networks designed to interconnect fixed hosts, it is now used as a means for content distribution to its users. The rise of video distribution is a prominent example: according to the Cisco Visual Networking Index [1], video traffic in 2022 was expected to account for 82% of all Internet traffic. However, this is a relatively recent development and the underlying protocol which is supporting all this traffic, the Internet Protocol (IP), was designed for an entirely different purpose. This has led to various mechanisms being developed on top, in order to achieve efficient content delivery (e.g. Content Delivery Networks, Peer-to-Peer networking platforms etc.). As a result, additional, unnecessary, complexity is added to the architecture, introducing significant overhead.

This realization resulted in the development of Information-Centric Networking (ICN [2]), a novel approach to networking which aims to solve the inherent problems of the IP-based infrastructure. In ICN, the focus is placed on "what" is being transferred, instead of "where", effectively making the Internet a native content distribution network. In this regard, it departs from the traditional IP design where access to the desired content is mapped to its location and embraces a content-centric model where access to content is mapped to the content itself irrespective of the location where the content is held. The focus of the network layer is no longer host addresses, but content objects, aligning the protocol design to this current major Internet use.

Several architectures have been proposed to realize the ICN paradigm. The most prominent one is the Named Data Networking (NDN) architecture [3]. NDN was originally a project based on the Content Centric Networking (CCN) [4] architecture, but the two implementations subsequently diverged. NDN makes named content objects the prime focus of the network layer, a design which stands in contrast to IP which was designed to interconnected fixed hosts.

Some of NDN's prominent features include receiver-initiated communication, hierarchical content names, in-network caching and data-centric security. Its forwarding and routing mechanisms operate according to content identifiers, instead of location identifiers.

Even though NDN has matured as an architecture during the past decade, there still exist open issues which hinder its adoption and have to be tackled. In this dissertation, we target two of those issues and propose relevant solutions. The first one is the architectural support of delay/disruption tolerance, an increasingly relevant feature in an era characterized by the spread of computing and networking in previously-unforeseen environments. Even though the ubiquitous nature of current

computing devices implicitly calls for intermittent connectivity support, the ability to network intermittently-connected devices is not a native feature provided by NDN. This observation provided the motivation for the first part of this dissertation.

The second challenge faced by NDN is the design of efficient trust and security mechanisms. Even though security aspects have been considered during the architecture's design based on the lessons learned from TCP/IP, new issues have arisen and need to be addressed before large-scale deployment. Many of those stem from the fact that NDN's design introduces new components and novel features, which create new security threats and increase its attack surface. This aspect motivated the creation of the relevant mechanisms which are presented in the second part of this dissertation.

Our work aims to tackle the aforementioned issues by delving into and leverage the existing work on two main areas. The first one is Delay-/Disruption-Tolerant Networking (DTN) [5], which was originally proposed as a protocol architecture for the Interplanetary Internet[1]. This need arose as as the constraints posed by space networks, such as frequent communication disruptions and large delays, could not be addressed by legacy networking protocols. Shortly afterwards, the architecture was generalized to also cater for other similar environments, such as challenged terrestrial networks.

The second one is reputation systems, which can provide Internet-wide trust by relying on user ratings. Even though they have been been used by the industry for the past two decades, mainly in the fields of e-commerce and peer-to-peer file sharing networks, they have been largely overlooked by the Named Data Networking research community.

In this dissertation, we posit that effective solutions to the previous challenges faced by NDN can be formed based on the scientific and technological advancements of the former fields. To this end, we explore the use of the DTN architecture to provide delay/disruption tolerance in NDN, as well as the employment of reputation systems to consolidate NDN's trust and security mechanisms.

1.2 Problem Definition

The first problem we tackle in this dissertation is *the introduction of delay and disruption tolerance into the Named Data Networking architecture*. Such a feature is highly desirable, as several circumstances may require disruption-tolerant network operation.

For instance, network disruptions are hard to avoid in remote locations such as forests and mountains, in which sensors and tiny IoT devices are regularly deployed nowadays to support applications such as environmental monitoring. Moreover, in the event of natural disasters, the network might suffer from both fragmentation and extreme congestion which leads to the impairment of typical, continuous, network connectivity (even though the ability to communicate can prove more crucial than before). Last, populations living in underpopulated and/or underdeveloped areas can also suffer from similar issues, as it not cost-effective for Internet Service Providers to install the infrastructure required to ensure Internet access.

[1]The term 'DTN' can refer to both delay/disruption-tolerant networking as a research area and a specific architecture which was originally designed to operate in space networks. In this work, we are going to use the term 'DTN' to refer to the architecture and 'DTNs' to refer to the networks which share those characteristics.

In such cases, it is highly difficult or impossible to establish continuous connectivity. However, NDN is by-default unable to operable in such challenged network environments due to the fact that a well-connected environment was assumed while its core features were designed.

The fact that the NDN architecture was tailored for fixed networks with seamless connectivity is mostly evident in its use of symmetric paths to route content requests and responses, also known as "breadcrumbs routing". To retrieve a content object, a content request is issued by the Consumer in the form of an Interest packet. The Interest is forwarded by the network to the closest content instance, using a structure called a Forwarding Information Base (FIB). Once the content is found, it is then sent back encapsulated in Data packets, which follow the reverse path to the requester. This approach is called "breadcrumbs" routing and uses the information available in the Pending Interest Table (PIT) of each router.

For this approach to work, a static network topology is needed; in the case of a significant topology change, the reverse route back to the consumer will not be available for Data packets to follow, which can cause delivery failure. This situation, however, is the norm in terrestrial DTNs (such as mobile opportunistic networks) which are comprised by nodes with frequent connectivity changes and elaborate mobility patterns. As a result, highly dynamic topologies are created which cannot be sufficiently accommodated by the original NDN architecture.

This is a difficult problem to address while maintaining interoperability with the original implementation that was designed to be employed in fixed networks. This dissertation proposes a solution to achieve this, using a layered "NDN over DTN" protocol stack and an accompanying deployment strategy.

The second problem tackled in this work is *the enhancement of NDN's trust mechanisms with reputation-based approaches, to address those security vulnerabilities which are difficult to address by the sole use of credential-based trust*.

By making named data the central piece of its architecture, NDN secures the data itself, independent of the underlying communication channels. This stands in contrast to the current approach employed by TCP/IP, which mainly focuses on securing the communication channel itself. NDN ensures the integrity and authenticity of all the data transferred by the network. Confidentiality can also be optionally offered. This is achieved by requiring each Data packet to carry a cryptographic signature, generated at the time of data creation using its Producer's key. This approach effectively binds the packet's name to its content and enables NDN to decouple the trustworthiness of a data object from the specifics of how and where it was obtained from.

Even though this design can avoid some of TCP/IP's inherent problems, the introduction of new features also increases the attack surface that can be exploited by malicious users. One such feature is in-network caching. In NDN, each router has a cache so that future requests can be answered by intermediate nodes. This negates the need for content requests to be constantly routed towards the content source, resulting in performance gains such as decreased latency and load balancing potential. However, the same feature can render the architecture vulnerable to attacks targeting the router caches.

In this dissertation, we identify two types of attacks which can be mitigated using reputation-based trust mechanisms and design two systems which can mitigate those threats. One such attack, which is made possible in NDN, is Bitrate Oscillation Attacks (BOAs). Those attacks exploit the behavior of the Dynamic Adaptive Streaming (DAS) protocol when deployed on top of an NDN network layer, with its

in-network caching capabilities. DAS is designed to automatically adjust the quality of the video segments being sent according to the available network bandwidth.

In BOAs, an attacker proactively requests non-consecutive video segments, so that some of the segments will be stored in the cache when a DAS client requests the video in the future. As a fraction of the segments will be received from the router cache with decreased latency and another fraction from the original source with an increased one, the bandwidth estimation will not be accurate and the video bitrate will oscillate. This results in decreased Quality of Experience (QoE) for the end users.

The second attack which is considered in this dissertation, includes Content and Cache Poisoning Attacks (CPAs). Although similar attacks were feasible in earlier, overlay content-based networks (e.g. peer-to-peer file sharing), they were not native to the TCP/IP network architecture and only affected the overlay networks. In CPAs, the attackers aim to spread fake and/or malicious content in the network. Content Poisoning Attacks also result in Cache Pollution Attacks in NDN, as the malicious content will be stored in network caches on its way to the requesting users.

The attack is frequently performed by a pair of colluding malicious users, one assuming the role of an NDN Consumer (receiver) and the other one of an NDN Producer (sender). The Consumer requests malicious content from the Producer and the responses populate the router caches on their way to the Consumer. Even though the signatures may be corrupt, the Data packets can still be stored in the caches as signature verification is a cryptographically heavy operation and is not mandatory for routers.

As a result, legitimate users subsequently requesting that content will receive those content objects instead of the legitimate ones. In this case, even though this content is never received by the honest network users, it still impacts network performance as the caches cannot store the most popular content to serve subsequent requests and the use of in-network caching is effectively negated. From an attacker's perspective, there is also the option to flood the routers with legitimate but less popular content. Those attacks are called "Cache Pollution Attacks" and effectively achieve the same effect.

Both of those threats exploit features unique to NDN such as name-based content retrieval and in-network caching and are either difficult or impossible to overcome by solely using the existing NDN credential-based security. Therefore, this dissertation explores the use of reputation-based trust mechanisms against them.

1.3 Contributions

The previously-described problems have been addressed in various ways in the present doctoral thesis. In this section, our contributions are highlighted.

With respect to the introduction of delay and disruption tolerance in the NDN architecture, this dissertation introduces NDN over DTN (NoD). NoD is a protocol stack and deployment scheme which uses the DTN architecture as an underlay to NDN in order to transfer packets to/from intermittently-connected devices using mobile data mules.

Our design leverages prior work on the DTN architecture along with the concept of 'faces' in NDN, to create a delay-tolerant communication channel in the form of a DTN tunnel. One of the advantages of this design is that it preserves compatibility with the original NDN architecture, as it is merely an extension of the architecture and does not modify any of its pre-existing features.

We validated this approach in a real-world deployment assuming a disaster scenario which uses edge computing to deploy emergency services in a remote area. During the validation process, the NDN/DTN stack was deployed in edge nodes and data mules were used to transfer and instantiate services in the intermittently-connected devices.

Furthermore, we deployed NoD in a different setting which included intermittently-connected IoT devices. We conducted an extensive experimental campaign which resulted in a detailed performance evaluation. By taking into account the results of our first experiments, we designed an improved version of our mechanism which utilizes cross-layer information to reduce network overhead.

In addition to the experiments, we created a mathematical model which represents its behavior. Our system model is tested against the experimental results and its validity is verified. Apart from its accurate reflection of our system's properties, our model is - to the best of our knowledge - also the first attempt to integrate information-centric and delay-tolerant metrics in a joint mathematical framework.

As an extension of the aforementioned work regarding NoD, we also performed some additional simulation experiments to assess the way in which the selection of large packet sizes in the edge computing scenario can impact fixed-network NDN deployments.

With respect to the enhancement of the NDN trust and security fabric, we propose reputation-based trust as a complementary approach to achieve trust establishment and attack mitigation.

To illustrate the feasibility and effectiveness of this approach, we identify two relevant cases in which NDN security can be supported by reputation mechanisms. Those cases include a) Bitrate Oscillation Attacks and b) Content Poisoning Attacks.

For each one of those cases, we proceed to design a different attack mitigation mechanism. In the BOA case, the DAS protocol operation over NDN is simulated. In this design, routers assigns reputation values to the disseminated content to enable the detection of compromised namespaces and adjust their caching policies to avoid caching potentially malevolent content.

In the CPA case, a peer-to-peer file sharing network is simulated. In this design, users assign ratings to the peer providing each content object. A blockchain-based incentive system is used to securely store the ratings and, at the same time, reward legitimate users by financially incentivising honest behavior. This way, attacker identification is achieved and the spread of malevolent content can be inhibited. Moreover, financial counter-incentives are utilized as a pro-active mitigation measure.

The common denominator between the two approaches is the fact that they are both based on reputation systems. Subsequently, we evaluate the performance of those systems in simulation environments. The results from those experiments evidently prove our initial position that reputation systems can provide an effective alternative means to consolidate the NDN trust and security plane.

1.4 Structure

The structure of this dissertation is the following:

- Chapter 1 introduces the topic and presents an overview of the research problem and our contributions.

- Chapter 2 presents the background of our work and provides a detailed analysis of the related work in each respective field.
- Chapter 3 is concerned with our solution towards delay/disruption-tolerant NDN. Specifically, we elaborate on the design of NoD and its context.
- Chapter 4 is concerned with reputation-based attack mitigation. We present the design of our reputation mechanisms for Bitrate Oscillation Attack and Content Poisoning Attack mitigation.
- Chapter 5 presents our evaluation methodology, including details regarding the experiments conducted.
- Chapter 6 includes the experimental results.
- Chapter 7 summarizes our conclusions.

Chapter 2

Background and Related Work

2.1 Information-Centric Networking

2.1.1 Overview

The Internet, as we know it today, was once a very different entity. Its beginning can be traced back to the 1970s when it started as an academic "network-of-networks". At the time of its design, the problem it tried to solve was forwarding packets of data between stationary hosts which existed in different networks. During the subsequent decades, it experienced massive growth, mainly due to its "hourglass" design: Its centerpiece, the "network layer" - used to interconnect different networks - was transparent enough so that any service could run on top of it but, at the same time, it was simple enough so that it could run on top of any lower-level protocol.

However, as its user base, geographic coverage and overall adoption grew, new needs arose which were previously unforeseen. Mobile computing devices were non-existent at the time of the Internet design, while security and trust were not an issue in a setting which included specific devices of known entities and organizations. Thus, "patches" started to accumulate on top of IP to cover those needs.

Those patches, in turn, increased the overall architecture's complexity which can manifest in its performance. For instance, a larger protocol stack suggests an increased communication overhead (e.g., more bytes being transmitted in the physical layer, as more packet headers are added by higher-layer protocols and overlay networks used for information retrieval). While this complexity can also produce performance constraints in today's era of low-power Internet-of-Things devices, in which computing power and energy can be a scarcity and a large networking stack can be seen as resource-consuming and inhibiting deployment.

For those reasons, clean-slate architectural approaches were proposed to discontinue the addition of "patches" and Future Internet emerged as a distinct research area in the mid-00s. Different architectural approaches made differing assumptions about how the future of the Internet would look like, as well as on what the biggest limitations of the current IP architecture are.

Sometimes the different designs would take the Internet architecture in opposing directions. Nevertheless, many identified a particular feature of the IP architecture as a common source of many illnesses: the fact that an IP address served both as a) an end-point identifier, providing the host's identity b) an end-point locator, providing the host's location. Even though this may have seemed a natural choice in the early years of the Internet when fixed hosts were the only internetworking option, mobile computing devices became a reality since the turn of the millennium. Thus, several designs emerged which tried to achieve an identity/location split, aiming to efficiently support mobility, multihoming and several other features.

Out of those designs, emerged a particular line of research which pursued to change the naming and routing fabric of today's Internet. This line of thought challenged the efficiency of the existing Domain Name System (DNS), which resolves names into addresses, and questioned the current practice of taking for granted that Internet routing operations had to be based on IP addresses.

At the same time, several researchers noticed that since the Internet's expansion beyond the academic world, users were mostly interested in retrieving content rather than connecting to specific hosts. They argued that - over time - content retrieval became the primary "use case" of the Internet and the current primary Internet usage, i.e. content distribution, does not align well with the underlying communication model. The TCP/IP architecture is, by design, host-centric and its network-layer mechanisms cannot be used to retrieve information from the best possible location. This later line of research borrowed concepts and mechanisms from the former and, eventually, the research area of Information-Centric Networking (ICN) was formed.

In a nutshell, the process which led to ICN as a distinct field began with the aforementioned realization that IP addresses denote both the location and identity of a host, resulting in several designs which included location-identity separation. This idea was then taken one step further by proposals which advocated to abandon the use of location identifiers altogether and instead start routing directly on identities (e.g. FQDNs[1]), which resulted in name-based routing. The next logical step was the transition from using host identifiers to using content identifiers for name-based routing, hence the emergence of content-based routing mechanisms and proposals for an infrastructure to support it on top of IP. The last development was the abandonment of IP-based routing altogether, which led to clean-slate ICN designs. Those 'native' ICN architectures favoured network-layer content routing, as opposed to the earlier overlay architectures. This architectural evolution is presented in the rest of this Section.

Other Future Internet architectural approaches exist as well, some which include the XIA [6] architecture, which was designed to address extensibility and security issues, RINA [7], which adopts an IPC-based networking model to address security, mobility, manageability and scalability, the NEBULA architecture [8] which was tailored for cloud computing and its increasing penetration, as well as several others.

ICN, as a research field, is comprised of several architectures which are united by a common principle: to make information retrieval the centerpiece of the Internet architecture. Essentially, ICN architectures try to solve the inner conflict which exists in today's Internet architecture between historical design (host-centric) and current usage (data-oriented), by redesigning the former using design principles based on the later. Most of those include clean-slate designs of the Internet architecture and are built around the concept of networking content, instead of hosts. Consequently, they frequently disregard IP addresses at the networking layer and place a particular focus on developing efficient name-based mechanisms. The later permeates the architectures on all levels, but can be seen particularly on L3, in which name-based content routing is typically introduced.

Although traces of a data-oriented approach can be found since the early days of the Web (sometimes attributed to Ted Nelson and the Xanadu project [9] [10]), the first proposal for a content-centric Internet (and main precursor to subsequent ICN architectures) was perhaps the TRIAD project at Stanford (1999-2002).

[1] https://en.wikipedia.org/wiki/Fully_qualified_domain_name

The paper presenting the TRIAD architecture [11] was the first to view the problem of a user efficiently accessing content/information in today's Internet as a routing problem per se, formulating it as the "content routing" problem. It was also the first to propose a content-centric Internet architecture, in the form of an Internet "content layer", to solve it. The authors formulate the content routing problem, as follows: *"Clients (and users) desire connectivity not to a particular server or IP address but to some piece of content, specified by name (typically a URL)."*

Replicated servers were, thus, viewed in the light of offering alternate routes to that content. The goal of content routing is to reduce the time needed to access content, which is accomplished by directing a client to one of many possible content servers.

Solving this problem was the main motivation for creating all subsequent ICN architectures, as well as the main driver for their core design principles. Therefore, it is not surprising that several of the features appearing in later architectural designs, or early versions of them, can be found in TRIAD.

Some examples include the proposal of "content routers", which manage information regarding content reachability, as well as the creation of a name-based routing protocol (NBRP), the explicit aggregation of name suffixes that map to the same routing information into routing aggregates (to achieve routing scalability), and others.

The TRIAD architecture, however, did not assume a clean-slate design but, rather, an incremental deployment. Content routers were designed to support both IP and name-based routing. Furthermore, the authors did not envision that every Internet router needed to be a content router; instead, they proposed that firewalls, gateways and BGP-level routers were augmented, while the vast majority of routers remained oblivious to the content layer.

On their approach, the domain name of the desired web site or content volume is handled by a specialized ISP name server. When the client initiates a name lookup on the DNS portion of the content URL, the request goes to this specialized name server which then returns the address of a server "near" the client, an approach which could fail over to DNS behavior for unrecognized names.

Roughly concurrent to the TRIAD project, were also some other works which explored Internet-wide communication focused on content retrieval. For instance, Carzaniga and Wolf were the first to introduce the term "content-based networking" in their 2001 paper [12]. In this work, they present a general communication model for content-based networks, focusing on the design options to implement functionality such as content-based forwarding, routing, subnetting and supernetting etc.

As was also the case with TRIAD, they did not adopt a clean-slate approach and explicitly state that they *"do not propose content-based networking as a replacement for IP, nor advocate an implementation of a publish/subscribe middleware at the network level (i.e., within routers)."* Rather, their approach is intended to implement the specific communication style typified by publish/subscribe middleware services as application-layer networks, *"most likely on top of existing Internet protocols"*.

In later works [13], they propose that a content-based communication service is provided as a datagram, connectionless service through an overlay point-to-point network. Important to understanding their system design, is the fact that they see their proposal for content-based networking as an evolution of their previous work on distributed publish/subscribe event notification systems [14].

A different architecture which adopted a publish/subscribe-style communication paradigm was the *Internet Indirection Infrastructure (i3)*, proposed in 2002 [15]. Again, this approach aimed at incremental deployment and presented itself in the

form of an overlay network. Its importance lies in the fact that it first introduced the concept of rendezvous and an Internet-wide pub/sub communication infrastructure for content retrieval.

i3 created a general-purpose communication abstraction in the form of an overlay network, to support multicast, anycast and mobility. Its main purpose was to decouple the act of sending from the act of receiving (i.e. "to provide indirection"), by adopting a rendezvous-based communication model. In this model, sources send packets to a logical identifier and receivers express interest in packets sent to an identifier.

Packets are pairs (*id, data*), where *id* is an identifier and *data* is the payload. Triggers ((*id, R*) are inserted into the network by a receiver *R* (where *R* denotes the receiver's address) to receive packets sent to *id*. The identifier, thus, represents a logical rendezvous between the sender's packets and the receiver's trigger. This way, senders need not be aware of the number or location of the receivers, or vice-versa.

The matching between packet and trigger identifiers is performed the following way: At any given time, there is a single i3 node responsible for a given *id*. When a packet is sent to *id*, it is routed by i3 to the node responsible for that *id*.

This communication paradigm is similar to the publish-subscribe paradigm. The design was meant to be incrementally deployable and compatible with legacy applications, while its proof-of-concept implementation was based on the Chord DHT [16].

Another early architecture which attempted to route directly on identifiers themselves, getting rid of location information in the network layer and packet header, was ROFL [17]. The authors acknowledge the fact that the current Internet architecture conflates network locations and host identities and describe previous attempts at clean-slate architectures which split identity from location. ROFL takes a different approach and gets rid of location altogether. Even though they acknowledge previous works which essentially achieved the same goal by routing directly on FQDNs, such as TRIAD [11] and IPNL [18], ROFL's contribution was that it was the first such design to use flat, self-certifying names.

They choose this approach for two main reasons: First, the authors claim that flat names are the easiest way to build persistent identifiers. Their argument is that persistence can only be achieved if names are free from any mutable semantics, which can be ensured by actually giving the name no semantics at all. This is achieved in ROFL by using a flat namespace in which names have no semantic content, but can have a cryptographic one. Second, when using self-certifying identifiers, a node's identity is tied to a public-private key pair and its identifier is a hash of its public key. Thus, security can be enhanced.

It is important to note that in ROFL, flat names do not yet refer to content objects, but to network node identities (towards which, paths are created). However, its introduction of flat, self-certifying names and concept of routing on labels can be considered an important point in the evolution process which led to features later included in ICN architectures.

2.1.2 Main architectures

Those previous advancements finally led to the first clean-slate content-centric architecture – and for this reason, sometimes considered the first ICN architecture [2] - the Data Oriented Network Architecture (DONA). DONA was designed by UC Berkeley and presented in a 2007 paper [19]. The authors reference TRIAD as an

inspiration, combined with concepts brought in from previous work on flat, self-certifying names.

DONA, although adopting a clean-slate approach, does not propose the replacement of IP at the network layer, as do later architectures. Instead, it introduces some form of a content-centric layer which exists between the network and transport layers. Its basic design does not require modifications of the IP infrastructure and, thus, could function over the current IP layer with its present form of addressing and routing. The radical change it brings lies is its clean-slate redesign of both Internet naming and name resolution, designed to entirely replace today's DNS system.

DONA proposes the replacement of DNS names with flat, self-certifying names and DNS name resolution with a name-based anycast primitive. As name resolution is identified by the authors as one of the biggest challenges in such a design, they dedicate a large part of this work on solving it.

To achieve this, they propose the replacement of DNS servers by new network entities which are called Resolution Handlers (RHs). The RHs operate using two basic primitives: FIND and REGISTER. FIND messages are issued by a client to locate an object and are routed by RHs to the nearest copy, while REGISTER messages set up the state necessary for the RHs to route FINDs effectively. In this architecture, the authors consider the client as the source of the FIND and the server as the node that responds to the FIND.

Reverse-path routing is followed by the response, in the form of providing the next-hop AS. As path discovery is performed above the IP layer, there is some added overhead (in the form of endpoints being required to detect AS-level path failures and resend a FIND message).

Possible extensions are also discussed, in the form of added functionality to the RHs. The goal is that they do not only forward FINDs and REGISTER, but also take a more active role in the architecture. Such extensions include the possibility of caching in RHs, or the enablement of subscriptions (which could be achieved in the form of TTL'ed FINDs, combined with the already-existent reverse-path routing functionality). Indeed, DONA's application interface can be considered as semantically similar to publish/subscribe systems, which decouple the application endpoints in space and time.

Another pioneering ICN architecture is the *Content Centric Networking (CCN)* architecture from PARC. The origins of CCN can be traced back to a 2006 Google tech talk from Van Jacobson, while the architecture was fully presented three years later [20].

CCN was the first architecture to go as far as to propose a replacement for the IP protocol at the network layer, with a stated goal of making *named data* the thin waist of the Internet protocol stack. In CCN, names are hierarchical to achieve scalability by allowing name resolution and content routing information aggregation. This approach contrasts with the flat names used in DONA and is more similar to the one used in TRIAD. However, other DONA parallels can be found in CCN, such as Interest and Data packets (FIND/REGISTER), Content Routers (RHs), in-network caching etc.

As the CCN architecture was later developed further by UCLA and evolved into the *Named Data Networking (NDN)* architecture [21], which is the main subject of this dissertation, we reserve a complete presentation of its features for Subsection 2.1.3.

A different clean-slate architecture, designed to completely replace the IP protocol stack, was proposed by the PSIRP project (2008-2010) and developed further by PURSUIT (2010-2013).

PSIRP/PURSUIT explicitly adopts a Publish/Subscribe paradigm, which is seen as a viable alternative to the send/receive communication paradigm. For this reason, it redesigns the Internet architecture from this point-of-view and uses publish/subscribe primitives in all its levels, including core networking functions.

As a publish/subscribe architecture, PSIRP/PURSUIT is composed of three main elements: publishers, subscribers and a network of brokers. Publishers are either owners or disseminators of information, which advertise information availability by issuing publication changes. Subscribers are information consumers and express their interest for specific information objects by issuing related subscriptions. Between those two, there lies a network of brokers which routes publication and subscription messages, matches the publication with subscriptions and is responsible for the information forwarding process.

A unique feature of PSIRP/PURSUIT, is the fact that it includes a layer-less networking stack. It is based on a modular, extensible core - the "component wheel" and views the global network of information as an acyclic graph of related pieces of data. The protocol organization is achieved via the efficient structuring of information, identifiers and their interactions among network elements.

For its operation, the architecture utilizes three distinct functions: rendezvous, topology management and forwarding. The relationship between them is the following: A subscription is matched to a publication by the rendezvous function, which subsequently directs the topology management function to create a route between the publisher and the subscriber. The forwarding function uses the aforementioned route to perform the actual transfer of data.

The broker in which a publication matches a subscription is called the "Rendezvous Point" of a publication. The existence of those points is claimed by the authors to introduce several advantageous features, allowing publisher-subscriber decoupling, location-identity split, multihoming, mobility and anonymity. Those Rendezvous Nodes collectively create a Rendezvous Network (RENE), which offers the name resolution functionality and is implemented as a hierarchical DHT.

Identifiers in PSIRP define the relationships between the pieces of information in the system on the different levels and are grouped on 4 distinct classes: Application identifiers, Rendezvous Identifiers, Scope Identifiers and Forwarding Identifiers.

Each item is uniquely identified by a pair of identifiers: the Rendzvous Identifier and the Scope Identifier. The Rendezvous ID is information-specific and represents the network-level identity of a publication. It has to be unique within a scope. The Scope ID denotes the scope in which the information item belongs. Both are flat and location-independent; however, while PURSUIT names are flat, scopes can be organized in hierarchical scope graphs. As a result, a complete name consists of a sequence of scope IDs and a single rendezvous ID, effectively generalizing the DONA naming scheme.

Data exchange is realized using publications and subscription operations, which are decoupled. The publication process is the following: First, a publisher has to know the Sid of a scope within which it wants the publication to be published. Furthermore, it has to know the Rid for the publication. Subsequently the publication's Rid is forwarded to the rendezvous node of the Sid rendezvous network, which manages the publication's Rid. The publication message which is created, carries both the publication's Rid as well as the scope's Sid.

Subscriptions follow a similar process: First, a subscriber learns the Rid and Sid of a desired information item and issues a subscription message towards the appropriate rendezvous point. When its subscription message reaches this point, a forwarding path is created from the publisher towards the subscriber and the desired

piece of information is sent through this path. Each active publication is assigned an Fid that denotes the forwarding path this item has to follow.

Another important architecture was the Cache-and-Forward network architecture [22], which later evolved into the MobilityFirst [23] architecture. The main motivation for the design was the efficient support of mobility in the Future Internet, therefore its main features were tailored to serve mobile environments (e.g. the explicit support for delay/disruption tolerant routing, assuming a continuum of fixed and intermittently-connected hosts).

It is built on the premise that each network entity - regardless of whether it is a content object, a networking device or a service - has a both a globally unique name and a network address. The two are separated in the architecture and names can be dynamically translated into addresses by the network in various points. This design favours mobility, as redirection is facilitated and can be efficiently supported. Some high-level features of MobilityFirst include a) late binding between names and addresses, b) on- and off-path caching and c) the decouplement of name resolution and content routing, with the request and response messages being routed individually.

The content-centric nature of communications is realized using two main components: The aforementioned Globally Unique Identifiers (GUID) and a Global Name Resolution Service (GNRS). Content Routers (CR) are also used for routing purposes. However, the entire name resolution and content routing operation is not entirely name-based but also relies on the existing IP routing techniques. A GUID is mapped to GNRS server addresses using hashing and CR routers essentially use regular IP routing, only communicating with the GNRS to obtain the mapping between the destination GUID and a network address. Therefore, the infrastructure for name-based routing, GNRS, is in practice only used to map GUIDs to network addresses. Moreover, its in certain communication instances is optional, as once a GUID has been translated to an address, the architecture can disregard GNRS and operate entirely based on network addresses. Nevertheless, the GNRS has to be consulted to obtain the latest mappings - an important aspect for mobile environment support.

4WARD, which later evolved into SAIL, was also an important ICN project and architecture. Those projects were based on the concept of Network of Information (NetInf) [24], a specific ICN approach which aiming to connect different technology and administrative domains on a global scale using a hybrid name-based routing and name resolution scheme.

The more mature of the two architectures, SAIL, is a general architecture which uses the concept of convergence layers to support different routing and forwarding mechanisms. One of its definitive features is its flexibility. Content names are defined by a URI scheme which consists of two parts: an authority part A and a local part L. Both exact and longest-prefix matching is allowed on the names, depending on the operation and, as a result, SAIL names can behave both like flat and hierarchical names in different circumstances. Name resolution can be both coupled and decoupled with data routing, allowing even hybrid operation. When decoupled, a form of DHT is used to map names to locations. This infrastructure is called the Name Resolution System (NRS). When decoupled, routing protocols are used for content name advertisement and routing table population. In this case, requests are propagated by Content Routers in a hop-by-hop fashion towards the content source, while responses are retrieved utilizing reverse path routing. Caching can be performed both in an on- and off-path manner. On-path caching can be solely supported by the Content Routers, while off-path caching also uses the NRS and a hierarchical caching structure.

By the mid-2010s, the research community slowly realized that the wholesale replacement of IP as the main internetworking protocol required not only significant standardization efforts, but also agreement between a wide range of stakeholders, including operators, vendors, software developers, end-device manufacturers and policymakers [25], which made this transition unlikely in the near future.

Therefore, during recent years the interest for developing new clean-slate designs waned and the research focus shifted back to incremental deployability, as was the case in precursor architectures. This time however, the focus was not on proposing new information-centric protocol architectures, but on making the existing ones deployable on the current Internet infrastructure.

Different approaches were followed towards this direction: One such case was the POINT project [25] [26], which explored an IP-over-ICN approach to facilitate incremental deployment. It achieved this by deploying the PURSUIT architecture in individual operators as a drop-in replacement for the existing network. This approach required no change in the existing User Equipment or the IP routers of other operators and therefore sought to realize an incrementally-deployable IP-over-ICN paradigm on an Internet-wide scale which could be realized by individual operators to replace their existing network infrastructure in favor of an ICN-based one. The only requirement was that there existed sufficient performance (and, therefore, financial) incentives for operators to deploy ICN in their own private networks. As a result, the main novelty of POINT lies exactly in this proposal: the vision and roadmap for an existing ICN architecture to be deployed without the requirement for an Internet-wide consensus between all the different stakeholders.

On a technical level, POINT achieves interoperability by utilizing the following components: Hardware-wise, an ICN Border Gateway and a Network Attachment Point offer the required functionality: the former enables communication to peering IP networks latter serves as an entry point to the ICN network for the IP-based User Equipment. Software-wise, a main Pub/Sub ICN layer forms the core of the protocol stack. On top of this layer lie several protocol abstractions (IP, TCP, HTTP, CoAP) which serve as a northbound API and allow existing applications to use the ICN-enabled network. Below this layer, there is an ICN-over-SDN shim layer, which leverages SDN to deploy ICN on top of Layer-2 networks.

A different approach towards the same goal was proposed by Cisco in their hICN design [27]. This approach focuses on making the NDN/CCNx [3] architecture deployable on top of today's IPv6/IPv4 infrastructure [28]. The goal is not medium-to-long term deployment, such as the one which might be achieved by virtualization and network slicing techniques, but short-term ICN deployment utilizing the existing infrastructure.

To this end, hICN proposes a deployment strategy which targets a few nodes in the network edge. Its design aims to bring ICN capabilities into existing IP networks and entirely avoids encapsulation, tunneling or overlay deployments. To this end, the design is based on three main principles: a) the accommodation of all ICN features b) the enablement of transparent hICN packet forwarding by allowing the packets to be treated as regular IP packets and c) the reuse of most of the existing software and hardware to facilitate near-term adoption. The reference technology for hICN implementations is considered to be IPV6, even though the design does not prohibit the use of IPv4.

In hICN, content names are network-level names used by hICN routers to forward packets and are different from application-level names. The mapping between the two is made by the applications. A name consists of two parts: the name prefix and the name suffix. The former is used by the routers for forwarding purposes,

while the latter contains segmentation information. The prefix and suffix combined create hICN names which can be used to uniquely identify data.

Request/Response-type data packets are used in the form of NDN/CCNx Interest and Data packets. The two respective Protocol Data Units include IPV6 and TCP headers in which the semantics of certain fields were modified. Name-based forwarding is realized by having the name prefix being carried in the IP destination address of Interest packets and the IP source address of Data packets. Hop-by-hop forwarding is realized by having hICN routers rewrite certain fields in the header of hICN packets, namely the IP source address of Interests and IP destination address of Data packets, during the forwarding process.

From a different perspective, some recent developments combine advancements in various areas (such as Peer-to-Peer networks, Distributed Ledger Technology, Distributed Hash Tables etc.) with ideas prominent in ICN to provide distributed file storage and retrieval in an Internet-wide scale. Such networked systems are expected to form the next generation of P2P networks and have been variously termed as content-addressable networks, peer-to-peer data Networks etc. As they frequently share many common features with ICN, such as content-based addressing and name-based data retrieval, they are sometimes conceived as overlay implementations of Information-Centric Networking [29].

The most prominent example of this category is IPFS [30], a content-addressed distributed file storage system which uses the Kademlia DHT [31] for peer lookup. In IPFS, a content object can be split into chunks or blocks and each block is identified using its CID (a content identifier). These blocks are then used to create a Merkle Directed Acyclic Graph (Merkle DAG) data structure. IPFS can be used to store and share files in a decentralized manner and is frequently used as an off-chain blockchain storage service. Although it operates independently, Filecoin [32] can also be used to provide financial incentives to improve content availability.

Up to this point, we have provided some general information about Information-Centric Networking as a concept and presented its evolution as well as some of its main architectures. As a complete presentation and comparison of all ICN architectures is out of scope of this dissertation, we will focus hereafter in the Named Data Networking architecture, as it provides the basis for this doctoral thesis. A more extensive analysis of ICN architectures can be found in other works, such as [2] or [33].

2.1.3 The Named Data Networking Architecture

Named Data Networking (NDN) [3] is the most prominent ICN architecture. It is essentially a continuation of the previous Content-Centric Networking (CCN) project and respective architecture [4][34], which originated at PARC. As CCN and NDN are essentially two different implementations of the same architecture, this architecture is often referred to as the CCNx/NDN architecture [35].

NDN takes a different approach than TCP/IP to create a networking namespace for data delivery; instead of naming the *locations* to which data bits need to be shipped to, Named Data Networking names the *data bits* themselves [36]. This way, NDN makes the concept of named data objects the centerpiece of its architecture, forming a new hourglass (see Figure 2.1), which leads to a different overall network design. For instance, the direct use of application names in the network layer can greatly simplify the overall Internet architecture, as well as the protocol stack employed in individual devices. Using this approach, the need for IP address allocation

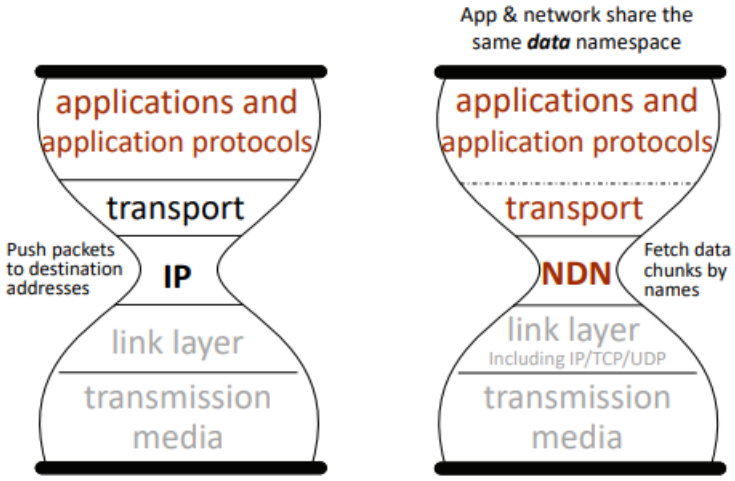


FIGURE 2.1: The NDN architecture

is eliminated. The same holds true for DNS, as the need to translate application-level names to IP addresses is also made obsolete.

NDN names are hierarchical and identify a single piece of data, e.g. */ndn/duth/dissertations/sarros.pdf*. Data retrieval in NDN is receiver-initiated and follows a request/response pattern ("pull-based" design). On a high level those features can be paralleled to HTTP's semantics, however such an operation is rendered a part of the networking layer (Layer 3).

Communication is performed using two types of packets: *Interest* and *Data* packets, which both carry the content name. Interest packets are essentially requests for data objects and Data packets are the replies containing the specific object. There is an one-to-one correspondence between an Interest and a Data packet: a single Interest packet corresponds to a unique content object which can be fetched back. Segmentation is also possible, as is the case in TCP/IP: if an application-layer data object is large, the object is segmented and the segment number is included as a part of the content name.

Content receivers in NDN are called *Consumers*, while content senders are called *Producers*. Consumer applications initiate communications by sending an Interest packet. The Interest is then forwarded towards a Producer app by intermediate routers. When the packet reaches a Producer that can satisfy the Interest, it responds with a Data packet which contains the requested data object.

One important feature of NDN is its in-network caching capabilities. Each NDN router is equipped with a cache, which is used to store content objects (i.e. downstream Data packets). NDN utilizes on-path caching, which means that a router will only cache objects when it on the return path from Producer to Consumer. Different cache replacement policies can be used and the most popular ones are natively supported, such as Least Recently Used (LRU), Least Frequently Used (LFU) etc. Data packets can either be fetched from Producers or intermediate caches, the latter being the case if an Interest packet arrives at a node which had previously cached the requested content object.

NDN nodes forward packets using *faces*. An NDN face delivers NDN network-layer packets and can represent either an inter-node communication channel (which transfers packets to other nodes), or an intra-node communication channel (which sends packets to another process on the same node) [35].

Inter-node faces can implement either physical network interfaces (e.g. Ethernet,

WiFi, Bluetooth adapters), or overlay inter-node channels (e.g. UDP, TCP tunnels). In the former case they are considered to implement an *interface*, while in the latter case an *adjacency* [37]. Intra-node faces, respectively, are used to forward packets over inter-process communication (IPC) channels to a local application (e.g. unix sockets).

As all of the aforementioned different and distinct communication abstractions are represented by faces, an NDN face is frequently described as a generalization of the concept of a network interface.

An interesting feature of the NDN architecture is that it clearly separates forwarding from routing. The routing plane in NDN assumes the function of responding to longer-term changes in the network, while the forwarding plane is concerned with shorter-term changes in topology and content availability [38]. NDN achieves this dichotomy by employing stateful forwarding [39]: each router keeps a forwarding state for each pending Interest.

NDN routing protocols can either be variants of IP networking protocols (e.g. NLSR [40], a variant of OLSR [41]), or new ones which are not considered suitable for IP networks (an example being hyperbolic routing [42]). Whatever the case, the difference with respect to IP is the fact that NDN routes directly on content names, instead of host addresses. On a high level, this means that NDN routing algorithms are not concerned with disseminating information about *host reachability* or finding routes to a particular location. Rather, they are concerned with *content reachability*: they focus on the dissemination of information on how to reach particular content objects.

Forwarding, on its part, is concerned with hop-by-hop data transfer and does not require any specific protocols per se. Its main operation is based on "breadcrumbs" paths, which are created by Interest packets as they are routed towards a content source. Each intermediate node stores information about the Interests it receives and forwards, while the corresponding Data packets follow the reverse path back to the Consumer the Interest originated from.

The decision on where to forward an Interest to is made by the so-called *Forwarding Strategies*. Example strategies include the Best Route strategy, which forwards an Interest to the upstream with lowest routing cost, the Multicast strategy, which forwards every Interest to all upstreams, the Client Control strategy, which allows a local consumer application to choose the face towards which an Interest is forwarded, etc.

The inclusion of a stateful forwarding plane introduces some tradeoffs. On one hand, increased 'smartness' is available for forwarding operations. Fault detection and adaptability are two features which are enabled by this approach, since a router can calculate the RTT for any Interest/Data exchange and adjust the forwarding process of subsequent Interests based on the collected measurements and other statistics. For instance, a face towards which Interests are forwarded but no Data are received could signify a change in network topology or connectivity issues. In this case, a different face may be chosen for subsequent Interests after observing increased timeouts. On the other hand, the enhanced functionality results in increased complexity in the networking layer. This fact has raised concerns for issues such as a potential increase in the architecture's attack surface [43], the ability of NDN routers to forward packets at line speed [44] [45] etc.

When Data packets arrive, each router uses their names to match them with the corresponding Interest packets and forward them to the appropriate interfaces.

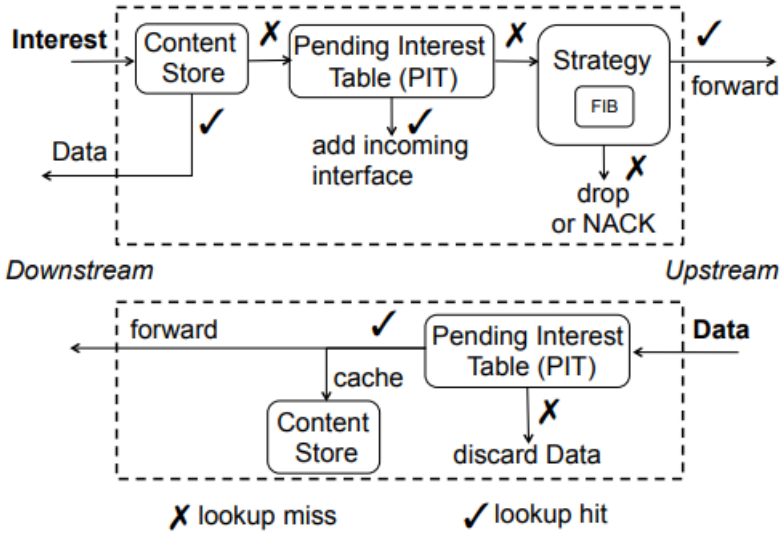


FIGURE 2.2: NDN forwarding process

Loop-free routing is achieved by the use of two means: First, no Interest can be forwarded to the face it was received from. Second, each Interest carries a 'nonce' field, which is used to uniquely identify it and detect forwarding loops.

To achieve the above, each node in NDN includes a forwarding module which is equipped with three different structures: the Pending Interest Table (PIT), the Forwarding Interest Base (FIB) and the Content Store (CS). The PIT contains information about all the Interests that have been forwarded but not been satisfied and their corresponding faces. The FIB contains forwarding information and is used to forward Interests for certain prefixes towards faces that are most likely to return the corresponding Data packets. The CS serves as each node's cache and stores the node's received Data packets utilizing specific cache replacement policies. An overview of the main NDN forwarding operations, in the form of pipelines, is presented in Figure 2.2 (retrieved from [36]).

There is also an additional structure which is peripheral to NDN forwarding, the Routing Information Base (RIB). RIB stores routing information, which is used to calculate next hops for FIB entries [46]. Each route is associated with a namespace. The RIB can be updated by different parties in various ways, some of which include the use of routing protocols, application prefix registrations, as well as command line arguments by users. The routing module processes all these requests to generate a consistent forwarding table which it then syncs with the FIB (which solely contains the minimal information required to make forwarding decisions).

One component which - although not part of the NDN architecture's core - is important in certain scenarios, is the NDN data repository (repo). A Repo can be used for data persistence in any node, in the case that applications need that functionality. It supplements network operations by responding to Interests requesting the content present in the repository. An NDN repo protocol has also been defined, as a specification of repo operations including reading, insertion and deletion of data objects in repo.[2]

The data-centric design of the NDN architecture is also reflected in its security mechanisms, resulting in a radically different approach with respect to TCP/IP. In

[2] https://redmine.named-data.net/projects/repo-ng/wiki/Repo_Protocol_Specification

particular, NDN departs from the current approach of trying to secure an end-to-end communication link and, instead, focuses on securing the data directly.

It achieves this by requiring that Producers cryptographically sign each Data packet at the time of its creation. This signature binds together the data name, producer and content [47]. The Data packets can also be encrypted, in cases when confidentiality is required. Those signatures are then verified by data Consumers, once the Data is received.

Those security properties, authenticity and confidentiality, stay with the data itself and are independent of communication channels. NDN security is therefore built upon the premised of named and secured data which are immutable, verifiable and, optionally, confidential. NDN security is therefore built upon the premise of named and secured data packets, which provide a foundational building block for the architecture's security.

Furthermore, NDN's design aims to incorporate the lessons learned from today's architecture. For this reason, it introduces some features which can provide native mitigation measures against networks attacks which frequently plague today's IP-based Internet, such as Denial of Service and Distributed Denial of Service (DoS/DDoS) attacks. One such feature is its receiver-driven, pull-based communication model effectively which takes the power away from packet senders by not allowing the receipt of unsolicited Data packets by NDN Consumers. In NDN, the only way for a Consumer to receive any data, is to have previously requested it by sending an Interest packet. This approach effectively enables proactive mitigation of DoS/DDoS and other similar attacks.

Nonetheless, receivers - who are the communication initiators in NDN - can still mirror the aforementioned attack pattern and send large volumes of Interest packets to achieve the same results. NDN's design also incorporates some features which effectively alleviate the pressure from such attacks. One such feature is the use of Interest aggregation, a technique also called *Collapsed Forwarding* in some commercial cache systems [48] and used since the early days of CDNs. Using this technique, multiple user requests for the same content are aggregated and only a single request is forwarded to the backend server.

In NDN, Interest aggregation refers to the aggregation of consecutive Interests for the same content, at intermediate routers. Should an NDN router receive an Interest packet with a specific name for the first time, it will forward this Interest to the next hop and create a PIT entry. This PIT entry will only be removed when a timer expires, depending on the Interest's Time-to-Live (TTL) field. Should a second Interest for the same prefix be received before the timer has expired (i.e., the PIT entry still exists at the reception of the new Interest), then the second Interest will not be forwarded further. Instead, the timer will be renewed and the PIT entry, which denotes a pending Interest, will remain for an additional period of time. The same happens for all subsequent Interests.

This proves a useful tool on several levels: First, it ameliorates some of the drawbacks which come with NDN's stateful forwarding plane, as the need for even larger router tables - which would keep track of every single packet on the network - is removed. Second, it improves the architecture's bandwidth usage, as not all Interest packets need to be forwarded upstream, resulting in less overall network traffic. Third, up to a certain point, it can negate Interest-based attacks which mimic DDoS, since similar Interests will be aggregated and only a fraction of packets will reach the servers which act as the content Producers. This stems from the fact that caches, combined with Interest aggregation, effectively provide some form of load balancing for the servers.

For those reasons, Interest aggregation is a native feature which is widely employed in CCNx/NDN implementations, even though the practical effectiveness of this practice has been questioned by some researchers [49]. For more extensive overview of NDN security, we refer the reader to Section 2.4.

2.2 Delay-/Disruption-Tolerant Networking

2.2.1 Overview

The Delay/Disruption-Tolerant Networking (DTN) paradigm was originally conceived as an architecture for the InterPlanetary Internet [50]. Its development arose from the need to address the challenges posed by the expansion of the Internet to non-terrestrial environments. The unique requirements of space communications, such as the ability to accommodate long delays, intermittent connectivity, heterogeneity in local network characteristics and device capabilities, etc., could not be satisfied by traditional transport-layer protocols and called for new approaches.

For instance, TCP's connection establishment process would break in the presence of long delays and intermittently-connected links, while its in-order data delivery requirements may not be desirable under circumstances in which data loss and retransmission is performed under hour- or day-long round trip times. UDP, on the other hand, does not offer reliable transport - a feature required by many applications for their proper operation. As such, it could not provide by itself a solid basis for an Interplanetary Internet architecture. As a result of those circumstances the DTN architecture was created, aiming to provide a solution for the space Internet.

However, other researchers soon observed that such network characteristics were not only present in space communications, but also existed in terrestrial environments with similar operational conditions. Thus, the DTN architecture was soon generalized to provide an all-encompassing solution for challenged networks [5].

DTN has been described as employing a postal model of communications. Due to the arbitrarily long delays and frequent disruptions, applications cannot not employ a conversational communication style. Instead, they issue messages in an asynchronous manner without waiting for a response.

Data transfer operations are based on a store-and-forward networking paradigm. It is performed hop-by-hop, as an end-to-end path between the sender and receiver may not always be available at any given time. In certain cases, some links may not be available for large periods of time, a situation which requires long-term packet storage at intermediate nodes. To satisfy this need, the DTN architecture includes features such as permanent in-network storage and reliable hop-by-hop transfer.

A frequent means to achieve data transfer in terrestrial DTNs is data muling [51], also known as data ferrying [52]. Data muling is the use of mobile devices to physically transport data from one location to another, which can prove especially useful in the absence of other connectivity options, or even as a means to achieve low-cost data transfer in remote areas [53].

To support interoperability between highly heterogeneous devices and networks, the DTN architecture is based on a simple-yet-effective overlay design. The DTN layer is effectively placed on top of the TCP/IP protocol stack and can act as a glue for different networks with different capabilities: the only requirement is that the DTN protocol is supported by the networked devices, regardless of the underlying protocols and networking stacks. An example of a DTN-capable network can be seen in Figure 2.3, as envisioned by the original DTN paper [50]).

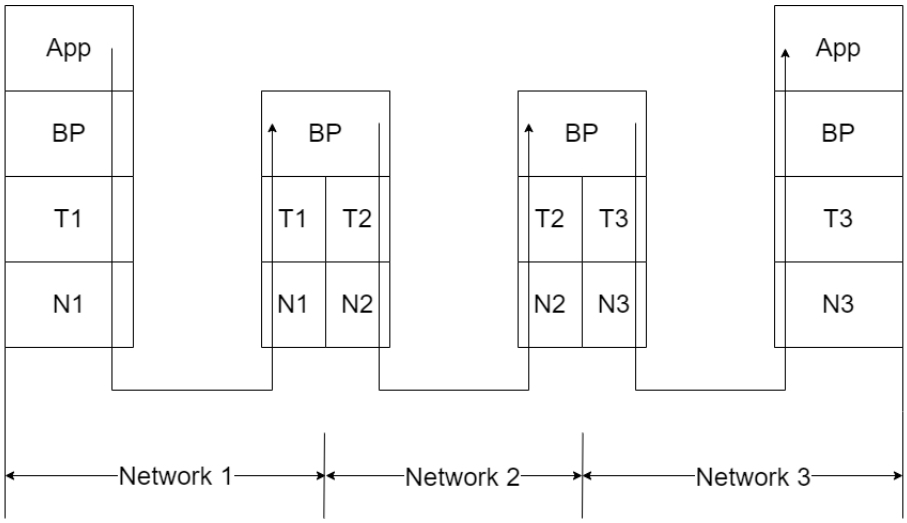


FIGURE 2.3: The DTN protocol architecture. The Bundle Protocol (BP) implements the DTN layer.

DTN creates this overlay network by using convergence layer adapters, which enable communication with the underlying internetworks (utilizing protocols such as UDP, TCP [54], etc.). Although DTN deployments frequently deploy DTN on top of transport-layer protocols, this is not a requirement. Indeed, DTN can be deployed directly over link-layer protocols and such convergence layer implementations do exist (e.g. Bluetooth [55], AX.25 [56]).

An interesting feature of the DTN architecture is its replacement of IP-style addresses in favour of a URI-based addressing scheme, somewhat similar to the name-based addressing used in ICNs.[3] It should be noted, however, that DTN does not name content, but, rather, endpoint hosts. A URI scheme enables the encapsulation of different naming and addressing schemes in the overall naming syntax.

The original DTN design of a naming scheme for the InterPlanetary Internet [50], uses a naming scheme in which addresses were comprised of two parts: The first part identifies the region in which a DTN node resides, while the second part includes the (regional) destination identifier. It utilizes *late binding*, which means that regional destination identifiers are only mapped to regional addresses once they arrive at the destination region (rather than being mapped at the time of their transmission). This simplified naming style is generally followed in space networks [59], while permitting naming flexibility for other types of DTNs (Wireless Sensor Networks, Opportunistic Mobile Networks etc.).

A particular category of DTNs which emerged out of terrestrial challenged networks are Opportunistic Networks (or *OppNets*).[4] The emergence of such networks

[3] This is, in fact, acknowledged by the seminal paper of Kevin Fall [5], which discusses DTN's naming similarity to the one used in the TRIAD project [11]. DTN and NDN [57] were both influenced by the attributed-based naming scheme used in [58].

[4] Although sometimes Opportunistic Networks are presented as a special case of Mobile Ad Hoc Networks (MANETs) [60], in this work we treat them as a special case of DTNs. This is due to the fact that a) frequent disruptions and long delays are the norm in opportunistic networks, b) they rely on the store-and-forward paradigm employed by DTN and c) historically, some of the first opportunistic routing algorithms were considered to be DTN routing algorithms [61]. Nevertheless, we note that in reality a continuum exists in networking environments based on the degree that delays, disruptions, mobility and other networking conditions are present. Therefore, even though no clear lines can be drawn, OppNets can be vaguely considered to occupy a middle ground between MANETs and DTNs.

was made possible by the advent of mobile computing devices and wireless short-range communication protocols (such as WiFi and Bluetooth). OppNets are characterized by high device mobility and irregular contacts between nodes, which result in dynamic network topologies which frequently change.

OppNets share with space DTNs the fact that they are characterized by long delays and disruptions. However, future network contacts in space communications can - for the most part - be predicted in advance. This is due to the fact that contact opportunities, as well as network disruptions, are caused by celestial body orbits which can be calculated accurately ahead of time.

In contrast, OppNets are formed by terrestrial networked devices (e.g. smartphones, vehicles) and network contacts are in fact the result of human mobility patterns. Such networked environments are characterized by opportunistic contacts, i.e. contact opportunities which cannot be deterministically predicted in advance.

Due to those differences between space DTNs and OppNets, the challenges posed by each one have to be approached from a different angle by network researchers and engineers. Perhaps the most significant area which is impacted by this division is routing, which is discussed in Subsection 2.2.3.

2.2.2 The Bundle Protocol

At the heart of the architecture lies the Bundle Protocol (BP), which is the protocol that implements all the aforementioned functionality. The Protocol Data Unit (PDU) of BP is called a Bundle. Each bundle comprises a sequence of two or more 'blocks' of protocol data; multiple instances of the same bundle may exist at the same time, in different parts of the network.

BP endpoints are usually singleton, i.e. they identify a single DTN node, but the general notion of an endpoint is broader. For example, the nodes in a sensor network might constitute a set of bundle nodes that are all registered in a single common endpoint and will all receive the data delivered at that endpoint.

The latest specification of the BP at the time of writing is BP Version 7, which is defined in the IETF RFC 9171 [62]. This specification is adapted from the earlier Bundle Protocol Specification document (IRTF RFC 5050 [63]) which was a reference point for several years. RFC 9171 specifies the BP details by defining the packet format and main processing operations (i.e. Bundle transmission and reception, forwarding, fragmentation etc.) for any potential BP implementation. Some definitions and key features of the BP are:

- Application Data Unit: An application data unit (ADU) is the unit of data which, typically, originates from an application and needs to be sent to the bundle's destination.
- Endpoint ID (EID): The destination of a bundle is a BP endpoint, which is identified by a text string, called Endpoint ID (EID). EIDs are URIs. As such, they conform to the general URI structure < *scheme name* > : < *scheme-specific part, or "SSP"* >. They need not necessarily be addresses, as neither the scheme

name nor the SSP is required to have topological significance and could, alternatively, be names [5]. An endpoint may not be singleton, i.e. multiple end hosts can be identified by a single EID.

- Destination EID: The Destination EID field identifies the bundle endpoint that is the bundle's destination, i.e., the endpoint that contains the node(s) at which the bundle is to be delivered.
- Source node ID: The Source node ID field identifies the bundle node at which the bundle was initially transmitted. In case anonymity is desired by the bundle's source, it can be supported by replacing the source node ID with the null endpoint ID.
- Bundle Lifetime: The Lifetime field indicates the time at which the bundle's payload will no longer be useful, encoded as a number of milliseconds past the creation time. When a bundle's age exceeds its lifetime, bundle nodes need no longer retain or forward the bundle and the bundle will typically be deleted from the network.
- Fragmentation: A fragment, or "fragmentary bundle", is a bundle whose payload block contains a partial payload. Sometimes, it can be advantageous do reduce the sizes of bundles so that they can be forwarded. This can, for instance, be desirable if a bundle needs to be forwarded to a certain node which is accessible only via intermittent contacts but no upcoming contact is long enough to enable the forwarding of the entire bundle. In this case, a bundle can be replaced with two fragments which are essentially two bundles with the same source node ID and creation timestamp as the original bundle. As fragments are bundles themselves, they can also be fragmented fragmented and consequently, multiple fragmentation cases may take place so that an original bundle is, ultimately, replaced with more than two fragments.
- Custody Transfer: The concept of custody transfer was introduced in DTNs to achieve hop-by-hop and, ultimately, end-to-end reliability [64]. A DTN node can optionally accept the custody of a bundle, in which case it is called a *custodian*. Applications can request custody transfer per message and are delivered a custody acknowledgment the message is moved to one or more other nodes that accept to take custody for the message. The custody acknowledgment is, therefore, not an end-to-end acknowledgment. It only indicates that the responsibility for end-to-end reliable delivery has been delegated to the node accepting the custody. Although custody transfer was described as a feature of the BP specification in RFC 5050, it is not present in RFC 9171 and has since been proposed to be included as a part of the Bundle-in-Bundle Encapsulation (BIBE) convergence layer [65].

The components of a Bundle node implementing the RFC 9171 specification [62] can be seen in Figure 2.4.

[5] An address has topological significance and identifies a location in a certain space. Given that space, an address provides all information that is required to send information to that location, but does not identify the entity which occupies the location. A (host) name identifies an entity to which one might send information, but does not provide any information regarding the location of that entity [59]

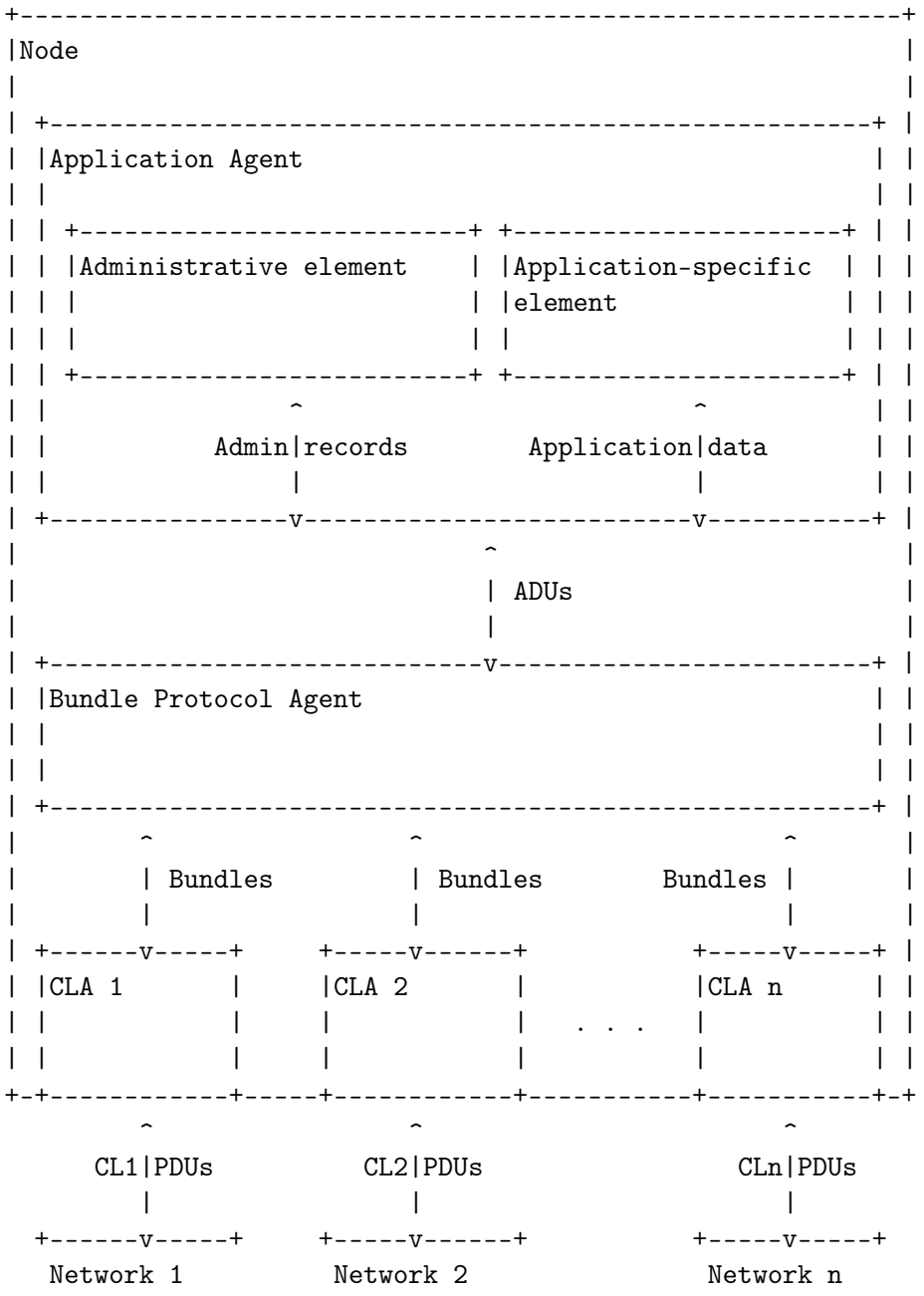


FIGURE 2.4: Components of a Bundle Node

2.2.3 Routing

Routing can be defined as "*the procedure by which, at each point in the path from A to Q, we select a neighboring branch point to transmit the data to, believing that branch point to be on the best path for conveying the data to its destination*" [66]. At this point, a new route may be computed based on the network state information available, or - alternatively - a path which has been previously computed by another node may be followed.

When DTN arose as a discrete network area, it was soon realized that routing posed a very different problem in such networks: Internet routing algorithms and approaches could not be used and new solutions had to be invented [67].

In the Internet, connections are stable and connectivity is continuous, which means that changes in the network state can be propagated to other nodes almost instantly. For this reason, Internet routing protocols are based on the assumption that each node's understanding of the network is, in most cases, correct. Therefore, the selection of a neighbouring node to forward the data to can be made reliably.

In DTNs, however, this is not always the case. Due to the network delays and disruptions which are inherent in such networks, changes in network state may occur far more frequently than they can be propagated to the rest of the network. As a result, the actual network state at any time may not be known by a node and therefore, the best path cannot be determined with high confidence. What is more, data transmission to the specific neighbouring node which lies on the best path may not be possible at any time.

Furthermore, one additional problem which has to be solved by DTN routing, is that it is not always sufficient to select *who* to send the data to (i.e., the best neighbouring node), based on the topological characteristics of the network. What needs to also be considered is *when* to send the data, based on the expected time of its arrival to the destination.

Based on the degree of network knowledge which is available, DTN routing algorithms can be broadly split into two main categories: *deterministic* routing and *opportunistic* routing approaches. The former approach assumes the presence of complete and accurate network state information, while the latter assumes none.

On one hand, deterministic routing approaches can guarantee successful delivery with efficient use of bandwidth and buffer space in networks with predictable establishment of contacts. On the other hand, opportunistic routing can offer a higher delivery ratio in highly mobile networks with unpredictable contacts (albeit with the tradeoff of higher storage space and bandwidth use). As is easily deduced, deterministic routing approaches are mainly applied in space DTNs with predictable contacts, while opportunistic routing ones in terrestrial mobile DTNs with opportunistic contacts (i.e., OppNets).

Regarding deterministic routing, the de facto approach is Contact Graph Routing (CGR) [66][68], a dynamic algorithm which assumes a time-varying topology of scheduled communication contacts to compute routes.

CGR leverages the fact that space flight missions are planned in advance with extraordinary detail and the same holds true for their communication operations. As those plans are know a priori, CGR uses that information to infer the communication opportunities and routes between any pair of DTN nodes. By relying on the accuracy of this information regarding network topology and its expected changes over time, routing can be performed confidently. In this aspect, CGR resembles traditional Internet routing, as topology state information is assumed to be complete.

However, in the former case, the topology is not the actual topology known at the time of routing but, rather, an expected one (albeit, with very high precision).

The cornerstone of its operation is the *contact plan*, which is a time-ordered list of expected changes in network topology. Each list entry is denotes a contact: it includes the time instances at which transmission from a certain node to a another commences and finishes respectively, as well as the nominal data rate for the contact.

Routing tables are constructed by each node using the contacts which are present in the contact plan. Each routing table includes several route lists, one for each possible destination node. The neighbouring node which forms the initial node for a route, is called its "entry node".

The *contact graph* is a Directed Acyclic Graph (DAG). The DAG root node is a contact between the local node and itself. Likewise, its terminal node is a contact between the destination node and itself. The rest of the graph's vertices are created by all contacts which are part of an end-to-end path from the local node to the destination one. Dijkstra searches are performed in the contact graph to find the lowest-cost route from the root to the terminal node. After each search, the root is added to the local node's route list and the first contact of that route is deleted from the graph, before the search for the next best route initiates. The search procedure is terminated once it fails to find a route.

One of the benefits of CGR is the fact that it can not only be applied to interplanetary communications, but also to Low Earth Orbit satellites as they share similar characteristics (i.e. scheduled connectivity). Furthermore, it has been shown to be extensible and capable to accommodate opportunistic contacts [69].

Opportunistic routing algorithms *per se* form the other major approach on DTN routing. Most of those approaches rely on bundle replication strategies to ensure data delivery in the absence of knowledge about future contacts and end-to-end paths.

The first such approach was Epidemic routing [70], a flooding-based approach. Its simple, yet effective, design was based on the following rationale: mobile nodes forward a bundle to all nodes which they encounter, only excluding those nodes which have previously received a copy of the specific bundle. Using this technique, the probability of delivering a bundle to its final destination is increased. However, the transmission and storage costs are very high.

Subsequent algorithms such as Spray-and-Wait [61] attempted to retain a comparable bundle delivery ratio, while limiting the network overhead. To do so, Spray-and-Wait utilizes an approach based on controlled flooding [6]: it effectively stops forwarding copies of a certain bundle to the network, after a certain number of copies has been reached.

Other algorithms were built on a different premise: they tried to predict the likelihood of future contacts. The first of those algorithms (sometimes collectively called *probabilistic routing* algorithms), was ProPHET [71]. ProPHET aims to reduce the network overhead of flooding, by only forwarding a bundle copy to those nodes which have a higher probability of delivering the bundle. This delivery probability is calculated based on each node's history of encounters, which is propagated across the delay/disruption tolerant network. To increase delivery probability, multiple copies can be sent via different paths.

A subsequent variation of probabilistic routing can be found in MaxProp [72]. MaxProp assigns a cost to each destination - based on packet delivery probability - and, according to this cost, creates a ranked list of packets. It also includes the use

[6] Also known as *controlled replication*, or spraying.

of complementary mechanisms. Acknowledgments for packet delivery are utilized, which are not only sent to the packet source but to all peer nodes. Prioritization is also an important feature, as MaxProp assigns a greater priority to new packets while attempting to prevent the reception of duplicate ones. The packets with the highest priority are the ones transferred first during a contact, while those with the lowest priority are the first to be deleted when storage space is depleted.

Finally, a more recent class of opportunistic routing algorithms go one step further and do not assume that contact opportunities in OppNets are entirely random. Rather, they argue that - as many such networks are based in human mobility - some underlying patterns which characterize human behavior can be identified and exploited to improve opportunistic routing performance in Pocket Switched Networks (PSNs) [73][74].

Perhaps the first protocol which exploits this insight is BubbleRap [75]. Bubble Rap attempts to avoid the excessive control traffic which is used in other opportunistic routing protocols to create routing structures (which are not always reliable), by searching for network characteristics which are more stable than mobility. The authors argue that such characteristics can be found in people's social behavior, which shows less variation and changes in larger periods of time. Therefore, they state that the decentralized detection of such patterns to select the best forwarding paths is the primary goal of this work

The BUBBLE algorithm is based on the following intuition: First, that the various roles and popularity of people in society should also be reflected in the network layer. Second, the fact that communities are formed by members of a population in their social lives and, again, this should also be observed in the network layer. Thus, BUBBLE utilizes the concepts of community structure and node centrality to identify the most suitable nodes to forward packets to.

Those main principles are translated into the following specific forwarding decisions by BUBBLE: a) the nodes chosen as relays should be more popular than the current node (as identified by their high centrality) and b) the relay nodes should be identified as members of the destination communities. The authors show that this approach can outperform earlier approaches which do not utilize social information, by comparing their results to the ProPHET routing algorithm. Those results show that BUBBLE can achieve comparable delivery ratio, but with lower resource utilization.

Another approach following those steps, is dLife [76]. dLife embraces the idea of underlying social structures which can be exploited by routing algorithms in OppNets, but takes this idea one step further. Instead of assuming that social structures are stable over time, dLife takes into account the dynamism of social relationships and tries to incorporate those dynamics into its routing algorithm, as they are represented by time-evolving social ties between pairs of nodes. It relies on the observation that users have routines which can be leveraged to predict their future behavior and essentially aims to create a more accurate representation of future social contacts than the one which is created using proximity graphs inferred directly from inter-contact times.

To do so, it creates a weighted contact graph, in which the weights denote how long two nodes are in contact over various periods of time. To this end, two utility functions are considered: The first one is Time-Evolving Contact Duration, which attempts to express the evolution of social interaction between pairs of users in the same daily time period over consecutive days. The second is Time-Evolving Contact Duration Importance, which attempts to express how a user's importance evolves,

as denoted by the node degree metric and the strength of its social ties towards the neighbors.

Latest trends in opportunistic routing include the development of algorithms specifically designed for vehicular and flying networks[7] which typically incorporate geographical information (e.g. regarding route/trajectory) to improve routing performance [77][78][79]. Notable is also an increased focus on energy efficiency [80][81], as well as a recent effort to incorporate machine learning techniques into the routing process, to increase its performance [82] [83].

2.3 Information-Centric Delay-Tolerant Networks

2.3.1 Overview

Since the appearance of the DTN (initially) and ICN (subsequently) research areas, there has surfaced a particular research direction which attempts to combine the two principles on which the two networking approaches are based. The works which followed this direction have variously attempted to create networking protocols or architectures which integrate the principles of both information centricity and delay/disruption tolerance in their operation.

Different works have approached this issue from different angles. Most works use one of the two networking approaches as their starting point and attempt to introduce features of the other approach into the protocol operation or architectural mechanisms. Most often than not, this involves modification of the original protocol operation and/or semantics, which can have an impact on interoperability.

Before ICN was formulated as a discrete field, certain works appeared in the DTN research community which could be considered the precursors of later attempts to combine the two networking approaches. Those early works opened up a new line of research in DTN by shifting their focus away from finding more efficient ways to route bundles to intermittently-connected hosts (which was the primary research current at the time) and, instead, tried to develop efficient content retrieval mechanisms.

For instance, the authors of [84] note that application protocols in DTN are not interactive and messages are assumed to be semantically self-contained. Based on this observation, they propose that the bundles queued in DTN routers are used for caching and distributed storage. To achieve this, they define a bundle protocol extension header to embed *application hints* in bundles. This enables the identification of resources, as well as the requested operation type (e.g. ability to differentiate between request and response-types bundles). Furthermore, intermediate DTN router nodes need to be enhanced with an application-aware processing module; this module interacts with the queue management and - optionally - the routing logic and allows different forwarding and queuing decisions to be made. Last, by setting the message TTL to larger values than is needed to deliver a bundle, cooperative caching and distributed storage schemes can also be enabled (even for fragmented resources [85]).

A similar goal was pursued concurrently in another work [86], which exploits DTN's late binding property to build a content retrieval system comprised of three components: data caching, query dissemination and message routing. One of the

[7] Frequently called Vehicular Ad Hoc Networks (VANETs) and Flying Ad Hoc Networks (FANETs) in the literature, a nod to the earlier Mobile Ad Hoc Networks (MANETs). In this thesis we avoid using the terms VANETs/FANETs when referring to OppNets which are formed by vehicular/flying networked devices, as MANET routing differs in its design and operation from OppNet routing.

main features which enables this functionality is the use of beacons - which are broadcasted periodically - to disseminate content-specific information. In this design, those beacons include a list of the content objects stored at each node and a list of content objects which lie within the node's neighbourhood (as defined by a hop count). Likewise, each node maintains two lists of cached content objects: one which contains all items cached locally and another one which contains objects cached in its neighbourhood. Another interesting feature is that, in this scheme, queries are also cached and inserted into a Cached Query List.

Other researchers attempted to induce publish/subscribe functionality in DTNs. For instance, the authors of [87] developed the DTN Pub/Sub Protocol (DPSP), which was created in the form of a probabilistic routing protocol. DPSP leverages DTN's multicast capabilities to implement pub/sub in a distributed way. The content is identified using a channel-based system: receivers interested in specific content can subscribe to channels (identified by URIs), which are then used by senders to publish their content. The authors note that those channels can be used as EIDs in the Bundle Protocol. Similar to [84], content is transmitted to self-contained application data units. This feature is used to decouple senders and receivers, as receivers are rendered capable of processing content the moment they receive it, regardless of whether the sender is reachable at the time.

In a similar spirit, the authors of [88] introduce a different pub/sub system: the Time-Aware COntent-based dissemination system for Delay Tolerant Networks (TACO-DTN). TACO-DTN assumes that the network is comprised by both mobile DTN devices, which connect with each other in an infrastructureless way, as well as devices which act as infostations (edge nodes with connectivity to the backbone). Their approach builds a content routing system based on the concept of temporal utility for subscriptions and events. Their main idea is the following: The validity of both subscriptions and publications is time-dependent. This observation is incorporated into the pub/sub system for temporal event matching: publications are matched against subscriptions not only based on their content, but also based on their temporal information. Infostations are central to this design: depending on the subscriber interest and their periodicity in an infostation area, only publications relevant in the near future will be routed to this infostation. Although focused on DTNs, this work did not include any specific directions regarding implementation over the Bundle Protocol (as the design was evaluated on the OMNeT++ simulator).

Later works on this line of research build and expand on the previous concepts. An example is the concept of Floating Content [89], an ephemeral content sharing service for mobile opportunistic networks, which limits information dissemination to a geographic area. It achieves this explicitly defining each object's *origin* and emphvalidity radius, to tie the object to a specific *anchor zone*. Other works focused on improving cooperative caching, proposing to cache the content at network central locations and working on effective mechanisms to achieve this [90].

Content retrieval functionality is not only generally compatible with DTNs, but could also be integrated in the DTN reference architecture. Such an attempt is the SPINDLE system [91], which is built upon DTN2 - the reference implementation of the Bundle Protocol. SPINDLE attempts to achieve several goals, some of which include opportunistic caching, indexing and data retrieval mechanisms to support content-based access to information. To this end, they create a a disruption-tolerant pub/sub system that supports push, pull, and third-party push metaphors. Although specific implementation details are not provided for the pub/sub system, the authors state that their main idea is that content flow and caching self-organizes

around supply and demand (who provides what/who needs what) and that this supply and demand is disseminated within the disruption-tolerant network.

Some researchers even went as far as proposing to standardize content retrieval functionality for the DTN reference architecture, by integrating it into the Bundle Protocol itself. To this end, they designed - and suggested the adoption of - the Bundle Protocol Query Extension Block (BPQ) [92]. BPQ leverages the fact that DTN naming does not require EIDs to represent host addresses, but is generic and flexible enough to accommodate any kind of URIs[8], such as ones specifying content objects. On this ground, the authors proposed an extension block which allows queries for information objects, which can be answered by intermediate BP nodes (so that content objects copied in intermediate nodes can be retrieved without the queries arriving to the query destination/content source). Using BPQ, intermediate nodes do not have to be addressed using the destination EID of a query and, as a result, some form of name-based content retrieval and in-network caching is essentially enabled in the DTN architecture, using the Bundle Protocol itself. The status of this proposal, however, did not proceed further than an (obsolete) DTNRG Internet Draft and was never standardized as part of the DTN architecture.

Although the aforementioned works did not explicitly attempt to integrate Delay-/Disruption-Tolerant Networking with an ICN architecture, they did in fact attempt to incorporate information-centric *principles* and *mechanisms* in the operation of delay-/disruption-tolerant networks. Functionality such as in-network caching, publish / subscribe semantics, content naming and routing (which was discussed in the context of those works) later constituted some of the defining features of ICN architectures.

A different and, perhaps, unique approach was provided by Haggle [93] [94]. The Haggle architecture revisits delay/disruption-tolerant networking at the architectural level to enable opportunistic content dissemination. It is built on the principle of ranked search to create a data-centric communication model and match content against nodes, inspired by web and desktop search. The novelty that this approach brings to other pub/sub systems is the fact that searching enables relative matching - which may change over time as new content becomes available - instead of the consistent matching which is performed by other approaches. The use of metadata is also a feature which adds flexibility to content dissemination, as it allows for data ranking to enable prioritization during the dissemination process. Despite its applicability on delay-tolerant and opportunistic networks, Haggle is radically different from the DTN architecture, as it focuses on data-centric (rather than host-centric) communication and does not name end-hosts or provide any bundling functionality.

As a clean-slate, data-centric network architecture for delay-tolerant and opportunistic content dissemination, Haggle can arguably be called the first Information-Centric Delay-Tolerant Networking (ICDTN) architecture[9]. Contrary to most ICN architectures, data dissemination in Haggle is based on a "push" communication model, to assist out-of-community data forwarding. Content holders decide themselves the next best hop to forward the data to and forwarding decisions are taken hop-by-hop, based on the interests (i.e. subscriptions) which are collected as other nodes are encountered. The bindings between content and data are performed by searching and renewed on a hop-by-hop basis, resulting in late binding (which the authors consider to yield more accurate results, as they expect more information

[8] 2.2.2

[9] In fact, it could also be considered the first 'pure' ICN architecture, as DONA was presented one year later (2007)[19] than the original Haggle paper (2006)[93].

about the network to be collected the later the binding is performed). Overall, Haggle can be described as realizing a push-based data dissemination service with a pub/sub API.

Since the appearance of the first clean-slate ICN architectures and the subsequent proliferation of research in the field, researchers noticed certain similarities between the (new) ICN and the (more mature) DTN networking approaches: both included in-network storage, late binding between names and location, long-term object persistence (when compared to IP packets) and flexible routing [95]. To this end, potential synergies were identified and the term Information-Centric Delay-Tolerant Networks (ICDTNs) came into existence to networks which adopt both principles in their operation.

Gradually, the efforts to unify the two networking approaches gradually shifted to the other end. In contrast to the earlier DTN-based approaches, most works started adopting ICN architectures as a starting point, into which they attempted to integrate delay-/disruption-tolerant networking principles.

In some cases, support for delay/disruption tolerance was considered since the early stages of an architecture's development. For instance, the original DONA paper [19] suggested that the DONA architecture could provide a substrate for DTN deployments and provided some hints on how to implement features such as custody transfer on DONA. However, to the best of our knowledge, no subsequent works followed this direction.

Another example is the NetInf architecture [24], which uses convergence layers to provide ICN functionality on top of existing protocols. Using those convergence layers, NetInf can operate on top of very heterogeneous environments, including Delay-Tolerant Networks. Towards this end, the SAIL project has implemented a convergence layer which utilizes the Bundle Protocol as a transport for NetInf messages. This is achieved by utilizing the BPQ as a way to carry the name of a content object - or search strings needed to identify content - in NetInf messages [96].

The MobilityFirst architecture [23] also included operational considerations for delay-/disruption-tolerant networking environments during its design and development. It aims to achieve this mainly using two components: The first one is a Global Name Resolution Service (GNRS), which dynamically binds the name of an entity and its current location. Flat, globally unique names (GUIDs) are used, which may correspond to a content object or devices. The second one is an intra-domain routing protocol (GSTAR [97]), which utilizes local storage and path quality information in a hop-by-hop fashion to operate over a spectrum of disruptive mobile networks (ranging from MANETs to DTNs). MobilityFirst supports the use of both GUIDs and addresses, which can be both used for routing decisions.

Contrary to the previous ICN architectures, other ones (e.g. PSIRP/PURSUIT, CCN/NDN) were not originally developed to support delay and disruption tolerance. When the inclusion of such functionality came as an afterthought, the researchers who tried to solve this problem were essentially presented with two options.

The first one is to modify the architecture to support delay/disruption-tolerant communications. The price for this, when considering clean-slate ICN architectures, is that those modifications can impact the compatibility with the original architecture (especially when changes are introduced in the behavior of the network layer protocol).

The second choice is to keep the ICN layer intact and attempt to support delay/disruption tolerance at higher or lower layers of the protocol stack. Even though

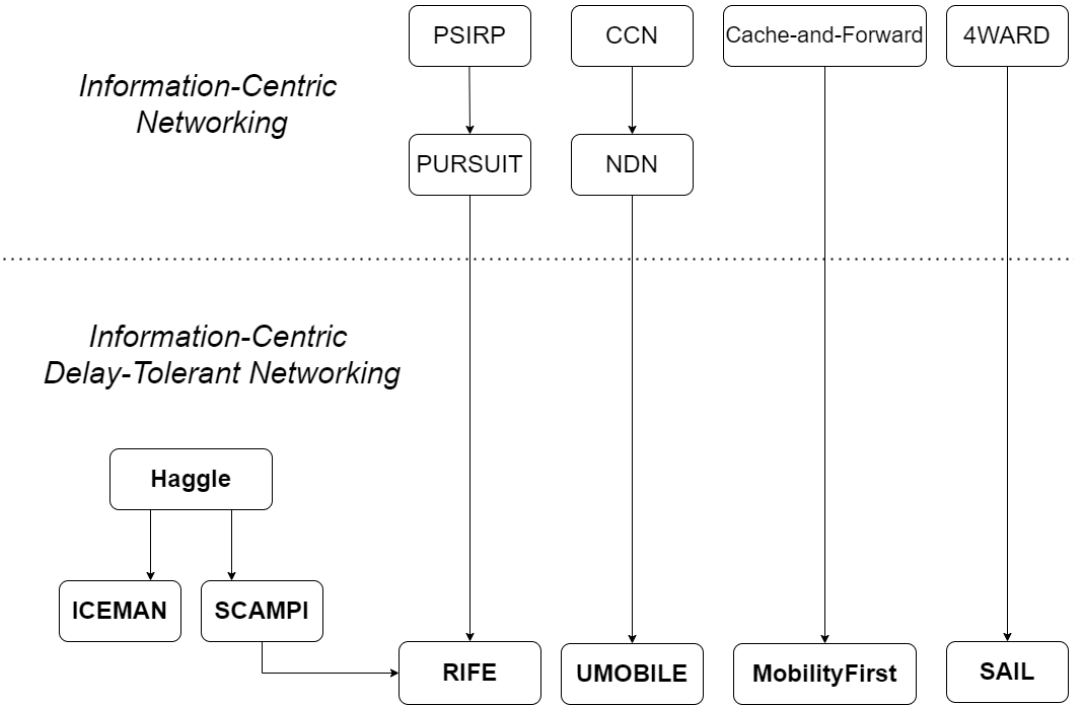


FIGURE 2.5: Information-Centric Delay-Tolerant Networking projects and their relationship with the original ICN architectures.

DTN was originally conceived as an overlay which unified communications between heterogeneous internets (and there is nothing in principle which prohibits BP deployment on top of an ICN network layer), most designs which follow this direction seem to choose the opposite layering and utilize BP as an underlay, effectively acting as one of the available transport options for ICN.

Such an approach was followed by RIFE [98], which uses the PSIRP/ PURSUIT architecture. The RIFE design considers a common information-centric (pub/sub) abstraction which overlays different connectivity options, called dissemination strategies. One such strategy for DTNs is the "disconnected dissemination strategy" [99], which is implemented using the SCAMPI DTN middleware [100].

As this dissertation is based on the Named Data Networking architecture, hereafter we are going to focus solely on works which attempt to make CCN/NDN operable on delay-/disruption-tolerant networks. An overview of ICN projects which have attempted to accommodate delay-/disruption-tolerant networking (along with their ancestor ICN architectures and related projects), is provided in Figure 2.5.

2.3.2 CCN/NDN-based DTNs

One of the first works to tackle opportunistic networking issues in the CCN architecture was [101]. The authors note that certain features of CCN can prove beneficial in opportunistic networking environments, such as the absence of the need for a neighbour discovery process (which requires periodic beacons and drains mobile device battery in host-centric opportunistic networks). To this end, they design and evaluate two different mechanisms for content discovery in dynamic environments with no central directories, namely Enumeration Request Discovery (ERD) and Regular Interest Discovery (RID). The goal is to allow users to obtain information about the available content without them having to download the entire content. Both of

the mechanisms are based on wireless broadcasting and focus on single-hop connections, such as the ones present in OppNets. The authors assume the existence of a repository with persistence storage on each CCN node, a requirement which may not be generally true in CCN but is a reasonable assumption for DTN scenarios (as DTN itself requires persistent storage). The common concept between the algorithms is that a node first broadcasts an Interest with a general prefix and waits for responses. Based on the responses, subsequent Interests are sent. Those subsequent Interests exclude the previously discovered content, until no more content objects in the particular namespace are deemed to be available in the neighbour repositories.

The same authors observed that when using this mechanism, content objects may be cached locally, but persistence is not guaranteed and the content chunks may soon be replaced by subsequent data. As this is a situation frequently encountered in mobile DTNs - where short network contacts are more frequent and in which content transfers may be interrupted before completion - the authors devote a later work [102] to the development of a mechanism for storing and retrieving partial data files from secondary storage. To this end, they still utilize their previous CCN-based multicast mechanism for content discovery in OppNets, but propose the addition of another mechanism for content retrieval, which effectively extends caching to persistent storage to resume disrupted content transfers.

To achieve this, they propose a design which is based on two tenets: The first one requires the storage of specific information by the requester side on secondary storage, namely the partial file received and some relevant MetaData, to obtain storage persistence and allow for resume operations in case of a disrupted, incomplete transfer. The second one is an algorithm to be used in content transfers when network disruptions occur, vaguely reminiscent of TCP congestion control algorithms. This algorithm utilizes the aforementioned metadata and defines clear steps for the download sequence in such cases. To avoid some issues which may arise when deploying their design (such as the storage of partial files for transfers that never get completed, or treating real-time traffic as needing to resume), the authors add an expiration timer, after which the metadata and partial files can be deleted from permanent storage.

The authors continue their work on CCN-based OppNets in [103]. In this paper, they move away from single-hop scenarios in which a requester encounters a content source directly. Instead, they develop a mechanism to achieve multi-hop forwarding. They state as a main challenge the fact that CCN requires symmetric forwarding, i.e. that CCND Data packets have to follow the same path back to the requester that the preceding Interests have followed to reach the content source, as well as the fact that Interests may time out at intermediate nodes.

To overcome these problems, they propose a scheme in which content requesters delegate content retrieval to one-hop neighbor nodes which retrieve the content on their behalf. Their approach consists of 3 phases: i) The agent delegation phase, ii) the content retrieval phase and iii) the notification phase. During the 1st phase, the requester attempts to find an available agent to delegate content retrieval. During the 2nd phase, the agent searches for the content and retrieves it. During the 3rd phase, the requester polls the agents to determine whether the content was retrieved. After all phases have been completed, the requester can retrieve the content from the agent by simply sending a regular Interest. Multicast is used in most operations (as in the authors' previous work), however once the requester determines that an agent has received the content, it can be retrieved from that particular agent using unicast.

This approach is initially evaluated using a CCNx implementation on Android devices, on a network consisting of 3 nodes, The authors later refined some aspects

(agent delegation and content notification in disrupted interactions, message naming in accordance to the CCNx naming conventions) [104] and evaluated its performance on ns3-DCE on a Linux cluster, deploying their code in simulated mobile devices (supporting up to 100 nodes).

One of the benefits of the author's approach is that they do not modify CCNx message processing or message structure. Therefore, their approach is compatible with the original architecture and other routing protocols. However, there are still some requirements which need to be fulfilled prior to deployment. Perhaps the most restrictive one is the fact that, when considering an infrastructureless OppNet environment[10], the content requesters need to meet their agents again to retrieve the desired content. This assumption may not always hold true (depending on the node mobility patterns, network size and general characteristics) - at least not for extended periods of time. Furthermore, only after the completion of the 3 delegation/retrieval phases, can the Consumer request the content. Therefore, it effectively requires 4 distinct stages until the actual content retrieval.

Last, the entire agent-based content retrieval process is actually not a protocol integrated into the network stack, but requires the installation of application modules in mobile devices (e.g. agent and repository apps). While - on the upside - this loose coupling enables the integration with different CCNx routing protocols and possible operation in both dense and sparse environments, it nevertheless constitutes an additional barrier for deployment in mobile devices (e.g. the requirement of additional computational resources), instead of being integrated into the protocol architecture and working out-of-the-box. We note however that the last requirement seems reasonable for DTNs and OppNets, as a comparison with host-centric solutions shows that large storage capabilities are practically always required, while solutions such as the Bundle Protocol which constitute an overlay to TCP/IP may also be deployed in mobile devices in the form of mobile apps[11].

The aforementioned works were some of the first attempting to achieve delay/disruption tolerance in ICN. Other early works also include ICON [105], CEDO [106] and DID [107].

ICON is an information-centric architecture for opportunistic networks. Its semantics and operation is inspired by CCN and was developed with interoperability in mind, while its implementation is modular - similar to the Haggle project. Its implementation consists of 3 main modules: the Data engine (which handles transient and persistent storage of information/messages), the Network engine (which is responsible for interface management) and the Decision engine (which takes all decisions and can support human behavior inference and contextualization). A separate module, the Protocol module, includes submodules for data naming (according to CCN conventions) and expressing Interest/Data packet-type messages.

Although ICON sketches a platform design for content-centric OppNets and implements a prototype, the authors simply validate the node configuration and verify

[10] The authors note [104] that in hybrid environments in which access points connected to each other (e.g. using wired links) are available, delegation can be performed in one access point and retrieval on the other one. However, this presupposes the existence of fixed links and moves outside the original scope of infrastructureless OppNets.

[11] Perhaps due to the fact that DTNs have looser requirements, when compared to fixed networks, regarding some performance aspects. For instance, high-speed packet forwarding is (although desirable) typically not a strict requirement for mobile DTN routers, contrary to fixed network wired routers which require high-performance implementations. Another aspect is the fact that most of the open-source DTN implementations arose out of academic - instead of industrial - environments and were developed mainly for experimentation purposes

the correct flow of messages between 3 laptop nodes. Moreover, the details published in [105] denote that ICON was work-in-progress, as some of the modules (i.e. the Decision engine) were still under development and no algorithms or mechanisms to enable opportunistic networking were provided. Moreover, interoperability with CCN is claimed as a goal, but does not seem to have been validated. Lastly, no performance evaluation was provided. Therefore, ICON could be considered as an important, yet incomplete, first step: It posed the problem and general directions towards its solution, but it did not tackle specific, individual problems which arise in content-centric opportunistic environments.

In this regard, CEDO [106] can be considered the first fully-fledged CCN-based solution for DTNs. The CEDO algorithm considers a mobile DTN environment, in which nodes can store only a subset of the available content copies which exist in the network. In this environment, the authors argue that the scheduling and drop problem can be turned into a storage allocation problem. Thus, they try to find the optimal way that content objects should be replicated so that network throughput is maximized.

The authors present a mathematical model, through which they arrive at the following conclusion: delivery rate of all contents is maximal when the delivery rate utility - the miss rate - is constant. Therefore, CEDO essentially strives to equalize the miss rates of all contents to optimize the network throughput. Other performance objectives are left as future work.

Having defined this particular optimization problem, the authors posit that the optimal policy to achieve their goals is the following: When a node's buffer of a device is full, CEDO drops the content object with the smallest rate of unsatisfied requests. When another node is encountered, CEDO sends the content with the highest rate of unsatisfied requests. To calculate miss rates for content objects, nodes estimate the corresponding request and delivery rates by recording the time when the application requests an object and the time when the respective request is satisfied. This local estimation is merged with estimates received by other nodes.

The algorithm is implemented n CCNx v0.6.0. Experiments are carried out using ns3-DCE, which enables the porting of code implemented for real mobile devices in an emulation setting. During the evaluation experiments CEDO shows good results and seems to exhibit the desired behavior (and better performance when compared to an LRU policy).

CEDO's main drawback lies in the fact that it requires the modification of CCNx packet processing operations. In particular, the authors need to modify the header of content objects to include the request and delivery rate estimates. Furthermore, they disable the propagation of Interest messages to peers (FIB Interest forwarding) and Interests stay in the PIT until they are satisfied. The last modification is the introduction of 2 additional control packets which implement CEDO-specific messages ("Hello" and "Pending Interests"). Therefore, their implementation is not interoperable with the original CCN architecture.

DID [107] focuses on disaster scenarios. The authors consider a setup which consists of network fragmentation and data mules which move between different areas. Each separate fragmented community is assumed to deliver messages to a specific gateway node, which communicates with the data mules and the rest of the external world.

The authors' ICN architecture is conceptually based on NDN and introduces the concept of logical faces. Logical faces are not tied to a specific physical interface, as the authors argue that this may cause problems in case of gateway, or other device failures. They posit that by using logical faces, other nodes can assume the role

of a gateway in a community without messages being lost. Therefore, in DID, the mules and community gateways are considered mobile routers with logical faces to one another, which become active in different periods of time (when the respective entities are encountered). Besides logical faces, the authors also introduce a protocol for communication in such a scenario, which uses specific message priorities to improve the delivery of high-priority messages. They evaluate their work using the ONE simulator and compare their algorithm to the *Epidemic* and *Spray and Wait* ones, showcasing improved performance.

The solution solves an important issue, which is the use of different mules for upstream Interest and downstream Data packets. Its main goal seems to be achieved, however this goal is a rather narrow one, as this work does not propose a generic solution for ICDTNs and includes several scenario-specific assumptions. Furthermore, the authors introduce several modifications to the original NDN architecture to achieve their goals and, thus, the two architecture are not interoperable. For instance, in DID, Interest messages may denote a standing query (similar to a subscription in pub/sub systems), contrary to the one-to-one correspondence between NDN Interest and Data packets. Furthermore, the FIB tables also includes an additional field which denotes the probability of the next hop community being the best next hop for the Interest. Last, data mules also include an additional structure, the Encounter Table, which associates the logical faces towards each known community gateway, with the probability that the mule meets the specific gateway in the future. Therefore, DID can be thought more as a scenario-specific architecture, than a generic way to extend CCN/NDN in delay-tolerant networking environments.

A similar environment - a disaster scenario with fragmented networks and data mules - is considered in [108]. In this demo, the authors implemented a routing and retransmission mechanism for CCNx to enable disruption-tolerant data forwarding. Their routing algorithm (DSDVN) is based on DSDV MANET routing one, which is extended to convey name prefix information in the routing message. DSDVN manages the status of links between neighboring nodes, which can be either 'connected' or 'disconnected'. The status information is included in CCNx faces and is used by the retransmission mechanism. The latter assumes that intermediate nodes do not suppress Interests (as Producer/Consumer nodes do) and uses large Interest Lifetime values. The main logic is rather simplistic: IF a PIT entry exists AND the status of the face towards the next-hop node is *connected*, intermediate routers retransmit the message. Otherwise, retransmissions are suppressed. As this work accompanied a demo, the authors simply validate their solution using 4 nodes and do not provide any quantitative evaluation. As was the case with the previous work, due to the very specific setup and related assumptions, their solution is not a generic one for ICDTNs. This is also hinted by the fact that their solution is based on DSDV, a MANET protocol which cannot operate in highly mobile and opportunistic DTNs.

The authors of [109] proposes PIFP, an Interest forwarding protocol for the CCN/NDN architecture which is inspired by PRoPHET. The authors' main proposal is that the Interest forwarding procedure should be based on the probability of each node "meeting" the particular content object being requested. Considering a connection between two nodes A and B, if node B often comes into contact with a certain content object, then that node is a good node to forward Interests from node A for that content object.

The main assumption made by PIFP, is that if a node often "meets" a content object, then it has a high likelihood of "meeting" again with that content object in the short term. This likelihood is reflected in a metric called Delivery Predictability, which is maintained for each content object and takes into account a) the *frequency*

of encounters between nodes and content objects and b) the *freshness* of information - essentially, the frequency of information updates. To decrease the number of Interests being forwarded, a node forwards an Interest packet only to nodes with a higher Delivery Predictability than itself.

To achieve that operation, several modifications are made to the NDN message processing operations. The most significant is perhaps the one related to the actions performed on incoming Interests. Like regular NDN, PIFP first checks if the content exists in the Content Store and returns the content if available locally. However, in case there is no match in the Content Store, the Forwarding Information Base is also checked for routing entries. Only if this second match is also true, denoting that the node has knowledge of that particular content, is the unsatisfied Interest stored in the Pending Interest Table. Another important modification is the fact that the authors disregard entirely the concept of Faces, which they argue is irrelevant to DTNs. Instead, they use the node IDs as Faces, which would either require an actual NDN/CCNx implementation to create of a separate Face for each encountered node (a requirement which may cause implementation issues, considering the device resources which may be needed to support per-node faces in large networks), or the removal of an entire - central - component of the NDN/CCNx architecture.

Other modifications include a change in FIB to also store Delivery Predictability values for each object, as well as the exchange during a contact of a) digests which announce the cached content names and b) a summary of the FIB information which includes the corresponding Delivery Predictabilities. Notable is also the fact that the authors deal with the 'breadcrumbs' problem by allowing PIPF to create multiple Interest forwarding paths.

The authors also include a second variation of the algorithm, PIFP-Proactive, which introduces even more explicit modifications in NDN/CCNx to allow the reception of Data packets by nodes which have never received the corresponding Interest packets (allowing for 'unsolicited' Data). This allows the utilization of all available storage space and the spread of multiple content replicas to achieve proactive content delivery, with the tradeoff of increasing the network load.

The evaluation of PIFP is performed using a modified version of the ONE simulation - ICONE - which implements a simplified version of NDN . During the evaluation experiments, a comparison is performed with STCR [110] an ICN protocol for delay-tolerant MANETs.

Other solutions such as the MobCCN protocol [111] [112], focus on efficient access to data generated by IoT devices in an environment where mobile opportunistic networks are formed. The main goal of MobCCN is to implement an opportunistic routing protocol which populates the FIB of CCN-capable mobile devices to guide Interest packets towards node holding the requested data. Its primary focus is to enable data access in the same area that the data is generated, in contrast to data retrieval from remote locations.

To achieve its goal, MobCCN proposes a routing and forwarding scheme to enable opportunistic networking in CCN. To this end, it introduces some modifications to the original CCN forwarding and routing mechanisms which the authors claim that can be compatible with the original architecture, as the main CCN structures and procedures are not altered.

The main idea is that the original FIB entry establishment in CCN is not suitable for OppNets due to the fact that those next hops may not be relevant for subsequent packets due to the high node mobility. To counter this, the authors propose that FIB tables are enhanced with auxiliary information about the suitability of each node to forward Interest packets towards devices capable of serving the requested data. To

achieve this, MobCCN groups content into a number of different types and collects statistics regarding node encounters with each data type. Based on this, it calculates node utility values (which reflect a node's suitability as a forwarder for a specific data type) and creates a gradient-based graph which is used in the forwarding and routing process.

Each node calculates two values for each content type: a direct utility (based on its inter-contact time with nodes providing that content type) and an indirect utility (based on the respective utility values of other nodes not directly serving the content, as well as its inter-contact time with those nodes). The overall utility of the node for that content type is defined as the maximum of those two values (direct and indirect utility).

Utility values between nodes are exchanged using Hello messages and stored in a) the FIB tables of each node and b) an additional data structure, the "Current Neighbours Utilities" (CNU). An interesting feature of the MobCCN approach is that the typical CCN/NDN breadcrumbs routing seems to be maintained and Data packets are transferred back to the content requesters along a single route. By propagating and storing additional information about each node's forwarding suitability, Interest and Data packets are not flooded in the network, nor are multiple paths being created to assist data delivery, avoiding the excessive replication used in Epidemic-based approaches.

The authors evaluate the efficiency of their approach using the OMNeT++ simulator and benchmark their approach against various Epidemic-based routing approaches. The results show that a high delivery ratio can be achieved by MobCCN with a low overhead, at the cost of an acceptable increase in end-to-end delay. Based on the results, the authors conclude that the additional traffic required to build efficient FIBs can result in the creation of more efficient forwarding paths, without the introduction of excessive overhead (which was a frequent assumption in earlier works).

The DIRESC protocol [113] employs social cooperation to achieve content announcement, discvery and retrieval in CCN-based opportunistic networks. The overall goal of the authors is to achieve adequate performance by choosing appropriate relay nodes, without resorting to one of the two extreme strategies followed by other works: either choosing an Epidemic approach which floods the network with packets (requests or content) and causes unnecessary overhead, or resorting to a single-copy strategy which may limiting data delivery to a single path (potentially interrupted by network disruptions). The proposed forwarding mechanism proposes that data are sent in a distributed way, based on each node's local profile of social interactions with other network nodes.

This is achieved by defining a local social-tie value which represents the closeness of a pair of nodes based on their history of contacts (contrary to other similar approaches which utilize such relationships on a global scale, as measured e.g. by node centrality values). This relationship is leveraged to decrease the distance between requesters and the data being requested.

The protocol operation is split into several phases: content announcement, discovery and recovery (i.e., retrieval). During each one, different algorithms and procedures for content exchange and packet processing are defined. The nodes encountered may be part of some major categories, which are crucial for subsequent operations in those DIRESC phases. Those categories are content providers, content requesters, cluster nodes and friend nodes. Content providers and requesters are self-explanatory categories. Friend nodes are determined according to whether their

social-tie to the current node exceeds a certain value, while cluster nodes are composed of those nodes which received prefix announcements (and, thus, defined during the announcement phase). In any case, when two nodes meet, they exchange the summary vectors of all their major CCN components (Content Store, Pending Interest Table and Forwarding Information Base) to disseminate all information needed for DIRESC operation.

The protocol's performance is evaluated using the ONE simulator and compared against three routing protocols which are able to operate in CCN-based Disruption-Tolerant Networks: CCN broadcasting [20], STCR [110] and PIFP [109]. The results show DIRESC exhibiting consistently better behavior than all the aforementioned protocols, although the performance difference to PIFP is less prominent than to the other two. This could be expected, as CCN broadcast and STCR do not constitute solutions designed for opportunistic networking environments (such as the ones utilized in during the evaluation).

DIRESC is a detailed solution, but does not avoid the modification of CCN message processing pipelines and is not compatible with standard CCN forwarders. Therefore, it would not be operable in hybrid wired/wireless networks which would include standard CCN routers in the fixed part of the network. Neither can be deployed globally - both in the wired and wireless domains - as a unified CCN-based routing protocol. Therefore, even though it seems like a promising solution for locally-formed opportunistic wireless networks, it is orthogonal to the current dissertation which aims to form a solution for wired-to-wireless communication (and vice-versa). In this respect, it is similar to MobCCN.

Contrary to the previous approaches, other works focus on the utilization of data mules to enable communication between remote areas (or between those remote areas and the Internet). Typically, well-connected local networks are formed on those areas. Those local networks can be either mobile ad hoc networks, wireless sensor networks, or network fragments of a previously well-connected core network in disaster cases [114]. Whatever the case, connectivity between those areas is intermittent and, therefore, those works focus on solving inter-area communication problems (instead of intra-area ones) by deploying data mules which transfer information from one area to another. Gateways may be deployed to handle communication on behalf of the other nodes.

One such work proposes Decentralized Interest Counter Aggregation for ICN (DICAI) [115], a mechanism to facilitate Interest prioritization by data mules in a disaster scenario. Although DICAI is abstract and generic enough to be applied to any ICN architecture which uses request and response-type packets, the authors use CCN/NDN semantics to describe their protocol operation.

Their main goal is develop a mechanism which allows data mules to estimate the popularity of content in a decentralized manner, such as the one encountered in a disaster scenario. Subsequent message exchanges (i.e. forwarding, scheduling or prioritization algorithms) which can be performed based on the estimated popularities, are out of scope of this work.

The main idea of DICAI is to use a [nonce, counter] tuple for Interest aggregation, which is assigned to end-users by data mules and subsequently exchanged when two data mules meet. Based on those values, the authors define all specific cases which may occur and how aggregation is performed on each case. In any case, when two mules encounter each other and exchange that information, the two mules aggregate that information and subsequently use either a) the sum of the two counters or b) the largest of the two counters. To avoid loops which may lead to

overestimation of content popularity, the ID of each mule is stored in a memory list and checked in subsequent encounters.

The evaluation results, obtained using an analytical model as well as a custom simulator, indicate that content popularity can be predicted with acceptable accuracy. Unfortunately, the authors do not implement their solution on top of CCN/NDN and, therefore, some implementation issues remain open from an architectural perspective (e.g. the way that memory lists are implemented, which could be either as a separate face for each node, an ID stored on an application running on a mule etc.) and the corresponding tradeoffs of each choice are not quantified. Furthermore, DICAI is far from a complete solution for CCN-based OppNets, as data forwarding and routing operations are not considered in this work.

The authors of [116] present a different perspective: they approach the problem of data dissemination in DTNs as a dataset synchronization problem. Their work is inspired by the fact that data propagation in Epidemic Routing traces its roots the Epidemic Algorithms [117], which were originally developed for synchronization of updates between distributed databases.

Thus, they argue that NDN's synchronization protocols - particularly, DDSN [118] - can provide a (better) alternative to Epidemic Routing in DTNs. They posit that NDN's features such as in-network, node-level, storage and data-centric design which natively supports broadcast environments, makes it easier to support DTNs and - in fact - provide a much more solid base for data exchange in these environments.

To this end, they present a detailed comparison between the features of the two protocols (Epidemic Routing and DDSN) and proceed to evaluate their performance on the ns3 simulator. The delay and evaluation metrics were used for comparison and in all cases, DDSN outperforms Epidemic Routing.

Even though DDSN seems well-suited for environments involving minor-to-moderate connectivity disruptions such as MANETs, it is not clear whether its performance would still be acceptable in highly mobile DTNs. Epidemic Routing was one of the first routing algorithms proposed for DTNs and much more capable solutions have been developed since, particularly for mobile opportunistic networks. DDSN most probably would not be suitable for highly mobile opportunistic networks, as its design does not integrate any features found in more advanced such protocols (e.g. social-based forwarding) and the only comparison provided is with Epidemic Routing.

When considering data transfers between intermittently-connected remote areas, DDSN is more suitable for intra-area and area-to-mule data synchronization (for which it can improve the respective delay and overhead), than for selection of the mules most capable to deliver the data (i.e. to increase the data delivery probability for data transferred between those areas).

Only a handful of previous works specifically explore an NDN/DTN protocol coexistence. One of those is [119], in which the authors propose a middle layer approach for the integration between the two protocols. This work is based on two main ideas: the definition of this ICDTN middle layer and the concept of geo-region faces.

Regarding the middle layer, this approach is presented as an intermediate one between "high" and "low" integration approaches. The core concept is to have the NDN layer "direct" the DTN layer towards the direction in which packets should be forwarded, while at the same time the DTN layer performs the disruption handling functionality which is needed by NDN. In the authors' words, NDN provides the "next leap", while DTN provides the "next hop".

The purported benefits of this approach is that all nodes can support in-network caching (contrary to "low" integration approaches, in which only specific nodes have that functionality), while at the same time, the ICN and DTN protocols are left completely intact and can be replaced with different protocols as needed (contrary to "high" integration approaches, in which one or both protocols are modified).

Regarding their geo-region approach for face construction, the authors first discuss the trade-offs associated with different types of faces and proceeds to propose the use of "geo-region faces" to construct their ICDTN middle layer. Those faces represent geographical regions, as opposed to representing geographical directions ("geo-faces") or specific nodes[12], which are two other possible choices.

On a more concrete level, the idea is that there is a separate "face layer", which exists below NDN and above DTN. This layer can autonomously decide whether or not to forward Interest and Data packets "up" to NDN. In practice, those packets are not forwarded to the upper NDN layer if they are destined for a different region than the one the node is currently located. To make this information available to the middle "face" layer, NDN uses to/from region tags which are included in the Interest and Data packets.

The authors evaluate the performance of their approach using the ONE simulator, on top of which some minimal NDN functionality has been implemented. They perform two sets of experiments: First, they compare the performance of their approach in DTNs to a baseline non-DTN scenario for different underlay DTN routing protocols (Binary Spray-and-Wait and two versions of Epidemic). Secondly, they compare their "medium-integration" approach based on "geo-region faces" with a "high-integration" one. To implement the second one, they substitute the "geo-region faces" in ONE with "per-node faces"; essentially, they replace an existing middle layer with another one. The results show that the middle layer can achieve a better Interest Satisfaction Ratio and Delivery Time when compared to a high-integration node-face approach, with the trade-off of an increased network overhead (as measured by the *Transmissions per Interest/Data* metric).

Although the approach presented is novel and - compared to the "high integration" approach - avoids heavy modification of either architecture, some modification is still needed and - therefore - compatibility with vanilla NDN nodes is an open question. Unfortunately, this question is not answered in the paper, as the authors have not implemented their architecture in real-world devices. A design choice which hints at incompatibility is the separate "face layer" being proposed, which means that the vanilla NDN architecture needs to be altered on a structural level (by moving faces outside the main NDN architecture and data structures). This additional layer may also contribute to an increased overhead, which is observed in the simulations but may be worse in a real-world implementation.

Furthermore, there are also some additional requirements posed by this design. For instance, the forwarding/routing space should be split into well-defined regions. This split should be performed in a globally-homogeneous way, as the individual choice of different schemes by each node would result in routing malfunction. Several choices exist with different semantics and degrees of granularity (e.g. a region could either correspond to a country, a prefecture, an urban area, a building etc.) and the definition of such a scheme is not addressed. Furthermore, as each node is associated with a region, the NDN implementation needs access to a node's location information. While this may be an acceptable requirement in other scenarios (e.g. vehicular networks, in which navigation is based on GPS information), the work is

[12]Referred to as "per-node faces" in this dissertation

not oriented towards VANETs but more general wireless mobile DTNs which often include mobile handheld devices (i.e. smartphones). In this case, privacy is often a concern and users may not want to share location information.

Last, as nodes are mobile, they are expected to change regions more or less frequently. This means that they become associated with different regions and - due to the use of geo-region faces - their faces will change and FIB tables will become invalid. As FIB maintenance is complicated in this situation, the authors choose the solution of having the nodes drop their FIB tables and any routing messages they are carrying when they move from one region to another. When they come again into contact with another node in that region, they obtain their own FIB and PIT tables (which have been constructed based on information related to the new region). Nodes with pending Interests, need to re-express them.

The same line of work is continued on another paper [120], which focuses on the aspect of large-scale data advertisements in ICDTNs. The authors depart from their previous assumption that data advertisement is simply solved by flooding advertisements, as well as from previous works in ICDTNs which either assume that this function is performed in an infrastructure-based environment, or by flooding Interest requests (but without specifying how nodes are informed about the existence of such data).

The authors' goal is to close this gap using network coding techniques, which they claim can improve data advertisement. To do so, the authors use Random Linear Network Coding, a distributed network coding technique which does not rely on network topology information. Their motivation was the similarity of the data advertisement problem to the multi-source broadcast problem, in which network coding has been shown to reduce the number of transmissions. To this end, they address some of the specific issues which arise when applying network coding in an ICDTN environment.

They evaluate their approach on the ONE simulator, in two distinct experiments: The first one compares their approach (based on "node faces") with a simple flooding technique, in a simulated non-DTN and a DTN environment (with a static topology and network disruptions). The second experiment introduces node movement and studies the scalability of their approach (which, in this experiment, is based on "geo-region faces" to support mobility). In both cases, the key take is that network coding can improve data advertisement in cases when users need to acquire more than a specific percentage of announcements. In cases when they only need a small percentage to be functional, flooding may be preferable as the coding/decoding time introduces some trade-offs. The difference between the two experiments is that in the second one, the crossover happens at a larger percentage (therefore, flooding may be preferable in more occasions) as the introduction of geo-region faces requires that nodes drop both their FIB tables and decoding matrix and have to reconstruct these tables when changing regions. Furthermore, the decoding time is also much higher in mobile scenarios, as the network can experience regular partitions and nodes which receive an encoded packet may move out of range and become unavailable for large periods of time.

Last, there also exist some other designs for NDN-based OppNets which, however, are preliminary and include no performance evaluation of the proposed solutions. One such example is [121], which explores the deployment of NDN data mules without any additional DTN layer. The authors use the vanilla NDN implementation and exploit the incidental exchange of regular peers in an NDN network to carry incremental upgrades of content available in different locations. Their approach enables disruption tolerance through the use of: a) unsolicited Data packets,

the reception and caching of which is allowed for NDN data mules and b) retransmission of Interest packets each time an NDN mule connects to an island, using the NDN forwarding strategies.

An important component which is used in this approach is a local NDN repository (repo-ng), in which local users upload data which are to be shared with other islands. This repository is periodically probed by remote islands to pull data. To allow delivery of Data packets by different mules, the authors introduce a subscription service which re-transmits any remote Interests to enable Data muling in the Content Stores of mules other than the original. Interest satisfaction is monitored using the watched prefix insertion protocol[13] between subscriptions, which can also be used to indicate whether subsequent Data objects should also be transmitted. The authors' main contribution is this local subscription service, which re-expresses the Interest packets on behalf of a remote area and - when combined with the permission of unsolicited Data packets - can reduce the data delivery delay by allowing Data delivery using alternate mules.

A downside of this work is the fact that the permission of unsolicited Data packets in NDN is a rather controversial issue: in fact, it violates NDN's receiver-initiated, pull-type communication model which was based on the experience of sender- initiated push-type communications creating security issues in IP networks (e.g. DDoS attacks).

Another such work is NDN-Opp [122], a framework which aims to bring Named Data Networking to Opportunistic Networks. Its architecture is comprised of a) an NDN Opportunistic Daemon, which includes all NFD functionality in addition to Long-Lived Interests[14] and pushed Data[15], b) an Opportunistic Face, which implements a queuing system to support intermittent connections, c) an Opportunistic Channel and Opportunistic Receiver, which implement the packet transmission and reception logic for lower-level channels, d) the Opportunistic Face Manager, which manages the RIB to attach discovered name prefixes to Opportunistic Faces (corresponding to known peers) and e) The Opportunistic Connectivity Manager, which maintains the channels for communication with other nodes.

As the solution uses NDN semantics and inherits all of NDN structures and tables, the two implementations are generally compatible. However, some of the mechanisms used in its operation are features not by-default used in NDN (e.g. Long-Lived Interests and pushed Data) and, as a result, compatibility with vanilla NDN forwarders would most likely not come out-of-the-box.

2.4 Trust and Security in Named Data Networking

2.4.1 Security Issues

In this section, we first present the inherent security vulnerabilities of the NDN architecture and, subsequently, introduce the most common novel types of attacks.

Attack Surface

The team behind NDN has placed a particular focus on creating a secure internetworking architecture by design, contrary to TCP/IP in which security mechanisms

[13] https://redmine.named-data.net/projects/repo-ng/wiki/Watched_Prefix_Insertion_Protocol

[14] i.e., Interests not removed from the PIT when the corresponding Data is received

[15] i.e., Data packets which do not adhere to reverse-path routing but use FIB tables for delivery

were an afterthought and are sometimes characterized as "patches" to the original architecture. NDN achieves this by following radically different approach to network security.

Its main premise lies on the concept of securing data directly, instead of securing the channels that transfer that data. To achieve this, NDN mandates that every data object (i.e., Data packet) is signed by its Producer. This signature is generated using the Producer's cryptographic key and carried by the packet. The fact that the content and its name are signed together creates an effective and secure binding between the two. This way, the content of every Data packet can be verified and - if needed - confidential. Name semantics allow applications to reason about trust and security, automate cryptographic key usage and decouple the trustworthiness of a data object from the specifics of how and where it was obtained from [47].

In addition, the pull-based design of NDN (and other ICN architectures) can inherently provide protection against some common network attacks which plague IP networks. An example are DDoS attacks, which can be avoided in NDN due to fact that communications are receiver-initiated; meaning that Data cannot be sent to a node unless an Interest has been previously issued by a Consumer for that content. Furthermore, if multiple Interest packets for the same content arrive at a router, only the initial one is forwarded and the subsequent ones are suppressed. This feature is called Interest suppression and not only reduces traffic and core bandwidth usage, but can also help to mitigate Interest-based attacks.

Nevertheless, despite the efforts to address issues which are present in today's TCP/IP Interent and the integration of security into NDN's design from the beginning, the architecture is not free from security problems. Many of those problems stem from the fact that the NDN architecture essentially aims to improve networking performance by placing additional storage and processing requirements at the network layer, through the incorporation of features such as in-network caches and a stateful forwarding plane. To support this functionality, NDN routers include data structures not present in IP, such as the Pending Interest Table (PIT) and Content Store (CS).

The CS and PIT enable in-network caching and aggregation of closely spaced interests for popular content, which - as already discussed - are key features of the NDN architecture. Thus, even though those structures enhance system performance (by reducing the overall latency and improving bandwidth utilization for popular content), they also increase the architecture's attack surface and make it susceptible to new forms of attacks. In those new types of attacks, the adversary aims at exactly those components and/or features.

Moreover, despite every NDN content object being signed, signature verification is only obligatory for NDN Consumers. NDN routers, on the contrary, are not required to do so. Three are the main reasons for this: (1) the computational cost of in-network cryptographic operations such as signature verification, can be considered prohibitive to achieve at line speed (at least for high-speed networks), (2) additional overhead is introduced to fetch the associated certificate chain, and (3) there is no unified trust model in NDN [123]. This aspect can also be exploited by potential attackers to target router devices, precisely due to the difficulty to achieve line-speed, in-network signature verification.

Of the aforementioned reasons which lead to routers not verifying content signatures. perhaps the most important one is that signature verification is a heavy cryptographic operation - more so than signature generation [124] [125]. At the time of writing this thesis, NDN offers two signature algorithms by default, RSA

and ECDSA. Based on prior results, some authors [126] argue that stretching existing state-of-the-art signatures (and their computationally expensive cryptographic primitives, such as modular exponentiations or elliptic curve point multiplications) to perform wire-speed verification over line-rates in the order of 10 or 100 Gbps is hard. Other works [127] have shown that an optimized software implementation of RSA1024 signature verification running on Intel Core 2 Duo 2.53 GHz CPU allows a router to verify about 150 Mbps of traffic, assuming 1500 Bytes per content packet, or even worse with smaller-sized packets. As a result, the authors conclude that NDN routers with multiple Gigabit-speed interfaces would need an unrealistic amount of computing power to verify signatures of Data packets at wire speed.

Another relevant study [128] which focused on the per-packet verification cost, arrived at the conclusion that the default NDN signature verification algorithm requires 0.0839 ms for verification of a 1000 bytes content packet on an Intel Core 2 1.83 GHz processor, using MD5 as the message-digest algorithm, RSA1024 as the asymmetric cryptographic algorithm, and DES-XEX3/CTR as the symmetric cryptographic algorithm. This amounts to 11,918 packet signature decryptions per second, which is far from meeting Gbps forwarding requirements. Other authors [124] calculated even worse average times for signature verification (ranging from 1.274 ms for the RSA algorithm, to 6.869 ms using Ring signature) on an Intel R CoreTM i7-7700 CPU @3.6 GHz, 16 GB RAM desktop computer. In the latter study, the packet size was not specified.

In addition, a different work [129] showed that the performance impact by signature verification may be high even on end-user client nodes for certain NDN implementation. In this work, the authors concluded that the throughput decreased by 5 to 20 times on web browsers running NDN.JS when signature verification was enabled. The severity of the decrease is attributed to the fact that JavaScript is not optimized for computation-intensive tasks, which makes signature verification the bottleneck in content processing.

Possible solutions to the problem have been proposed, such as equipping routers with Bloom Filters so that they mainly perform lookup operations, or probabilistic signature verification which has been proposed for access control [130] [131]. These techniques could bypass signature verification for each Data packet by all routers (e.g., by storing the namespaces of identified attackers), but additional issues remain.

Besides the excessive cost of signature verification, routers would need to fetch multiple public key certificates in order to trust the public key that verifies a content signature, which increases the overhead significantly. Furthermore, each content-producing NDN application can independently specify the trust model it utilizes. Therefore, there are diverse trust policies defined by different applications at the same time. This means that each router should know the specific trust schema used by each NDN application. Since there is a wide variety of applications in a network, this requirement entails considerable trust-related complexity for routers.

As involving routers in the specifics of trust management is cumbersome, routers have little defense means to mitigate attacks. In this light, we argue that alternative, more lightweight solutions are needed to cope with attacks and improve the security of the NDN architecture.

Attack Vectors

In this subsection, we provide a brief overview of the new types of attacks in Named Data Networking. Those attacks are made capable by the exploitation of features mentioned in the previous section.

Interest Flooding Attacks: One new attack which is introduced in NDN, is the Interest Flooding Attack (IFA). IFAs target the PIT of NDN nodes, a structure which tracks the state of the Interest packets that have been received but not yet answered. The PIT is also involved in the Data packet breadcrumbs routing process, as on the arrival of a Data packet it is forwarded to all the interfaces listed in the corresponding PIT entry. Therefore, it is an important component of the architecture which - if successfully attacked - can severely degrade its performance.

When an Interest Flooding Attack is launched, an adversary floods a victim's PIT with Interests for non-existent content. Those Interests fill up the router's PIT and their trails remain there until their expiration time. Additionally, they can also cause excessive network bandwidth usage and place additional load on Producer nodes - if the attacker achieves Interest propagation inside the network up to the Producer nodes.

As a consequence of the network overload this attack can cause, legitimate Interests issued by honest users may be dropped and the overall user Quality of Service can be degraded. For this reason, Interest Flooding Attacks are considered as a major security problem in NDN [132] [133].

The most popular countermeasures for IFAs involve some form of pushback mechanism [133]. However, researchers have gone as far as proposing to eliminate the PIT structure entirely from NDN in order to mitigate such attacks [134].

Content/Cache Poisoning Attacks: Another attack type consists of attacks which target the content itself. The goal is to have the NDN routers forward and cache corrupted or fake content, which - in turn - prevents the Consumers from retrieving legitimate content. As can be inferred from the above, those attacks frequently target the NDN router devices which are especially vulnerable in NDN.

The most general type of such attacks are Content/Cache Poisoning Attacks (CPAs). Content Poisoning and Pollution Attacks predate the NDN architecture and were also common in earlier content-addressable systems which did not include in-network caching, such as Peer-to-Peer content sharing networks [135][136].

In NDN, Content Poisoning is intertwined with Cache Poisoning as the content will inevitably be inserted in the router caches. For this reason, the two attacks are treated in a joint manner in the literature. This dissertation follows the same approach. Thus, Content/Cache Poisoning Attacks will hereby be both indiscriminately referred to as CPAs.

In CPAs, the goal is that Consumers receive fake or corrupted content. The authors of [127] initially consider a Data packet can be considered as *corrupted* if its signature is invalid and as *fake* if its signature is valid but generated with the wrong private key. While the authors of [137] (some of which are shared with the previous work) consider any content injected by attackers as *fake*, which is defined as any content object which has: a) an invalid signature, b) a valid signature which is generated (signed) with the wrong key or c) a signature field that is somehow mal-formed. As there is no strict consensus on the definition, we will mostly use the term malicious content when referring to content injected to the network by attackers, regardless of its form.

Cache-related Attacks: Some variations of this attack type are distinguished from the more general case, based on their specific focus on compromising the routers' cache (Content Store). Three variations of cache-related attacks are Cache Pollution

Attacks and Cache Poisoning Attacks, while there are also exist some less popular attacks such Cache Snooping Attacks.

The difference between *Cache Poisoning Attacks* and *Cache Pollution Attacks* is the following: When it comes to cache poisoning attacks, the adversary anticipates genuine interests for a valid content name and injects fake content under the same name into router caches. As poisonous content travels back to the issuer of the Interest, it is cached by the routers along the path. To make things worse, subsequent Interests for the same content are satisfied by the infected CS and the fake content is spread to the network.

In Cache Pollution Attacks, the attacker targets the router's cache as was the case in the previous attack. However, this time, an attacker sends multiple requests for legitimate - yet unpopular - content and misleads the routers into filling up their caches with this unpopular content. This violation of content locality in the caches results in cache misses for benign users and increased network traffic, as legitimate Interests have to be forwarded to the network as the cannot be answered by the router caches.

Finally, another cache-related attack consists of *Cache Privacy Attacks*, also referred to as *Cache Snooping Attacks*. On a Cache Snooping Attack, the adversary's goal is to gain complete knowledge of a cache's content. To achieve this, the attacker sends multiple Interests for various name prefixes and analyzes the time needed for the Interests to be satisfied. This attack usually targets edge routers to which a limited number of consumers is attached. Whenever the attacker succeeds, the content that the router's adjacent consumers are interested in, can be inferred. Therefore, the consumer's privacy is being violated by the attacker.

Bitrate Oscillation Attacks: Attacks can also target other protocols deployed on top of NDN. Such an example is DASH [138], a dynamic bitrate streaming protocol which was originally developed for HTTP-based streaming in IP networks but has - in recent years - also been deployed on top of NDN [139].

DASH involves a client-based mechanism which requests video chunks by name, a design that is well-suited with NDN. An important aspect of DASH are Media Presentations. A Media Presentation is a structured collection of encoded data of some media content, e.g., a movie. Media Presentations can be ultimately broken down into individual Media Segments, which represent units that can be uniquely referenced by an HTTP-URL. A Media Presentation is described in a Media Presentation Description (MPD) file, an XML file which contains information about the accessible Media Segments and their timing [140].

Unlike HTTP-based multimedia streaming, in DAS over NDN the MPD file lists the NDN names (i.e., Uniform Resource Identifier (URI)) for the segments as URL alternatives [141]. In addition, the hierarchical naming structure of NDN explicitly supports versioning and segmentation which indicates various representations of DASH-based multimedia content. Interest packets are issued to retrieve Media Segments, which are retrieved either from the original content source or router caches.

However, despite the fact that NDN can improve the performance of adaptive multimedia streaming by reducing bandwidth and offering a higher user Quality of Experience (QoE) [142] [143] [144], it also creates new security vulnerabilities for DAS. NDN's in-network caching and Interest aggregation features can be exploited by an adversary to manipulate DAS's adaptive behavior and degrade users' perceived QoE.

In particular, it has been well-documented [139] that caching introduces oscillatory dynamics and modify the estimation of the available bandwidth between the

end user and the repository from which the user is retrieving the chunks. If an edge cache holds a low resolution representation near the user, the user retrieving those low-resolution chunks will observe good performance and request higher resolution chunks. If, in turn, those chunks are hosted on a poor-performing server, the client would subsequently switch back to the low representation. This oscillation can be detrimental to the perceived QoE of the user.

Those dynamics are exploited by a new attack type, Bitrate Oscillation Attacks (BOAs) [145], which are made possible in DAS-over-NDN and can be launched during video streaming. In BOAs, an adversary proactively requests certain Media Segments so that they are cached in intermediate routers. When a legitimate user tries to retrieve the multimedia content, DAS clients experience redundant bitrate oscillations during video streaming and eventually, users perceive a degraded QoE [146].

Traditional NDN security schemes fail to safeguard the network from BOAs. Although some mitigation methods have been proposed, they either do not prevent the attack [145], or require continued network monitoring which introduces additional signalling overhead [147].

In this dissertation, we focus on the development of efficient mitigation methods for two of the aforementioned attacks: Content Poisoning Attacks and Bitrate Oscillation Attacks. Specific details regarding the attack models used in the experiments and the proposed mitigation methods are reserved for Chapter 4.

2.4.2 Potential Solutions

Reputation systems

One of the definitions of trust is “the extent to which one party is willing to participate in a given action with a given partner, considering the risks and incentives involved” [148]. A trust decision is based on the balance between trust and risk, and has an effect on the trustee.

Named Data Networking (NDN) [3], the most prominent ICN architecture, currently utilizes trust mechanisms that are based on cryptographic signatures and certificates. This approach is called credential-based [149] or policy-based [150] trust. It be considered as a “strong and crisp” approach, where decisions are founded on logical rules and verifiable properties encoded in digital credentials.

However, there exists another major approach to trust management, a “soft and social approach”, based on reputation measures gathered and shared by a distributed community. This model is called “reputation-based trust”. In this approach, when agents decide whether to trust another agent or not, they rely on evidence of past behavior supplied by trusted sources, as opposed to credentials.

In this dissertation, we argue that important trust and security problems which plague NDN can be solved by introducing reputation in the architecture as a general trust establishment method. Reputation can create an alternative to computationally-heavy cryptographic operations such as signature generation and verification, which has proven to cause bottlenecks for in-network processing (especially in routers, which operate on wire speed) and create security problems.

We do not posit that "soft" trust based on reputation should entirely replace the current NDN "hard" trust establishment methods. Rather, our vision is that the two approaches co-exist and complement each other. Our case is that reputation-based trust should also be considered as an equally viable solution for achieving trust in NDN in use cases where conventional trust fails. To this end, we hereby provide

a general introduction to reputation systems, before focusing on reputation-based schemes specifically designed for NDN in Section 2.4.3.

In reputation systems, users assign ratings to service or resource providers. Those ratings represent a judgment of their direct interactions' quality. Eventually, trust is computed from the aggregation of ratings concerning local experiences, taking into account the feedback that is being provided by other network entities. A prominent example of a reputation-based system implementation is electronic commerce. E-markets, such as eBay and Amazon, have adopted reputation as a means for trust enforcement as well as an incentive to encourage transactions with unknown providers. On these platforms, reputation management is achieved through a centralized framework. This requires the existence of a trusted third party which collects the ratings and computes the final trust scores.

The initial deployment of trust management in commercial fields has triggered the extension to research in peer-to-peer (P2P) networking and multi-agent systems (MAS). The main feature of these networks that necessitates the existence of a mechanism for trust establishment, is the need of peers to identify if the intentions of other agents are reliable or malicious [151][152]. Contrary to online reputation models, the decentralized design of peer-to-peer networks and multi-agent systems requires a distributed approach. PeerTrust [153] proposes a distributed mechanism aiming to effectively evaluate the trustworthiness of peers and to detect various malicious behaviors in a P2P eCommerce community.

At the same time, trust establishment and management has been an issue in Mobile Ad-hoc Networks (MANETs) and Wireless Sensor Networks (WSNs). In MANETs, multiple trust management schemes have been designed to address different goals, namely secure routing, authentication, intrusion detection, access control, key management and trust evidence [154]. More specifically, reputation-based approaches have been typically used to secure routing, while credential-based approaches are preferred for the rest of the applicability domains. Characteristic of MANETs, is the fact that many trust-based protocols for secure routing derive trustworthiness from network-layer behavior and hence, networking parameters such as throughput or overhead are considered trust metrics.

Another research field in which reputation was applied, are WSNs [155]. Similar to MANETs, trustworthiness in WSNs has been established either by considering networking aspects (e.g., throughput, delay etc.) or by using typical reputation- and policy-based approaches. However, the implementation of reputation systems in those environments is difficult, as the models require a reputation exchange protocol which introduces significant communication overhead and increases energy consumption in resource-constrained WSNs. Solutions have nevertheless been proposed, such as piggybacking reputation values on the routing packets. In any case, the trade-offs between the desired level of trust and resource availability in a WSN demand that protocol design is tailored to the specific application requirements.

There are various design options for reputation systems, which depend on the specifics of their implementation domain. While analyzing the use cases in which the reputation-based scheme will be deployed, three basic dimensions need to be examined to form a comprehensive framework. Those dimensions are *formulation*, *calculation* and *dissemination* [156].

The formulation dimension describes the mathematical model and input for the assessment of reputation values. Formulation includes two main aspects: the reputation measure and the mathematical model used to aggregate ratings. Reputation can be measured using binary, discrete, continuous, string or vector type of values.

The aggregation model can be a simple summation or average of ratings, discrete, fuzzy logic, flow-based or probabilistic [157].

Regarding reputation calculation and dissemination, the calculation dimension addresses the practical design and implementation of the reputation algorithm. The dissemination dimension focuses on the mechanisms used for distribution and storage of ratings and reputation values among entities within the system. Based on classifications [158] and measures [157] for reputation-based schemes, we highlight below the general design options which are most relevant for NDN (divided between primary and secondary ones). A more elaborate discussion on the implications of each of those choices for NDN-based reputation systems is provided in Section 2.4.3.

The *primary* design options include:

- *Centralized versus Decentralized*: A primary design decision regards the process through which reputation is being calculated and/or distributed in the network. Based on this, trust models are divided into centralized or decentralized ones. In the centralized case, a trusted central authority undertakes the responsibility to collect the ratings, compute the reputation values, store and announce them to network entities. Although this method limits computational complexity, since there is no need for every node to evaluate reputation, there is a trade-off with the introduced overhead for ratings' and reputation values' exchange. Another important drawback of this approach is that the single trusted entity represents a single point of failure. Once this is compromised, the entire network operation can fail.

  On the other hand, there are decentralized approaches, whereby ratings are calculated by each node independently and then may be (optionally) distributed between the nodes by a dissemination protocol, in a P2P fashion. This approach eliminates the single point of failure problem, but requires that the individual nodes performing the calculations and providing the reputation values can be trusted, as well. Blockchain can provide a solution to these problems, as it offers both the trustworthiness a central authority would (e.g., by recording ratings in the ledger, making them immutable and verifiable), combined with the distributed announcement of reputation values.

- *Transaction-based versus Opinion-based trust*: The former relies on objective information derived from transactions. The latter, opinion-based trust, is based on the subjective opinion of an entity for another based on their past interactions. Although opinion-based mechanisms come with a lower overhead, they are more vulnerable to attacks. As a result, transaction-based schemes are preferred in security-related domains while opinion-based are most suitable when the main objective is data quality assessment.

- *Global reputation versus Local reputation*: Reputation can either be seen as a global property shared by all, or as a subjective property assessed locally by each entity. The localized approach is privacy-preserving and more suitable for decentralized environments. However, the global approach produces more accurate results as the calculation uses a more sufficient, representative amount of ratings.

- *Rank versus Threshold based schemes*: Rank-based approaches return a relative ranking of all entities' reputation. In contrast, a threshold-based approach excludes entities which do not meet the desired criteria.

- *Incentive- versus Rule-based control*: A rule-based system forces entities to follow its rules (e.g., rating according to specified parameters). An incentive-based ones, motivates entities to follow desired behaviors (e.g., provide ratings)

Some *secondary* design options include:

- *Data Filtering versus Data Aging*: Data filtering favors a subset of the ratings by assigning a higher weight factor. This selected subset includes ratings based on measures, for example time freshness. Data aging assigns an aging factor to ratings. As information gets older, its weight in the calculation diminishes, and eventually it gets discarded. Assigning a higher weight factor to latest measurements can allow for faster detection of misbehaviors. The main drawback of these approaches is that they introduce complexity in reputation calculation.

- *Collection*: It refers to the difference between information sources taken into account for the calculation of ratings and reputation values. Direct experiences and witness information are typical information sources used by reputation-based models [154]. These options have to be compared in the case of a localized scheme, since a global approach requires sharing the complete information range among entities.

Distributed Ledger Technology and useful work reward systems

Blockchain and Distributed Ledger Technology (DLT)[16], have seen a huge surge in popularity in recent years. This popularity is reflected both market-wise, with a boom in new products and startups, as well as research-wise, with a surge in publications in the area.

Blockchain technology was initially designed to support cryptocurrencies such as Bitcoin [159]. Its main innovation was a technique to create a distributed ledger which could solve the "double spending" problem, one that had previously plagued decentralized payment systems and hindered their adoption.

However, it soon became clear that the same technology (i.e., the blockchain) could become a general building block for distributed ledgers and did not have to be confined to a single application area such as payments. New blockchains emerged and solutions such as Ethereum [160] widened its use by enabling the general programming of decentralized applications through the use of smart contracts and a Turing-complete scripting language.

This has led to its application in various domains to build secure and trustworthy systems, an instance of which includes efforts to provide Internet-wide decentralized trust and security mechanisms. Many works have focused on leveraging blockchain to decentralize existing "hard trust" mechanisms of which the current Internet infrastructure is comprised, to avoid single-points of failure and censorship. One example is Namecoin [161] which allows the design of decentralized namespaces and was designed to build a decentralized Domain Name Service (DNS). Another one is Certcoin [162], which was designed to build a decentralized Public Key Infrastructure (PKI). Similar goals were also pursued by Blockstack [163], a system using the Bitcoin blockchain.[17]

However, blockchain can also be used to build decentralized "soft trust" mechanisms. As a matter of fact, Distributed Reputation Management Systems have been

[16]The two terms do not generally equate: DLT is the general concept, while blockchain technology is a specific form of a distributed ledger. However, they are often used interchangeably in the literature.

[17]Later renamed into "Stacks".

proposed to provide trust and security since the early days of P2P networks [164] - long before the appearance of bitcoin. In P2P networks, distributed reputation systems focus on choosing reliable resources, ensuring honest peer behavior and rating the quality of a shared file's content. Those properties has prompted their use to mitigate content pollution/poisoning [165][166] - a recurring problem which is not unique to NDN but also plagued Peer-to-Peer networks [135][136].

Despite reputation systems being successfully deployed in distributed environments to mitigate attacks, they can - perhaps ironically - be themselves prone to other kinds of attacks [167].

This is why DLT and Blockchain have been suggested as a means to secure and consolidate reputation systems. As a result, various blockchain-based Distributed Trust and Reputation Systems have been recently proposed [168], a direction explored in the context of this dissertation.

At the same time, blockchain has also been proposed as a way to create incentives for users to perform useful work, such as providing storage space [32] or computing power [169]. On a high level, the users of those systems form P2P networks to complete a specific task.

Such systems require automatic task verification to ensure that rewards are only received when the tasks are truly completed. To address cases where task completion is hard to verify (e.g. file transfer between untrusted parties), reward systems were developed to incentivise users.

Motivated by these approaches, in this dissertation we consider blockchain to be a building block for decentralized reputation systems and use it as a means to provide incentives for honest behavior in a peer-to-peer environment.

We argue that there are important benefits to this approach: On one hand, the use of blockchain as the basis for our reputation system enables notarization: all user activity, including each rating, is stored on the ledger and the users participating in the P2P network can have a transparent, agreed-upon, view of other users' past behavior and calculate their reputation values accordingly. On the other hand, the inclusion of a useful work reward system can provide users with incentives for honest behavior, which in turn contributes to proactive attack mitigation. Therefore, we hereafter present the blockchain-based reward system which was used in this dissertation.

The Proof-of-Prestige reward system :

Recently, blockchain has been identified as a building block towards a shared economy. To this end, cryptocurrencies have been leveraged to incentivize users for performing useful work. Filecoin [32] and Golem [169] are two such instances: the former rewards miners for renting storage capacity, while the latter rewards them for rental of computational power. The overall goal is a decentralized system, in which miners are incentivized for performing useful work, securing the transactions and automatically receiving rewards when the tasks are completed.

Many such systems assume two main actors: "contributors" and "beneficiaries". A beneficiary submits a task, as well as a reward for the specific task. The reward is used to ensure that the contributors will be paid for executing the task. When the contributor completes the task, the aforementioned payment is unlocked. For instance, the task may be video retrieval. In this case, the beneficiary would submit a request for a video in the system and a contributor would send the video, in exchange for the reward (e.g. a certain amount of money or tokens).

Proof-of-Prestige embraces an approach which adds a third actor to the system, to ensure that users can freely access the service. In the opposite case, the system

would not be competitive to the current cloud-based, free-access platforms (e.g. YouTube for the video delivery case).

This third actor is called the "motivator". The motivator can benefit from an increase in the network's popularity and – for this reason - rewards the contributors for their useful work, to keep the access free for the end users. One way the motivator can profit by doing so, is by running advertisements in the platform, which provide more revenue than the rewards which would be handed to the contributors. A platform which also follows this approach is Steem [170], a "social blockchain" which incentivizes users to upload content in the platform by rewarding the creators of the most popular content.

However, the verification of task completion is not equally easy for all tasks and depends on their nature. For instance, solutions for file storage exist, but not for file transfer between two untrusted parties. In the latter case, the only available source of truth are beneficiary acknowledgements, which makes systems created for file transfer susceptible to attacks (such as Sybil and Collusion attacks). [18]

Proof-of-Prestige [171] was created with those cases in mind. It assumes the existence of the aforementioned three actors, namely Beneficiaries, Contributors and Motivators. Each user is tied to an identity and can assume either one of the aforementioned roles (or more than one). None of the actors can be trusted and they may, at any time, try to tamper protocol messages or orchestrate attacks – solely or collaboratively - to enlarge their profit.

Proof-of-Prestige assumes that each user account is associated with two values: *Coins* and *Prestige*. Coins function the way cryptocurrencies do in typical blockchains, in which each user owns a wallet and this wallet is associated with a specific number of coins. The same is true for Proof-of-Prestige. Coins are associated with user accounts and can be sent or received directly by other users, when transactions take place.

Where Proof-of-Prestige differs from cryptocurrency-oriented blockchains is its addition of a second value to each user: Prestige. A user's Prestige value determines that user's probability of minting a new block. It cannot be transferred between accounts directly, as is the case with coins but is provided in exchange for another user's useful work. This way, PoP enables credit-less rewards, i.e. rewards which do not necessitate the transfer of virtual currency per se, but instead increase a contributor's chance of mining new blocks.

Coins and prestige values are not independent of each other. Instead, one influences the other in the following way: On one hand, coins generate prestige over time. On the other hand, a user's probability of mining a new block is determined by their prestige (and is actually proportional to it), which can be compared to how stake functions in Proof-of-Stake consensus. But, again, when a new block is mined, users get rewarded with new coins and may collect transaction fees.

In general, the number of coins remains fairly constant in time, unless they are sent/received during transactions or new ones are mined. Prestige does not follow this pattern, as it is a much more volatile resource. It is also renewable, as it regenerates over time. It can be obtained by performing useful work for other users, or depleted when used to profit from another user's useful work. Thus, greater prestige values allow users to essentially benefit from more services in a certain amount of time.

[18] Sybil attacks occur when users create fake identities, which can be used to falsely claim that another identity contributed with useful work and claim rewards that the user is not actually entitled to. Collusion attacks work in a similar manner, but this time the identities are all real and belong to different users. Those users, however, collaborate to maximize their rewards.

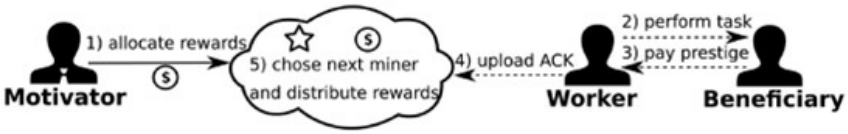


FIGURE 2.6: Proof-of-Prestige overview.

At this point, it needs to be stated that Proof-of-Prestige was not designed as a new consensus protocol, despite its (probably misleading) name and the fact that its design has been obviously inspired by Proof-of-Stake consensus. Its primary goal is to provide a reward system for useful work, which uses blockchain as its primary building block. Thus, due to its requirement for an underlying blockchain, it can be thought as a Layer-2 system. One main aspect which differentiates Proof-of-Prestige from Proof-of-Stake consensus protocols, is that they approach rich users in a different manner. Proof-of-Stake gives rich users more power by design, as in PoS each user's number of coins (i.e., their Stake) solely determine the miners of next blocks and, as a result, allows them to reap more rewards. In other words, it is by-design easier that "the rich get richer" in PoS blockchains.

However, PoP allows poor users to perform useful work and increasing their prestige. This, in turn, increases their probability of mining new coins regardless of whether they initially entered the system with a low amount of coins. As a result, even though 'rich' users are still able to buy more coins and attain higher initial prestige values, 'poor' users with a low amount of initial coins are not excluded as they can still perform useful work and accumulate prestige.

The PoP reward system can be implemented on top of blockchains utilizing different consensus protocols: either directly on top of a Proof-of-Stake blockchain [19], or as a smart contract on top of Proof-of-Work blockchains. Currently, two implementations exist: natively in Cosmos [172] (PoS) and on top of Ethereum (currently PoW). Furthermore, a Python3 simulator exists for large-scale experiments.

Operation

First, motivators allocate rewards in the form of coins. This allocation is fairly flexible, as rewards can be granted either for a specific task or in a global manner. The amount of prestige which will be exchanged for each task, is negotiated before the actual work between the contributor and beneficiary. Thus, prestige represents some form of transaction fee for the task.

Subsequently, the contributor performs the task and - after the task has been completed – receives an acknowledgment from the beneficiary. This acknowledgment confirms that the prestige transfer has taken place. To solve the issue of beneficiaries potentially not acknowledging the transactions (to avoid paying for the service), PoP incorporates a modified version of FairSwap, a fair exchange protocol for decentralized cryptocurrencies. However, different solutions can be chosen, based on the nature of the specific tasks.

After receiving an acknowledgment, the contributors periodically upload them to the blockchain to prove that they provided a service to other users. Based on those acknowledgments, the users' prestige values are updated. Each user's prestige is taken into account to select the miner of the next block. Once chosen, the miner receives the rewards allocated by motivators, in addition to typical transaction fees and new coins.

The main actors and operations are depicted in Figure 2.6, retrieved from [171].

[19] Excluding specific PoS blockchains which require users to lock their coins for a long period of time

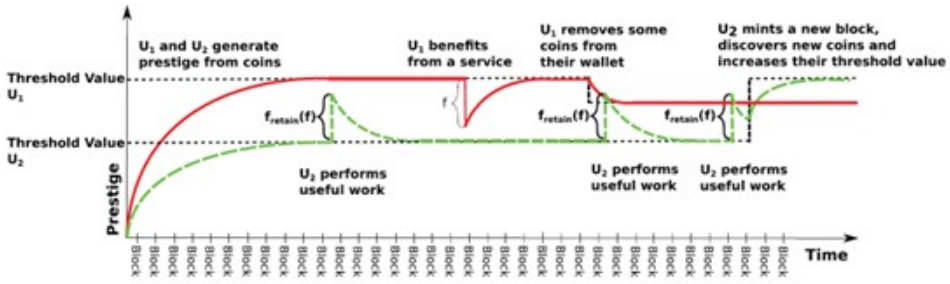


FIGURE 2.7: Example of prestige and threshold value evolution over time (expressed in blocks).

When a user joints the system, his/her initial prestige is zero. Each time a new block is mined, each user's prestige is increased by the number of coins included in the user's wallet. To avoid a behavior in which a user's owned coins increase their prestige indefinitely despite no actual work being performed, a decay factor is introduced in the system. This decay factor determines the percentage of the user's prestige which is lost as each new block is being generated.

Following this design, prestige can only grow until it reaches a certain value, at which point the decay evens the prestige generated from coins. This value is called the "Threshold Value" and is proportional to the number of coins a user owns. The only way a user can increase its prestige above this value, is by performing useful work which is acknowledged by the beneficiary. Thus, users are incentivized to continuously contribute useful work to the network.

It should be noted that when a user pays for a service with a prestige transfer, his/her prestige will slowly regenerate unless the user has no coins. As prestige is also lost when its value is higher than the Threshold Value (due to the decay), the entire network can be considered a closed-loop system which tends to "push" user prestige towards the Threshold Value.

An indicative example which depicts the evolution of prestige over time, incorporating the effect of prestige decay and regeneration, as well as prestige transfer events, is depicted in Figure 2.7, retrieved from [171].

Simple and Progressive mining

Two different mining schemes are designed for Proof-of-Prestige: Simple Mining and Progressive Mining. Each one is tailored to different use cases.

Simple Mining is the simplest form of PoP's mining algorithm and can be used to reward users for transactions (i.e., tasks performed) which are unique between one contributor and one beneficiary. One such instance is the renting of a contributor's computational resources to a beneficiary. In this mode, when a contributor uploads an acknowledgment to the blockchain he/she preserves all of the transferred prestige.

Progressive Mining is a more complex form of mining which is provided by PoP. Progressive Mining is designed for cases in which the act of benefiting from a service allows the beneficiary to subsequently perform useful work for other users. Such an instance is the act of seeding a file in a content distribution system after its initial reception. In this scenario, a user will first have acted as a beneficiary, then as a contributor to the peer-to-peer network. In this case, a file may traverse several users acting in both roles until reaching the "final" beneficiary. Thus, the task performed is no longer limited between a single contributor and a single beneficiary as was the case in Simple Mining.

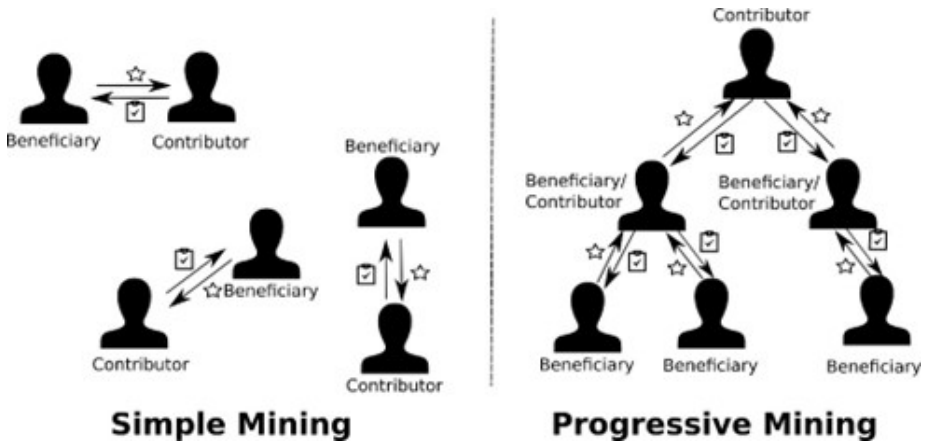


FIGURE 2.8: Simple and Progressive Mining.

In Progressive Mining, contributors are not only rewarded for the work they perform themselves, but also for the work of their beneficiaries. It can be envisioned as a Directed Acyclic Graph (DAG), in which the nodes of the graph represent its users, while the edges represent the work performed for other users. Prestige flows from the DAG leaves to the root. When a contributor performs a task for a beneficiary, the prestige is not instantly added to the contributor's account but must be shared with its DAG predecessors. Thus, intermediary nodes only retain a fraction of the total transferred prestige, which depends on each node's prestige and the total prestige of its predecessors.

The two mining schemes can be better understood by observing Figure 2.8, which has been retrieved from [171] and depicts both Simple Mining and Progressive Mining.

One important property of PoP is that both of its mining schemes are designed in such a way that users cannot increase their prestige gain (and, thus, mining power) by spreading their coins over several artificial identities, neither can they increase their total prestige by colluding with other users. Thus, both schemes are resistant to Sybil Attacks, as well as to Collusion Attacks.

2.4.3 Trust Establishment

Original Approach

As already stated, the NDN security and trust framework is based upon public-key cryptography. It can achieve data authenticity, integrity and – optionally – confidentiality regardless of what the underlying communication channels are.

In this section, NDN applications and other communication participants will be referred to as *entities*, while certified names will be called *identities*, following the terminology used in [47].

The original NDN trust management framework is based on the premise that each entity owns one or more names and proves its ownership using an NDN certificate, which provides a binding between the name and the entity's cryptographic public-private key pair. Each entity can also delegate sub-namespaces to other entities and issue certificates for those sub-namespaces. This way, a chain of trust can be created. An example depicting a certificate chain is depicted in Figure 2.9 (retrieved from [47]). Details are explained later in the section.

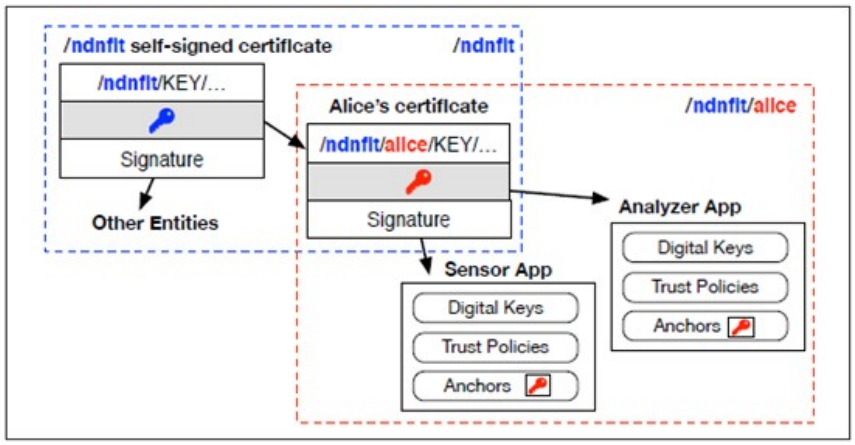


FIGURE 2.9: An example of the relationship formed between different namespaces as defined by the corresponding certificates.

The NDN trust and security framework is built upon three main components: digital keys, certificates and trust policies. Regarding digital keys, the only thing of note is that NDN considers them no different than any other content and they can be retrieved directly at the network layer.

Certificates are similar: they are Data packets which carry a public key and can be retrieved as any other Data packet. They contain their issuer's signing key name, along with other additional information into their signature info field. What they essentially represent is an endorsement of the binding between the name and the public key by their issuer.

A certificate name is not arbitrary. It must follow a specific naming convention: "/<prefix>/KEY/<key-id>/<issuer-info>/<version>". The "prefix" component denotes the name the key is bound to. The rest of its components are mostly self-explanatory: "key-id" is the id of the key, "issuer-info" contains information about the issuer, while "version" includes the version number of the certificate.

To make certificate naming conventions clearer, assume a certificate name such as /duth/intersyslab/bob/KEY/002/N-testbed/004. In this case, the certificate owner is /duth/intersyslab/bob. The certified key's id is 002, used to identify an instance of Bob's public key. The certificate signer has set the issuer information to N-testbed to indicate it is a certificate issued by the NDN testbed. The certificate version is 004.

The third component of the NDN security framework are trust policies. Trust policies are used to specify which entities can be trusted for producing which piece of data, as well as which key should be used for which namespace and what purpose. Several policies can be defined; for instance, it may be required that encryption keys cannot be signed by the same key used for data authentication. Entities can retrieve certificates and learn trust policies from trustworthy parties.

Those three components are used in specific ways to form the overall NDN trust and security framework. Its overall goal is centered around achieving data-centric security in the form of (mainly) data authenticity and integrity. Before providing details on how the latter are achieved, we are going to first focus on some of the challenges which arose out of the framework's design.

To create a usable overall framework, solutions in several areas were devised: This included the establishment of trust anchors, the design of effective certificate management solutions, the efficient incorporation of fine-grained trust policies, as well as the creation of usable key management mechanisms. Significant effort has

also been devoted to security bootstrapping, the process which is used by an entity to learn about trust policies and obtain its trust anchor and certificate.

Hereafter, we present an overview of the solutions which were developed by the NDN team, beginning with trust anchor establishment.

Trust Anchors: Trust anchors are used by entities to verify other entities' authenticity. All cryptographic verifications must terminate at a trust anchor, which is expected to either be pre-configured or securely obtained through some out-of-band means. NDN utilizes a different trust anchor model than the ones commonly used today. Instead of obtaining certificates from commercial certificate authorities or installing a single global trust anchor, NDN assumes that each system authority establishes its own trust anchor (or anchors). All entities under that authority can subsequently develop their own means to discover and obtain these trust anchors, or decide on using an existing means. This model is similar to Simple Distributed Security Infrastructure (SDSI/SPKI) [173] in this aspect. Once a trust anchor has been installed, an entity can verify the signatures of other entities by verifying their certificates along the certificate chain which leads to the trust anchor.

Certificate Management: An NDN certificate allows an entity to generate verifiable signatures for its data and build trust relationships with other entities. However, to generate valid Data packets, a Producer application must first obtain a name, as well as a certificate which certifies its ownership of that name. Consumer applications do not need to obtain identity certificates for Data consumption, although they must obtain trust anchors for data verification. Once the trust anchor is obtained, an entity can identify a trustworthy certificate signer by checking its certificate (e.g., a signer's certificate is the trust anchor or endorsed by the trust anchor), then request a certificate for itself.

NDN security offers flexibility to application developers on how to obtain certificates, according to the needs of each system (e.g. using a centralized service, or directly by a distributed application's users). The NDNCERT protocol [174] was designed to ease this process and allows for automatic processing of certificate requests.

Key management is also closely related to certificate management. It is crucial to NDN's operation as signing, verification, encryption, and decryption all involve the use of cryptographic keys. One of NDN's features is its use of structured, semantic names for content objects in NDN, which in turn allows developers to use naming conventions for the systematic construction of key and certificate names. This ability to construct the names of the required keys for a certain data name and retrieve them simplifies NDN key management (with respect to issuing and retrieving certificates).

Trust Policies: A trust policy denotes to an entity which cryptographic keys are legitimate to use for signature generation (when producing new Data packets) and verification (when receiving Data). NDN trust policies define the relationships between data names and signing key names. An application needs to obtain trust policies after obtaining the trust anchor.

NDN apps can define their trust policies using a trust schema, which is simply a piece of named content that can be retrieved like any other content. After obtaining the trust anchor, an application can fetch and verify the trust policies from trusted sources.

Note that there must be a preconfigured default trust policy, which can be used to validate the Data packets carrying trust policies. A simple default policy may define that Data packets carrying trust policies must be directly signed by a trust anchor with a given name.

Data Authenticity: After obtaining their certificates, an app can produce Data packets under its corresponding namespace(s) and sign them using its private keys. This allows consumers to authenticate the received Data packets.

One means through which NDN ensures Data authenticity, is packet signature verification. To verify a received Data packet's signature, an NDN Consumer retrieves its producer's certificate, which is identified by the key name in a specific packet field. This certificate recursively points to its signer's certificate and, eventually, arrives at a trust anchor. The Data packet is considered valid only if all the certificates in the above chain have valid signatures and satisfy the trust policies.

As can be inferred, the second means used to achieve data authenticity are trust policies. Trust policies leverage the fact that NDN uses structured namespaces and semantic names, which allows applications to reason about trust and define corresponding data acceptance policies.

The way policies help consumers validate a received Data packet is by defining rules based on which only packets signed by keys with specific names are accepted and subsequently checking whether the packet is signed by the correct key. This way, trust policies limit the power of each signing key to Data packets with specific names, supporting data authenticity at a fine granularity. For instance, the key /duth/intersyslab/bob/KEY/... can be limited to sign packets under the prefix /duth/intersyslab/bob only.

Once data acceptance policies have been defined by a user, NDN can subsequently confirm data authenticity in an automatic manner. This can be achieved using *trust schemas* [175], a solution developed to allow users/applications express their trust policies in a form that can be automatically executed by applications.

Trust schemas make use of NDN's naming conventions to enable systematic descriptions of trust policies, namely: how Data packet names should be structured, how packet signing key names should be structured, how the components in a Data packet name should be related to those in its signing key name, and which trust anchor is acceptable.

Upon receiving a Data packet (and before any signature verification takes place), a Consumer application first uses its trust schemas to assess the packet's trustworthiness by examining its certificate chain to the trust anchor.

Let us assume that both Alice and Bob use the same application and their certificates are signed by the same trust anchor. Each user device produces Data under its own prefix (e.g. /ndnfit/alice/sensor/example, /ndnfit/bob/sensor/example).

Figure 2.10 (retrieved from [47] depicts two different trust schemas and their effect on incoming Data packets. The first schema ("Rule 1") only accepts Data whose name prefix is /ndnfit/alice, its signing key name prefix is /ndnfit/alice/KEY, and its certificate chain arrives at the trust anchor /ndnfit/alice. This means that solely the packets which are signed by Alice and are under Alice's prefix will be accepted.

The second schema ("Rule 2") is not so strict and incorporates looser conditions. Data with the name and key name prefix /ndnfit, which include a certificate chain ending up at the anchor /ndnfit are accepted. Using the second schema, Data produced by either Alice or Bob will be accepted.

Reputation-based Trust Approaches

Although not the default approach used by NDN, reputation has also been investigated by some authors as a solution to NDN's trust and security issues. A general overview of reputation systems has been provided in Section 4. In this section, we

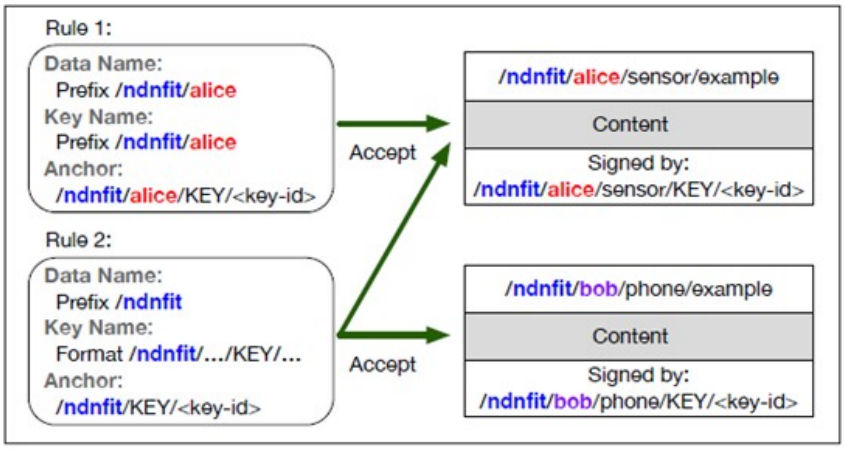


FIGURE 2.10: Data packet authentication using trust schemas. Different schemas lead to different results.

focus specifically on works which use reputation-based trust approaches to secure the Named Data Networking architecture.

As has already been noted previously, NDN routers and their caches are a frequent attack target precisely due to the architecture's use of credential-based trust mechanisms. As a result, most works which utilize reputation focus on securing router caches.

When a cache poisoning attack is detected, the compromised router should discard the forged content from its Content Store. NDN's inherent way to flush content from caches are cache replacement policies (e.g. LRU, LFU etc.). However, those policies rely on natural content aging and cannot discern between legitimate and poisoned content objects.

To solve this issue, Ghali et al. [137] have proposed a reputation-based technique which utilizes explicit exclusion filters to reduce the number of poisoned content objects in the caches.

This mechanism leverages an optional field which is included in Interest packets, the *Exclude* field. This field denotes specific name components which must not be included in the name of the returned Data packet and can be used to exclude specific content.

The main concept of the proposed mechanism is that Consumers (which verify Data signatures), can detect invalid content and issue a new Interest. This Interest uses the Exclude field to subsequently prevent the invalid content from being returned by Interest packets.

Routers rank the cached content based on the exclusion information provided by the Consumers, ranking legitimate content higher than the fake one. A content object's rank is based on several factors: the number of times a content was excluded, when it was excluded, as well as the number of interfaces which excluded this content.

Although this algorithm seems to achieve efficient mitigation of cache poisoning and delivery of valid content to consumers, it is still susceptible to these attacks. On the one hand, malicious end users can target a valid content object and collaborate to exclude it from caches. On the other hand, a group of malicious consumers can explicitly request fake content without excluding it. Therefore, poisonous content may still be prioritized by routers and serve subsequent interests, propagating into

the network. Another drawback of this method is that every content object, either valid or fake, will be cached, increasing memory cost.

The aforementioned vulnerabilities are considered important by other authors [176], which propose a different reputation-based method. Their approach also exploits Exclude filters and works by assigning a credibility value to each content provider, additionally to a trust value which is being assigned to each content object. A user's credibility value is calculated based on the trust value of the content that has been provided by that user.

When a Consumer issues an Interest excluding fake content, the content's trust value will be decreased, impacting the credibility of the sender, as well. Countermeasures have also been taken to prevent on-off attacks, in which adversaries alternate between sending valid and invalid content to maintain a level of credibility. In particular, the rate of decrease of the credibility value (when poisoned content is detected) is greater than the rate of increase (which occurs valid content objects are sent).

The calculated trust and credibility values are used by routers as criteria to accept Interests and perform selective caching. To be more specific, a router caches received content with a probability which is equal to its trust value. In addition, routers accept the incoming Interests with a probability which is equal to the credibility value of the Consumer who sent each Interest packet. The authors argue that this act of associating the probability to satisfy a Consumer's Interests with their credibility value, essentially incentivizes users to provide legitimate content.

Despite the fact that the aforementioned two mechanisms provided demonstrated the viability of reputation-based approaches for NDN attack mitigation, it should be noted that the exclusion functionality which was exploited by them is no longer available in NDN since version 0.3, rendering the previous methods rather obsolete (since they are based on a deprecated mechanism).

Cache poisoning attack mitigation is examined from a different angle in Router-Oriented Mitigation (ROM) [177]. ROM is based on the idea that content-oriented mitigation methods might be inadequate if the router itself tampers content passing through. In ROM, every router assigns a reputation value to each of its adjacent routers. Authors argue that the further a router is from an attacker, the more poisoned copies are received for a specific content piece because of multi-path propagation. Based on this assumption, they use the number of received poisoned copies of a certain content as a means to decide how much to decrease the reputation of routers that forward fake content.

Reputation values are updated based on negative verification results generated by Consumers and forwarded back to the routers following the reverse transmission path. This is similar to the previous approach [176].

The computed reputation value of a router represents how trusted this router can be and determines how likely it is for this router to be included in the transmission path. In other words, routers choose their well-reputed neighbors as next hops to forward Interests and thus, malicious routers are temporarily excluded from the transmission path.

ROM's efficiency is evaluated by comparing the mechanism with the Interest-Key Binding (IKB) rule [123], which states that an Interest must reflect the public key of the Producer. More specifically, IKB exploits an Interest field called PublisherPublicKeyDigest (PPKD). PPKD contains the SHA-256 digest of the publisher public key. According to this method, the public key of every received content is being hashed by routers and compared to the PPKD of their related PIT entry. The

content object is forwarded and cached by routers and its signature is verified by Consumers if and only if the hash and the PPKD match. Otherwise, it is discarded.

However, since this approach is credential-based, it has some specific shortcomings. A Consumer that issues an Interest has to fetch the PPKD of the desired content's provider in advance. In addition, the PPKD has to be verified at every hop. As these features increase the delay and reduce the network throughput, ROM outweighs IKB. In particular, evaluation results provided in [177] indicate that ROM improves network latency by 85.5% and throughput by 84%.

Moving away from content/cache poisoning, reputation has also been used to mitigate cache snooping, by allowing the detection of adversaries [178]. As explained by the authors of this mechanism, a snooper may not only request but also exclude content to make sure that he obtains a full picture of the cached content. Based on this assumption, a user is deemed malicious when high interest and exclusion rates are measured in a short time period, as well as a high cache hit rate from the local cache at the same time. These features are used in the calculation of a user's reputation and subsequently, in the evaluation of another metric, the user's trust value. The trust value is compared against a threshold in order to detect snoopers. Unfortunately, as was the case in the first two mechanisms, the use of the currently-deprecated exclusion filters poses an obstacle for wide NDN deployment.

Besides cache-related attacks, reputation-based schemes have also been used to mitigate Interest Flooding attacks. ICRP, an Interest flow control method based on user reputation and content name prefixes, [179] assigns a reputation value to each consumer which represents the transmission degree of Interests requiring existing content objects. Aiming at preventing on-off attacks, while calculating reputation values, ICRP weighs both the past and the current behavior of users. Computed reputation values are compared against a predefined threshold, and if a user's reputation is below this threshold, he/she is considered malicious. Upon the detection of a malicious consumer, its Interests are accepted according to its reputation value. The lower the reputation value, the higher the Interest drop rate. Moreover, ICRP observes and counts the Interests sent by detected attackers and creates a blacklist which includes non-existent name prefixes. Therefore, ICRP identifies the malicious users based on reputation values of consumers, while it detects and records non-existent content names and thus, limits the flow of malevolent Interests in the network.

Based on the evaluation results provided by the authors, reputation-based approaches have been shown as a realistic solution to efficiently mitigate cache- and PIT-centric attacks, capable to consolidate the existing credential-based NDN security schemes. Table 2.1 presents the efficiency of each work, based on the provided evaluation results.

In Section 4, we provided an overview of the general design options for reputation systems. Hereafter we categorize the aforementioned NDN-based reputation schemes with respect to the dimensions mentioned in that Section.

Content rating versus Network Entity rating: When formulating the reputation system, a basic dilemma in an NDN scheme is whether the feedback (i.e., ratings or reputation value) should be tied to the disseminated content itself or to the entity which forwards the content (e.g., initial producer or intermediate router). The policy-based approach in force, relates content objects to certificates. However, using a credential-based approach, consumers and intermediate routers cannot make any proactive assumptions regarding the trustworthiness of providers, or their future content quality.

TABLE 2.1: Efficiency comparison of reputation-based schemes

Scheme	Evaluation Metric	Results
Ghali et al. [137]	Percentage of benign consumers receiving valid content	100% (after 5–60 s)
Rezaeifar et al. [176]	False Positive Error Ratio (FPER) False Negative Error Ratio (FNER)	21%–25% 10%–14%
Wu et al. [177]	False Positive Error Ratio (FPER) False Negative Error Ratio (FNER)	<=4% 0%
Umeda et al. [179]	Average data acquisition rate	>70%

In a soft trust scheme, reputation values can be assigned to providers and function as a criterion to (proactively) avoid transactions with untrustworthy providers, or perform selective forwarding and/or caching. Alternatively, rating content itself can be useful for consumers or for detecting unexpected or malevolent behaviors (e.g. corrupted or fake data).

Even though rating content has been investigated as an option for either ranking already cached content [137] or to determine the probability of content to be cached [176], the majority of previous works assume that ratings are assigned to network entities (e.g., end-users [176] [178] [179] or routers [177]).

Transaction-based versus Opinion-based trust: Both options could be used by NDN-based systems, although transaction-based schemes are mostly preferred in security applications. A transaction-based system can be built in NDN by considering aspects such as data integrity verification and provenance authentication as trust metrics, since they are supported by NDN's default trust management scheme. Other networking parameters (e.g. delay) can also serve as a viable option.

Global reputation versus Local reputation: In NDN, global availability of reputation values can complement attack mitigation, (e.g., cache-poisoning). A local approach is more appropriate for assessing the quality of data. All the aforementioned NDN-based works propose decentralized mechanisms which compute trust locally. In addition, ratings are assigned based on information derived from completed transactions. In particular, some rely on exclusion information to provide ratings [137][176], while others quantify the reputation values by evaluating the number of received poisoned copies for certain content [177], or by taking into account measurements of Interest [179] and exclusion[178] rates.

Rank versus Threshold based schemes: Rank-based approaches return a relative ranking of all entities' reputation and have been used in NDN to prioritize content with higher quality or providers with higher reputation. Such an approach has been leveraged to rank valid over fake content [137] or to favor the most trustworthy routers as next hops during forwarding [177]. However, rank-based methods come with their drawbacks. For instance, high-ranked content is not always the best match for an Interest. In contrast, a threshold-based approach excludes entities which do not meet the desired criteria. Due to its strictness, it can be more efficient when leveraged to make forwarding and caching decisions [176],[179] or to detect an attack [178],[179].

TABLE 2.2: Classification of NDN-based works

	Ghali et al. [137]	**Rezaeifar et al. [176]**	**Wu et al. [177]**	**Ntuli et al. [178]**	**Umeda et al. [179]**
Content rating	*	*			
Entity rating		*	*	*	*
Centralized					
Decentralized	*	*	*	*	*
Transaction-based	*	*	*	*	*
Opinion-based					
Global reputation					
Local reputation	*	*	*	*	*
Rank-based	*		*		
Threshold-based		*		*	*
Rule-based	*		*	*	*
Incentive-based	*				

Incentive-based versus Rule-based control: Most NDN-based approaches follow specific rules [137] [177] [178] [179] follow specific rules. The only exception is [176], which associates a consumer's credibility with the probability that his/hers Interests are satisfied and exploits it as an incentive for users to provide valid content.

Table 2.2 provides a classification these works in a concise form.

Chapter 3

NDN over DTN

One of the main research goals of this dissertation, is to introduce support for delay and disruption tolerance in the Named Data Networking architecture. Towards this end, we designed NDN-over-DTN (NoD), a solution comprised by a protocol stack and an accompanying deployment strategy for intermittently-connected devices. Our contributions are the following:

- We first design the NoD protocol stack in the context of the UMOBILE architecture, to enhance NDN with a DTN tunneling option.
- We validate our design in an edge computing scenario which involves the use of data mules to transfer and deploy services in remote, challenged networks.
- We measure the impact of our previous design choices (namely, large packet sizes) on the NDN architecture, if adopted on a larger scale.
- We develop a mathematical model which integrates NDN and DTN metrics in a common framework which reflects the behavior of NoD.
- We deploy NoD in an IoT scenario and evaluate its performance using both fixed-route data mules and opportunistic contacts for data retrieval.

Specific details are presented in the following Sections. First, in Section 3.1, we delineate our main goals and motivation with respect to the NoD design process. In Section 3.2 we paint the general picture by providing an overview of the UMOBILE architecture, of which NoD is a major component. Section 3.3 offers a detailed presentation of the NoD protocol stack and deployment scheme. The next Sections (3.4.1 and 3.4.2) are concerned with its deployment in specific use cases: as an enabler for a) edge computing in challenged networks and b) content retrieval from intermittently-connected IoT devices, respectively. Finally, in Section 3.5, we present our mathematical model which captures the behavior of NoD with a high degree of accuracy (as demonstrated during the experiments[1]).

3.1 Motivation and design goals

The design of NoD was based on specific considerations, which are unfolded in the rest of the present section:

First, our aim was to create a solution interoperable with the original NDN architecture. We did not to arrive at an "isolate" solution which could only be deployed in a particular domain such as opportunistic wireless networks, but one which could be

[1] A description of the experimental evaluation process is reserved for Chapters (5-6).

easily integrated with other solutions being developed by the larger NDN research community (e.g. ones targeting wired networks or vertical to the network type).

Most of the existing work disregards this aspect and instead heavily modifies components to build custom features tailored to delay-/disruption-tolerant environments. Those solutions, however, cannot exchange messages with legacy NDN nodes deployed in wired networks, outside the solutions' target area. Therefore, to achieve interoperability, we had to introduce as little modifications as possible to the original NDN architecture (regarding its components, message processing operations, semantics etc.).

Second, our design had to solve the "breadcrumbs" routing problem which arises in NDN-based opportunistic mobile networks (while still maintaining interoperability with the original architecture). The "breadcrumbs" routing problem stems from the fact that NDN was initially designed to accommodate networks with seamless connectivity, small delays and more-or-less fixed network topologies.

To reduce routing complexity in well-connected networks, the use of symmetric forwarding paths for Interest and Data packets (or "breadcrumbs routing") was introduced in NDN. Its name stems from the fact that NDN Interest packets leave some form of trace in the routers' PIT while they are forwarded upstream to the content source. Those traces ("breadcrumbs") are subsequently followed by the corresponding Data packets to reach the source of the request (i.e. the NDN Consumer). This results in Interest-Data exchange sequences in which Data packets have to follow the exact same path that the corresponding Interest packets flowed through (albeit in the reverse direction).

However, the requirement for symmetric routing paths cannot be fulfilled in a highly volatile network topology. An example showcasing the inefficiency of the approach in wireless mobile networks is presented in Figure 3.1. In this example:

- A smartphone (*mob1*) initially comes within reach of a Consumer's wireless network and receives an Interest packet for content */ndn/data1* (image #1).
- *Mob1* subsequently moves out of the wireless network and eventually arrives at the wireless network of a Producer. *Mob1* has physically carried the Interest packet towards its destination - acting as a data ferry - and sends this Interest to the Producer upon arrival (image #2).
- The Producer responds to the Interest packet by sending a Data packet with the requested content to *mob1* (image #3).
- *Mob1* receives the Data packet but does not return to its previous location (towards the Consumer network), instead continuing its movement towards a different direction. By now, it is clear that mob1 will not physically carry the Data packet back to the Consumer. Nevertheless, a new smartphone (*mob2*) appears within the wireless reach of the NDN Producer. This device is perfectly capable of carrying the Data packet, but does not receive it as it has not previously received an Interest for that content name (image #4).
- *Mob2* heads towards the Consumer's wireless networks and eventually arrives there. It does not deliver any content, despite the fact that a delay/disruption-tolerant return path was created for the Data packet by the *mob2* device (image #5).

As illustrated by the previous example, NDN is - by default - either rendered entirely unable to operate in environments such as those covered by opportunistic

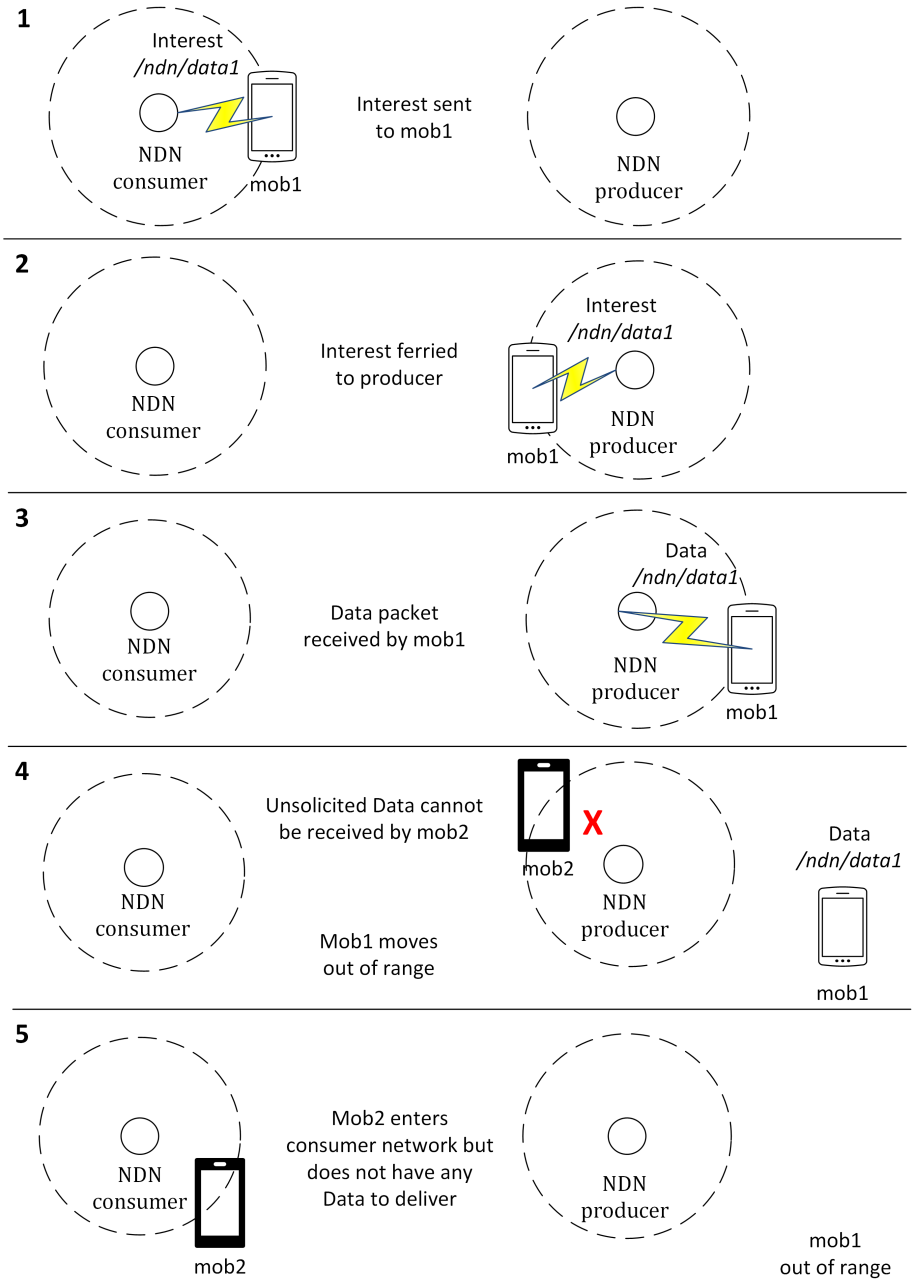


FIGURE 3.1: An illustration of NDN limitations in OppNets/DTNs, caused by its use of symmetric Interest-Data forwarding paths

mobile networks, or only exploits a small fraction of the communication opportunities (the ones involving symmetric paths). This is a major problem that needs to be tackled by any work which is concerned with NDN deployment in opportunistic wireless environments. Thus, we take this fact into consideration for the design of our solution.

A third consideration concerns the existing body of work on delay/disruption-tolerant networks and the ability to exploit any relevant solutions (e.g. algorithms and implementations). Several of the research works on delay/disruption tolerant Information-Centric Networks use an ICN architecture as a starting point, then proceed to build mechanisms for achieving delay/disruption tolerance from scratch. However, those solutions often end up mimicking the behavior of mechanisms and algorithms already developed in the context of DTN research. Our approach aims to readily leverage those features by utilizing DTN and its inherent mechanisms to handle delay/disruption tolerance in NDN.

Finally, another consideration was the deployability of our solution in mobile opportunistic and ad hoc networking environments. This aspect called for a design which took into account the characteristics of the devices constituting those environments, which have limited resources (i.e. smartphones and resource-contrained IoT devices). To effectively support network deployments in such environments, the NoD scheme does not necessitate deployment of the full protocol stack on all network devices indiscriminately; only specific nodes need to support the full stack.

3.2 The UMOBILE architecture

NDN-over-DTN was a major component of the UMOBILE architecture [180], an ICN architecture based on NDN. To provide the context of our NoD solution, we hereby present a general overview of UMOBILE.

UMOBILE is a universal, mobile-centric and opportunistic communications architecture. It extends NDN in various ways to support mobile communications at the network edge. Special focus is placed on localized communications and service deployment to improve the user Quality of Experience and alleviate congestion in the backbone network. Its goal is a tighter integration of opportunistic and delay-tolerant communications with the core of the network, to achieve universal coverage and enhance network resilience when the infrastructure is unavailable or impaired.

The motivation for UMOBILE came from the realization that there are areas where technical difficulties in communication are unavoidable; remote areas such as forests and mountains, or even places in the heart of the cities such as the subway. Similar problems exist in crowded areas (e.g. stadiums) and multiply in disaster cases, where the network might suffer from both fragmentation and extreme congestion, rendering the backbone inoperable. Connectivity problems also arise due to economic constraints, as mobile users may find it costly to use their provider's data services when roaming in a city, while sparse populations living in physically remote locations are plagued by the fact that it is not cost-effective for Internet Service Providers (ISPs) to install the required infrastructure.

Its ultimate purpose is to push content and services as close as possible to the edge. This way, aspects such as bandwidth utilization and resource management can be optimized and service availability in challenged network environments can be improved.

UMOBILE tackles those problems by changing how users communicate and access information, by moving from the traditional host-centric access paradigm to an

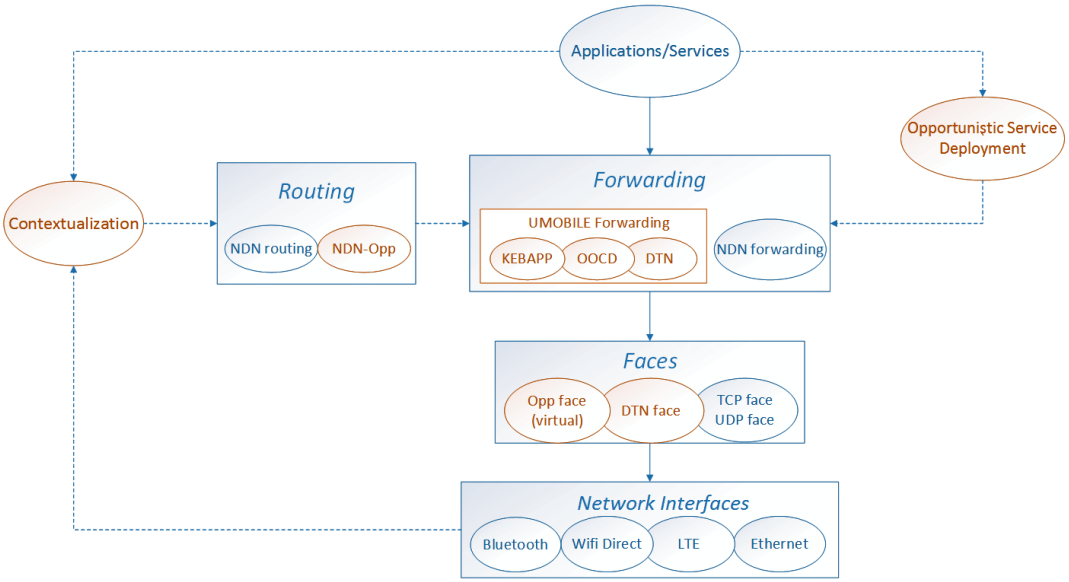


FIGURE 3.2: The UMOBILE architecture

information-centric model which adopts opportunistic and delay-tolerant networking primitives, cancelling the need of end-to-end connectivity and natively supporting localized communications. It also extends the content-centric, opportunistic and delay-tolerant networking concepts by incorporating computing aspects (as opposed to static content transfer/delivery). To this end, specific mechanisms are developed to exploit the huge amounts of computation available in edge devices.

On an architectural level, UMOBILE modifies and enhances the NDN architecture to enable support for the edge of the network (especially for mobile opportunistic wireless environments). To exploit all available communication opportunities, it utilizes an Ethernet interface in the fixed part of the network and Wi-Fi/Wi-Fi Direct and LTE interfaces in the mobile part. Its support of context awareness allows the network to adapt to different user behaviors by leveraging contextual information available through the Bluetooth and Wi-Fi/Wi-Fi Direct interfaces. New APIs are defined and QoS mechanisms are designed to enable different levels of service.

The architectural diagram of the UMOBILE platform is provided in Figure 3.2. Red modules depict the new components that have been developed in the context of UMOBILE, whereas blue ones depict the original NDN framework. As UMOBILE is a modular architecture, different parts of it have been deployed over Linux-based fixed network devices and Android-based mobile devices. In the rest of the Section, we present the new components introduced in UMOBILE.

Opportunistic Off-Path Content Discovery: Forwarding in UMOBILE is comprised by two parts: The first one consists of typical NDN forwarding strategies, such as Best route, Broadcast, Client control, NCC, etc. The second one consists of new forwarding mechanisms which enhance NDN. The latter includes the use of DTN tunneling for communication with intermittently-connected devices, as well as of OOCD [181] for content discovery in off-path caches. As the NDN-over-DTN protocol stack will be covered in Section 3.3, we reserve an extensive presentation for that particular Section.

Opportunistic Off-path Content Discovery (OOCD) was developed as a response to the fact that NDN, like most ICN architectures, has focused on scaling content resolution towards the main origin (or CDN surrogate server). OOCD argues that

this focus on "how to route requests towards the core of the network" goes against the original vision of a "native content distribution network" and prevents requests from discovering nearby content, unless the content is on the shortest path to the core of the network.

With a view to improving the content discovery capabilities of the NDN architecture, OOCD provides an enhancement to the routing fabric which keeps track of successful (i.e., served) content requests in a new, separate, table called "Downstream FIB" (D-FIB). D-FIB effectively acts as a FIB table that points to downstream nodes that have recently received (and therefore, cached) requested content. D-FIB uses trails left behind by data packets from the content origin to the sources in order to discover off-path cached content.

OOCD's opportunistic discovery mechanism can significantly increase cache hit rate compared to NDN's default forwarding strategy, while limiting the overhead at acceptable levels [181]. This not only creates a native content discovery and distribution system, but can also serve the purposes of communications in case of network fragmentation. When the path to the origin server is broken due to link or node failures, the D-FIB approach can discover off-path content which is stored or cached in end-user devices.

The KEBAPP Application Sharing Platform: The development of KEBAPP was motivated by the the expectation that mobile applications will follow mobile computing devices and become computation-heavy, bandwidth-hungry and latency-sensitive. KEBAPP introduces a new mobile computing paradigm to alleviate some of the network stress that mobile applications are already putting into the network, e.g., in case of crowded areas and events, where the mobile network effectively collapses.

Through the use of this framework, UMOBILE takes a step further from host-centric content sharing and focuses on the prevailing application-centric computation and communication model. KEBAPP [182] explicitly enables access to the desired processed and non-personalized information through the concept of application sharing, effectively leveraging on a pool of application resources. It effectively builds on application-centrism to facilitate information discovery through application-driven and application-defined, hierarchical namespaces. Given the ad-hoc nature of the proposed computation framework, KEBAPP further extends these namespaces by introducing the concept of keywords, i.e., free text or application-driven (GUI) parameters used to enable the invocation of applications at co-located mobile devices.

This enables the description, discovery and retrieval of processed information, further supporting results with varying degrees of accuracy (e.g., a search result that does not contain all search terms), instead of only exact matches. We note that the invocation of remote processing (in co-located smartphone or Wi-Fi AP devices) is central to KEBAPP, as opposed to previous work which has centered on retrieving static content from nearby devices. KEBAPP also manages connectivity in an application-centric way, i.e., coupling connectivity options and opportunities to applications and their namespaces.

With the introduction of KEBAPP, UMOBILE extends CCN/NDN to form a generic solution across different applications which can overcome the pitfalls of IP. Even if only a small percentage of users utilize KEBAPP for application sharing purposes, KEBAPP can achieve considerably low request response times [182].

Routing: Routing in UMOBILE extends the existing NDN options to support opportunistic wireless environments. To this end, a new routing module has been

developed to exploit such communication opportunities: NDN-Opp [122]. NDN-Opp handles the dynamics of opportunistic networks by forwarding interest packets towards neighbors with high probability to meet nodes carrying the interested data.

It implements both the standard NDN pull communication model, supporting data sharing applications such as Now@ [183], as well as a push communication model used by interactive applications, such as the short message application Oi![2]. To handle intermittent connectivity, NDN-Opp includes the concept of Opportunistic faces (Opp Face), which relies on Wi-Fi Direct for all device-to-device communications. An Opp Face can be in two states (ON and OFF), and is named after an identified neighbor. Since forwarded packets may not be immediately dispatched, Opp Faces implement two queues: a) Interest Queue, storing Interest packets to be sent and b) Data Queue storing pointers to the Data block (in the Content Store) to be sent.

The operation of an NDN-Opp node is similar to the NDN operation, as the node state is changed by the arrival of Interest or Data packets. Contrary to the faces used by NDN, Opp Faces create an indirection: NDN operation finishes with a packet being sent while NDN-Opp operation ends with a packet being stored in an Opp Face. Packets are transmitted by serving Interest and Data queues when Opp Faces get up.

NDN-Opp implements a novel forwarder and routing engine. Upon reception of an Interest packet, the NDN state machine is used, but the PIT also stores information related to the Interest duration (Long Lived Interests). Interest packets are forwarded based on the NDN best route forwarding strategy, as the cost of name prefixes (stored in the FIB) is related to social weights computed by a social-aware routing module. Data packets are forwarded following the typical NDN operation.

The social-aware routing engine computes the costs of using Opp Faces to reach data related to specific name prefixes. It can run any type of social-aware opportunistic routing algorithm. Computing routing information based on social interactions has great potential as less volatile proximity graphs are created.

A new information-centric opportunistic routing protocol was also developed to support opportunistic wireless communication. This protocol is DABBER [184] and incorporates data reachability metrics in the routing process by taking into account available information such as the availability and centrality of neighboring nodes, as well as the availability of the various data sources. Those metrics can be provided in UMOBILE using the Contextualization module, which is described in the following subsection.

Contextualization: A novel feature of UMOBILE is the incorporation of social awareness to improve data dissemination, using its contextual plane. Social awareness can exploit the mobility patterns of personal devices and leverage traffic locality to improve service/content delivery.

The UMOBILE contextual plane is realized by the Contextual Manager module, a service that runs in background on an end-user device or on an Access Point. The Contextual Manager captures information concerning the device affinity network (roaming patterns and peers over time and space) as well as usage habits and interests (internal device information). Costs derived from this contextualization process are then passed - upon demand or periodically - to other UMOBILE modules to assist in different network operational aspects.

[2]https://play.google.com/store/apps/details?id=pt.ulusofona.copelabs.oi

The Contextual Manager holds three different interfaces towards other modules (two interfaces for the routing module and one interface for native applications), allowing other modules or applications to query or obtain information from the Contextual Manager. The type of information that any Contextual Manager interface provides can be categorized into two main sets: i) affinity network characterization data; ii) usage and similarity characterization data. The affinity network information involves aspects such as the status of peers (as evolving over time and space) as well as the affinities (matches) between source nodes and peers. Several indicators are considered in the characterization of affinity networks and are provided via the different CM interfaces to other UMOBILE modules. Examples include peer lists, the Interests associated with each peer, as well as indicators of peer status (e.g., battery) and connectivity.

The indicators provided which concern usage and similarity characterization are built upon data collected internally in the device, taking into account user privacy. Examples of indicators that fit this category are, for instance, geo-location, as well as categories of preferred applications.

Push services: NDN natively supports the pull communication model, in which receivers are the ones who initiate the transfer of named content. This model is suitable for applications that involve the transfer of large and persistent content, but not for ones involving dissemination of frequently produced ephemeral contents. For this class of applications, the push communication model is more convenient. In this model, the producer initiates content transfer to consumers. UMOBILE enhances the NDN pull model with push-based communications, implementing three different variations.

The first one is *Interest polling*, suitable for applications in which the Consumer does not know when the content is made available by the Producer. It relies on a polling mechanism which allows the Consumer to issue periodic content requests which are received by the Producer. This way, a form of active "path" is created from the Consumer to the Producer and maintained, allowing the latter to push content to the former once it is available. The drawbacks of polling are that it induces network overhead and that the freshness of the received data depends on the polling frequency.

The second one is *Interest notification*, which is useful when the content is a piece of text that is small enough to be appended in an Interest name by the producer. In this variation the content is actually transferred by Interest packets, reversing their original semantics. Upon the reception of the Interest packet, the receiver can optionally send an ACK Data to acknowledge reception of the content.

The third one is *Publish data dissemination*, which allows for the transfer of content regardless of type and size. When the producer makes the content available, it sends a notification Interest to the consumer who replies with an Interest message requesting the content. The first chunk of the content is sent together with information about the remaining chunks which are transferred in subsequent exchanges.

Opportunistic Service Deployment: PiCasso [185], an edge computing platform, was developed in the context of UMOBILE as a form of QoS mechanism. PiCasso is used to transfer services to the edge and includes opportunistic service deployment as an application-level mechanism to overcome latency and availability constraints. This mechanism can reduce traffic at the core and enabling service deployment at remote regions where they would otherwise be unavailable.

PiCasso assumes a business model in which the network provider is in possession of network, computational and storage resources and services are offered

through virtualization. In this model, the content producer delegates the responsibility of deploying QoS-constrained services to the network provider.

PiCasso leverages the abstractions offered by NDN and takes advantage of monitoring and lightweight virtualization (i.e. dockerization) to deploy on-demand services at the edge of the network. UMOBILE APIs are exploited, such as the push-based communication model (publish data dissemination).

The Service Manager module implements all the necessary functionality a service provider needs to deploy its services. It includes a service repository (service repo) where dockerized compressed images ($service_1, \ldots, service_n$) of the services are stored. It also includes a Decision Engine that is responsible for decisions regarding what services to deploy, in which UMOBILE Hotspot. The Decision Engine makes its decisions based on monitoring information which is collected by the Monitoring component. This component uses pull requests to collect metrics about the resource consumption status of each monitor deployed in each UMOBILE Hotspot.

The Service Manager uses Push requests to transfer the dockerized service image to the UMOBILE Hotspot selected by the Decision Engine and instantiate it. The implementation benefits from the efficient data dissemination achieved by the UMOBILE architecture. For instance, if a service is already deployed in neighboring UMOBILE Hotspots, the UMOBILE Hotspot selected by the Decision Engine can opportunistically retrieve the service from a neighbor. PiCasso has proven to be a viable approach for lightweight service deployment at the network edge [185][186], even including remote areas [187].

3.3 The NoD protocol stack and deployment scheme

To enhance NDN with delay-/disruption-tolerant networking functionality, we designed NDN-over-DTN (NoD) by taking into account all the aforementioned considerations. In this dissertation, NoD refers to the NDN/DTN protocol stack accompanied by its deployment scheme.

NoD was initially deployed in the context of the UMOBILE architecture [180]. It was used for offering service access to intermittently-connected mobile users by transferring content and services to remote hotspots [187]. We subsequently extended its use and evaluated its performance in Internet of Things (IoT) scenarios, which focused on content retrieval from intermittently-connected Wireless Sensor Networks (WSNs) [188] [189].

In the following subsections, we first present the conceptual design of the NoD protocol stack, before proceeding to the logical design of our implementation. We subsequently present the deployment scheme which accompanies the protocol stack. Last, we present a second, modified version of NoD which exploits cross-layer information to reduce network overhead.

3.3.1 Conceptual design

To introduce delay and disruption tolerance in NDN, we depart from most of the previous work which developed new delay-tolerant NDN implementations by modifying the original architecture. Instead, we employ a design which leverages the DTN architecture for communication in intermittently-connected environments. This

way, the well-mature features offered by DTN for communication in challenged networks (such as the various DTN routing protocols, or the Custody Transfer hop-by-hop reliability mechanism[3]) can be readily leveraged.

To this end, we propose the use of a protocol stack which consists of two discrete layers: an NDN and a DTN layer. Each layer is assumed to handle communication in a different domain (well-connected and intermittently-connected networks, respectively) with its fine-tuned protocols, algorithms and mechanisms.

It is important to note that, originally, the DTN architecture was conceived as an overlay architecture which could unify different heterogeneous networks. This overlay is formed by the DTN Bundle Protocol which is located at the application layer of the OSI model and uses Convergence Layer Adapters to operate over different lower-level protocols (e.g. TCP, or even link-layer protocols).

However, in the current dissertation we propose a different design. Instead of using DTN as an overlay to NDN-based networks, we reverse the design and propose that an efficient solution can be formed by utilizing DTN as an underlay, effectively acting as a transport option for NDN. This DTN layer allows the creation of tunnels either to intermittently-connected individual devices, or between intermittently-connected NDN network fragments (such as those found in disaster events). Other transport options remain available to NDN for communication with well-connected networks (e.g. UDP/IP for overlay NDN deployment, or Ethernet for native network-layer deployment).

This is in line with the (few) other layered designs in the ICDTN literature, which follow the same pattern as, to the best of our knowledge, no previous work overlays DTN on top of NDN (or any other ICN architecture). The most probable reason for this is the fact that the Future Internet was conceived by those authors to be information-centric and, as such, most works used an ICN architecture as their basis.

In our case, the utilization of DTN as an NDN transport option allows intermittent connectivity to be handled by DTN, as the lower-level protocol. By abstracting this functionality from NDN, the need to develop elaborate higher-level mechanisms no longer exists. This fact not only reduces the complexity of the NDN layer but also assists interoperability with vanilla NDN, as heavy modifications of the NDN layer are no longer required to incorporate delay/disruption-tolerant features into the architecture.

Additionally, this design does not limit our approach to intermittently-connected networks, as our approach communication with vanilla NDN nodes deployed in well-connected fixed networks is feasible. Therefore, the well-mature DTN mechanisms for data ferrying and routing in challenged environments can be exploited when such need arises, while - at the same time - typical NDN operation continues at the fixed part of the network. As a result, NoD can form the basis for a unified internetworking solution which can operate across all network domains and exploit all communication opportunities.

Another advantage of utilizing an NDN-over-DTN protocol stack is that, when paired with an appropriate deployment scheme, it can solve the "breadcrumbs routing" problem. Previous work has tried to mitigate this implication by introducing modifications to the NDN architecture and its semantics (see Section 2.3). However, not only did those modifications impact the solutions' compatibility with the original architecture and limit their deployability to a single network area (i.e., local OppNets), but they also frequently ended up re-creating solutions which had been

[3]For more information, we refer the reader to Section 2.2

previously proposed by the DTN community and realized by existing DTN implementations (e.g. Custody Transfer-like functionality, or minor variations of existing opportunistic routing algorithms to make them ICN-compatible).

Moreover, certain solutions to this problem introduce issues in other areas. In particular, one solution to by-pass the requirement for symmetric paths is by allowing NDN nodes to accept any Data packet, regardless of whether they have previously forwarded an Interest which requests that content. Data packets received in this manner are called "unsolicited Data" and can be used to create different upstream and downstream paths. Even though the "breadcrumbs routing" problem can be solved this way, we do not embrace the use of unsolicited Data packets. The reason is that they pose an additional security risk and increase the architecture's attack surface, e.g. by enabling attacks which are otherwise mitigated by design (such as DDoS).

Based on the above, we posit that an NDN-over-DTN protocol stack is an effective solution to enhance NDN with delay/disruption tolerant functionality, solving the breadcrumbs routing problem while enabling compatibility with the original architecture and allowing communication across both wired and wireless networks.

3.3.2 Logical architecture of an NDN/DTN node

In this section, we present the technical design of the NoD implementation used during the experiments. To this end, we focus on the architecture of a node running the full NDN-over-DTN stack, even though that is not required for all network nodes in our architecture (we elaborate on the subject in Section 3.3.3.

A device deploying NoD needs to run both an NDN and a DTN implementation, which communicate with each other. On the NDN side of an NDN/DTN device, the default NDN implementation is used and the NDN Forwarding Daemon (*NFD*) is deployed. On the DTN side, the IBR-DTN implementation is used and the *daemon* is deployed. We note that our logical design uses integral components of the two architectures which need to be realized by any implementation of the protocols, such as NDN faces and DTN Convergence Layer Adapters. As a result, the individual protocol implementations could be substituted with different ones with the general design remaining the same.

We do not claim that this is the most efficient implementation of the protocol stack, as running two separate daemons is resource-consuming. Even though implementations could be created, the present one was based on the tools available and is sufficient for our experimentation purposes. More details regarding the specific tools are provided in Chapter 5.

Figure 3.3 depicts the design of an NDN/DTN network node in our NoD scheme.

As already mentioned, NDN nodes communicate with each other using faces which represent communication channels. Those faces can implement either physical network interfaces, or overlay channels in the form of tunnels. Based on this, communication between the two protocols is realized using a DTN face, which essentially creates a DTN tunnel to adjacent NDN/DTN nodes.

The DTN face communicates with an underlying DTN Bundle Protocol implementation, which encapsulates NDN Interest and Data packets into DTN Bundles. Those Bundles are transferred to other DTN-capable nodes using typical DTN routing protocols to handle intermittent connectivity and mobility. This way, NDN operation is extended to intermittently-connected environments.

It should be noted that DTN also communicates with lower-level protocols and encapsulates the Bundles into the respective protocol data units. Even though this

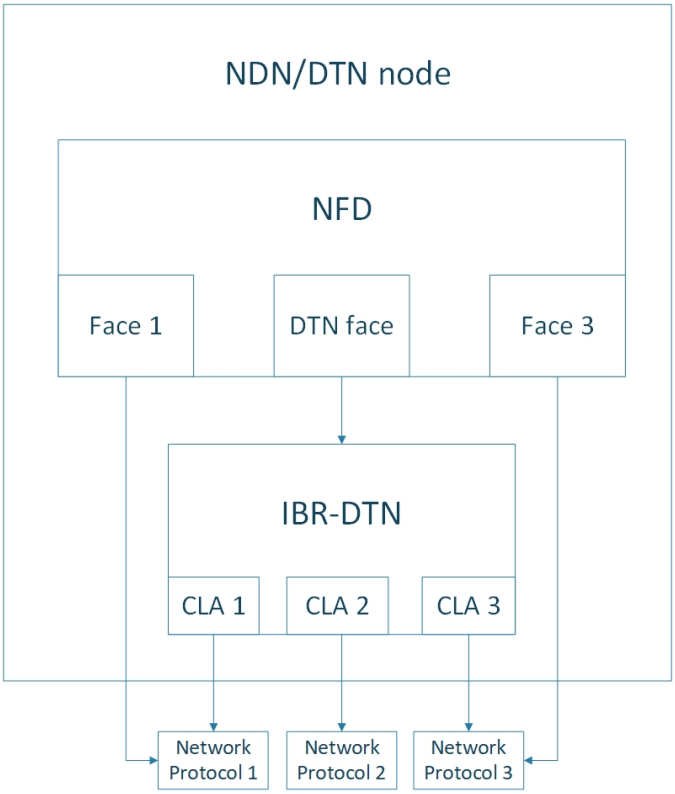


FIGURE 3.3: Components of a network node deploying the full NoD stack.

is evident for any network architecture, we point this out as DTN is frequently deployed at Layer 7 of the OSI model which may result in a large network stack and introduce significant overhead (e.g. consider the increased packet size caused by the headers).

However, DTN is quite flexible in its deployment and can even run directly over Layer 2 protocols, as long as a corresponding Convergence Layer Adapter (CLA) has been implemented. Therefore, in case constrained devices need to be supported, a large portion of the protocol stack can be done away with (e.g. resulting in NDN-over-DTN-over-MAC deployments).

This aspect is also mentioned due to the fact that DTN and NDN share some similarities in the fact that they were designed to be flexible in their deployment and compatible with popular Internet protocols. For that reason, they both use abstractions to enable deployments over network protocols (which may belong to different layers). NDN uses faces for this purpose, while DTN uses CLAs. While those abstractions differ in several ways, they share some common ground in the fact that they can be implemented to support various protocols.

A consequence of that particular design is that NFD can communicate over lower-level protocols which are supported by the corresponding DTN CLAs, despite the fact that NDN faces have not been implemented for those protocols. The opposite also holds true as, for instance, applications can use both a (direct) NDN-over-UDP and (indirect) NDN-over-DTN-over-UDP setup, depending on the deployment requirements.

As the core of this dissertation is formed by NDN, we will - for the most part - view DTN as one of the (several) available NDN transport options and refrain from providing much detail regarding protocol operation further down the protocol

stack. Therefore, the interactions between DTN and lower-level protocols will be abstracted away.

In this light, the DTN face creates an alternative communication channel for NDN in situations where typical TCP and UDP faces would fail. We effectively extend NDN operation by allowing it to tunnel NDN packets, offering reachability to remote areas where no typical Internet connectivity exists and resilience to services in challenged environments. Each time that information has to be transferred to/from an intermittently-connected network, the DTN face can be used. We stress that, as it forms an extension and is not the sole forwarding option, the DTN face can be used in conjunction with other faces by NDN forwarding strategies, allowing NDN adaptation in different environments.

Using this DTN face, two NDN/DTN devices can communicate the following way: Initially, NDN Consumer and Producer applications generate Interest and Data packets respectively, as they would typically do in vanilla NDN. Assuming those packets are forwarded using the DTN face, NFD subsequently encapsulates them in DTN Bundles. Those Bundles are then handled by the DTN daemon (dtnd) as a regular Bundle. Once a communication opportunity arises, i.e. an NDN/DTN device comes into contact with another DTN-capable device, the dtnd uses DTN routing protocols to decide whether the Bundle should be forwarded to that device or not.

In any case, the Bundle will eventually be forwarded to other DTN nodes and, should a path be formed, will arrive at its destination. We note that DTN operates in a host-centric manner, therefore each Bundle must have a destination. Intermediate DTN nodes can store-carry-and-forward the packet towards this destination.

Remember that in NoD, faces are used to create a tunnel between two NDN/DTN nodes; therefore, each node points to the other end of the tunnel, i.e. the next-hop NDN/DTN node. As a result, the -required- destination Endpoint IDs of a Bundle encapsulating an NDN packet are simply the EIDs of the next-hop NDN/DTN node.

NFD faces are identified using face URIs, while DTN endpoint is identified using EIDs (which are also URIs). This fact is leveraged by NoD, which utilizes the same URI for both. For instance, consider a node running both the NDN and DTN implementations. If that node's DTN Endpoint ID is *dtn://node1.dtn*, the corresponding NDN face URI would be *dtn://node1.dtn/nfd*.

This way, as long as dtnd is running in an NDN/DTN node, once NFD is started it automatically creates a DTN face with the same URI as the underlying IBR-DTN implementation. This face can be readily used for forwarding and when NDN packets are encapsulated into DTN bundles, the Bundle endpoint IDs can be automatically assigned by NFD based on the Face ID of the next-hop NDN/DTN node which has been chosen by the forwarding strategy.

Each NDN node includes forwarding information regarding the available routes and faces in its FIB, which can be populated with information regarding the available NDN/DTN neighbors the same way it is handles other faces/routes (either manually, or using an NDN routing protocol).

All NDN packets that are sent via some device's DTN face, are subsequently forwarded by DTN mobile devices creating a 'DTN tunnel'. Upon reaching their destination, the original Interest/Data messages are received and decapsulated by NFD through its DTN face, with NDN communication resuming at the NDN layer. This tunnel enables data forwarding between two remote, intermittently-connected NDN network parts (or single devices).

3.3.3 Deployment scheme

We accompany the NDN-over-DTN protocol stack, with a proposed deployment scheme. In particular, we posit that the full stack only needs to be deployed in certain network nodes, while the rest of the nodes comprising the network may only support one of the two protocols.

The choice of which protocol a node should deploy, however, is not random. Instead, it is based on the fact that the two protocols are complementary: NDN was mainly designed for fixed networks, while DTN for highly dynamic wireless networks. Therefore, we assume that each protocol is deployed in each respective domain. Mobile nodes which are part of an opportunistic wireless network, need only run the Bundle Protocol. Nodes which are part of a well-connected wired network, need only run the NDN protocol. Finally, edge/gateway nodes which lie between the two domains, are the only ones required to deploy the full NDN/DTN stack to provide interoperability between the different networks.

This setup offers substantial advantages: First, it increases our scheme's general deployability potential, as only a few specific nodes need to support our novel solution. No modifications are required on the NDN and DTN protocols running in the other devices.

Second, it enables data forwarding between two remote NDN nodes by using different upstream and downstream intermediate (DTN-capable) routes/nodes, overcoming breadcrumbs routing limitations without any modification of NDN semantics.

Third, it specifically increases deployability in resource-constrained environments, as only high-end devices need to run the full protocol stack (i.e. the edge devices acting as gateways). Resource-constrained devices such as smartphones and low-end sensors which have energy and memory/storage/processing constraints, can selectively use one of the two protocols depending on their network context. We point out the fact that although there is nothing prohibitive at a protocol level for more efficient NDN/DTN implementations to be supported by lower-end devices in the future, the Proof-of-Concept implementation used for this dissertation is resource-consuming and poses specific deployment restrictions.

For this reason the NoD scheme proposed in this dissertation only deploys the entire stack in high-end nodes such as WiFi Access Points and IoT Gateways, even though it can be potentially deployed in any node. Nevertheless, the experiments presented in Section 6 show that NoD can still significantly improve performance in intermittently-connected environments compared to a traditional host-centric DTN stack.

There are several setups which can follow this general scheme, allowing NoD to be deployed in various use cases. The protocol stack along with indicative deployments is illustrated in Figure 3.4.

The upper deployment depicts two NDN-based networks (Networks 1 and 2) which are connected by a mobile DTN. Those can represent two intermittently-connected network fragments which may occur in disaster cases. In this case, NoD can be used for information exchange between the two fragments using mobile data mules (e.g. smartphones). The NDN/DTN stack is deployed in the Access Points coming into contact with the DTN mules, enabling the NDN networks to send and receive information using the mules. The same mule can go back and forth, transferring Interest packets and returning the corresponding Data packets. Different mules can also be used, due to the fact that DTN is employed at the mobile network and, therefore, the symmetric routing limitations do not apply to the DTN mules.

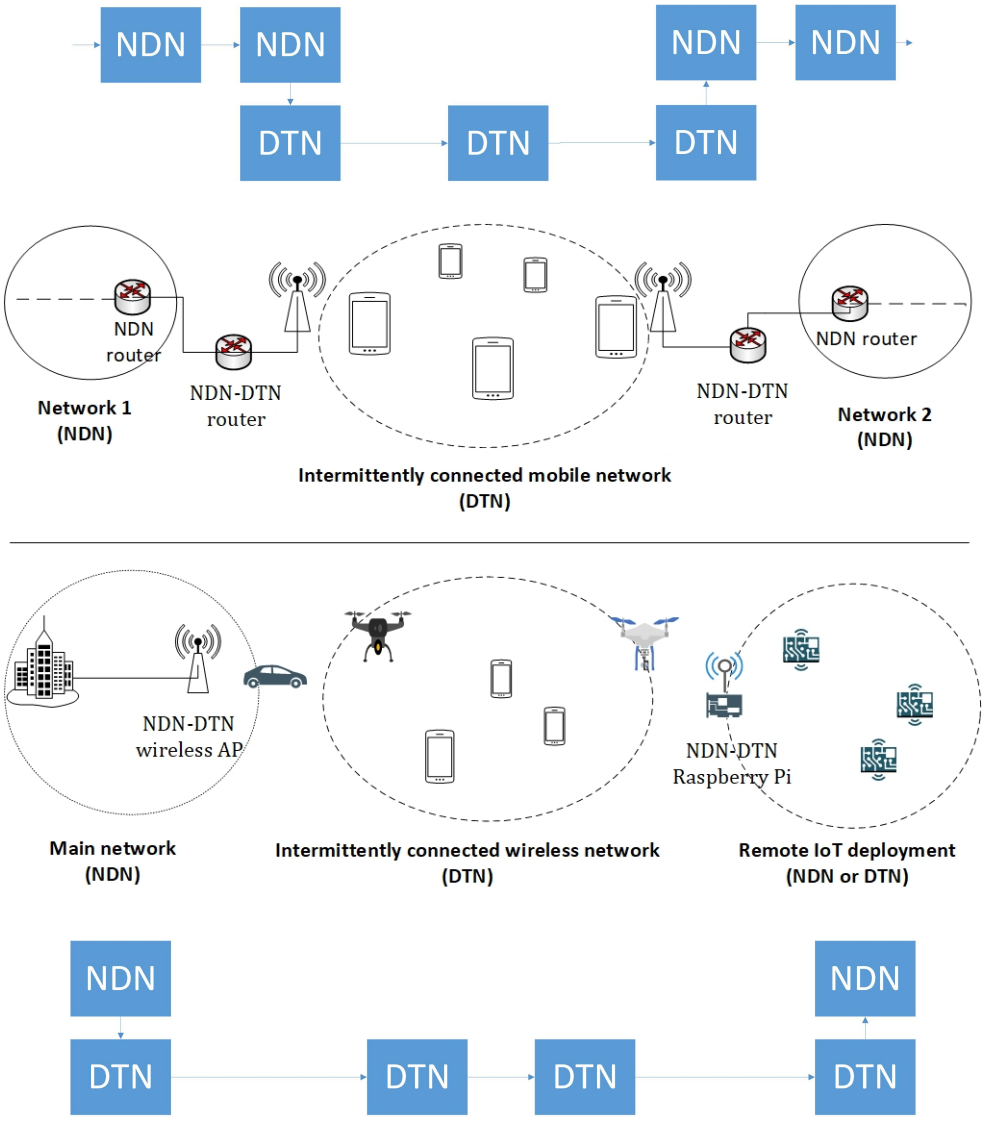


FIGURE 3.4: Protocol architecture and indicative deployments.

This way, the breadcrumbs routing limitations can be effectively tackled. The only requirement is that the NoD-enabled Access Point maintains FIB/PIT entry towards the Access Point of the other network fragment. This example showcases the capability for data forwarding between two intermittently-connected regions without modifications to NDN semantics, with the trade-off of not being able to support in-network caching in mobile DTN devices. Mobile DTNs can, however, retrieve cached content from the edge nodes which can still allow for an increased performance increase.

The lower deployment depicts a well-connected fixed network ("main network"), which is connected to a remote IoT deployment via a mobile opportunistic network. The main network is NDN-based, while the wireless sensor network is either NDN- or DTN-based. The mobile opportunistic network is comprised of several different types of networked devices: smartphones, cars, drones etc. In such a setup, users located in the main network need to retrieve sensor measurements produced by the remote sensors. NoD can be used to offer access to that information by deploying the NDN/DTN stack in a) the IoT Gateway (in this example, a Raspberry Pi assumes this role) and b) the Access Point which lies between the fixed network and mobile opportunistic one.

Other setups are also feasible. For instance, a variation of the second setup may consist of a main network and a network fragment which is comprised of an NDN/DTN Access Point and several smartphones which are connected to it. This is essentially the setup described in Section 3.4.1, which uses NoD to enable service deployment in challenged regions. In this case, data mules are leveraged to transfer not only single requests, but entire services (i.e. execution code that can run on hotspots) to remote regions. This way, service availability and latency are improved as the requests do not have to be periodically transferred back to the core of the network and basic computations are offloaded to the hotspots.

Figure 3.4 also showcases NoD's ability to act both as a stand-alone solution, as well as a solution which integrates into existing networks. In the former case an NDN/DTN node represents a communication endpoint running a Producer or Consumer app, while in the latter case the NDN/DTN node acts merely as a forwarder, allowing communication between different networks. Once Bundles are decapsulated, typical NDN communication resumes and the corresponding NDN packets are forwarded to adjacent NDN nodes in the fixed network using other faces.

3.3.4 Cross-layer Neighbor Discovery

In this section we present a variation of NoD, which includes some modifications to our original scheme. This new version is named "NoD-Discovery" and utilizes cross-layer information to improve forwarding by NDN/DTN devices regarding Interest/Data packets which sent to the DTN realm. The main objective of this variation is to decrease network overhead, especially important in some environments.

For instance, an important issue in ICDTNs using the NDN/CCNx architecture concerns the trade-off between short-lived and long-lived Interest packets. Long-lived Interest packets will be transmitted infrequently and incur less network traffic, but will stay in each PIT entry longer and may burden the memory further. Furthermore, in cases in which content is updated frequently, long-lived Interests may inhibit the retrieval of updated content versions by consumers. Conversely, frequent short-lived Interests will no longer produce permanent PIT entries and can facilitate retrieval of updated content versions, but will cause increased traffic.

As short-lived Interests may be preferable in certain scenarios, this would add a large traffic demand to a network deploying NoD. In NoD, a traffic increase in the NDN layer will subsequently cause an even larger network traffic in the DTN layer, due to the fact since DTN routing algorithms will typically replicate each bundle multiple times to ensure its delivery. Therefore, a single NDN Data packet may produce multiple identical DTN Bundles in the DTN domain.

To demonstrate the particular problem in more detail, consider the following scenario: A Consumer is using short-lived Interests and follows the vanilla approach, re-transmitting an Interest packet the moment its Lifetime expires.

However, content retrieval in ICDTNs mainly depends on the (large) time needed by the DTN mules to retrieve the content object. Therefore, in cases where e.g. the DTN part of the network requires minimum 1 hour to fetch a content object from a particular intermittently-connected producer, repeating the request (i.e. re-transmitting the Interest packet) every 3 seconds will not make any sense; instead, it will only increase network traffic. Content retrieval will only be feasible when the content object arrives at an edge router by a mule. If a PIT entry for that content object is active on the NoD wireless edge router/gateway at that time, the Data packet will be instantly forwarded back to the NDN Consumer. Otherwise, a new Interest is needed to arrive at the NoD wireless edge router/gateway to retrieve the object.

Initially, we assumed that a PIT entry for content that has been retrieved from the delay-/disruption-tolerant network part exists permanently on the wireless edge router/gateway so that the content can be instantly retrieved. The only way to satisfy this requirement is to re-transmit Interest packets immediately after their Lifetime expires and, hence, re-new the PIT entry. This is how re-transmissions are performed in our "NoD-Default" approach.

However, we observed that most of those re-transmissions are not needed and only produce additional overhead without increasing performance. In response to this, we designed "NoD-Discovery" which uses cross-layer interaction to reduce packet transmissions.

To deal with this challenge, we extended the operation of the NoD edge gateway, which interconnects the NDN with the DTN part of the network. This allows NDN to identify the existence of neighbor nodes prior to forwarding the corresponding Interest packets. The reasoning behind this is that when there is no connectivity between the NoD gateway and a DTN mule, there is no need to encapsulate a newly-arrived Interest packet into a DTN bundle since, otherwise, that bundle would have been forwarded by the DTN daemon when a mule arrives resulting in redundant traffic. Consider that each new Interest re-transmission may potentially result in several new Bundles. This can escalate quickly; in the case that 10 re-transmissions are performed in the NDN layer and each Bundle is replicated 10 times in the DTN layer, 100 Bundles will be produced for the retrieval of a single content object.

As Bundle replication depends on the DTN routing algorithm being used and is generally necessary to ensure their delivery, we focus our attention on reducing the unnecessary overhead on the NDN layer. Our selected strategy is to suspend forwarding of Interest packets to the DTN face on a NoD gateway, unless a connection to a DTN mule is established. An analogy for this approach is the 'stop-and-wait' algorithm: We 'stop' Interest transmissions at the wireless gateway when no connectivity is present and 'wait' for a neighbour to appear, in order to forward an Interest.

Previously, each re-transmitted Interest would be instantly encapsulated into a DTN Bundle at the Gateway. This Bundle would then be stored at the DTN layer and transmitted when the mule arrived, regardless of whether the corresponding Interest packet had expired on the NDN layer. Several Interest re-transmissions

which would occur between mule arrivals would result in several DTN Bundles. Those would be all be transferred by the DTN layer during the next available contact opportunity with a mule. Using cross-layer information, an NDN/DTN node can adjust its forwarding and only encapsulate 'active' Interests into DTN Bundles at the time of a node's arrival (instead of proactively creating Bundles which may outlive their corresponding, previously-expired Interests).

Note that NDN has no native mechanism for retrieving connectivity information with other devices (i.e., neighbour maps) through each of its faces. To surpass this, we made use of the lower-level DTN neighbor discovery mechanism (IPND) to make that information available to the NDN layer. As DTN is the layer that primarily handles intermittent connectivity in NoD, retrieving this information from that layer comes naturally as a design choice.

In Algorithm 1 we present the main steps of this cross-layer operation. One of the key components of our approach is the *BeaconInt* parameter, which indicates the beacon advertising interval. This parameter corresponds to the interval time between beacons and thus to the minimum time required to detect new contacts. The fine-tuning of this parameter depends on network conditions and requirements.

During the operation of the cross-layer interaction, a number of Interest packets sent by NDN Consumer applications are accumulated at the NoD gateway and the corresponding Interest List (*IntList*) is constructed. Next, NoD gateway retrieves the list of the most recent DTN neighbors, as the outcome of the IPND mechanism. If a DTN neighbor in the network is present at that time, every Interest packet currently included into the Interest List is encapsulated into a bundle, addressed to the destination Endpoint Identifier (*DestEID*). We note that, in our experiments, we applied flooding routing and, therefore, upon discovering new DTN neighbors, all the bundles were forwarded instantly. More sophisticated routing approaches could be used to cover each scenario's requirements.

Algorithm 1 NoD-Discovery.

```
Input: BeaconInt, DestEID
while true do
    IntList ← get.Interests          ▷ Construct the list of Interests to be transmitted
    NeighborList ← get.neighbors(BeaconInt)          ▷ Retrieve DTN neighbor list
    if NeighborExists(NeighborList) then
        for each Interest i ∈ IntList do
            CreateBundle(i, DestEID)          ▷ Forward Interest via the DTN face
        end for
    end if
end while
```

3.4 Use cases

3.4.1 Edge computing in challenged networks

Service deployment in post-disaster scenarios and remote areas is a challenging task. Nevertheless, effective solutions are needed, as the ability offering emergency services to users can be vital. The research community has been concerned with this issue, proposing ICN as a potential solution to provide connectivity in disaster scenarios [190] [191].

In this dissertation, we posit that NoD can be used to provide communications in such a scenario. To this end, we have used it for service deployment in remote areas which are only accessible using data mules and validated the feasibility of this approach [187][180].

To support such a scenario, we apply Service-Centric Networking (SCN) principles [192] and use PiCasso [185] as an application-layer service orchestration mechanism to deploy virtualized, lightweight services. Those services are implemented as docker containers and can be transferred to intermittently-connected areas for execution using NoD.

Following the ICN paradigm, services and content are decoupled from their physical locations and can be retrieved by any location in which they are available; this also includes in-network caching of service segments in NDN routers. Furthermore, the edge computing paradigm offered by PiCasso allows simple information processing to take place locally in remote areas, significantly reducing the need for data ferrying to/from the main network.

Regarding the service deployment functionality, we use lightweight containerized services which are treated as regular content by NDN. However, contrary to typical content, they can be deployed and executed across all compatible devices (e.g., a hotspot). Those services are coupled with a semantic naming scheme to take advantage of caching and name-based routing, allowing users to access nearby copies of services. Furthermore, semantic naming allows a user can request a service that matches certain criteria. For instance, the location of users can be embedded into an Interest message to request customized services (e.g., information of local emergency authorities, local map).

To support such a scenario, in which large services are transferred by data mules, we have developed and employed two relevant mechanisms. The first one is 'multi-Interest forwarding', which controls Consumer-side Interest transmissions to improve large file transfers in challenged environments. The second one is the use of large packet sizes, which addresses issues related to network-layer service fragmentation and instantiation.

Multi-Interest forwarding NDN natively follows a synchronous communication model. In this model, a Consumer application sends each time one Interest packet to retrieve one Data packet. This model is inadequate in scenarios in which a data mule forms a disruptive communication link between the Consumer and Producer, as is the case in the disaster scenario considered here. In this scenario, a smartphone acting as a data mule would need to travel several times between the main network and the disaster area to deliver all the data chunks comprising the service. Our goal is to minimize the number of physical round trips a data mule has to perform in order to fetch the service.

As a solution, we implemented a *multi-Interest forwarding* model: the main idea is to have a Consumer application issue a burst of Interests (N) to retrieve multiple Data chunks at once. The adoption of a multi-Interest forwarding model allows a mule to avoid retrieving and delivering one chunk at a time and retrieve the content/service in the fewest possible round trips. This is especially important in DTNs, as a single round trip may lie in the order of minutes/hours.

To achieve this, we need a mechanism to inform the received about the total number of chunks which will be required for service retrieval. This mechanism is developed on the application level and operates the following way: A Consumer initially sends an Interest packet requesting a service, which is forwarded towards the original content source or nearby cache. Upon receipt of the first Data packet, the Consumer also receives information about the total content/service size and data chunk

size (included by the Producer). On this basis, the Consumer calculates the total number of Interest packets which are required to retrieve the entire content/service TNc.

Having retrieved that information, the Consumer can then send multiple Interests to the mule. For instance, all remaining Chunks can be requested at once by setting $N = TNc$. The Interests would be transferred by the mule and the entire service can be potentially retrieved upon the second arrival of the mule (resulting in a minimum of two overall round trips). However, this approach poses the risk of creating network congestion. To avoid the problem, the Consumer can set N at a smaller value. For our validation purposes we statically set $N = 30$, however adaptive mechanisms can also be used for congestion control.

Deployment: We have successfully used those mechanisms for service deployment in remote hotspots, validating the ability of our default NoD scheme to support edge computing in challenged environments. The deployment setup can be seen in Figure 3.5.

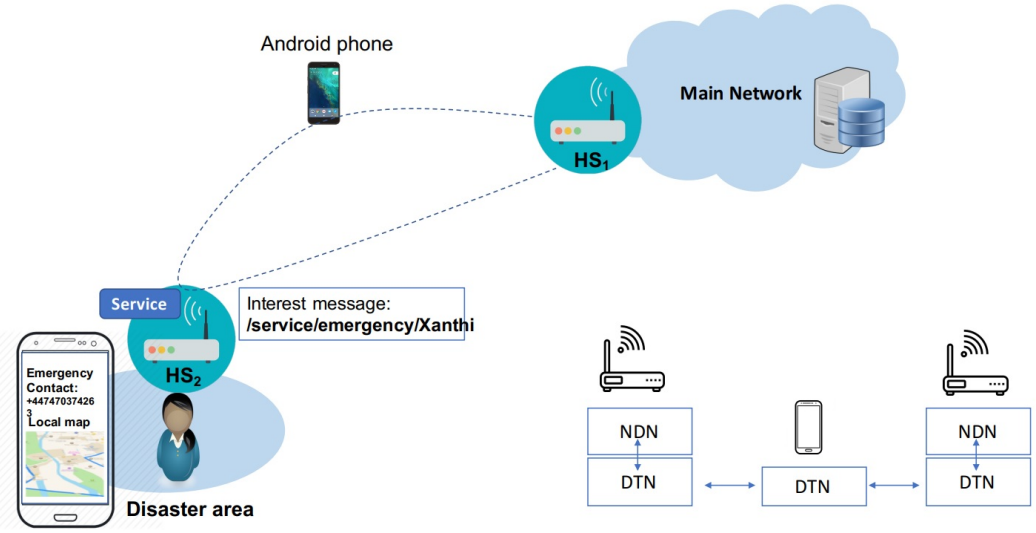


FIGURE 3.5: Edge service deployment in challenged networks.

We assume that HS_1 is at the edge of the main network and that the disaster area network is miles away and disconnected. The aim is to retrieve a service (S) from the *Main Network* and deploy it in the *Disaster area* (HS_2). We assume S to be a stateless service, e.g., a self-contained web server.

Both Wi-Fi hotspots HS_1 and HS_2 are equipped with the NoD. *Android phone* is a physically mobile, DTN-enabled node that can travel backwards and forwards between HS_2 and HS_1. Drones or other mobile devices can, of course, replace the Android phone, as long as they are DTN-enabled.

An Interest request is initially constructed by HS_2, including the name of the desired service (S). The location of the affected area is embedded into the Interest name (`/service/emergency/Xanthi`). This Interest is forwarded to the DTN face when the phone comes within the wireless range of HS_2. The Interest is encapsulated in a DTN bundle and stored in the phone's persistent storage. The phone loses contact with HS_2 when it travels towards HS_1.

When it reaches HS_1, the Interest is decapsulated and delivered to the NDN layer. It is then transferred by HS_1 through the main network, towards the service producer or the nearest cache. The latter —or another intermediate NDN node in

possession of S— retrieves it and prepares a service image with customized information (i.e., emergency contact of local civil protection authority or map in `Xanthi`). The reply includes one or more Data messages sent along a reverse path using the DTN face. Communication follows the Multi-Interest protocol. When HS_2 receives all the chunks of S, it calls a service deployment function to execute S to provide the emergency service to local users via WiFi. They can access S from any standard web browser.

Packet size considerations : As a supportive mechanism for similar deployments, we have experiment with the use of large NDN packet sizes. In particular, in the aforementioned deployment we have allowed the creation of packets up to 20MB.

Although an unconventional approach, our rationale was based on observations we made during deployment. The most important one was the realization that the maximum packet size in NDN (8800 Bytes)[4] caused our services to split into a very large number of chunks. This would not pose an issue in a fixed network, as service retrieval is treated by the network like the retrieval of a large file: all of the chunks would be eventually retrieved in a more-or-less finite amount of time. However, this is not the case in a DTN setting.

We noticed that, quite frequently, even though we deployed our Multi-Interest forwarding mechanism, some of the chunks would not be returned upon arrival of the mule as they either were lost, or the corresponding Interest packets were never delivered to the hotspot. This called for end-to-end reliability mechanisms and additional round trips.

This is due to the fact that DTN only offers hop-by-hop reliability and end-to-end delivery is not guaranteed for each Bundle; it depends on the mobility and connectivity patterns of the intermediate DTN nodes. Contact duration, wireless link error rates and link congestion and contention also affect packet losses.

In other scenarios, such as live streaming, those packet losses are totally acceptable. However, in our case, the service could not be instantiated at the hotspot unless all of its fragments were received. This turned out to be a significant issue when employing multi-Interest forwarding, as the more the fragments, the bigger the chance that at least one chunk would not be delivered. Even though the data mule employed in this setup followed a fixed route (as it would go back and forth between the hotspots) and would eventually retrieve the entire content, service instantiation would be increasingly delayed due to the additional round trips. In critical situations, such as emergency response, this is a highly undesirable situation.

This is why we experimented with a larger-than-typical packet size, which would result in only a small number of service fragments. This improved service retrieval and instantiation and facilitated the management of the deployment process by PiCasso, as loss of a service fragment could be easily identified and handled even at the absence of end-to-end reliability mechanisms. The motivation for this approach came from the fact that the Bundle Protocol (BP) allows for very large DTN-layer Protocol Data Units (i.e., Bundles) to be constructed. In fact, MB-sized Bundles are not that uncommon - this lies in the original design of the BP, which was conceived to transfer bundles of lower-level PDUs.

Following this line of thought in our design (which uses DTN as an underlay), we realized that allowing MB-sized NDN packets would result in those packets being

[4]Currently defined in the *tlv.hpp* header file by the *MAX_NDN_PACKET_SIZE* variable, as of August 2022. Link https://github.com/named-data/ndn-cxx/blob/47a94d10dcfd74970448c732c76276ef53e629ea/ndn-cxx/encoding/tlv.hpp

encapsulated into a single Bundle. This Bundle could then be transferred using the hop-by-hop reliability offered by DTN which could, in turn, ensure its delivery in scenarios in which a fixed-route mule is deployed to transfer the service to a specific location. In the extreme case, a service can be sent as the payload of a single Data packet, encapsulated into a single Bundle and reliably delivered by a DTN mule[5]. Fragmentation, of course, happens in lower layers of the protocol stack.

Even though such a solution may be - borderline - acceptable for fringe deployments in very specific use cases (such as the one described in this section), it is highly unorthodox and we do not recommend the general use of MB-sized packets in NDN. We were, however, motivated to further explore the implications of such a choice and quantify what would be the impact of packet sizes slightly larger than the 8.8 KB default limit in typical deployments[193], as little attention has been given to the impact of packet size on NDN performance. The corresponding evaluation and results are described in Chapters 5 and 6.

3.4.2 Content retrieval from intermittently-connected IoT devices

In the addition to its deployment on an edge computing use case, NoD has also been deployed to enable content retrieval from remote IoT devices [188] [189]. A detailed description of the experiments and corresponding results are provided in next Chapters. In this section, we provide the use case background and motivation for NoD deployment in such a scenario.

Some of today's common IoT use cases necessitate real-time operation and ultra-low latency (e.g., Industrial and Smart Home applications). However, there also exist some challenging scenarios in which IoT devices are located in remote or poorly-connected areas (sometimes referred to as the "Internet of Remote Things")[194]. Environmental Monitoring [195] and Smart Farming [196] are two examples which fall into the latter case.

In those scenarios, remotely-located IoT devices are often connected to the Internet using cellular communication technologies such as LTE or NB-IoT [197]. However, these technologies require an infrastructure which is not always present or even easy to install in isolated areas, mountains or large crops. Furthermore, cellular communication may not be a viable option in remote IoT deployments due to cost or energy limitations.

A cost-efficient alternative solution suitable for rural and remote IoT deployments is the use of mobile devices to reach these areas, gather the produced IoT data, store and ferry them back to Internet-enabled areas. These devices which act as "data mules" could be robots, drones [198], vehicles or even satellites [194][199].

Relevant content retrieval use-cases include UAVs that move across a forest in order to collect environmental sensor measurements, buses that repeat the same route periodically and fetch data from IoT gateways that are deployed in a city, agribots moving in a field and aggregating agricultural measurements, nanosatellites orbiting the Earth and periodically connecting to isolated areas.

In such environments, communication of "things" with a mobile node can be established either directly or indirectly. Using the indirect approach, protocols need to be deployed on low-end devices and each device can independently communicate with mobile nodes. Although this approach often forms the primary choice,

[5]Although not documented, we have performed such a trial and validated that it is feasible. In particular, a small service (20MB) was successfully transferred in a single Data packet using a data mule and subsequently instantiated in a remote hotspot.

the deployment of complex networking stacks on low-end devices is not always straightforward due to resource limitations (e.g. energy consumption).

Thus, in this dissertation we follow the indirect connectivity approach and deploy NoD stack on an IoT gateway. An IoT gateway is a high-end device which collects sensor measurements from the adjacent low-end IoT devices, performs efficient IoT data management and simultaneously interacts with the mobile devices. Given the intermittent connectivity and limited energy resources, desirable features for a remote IoT gateway include mobility support and reduced communication overhead.

We posit that the deployment of NoD in such a scenario in the form of a gateway-oriented scheme can offer such features and confront the challenges of remote IoT networks. NoD is able to combine the lightweight protocol stack, packet naming scheme and in-network caching capabilities of the NDN architecture with seamless operation in delay tolerant scenarios, by exploiting the store-carry-and-forward mechanisms of the DTN architecture. The functional autonomy of NDN and DTN is feasible, allowing for selective use of those protocols by individual resource-constrained devices. At the same time, the employment of the full NDN/DTN stack and cross-layer interaction (see 3.3.4) mechanisms by IoT Gateways can minimize network overhead and facilitate deployment in remote IoT environments.

3.5 Mathematical model

In this section, we present the mathematical model we developed for our system. In sub-section 3.5.1 we present our general model, which focuses on the interrelations among NDN and DTN metrics and their inter-dependencies in NoD, and particularly on the *Delay*, *Interest Satisfaction Ratio* and *Overhead Ratio* metrics. This model is valid for any NoD deployment, regardless of whether the DTN network part is operating using one or more mules, or whether the mules operate under pre-determined or opportunistic connectivity and mobility patterns. It essentially models the process of retrieving content objects/data from intermittently-connected devices using data mules and edge caching functionality.

In sub-section 3.5.2, we graft more details to the general model with respect to data muling operations, such as the ones present in our first round of experiments[6]. The relations presented the aforementioned sub-sections are agnostic of the re-transmission mechanisms that are in use by NoD Consumers, or the specific implementation of the architecture.

In sub-section 3.5.3, we focus on modeling the operation of our specific re-transmission mechanisms, as well as the behavior of our particular implementation in the specific experiments presented in this dissertation. Notations are summarized in Table 3.1.

3.5.1 General model

We model the relation between DTN and NDN performance metrics in terms of the NDN *Interest Satisfaction Ratio* and *Overhead Ratio*, given the NoD gateway *Cache Hit Ratio* and the DTN *Delivery Ratio*. We also model the impact of NDN- and DTN-related parameters on the average NoD content retrieval *Delay*.

To the best of our knowledge, our work is the first attempt to combine NDN and DTN metrics in a common framework. The model allows the evaluation of

[6]Presented in Section 5.1.4.

Notation	Definition
D_c	(Local) Cache-retrieved Data delay
D_p	(Remote) Producer-retrieved Data delay
D_{avg}	Average Data retrieval delay
C	Number of Data packets received from (local) cache
P	Number of Data packets received from (remote) Producer
R	Total number of received Data packets
S	Total number of sent Interest packets
T_{int}	Number of Interests transmitted via DTN
h	Cache Hit Ratio
ISR	Interest Satisfaction Ratio
DR	Delivery Ratio
OR	Overhead Ratio
n_{RT}	Data mule Round Trips
d_{RT}	Data mule Round Trip Duration
d_C	Contact Duration
d_S	State Duration (used in experiments)
L_{int}	Interest Lifetime
p	Probability of delivery within a single muling round trip
n_{ppc} or ppc	Number of Interest packets-per-contact sent by Consumer

TABLE 3.1: System model notation

the performance improvement provided by deploying NoD instead of a host-centric DTN implementation.

The generality of the model is appropriate for quantifying the performance improvement produced by edge caching techniques for intermittently-connected content retrieval using DTNs, even when NDN is not the network-layer protocol of choice on the core network (i.e., in case caching is performed in higher layers of the protocol stack).

The model assumes an NDN Producer/Consumer pair which is intermittently inter-connected using DTN data mules that relay Interest/Data packets back and forth between them. This assumption is valid when an NDN Consumer is located on the fixed part of the network and needs to obtain Data packets from a remote device (e.g. sensors).

In a typical DTN implementation, all requests would have to reach the 'Producer'/'server' and the response would be carried back. This usually involves very large delays, in the order of minutes or even hours.

However, when NDN is used as an overlay to a DTN implementation, the received Data packets are cached in the fixed part of the network and subsequent Interests may be satisfied by the cache. As delays in the fixed (NDN) part of the network are several orders of magnitude smaller than the ones in the DTN part (minutes/hours vs milliseconds), NoD can provide significant performance benefits.

To quantify the performance improvement, we developed a model, which employs typical NDN- and DTN-related metrics and assesses the influence of caching by local devices. We assume that: (i) all requests are related to content originating from intermittently-connected areas (i.e. the Producer and Consumer are not located

in the same network), (ii) the corresponding content is always available and can be sent by the remote Producer when these requests arrive, (iii) no caching is performed by the intermediate mules. Hence, a content object can be received either from a remote Producer, using DTN, or from a local cache (if previously requested by another NDN Consumer located on the same network and cached by the NDN/DTN Gateway or another on-path NDN node).

We begin by focusing on the popular NDN metrics of *Interest Satisfaction Ratio* ($ISR = SatisfiedInterests/SentInterests$), *Cache Hit Ratio* ($h = CacheHits/ReceivedInterests$), and the DTN *Delivery Ratio* metric ($DR = DeliveredBundles/SentBundles$).

We have formulated a relationship between those metrics, which can be used to calculate the Consumer's *Interest Satisfaction Ratio* on an NDN/DTN deployment. This relationship is provided by the following equation:

$$ISR = h + (1 - h) * DR^2 \tag{3.1}$$

To provide a better insight into this relationship, we include an example of an Interest-Data exchange pattern in Figure 3.6.

In this example, we assume that there is an one-to-one correspondence between NDN Interest/Data packets and DTN Bundles and that only a single copy of each Bundle is routed each time in the DTN part of the network, without loss of generality. The topology considered is one in which an NDN Consumer located in one network requests data objects from an intermittently-connected Producer located in another network. The two communicate using Data mules, while a NoD Gateway ("NDN-DTN router") is located in the same network as the NDN-only Consumer and facilitates communication between the NDN and DTN domains. The NoD Gateway is assumed to have previously cached content objects (possibly requested by a different Consumer), which can be retrieved by the NDN Consumer depicted in this example. Assuming that 20% of the content objects requested exist in the cache (which indicates a *CacheHitRatio* of 0.2) and that 75% of the Bundles are delivered by the delay-tolerant network in the timeframe of the example (indicating a Delivery Ratio equal to 0.75), the following exchange sequence takes place.

Initially, an NDN Consumer sends 20 Interests to a remote producer via a NoD Gateway. Once they arrive at the Gateway, the Interests will either be satisfied by the cache or forwarded to the DTN face. In our example h*20 (4) Interests are satisfied by the cache, while the rest (16) are forwarded through the DTN face and encapsulated into DTN Bundles. DR% (12) of these bundles arrive to the remote producer and the Interests are decapsulated by the NFD (the rest have not been delivered). The NDN producer receives those 12 Interests and responds with an equal number of Data packets. Those Data packets are again encapsulated into DTN bundles and forwarded by DTN devices to the NDN/DTN router which requested them. DR% (9) of those Data packets arrive back, are decapsulated, cached in the NDN layer and forwarded back to the consumer. By the end of the exchange, the NDN consumer will have received 13 Data packets (4 from the cache and 9 from the remote producer). Therefore, its ISR will be 13/20 (65%).

The same value can be calculated using Equation 3.1. This way, the ISR at the consumer at any given time can be calculated by using the local router's NDN Cache Hit Ratio and DTN Delivery Ratio. ISR estimations can also be made, using the expected values of h and DR.

The dynamics of those metrics are able to capture the potential improvements in the NoD architecture's performance made feasible by innovation happening in the

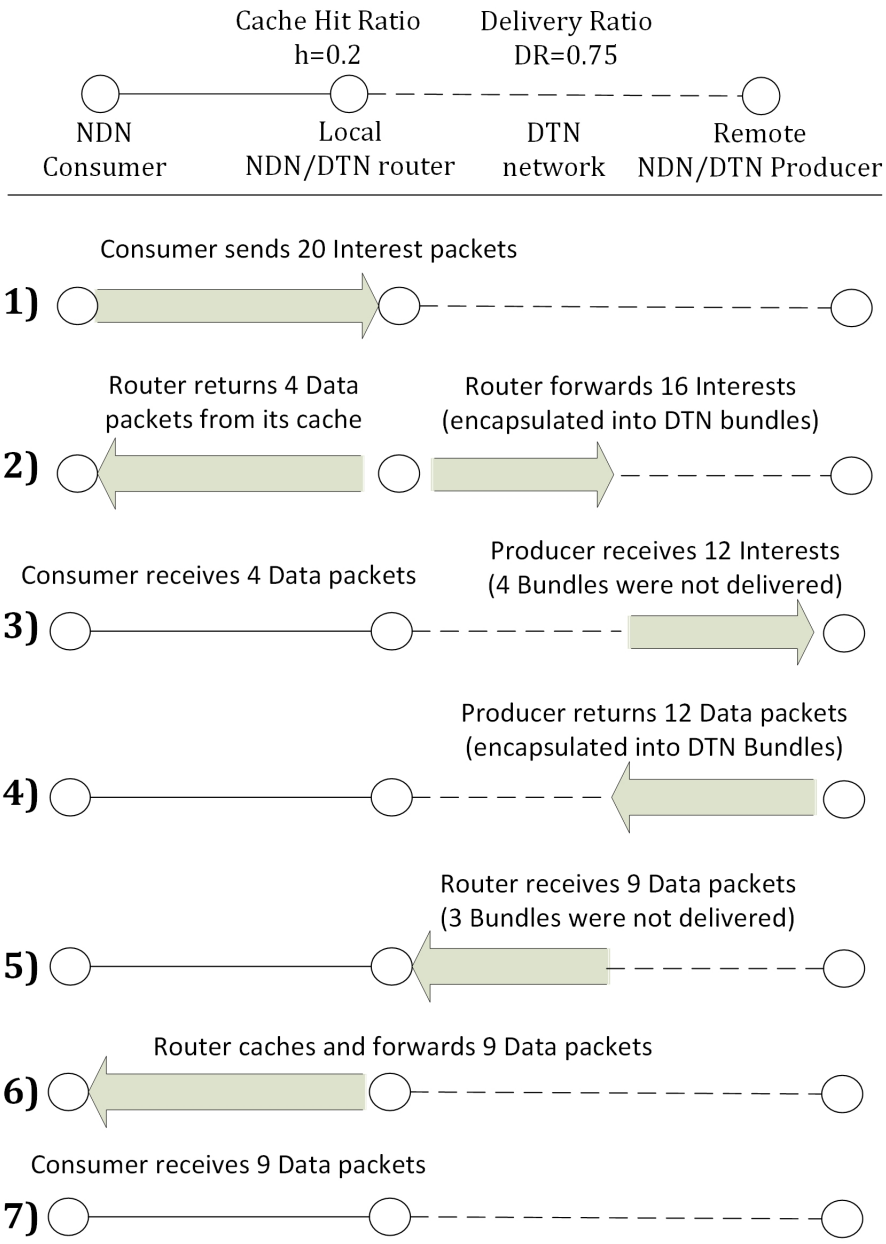


FIGURE 3.6: Example of an Interest/Data exchange pattern in a NoD deployment

individual layers of the stack. For instance, better-performing DTN routing algorithms can increase the corresponding *Delivery Ratio* and, in turn, the *Interest Satisfaction Ratio*. The same also applies for improvements in NDN caching algorithms, which can boost the *Cache Hit Ratio* and improve the *ISR*. As a result, although the NDN and DTN domains are functionally independent, performance-wise they impact implicitly one another. To the best of our knowledge, this is the first work which describes this relationship and combines NDN and DTN metrics into a common framework.

Next, we shift our focus to a more generic network metric, the delay. The *Average Delay* (D_{avg}) for receiving a content object from intermittently-connected devices, can be calculated as:

$$D_{avg} = (1 - \frac{h}{ISR}) * D_{p_{avg}} + (\frac{h}{ISR}) * D_{c_{avg}} \tag{3.2}$$

where h is the local Consumer-side *Cache Hit Ratio*, while $D_{p_{avg}}$ and $D_{c_{avg}}$ are the average delays for retrieving a content object from a remote Producer or local cache, respectively. Using Equation (3.1) to substitute *ISR*, Equation (3.2) can be written as:

$$D_{avg} = D_{p_{avg}} - (\frac{h}{h + (1-h) * DR^2}) * (D_{p_{avg}} - D_{c_{avg}}) \tag{3.3}$$

Typically, $D_c \leq\leq D_p$, especially in delay-/disruption-tolerant scenarios, so the previous equations can be simplified to:

$$D_{avg} = (1 - \frac{h}{ISR}) * D_{p_{avg}} \tag{3.4}$$

and

$$D_{avg} = (1 - (\frac{h}{h + (1-h) * DR^2})) * D_{p_{avg}} \tag{3.5}$$

To depict NoD's behavior for different ranges of the aforementioned parameters, we have used the previous equations to plot Figures 3.7-3.10, which focus on the ISR and Delay metrics. Those Figures can give an intuitive grasp of the system behavior under various conditions.

Figure 3.7 depicts the Interest Satisfaction Ratio (ISR) as a function of the Delivery Ratio (DR), for different local Cache hit ratio (h) values. It can be inferred that using an NDN/DTN architecture provides significant benefits compared against a host-centric DTN, as caching allows retrieval of remotely-produced content inside NDN islands even at times when there is no routing path to a remote producer (DR=0). This ability of NDN consumers to retrieve content that has been previously fetched from remote producers and cached inside the Named Data Network where consumers are located, is proportional to the local routers' Cache Hit Ratio.

Figure 3.8 quantifies the ISR improvement for different levels of local Cache Hit Ratios, using a no-caching scenario (h=0) as benchmark. This way, we assess the influence of NDN/DTN caching, when compared against host-centric DTNs. As inferred from Figures 3.7 and 3.8, the lower the DTN Delivery Ratio, the bigger the relative improvements. Even a Cache Hit Ratio as low as 1% (h=0.01) can improve ISR by 99% in networks with a Delivery Ratio as low as 10% (DR=0.1). Thus, our NoD scheme can prove particularly valuable for DTNs with low Delivery Ratios.

Using Equations 3.2-3.5 we can also model the average delay for a number of scenarios. Two particular cases call for special attention:

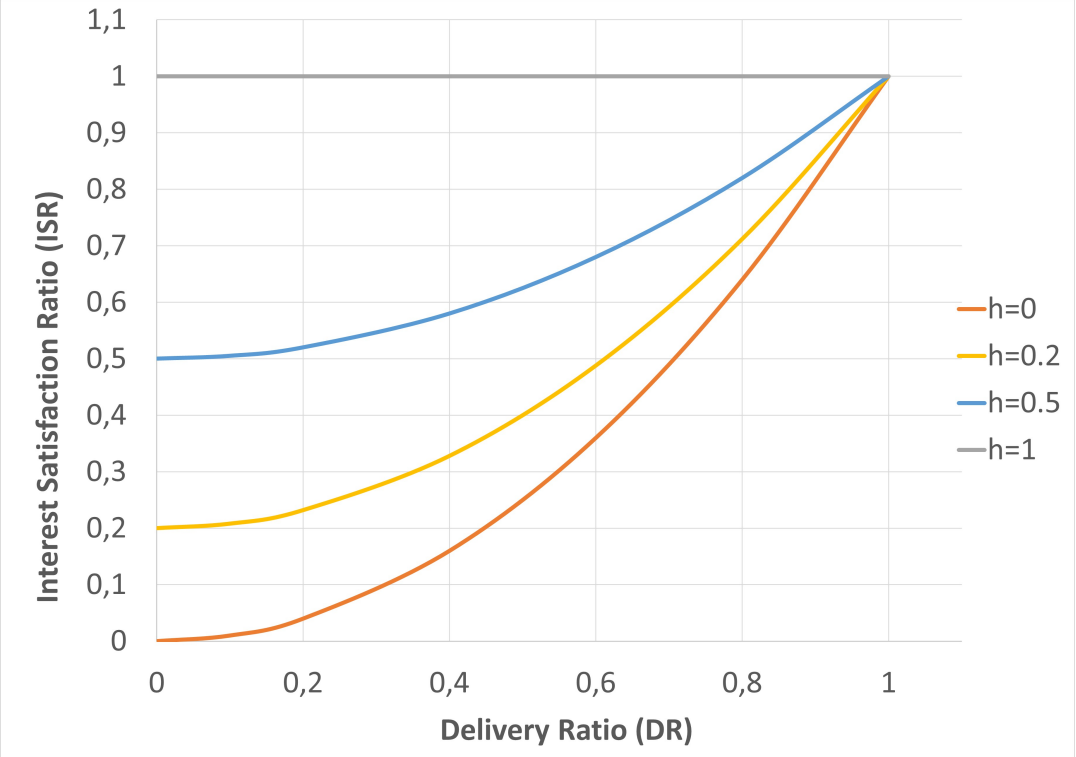


FIGURE 3.7: Interest Satisfaction Ratio (ISR) as a function of the Delivery Ratio (DR), for different Cache Hit Ratio (h) values.

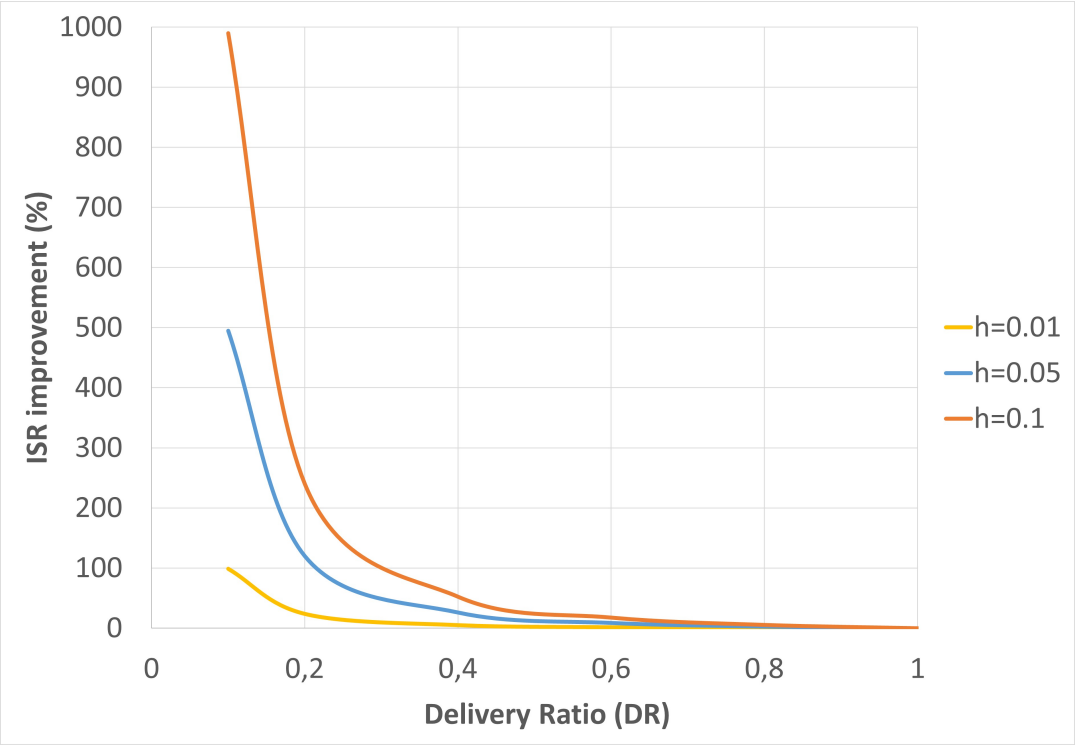


FIGURE 3.8: ISR improvement achieved by NDN-DTN, compared to a baseline no-caching scenario

- **No caching scenario** (h=0): $D_{avg} = D_{p_{avg}}$
  This is the average delay when there are no cache hits. As it can also be intuitively inferred, it is equal to the average delay for retrieving a packet from the producer. As host-centric DTNs do not offer caching capabilities, we use this scenario as a benchmark to measure the improvement of the NoD delay.

- **No-loss scenario** (DR=1): $D_{avg} = (1-h) * D_{p_{avg}} + h * D_{c_{avg}}$
  This can occur if all bundles are delivered to their destination within a given time constraint. DTNs that use data mules explicitly deployed to ferry packets towards specific destinations could achieve this, when available time suffices. In this case, if we assume that the caching delay is negligible ($D_{c_{avg}} = 0$), the delay reduction introduced by caching is equal to the Cache Hit Ratio h.

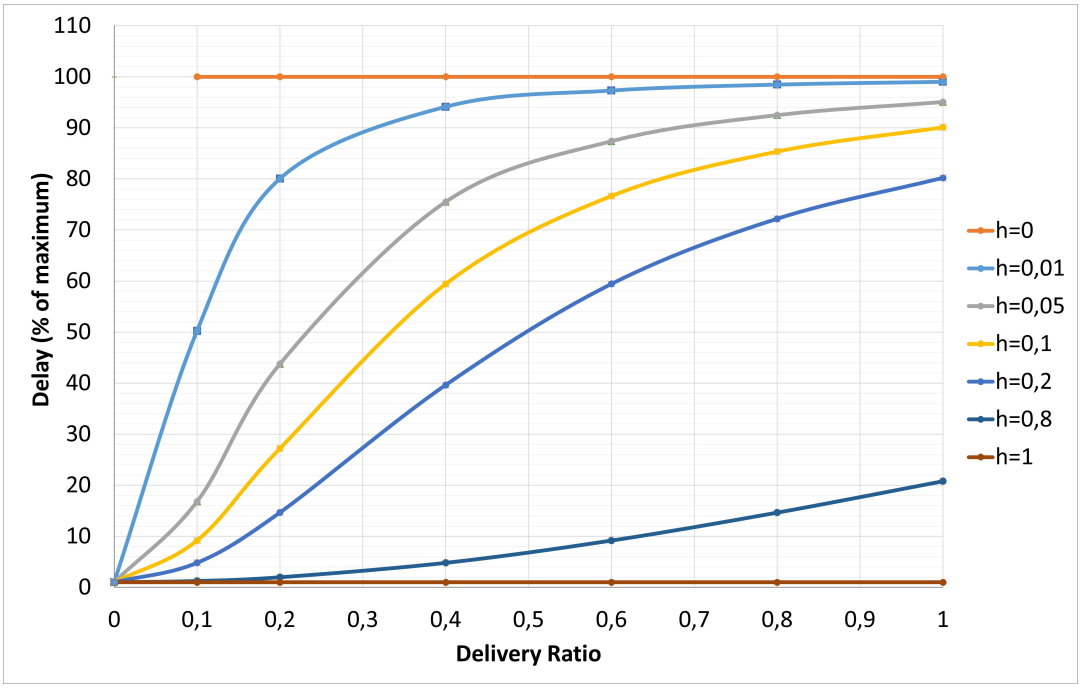


FIGURE 3.9: Average Delay achievable by NDN-DTN, as a percentage of the worst-case delay (Dp). The delay is modeled as a function of the Delivery Ratio (DR) and Cache Hit Ratio (h).

The influence of the Cache Hit Ratio on the Average Delay for remotely-received content is shown on Figure 3.9. The values are plotted by setting Dp/Dc=100 (a conservative estimation, as higher ratios occur in real DTN deployments).

Based on the Figures and related equations, our general model indicates that following behavior takes place in any NoD deployment:

- In networks with a very low Delivery Ratio, even a small Cache Hit Ratio can make a tremendous difference on the (remotely-retrieved) content delay. For instance, a 1% Cache Hit Ratio can reduce the Average Delay by 50% in a network with a Delivery Ratio equal to 0.1.

- In networks with a higher Delivery Ratio, there is lesser influence of cache hits on the average delay. Larger Cache Hit Ratios are required in order to significantly reduce the average delay (for a given Delivery Ratio).

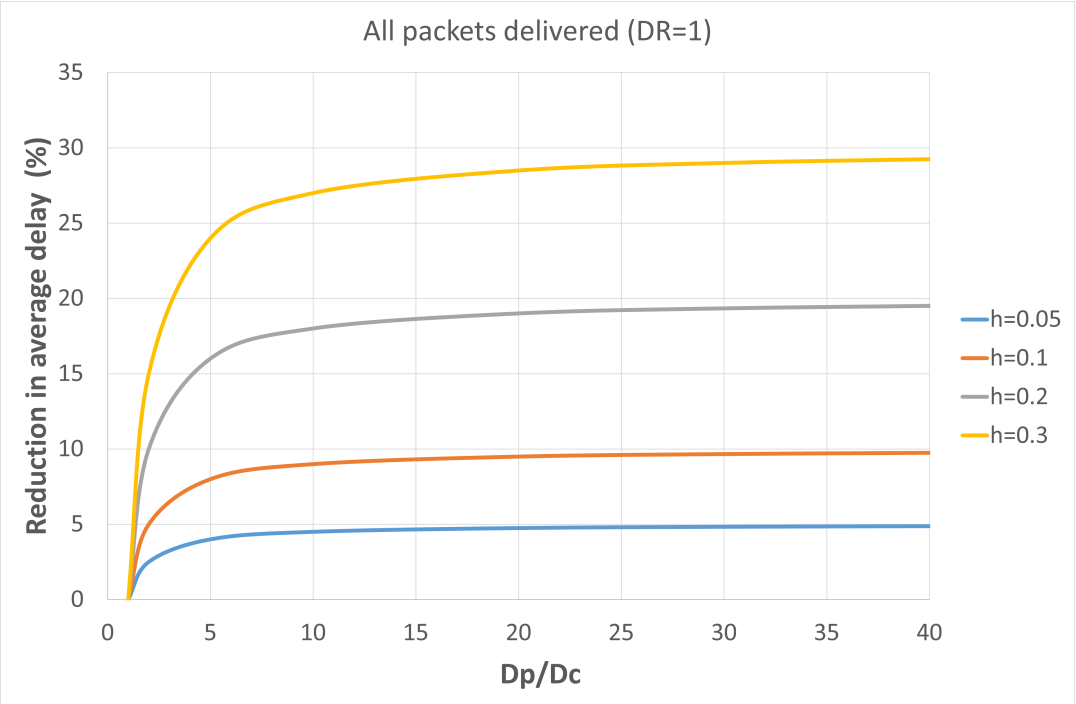


FIGURE 3.10: Average Delay reduction in NDN-DTN in a no-loss scenario ($DR = 1$), modeled as a function of Dp/Dc, when compared to host-centric DTNs.

- The upper limit of the Average Delay is the (average) delay when content is retrieved from remote producers (Dp). This is the worst-case value achieved in no-caching scenarios, which can be paralleled to host-centric DTNs. The Average Delay values tend to approach Dp for higher Delivery Ratios and lower Cache Hit Ratios.

- The lower limit of the Average Delay is the average delay of content retrieved from a local cache (Dc). This is the value achieved for h=1 (all content requests are answered by a local cache) and Dp=0 (no content is delivered by the DTN part of the network - all content that can be retrieved is the content that exists in local caches). The difference between the two cases is that in the former, all requests are satisfied (ISR=1), while in the latter only a fraction of them is (ISR=h).

- When all packets are delivered (DR=1), the delay decrease by introducing local caches is equal to the Cache Hit Ratio. We illustrate this case on Figure 3.10, plotted for variable Dp/Dc ratios (larger ones approach DTNs).

Finally, our model also incorporates another typical DTN metric, the *Overhead Ratio* (OR). OR reflects how many redundant packets need to be sent in order to receive a single packet. The lower its value, the better the network performance, as a large number of additional copies for a certain object can deplete the data mules' storage and hinder delivery of other objects.

We posit that the use of this metric can also be extended to ICDTNs, since they exhibit similar characteristics. The *Overhead Ratio* is generally defined as:

$$OR = \frac{SentPackets - ReceivedPackets}{ReceivedPackets} \tag{3.6}$$

Given that we are interested in the *Overhead Ratio* from an NDN perspective, the *Overhead Ratio* can be expressed as:

$$OR = \frac{SentInterests - ReceivedData}{ReceivedData} \tag{3.7}$$

or, in short (see Table 3.1)

$$OR = \frac{S - R}{R} \tag{3.8}$$

From where we derive the following forms (details are included in Appendix A.2):

$$OR = \frac{1}{ISR} - 1 \tag{3.9}$$

$$OR = \frac{1}{R} * T_{int} + \frac{h}{ISR} - 1 \tag{3.10}$$

The last form presents a relationship between the number of Transmitted Interests and the Overhead Ratio, which is not as straightforward as in host-centric DTNs due to the effect of caching. An approximation of the expected value of T_{int} for our experimental setup is presented in Sub-section 3.5.3.

By using the generalized definition of OR for NDN/DTN deployments, additional relationships can also be derived between the aforementioned metrics. At this point, however, we refrain from an exhaustive presentation of all possible forms of the equations. More specific forms that are relevant to our experimental setup, will be found in the following sub-sections.

It should be noted that, in our equations, fixed average values are assumed. However, different values may hold true for different points in time. An example is the DTN *Delivery Ratio*, which typically increases with time, as new contact opportunities appear and new paths to the bundle destinations are found by the DTN routing protocols. Likewise, the NDN *Cache Hit Ratio* largely depends on potential matches between past and future requests, which also are time-variant, as traffic patterns and request distributions in real deployments may change dynamically.

3.5.2 Data muling model

We also model data muling scenarios in which a single, dedicated, mule is going back and forth between a region to retrieve and deliver data. We begin by modelling the delay to retrieve a content object from a remote Producer (D_p). In such context, D_p can be calculated as:

$$Dp_i = \sum_{j=1}^{n_{RT_i}} d_{RT_j} \tag{3.11}$$

where n_{RT_i} represents the Round Trips that the intermediate node/data mule has to complete between the two regions, a local and a remote one, in order to retrieve content object i, while d_{RT_j} represents the duration of each round trip j.

Therefore, the total retrieval delay for P content objects can be calculated as:

$$Dp_{total} = \sum_{i=1}^{P} Dp_i = \sum_{i=1}^{P} \sum_{j=1}^{n_{RT_i}} d_{RT_j} \tag{3.12}$$

In this case:

$$Dp_{total} = P * n_{RT_{avg}} * d_{RT_{avg}} \tag{3.13}$$

Essentially, Dp_{total} depends on the total number of content objects that were retrieved from the Producer (P), the average number of Round Trips for each object $(n_{RT_{avg}})$ and the average duration of each round trip $(d_{RT_{avg}})$.

Using equation (3.13), we derive:

$$Dp_{avg} = n_{RT_{avg}} * d_{RT_{avg}} \tag{3.14}$$

In our experiments, $d_{RT_{avg}}$ is influenced by the mule's *Contact Duration* (d_C) parameter, and also by the total number of alternating states of the mule (non-connected, or connected to one of the available *ad hoc* networks). As those are experiment-specific parameters, the relationship between them is presented on the following sub-section.

Another parameter needed to calculate Dp_{total} is $n_{RT_{avg}}$ - namely, the average number of Round Trips performed in order to retrieve a content object. To calculate the model values presented in the results, we either use the parameter values which are readily obtained by our experiments or, alternatively, estimate its value based on various factors (e.g., the *Delivery Probability*). The expected value for $n_{RT_{avg}}$ is computed based on the properties of geometric distribution: Assuming that each time the mule performs a round trip, the trip is either a success - when a content object is returned - or a failure. Hence, it matches the properties of Bernoulli trials and consequently the probability for the mule to perform k Round Trips to retrieve a content object is described adequately by a geometric distribution:

$$Pr(n_{RT} = k) = (1-p)^{(k-1)} * p \tag{3.15}$$

for $k = 1, 2, 3...$, where p is the probability of content retrieval on a single Round Trip. Therefore, the relationship between RT and p is given by:

$$n_{RT_{avg}} = E(n_{RT}) = \frac{1}{p} \tag{3.16}$$

3.5.3 Deployment- and implementation-specific behavior

One main goal of our work was to approximate the behavior of certain parameters inline with our specific experiments. Specifically, we were interested in approximating the value of the Dp and correlate its behavior with other network parameters appearing in our experiments, as Dp appears in most of the previous equations and influences both the D_{avg} and OR metrics (either directly, or indirectly). We note that some of the correlations found between the metrics and other parameters (i.e. the dependent variables and the independent ones) are specific to our experimental setup. For instance, the approximation of T_{int} is valid as long as Interests are re-transmitted once their Lifetime expires; when this assumption is violated (e.g. in NoD-Discovery), the equations (3.17) and (3.18) no longer describes system behavior.

Part of the work in this section was influenced by empirical results: equations (3.24) and (3.25) were derived from the experimental results using curve fitting and regression techniques and are implementation-dependent. This is also true for the specific parameter values $(\alpha, \beta, \gamma, \delta, m)$ which are valid for our deployment settings, but may vary when settings differ. Consequently, even though our system performance can indeed be accurately predicted using these equations within the range of

system parameters used in the experiments, the accuracy may drop when the independent variable values lie outside this range.

Transmitted Interests and Overhead Ratio

The default mechanism used in our experiments relies on the *Overhead Ratio* metric. The number of Interests expected to be transmitted towards the intermittently-connected Producer using the DTN face (T_{int}) is approximated. Those Interest packets are not cached in the local cache and hence they typically increase network overhead, since they may need to be eventually re-transmitted if the Interest Lifetime cannot accommodate the delay of the DTN part of the network. Instead, Interests answered by the NoD gateway are assumed to have yielded a cache hit and, hence, they do not contribute to our target investigation and the associated equation(s). Consequently, those Interests are not included in T_{int}.

In this context, the value of T_{int} can be approximated as:

$$T_{int} = P * (1 + \frac{Dp_{avg}}{L_{int}}) \tag{3.17}$$

or alternatively:

$$T_{int} = (1 - \frac{h}{ISR}) * R * (1 + \frac{Dp_{avg}}{L_{int}}) \tag{3.18}$$

Substituting T_{int} (using equation (3.10)) into equation (3.8), we end up with the following approximation for the *Overhead Ratio* of our Default mechanism:

$$OR = (1 - \frac{h}{ISR}) * (\frac{D_{p_{avg}}}{L_{int}}) \tag{3.19}$$

But $OR = \frac{1}{ISR} - 1$ (details included in Appendix A.2) and hence we approximate the expected value of *ISR* for NoD-Default as:

$$ISR = \frac{(1-h) * Dp_{avg}}{h * Dp_{avg} + L_{int}} \tag{3.20}$$

Producer delay for a single data mule

In our experiments, the intermediate nodes alternate between some predetermined states. The muling node stays at each state for a certain duration. This duration is either fixed throughout the experiment, or chosen randomly among a range of possible scales. Either way, the mule alternates between a fixed number of states before returning back to the Consumer. Therefore, the round trip for a certain content object equals to:

$$d_{RT_{ij}} = \sum_{k=1}^{states} d_{S_{ijk}} \tag{3.21}$$

Since the number of states on each round trip is fixed in each experiment, we can calculate d_{RT} using:

$$d_{RT_{avg}} = states * d_{S_{avg}} \tag{3.22}$$

In some experiments, the value of *states* is set to four (4), while in others its value is set to eight (8). The value of $d_{S_{avg}}$ acts as an independent variable in the experiments and is either fixed, or can be easily calculated, in each sub-experiment.

The average number of muling Round Trips (n_{RT}) needed to retrieve a content object depends on p, the probability of delivering a content object within a single muling round-trip (see equation (3.16)). Therefore, the value of n_{RT} can be approximated using p in each experiment. Given that this probability is influenced by various implementation-specific parameters, we focus our efforts at this stage on the relation between p and the three independent variables used in our experiments: 1) number of packets-per-contact (n_{ppc})[7], 2) *Contact Duration* (d_C) and 3) Interest Lifetime (L_{int}).

We can, intuitively, expect p to decrease: (i) as more packets are exchanged during a contact and (ii) as the *Contact Duration* between two nodes decreases. The former is true when n_{ppc} is larger or L_{int} is smaller, due to the Producer sending more Interest packets. The latter is true when the *State Duration* decreases in the experiments, as the two parameters, *Contact* and *State Duration*, have identical values in our setup ($d_C = d_S$).

We first correlate p with d_C *Contact Duration*. For larger *Contact Duration*, the probability should approach its maximum value, while for smaller *Contact Duration* the probability should become negligible. In any case, the probability should lie within the range [0,1]. In our model, we capture this relationship using a sigmoid function:

$$p = \frac{p_{max}}{1 + e^{-(m*d_C+b)}} \tag{3.23}$$

where $0 \leq p \leq 1$, $0 < p_{max} \leq 1$ and $0 < m$.

Parameters p_{max}, m and b are implementation-dependent parameters. When d_C is large enough, p will reach its maximum value. Theoretically, p_{max} should be equal to one (1) and packets would always be delivered as $d_C \to \infty$. However, in a real-world deployment, the value is lower than one; we approximate its value later in the section. Regarding the value of the other parameters, we found that $m = 0.8$ works well within our context, while b depends on the number of Interest packets sent by the Consumer (n_{ppc}).

p_{max} depends on n_{ppc} and obtains its maximum value for $n_{ppc} = 1$, decreasing as more Interest packets are sent, simultaneously, by the Consumer. We use that last observation to incorporate the influence of n_{ppc} into our model.

Aiming at sufficient approximations that match our results, we use linear and non-linear regression to find the right curves that fit our experimental data. Towards this end, we arrived at the following, empirical, equations:

$$p_{max} = \alpha - \beta * \log_2(n_{ppc}) \tag{3.24}$$

$$b = \gamma + \delta * n_{ppc} \tag{3.25}$$

The specific values of α, β, γ and δ (which are valid for our deployment setup), are: $\alpha = 0.86$, $\beta = 0.02$, $\gamma = -0.2$, $\delta = -3.3$.

The equations above can be used in data muling scenarios, when a single mule is used to transfer the data. In this case, the model presented above can give an accurate estimate of the expected delay in our experiments.

[7] For mathematical convenience, n_{ppc} and ppc are used interchangeably.

Equations (3.24) and (3.25) can provide an insight on what is the effect of the n_{ppc} variable on the possibility of delivery. Increased n_{ppc} values cause p_{max} to decrease (see equation (3.24)) and, as a result, p_{max} also decreases. Consequently, the delivery probability of a packet within a single round trip, for a given *Contact Duration*, decreases. Furthermore, larger n_{ppc} values also cause b to increase (see equation (3.25)). As a result, the sigmoid function $p(d_C)$ used in equation (3.23) is shifted to the right. What this means, qualitatively, is that more packets (n_{ppc}) will need larger d_C to achieve an adequate probability of delivery (p).

Chapter 4

Attack mitigation using reputation mechanisms

In this Chapter, we present our work towards the enhancement of the NDN trust and security fabric with reputation mechanisms. To this end, this dissertation's contributions are the following:

- We survey the existing literature and propose reputation-based trust as a general, complementary way to achieve trust in NDN.
- We identify two relevant security use cases in which reputation can be utilized, namely Bitrate Oscillation Attacks and Content Poisoning Attacks.
- We design two concrete, specific mechanisms to achieve attack mitigation (one for each use case).
- We evaluate their performance and assess the overall effectiveness of our approach.

Our motivation stems from two main observations. First, that cryptographic operations - especially signature verification - are computationally heavy, which renders routers particularly vulnerable as they cannot use the default NDN trust mechanisms. This increases the attack surface of the entire architecture, an aspect which has been detailed in Section 2.4.1.

Second, the fact that the policy-based approach currently adopted by NDN, relates content objects to provider certificates. This allows Consumers to determine whether individual content objects are deemed trustworthy and subsequently accept or reject the corresponding Data packets. However, once a Data packet has been signed with a valid certificate and has not been tampered while in transit, the packet is considered as trustworthy and accepted by Consumers regardless of the sender's prior behavior. We argue that this is not a rational choice if the sender has been previously sending/forwarding malicious content. Furthermore, a a legitimate certificate and valid signature are not sufficient to infer trustworthiness in case the Producer device has been compromised, as the content itself may be tampered at the source.

Thus, we argue that an important shortcoming of the default NDN trust framework is that it does not provide any native mechanisms to infer a network entity's trustworthiness (e.g. Producer, router) based on its actual behavior. Consequently, there are also no means to adapt to such circumstances to mitigate attacks, for instance by having Consumer applications adjusting their content requests or router devices their forwarding/caching mechanisms.

The introduction of reputation-based trust can solve those issues. Reputation values do not have to be tied to content objects, but can also be assigned to providers.

This allows the use of reputation scores by Consumers and intermediate routers to proactively infer the trustworthiness of content providers and avoid transactions with third parties not deemed trustworthy, as well as to reactively adapt to sudden changes in behavior (e.g. a previously trustworthy entity behaving malevolently and vice-versa).

Furthermore, those reputation scores can now be leveraged by routers to perform selective forwarding and/or caching. This way, routers can be effectively secured without performing signature verification and assist in attack mitigation by designing appropriate forwarding and/or caching strategies.

In this dissertation, we build on those observations and investigate the use of reputation for attack mitigation. Two different use cases are considered: Adaptive Multimedia Streaming using the DAS protocol and Peer-to-Peer (P2P) content distribution. Bitrate Oscillation Attacks (BOA) are launched in the former case and Content Poisoning Attacks in the second case. For each one of the use cases, we develop a distinct reputation mechanism to mitigate the attacks. Those mechanisms follow different approaches:

1) The reputation mechanism for DAS is centered on routers. Routers assign reputation values to the disseminated content, enabling the detection of compromised namespaces. They adjust their caching policies according to the received content and the reputation of its namespace. This design essentially focuses on *reactive* attack mitigation.

2) The reputation mechanism for P2P content distribution is centered on the content producers. End users assign reputation values to the content sources based on the content they receive, enabling the detection of malicious activity. Both the end users and routers can subsequently take action based on that information. In our case, end user applications avoid requesting content from non-trustworthy peers and a reward system is employed to incentivize honest user behavior. This design mostly focuses on *proactive* attack mitigation, but can also be used for reactive action.

Both of our frameworks use decentralized reputation mechanisms, as they are better-suited to the use cases and can avoid the overhead of a centralized reputation system. While in the former use case our approach employs a typical reputation system, in the later use case we also experiment with the use of blockchain to support decentralized reputation.

The reason for this is that, on one hand, blockchain provides a solid basis for decentralized reputation, as consensus is guaranteed by the blockchain and all ratings are transparent and auditable by any user. Consequently, not only the network but also the reputation system itself is secured from attacks. On the other hand, the crypto-economic aspects of blockchain enable the integration of a reward system, which can be used to achieve proactive attack mitigation by encouraging honest user behavior.

At this point, we simply note that the use of reputation systems also open up other possibilities which are not explored in this dissertation. For instance, reputation systems can also be used to assess the quality of the content itself, independent of whether the packet signature is valid or not. The dissemination of such information opens up new capabilities, such as the detection of unexpected or malevolent behaviors in which faulty data is included at the payload (e.g., malfunctioning sensors sending corrupted data).

Based on the above, we posit that reputation-based trust can complement the existing credential-based trust mechanisms in NDN. We do not recommend that the credential-based mechanisms in place are entirely substituted with reputation, as the two approaches are vertical and can co-exist in a common trust management

framework. In this setting, hybrid approaches are also feasible. For instance, the ratings collected by the reputation system to derive the reputation scores can be automatically generated by a Consumer application, depending on the outcome of the signature verification process.

In the rest of the Chapter, we present the relevant use cases and describe our reputation-based trust mechanisms. It is divided in two parts, each one devoted to the corresponding use case/mechanism: a) Bitrate Oscillation Attack (BOA) mitigation in NDN-DAS and b) Content Poisoning Attack (CPA) mitigation using blockchain.

4.1 Bitrate Oscillation Attacks on Dynamic Adaptive Streaming

4.1.1 Overview

The massive growth of traffic, witnessed during the last few years, calls for novel architectural solutions to better assist in content distribution. By grafting content at the core of its design, Information Centric Networking (ICN), and especially the Named Data Networking (NDN) paradigm [21], NDN appears to be beneficial for resource-consuming services such as multimedia streaming [139], particularly when investigated in combination with the Dynamic Adaptive Streaming (DAS) standard [200], [201]. One of NDN's fundamental features is in-network caching by routers. Although this feature contributes to reduced overall latency and bandwidth requirements, it also raises new security issues.

In this Section, we investigate the use of a reputation-based trust mechanism as means to complement NDN's current credential-based security in multimedia streaming. More specifically, we examine the mitigation of Bitrate Oscillation Attacks (BOAs) [145], which are launched during video streaming using the DAS protocol over NDN. In BOAs, the adversary forces a benign DAS client to experience redundant bitrate oscillations during video streaming and eventually, to perceive degraded Quality of Experience (QoE) [146].

Traditional NDN security schemes fail to safeguard the network from BOAs. Although some mitigation methods have been proposed, they either do not prevent the attack [145], or require continuous network monitoring which introduces additional signalling overhead [202]. Motivated by this, we propose and implement a reputation-based trust mechanism. This mechanism overcomes the vulnerabilities identified in previous methods and can assist NDN in the mitigation of the attack's impact. Moreover, we experiment with an on-off attack model [179], [203] aiming to evaluate our approach under realistic conditions. The results show that our technique efficiently shields the network from BOAs [204].

By doing so, we:

- Evidently prove that reputation-based trust can be efficiently utilized in a specific use case and argue that reinforcing the overall NDN's security framework by leveraging reputation-based trust, is feasible.
- Enhance an existing reputation-based trust protocol and evaluate its efficiency in a multimedia streaming scenario. Our refined model, thanks to our modifications, is also resistant against Sybil attacks.

- Move beyond the previous state-of-the-art regarding BOA mitigation, as we consider an on-off attack model, which is more complicated and realistic. Moreover, our approach suppresses ongoing attacks without introducing communication overhead.

4.1.2 System design

Overview

The fundamental idea of our reputation-based approach is to allow NDN routers adjust their caching policy. More specifically, routers compute a reputation value for each multimedia content object based on the level of consecutiveness of its received segments. By relying on the inferred reputation values, routers decide whether they will cache the following segments of the specific multimedia content or not.

Our reputation-based model is divided in four distinct phases:

1. The rating process of a specific multimedia file is initialized upon the arrival of an MPD file at the router. Once a router receives a multimedia content object's MPD file, it firstly checks whether it is the first time that the specific content is received. If so, the router, thereafter, assigns a default initial reputation value to it. In other words, routers keep a record of the multimedia content that has passed through them, along with their corresponding reputation values

2. Once the MPD file has been parsed, the following data packets received, are expected to concern the segments of the multimedia content. Upon each segment receipt, the router checks if there is a gap between the segment that just arrived and the latest received segment, by comparing their segment numbers. The router provides a negative rating if a gap is detected, or a positive one, otherwise

3. A certain time interval is defined and once it expires, the segment ratings calculated during this time interval are aggregated. The updated reputation value is computed by considering both the current ratings' aggregation result and the past reputation value. Therefore, the multimedia content's reputation value is updated on a periodic basis.

4. By relying on the derived reputation values, routers make suitable caching decisions, determining whether the upcoming segments should be cached or excluded from routers' caches.

The above-described design is captured in Algorithm 2. It is important to note here that we propose a decentralized framework where no coordination between routers is needed, i.e. each router calculates and stores its individual reputation values. By making this design choice, we avoid the undesirable overhead introduced by an exchange protocol required in a centralized framework [205].

As it is implied by the design analysis, the effectiveness of our model is premised on two key features: the proper calculation of the reputation values and the adaptation of a suitable caching policy in order to weaken the attacker. In the following subsections we provide a thorough description of those two features.

Reputation Values

Regarding the updating process of the reputation values, we have adopted the rationale of the reputation-based trust protocol, presented in [177]. Nonetheless, we

Algorithm 2 Reputation-based trust scheme for DAS.

```
Input: Incoming Data Packet
if Incoming Data packet corresponds to MPD file then
    if Multimedia file has not been received before then
        reputation value ← default initial reputation value
        Record content & its reputation value;
    end if
else
    if Incoming Data packet corresponds to video segment then
        Rate the segment;
    end if
end if
if Time interval [t − 1, t] expired then
    for each content object do
        for j = t − 1 to j = t do
            Aggregate segment ratings;
        end for
        Update content reputation value;
        Adjust caching policy;
    end for
end if
```

have made any necessary adjustments so that the initial primitives become suitable for our BOA mitigation method.

At this point, it is noteworthy that our refinements have rendered the used reputation-based protocol resistant against Sybil attack [203]. In the initial protocol, which is designed for cache poisoning attack mitigation, every router assigns a reputation value to each of its adjacent routers. By relying on the inferred reputation values, routers choose their well-regarded neighbors as the next hops towards which they forward Interests, and thus, malicious routers are temporarily excluded from the transmission path. However, an attacker can act by creating multiple identities, which are attached to multiple routers, at least one of which is likely to be included in the transmission path, and hence accomplish to inject poisonous content in the network. Unlike [177], we have chosen to assign reputation values to the disseminated multimedia content itself instead of the interface where malicious requests come from. That way, our model becomes invulnerable to Sybil attacks.

We define three modified metrics which replace the initial ones defined in [177]: the *skipping ratio*, the *rating* and the *reputation value*.

Skipping ratio. We define the *skipping ratio* $(SR(C_i))$ as the ratio of the number of detected gaps between received segments of the certain multimedia content to the segment number of the last received segment. Therefore, the skipping ratio can be calculated by:

$$SR(C_i) = \frac{G(C_i)}{L(C_i)} \tag{4.1}$$

In this equation, $C_i, G(C_i)$ and $L(C_i)$ denote the multimedia content object, the number of detected gaps and the segment number of the last received segment, respectively. We use the segment number of the last received segment as the denominator because it specifies the total number of segments that would have been received if there were no gaps. The skipping ratio metric, consequently, indicates the

frequency of the detected gaps.

Rating. Upon the receipt of a new segment, there are two potential scenarios. In the first one, the segment number of the received segment is the expected one, i.e. received segments are consecutive. In the second case, the received segment number is greater than expected, i.e. some segments have been skipped.

Based on this assumption, the router rates the received segment either positively or negatively. Similar to [177], routers utilize a rating system which follows the "linear increase, exponential decrease" principle. The skipping ratio metric, is factored into the calculation of the *rating* in time t, as indicated by the following equation:

$$R(C_i) = \begin{cases} 1 & \text{if segments are consecutive} \\ -\theta e^{\alpha}(1 - SR(C_i) & \text{if a gap is detected} \end{cases} \tag{4.2}$$

As described in [177], θ is a penalty factor representing the strength of the punishment and α is an adjustment factor.

We have adopted this rating system in order to detect content which causes suspicious oscillations and thus, it is likely that the specific content is being requested by an attacker. The computed rating is included in the calculation of the updated *reputation value*, increasing it if there no gaps have been detected or decreasing it otherwise.

Reputation value. The *reputation value* $(RV(t))$ represents an assessment of the trustworthiness of a certain multimedia file. In particular, a low *reputation value* means that segments are usually skipped and thus, it is possible for an adversary to utilize the content as weapon, while high reputation scores signal requests of consecutive segments. The updated *reputation value* is calculated by:

$$RV(t) = \delta RV(t-1) + \sum_{1}^{N} R(C_i) \tag{4.3}$$

In Equation 4.3, N denotes the number of segments received in the time interval $[t-1\ ,\ t]$. It is, also, clear that we incorporate both the past and the current inference regarding the multimedia content's trustworthiness. By taking into account both the past (multiplied by a time decay factor δ) and current inferences, the mechanism becomes capable of lessening the impact of on-off attacks [179].

By relying on the *reputation value*, each router decides whether it should allow for caching the incoming video segments or not.

Caching Policies

Based on the *reputation values*, routers determine the caching policy they will follow. We examine three different potential caching policies:

- **Threshold-based.** The derived *reputation values* are being compared to a predefined threshold value T. In the beginning, the caching of segments is not granted. In order for the segments to get cached, the content's *reputation value* needs to reach the threshold. In other words, the caching of video segments is enabled as soon as the content gains the reputation of the routers.

$$Caching(C_i) = \begin{cases} Permitted & \text{if } RV(t) \geq T \\ Denied & \text{if } RV(t) < T \end{cases} \tag{4.4}$$

However, exceeding the threshold value does not guarantee caching the rest of the segments. If punishing the multimedia content according to Equation 4.2 results in an updated *reputation value* value below the threshold, then the caching of the content segments is explicitly disabled until the *reputation value* recovers.

- **Probabilistic.** In that case, a drop probability (DP) tied to the *reputation value* is introduced. As the *reputation value* decreases, the drop probability increases.

$$DP(t) = 10 - RV(t)$$

The drop probability defines how likely it is for the incoming segments to get accepted to routers' caches.

- **Hybrid.** In this approach we define two threshold values, the lower (LT) and the upper (UT) threshold. When a segment is received there are three potential cases:

1. The *reputation value* is below the lower threshold.

$$RV(t) < LT$$

2. The *reputation value* is equal to or greater than the lower threshold and less than the upper threshold.

$$LT \leq RV(t) < UT$$

3. The *reputation value* is equal to or greater than the upper threshold.

$$UT \leq RV(t)$$

If the *reputation value* is less than the lower threshold value, then the router explicitly avoids caching the video segments. If the *reputation value* falls under the second case, then the router decides to cache or not to cache the incoming segments according to a drop probability (the lower the *reputation value*, the higher the drop probability). In the last case, routers cache all incoming segments without exception.

4.2 Content Poisoning Attacks in Peer-to-Peer Networks

4.2.1 Overview

In this Section, we present our approach for Content Poisoning Attack mitigation in a Peer-to-Peer (P2P) content distribution scenario. This is achieved by utilizing a blockchain-based reward system, which we enhance with a decentralized reputation mechanism. The resulting system can act against CPAs by a) enabling the identification of malevolent users and b) providing financial incentives for honest behavior.

Our work builds on a recent research direction which leverages cryptocurrencies as an incentive for users to perform useful work. Such approaches reward users contributing to the execution of a specific task in a Peer-to-Peer (P2P) network. Motivated by these approaches, our work focuses on P2P file sharing and assesses

whether a decentralized reputation and rewards system can provide sufficient incentives for honest behavior in this setting.

One such system is Proof-of-Prestige (PoP) [171], which can run on top of Proof-of-Stake (PoS) blockchains or be implemented as a smart contract on Proof-of-Work. PoP leverages the notion of users getting rewards from the network for performing useful tasks. In this work, we use PoP as the basis for our reward system, as it is by-design resistant against frequent attacks in P2P reputation systems, such as Sybil and Collude attacks. To this end, we enhanced PoP with a reputation mechanism to achieve the following desired, overall behavior:

- Content ratings are provided and stored in the distributed ledger, once a content object is received.
- Those ratings are tied to the user which provided that content and used to calculate its reputation.
- Potential attackers can be identified using the reputation information.
- Once honest users detect attackers, they refrain from requesting their content. Consequently, content from malevolent users becomes less popular.
- Honest users which regularly provide useful work (i.e. trustworthy content) to their peers will obtain a higher prestige and, as a result, an increased chance of mining new coins. Malevolent users will follow the opposite direction.
- This way, malevolent users get inherently punished for such behavior and counter-incentives are effectively created. A proactive attack mitigation mechanism is formed as a result, since malevolent behavior can provide less gains than an honest one.

This way, content poisoning is mitigated by the joint action of several features being offered: a) the identification of potential attackers, b) the avoidance of sending requests for their content and c) the introduction of financial incentives which favour honest behavior.

The attacker identification capability is offered by the reputation system itself. This allows network nodes to adjust their behavior and avoid content issued by malicious users. What Proof-of-Prestige brings, as a blockchain-based reward system, are two additional features.

The first one stems from its original conception as a general means to encourage user activity, by rewarding those users which contribute useful work to the system. In our case we leverage this functionality to not only encourage activity, but also create incentives for trustworthy behavior. We achieve this by adding a means to infer malicious activity and only reward the transfer of legit/valid files. The second one is that our system not only rewards truly honest users, but does so in a decentralized way. That is, without coins being explicitly removed from attackers by a third party, global user behavior results by itself in the attackers accumulating less mining power and, eventually, less rewards.

To assess the feasibility of such a design in practice, we enhanced the Proof-of-Prestige [206] reward system with a novel reputation mechanism and subsequently evaluated its behavior in a CPA scenario, a process detailed in Sections 5 and 6.

In this scenario, end-user NDN devices form an overlay P2P network and are able to request content from each other. The feasibility of implementing overlay P2P networks using an NDN network layer has been demonstrated in prior works for

the BitTorrent and IPFS P2P systems [207] [208]. We also note the fact that the latest Ethereum implementations over NDN use a P2P overlay for peer discovery and block propagation [209], following the design of the original Ethereum implementation[1].

In this work, we make the assumption that a P2P overlay network has been previously formed and can support block propagation and content distribution using one of the available implementations. Our focus is placed on studying the behavior of the reputation-based reward system in the context of the overlay network, in a manner agnostic to the underlying network layer. As this reputation system is focused on the content sources themselves (mostly concerned with attacker identification and end-user behavior adaptation) this level of abstraction is more than sufficient for assessing the effectiveness of our mechanism.

Our designs only requires end-user nodes (i.e. P2P nodes which act both as both content Producers and Consumers) to support blockchain. As those nodes eventually reach consensus regarding reputation information, the available information is used to identify potential attackers and avoid requesting content from potentially malevolent nodes.

At this point, we clarify that content poisoning is an issue plaguing all content-based networks, including overlay P2P networks as well as ICNs. Our approach focuses on securing the overlay content distribution network namespaces from content pollution/poisoning attacks. This can can also secure the NDN namespaces, as the network layer uses application-layer names for forwarding and caching.

Our approach is built on the premise that peers will not request the content objects residing in a poisoned namespace. No requests for poisoned content means not only that this malicious content will not be received by the peers, but also that poisoned Data packets will not easily spread in a wide scale to NDN routers and their caches. As a result, our approach can not only mitigate content pollution, but also be potentially useful for the (indirect) mitigation of cache poisoning attacks.

Based on the above, the functionality provided by our system can be used as a building block for a larger framework which may include actions by other network-layer entities such as NDN routers. For instance, the authors of a recent work [212] use a similar mechanism to mitigate Content Poisoning Attacks: they identify notorious content sources and subsequently avoid any interactions with the respective namespace. However, this work is centered on NDN-layer actions and proposes two adaptive mechanisms a) *Forwarding adaptation* by access routers or Consumers, which aims to limit the flow of Interests for malicious content to the core network and b) *Caching adaptation* by core routers to shield their caches. Such an approach is shown to significantly decrease the percentage of Interests forwarded for notorious namespaces, as well as the percentage of poisonous Data packets being cached.

As this approach is vertical to the one described in this dissertation, they can be jointly used to prevent both cache and content poisoning attacks in multiple levels. For instance, the blockchain-based mechanism described in this dissertation can provide financial incentives for honest behavior [213] and a means to securely store and retrieve ratings and reputation scores, while a system such as the one described in [212] can obtain reputation information from the blockchain and adapt NDN forwarding/caching when other counter-measures do not suffice.

[1]Some earlier implementations did not include this layer in an attempt to simplify the design, by leveraging NDN's multicast capabilities [210] [211].

At this point, we simply note this potential use and refrain from elaborating further on possible extensions such as the one described above. For the rest of this dissertation, we focus on content pollution from an end-user/overlay network perspective and our efforts are placed on the development and evaluation of our blockchain-based reward system for CPA mitigation.

4.2.2 System design

In PoP, *prestige* is used to reward each user for performing useful work: a file transfer between two parties results in the receiver transferring a predetermined amount of prestige to the sender. However, prestige can be transferred to a user regardless of the actual content being sent; the only condition is that the receiver acknowledges the transfer. As a result, mislabeled or malicious content can still be disseminated in the network as there is no way to proactively identify it.

In this dissertation, we try to compensate for this fact by introducing a reputation system on top of PoP, which measures the trustworthiness of each user. In our approach, users can make informed decisions on who to trust and request content from, instead of blindly receiving content from anyone participating in the P2P network. This can help to proactively identify malicious users and avoid potentially malicious content by choosing alternate sources. Our goal is to assess whether this approach can indeed successfully a) identify malicious users and b) financially punish malicious behavior.

To integrate this functionality into PoP, we introduced the concept of *maliciousness*. Maliciousness acts as negative reputation and is a measure of each user's trustworthiness. The main idea of our approach lies on the fact that most users prefer to engage in transactions with trusted parties. Therefore, trustworthy users (i.e., users with lower maliciousness values associated with them) will participate in a higher number of transactions (content transfers) and, as a result, accumulate larger amounts of prestige than malicious ones. This, in turn, will cause trustworthy users to achieve a higher chance of mining new coins and, eventually, be financially rewarded over time for their honest behavior.

In our reputation system design, each time a sender transfers a file to a receiver, the receiver uploads an acknowledgement of the transaction to the ledger, along with a possible downvote for fake/malicious content.

We chose a downvoting approach, so that two or more Attackers cannot collude to increase each other's reputation. On our approach, colluding Attackers are only able to downvote other users, which will not have a significant effect on their own mining power. This is true for two reasons: a) Typically, Attackers are a minority and their votes will be outnumbered by the votes of Honest users. Therefore, in the worst case, the negative votes by Attackers may cause a few Honest users to be labeled as malicious, but most of the Honest users will continue to have low Maliciousness values. As a result, the Attackers will not have a better reputation than most Honest users and will not receive increased rewards. b) The mislabeling of Honest users as malicious will only last for a limited amount of time, as we consider a *maliciousness decay* factor which will cause their maliciousness value to drop over time, as long as they continue their honest behavior.

For the rest of the dissertation, a sender in a file transfer operation will also be called a *contributor*, while a receiver will be called a *beneficiary*, according to PoP

terminology [2].

Each user's maliciousness value ranges between 0 and 1 and is calculated based on the user's past behaviour as a contributor, using the following formula:

$$Maliciousness = \frac{FakeTransfers}{TotalTransfers} \tag{4.5}$$

This value is calculated and renewed each time the user takes part in a transaction as a contributor. When a user receives a file, he/she uploads an acknowledgment (increasing the sender's $TotalTransfers$). If the file was received by an Attacker and is deemed to be fake or malicious, the user downvotes the transaction, increasing the sender-Attacker's number of $FakeTransfers$, which ultimately results in an increase to the Attacker's $Maliciousness$. (see Equation 4.5)

All users start with an initial $Maliciousness$ value of zero (0), which is updated each time a file is sent to another user, based on the receiver's feedback. Beneficiaries request content from a certain user based on its maliciousness. The probability p that content will be requested from a certain user is:

$$p = 1 - Maliciousness \tag{4.6}$$

Therefore, the probability of a beneficiary requesting a content object from a contributor with maliciousness=1 is p=0. On the other hand, the probability that a beneficiary requests a content object from a user with maliciousness=0, is p=1.

As an initial downvote in the first transaction can cause a user to acquire a maliciousness value equal to 1 (see Equation 4.5), effectively rendering him unable to distribute content again, we also introduce a decay factor in our system. This decay factor reduces maliciousness over time, so that users which may have acted maliciously in the past - or were wrongly deemed malicious - can still contribute to the network in the future, as long as sufficient time has passed.

Each time a new block is forged, a validator for the block is elected and rewarded with newly-mined coins. The block validator is selected between all users, based on each user's prestige. The higher the prestige, the more the chances of that user being selected as the transaction validator. The selection is made using a weighted random choice function. In the original version of our mechanism (V1), each node's weight in the selection process is equal to its prestige ($Weight = Prestige$). In the alternate version of the selection algorithm (V2), the weights are equal to $Weight = Prestige - StaticValue$; the latter was conceived to favor users whose Prestige is higher than their StaticValue (expected to be the most active trustworthy users).

[2] In PoP, a user contributing useful work to the system is called a *contributor*, while a user benefiting from useful work performed by others, is called a *beneficiary*. In a P2P file sharing use case, useful work can be defined as the act of sending a file which is requested by another user.

Chapter 5

Evaluation Methodology

5.1 Content Retrieval from intermittently-connected IoT devices

This Section is concerned with the experimental evaluation of NoD. Our experimental methodology was dominated by the decision to perform real-world experiments instead of simulations, by deploying the NoD implementation in devices located in our lab. This choice introduces certain limitations and trade-offs to our experiments. For instance, the increased reliability of our results comes at the cost of scalability restrictions.

Before elaborating on this scenario, we note that the design of NoD was also validated in a separate setting. In the aforementioned setting, NoD was used to enable service deployment in a disrupted mobile edge computing scenario and its capability to support service deployment in challenged networks was demonstrated. However, since the aforementioned setting was only used for validation purposes and not concerned with performance evaluation, we chose to include any relevant information in a different Chapter (see Section 3.4.1).

5.1.1 Scenario

The scenario explored in the performance evaluation experiments is concerned with content retrieval from intermittently- connected IoT devices. In this scenario, measurements are requested from a remotely-located wireless sensor network and transferred to the requester using data mules. Our goal was to investigate the feasibility and efficiency of deploying NoD as an alternative to legacy DTN for remote IoT data retrieval.

From an NDN perspective, the setup consists of two outlying, intermittently-connected, NDN regions that are communicating using data mules. In the Consumer region, an NDN Consumer periodically attempts to retrieve sensor measurements from the Producer-side region. In the Producer region, WSN nodes collect sensor measurements independently and send them directly to the NDN Producer, over IEEE 802.15.4. The Producer node acts as an IoT Gateway and aggregates the available sensor measurements. Upon receiving an Interest packet, it replies back with the respective Data packet. NoD devices are used as gateways between the NDN and DTN network parts.

Upon receiving the requested Data packets, our Consumer node selects randomly a pool of measurement names and sends Interest packets, accordingly. If the requested content has been received at a previous exchange, it may be available at the Consumer-side cache. The procedure is repeated until the total number of required sensor measurements is received by the Consumer. Whenever an Interest's Lifetime expires, the Consumer transmits the Interest again.

In the **first round of experiments** (Experiments 1-5), a single dedicated data mule is deployed to follow a predetermined route towards the remote IoT devices. This way, we emulate a scenario with deterministic node mobility.

This set of experiments is focused on assessing the performance of our original NoD implementation, which will be referred to as *NoD-Default*. In particular, Experiments 1 and 2 were designed to compare the performance of NoD with a host-centric DTN implementation. The goal was to assess the performance gains which can be achieved by NDN's name-based content retrieval and in-network caching functionality in a delay-/disruption-tolerant networking environment.

Once the feasibility and meaningfulness of using an NDN over DTN stack was established, Experiments 3-5 were conducted to evaluate the impact of different parameters in NoD's behavior.

After assessing the behavior of NoD-Default in the first round of experiments, we use the insights gathered during the experimental process to create a second, improved version of NoD. This new version leverages cross-layer information to improve performance and will be referred to as *NoD-Discovery*.

The **second round of experiments** consists of Experiments 6-7 and was conducted to assess the performance of NoD-Discovery by comparing it against NoD-Default. This experimental round differentiates from the previous one in two ways: first, a second data mule is now deployed and second, randomness is introduced in the mules' mobility patterns to create opportunistic contacts between the nodes. Therefore, Experiments 1-5 are more akin to a fixed-route DTN setup, while 6-7 are closer to an opportunistic networking scenario.

We note that both rounds of experiments deploy the same topology and follow the same communication patterns, the only difference being the addition of a second data mule in the second round of experiments.

The validity of our mathematical model was also tested during the evaluation process, as it formed one of our major goals. To this end, we use the equations provided in Section 3.5 to derive the expected behavior of NoD and compare it with the experimental results.

In the rest of Section 5.1, we provide a description of the individual scenarios and experiments, the metrics used for the evaluation process, the experimentation tools and, finally, the experimental setup.

5.1.2 Tools

The experiments were carried out using the UMOBILE NDN/DTN implementation [214]. This implementation introduces the DTN tunneling functionality in NDN and is available for both Linux and Android devices. In this sub-section, we are solely going to provide details regarding the former one and avoid any reference to the Android version, as the Linux NDN/DTN implementation was used in our experiments.

This NDN/DTN implementation consists of two different tools, which have been integrated to form an NDN over DTN protocol stack: IBR-DTN is used for the DTN layer, while the default NDN implementation is used for the NDN layer. Modifications and enhancements are introduced to the NDN libraries to allow for communication with the underlying IBR-DTN daemon.

The original NDN implementation consists of a) the *ndn-cxx* library, which provides infrastructure facilities for the system, and b) *ndn*, which includes the actual forwarding daemon (NFD) and several related tools. Below, we list the modifications introduced in the original NDN implementation by NDN/DTN:

1) The *ndn-cxx* library was renamed to *ndn-cxx_umobile*. In this library, the *FaceURI* class was extended to handle DTN faces in the following ways:

- A "dtn" scheme was added. This scheme takes the endpointPrefix as the host name (e.g. dtn-node1) and the endpointAffix as the path name (e.g. /nfd)
- *DTNCanonizeProvider* was added to check whether a certain dtn URI is canonical.

2) The *ndn* library was renamed to *ndn-dtn*. The ndn-dtn daemon includes the following enhancements:

- A section was added to *FaceManager*, to initialize the DTN Face from the *nfd.conf* configuration file. The configuration allows to set the host and port for the ibrdtn daemon and the endpoint prefix and affix for ibrdtn. By default, the ibrdtn daemon host is the localhost, but a remote host could also be used. The *endpoint id* of the local dtn host is in the form of dtn://dtn-node1/nfd, where "dtn" is the scheme, "dtn-node1" is the endpointPrefix and nfd is the endpointAffix. nfd is the dtn application endpoint that the nfd daemon will subscribe to in order to receive ndn-related bundles.
- A *DtnFactory* class was created. This class is responsible for the creation and maintenance of the *DtnChannel*. When the daemon is initialized, a *DtnChannel* is created. Subsequent invocations of the *createChannel* function of the emphDtnChannel return the already created channel.
- *AsyncIbrDtnClient* was created to allow for asynchronous connection to the ibrdtn deamon. The client starts a new thread that runs in the background and is always connected to the dtn deamon. When a new bundle arrives, the client notifies the associated *DtnChannel*.
- A *DtnChannel* was created to receive and send data. The *DtnChannel* listens for incoming data and forwards them to the daemon through the relevant face. It also sends outgoing data using the relevant face.
  - Each *DtnChannel* creates an *AsyncIbrDtnClient* in the "listen" function and passes itself as a constructor argument. When a new bundle arrives the client calls the relevant callback at the *DtnChannel*. The client thread remains on for the entire lifetime of the channel and gets deleted in the channel destructor.
  - Handling of incoming bundles is queued in a global daemon event queue. Instead of *AsyncIbrDtnClient* calling directly the *DtnChannel* callback, it posts the task of running the callback with the appropriate arguments to the global I/O queue of the daemon. This way thread synchronization issues between the bundle receiving thread and the core nfd thread are avoided.
  - *DtnChannel* differs from the rest of the ndn channels in that it receives all bundles for all faces, even if bundles for the same destination have appeared earlier. Other ndn channels pass connected sockets to the relevant face, which then directly receives any other bundles that may arrive at this socket, bypassing the *DtnChannel*.

  - *DtnChannel* creates faces similarly to the way datagram channels work. The link service used is the *GenericLinkService* and the transport is a special *DtnTransport* described below. When the *processBundle* function is invoked by the scheduler, a face is created (or just retrieved if it already exists) and the bundle is passed on to the transport of the face.

- Created *DtnTransport* that inherits the *Transport* base class to send and receive bundles via the ibrdtn client.
  - A *receiveBundle* function was created. This function is invoked by the *DtnChannel* and accepts an incoming bundle. The function reads the payload from the incoming bundle, copies the DTN payload to an NDN block element and inserts the block element into a new NDN *Transport::Packet*. Finally, the *Transport::receive* function is called with the newly created transport packet, which propagates the packet into the NDN core for processing.
  - A virtual *doSend* function was implemented. This function receives a *Transport::Packet*, creates a bundle, and copies the contents of the packet into the bundle. The bundle destination address is set to the *remoteUri* of the face (a face is created for each remote peer). Finally, the ibrdtn bundle is sent through an ibrdtn Client. The client is created, connected to the ibrdtn daemon, fed the bundle and closed before the function exits. Contrary to the *AsyncIbrDtnClient*, which is constantly running in the background, the Client created for sending the bundle has only local scope.

5.1.3 Topology and setup

The reproduction of the envisioned scenario in our lab has been accomplished by creating the topology depicted in Figure 5.1. The IoT devices were deployed in close proximity due to the space constraints existent in a laboratory setup, thus the actual distance between each adjacent node was one meter.

In our experiments, the data mules do not physically move from one region to the other; instead, mobility is emulated by connecting and disconnecting to different wireless networks. By connecting and disconnecting to those networks, each data mule emulates some mobility pattern that corresponds to movements with predetermined waypoints. We applied both deterministic and probabilistic approaches to evaluate the performance of NoD stack in the particular IoT use-case.

To do so, we created three distinct *ad hoc* networks, each one corresponding to the particular communication link (e.g. Consumer to intermediate node, intermediate node to Producer, as well as an intermediate-to-intermediate node link in the case of two data mules) and we set up the intermediate nodes to be periodically connected to each *ad hoc* network. Between each connection, each intermediate node remains disconnected by completely deactivating the wireless interface. This was deemed sufficient to reproduce the effects of disruptions and delays in the higher layers of the networking stack (i.e., NDN and DTN).

A single data mule scenario is illustrated in Figure 5.1. In the first state, the intermediate node is connected to the Consumer node via the *Ad hoc network 1* (configured at channel 1). At the next state, the intermediate node is disconnected and finally, at the third state, is connected to the Producer node via the *Ad hoc network 2* (which is configured at channel 11). Regarding the opportunistic contact experiments which include two data mules, we added an intermediate-to-intermediate *ad hoc* network (configured at channel 6).

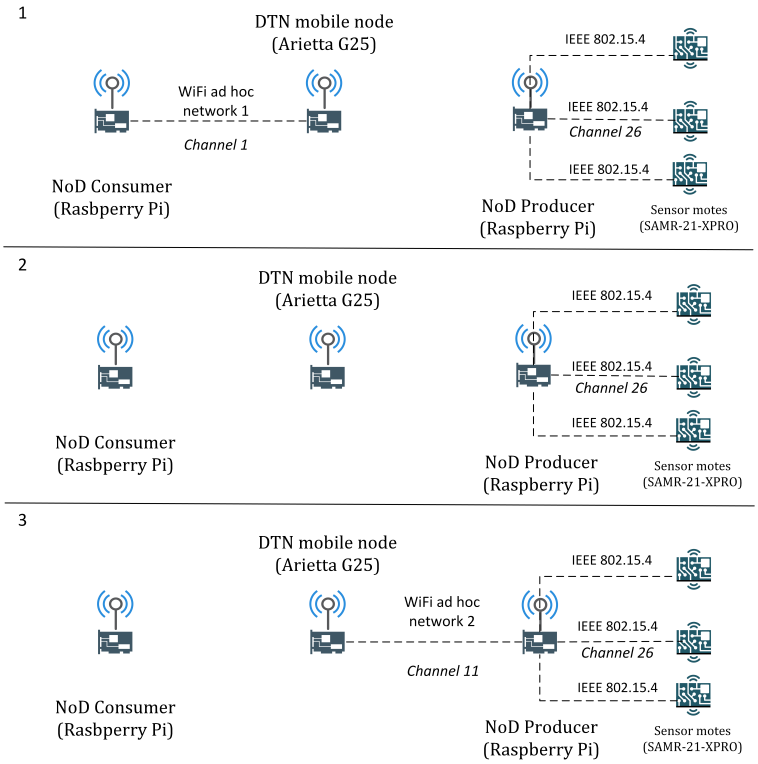


FIGURE 5.1: Experimental Topology

The wireless WiFi dongles were configured to operate in *ad hoc (ibss)* mode at 2.4GHz. Also, we selected 20MHz channel width and non-overlapping channels (channels 1, 6 and 11) for the three distinct *ad hoc* networks in order to avoid signal interference. Additionally, regarding the communication of the Wireless Sensor Nodes with the Producer node, we utilized the channel 26 of the IEEE 802.15.4, which do not interfere with the other IEEE 802.11 utilized channels.

The experimental topology comprises two network parts i.e., the Delay Tolerant Network and the WSN, given that the focus of this work is to assess the potential of NoD to retrieve content efficiently in intermittently-connected IoT networks. Details about each of those parts are subsequently provided.

Wireless Sensor Network (WSN) The WSN is constructed by three low-end IoT devices equipped with various sensors and communicating with the aggregator/ Producer node via 802.15.4. More precisely, the low-end devices are three SAMR-21-XPRO boards, which feature a 32-bit ARM Cortex-M0+ processor, clocked at 48MHz, with 32KB SRAM, 256KB flash memory storage and a fully compatible IEEE 802.15.4 radio transceiver. Additionally, we developed custom prototype boards that integrate temperature, humidity, light and hall sensors, that were all plugged to every WSN node. Every sensor node runs a UDP-based application over the lightweight RIOT OS [215]. The sensor node continuously collects sensor measurements; upon a new measurement, it sends a UDP packet to a known port of the data aggregator (Producer) and then resumes back measurement collection.

Delay tolerant network The Delay tolerant network accommodates three types of nodes:

1. **The Consumer node:** As Consumer node we used a Raspberry Pi 2B board, which features an ARM Cortex-A7 CPU clocked at 900MHz, 1GB of RAM and additional 32 GB of SD card storage. The board could run both NFD and IBR-DTN over the Linux Raspbian OS. Additionally, a WiFi USB dongle (TL-WN722N) was utilized to create the Consumer's *ad hoc* network.

2. **The Producer node:** A similar Raspberry Pi 2B board was also deployed as Producer node. Unlike the Consumer node, the Producer node has two distinct communication tasks i.e., to gather measurements from the WSN and to create an *ad hoc* network to communicate with the Delay tolerant network side. In order to address these two challenges, the Producer node was equipped with the MRF24J40MA transceiver, which is compatible with IEEE 802.15.4, and a WiFi USB dongle (TL-WN722N), which was used to create the *ad hoc* network. The measurements collected from the WSN were encapsulated as Data packets, whenever the corresponding Interest packets were received from the Producer.

3. **The intermediate node(s):** As a mobile node we used the Arietta G25 device. The Arietta G25 features AT91SAM9G25, clocked at 400MHz, 128 MB of RAM and uses additional 16 GB of SD card storage. Its connectivity is achieved by the support of a WiFi-compatible transceiver (WM1030WU). The device runs a lightweight version of Linux, specifically designed for Arietta with the buildroot tool, and supporting fully IBR-DTN. To emulate mobility patterns in our laboratory experiments, we activated and deactivated the device's WiFi interface using a bash script.

5.1.4 Experiment Configuration

In Section 5.1.4, we provide details regarding the configuration of each individual experiment. The experiments were conducted in two rounds and are grouped accordingly. In all of the experiments, the DTN Bundle Lifetime has been set to one (1) hour so that Bundles are always delivered and our results are not affected by the DTN layer. Along these lines, we have selected flooding as the DTN routing strategy.

The intermediate node remains connected to every *ad hoc* network for certain amount of time, so called *State Duration*. The intermediate node(s) can be in one of the available connectivity states. Four (4) possible states exist in Experiments 1-5, while eight (8) states exist in Experiments 6-7. The intermediate node's alternation between those states emulates a mobility pattern with predetermined waypoints.

Table 5.1 provides the overall picture with respect to the parameters used throughout the experimental process and how they vary in each experiment. Details regarding the individual experiments are provided in the following sections.

First Round of Experiments

In Experiments 1-5, we deploy a single data mule scenario with pre-determined routes and study the performance of NoD-Default. This round of experiments aims to investigate different aspects of NoD.

In these experiments, the intermediate node repeats the following four (4) *states*: (i) connected to Consumer's *ad hoc* network, (ii) disconnected from any *ad hoc* network on its way to the Producer, (iii) connected to Producer's *ad hoc* network and (iv) disconnected from any *ad hoc* network on its way back to the Consumer. Thus,

TABLE 5.1: Parameters used in each experiment

Experiment number	**#1**	**#2**	**#3**	**#4**	**#5**	**#6**	**#7**
Number of data mules	1	1	1	1	1	2	2
Content objects requested	10	10	10	20	varies [20, 40, 80, 160]	160	160
Available content objects	6	varies [5, 10, 15,..,40]	15	15	varies [15, 30, 60, 120]	120	120
State Duration	varies [20, 30,.., 60]	40	40	40	varies [5, 10, 15, 20]	rand (1,20)	rand (1,maxSD)
Maximum State Duration (maxSD)	-	-	-	-	-	20	{10, 20, 40, 80}
Interest Lifetime	1h	1h	varies [150, 160, ..., 210]	190s	1h	varies [10, 20, 40, 80, 160]	40s
Data Freshness	2h	2h	2h	varies [700, 1000, ..., 4300]	2h	2h	2h
Interest packets per contact	1	1	1	1	varies [1, 2, 4, 8]	8	8
Beacon Interval	5 s	5 s	5 s	5 s	1 s	1 s	1 s
Bundle Lifetime	1 h	1 h	1 h	1 h	1 h	1 h	1 h

the State Duration can represent either the duration of a disconnection, or the (*Contact Duration*) when two devices 'meet' one another (i.e., are connected to the same network).

Experiments 1 and 2 were carried out to study the performance of NoD-Default when compared to a DTN-only stack. In those experiments, the Consumer node requests 10 sensor measurements, one at a time. Each time, it chooses one out of six (6) available names (different sensors).

Experiments 3 and 4 were carried out to study the performance of the NoD architecture against different NDN-related parameters and compared the experimental results with with the expected ones based on our model. In both experiments, the State Duration was set to 40 seconds and the number of different prefixes to 15.

Experiment 5 was set up to assess the performance of NoD-Default performance in scenarios that involve short *Contact Duration* and simultaneous transmission of Interest packets for multiple content objects.

Experiment 1 - Different state duration values: In Experiment 1, we assess system performance for different, predetermined, values of the State Duration parameter, i.e. the time period that the mule remains in each state. Those values range from 10 to 60 seconds. In the NoD sub-experiments, NDN-related parameters (i.e. Interest Lifetime and Data Freshness Period) were all set at sufficiently-high values (one and two hours, respectively) to guarantee that no data were rejected due to the their influence. Each time, the consumer node chooses one of six available names (different sensors).

Experiment 2 - Different number of content objects: In Experiment 2, we varied the number of different content objects that are available for selection by the consumer. This way, we effectively modified the NDN-DTN Cache Hit Ratio (as a lower number of available names gives a higher chance of repeating the request in the future). We set the State Duration to 40 seconds, according to the results of Experiment 1, so that the number of muling Round Trips per measurement is minimized. Interest Lifetime was again set to one (1) hour.

Experiment 3 - Varying Interest Lifetime: In Experiment 3, we examine the impact of the Interest Lifetime values on the performance of NoD. The Freshness period was set to 2 hours.

Experiment 4 - Varying Data Freshness: In Experiment 4, we examine the impact of the Data Freshness values on the performance of NoD. Based on the results of Experiment 3, the Interest Lifetime value was set to 190 seconds.

Experiment 5 - Short contact duration and varying traffic volume: Experiment 5 differentiates from the previous experiments in two main aspects: First, the Consumer is configured to request multiple Interests at a time to speed up data retrieval, instead of sending a single Interest per contact (as was the case in Experiments 1-4). Second, the DTN discovery beacon interval value is set to 1 second, to allow for short connectivity scenarios. This is in contrast to Experiments 1-4 in which it was set to its default value (5 seconds), as those experiments used larger *State Duration* values.

The experiment starts with the Consumer sending a specific amount (1, 2, 4 or 8) of Interest packet(s). Those packets request objects which are randomly selected

from the respective available names (15, 30, 60 or 120). We choose to alter the available names for every sub-experiment in order to achieve the same *Cache Hit Ratio h* throughout the experiment and negate the influence of this parameter. Throughout the experiment, Interest Lifetime and Freshness Period were set to large values (1 and 2 hours respectively).

The goal is to investigate the performance of the NoD stack against a varying number of exchanged packets that could be sent per contact and the respective *State Duration*. *ISR* and *h* were equal to 1 and 0.4, respectively.

Second Round of Experiments: Multiple data mules

Experiments 6-7 were designed with the following goals: a) to assess the effectiveness of cross-layer interaction using our NoD-Discovery mechanism and b) to evaluate the performance of NoD when faced with opportunistic contacts.

In this second round of experiments, the topology slightly changes and an additional intermediate node is deployed, accounting for a total of two (2) intermediate nodes. Our goal is to evaluate the performance of NoD-Default against the proposed NoD-Discovery in an opportunistic environment with two intermediate nodes.

The DTN discovery beacon interval is set to one (1) second, similar to Experiment 5, to allow for short-connectivity scenarios. In addition, the NDN Freshness Period was chosen to be two (2) hours, in order to ensure the availability of previously requested Data packets in the Consumer's cache, during each experimental procedure. We conducted 10-20 iterations of each sub-experiment in order to ensure validity of the results.

In those experiments, the intermediate nodes remain connected for a certain number of seconds to each one of the following *states*: (i) connected to Consumer's *ad hoc* network, (ii) disconnected from any *ad hoc* network on its way to the intermediate, (iii) connected to intermediate's *ad hoc* network, (iv) disconnected from any *ad hoc* network on its way to the Producer, (v) connected to Producer's *ad hoc* network, (vi) disconnected from any *ad hoc* network on its way to the intermediate, (vii) connected to intermediate's *ad hoc* network, (viii) disconnected from any *ad hoc* network on its way back to the Consumer. Thus, those two intermediate nodes are alternating between eight (8) *states*.

The general communication pattern is the following: the Consumer sends a preselected amount of Interest packets and proceeds with next cycle of Interests only when all of the corresponding Data packets are received. Therefore, if some of the Interests expire, the Consumer only re-transmits the expired ones. The average bundle size was 165 bytes for an encapsulated Interest packet and 513 bytes for an encapsulated Data packet.

Experiment 6 - Varying Lifetime: In Experiment 6, every intermediate DTN node remains connected to each state for a duration randomly chosen within the range of 1 to 20 seconds. Thus, although the duration that a DTN node spends in each state differs, the average duration is constant throughout the experiment and equal to 10.5 seconds.

In this context, we investigate the validity and appropriateness of the stack in the presence of large numbers of exchanged packets-per-contact. The Consumer node transmits eight (8) randomly-selected Interest packets at each repetition. In the NoD-Default case, if the Lifetime of an Interest packet is timed-out, the Consumer instantly re-expresses the corresponding Interest packet unlike NoD-Discovery, where transmission and re-transmission occur only when a DTN node is available in the

network. Upon receipt of all the respective Data packets, the Consumer sends eight (8) new Interest packets, selecting randomly among 120 available prefixes. This procedure is repeated 20 times (i.e., until 160 Data packets are gathered from the Consumer).

The individual sub-experiments were conducted for various Interest Lifetime (L_{int}) values, in order to investigate whether our approach cancels the deficiency of redundant re-transmissions. Of course the amount of total sent packets inevitably exceeds the minimum number of exchanged packets (e.g. 160 Interest and 160 Data packets), given the intermittent connectivity and the low L_{int} values.

Experiment 7 - Varying Maximum State Duration: In Experiment 7 we create highly opportunistic contacts, by having data mules remain connected in every state for a random period of time which ranges from 1 to *Maximum State Duration* seconds. The intermediate node selects a random slot prior to connecting to the next state. The available number of *states* and data mules remain the same as in Experiment 6. The Interest Lifetime is 40 seconds, the number of different prefixes to 120 and the number of exchanged Interest packets-per-contact is eight (8). The Maximum State Duration of each sub-experiment ranges from 10 to 80 seconds and, thus, the Average State Duration also varies.

The experiment is designed to reflect the impact of short-lived Interests in an opportunistic environment with a large amount of exchanged packets-per-contact. The goal is to investigate the efficiency of NoD under particularly strained network conditions.

5.1.5 Metrics

The metrics we used to evaluate the performance of our scheme are the following:

- Average Number of Round Trips (n_{RT}): The average number of round trips needed by a data mule to fetch all the content from the Producer.
- Average Delay (D_{avg}): The average delay needed to retrieve all the content during an experiment. To retrieve this value, we calculate the average of all individual delays for each content object. A content object's delay is defined as the time elapsed from the first time that the content object was requested (i.e., the first Interest was sent) to the time that the corresponding content (Data packet) was received, to account for the total time in the case of packet loss and retransmission.
- Interest Satisfaction Ratio (ISR): The percentage of Interest packets which were satisfied during an experiment.
- Cache Hit Ratio (h): The percentage of cache hits.
- Content Retrieval Probability (p): The probability of receiving a content object on a given round trip.

- % of Data Received: The percentage of content objects that has been received (up to a given time [1], or on a certain round trip [2]).
- Caching Delay (D_c): The delay to retrieve a Data packet from the local cache (as opposed to retrieving it from the intermittently-connected Producer.
- Overhead Ratio (OR): The number of redundant Interests that need to be sent to receive a Data packet. [3]

In the cases that the model results are also used, they are calculated using parameter values obtained by the real-world experiments. For example, in an instance that the Producer Delay (Dp) was needed, the average delay for each DTN experiment was used as it provided us with the benchmark delay for a no-caching scenario. We then used the average Cache Hit Ratio (h) and Interest Satisfaction Ratio (ISR) values obtained by the results in each NoD sub-experiment to calculate the decrease in average delay theoretically achieved by a NoD implementation operating under the given network environment.

5.2 Impact of chunk size in NDN performance

5.2.1 Scenario

Motivated by our use of an increased chunk size during the validation of NoD in a service deployment use case (see Section 3.4.1), we proceeded to evaluate the impact of an increased chunk size on a more general setup.

The scenario in this section focuses on a fixed network. Our goal is to shed light on the effect of chunk size on generic NDN deployments, to conclude whether larger chunk sizes can be employed on a larger scale. Two individual experiments were conducted, each one introducing different levels of congestion. The first experiment is congestion-free, while the second one introduces mild congestion.

5.2.2 Tools

To realize this scenario in our experiments, we used the Named Data Networking Simulator (ndnSIM), version 2.7. ndnSIM is an open-source NDN simulator, implemented using the C++ programming language. It is based on the popular ns-3 networking simulator[4]. A first version of ndnSIM was released in 2012 [216] and its second version in 2015 [217].

[1] We remind the reader that the experiments run until a finite number of content objects has been received. In the case the metric represents the data received up to a given time, it can be perceived as quantifying transfer completion. For example, a value of 60% at 20 seconds means that 60% of the total Data requested during the experiment has been delivered to the Consumer at that point. In the same example, a value of 80% at 30 seconds means that 80% of the total Data has been delivered at the time. The values are, thus, cumulative.

[2] We also point out that the data are delivered to the Consumer in batches, each time the data mule returns to the Consumer-side network. In the case the value represents the data received on a certain round trip, a value of 60% for the 1st round trip means that 60% of the data has been received the first time a mule arrived at the Consumer-side NoD gateway. In the same example, a value of 20% for the second round trip means that another 20% was received the second time the mule arrived, and so on. This time, the values are not cumulative.

[3] For instance, in the case that 2 Interests were sent to retrieve a Data packet, OR=1. In the case that a single Interest was sent to retrieve a Data packet, OR=0.

[4] https://www.nsnam.org/

The difference between the two versions lies on the fact that while the first one was entirely ns3-based, ndnSIM 2 has integrated the libraries used by the original NDN implementation (namely, the ndn-cxx library and the NDN Forwarding Daemon). This way, it can support real-code experiments in a simulation environment with the trade-off of an increased memory and CPU consumption, when compared to ndnSIM 1.0. Still, large scale simulations of real-world NDN applications can be performed on general-purpose hardware, which was one of the main goals of its creators.

The simulator is implemented using a modular approach to allow for the independent replacement and/or alteration of any simulator component without that action impacting the operation of the other components. Different C++ classes are used to model the behavior of different NDN components, such as the various Faces which are used for communication other structures used by the NFD such as the Forwarding Information Base, Pending Interest Table and Content Store.

Its NDN stack is implemented as a network-layer protocol model and can run on top of any available link-layer protocol model (point-to-point, CSMA etc.). Moreover, ndnSIM includes a collection of helpers and tracers for component-level and traffic-flow behavior tracing purposes.

The selection of the NDN protocol stack as the Layer-3 protocol of choice on a simulated node, results in the creation of an NFD instance which is associated to the node. This has the effect that much of the code used for any experiments related to NDN forwarding operations can be readily used inside ndnSIM and vice versa. Moreover, its integration with *ndn-cxx* enables real-world applications written for the library to be simulated. The various components of ndnSIM are depicted in Figure 5.2, originally presented in the ndnSIM 2 paper [217].

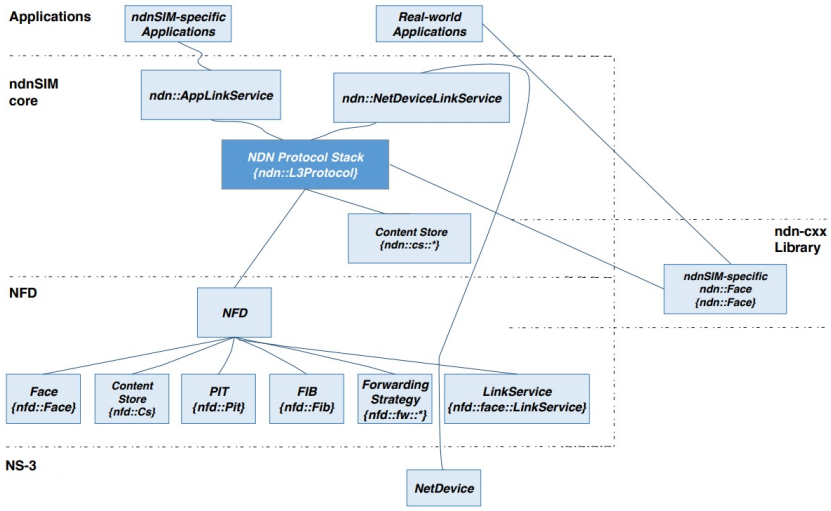


FIGURE 5.2: ndnSIM 2 structural components.

We note that during the experiments presented in this dissertation, the Old Content Store implementation was used, due to its support for a wider range of cache replacement policies.

5.2.3 Setup

A dumbbell topology was deployed, consisting of 6 network nodes: 2 Consumers, 2 Producers and 2 Routers. The Consumers were located on one side of the network and connected to one Router, while the Producers are located on the other side and connected to a different router; the two routers are connected via a backbone link.

The following simulator parameters were used: Point-to-point interconnection (P2P) links were set up between nodes. In the 'Congestion-Free' scenario, the Producer-Router and Consumer-Router links were set at 40 Mbps, while the backbone link (Router 1 - Router 2) was set to 160Mbps. In the 'Mild Congestion' scenario, the backbone bandwidth was reduced to 60 Mbps while the other ones remained at 40Mbps. The topology and settings of the two experiments are depicted in Figure 5.3.

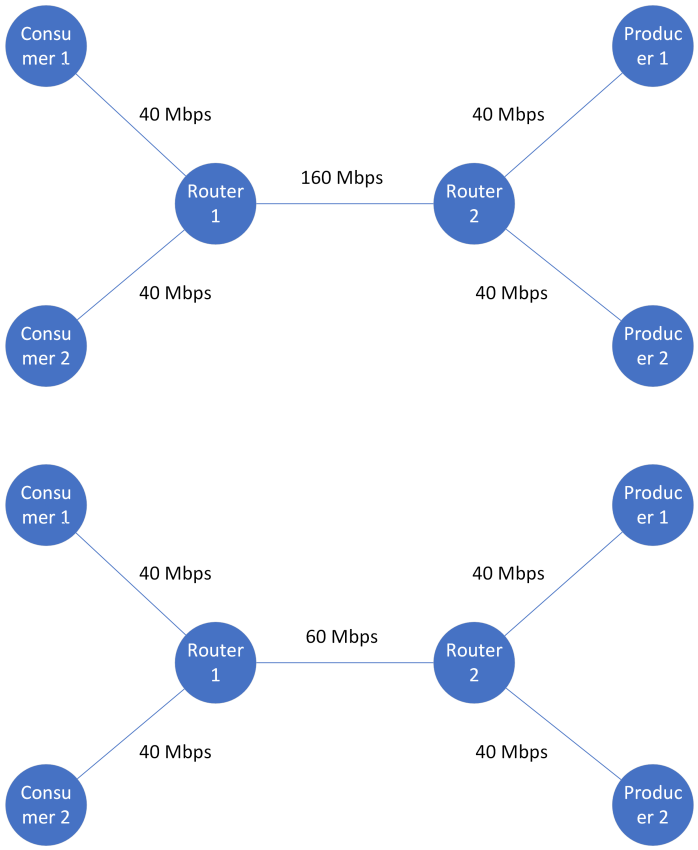


FIGURE 5.3: Experimental topology and link bandwidth.
Upper image: "Congestion-free"experiment.
Lower image: "Mild Congestion" experiment.

The propagation delay on all links was set at 10ms and kept constant throughout the experiments, while the cache size was set at 1000 content objects on each node, regardless of the size of each object. The ConsumerZipf application was used for Interest generation; the number different content names that could be selected by the application was set to 2000. Finally, the simulation time was set to 30 seconds.

During the simulations, we gradually increased the Data chunk size. To account for the payload increase, each time the chunk size was doubled, the application's Interest generation rate was halved. Thus, the total volume of data sent in all experiments was the same; this way we factored out the impact of payload size on

Packet Size [KB]	Interests [per second]
5	900
10	450
20	225
40	112
80	56
120	42
160	28

TABLE 5.2: Chunk size - Interest rate correlation

network throughput. The correspondence between the chunk size and number of Interests issued in each experiment is depicted in Table 5.2.

For each chunk size, we repeated the experiment using a different cache replacement policy and compared their results against each other. Overall, the following cache replacement policies were used: First In First Out (FIFO), Least Frequently Used (LFU), Least Recently Used (LRU), and Random Replacement (RANDOM).

5.2.4 Metrics

To assess the impact of chunk size on NDN performance we use the following metrics, calculating their average values for each experiment:

- Throughput
- Delay
- Cache Hit Ratio
- Cache Entries Lifetime

The first and last metrics were measured in the consumer-side Router, while the rest were obtained in each one of the Consumer nodes.

5.3 Bitrate Oscillation Attack mitigation

To assess the performance of our reputation mechanism for BOA mitigation in DAS over NDN, we consider a multimedia streaming scenario, as illustrated in Figure 5.4. The DAS-compatible multimedia data (S) is provided by a server, denoted as *producer* (P). S consists of n number of equal-length segments. Moreover, each segment is available in various representations, i.e., visual qualities. The *client* (C) streams each segment in the optimal representation. We consider that S is publicly available, and thus accessible by the adversary Adv. Furthermore, each request from Adv and C traverses one or more ICN routers before being satisfied by P or by an on-path CS. For adaptive bitrate streaming C, Adv and P utilize the DAS over ICN protocol, e.g., [201], [218].

5.3.1 Attack model

For the adversarial model, we first assume that Adv is attached to at least one on-path router between C and P. Existing techniques, such as the one presented in [219], could be used by the Adv to identify the router closest to the victim and simply

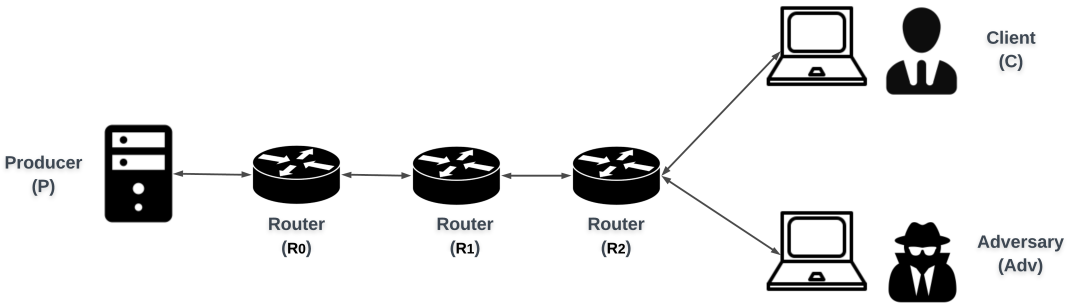


FIGURE 5.4: Scenario topology

connect to it. We also consider that *Adv* has prior knowledge of the video (S) that C is about to stream. This assumption is rational, since there are various techniques which can be used by the *Adv* to obtain this information. For instance, *Adv* could exploit timing attacks [220] and probe the MPD of the video, to discover S. Also, in case of identical wireless links, eavesdropping techniques [221] on the exposed traffic traces could be exploited as well, to analyze the online activities of C and predict S and its source location [222]. Last, we consider that the video is publicly available by Internet Service Providers (ISPs).

In Bitrate Oscillation Attacks (BOA), an *Adv* forces the C into frequently switching between high and low bitrates. The aim is to interfere with the adaptive behavior of DAS and result in a degradation of the perceived QoE. In particular,

At the beginning of the streaming session, the *Adv* obtains the MPD file containing the list of all available segments (S_n) and the associated representations[201]. To trigger oscillations for C, the *Adv* exploits the information included in the MPD to request a non-sequential subset of S. For each selected segment of S, *Adv* requests all the available representations to assure that the representation requested by C will be cached in an intermediate router. In particular, *Adv* issues a series of interests following an ascending order, i.e., $S(n+\gamma)$, where γ is the consecutive gap between the requested segments. As a result, when C subsequently requests all the segments of S, each segment has either been previously requested or skipped by the *Adv*. For instance, let us assume that a specific segment S_n requested by C, has been already requested by the *Adv* and a copy of S_n is then available in the CS of R_2. This segment would be delivered to C in the round-trip time between R_2 and C. In case that the earlier requested S_n has not yet reached R_2, the interest of C is aggregated at R_2, and it will be satisfied by R_2 in the round trip time a bit less than the round trip time between C and P. Finally, if S_n has been skipped by the *Adv*, C's interest will be forwarded all the way to P and S_n will be retrieved after the round trip time between P and C.

C, consequently, interprets short delivery times for the segments pre-issued by *Adv* as indicative of high network bandwidth availability. In contrast, C interprets longer delivery times for the remaining segments as indicative of lower bandwidth availability. Based on this assumption, by pre-issuing specific segments of S with the consecutive gaps, the *Adv* makes the DAS adaptation strategy at C to frequently switch between varying bitrates, e.g. from low to high. Therefore, the *Adv* accomplishes to cause unwanted bitrate oscillations to C, and eventually, degrade C's perceived QoE.

We need to note here that it is possible that a legitimate user also requests the non-sequential segments. However, as the requests of each of the benign users are

adapted to their individual network conditions, the chances of the representations of the segments requested by the two users coinciding, are extremely low. Therefore, many interests will be satisfied by the initial content provider. In contrast, our work focuses on the case in which an attacker intentionally mis-exploits NDN's native in-network caching and interest aggregation characteristics to inject all the representations of the non-consecutive segments in the network and thus, ensures that the victim will be forced to experience redundant oscillations.

5.3.2 Scenario

For evaluation purposes we have performed two different simulation experiments, investigating the influence of different factors in BOA mitigation.

In particular, *Experiment 1* evaluates the efficiency of our proposed reputation-based countermeasure by examining the fluctuations perceived by normal users due to BOAs which are launched by an adversary. We assume that intermediate routers have adopted a threshold-based caching policy and compare the performance of such a mitigation measure against the "no attack" and "on-off attack" cases. In this experiment, the caching threshold value is set to 7.

Experiment 2 explores the use of different caching policies and compare their performance. In particular, we experiment with three policies: a threshold-based policy, a probabilistic policy and a hybrid policy. The specifics can be found in Section 4.1.2. For the threshold-based policy, the threshold (T) value is set to 7. For the hybrid one, the upper threshold (UT) is set to 7 and the lower (LT) is set to 4.

5.3.3 Tools

In this section, we evaluate the effectiveness of our proposed reputation-based trust countermeasure for Bitrate Oscillation Attacks by using the AMuSt-ndnSIM simulator framework [223]. AMuSt-ndnSIM is a modified version of the Named Data Networking Simulator (ndnSIM [217]), which supports adaptive multimedia streaming over NDN by integrating the libdash software [224]. The latter is an open-source library of the MPEG-DASH standard.

In the rest of this subsection, we provide a high-level description of the integration of our reputation-based scheme in the NDN framework by outlining the functionality of each system component (corresponding to a C++ class) as well as their interrelations. In summary, our system model consists of the following applications:

- The multimedia producer application (*ndn-fileserver.cpp*)
- The bitrate oscillation attacker application (*ndn-boa-attacker.cpp*)
- The multimedia consumer application (*ndn-multimedia-consumer.cpp*)
- The router application (*ndn-router.cpp*)

Multimedia Producer: The AMuSt-ndnSIM codebase includes a file server (*ndn-fileserver.cpp*) application for hosting and providing DASH content. In our system, we use the file server application to implement the multimedia producer

Bitrate Oscillation Attacker: The bitrate oscillation attacker application was created in order to integrate the attack model in our framework. This application is based on the multimedia consumer application but several necessary modifications have been made in order to simulate a bitrate oscillation attacker's behavior. The main new features introduced in the bitrate oscillation attacker application concern the segments' download. First of all, the bitrate oscillation attacker application simulates an on-off BOA model. Moreover, the original multimedia consumer application has been modified so that the attacker requests and downloads all available representations of the video segments (instead of the most suitable one selected by the adaptation logic).

Multimedia Consumer: The multimedia consumer is provided by the AMuSt-ndnSIM codebase and enables requesting, downloading and consuming DASH content (i.e., parsing MPD files and streaming video segments). In our development, the multimedia consumer is also the parent class of the router class. Therefore, on the router nodes an instance of the multimedia consumer application runs along with an instance of the router application. The multimedia consumer application has been modified so that in the aforementioned case once the multimedia content's MPD file has been parsed, the multimedia consumer application does not proceed to downloading the video segments.

Router: We have created a router application which implements the reputation-based trust protocol. This application inherits from the multimedia consumer application in order to be capable of parsing the MPD files. The router application is responsible for providing ratings and updating the reputation values. Finally, the router application passes the reputation values to the NFD forwarder.

Furthermore, we have made some necessary modifications to the NFD forwarder (*forwarder.cpp*). In particular, we have added the *setReputationValueFromNdnSim* function via which the forwarder gets the latest reputation value. The reputation value is thereafter used to make caching decisions. The three examined caching policies (threshold-based, probabilistic and hybrid), have also been implemented in the NFD forwarder.

With the integration of the proposed reputation-based trust protocol and the overall system model in the NDN architecture, we have completed the development which feeds the evaluation-related activities and hence, the finalization of our feasibility study.

5.3.4 Setup

The topology considered in our simulations is the one depicted in Figure 5.4, while corresponding configurations are included in Table 5.3. Both the adversary and the legitimate multimedia consumer request the AVC-encoded Big Buck Bunny clip from the DASH/AVC Dataset [225]. The multimedia consumer uses the Rate-based adaptation logic (RB), i.e. he/she adapts the best possible representation based on throughput estimations. We have chosen this adaptation logic in order to maximize the attack impact [146]. Intermediate routers use the BestRoute forwarding strategy (i.e. they forward the Interests to the upstream with lowest routing cost) and the Least Recently Used (LRU) cache replacement policy to update the Content Stores' entries.

Furthermore, we assume that we have to deal with a rogue adversary who realizes that following the simple BOA model, i.e. simply skipping segments with a

TABLE 5.3: Simulation Configurations

Parameter	Value
Link Bandwidth	10 Mbps
Point-to-point link delay	2 ms
Link between Producer and edge router delay	350 ms
Content Store size	0 (unlimited)
Cache replacement policy	LRU
Forwarding Strategy	BestRoute
Video codec	AVC
No. of video segments	299
No. of available representations	20
Segment duration	2 s
Adaptation Logic	RB
Start up delay	0.1 s
Max. buffered seconds	30
Consecutive gap γ	5
Default initial reputation value	3.5
Penalty factor θ	0.5
Adjustment factor α	1
Time decay factor δ	0.75
Time interval $[t-1, t]$	100 s
On-off attack time interval	250 s

consecutive gap γ, would result in repeated punishments (according to Equation 4.2) and thus, a very low *reputation value*. In that case, incoming segments would be rejected from routers' caches and hence, the attack would fail. Therefore, the adversary attempts to launch an on-off attack, mixing normal and malicious behavior, aiming to render the attack hard to detect [203]. This assumption is reasonable, since on-off attacks are more realistic [179]. Hence, we assume that the adversary who launches the BOA, changes his/hers behavior from requesting consecutive segments for a certain period of time to requesting segments with a consecutive gap γ, and vice versa. In our simulations, we set this time interval to 250 seconds.

5.3.5 Metrics

The performance evaluation of our mechanism was inspired by previous works [226], [227] and focuses on the QoE perceived by the DAS client. To this end, the following metrics are used:

- Number of switches
- Average switch magnitude

The former metric is self-explanatory and refers to the total number of video quality switching events during an experiment. The latter metric refers to the average amplitude of video quality switches, effectively measuring the severity of each switch.

5.4 Content Poisoning Attack mitigation

This Section is concerned with the performance evaluation of our reputation-based incentive mechanism against Content Poisoning Attacks. To this end, we focus on quantifying the ability of our modified version of Proof-of-Prestige to reward honest users and punish potential attackers. The experimental process is presented in the following subsections and includes details regarding the simulation scenarios, tools, setup and metrics being used.

5.4.1 Scenario

Before evaluating the performance of our mechanism, we first performed some initial validation experiments to assess whether the mechanism exhibits the desired behavior. The questions which we tried to answer by performing these experiments are the following:

- Does our mechanism assign high maliciousness values to Attackers and low values to Honest users?
- Do the assigned maliciousness values result in lower prestige values for Attackers?
- Do lower prestige values also result in the Attackers mining less coins, when compared to Honest users?

Although we do not present the results of all validation simulations, certain examples which illustrate our mechanism's behavior and answer the aforementioned questions are indicatively included in Chapter 6.

Those initial experiments validated the fact that our approach achieved the proactive identification of Attackers, which resulted in the Attackers gaining less prestige and, eventually, less coins than Honest users. However, they did not demonstrate how consistent is this behavior on various scenarios. Once we established that our mechanism behaved in the desired way, we proceeded to perform two rounds of performance evaluation experiments.

In those experiments, we aimed to quantify the effectiveness of our mechanism and evaluated our system against two different attack models. The first model considered is a "Full-On" attack, in which the Attacker sends malicious content in every transaction. The second one is an "On-Off" attack, in which the Attacker tries to outwit the reputation system by alternating between legitimate and malevolent content transfers.

In the first round of evaluation experiments, we used Version 1 of the validator selection function to determine which node was going to mine the next block and obtain the rewards. The goal of those experiments was to determine whether an Honest user consistently earns more coins than an Attacker.

In the second round of evaluation experiments, we used both Versions (V1 and V2) of the validator selection function. The goal was to explore the use of an alternate rewarding scheme and its potential effect in the incentivization process, while comparing it with our original approach. Furthermore, we defined and used a new metric (the Gained Coins Ratio) to better quantify the efficiency of our mechanism and facilitate comparison across different experimental setups.

The Progressive Mining algorithm of PoP was used throughout all of our experiments, as it fits our peer-to-peer file sharing scenario better than the Simple Mining algorithm.

5.4.2 Tools

To evaluate our approach, we used a modified version of the publicly available PoP simulator[5]. The PoP simulator is a discrete event Python3 simulator which can simulate prestige transfers between users in a variety of scenarios.

It creates and maintains a user list, along with their associated prestige and coin values. In our modified version, each user also has an associated maliciousness value. The simulator periodically creates new blocks, iterating through all users and updating their prestige values.

In this dissertation, we integrated reputation-based trust in PoP by using the original simulator as a starting point and enhancing it. To this end, we introduced all necessary functionality to accommodate reputation-based trust mechanisms, as well as Content Pollution Attacks. We note that the simulator abstracts lower-level details and simulates PoP behavior "purely", i.e. in a way agnostic of the underlying blockchain or network layer on top of which PoP is deployed. Even though this design poses certain restrictions (e.g. making it hard to incorporate router caching policies and assess their impact), end-user behavior can be effectively simulated which allows us to evaluate our mechanism's incentivization potential and attacker identification capability. The modified version of the simulator can be found in gitlab[6].

Our main addition to the existing simulation logic was the introduction a new parameter, user *maliciousness*. This also includes all relevant functions and other parameters that were necessary for calculating *maliciousness* values during a simulation.

From a high-level perspective, each user has now two main parameters of interest, *prestige* and *maliciousness*. Prestige is calculated on the basis of whether a node is contributing useful work to the network, i.e. to content propagation. Maliciousness is calculated on the basis of whether a node's propagated content is considered malicious by users who have received it. Each node receiving a content object can indicate the content as malevolent ('downvote'), increasing the sending user's *maliciousness* value.

In the default PoP simulator, each user has a work probability. This is the probability that the user is working, i.e. contributing content to the network, during a specified time interval of the simulation (calculations are performed for each block added to the blockchain). In order to accommodate scenarios including malicious users, we also included another parameter, a 'maliciousness probability'. This entails the possibility that a working user is acting maliciously and contributing fake content to the network. Therefore, in each time interval of the simulation a user can be in one of three states: *not working*, *sending legit content* or *sending malicious content*.

From a lower-level perspective, our main modifications were made in the *simulator.py* and *user.py* modules. In the *user.py* module, we modified the *User* class and added new parameters for each *User* object. We also added a new function and modified an existing one, all related to introducing maliciousness into user behaviour and integrating it into the existing simulator mode of operation. For the same goal, we created several new functions in the *simulator.py* module, which are related to maliciousness calculation and integrated the *maliciousness* parameter into the existing functions (mostly used for simulation output). We designed the system so that the calculation of *prestige* and *maliciousness* are independent of each other and, therefore, did not modify the functions used for *prestige* calculations (instead creating new functions used for calculating *maliciousness*).

[5]https://gitlab.com/mharnen/pop/-/tree/master/simulator
[6]https://gitlab.com/alexsarr/tcn-pop

With these additions and modifications, the simulator can now perform simulations in which the producers are launching content poisoning attacks by providing malicious content to the network. It is also able to accommodate attackers with different behaviours, e.g. consistently sending malicious content or switching between sending legitimate and malevolent content in various degrees, while also being able to support the integration of different scoring functions used for maliciousness calculation. This functionality provides flexibility in the evaluation process.

5.4.3 Setup

TABLE 5.4: Simulation Parameters

Network size	100 nodes
Transaction fee	200
Prestige decay factor	0.03
Maliciousness decay factor	0.0001
Coins mined per block	10
Halving Interval	500 blocks
Simulation time	2500 blocks

Regarding the simulation parameters, the network size is set to 100 nodes and each node corresponds to a single, individual user. The transaction fee is set to 200: this does not correspond to coins, but prestige. Therefore, 200 units of prestige are transferred from a beneficiary to a contributor each time a transaction takes place between the two.

Many popular blockchain implementations reduce the number of coins mined after a number of blocks, to ensure coin scarcity. In order to imitate this behaviour, the coins mined per block in our simulation are halved after a predetermined number of blocks. This number of blocks is called the *halving interval*; during the simulations we set its value to 500 blocks (i.e., the number of newly mined coins was reduced by 50% every 500 blocks). The initial number of coins mined per block was set to 10 coins.

During the simulations, we set up an Attacker to send fake/corrupted content to other users. The rest of the users behaved honestly, i.e. did not sent malicious content. During each iteration of the simulation, an Attacker will either send a legitimate file, or a malicious one, depending on the attack model. We assume two main attack models: A "Full-On" model, in which the Attacker sends malicious content in every transaction, as well as an "On-Off" model, in which the Attacker tries to outwit the reputation system by alternating between legitimate and malevolent content transfers (i.e., malicious content consists of 50% of the transactions, while the other 50% consists of legitimate content).

The Full-On and On-Off attackers were simulated the following way: The former was modelled as a user which has 100% probability of sending malicious content during a file transfer, while the latter as a user with a respective probability of 50% (i.e. sending malicious content during half of the transfers). The Honest user always had 0% probability of sending malicious content. All the other network users were also set up to behave in an honest fashion, with a 0% probability of sending malicious content.

In all experiments, we took into account the fact that in PoP, the initial amount of coins a user has in its wallet when joining the network can affect its ability to accumulate coins later on; more initial coins cause a user's prestige to rise, which in

turn increases the mining chances of rich users. In other words, when comparing two users which are contributing to the network with the same amount of useful work (in this case, sending legitimate content), the user starting with more coins will also end up with richer at the end.

As we simulate a P2P network with 100 users, the results are not only affected by the initial coins of the monitored Attacker and Honest user, but also by the rest of the users. For this reason, we consider the users' initial amount of coins to be an independent variable (*Initial Coins*), which affects the dependent variable - the number of coins gained by the Attacker and Honest user (*Gained Coins*). All experiments were conducted by assigning different Initial Coin values to the Attacker and the Honest user, which are depicted in the x-axis.

We performed each evaluation experiment for two different setups: In half of the experiments the rest of the users start with zero (0) initial coins, while in the other half they are assigned a random amount of initial coins between 0 and 200. We repeated each simulation 20 times and calculated the average values of our metrics, to ensure that the final results are not affected by the randomized values of parameters, which can yield slightly different values from simulation to simulation.

During the 1st round of experiments, we used Version 1 (V1) of the validator selection function. The metric we used to quantify the effectiveness of our mechanism was the amount of Gained Coins during an experiment.

During the 2st round of experiments, we used both the V1 and V2 validator selection algorithm, comparing their results against each other. The metric used to evaluate performance was the Gained Coins Ratio.

5.4.4 Metrics

The metrics used during the evaluation process were selected to quantify our mechanism's incentivization efficiency. From this perspective, the following two metrics were chosen:

- *Gained Coins*: The number of coins gained by each user during the simulation experiments. It quantifies the incentivization effectiveness in an absolute manner and is able to capture the difference in rewards between honest users and attackers.

- *Gained Coins Ratio (GCR)*: The number of coins gained by an honest user divided by the number of coins gained by an attacker in the same setup. It quantifies the incentivization effectiveness in a relative manner and facilitates performance comparisons across different settings. The higher its value, the more effective our mechanism is. For instance, a GCR=5 means that an honest user will earn 5 times more coins than an attacker, while a GCR=100 means that an honest user will earn 100 times more coins. Formally, it is defined as:

$$GainedCoinsRatio = \frac{GainedCoins(HonestUser)}{GainedCoins(Attacker)} \tag{5.1}$$

Chapter 6

Experimental Results

6.1 Content Retrieval from intermittently-connected IoT devices

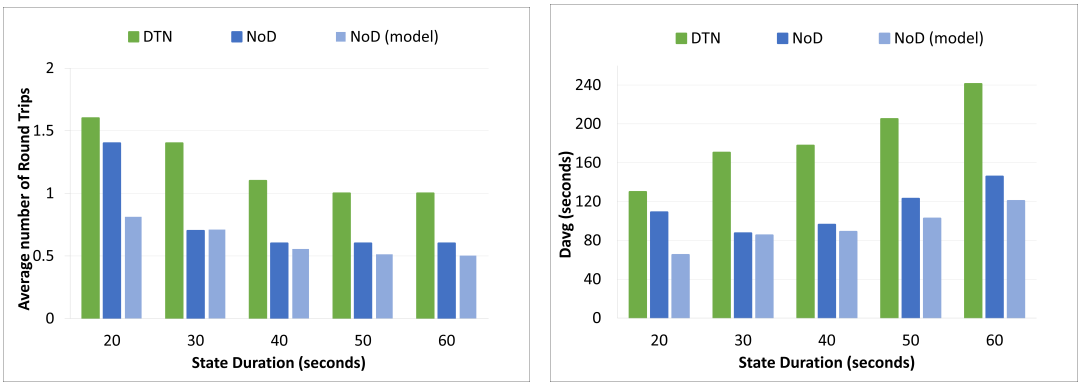
In this Section we present the results of our NoD experimental campaign. The first round of experiments is comprised of five distinct sub-experiments and assesses the performance of NoD-Default using a single data mule. The second round includes two additional experiments and uses multiple mules to compare the performance of NoD-Discovery against NoD-Default in an opportunistic short-contact setting.

6.1.1 First Round of Experiments: Single data mule

Experiment 1 - Different state duration values In this experiment, we compare the performance of NoD-Default against a host-centric DTN implementation for various State Duration values. The results are presented in Figure 6.1. We observe that the default NoD stack outperformed the DTN-only stack in terms of *Average Delay* for all *State Duration* values. In particular, NoD-Default achieved up to a 48.77% decrease in *Average Delay* compared to host-centric DTN (in the case of 30 seconds *State Duration*). This is mainly due to the in-network caching capabilities which enable local retrieval of previously requested content. Therefore in cache-hit cases, there is no need for additional Round Trips via the delay-tolerant network part. To be noted is the fact that even though the DTN-only stack has significantly smaller packet size and reduced implementation complexity, the NoD stack can still achieve smaller *Average Delay* and Average Number of Round Trips values. We plot the model curve for the *Average Delay* and the average number of Round Trips based on equations (3.4) and (3.14), respectively.

More specifically, to obtain the model results in Figure 6.1b, the delay of the corresponding DTN experiments were used as the Producer Delay ($D_{p_{avg}}$) values in equation (3.4). After also measuring the *Cache Hit Ratio* (h) and *Interest Satisfaction Ratio* (ISR) in our experiments and substituting those values in the same equation, we calculated what would be the expected *Average Delay* for each sub-experiment, under the same networking environment.

The model results in Figure 6.1a were obtained in a similar fashion. We use equation (3.14) to calculate the number of round trips, solving the equation for $n_{RT_{avg}}$. Again, $D_{p_{avg}}$ values were obtained from the corresponding experimental DTN delay, while the Round Trip Duration ($d_{RT_{avg}}$) was calculated using equation (3.22). In this scenario, the mule alternates between four (4) *states*, while the *State Duration* ($d_{S_{avg}}$) varies for each sub-experiment and its values are depicted on the x-axis. After calculating $d_{RT_{avg}}$ for Producer-received objects using equation (3.14), we then calculate its average value for all received objects, including cache hits, as illustrated in



(A) Average number of muling Round Trips per sensor measurement ($n_{RT_{avg}}$) for different *State Duration* values.

(B) Average delay (D_{avg}) for different *State Duration* values.

FIGURE 6.1: Experiment 1: NoD-Default and DTN comparison.

A.4. When measuring the corresponding experimental values of $n_{RT_{avg}}$, a cache hit is treated as yielding zero (0) Round Trips.

It can be observed that the best results regarding the Average number of data mule Round Trips (per sensor measurement) are achieved for *State Duration* = 40 sec. We recall that, in our experiments, the *Contact Duration* coincides with *State Duration* values. In this light, the fact that for durations longer than 40 sec, the average number of Round Trips remains constant is explainable, since there is always enough time for successful data exchange. Vice-versa, when the duration lasts less than 40 sec, the average number of Round Trips increases, since the *Contact Duration* may not always suffice.

Smaller *Contact Duration* values, as in the 20-second case, also seem to impact the ability of our NoD-Default implementation to deliver Data compared to the DTN-only stack. This can not only observed in the experiments, but also by comparing the NoD experimental results to the theoretical behavior of the model. Implementation details seem to drive real-world behavior away from the ideal one in extreme cases, as deployment requirements get stricter. An appropriate reasoning for that effect may be the fact that, in this experiment, the IBR-DTN discovery beacon was set to its default value (5 seconds), reducing the time available for data exchange in shorter *Contact Duration* values. The additional overhead introduced by the NDN overlay possibly impacted NoD; for instance, the bundle size in our DTN experiments ranged from 149 to 160 Bytes while the corresponding size in NoD increased to 513 Bytes when a Data packet was encapsulated and 160-170 Bytes when an Interest packet was encapsulated, instead. Beyond that, another contributing factor may be that at our solution can be further optimized implementation-wise, as two separate daemons run in each NoD device (IBR-DTN and NFD).

Experiment 2 - Different number of content objects In this experiment, we widen gradually the pool of available content objects in each sub-experiment, and consequently we effectively limit the chance for a specific content object to be requested again in the same experiment - decreasing, in turn, the chance of a cache hit in the NDN-DTN[1] experiment. Although DTN cannot cache any content - and therefore all data requests had to be muled to the original producer - we continue with the

[1]The terms "NDN-DTN" and "NoD" are used interchangeably to refer to the NDN over DTN protocol stack.

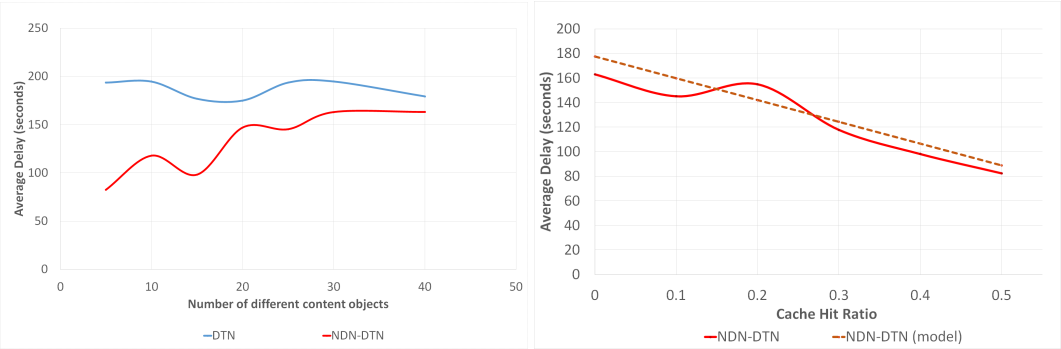


FIGURE 6.2: Experiment 2. Average Delay against Number of Content Objects (left) and Cache Hit Ratio values (right).

experiments to allow for a real-world, in pair comparison between DTN and NoD performance.

Figure 6.2 depicts the average sensor measurement delay for different numbers of content objects and Cache Hit Ratio values. It is clear that NDN-DTN can reduce the average sensor measurement delay and this decrease becomes more significant when Cache Hit Ratios increase (in our setup, this is true for smaller pools of available objects). Furthermore, the observed decrease in the experimental results is similar to the one predicted by our model. IBR-DTN, in comparison, spends roughly the same amount of time in all experiments in order to retrieve content from a remote location, as in-network caching is not supported and the request and response messages have to always be muled to and from the remote area.

Experiment 3 - Varying Interest Lifetime The results are presented in Figure 6.3 with respect to the Average Delay, ISR and Cache Hit Ratio. It can be observed that the Average Delay for receiving a sensor measurement remains relatively stable, despite the change in the Interest Lifetime values. This is justified by the Consumer setup, since it was set to retransmit an Interest for the same prefix as soon as the corresponding Lifetime expired. Therefore, an entry in the NDN Consumer's Pending Interest Table (PIT) always existed at the time the bundle was received by its DTN face. Consequently, retransmitted Interests could be satisfied by an incoming Data packet responding to a previous Interest.

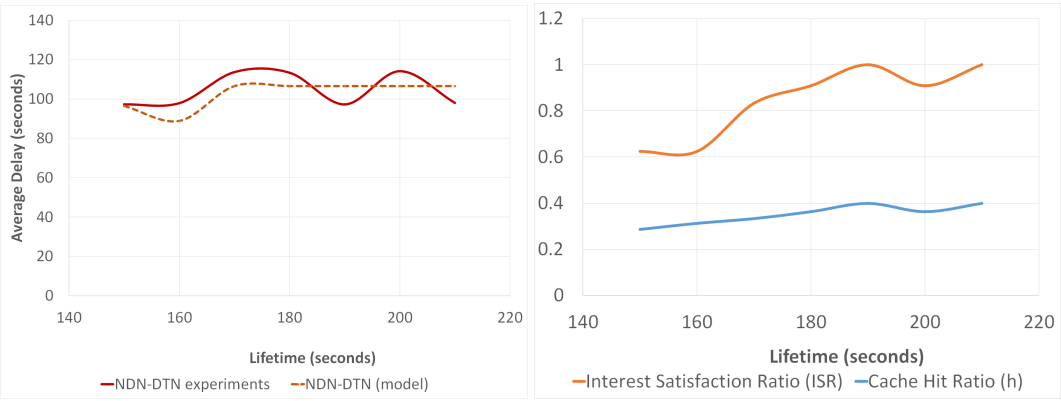


FIGURE 6.3: Experiment 3. Left: Average Delay for different Interest Lifetime values. Right: Interest Satisfaction Ratio and Cache Hit Ratio, for different Interest Lifetime values.

On the right side of Figure 6.3, we can see that the fact that the delay remains constant is not a consequence of the Cache Hit and Interest Satisfaction Ratio remaining constant. On the contrary, concurrent and continuous changes in both those parameter values occur. Indeed, even though the number of cache hits remains steady for different Interest Lifetime values, smaller Lifetime values trigger more re-transmissions and, therefore, a larger number of Interests are being sent, in response - resulting in a smaller Cache Hit Ratio on the local NDN-DTN cache. At the same time, smaller Lifetime values also reduce the chance of a receiving a Data packet before the Lifetime expires and, therefore, result in a lower Interest Satisfaction Ratio. We highlight the fact that their net effect on the delay appears null in our experiments - a result of the h/ISR ratio remaining relatively constant regardless of the Interest Lifetime value (see also equation 3.2).

Based on these results we can infer that, in NoD, the choice between short- and long-lived Interests reflects the corresponding exploitation of a tradeoff: frequent short-lived Interests cause more network traffic, while infrequent long-lived Interests stay in each PIT longer and may potentially burden memory further. Given that Interests are instantly retransmitted by the consumer when their Lifetime expires, the choice does not affect the Average Delay.

Experiment 4 - Varying Data Freshness Figure 6.4 shows that there is a significant reduction in the Average Delay as the Data Freshness is increased. This is mainly due to the increase in the Cache hit ratio, as Data with a high Freshness Period remains for in the Content Store for a longer period of time and more of the subsequent Interests can be satisfied by the Cache.

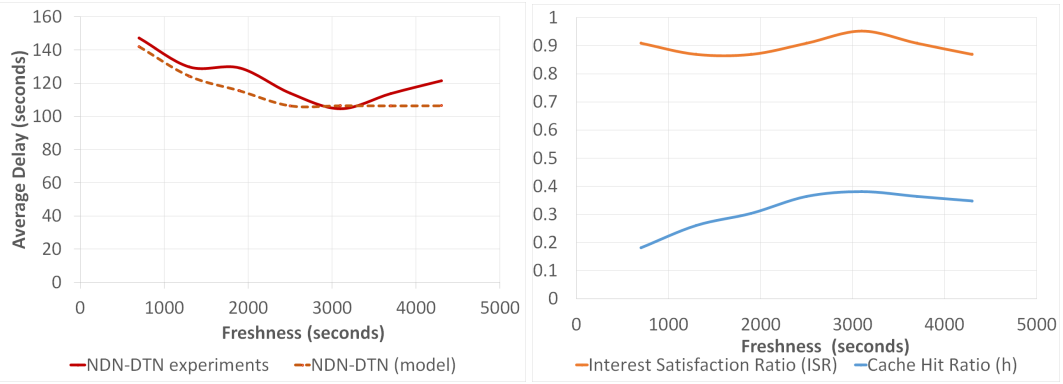


FIGURE 6.4: Experiment 4. Left: Average Delay for different Data Freshness Period values. Right: Interest Satisfaction Ratio and Cache Hit Ratio for different Data Freshness Period values.

However, we can see that setting the Freshness Period to values higher than 3000 seconds in our experiments, did not increase the Cache Hit Ratio further. This is related to the time needed for the data mule to complete a round trip between the producer and consumer. Each full cycle takes on average $4x40 = 160$ seconds (from connecting to wireless network 1, to connecting to the same network again), which also approximates the average delay for receiving a Data packet from the remote producer when the contact duration suffices.

Once a Data packet is received by the local router, a cache hit may happen for as long as the packet is considered 'fresh'. In this experiment, we repeated the process until 20 content objects are received. Therefore, a rough estimate of the duration that packets need to remain 'fresh' in our experiment is 20x160=3200 seconds. Not

surprisingly, this is also the highest Freshness Period value for which we can observe any improvement on the Cache Hit Ratio. Any further increase in Freshness Period does not change the Interest Satisfaction Ratio (which revolves around the 0.9 value throughout the experiment); this is also the Freshness Period value yielding the best results with regard to the Average Delay (see Figure 6.4. We note that an NDN consumer might not use the "MustBeFresh" optional field in the Interest header (as was the case in our experiments), in which case the results may be different.

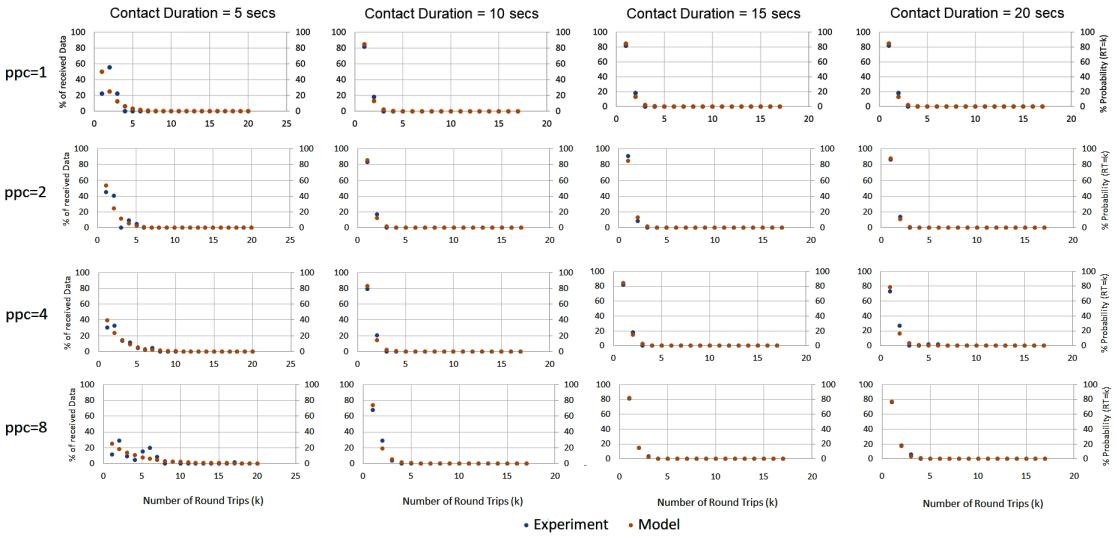


FIGURE 6.5: Experiment 5: Probability of content retrieval in exactly k Round Trips $Pr(n_{RT} = k)$ as predicted by our data muling model, compared to the percentage of content actually retrieved in k Round Trips during the experiments, for various packets-per-contact and *Contact Duration* values.

Experiment 5: Short Contact Duration and varying traffic volume Experiment 5 is actually comprised of several sub-experiments, the results on which are depicted in Figures 6.5-6.10. These Figures depict the performance of our mechanism from various perspectives, several of which were chosen to determine the degree of our model's accuracy.

First, we present Figures 6.5-6.6 where the accuracy of our model is validated. In Figure 6.5 we depict the percentage of delivered data at exactly k Round Trips during the experiments. To plot the model curve on Figure 6.5, the data muling model presented in Section 3.5.2 was used. To calculate the specific values, we first measured the average number of Round Trips ($n_{RT_{avg}}$) on each sub-experiment, then calculated the probability of delivery within a single Round Trip (p) using equation (3.16). Subsequently, we calculated the probability of data delivery within each number of Round Trips using equation (3.15).

As can be seen from Figure 6.5, the model is aligned well with the experimental findings, justifying our strategy to reflect the properties of the data muling system using Bernoulli trials and the Geometric distribution to capture the probability of delivery with respect to the number of muling Round Trips.

Equations (3.23)-(3.25) capture the probability of content retrieval within a single round trip (p). We associate p with the total number of sent packets-per-contact (n_{ppc}) and *Contact Duration* (d_C). Having already validated the veracity of equation (3.16), we first calculate p based on the average number of Round Trips measured

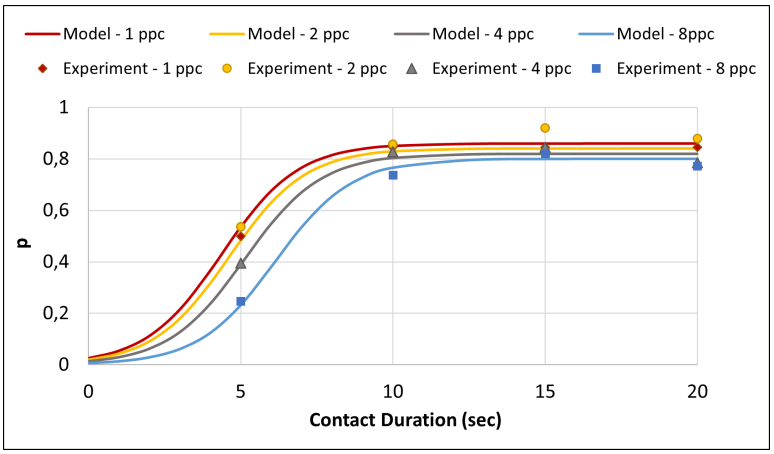
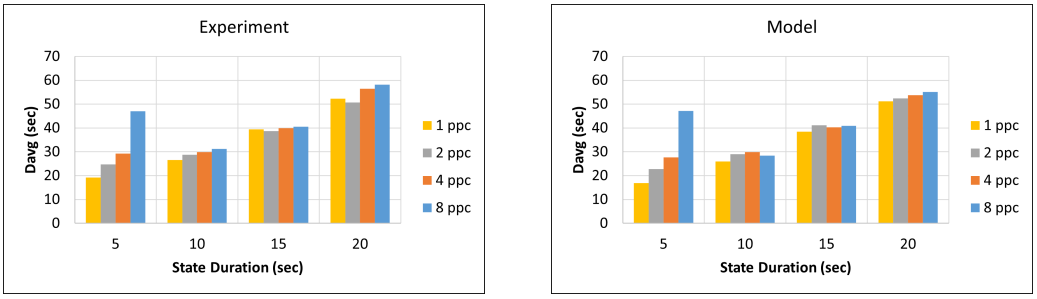


FIGURE 6.6: Experiment 5: Probability of content retrieval within a muling round trip (p), for various *Contact Duration* values: Experiments vs Model

during our experiments. We contrast those, experimental, results with the values of p as modelled by equations (3.23)-(3.25). The results is depicted in Figure 6.6, which confirm that the implementation's behavior is accurately reflected in the latter equations and, therefore, further estimations for p can be made for varying *Contact Duration* and packets-per-contact values.



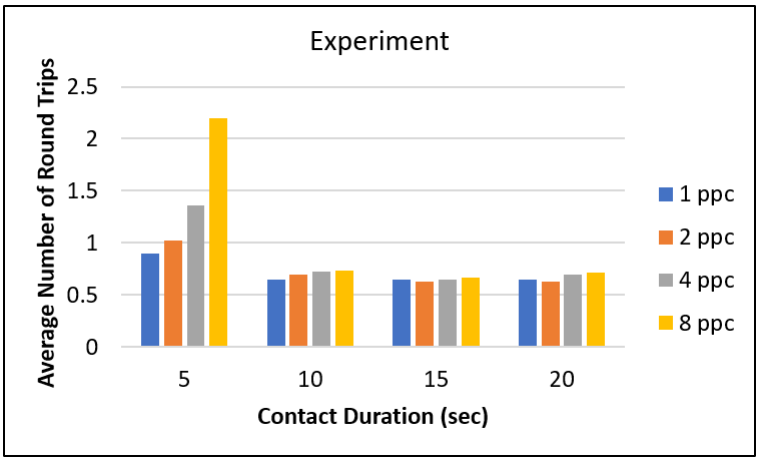
(A) D_{avg} for various *Contact Duration* values, as measured

(B) D_{avg} for various *Contact Duration* values, as expected by model

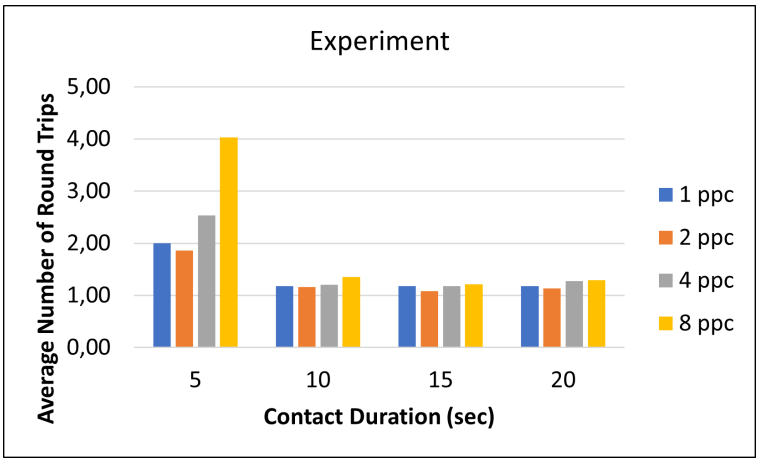
FIGURE 6.7: Experiment 5: *Average Delay* (D_{avg}), compared with Model

Figures 6.7-6.10 focus on the evaluation of our implementation. The impact of transient contacts (short connectivity) on the *Average Delay* for varying amount of exchanged packets-per-contact is depicted in Figure 6.7. As expected, the shorter the *Contact Duration*, the more unlikely is an *ad hoc* connection to be established and subsequently an exchange of packets between intermittently-connected devices, which justifies the increase in *Average Delay*. Furthermore, as the number of sent packets during a contact increases, the *Average Delay* also increases - especially when *Contact Duration* becomes shorter.

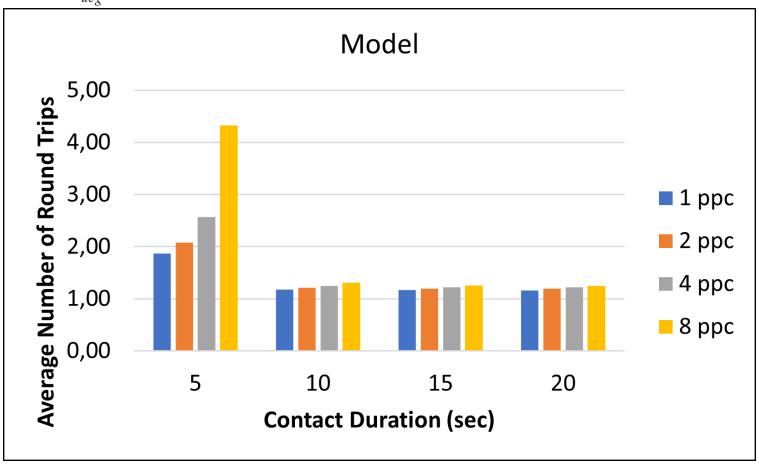
We note that a *Contact Duration* of 5 sec suffices for the exchange of 1 or 2 exchanged packets-per-contact, but fails for larger amount of packets. Similar observation can be made on other Figures: on Figure 6.8a (see the 1- and 2-packet case, where the average number of Round Trips tends to approach 1; this is more clearly depicted on 6.8b, where the Average Round Trips are computed without taking into account the influence of caching). We can also observe that, in the case of 10 sec



(A) $n_{RT_{avg}}$ for various *Contact Duration* values (including cache-received Data)

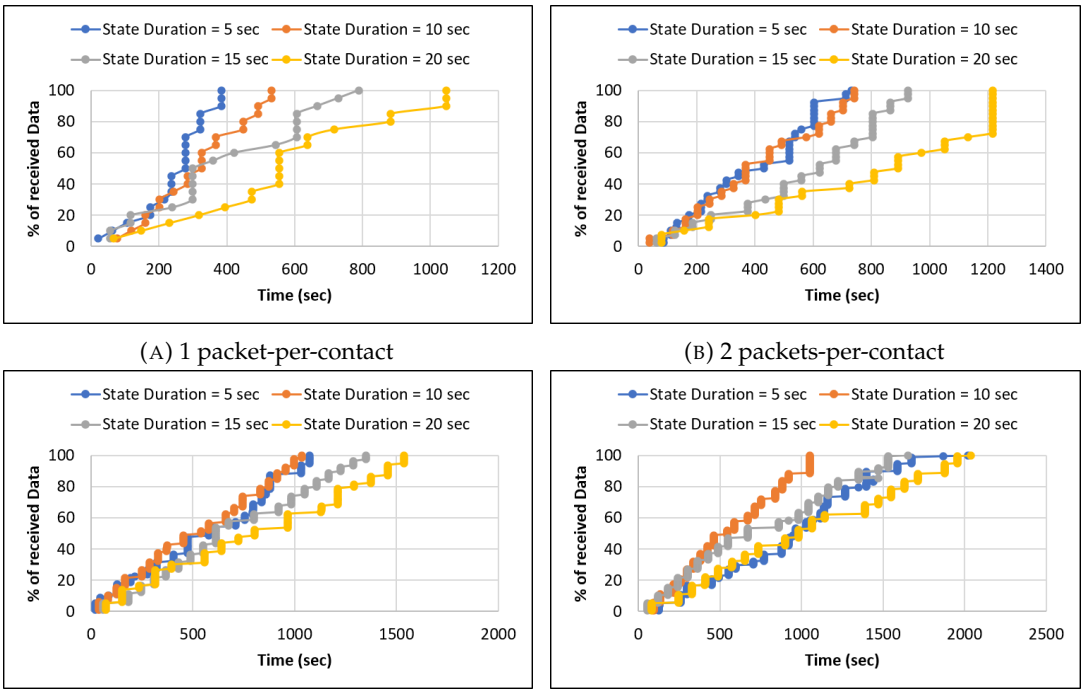


(B) $n_{RT_{avg}}$ for various *Contact Duration* values (Producer-received Data)



(C) Expected $n_{RT_{avg}}$ for various *Contact Duration* values (Producer-received Data)

FIGURE 6.8: Experiment 5: Average number of muling Round Trips (n_{RT}), compared with Model



(A) 1 packet-per-contact

(B) 2 packets-per-contact

(C) 4 packets-per-contact

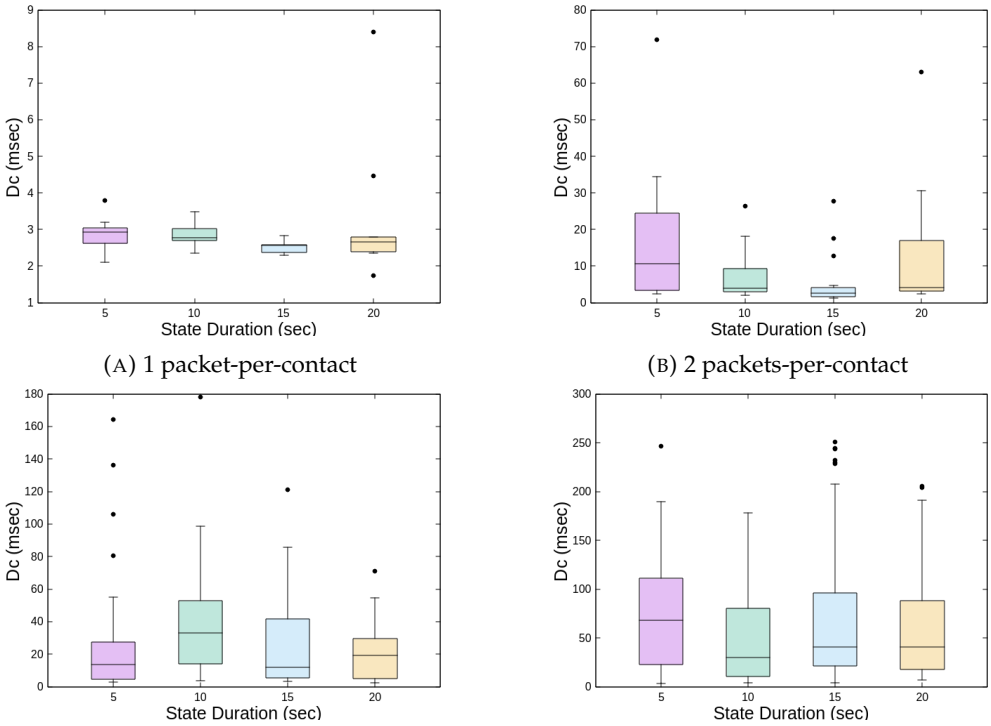
(D) 8 packets-per-contact

FIGURE 6.9: Experiment 5: Percentage of received Data packets in time.

Contact Duration, D_{avg} is slightly increased, proportionally to the number of packets-per-contact. The D_{avg} values of varying ppc values for 10 and 15 sec *Contact Duration* are relatively close, since the duration is sufficient for data exchange. In case of 20 sec *Contact Duration*, D_{avg} is increased proportionally to the number of packets-per-contact. The high D_{avg} values in case of 8 packets-per-contact are caused by the higher probability of losing a packet and the longer round trip duration.

The results depicted in Figures 6.9c and 6.9d indicate that a *Contact Duration* of 5 seconds does not suffice for the cases of 4 and 8 packets-per-contact. Particularly, in Figure 6.9c, the sub-experiment which used a 10-second *State Duration* concluded earlier than the one which used a 5-second *State Duration*. Furthermore, in Figure 6.9d, the Cumulative Distribution Function (CDF) plot of 5 seconds *State Duration* has similar performance with the plot of 20 seconds *State Duration*, even though the duration of a muling Round Trip is four times lower in the former case. Evidently, this behavior is caused by the higher total number of Round Trips needed to retrieve a content object in the 5-second case. Therefore, we conclude that that the exchange of a larger number of packets (e.g. 8 packets-per-contact) in our implementation, requires a *Contact Duration* of at least 5 seconds to guarantee some degree of reliability. This can also be observed in Figure 6.6, which relates our model with the experimental data.

Another interesting observation concerns the *Caching Delay* (D_c) for varying *State Duration* and packet-per-contact values. Although D_c could be considered negligible compared to the Producer Delay (D_p), which is caused by Round Trips, we note that D_c values are significantly increased as the volume of sent Interests increases. The latter, is a straightforward conclusion of Figure 6.10 and is mostly attributed to the heavyweight implementation that we use, which interconnects two distinct daemons for the two protocols (NFD and IBR-DTN). Indeed, we can observe that D_c in case of 8 exchanged packets-per-contact can be almost 100 times higher than



(A) 1 packet-per-contact

(B) 2 packets-per-contact

(C) 4 packets-per-contact

(D) 8 packets-per-contact

FIGURE 6.10: Experiment 5: *Caching Delay* (msec) for various *State Duration* (sec) values

the case of one (1) exchanged packet-per-contact. More refined implementations of the stack could reach significantly lower caching delays, without, however, affecting the general trend (i.e. *Caching Delay* increasing proportionally to the total number of packets sent to the caching node). Therefore, for large volumes of data, D_c may no longer be negligible but instead needs to be taken into account.

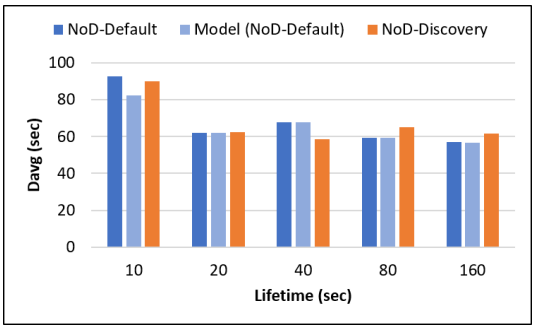
6.1.2 Second Round of Experiments: Multiple data mules

The second round of experiments comprises two distinct sub-experiments for evaluating NoD-Default and NoD-Discovery performance in the context of opportunistic networking. This set of experiments aims at a comparative evaluation of the two approaches, as well as an assessment of the our NoD-Default model's validity[2].

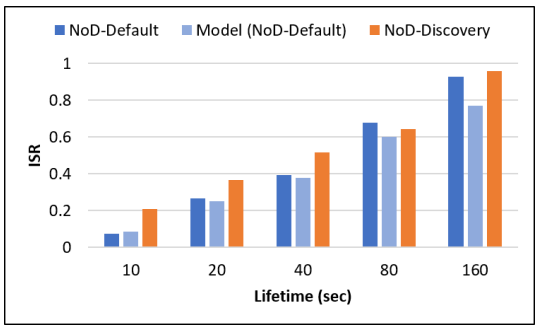
Experiment 6: Varying Lifetime In this experiment, we evaluate the performance of NoD-Default and NoD-Discovery for various Interest Lifetime values. Model results are also provided for NoD-Default to assess its accuracy.

The overall experimental results are summarized in Figure 6.11. First off, in Figure 6.11a we observe that in both cases D_{avg} are equivalent, for corresponding values of Interest Lifetime. This is justified by the fact that Interest Lifetime does not affect directly D_{avg}, and thus either a short- or a long-lived Interest packet will have roughly the same impact on the D_{avg} metric.

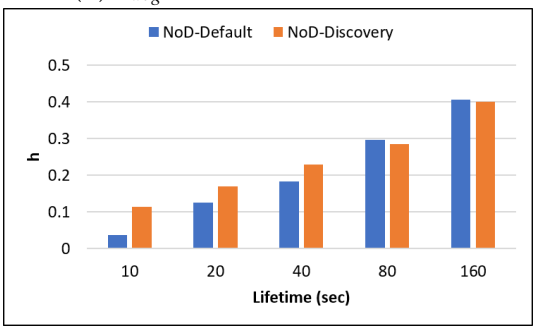
[2]In this dissertation, we only model NoD-Default, which is Consumer-driven and Interest Lifetime choices of the Consumer application directly affect the volume of sent Interests. NoD-Discovery is, instead, mobility-driven and depends on connectivity patterns of data mules and their contacts with the gateway. Given the randomized mobility patterns in our experiments, we consider the properties of NoD-Discovery model out of the scope of this work.



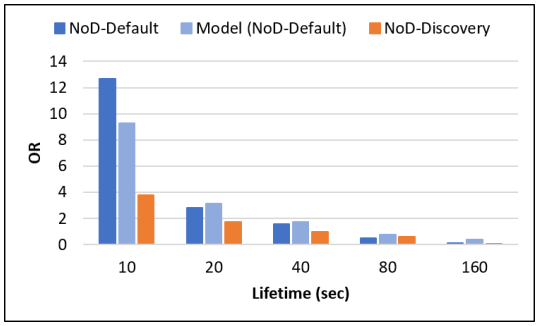
(A) D_{avg} for various Lifetime values.



(B) ISR for various Lifetime values.



(C) h for various Lifetime values.



(D) OR for various Lifetime values.

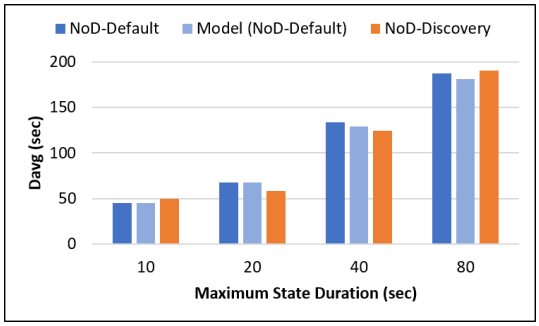
FIGURE 6.11: Results of Experiment 6

In the 10-second Lifetime case, we observe a deviation of the D_{avg} values in comparison to the other Lifetime values in both approaches. This is justified in this specific case by the outstanding number of NDN packets generated (673 transmitted Interests in Discovery and 2100 transmitted Interests in Default case). Each of these packets was encapsulated in a bundle and forwarded to the other DTN nodes using flooding routing. The same procedure was repeated with the Data packets, which are much larger in size. Therefore, network traffic increased significantly, subsequently impacting D_{avg}. This conclusion can also be approached by the large *Overhead Ratio* (OR) values which are observed in the 10-second case.

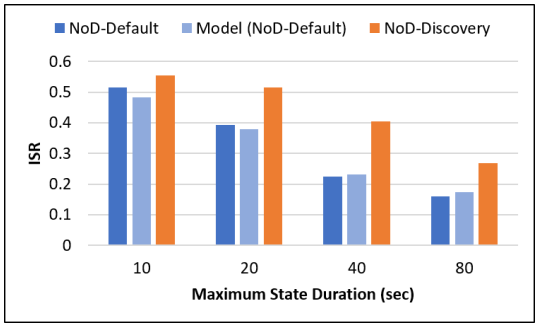
Specific conclusions can also be drawn about the influence of the cross-layer interaction on ISR and h by Figures 6.11b and 6.11c, respectively. Contrary to D_{avg}, the ISR and h metrics are highly dependent on the choice of Interest Lifetime. Specifically, NoD-Discovery causes an increase of up to 187.2% compared to NoD-Default's ISR and up to 208.76% regarding h, for short Interest Lifetime values (i.e., 10, 20, 40 sec). The reason behind this is the redundant Interest re-transmissions by NoD-Default, caused by small Lifetime values.

For larger Interest Lifetime values (i.e., 80, 160), the results coincide for both performance metrics, since the Lifetime values suffice for returning the Data packets. Therefore, we can claim that the impact of the cross-layer interaction becomes more significant when the Interest Lifetime becomes shorter.

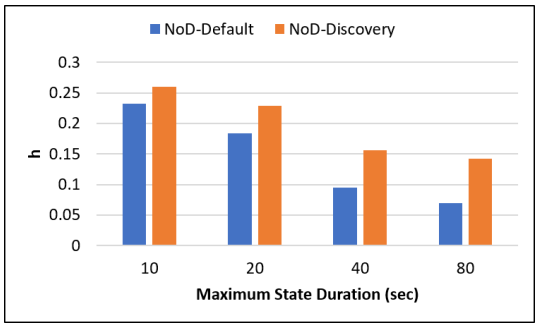
The same happens with the OR metric, which is depicted in Figure 6.11d and demonstrates clearly that the proposed cross-layer interaction could drastically improve network performance especially when the Interest Lifetime becomes shorter. Indeed, the NoD-Discovery achieved 70.34% decrease of the OR metric, compared to the NoD-Default, in the sub-experiment of 10 seconds Interest Lifetime.



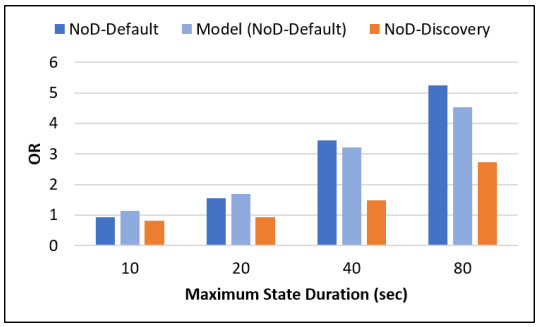
(A) D_{avg} for various *Maximum State Duration* values.



(B) *ISR* for various *Maximum State Duration* values.



(C) *h* for various *Maximum State Duration* values.



(D) *OR* for various *Maximum State Duration* values.

FIGURE 6.12: Results of Experiment 7

Experiment 7: Varying Maximum State Duration In this experiment we assess the impact of a wider range of possible *Contact Duration* values on our NoD-Default and NoD-Discovery approaches. Each mule stays at each state for a duration chosen randomly between 1 and *Maximum State Duration* seconds. The effect of this, is that higher *Maximum State Duration* values will: (i) increase the average *Contact Duration* between two devices during an experiment, therefore increasing the probability of data exchange and (ii) increase the average duration of each muling Round Trip.

In Figure 6.12a, we observe that both approaches and model converge to similar D_{avg} values for the respective case of *Maximum State Duration*. It is also noticeable that D_{avg} is increasing as the *Maximum State Duration* increases, due to the corresponding incremental muling delays.

Concerning the *ISR* and *h* we conclude from Figures 6.12b and 6.12c, that the cross-layer interaction outperformed the Default NoD operation in both metrics. Specifically, NoD-Discovery causes up to 80.05% increase in *ISR* (in the case of 40 seconds of *Maximum State Duration*) and 103.74% increase in *h* (in the case of 80 seconds of *Maximum State Duration*).

Furthermore, Figure 6.12d clearly proves that as the delay due to data muling increases, contribution of the cross-layer interaction to the *OR* reduction becomes more significant. In other words, the reduction of re-transmitted packets over the cross-layer design is notable even in short-connectivity scenarios, let alone the large delay scenarios. Particularly, NoD-Discovery contributed to 57.32% decrease of the *OR* metric (in the case of 40 seconds of *Maximum State Duration*).

6.1.3 Summary of experimental findings

- In Experiment 1, we showed that NoD reduces the *Average Delay* and average number of Round Trips significantly when compared to the DTN-only stack.
- Experiment 2 demonstrated that the Cache Hit Ratio achieved by NDN is important performance-wise, as larger Cache Hit Ratio values greatly improve the Average Delay.
- Experiment 3 suggested that the Interest Lifetime value does not influence the NoD delay in our experiments. We note that other trade-offs may exist and, therefore, such a choice should be tied to the specific deployment requirements (energy efficiency, DTN storage space etc.).
- Experiment 4 explored the impact of Freshness Period. Based on the results, we conclude that selecting Freshness values below a certain limit can cause the Cache Hit Ratio to drop and Average Delay to increase. This limit can be calculated in our setup, using the DTN delay.
- In Experiment 5, we focused on benchmarking our default NoD implementation. To this end, we established the particular requirements for its operation. One such requirement, is the need for a *Contact Duration* larger than roughly 5 seconds to guarantee packet exchange. A second requirement is to maintain a low amount of exchanged NDN packets, especially in scenarios with transient contacts (e.g. 5 seconds). This requirement stemmed from the observation that the transmission of more than 4 Interest/Data packets-per-contact caused a disproportional increase in Round Trips needed to retrieve the content objects and severely impacted the *Average Delay*. For environments with *Contact Duration* greater than 10 seconds, this property became less significant since our implementation was capable of handling more traffic.
- In Experiments 6 and 7, we compared NoD-Default with the improved NoD-Discovery version and have shown that leveraging cross-layer information between the NDN and DTN layers can reduce the network overhead significantly, without compromising performance (as measured by the average data retrieval delay). We have also shown that the *Overhead Ratio* is a function of Interest Lifetime and increases with shorter Interest Lifetime.
- Overall, our mathematical model has been shown to accurately predict our implementation's behavior and confirmed its soundness, since the experimental results indeed match our model.

6.2 Impact of chunk size in NDN performance

In this section, we present the results of our chunk size experiments. In each figure, we use a different metric to assess different aspects of NDN performance. Our overall goal is to assess the impact of various Data chunk sizes on different aspects of NDN performance. In all experiments, content requests by the Consumers follow a Zipf-Mandelbrot distribution to approximate real-world user behaviour. The results are grouped by the evaluation metric being considered each time and the two experiments ("no congestion" and "moderate congestion") are jointly presented and discussed.

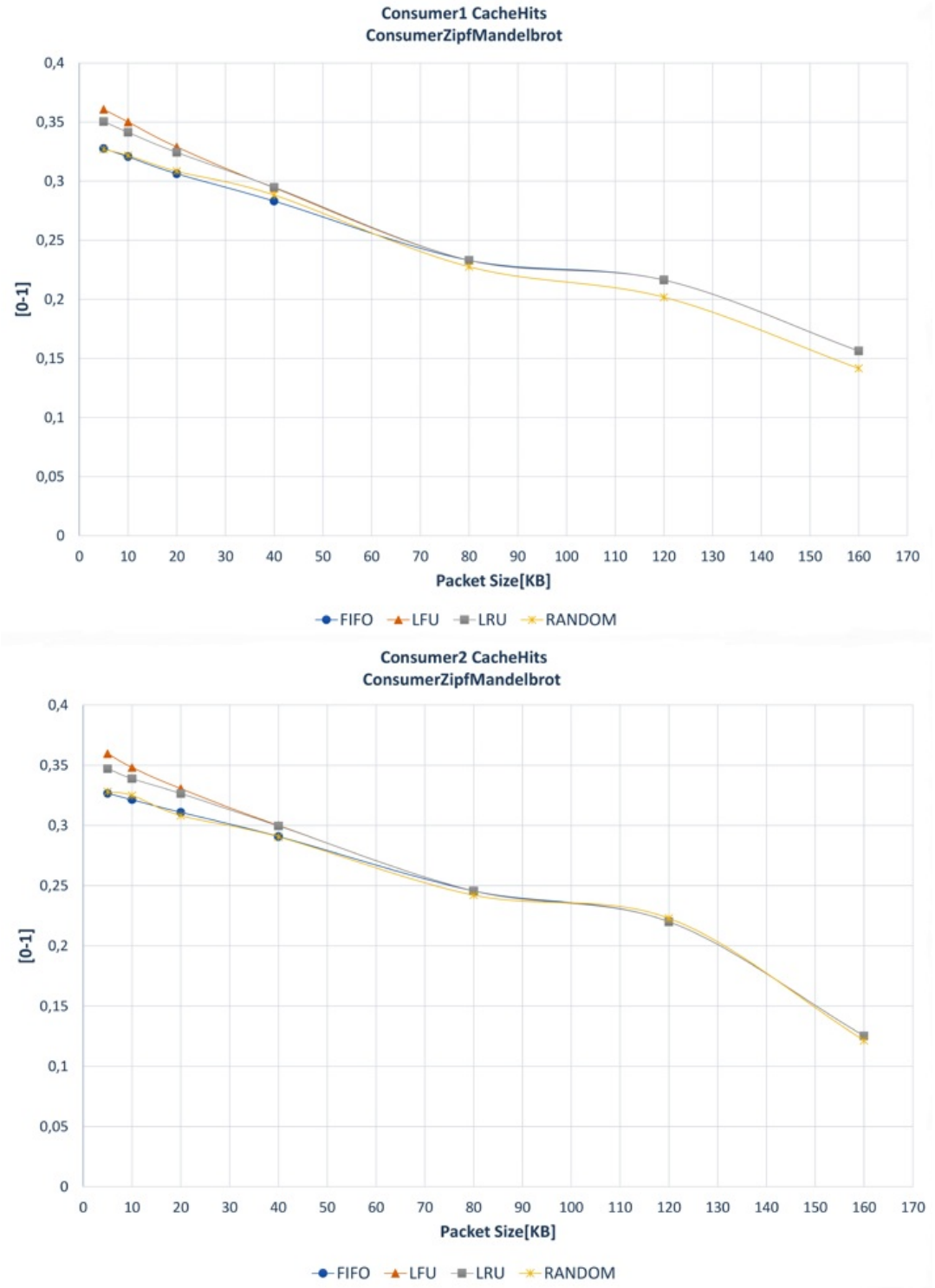


FIGURE 6.13: Average Cache Hit Ratio on the "no congestion" experiment.

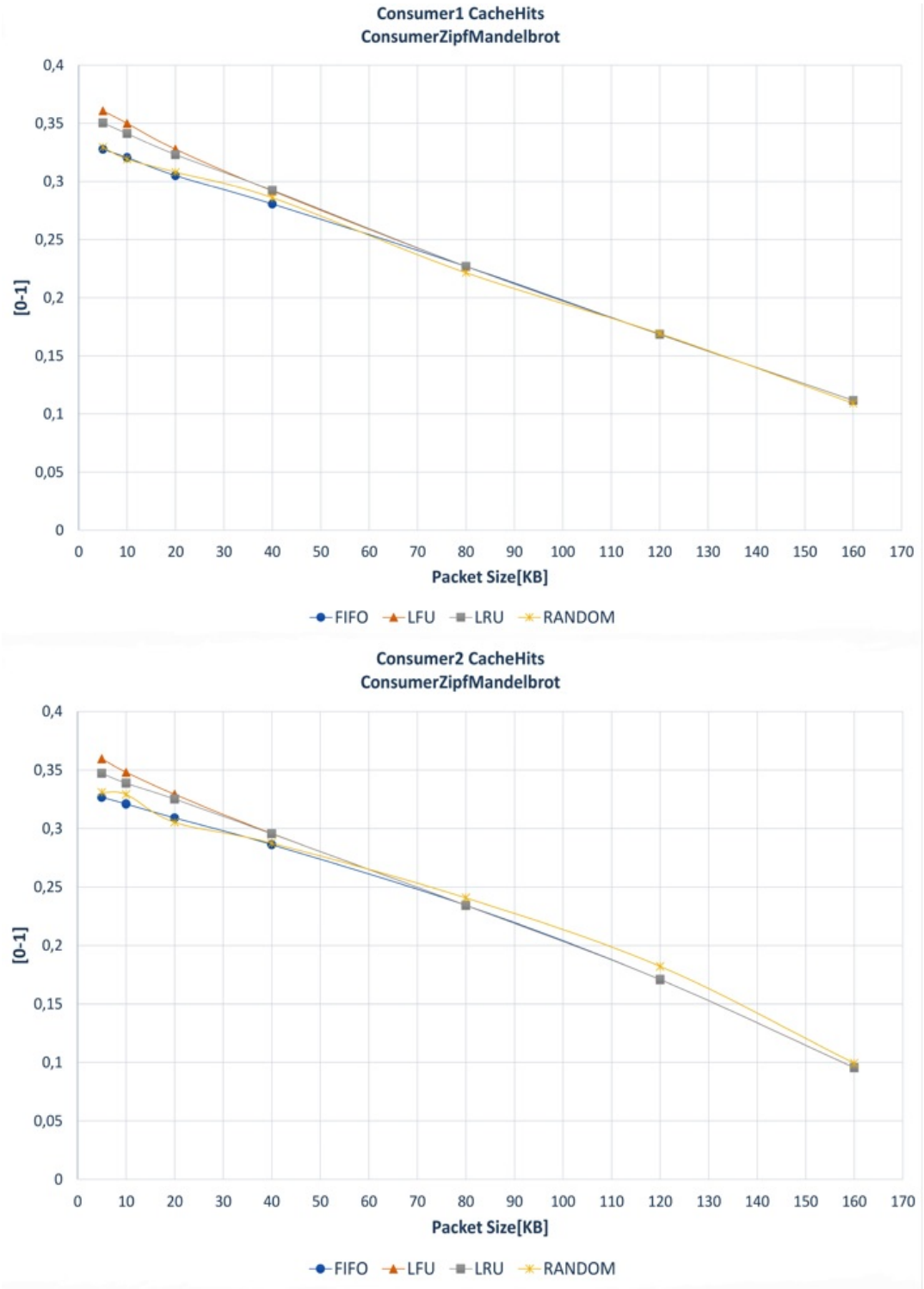


FIGURE 6.14: Average Cache Hit Ratio on the "mild congestion" experiment.

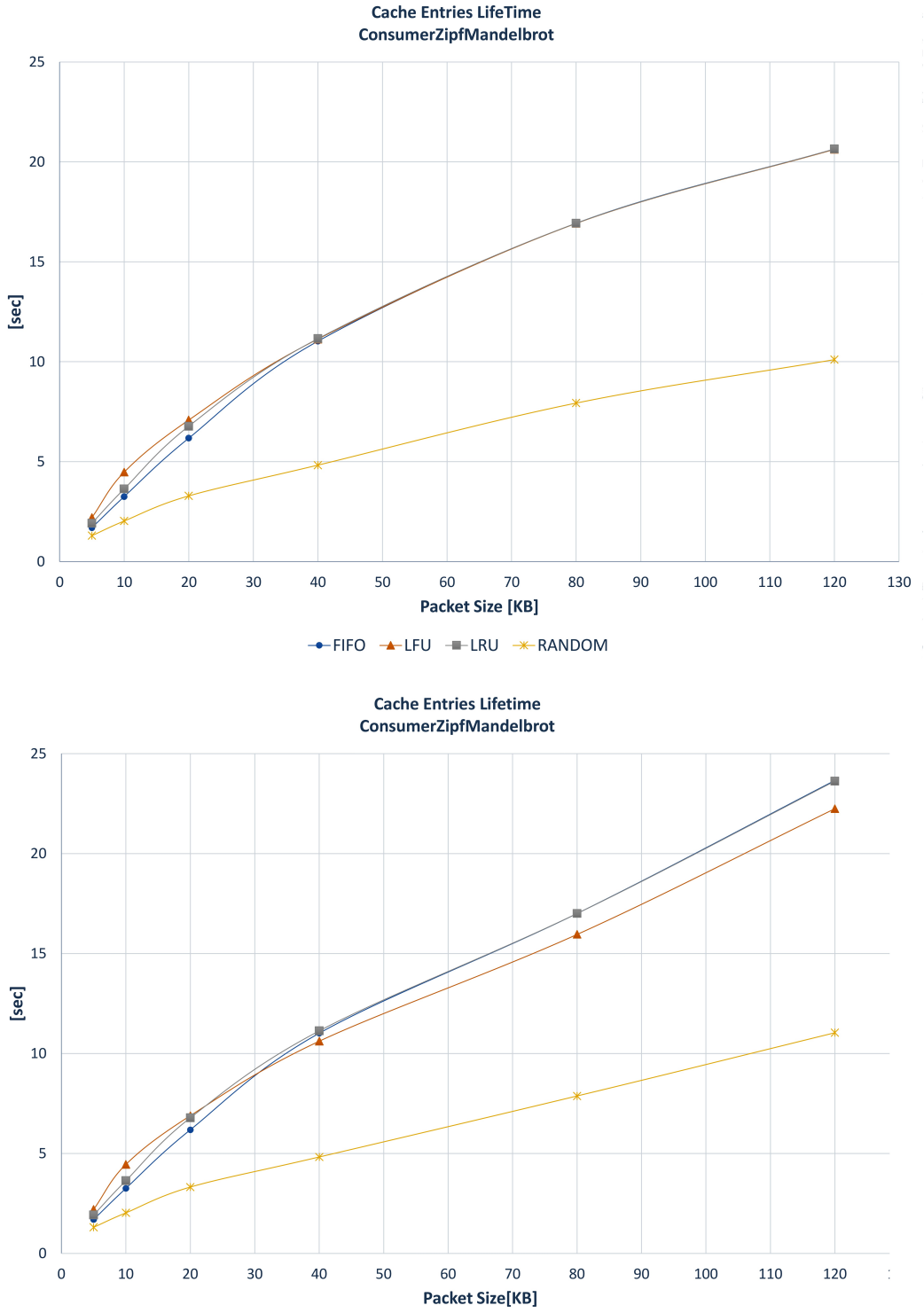


FIGURE 6.15: Cache Entries Lifetime (up: no congestion, down: mild congestion)

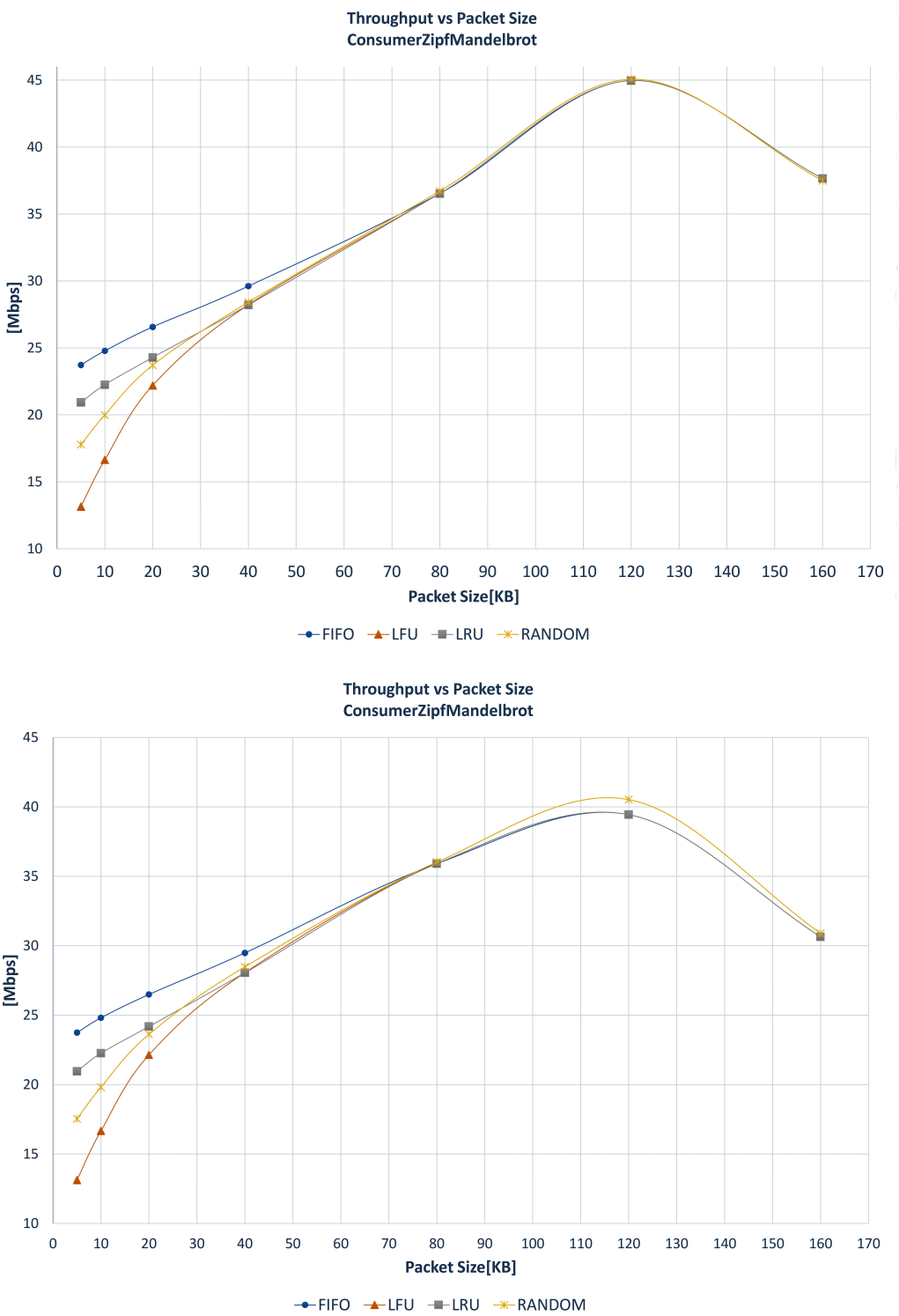


FIGURE 6.16: Throughput (up: no congestion, down: mild congestion)

6.2.1 Cache Hit Ratio

In Figures 6.13-6.14, each Consumer's average Cache Hit Ratio can be observed with respect to different Data packet sizes. Based on the figure, we can infer that larger Data packet sizes incur a lower Cache Hit Ratio. This holds true in all cases, regardless of the cache replacement policy being chosen. The aforementioned results can be explained by the fact that our experiments run until a fixed amount of data was retrieved (i.e. 135 MB of data). Therefore, increasing the Data chunk size meant that less Interests had to be sent to retrieve the specified amount of data. Less Interests meant that less Data packets were returned which, in turn, meant less overall cache entries during the experiment. For instance, 900 Interests per second were sent in the case of 5 KB packet size, while only 28 Interests per second were sent in the case of 160 KB packet size. This meant that for smaller packet sizes, when the last Interest of each sub-experiment was received by the cache, there were more entries in the Content Store which could potentially satisfy the Interest. Therefore, self-evidently, the chance that a subsequent Interest could be answered by previously-received content that was stored in the cache, was less for bigger chunk sizes, which eventually resulted in a lower cache hit ratio.

What can also be inferred by Figures 6.13 and 6.14, is the fact that setting the right cache replacement policies seems to make more of a difference (with respect to the Cache Hit Ratio) when using smaller Data packet sizes, up to 40 KB. In this range, RANDOM replacement yields the worst results, while LFU replacement yields the best ones. However, when increasing the packet size to 80-160 KB, the difference seems to be minimized and choosing a more sophisticated policy over random replacement seems to make less of a difference.

6.2.2 Average Cache Entries Lifetime

In Figure 6.15, the Cache Entries Lifetime can be observed, with respect to different Data packet sizes. On the left side, the results regarding the "No-congestion" experiment are depicted while on the right side, the results regarding the "Mild congestion" experiment are depicted. In both cases, there is a notable difference between using a random cache replacement policy and using more sophisticated ones, as cache entries stay significantly longer in the Content Store when using either LRU, LFU or FIFO (an 100% increase can be observed when compared to the time measured for the RANDOM policy).

Furthermore, it also holds true in this case that the difference between LRU, LFU and FIFO policies is more pronounced when using smaller packet sizes, with the LFU policy out-performing the rest. Again, as packet sizes increase, the performance difference between the caching policies decreases, as was the case concerning the Cache Hit Ratio metric.

Of note are also two things: a) the fact that the Cache Entry Lifetimes do not differ significantly between the two experiments when using the same packet size and b) the fact that the Cache Entry Lifetimes increase as packet sizes increase. This can be explained by the same cause as the one described in the previous sub-section: in our setup, larger Data sizes mean that fewer packets are being transmitted in an experiment. Therefore, there is less need for replacing the cache entries during the simulation, resulting in larger lifetimes for each entry.

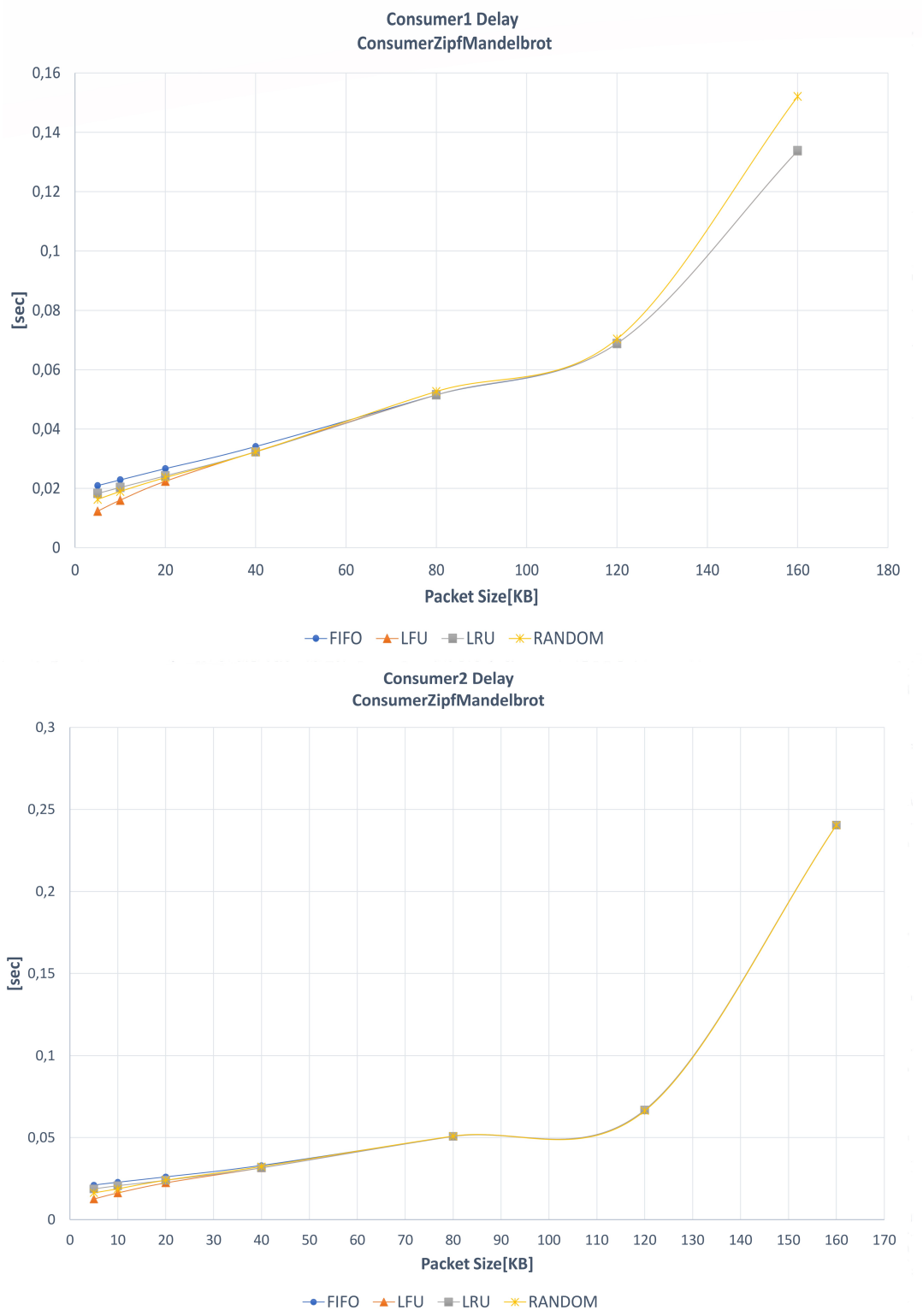


FIGURE 6.17: Average Delay in the "no congestion" experiment

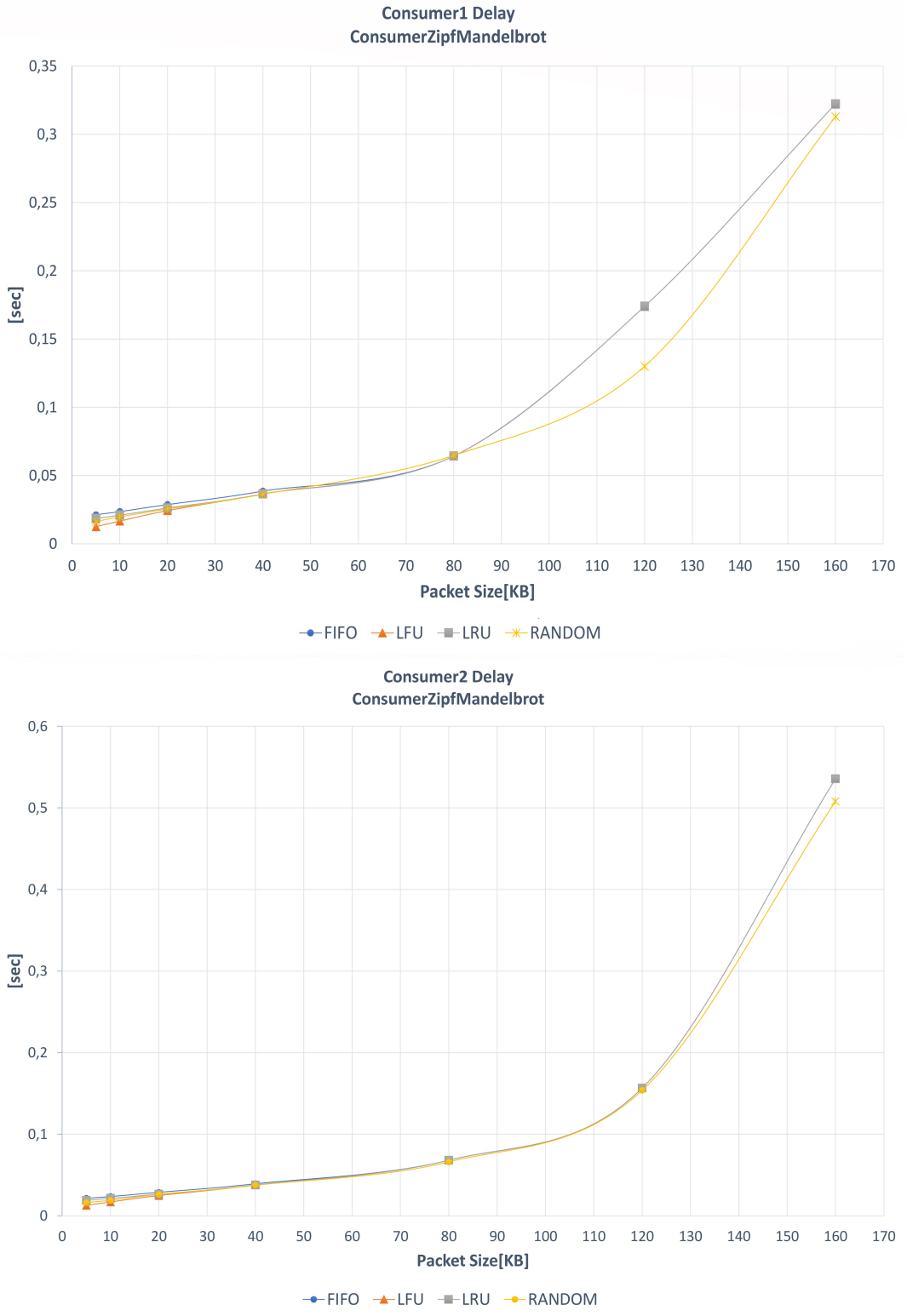


FIGURE 6.18: Average Delay in the "mild congestion" experiment

6.2.3 Average Throughput

In Figure 6.16, the Average Throughput can be observed with respect to the two experiments. While there is no significant difference for chunk sizes up to 80 KB, a deviation from the previous trend can be seen for 120 KB and 160 KB packet sizes. When using these sizes, the throughput is markedly larger in the non-congested case. It can therefore be inferred that deployments that use larger packet sizes might be more susceptible to the effects of congestion that the ones using smaller Data packets, resulting in degraded performance.

Of note is also the fact that, once again, the performance difference between the various caching policies is more pronounced for smaller packet sizes. For packet sizes larger than 80 KB, the performance is the same regardless of the cache replacement policy being chosen. In the smallest range of packet sizes (5-40 KB), FIFO outperforms the rest of the policies with respect to the average throughput being achieved. The worst performance is, counter-intuitively, achieved not by random replacement but by LFU. Lastly, we can infer that larger packet sizes result in increased average throughput. More specifically, increasing the packet size from 5 to 120 KB (24 times), results in a throughput increase ranging from 1.6 (FIFO, congestion) to 3.5 times (LFU, no congestion).

6.2.4 Delay

In Figures 6.17 and 6.18, the Average Delay for each chunk size is being shown. Based on the results, we can infer that the average delay per packet increases when experimenting with larger packet sizes. This is to be expected, as larger packet sizes incur larger processing delays in intermediate routers and Consumers. Furthermore, we can once more notice the fact that the performance of the different cache replacement policies converges for larger packet sizes. For smaller packet sizes, LFU performs better (smaller delays) while FIFO performs the worst (larger delays).

Furthermore, there is a notable difference regarding the average delays between the non-congested and congested experiment, which is more pronounced for larger packet sizes. For the 160KB case, we can see the average delay doubles when congestion is incurred, when compared against the corresponding average delay in the "no-congestion" experiment.

6.3 Bitrate Oscillation Attack mitigation

6.3.1 Experiment 1: Reputation-based countermeasure efficiency

In this subsection, we evaluate the efficiency of our proposed reputation-based countermeasure. The simulation results are illustrated in Figure 6.19. We compare our results against both the results of the on-off attack model and the absence of an attacker. Note that in this set of experiments, the intermediate routers have adopted the threshold-based caching policy and set the threshold value to 7.

The results show that despite the on-off behavior of the adversary, which acts legitimately for long periods of time and strives to confuse the reputation-based trust system, our countermeasure efficiently achieves to mitigate the impact of the ongoing BOA.

In the beginning, the adversary accomplishes to hide his/hers malicious behavior and cause some redundant oscillations. However, due to the punishments enforced by the trust model, the attack's impact is rapidly eliminated. In particular,

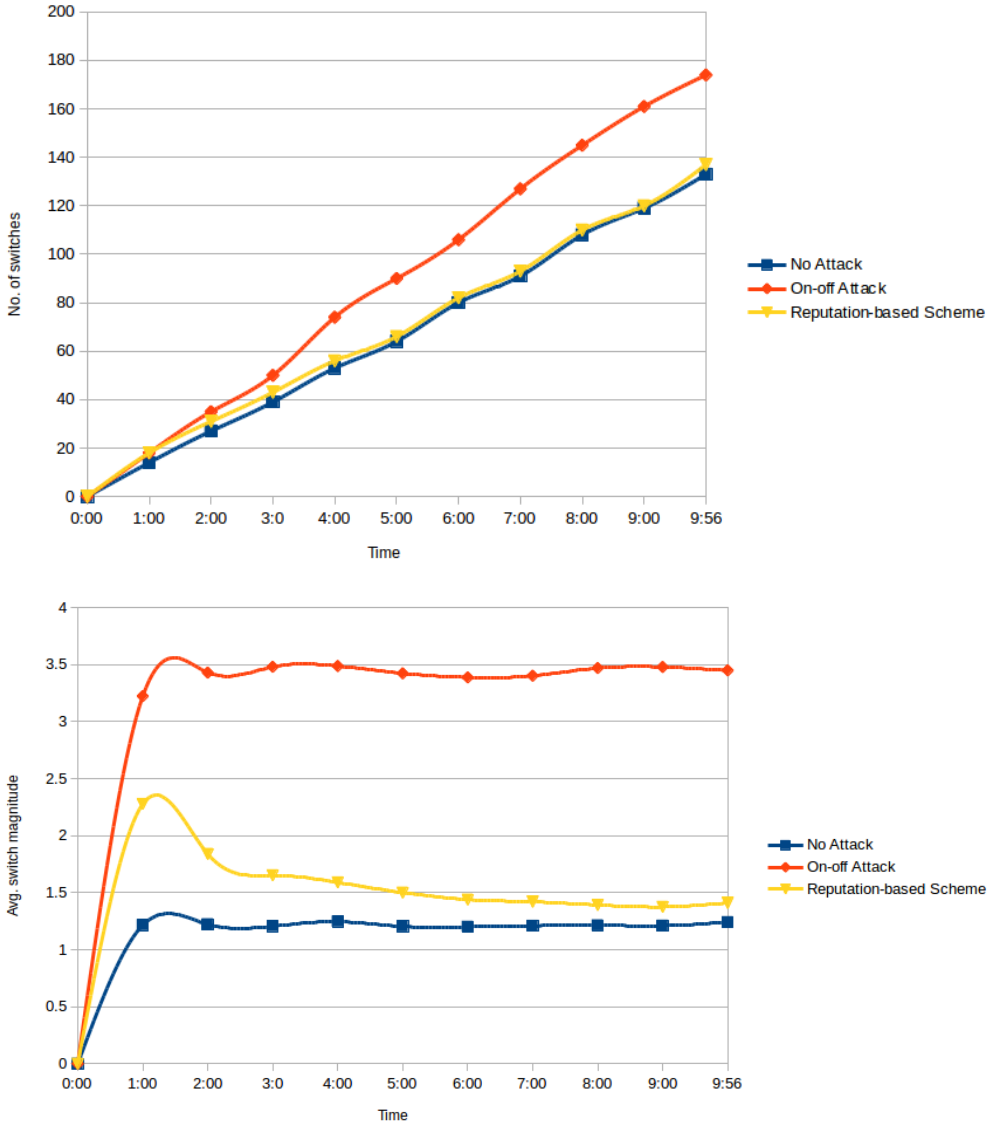


FIGURE 6.19: Experiment 1. Up: Number of switches. Down: Average magnitude

the number of the switches is decreased by 21%, while the average switch amplitude is decreased by 60%. The aforementioned percentages are very satisfactory since the final number of the switches and the average switch amplitude are only 3% and 13% greater than the respective values of the "No Attack" case. Overall, the proposed framework maintains both the number of the switches and the average switch amplitude at a reasonable level, despite the tactic adopted by the attacker.

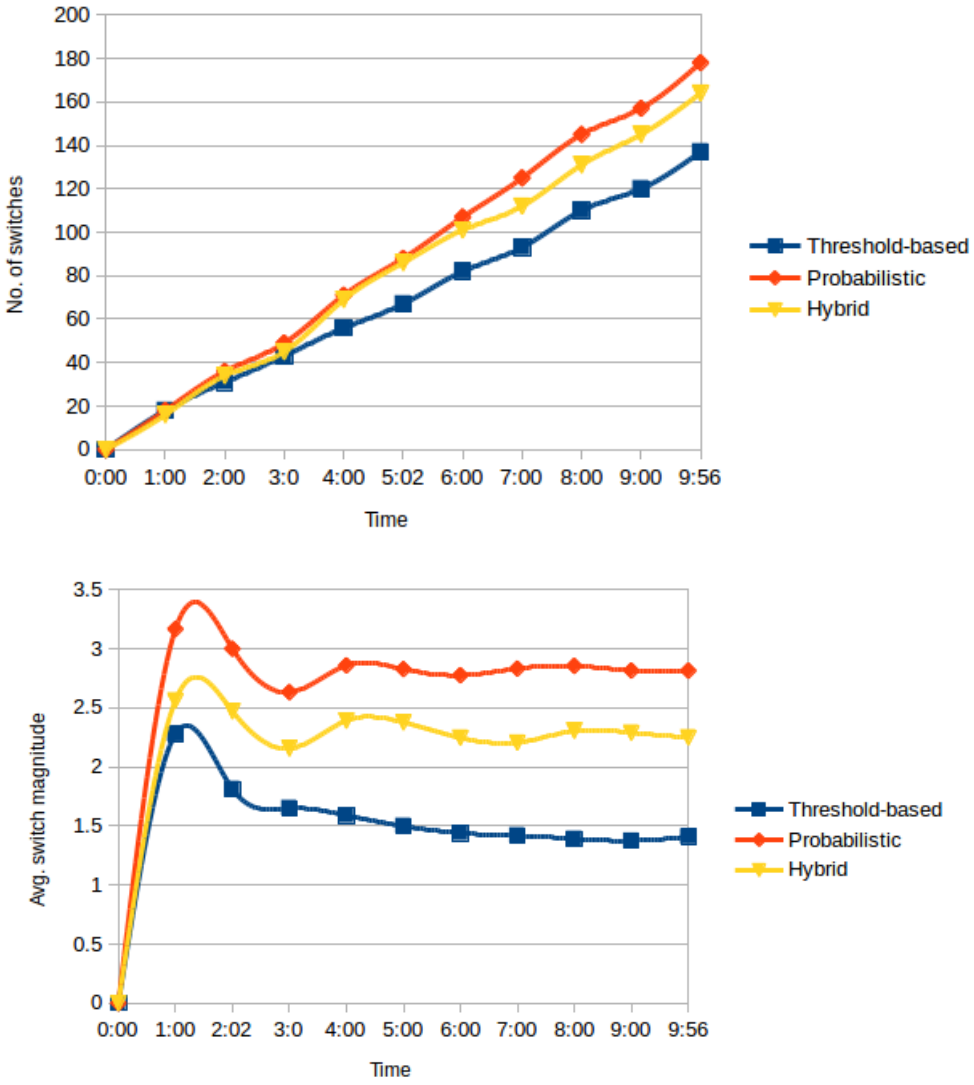


FIGURE 6.20: Experiment 2. Up: Number of switches. Down: Average magnitude

6.3.2 Experiment 2: Caching policies evaluation

Figure 6.20 shows the values of the evaluation metrics over time for the three caching policies: the threshold-based, the probabilistic and the hybrid one. For the threshold-based policy, T value is set to 7, while for the hybrid one, the upper threshold (UT) is set to 7 and the lower (LT) is set to 4.

Although all three policies produce results up to the mark, they have distinct differences. As observed, the threshold-based method seems to generate the best results. This is to be expected since this is the strictest method, explicitly excluding the

content which does not meet the criteria from routers' caches and hence, weakening the attacker.

In comparison, the probabilistic policy produces the least satisfactory results in terms of punishment efficiency. This is mainly due to its flexibility, as it neither explicitly allows the caching of well-reputed content in the network nor prevents the attacker from injecting potentially malicious segments. Moreover, since caching is associated with some probability, a specific segment might be cached only by certain routers of the path. As a consequence, retrieving segments from different locations increases the number of fluctuations.

Finally, the results achieved by the hybrid method lie in-between the performance of the other two. This is an intuitive result, since it combines the logic of the two previous policies, gradually shifting from explicit caching decisions to probabilistic caching.

We note that, on our scenario, all malevolent content - and only that - was assumed to be correctly identified as such by the routers. In this case, the threshold-based policy evidently yields the best results security-wise. However, in reality this might not always be the case and false positives may exist. Therefore, a more flexible caching policy - such as the probabilistic or hybrid one - might prove more appropriate in practice, since it will not drop the legitimate segments which were incorrectly identified as malicious, unnecessarily sacrificing performance.

Overall, we can conclude that our proposed mechanism can indeed provide an effective countermeasure against BOAs and different variations can be implemented, resulting in different trade-offs. As a result, those experiments prove that reputation-based trust mechanisms can indeed prove an effective alternative for security attacks in NDN.

6.4 Content Poisoning Attack mitigation

In this section, we present the results of the experiments regarding our reputation-based incentive method against CPAs. Those experiments are composed of three distinct rounds. First, we present some indicative results regarding some initial experiments concerned with the validation of our mechanism's behavior. Consequently, we present the results obtained from our first round of performance evaluation experiments, which investigate our mechanism's performance against Full-on and On-off attacks for different amounts of initial user coins. Subsequently, we compare the performance of two different reward schemes to identify the best-performing one with respect to attacker punishment.

6.4.1 Initial experiments

We first performed some initial experiments, to assess whether our mechanism exhibits the desired behavior. The questions which we tried to answer by conducting these experiments are the following:

- Does our mechanism assign high maliciousness values to Attackers and low values to Honest users?
- Do the assigned maliciousness values result in lower prestige values for Attackers?
- Do lower prestige values also result in the Attackers mining less coins, when compared to Honest users?

The answers to all of the aforementioned questions are positive. This can be observed in Figures 6.21-6.23, which provide some examples of our mechanism in work. All Figures presented in this section depict Honest users in blue colour and Attackers in orange.

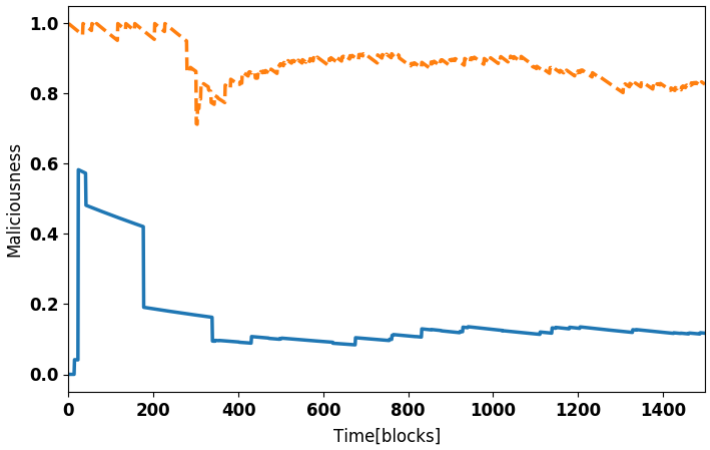


FIGURE 6.21: Maliciousness of an Honest user (blue) versus an Attacker (orange), as it evolves during an experiment. Y-axis: Maliciousness, X-axis: Time

Figure 6.21 depicts maliciousness values as they change over time during an experiment. Time is measured in blocks: 500 blocks is the time instance in which the 500th block was mined in the simulation. The time it takes to mine each block is irrelevant to our work.

It is clear that user maliciousness values accurately reflect user behavior and the Attacker can be clearly identified using its values. Therefore, our system achieves the design goal of assigning high maliciousness values to Attackers and low values to Honest users. But does this higher maliciousness value also result in lower Prestige values for Attackers? The answer is provided in Figure 6.22.

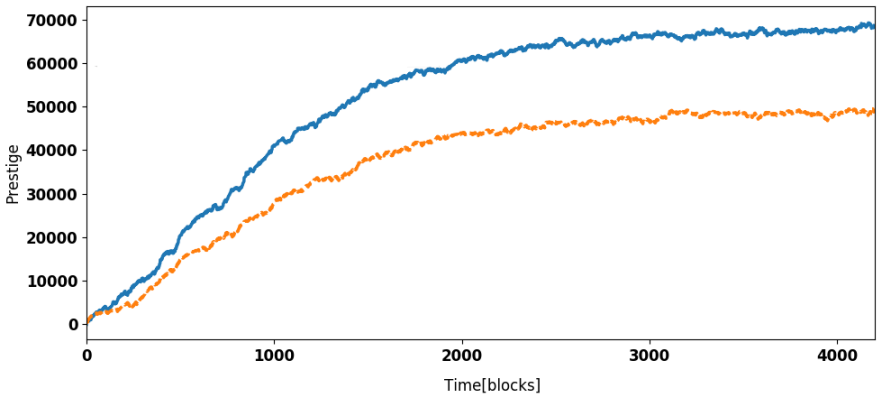


FIGURE 6.22: Prestige of an Honest user (blue) versus an Attacker (orange), as it evolves during an experiment. Y-axis: Prestige, X-axis: Time

Figure 6.22 depicts the evolution of prestige values during an experiment. The Honest user (blue line) clearly achieves higher Prestige values than the Attacker (orange line). Note that the prestige values reach an upper limit after some time due

to the *halving interval* parameter which reduces the number of newly mined coins per block by half, every 500 blocks.

Figure 6.23 demonstrates the number of Coins owned by each user during a simulation. What is interesting, is the fact that even though the Honest user initially starts with 0 coins in its wallet and the Attacker with 200, the former quickly gains more coins than the latter as its honest behavior is rewarded. This is an important feature of our approach: as opposed to typical Proof-of-Stake algorithms, where richer users can gain most of the rewards regardless of their behavior, in our approach, an honest user which is contributing to the file sharing activity may start with less coins but eventually earn more than a malicious one. Thus, our third question is answered: lower prestige values can indeed result in the Attackers mining less coins than Honest users and provide financial counter-incentives against malicious behavior.

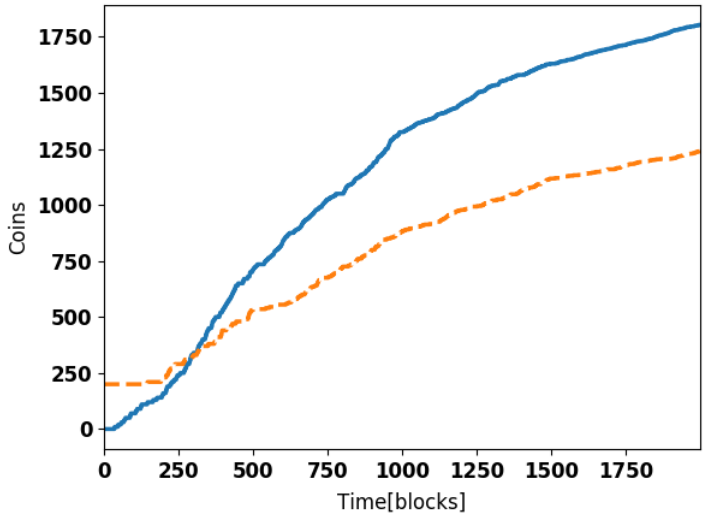


FIGURE 6.23: Coins owned by an Honest user (blue) during an experiment, plotted against those owned by an Attacker (orange). Y-axis: Coins, X-axis: Time.

6.4.2 1st round of evaluation experiments

The initial experiments described in the previous Section validated the fact that our approach achieved the proactive identification of Attackers, which resulted in the Attackers gaining less prestige and, eventually, less coins than Honest users. However, they did not demonstrate how consistent is this behavior on various scenarios. To investigate this issue further, we conducted the experiments presented in this Section.

In those experiments, we evaluated our system against two different scenarios, representing two types of attackers: a "Full-on" Attacker and an "On-off" Attacker. Furthermore, we investigated the influence of the difference in coins between our monitored Attacker and Honest user, and the rest of the honest users. This was achieved by running different experiments, in which we assigned different Initial Coin values to the Attacker and the Honest user. During the 1st round of experiments, we used Version 1 (V1) of the validator selection function and measured the number of Gained Coins during an experiment. We repeated each simulation 20 times and calculated the average values of our metrics, to ensure that the final

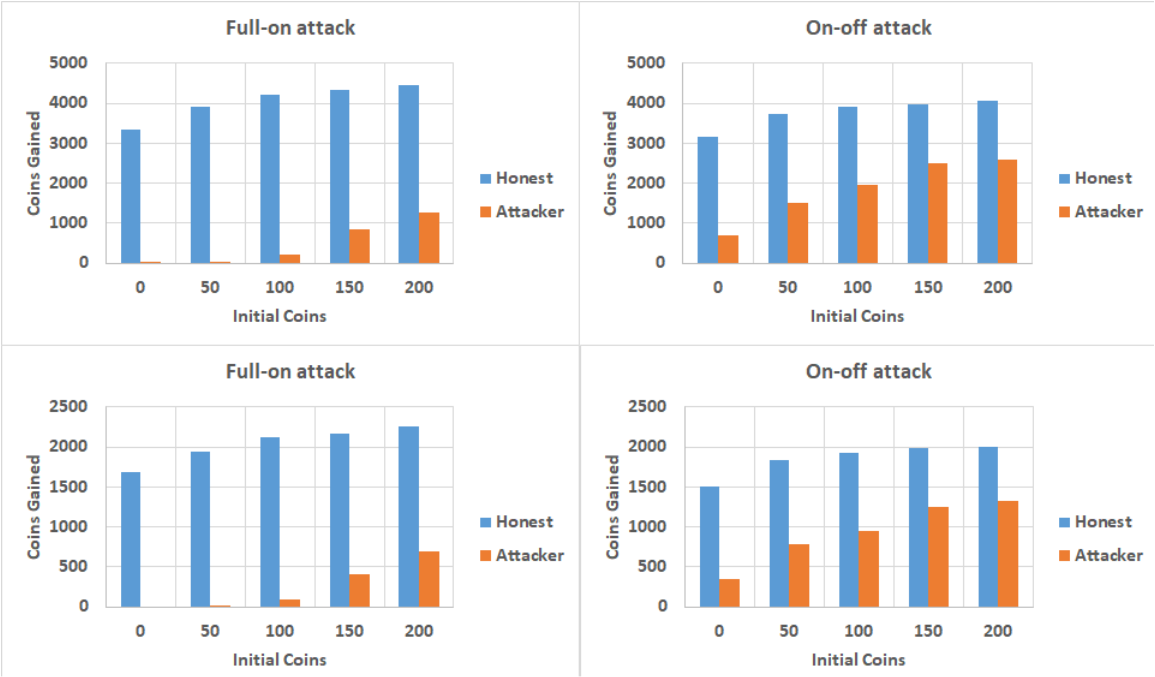


FIGURE 6.24: Average value of coins gained by an Attacker versus an Honest user (y-axis). Different initial amounts of coins in their wallet were used during the experiments (x-axis). Up: The initial amount of coins assigned to the rest of the honest users were equal to 0. Down: The amount ranges between 0 and 200 coins.

results are not affected by the randomized values of parameters, which can yield slightly different values from simulation to simulation.

The results of our first round of experiments are depicted in Figure 6.24. In the upper sub-figures, the "background" honest users started with zero (0) coins in their wallet, while in the lower ones they were randomly assigned an initial amount between zero (0) and two hundred (200) coins. The *Initial Coins* of the Attacker and Honest User are depicted on the x-axis, while the average number of coins earned during the experiments (*Gained Coins*) are depicted on the y-axis.

6.4.3 2nd round of evaluation experiments

In the 1st round of experiments, we confirmed that an Honest user consistently earned more coins than the Attacker. But we did not quantify how much more coins an Honest user earns compared to an Attacker, when the two operate on the same setting. To be able to answer that question across different settings, we introduced the *Gained Coins Ratio* (GCR) (defined in Equation 5.1) and used it to quantify our mechanism's performance in the second round of experiments. The higher its value, the more effective our mechanism is: a GCR=10 means that an honest user will earn 10 times more coins than an attacker on the same setting.

During the 2st round of experiments, we used both the V1 and V2 validator selection algorithm, comparing their results against each other. The results of the second round of experiments can be seen in Figure 6.25. As was also the case in Figure 6.24, in the experiments depicted in the upper sub-figures the "background" honest users started the simulations with zero (0) initial coins in their wallet, while

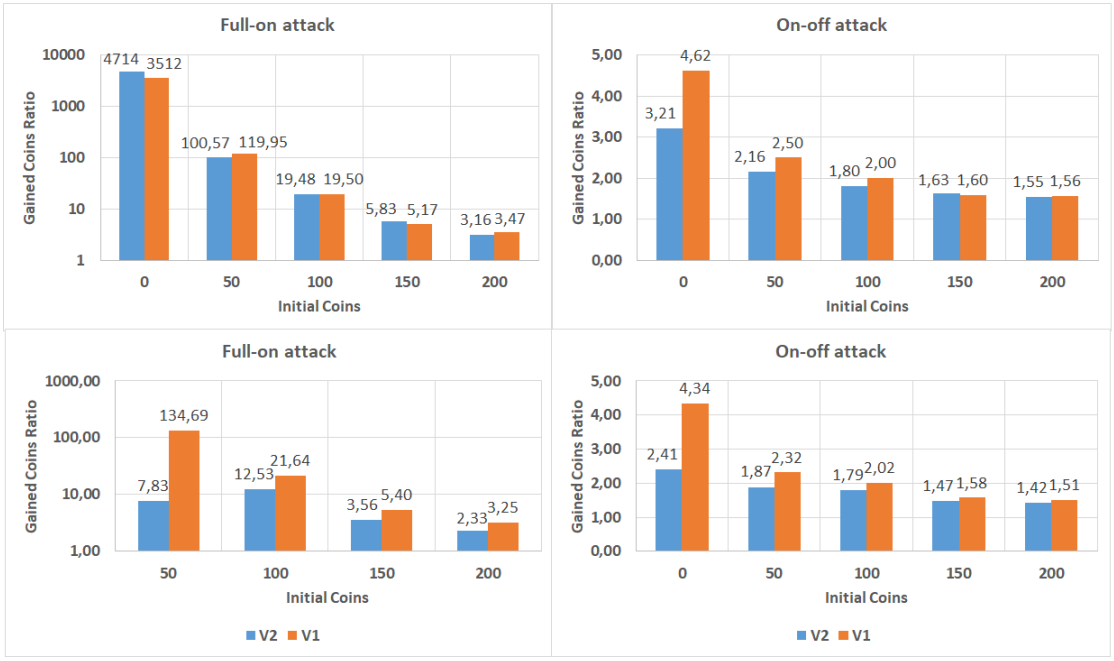


FIGURE 6.25: Average value of the Honest-to-Attacker Gained Coins Ratio (y-axis). Different initial amounts of coins in their wallet were used during the experiments, depicted on the x-axis. Up: The initial amount of coins assigned to the rest of the honest users is equal to 0. Down: The amount ranges between 0 and 200 coins.

in the lower sub-figures they were randomly assigned an initial amount between 0 and 200 coins.

6.4.4 Summary of experimental findings

Our experimental results demonstrate that:

- When comparing between Honest users and Attackers starting with the same amount of coins, honest users consistently earn more coins on average than the attackers. Therefore, our reputation system indeed punishes malicious behaviour and provides counter-incentives for attackers.
- Both Full-on and On-Off Attackers fare worse than Honest users. However, Full-On Attackers mint less coins on average than Full-on Attackers, therefore our algorithm can achieve different levels of punishment for different levels of malicious behaviour, which is a desired outcome.
- No significant difference in the behavior of our mechanism is observed when comparing the upper and lower part of Figures 6.24 and 6.25 (which correspond to different amount of initial coins for the other users). Of note is only the fact that in the lower part of those Figures, both the Attacker and Honest users earn less overall coins than in the upper-figure case.
- Our system fares better at attack mitigation when users start with less initial coins, as the difference between the coins minted by the Attacker and the Honest user, is far more pronounced (when compared to the experiments in which users have more initial coins). As the amount of coins earned by Honest users does not change significantly when having more initial coins, this is mostly

due to the Attackers earning more for larger initial coin values. This can be explained by the fact that more initial coins produce higher initial 'prestige' values, which increase the minting chance. Therefore, a higher percentage of the 'gained' coins are earned due to the users being 'richer' (having more coins, i.e. stake) than due to the user contributing with useful work to the network.

- When users start with a higher amount of initial coins, they earn a higher amount of overall coins during the mining procedure (regardless of whether their behaviour is honest or not).

Overall, our experimental results demonstrate that Honest users consistently earn more coins than attackers in all cases. An important feature is the fact that it is effective against both Full-on and On-off attacks, even though its efficiency diminishes in the case of On-off attacks.

When considering On-off attacks, honest users earned on average 142-434% more coins that the attackers, while in the Full-on attack case their honest behavior rewarded them with at least 325% more coins (reaching up to a 471,200% increase (!) in a certain experiment). Therefore, we can conclude that our approach is indeed successful in punishing malicious behavior.

An interesting observation is the fact that our system seems to fare better at attack mitigation when users start with less initial coins. This difference can be attributed to the fact that the chance of mining new coins is partly dependent on the user's existing coins and partly due to the useful work the user performs, as both influence the user's prestige. Therefore, when a user owns a large initial amount of coins, a high percentage of its new coins will be earned due to the user being richer (similar to Proof-of-Stake algorithms), than due to him contributing with useful work.

Last, based on the results of our second round of experiments, we can conclude that V1 of our algorithm is more effective than V2 and, thus, *prestige* can work sufficiently well as the sole input when selecting the next block validator.

Chapter 7

Conclusions

The primary goal of this dissertation has been the extension of the Named Data Networking architecture. Our aim was to address issues which were not foreseen by the original design and currently hinder the architecture's adoption. Our work focuses in two such aspects: The first one is the NDN's operation in delay/disruption-tolerant networks, which was not feasible by the original design that targeted fixed networks. The second one is the introduction of reputation-based trust mechanisms, to complement NDN's credential-based mechanisms in cases where the latter can fail in trust establishment or attack mitigation.

We addressed those aspects by designing several solutions and evaluating their effectiveness. The conclusions that were drawn by this process are summarized in the rest of this section.

Delay/Disruption Tolerance: The first mechanism presented in this doctoral thesis is NoD, a solution developed to allow NDN's operation in delay/disruption-tolerant scenarios. NoD consists of an NDN/DTN protocol stack and a deployment strategy targeting edge nodes, designed to allow compatibility with the original architecture and an easy integration in existing deployments.

This mechanism was deployed in two distinct scenarios, one for validation and a second one for performance evaluation purposes. The former consisted of a mobile edge computing scenario, which NoD supported by enabling service migration and deployment in intermittently-connected Access Points. The latter consisted of an IoT scenario, in which sensor measurement retrieval from an intermittently-connected Wireless Sensor Network was facilitated by NoD.

A mathematical model was also developed to capture our system's behavior from a theoretical perspective. The accuracy of this model was demonstrated during the experimental process.

The performance evaluation took place in two stages. The first stage included the deployment of our original NoD implementation (*NoD-Default*) in a topology including a dedicated data mule which emulated periodic movement of a node between two wireless networks. This stage consisted of five distinct experiments. In the first two we established the meaningfulness of our approach by comparing the performance of NoD with a host-centric DTN implementation. In the next three, we evaluated the impact of different parameters in our system's behavior.

The first round of experiments demonstrated that NoD can outperform host-centric DTN approaches and significantly lower the content retrieval delay. In this respect, the Cache Hit Ratio achieved by in-network caches largely determines NoD's performance improvement. In those experiments, the choice of Interest Lifetime by NDN Consumers did not exert an important influence on the total retrieval delay, as that was solely determined by the duration of the data mule's round trip time. The selection of a Data Freshness period below a certain threshold, however, can lower performance as the Cache Hit Ratio drops and the Average Delay decreases.

Furthermore, we experimented with different contact duration values to benchmark our specific implementation's performance for short-contact scenarios. In this experiment, we established the particular requirements for efficient operation of our NoD implementation. One such requirement, is the need for a Contact Duration larger than roughly 5 seconds to guarantee packet exchange. A second requirement is the low amount of exchanged NDN packets, especially in scenarios with transient contacts (e.g. 5 seconds); we realized that the transmission of more than 4 Interest /Data packets-per-contact caused a disproportional increase in Round Trips needed to retrieve the content objects, impacting severely the Average Delay. When setting the Contact Duration to values larger than 10 seconds, this property became less significant since NoD appears capable of handling more traffic.

Based on the results of the previous experiments, we created an improved version of our mechanism (*NoD-Discovery*) which leveraged cross-layer information exchange between the NDN and DTN layers. Our primary goal was to reduce unnecessary transmissions and, therefore, the network overhead. This version was evaluated in the second stage of experiments. In this setup, we deployed a second data mule and introduced movement patterns which created opportunistic contacts between nodes aiming to create a more challenging environment. In those experiments, our goal was to quantify the improvement achieved by our new version of the mechanism by comparing the performance of *NoD-Discovery* and *NoD-Default*.

This round of experiments showed that NoD-Discovery can indeed reduce the network overhead significantly, without compromising performance with respect to the average data retrieval delay. The impact of the cross-layer interaction becomes more significant when the Interest Lifetime becomes shorter, as the Overhead Ratio was reduced up to 70.34% in the 10-second case. Its impact is also larger when the delay due to data muling increases, achieving a 57.32% decrease of the Overhead Ratio in the case of 40-second State Duration case. We also demonstrated that the use of shorter Interest Lifetime values in delay-/disruption-tolerant scenarios are feasible, but will result in a larger network overhead in both NoD versions. Nevertheless, the reduction of retransmitted packets over the cross-layer design is notable even in short-connectivity scenarios.

In both experiments, there was a slight improvement on the Interest Satisfaction Ratio and Cache Hit Ratio by NoD-Discovery. However, we make no special mention on those results as they did not result from an increase on the cache hits or the number of Interests Satisfied. The observed improvement is only a reflection of the decrease in the metrics' denominators, i.e. the number of sent packets (which is measured by the Overhead Ratio).

Overall, we demonstrated that NoD can be deployed in disruptive IoT environments and outperforms host-centric DTNs in remote content retrieval. Finally, during all experimental rounds we confirmed our model's soundness, as our implementation's behavior - as demonstrated by the experimental results - indeed match the one predicted by our model.

Besides our work which explicitly focused on delay and disruption tolerant scenarios, we have also performed some relevant experiments on well-connected fixed networks. Those experiments were motivated by the use of large packet sizes in the service migration scenario and explored the impact of different NDN chunk sizes on fixed network performance.

Our results show that using larger chunk sizes is feasible, but offers some tradeoffs. For the same amount of data being retrieved, bigger data chunks mean that cache replacement policies assert less impact on network performance. Furthermore,

network throughput can increase, as the use of bigger chunks introduces less overhead per chunk. However, the average delay also increases, due to an increased processing requirements. Therefore, using bigger chunk sizes seems to be better suited for certain environments (such as bulk data transfers), while smaller chunk sizes seem to be better-suited for delay-sensitive applications.

Reputation-based Trust: The second set of mechanisms included in this dissertation are concerned with the use of reputation systems to enhance NDN trust and security. Two different reputation systems are designed and evaluated, each one targeting a different use case.

The first use case is concerned with adaptive multimedia streaming over NDN. In particular, Bitrate Oscillation Attacks targeting the Dynamic Adaptive Streaming protocol are considered and a reputation-based countermeasure is proposed to mitigate such attacks.

This mechanism leverages router ratings for the content they receive. Attack mitigation is achieved by allowing routers to adjust their caching policy, based on a multimedia object's reputation. Specifically, our design mandates that routers compute a reputation value for each multimedia content object depending on whether their received segments were consecutive or not. Subsequently, those reputation values are used for decision-making purposes, enabling routers to make informed decisions on whether they will cache the next segments of the specific multimedia content or not.

Our proposal is evaluated on the AMuSt-ndnSIM simulator. In the first experiment, a threshold-based caching policy was used by the routers so that only trustworthy segments are cached. The results demonstrate that, even though an On-Off attack model was employed on the attacker side to 'fool' our mechanism using legitimate behavior, the impact of an ongoing Bitrate Oscillation Attack is efficiently mitigated. Particularly, the number of the switches is decreased by 21%, while the average switch amplitude is decreased by 60% (results which only slightly differ from the no-attack case, indicating a high degree of efficiency).

In the second experiment, we explored the use of three different caching policies: a threshold-based policy, a probabilistic policy and a hybrid policy. Our results show that the threshold-based policy is the most efficient one with respect to attack mitigation, as it uses a "hard" limit which explicitly avoids caching of malicious segments.

The second use case is concerned with Content Poisoning Attacks, an attack type originally present in Peer-to-Peer file sharing networks which also plagues the NDN architecture. Our design leverages blockchain to create a reputation system which enables attacker identification and incentivizes trustworthy behavior. To this end, the Proof-of-Prestige reward system is enhanced to accommodate reputation systems and simulate Content Poisoning Attacks.

To evaluate our proposal, we performed simulation experiments using the Proof-of-Prestige simulator. We simulated Peer-to-Peer networks with 100 users, which included an attacker sending malicious content to the other users while the rest of the peers behaved honestly. Two different attack types were considered: a Full-on Attack and an On-Off attack.

Our initial experiments validated that our system exhibits the desired behavior, i.e. it assigned high maliciousness values to Attackers and low ones to honest users, while simultaneously resulting in lower prestige values for attackers and less coins being mined by them.

On next round of evaluation experiments we used our initial version of the validator selection algorithm and quantified the number of coins each user gained during an experiment. This experiment demonstrated that honest users consistently earn more coins than attackers in all cases, as well as that our mechanism is effective against both Full-on and On-off attacks.

On the final round of experiments, we experimented with a new version of the validator selection algorithm and compared their versions' results against each other. This experiment suggests that the initial version of the algorithm is more effective than the second one, indicating that effective incentivisation can be achieved prestige can be the sole criterion to be used when the next block validator is selected.

Overall, we posit that the entirety of our work has sufficiently demonstrated: a) the feasibility of introducing delay/disruption tolerance in NDN and the efficiency of using such an NDN over DTN protocol stack and b) the capability of reputation-based trust mechanisms to supplement the existing NDN security and trust fabric, as demonstrated by their use in two distinct use cases.

Discussion: Even though this dissertation proposed and evaluated a particular way of supporting delay tolerance in NDN, there do exist alternative approaches that could be followed - each one with its own trade-offs. Those aspects are related to the discussion in Section 2.3, which presents the existing literature for designing and deploying Information-Centric Delay Tolerant Networks.

Focusing specifically on delay-tolerant NDN, if interoperability with the original architecture is not one of the design goals, modifications to the NDN components and semantics could enable the operation of the architecture in delay and disruption tolerant environments. Such an approach can allow in-network caching by the data mules, a feature which can increase performance but is not offered by NoD (as caching functionality is only supported on edge nodes).

In this dissertation, we approached delay-/disruption-tolerant networks not as isolated environments, but as networks which can (and should) be integrated with the global Internet. This perspective, combined with a second assumption that global Internet-wide communication takes place using NDN, led us to embrace compatibility with the original NDN architecture as a design goal for any developed solution. However, more than a decade has passed since the CCN/NDN architecture's inception and wide-spread adoption has yet to be seen. Therefore, a relaxation of this assumption can be deemed as reasonable in future works.

Nevertheless, even if IP remains at the core of the global Internet architecture, NDN can still find its place in more localized deployments. A major factor inhibiting the adoption of any clean-slate architecture is its deployment cost. However, environments in which non-IP networks can still be deployed with a relatively low deployment cost do exist: one such case is formed by ad-hoc networks and, in particular, mobile opportunistic networks, as such networks are not dependent on any existing network infrastructure. They can be locally created in a spontaneous manner and do not require global-scale Internet connectivity. As long as the devices forming the network (e.g. smartphones, IoT devices) can support NDN, communications within them can take place using NDN semantics.

With those assumptions in mind, compatibility with the original NDN architecture need not pose a major constraint. Therefore, the opportunity for heavily-modified NDN versions to be implemented and evaluated as a more efficient means to handle communications in such network deployments is an interesting research direction in the area of Information Centric Delay Tolerant Networks.

As a last remark, the NDN extensions proposed in this dissertation - reputation and delay tolerance - were pursued independently of each other. However, future

works can investigate the capability of jointly incorporating them as architectural features. One such direction could be the use of reputation to develop information-centric routing protocols for delay-tolerant networks e.g. through the incorporation and use of node and/or content reputation scores for routing decisions.

Appendix A

Mathematical Proofs

A.1 Proof of Interest Satisfaction Ratio

A.1.1 Definitions

The following metrics are defined:

(i) The *Interest Satisfaction Ratio* (ISR), for NDN Consumers:

$$ISR = SatisfiedInterests/SentInterests$$

where $SatisfiedInterests$ is the number of Interests which were satisfied during a time interval and $SentInterests$ is the total number of Interests the Consumer has sent during that interval.

(ii) The *Cache Hit Ratio*, for NDN nodes:

$$h = CacheHits/ReceivedInterests$$

where $CacheHits$ is the number of cache hits during a certain time interval and $ReceivedInterests$ is the total number of Interests received by the node during that interval.

(iii) The bundle *Delivery Ratio*, for the intermittently connected network part:

$$DR = DeliveredBundles/SentBundles$$

where $DeliveredBundles$ is the number of DTN bundles which were delivered to their destination by a Delay-Tolerant Network during a certain time interval and $SentBundles$ is the total number of bundles which were sent by DTN nodes during that interval.

We integrate the metrics above into a single model, in which a Consumer's *Interest Satisfaction Ratio* in NDN-over-DTN can be calculated by:

$$ISR = h + (1 - h) * DR^2 \tag{A.1}$$

To calculate the ISR, we model our experiment as a Bernoulli trial and define as success the satisfaction of an Interest packet (an unsatisfied Interest being a failure, conversely). An Interest will be satisfied in the following cases: A) If the corresponding content exists in the local gateway cache, or B) If the content does not exist in cache, the Interest is forwarded through the DTN face. The Interest will be satisfied only if the Interest is successfully received by the Producer and the corresponding packet is successfully received by the Consumer.

The probability that A is true is equal to the *Cache Hit Ratio*, therefore $p_A = h$ The probability that the content does not exist in the local cache is equal to $q_A = 1 - p_A = 1 - h$. In this case, the content will be received by the Consumer only if both the bundles encapsulating the Interest and Data packets are successfully delivered by the intermittently connected network part. Each packet's delivery probability is equal to the network *Delivery Ratio* (DR), therefore the probability that both the Interest and subsequent Data packet are delivered is equal to DR^2. Hence, the probability that case B is true, is equal to $p_B = (1 - h) * DR^2$.

Based on the above, the overall probability that an Interest is satisfied can be expressed as: $p_s = p_A + p_B = h + (1 - h) * DR^2$. As, by definition, $ISR = p_s$, the *Interest Satisfaction Ratio* can also be described by the following form of the equation:

$$ISR = h + (1 - h) * DR^2$$

A.2 Proof of Overhead Ratio

The *Overhead Ratio* can also be expressed, alternatively, in the form found in Equation 3.9:

$$OR = \frac{InterestsSent - DataReceived}{DataReceived} \Leftrightarrow$$

$$OR = \frac{InterestsSent}{DataReceived} - 1 \Leftrightarrow$$

$$OR = \frac{InterestsSent}{InterestsSatisfied} - 1 \Leftrightarrow$$

$$OR = \frac{1}{ISR} - 1$$

A different form, as expressed in Equation 3.10, can also be reached by taking into account that the number of Interests transmitted via the DTN channel, is equal to the total number of Interests sent by the Consumer, minus the Interests which are being satisfied by its local cache. This is true for our setup, as DTN is the only available transport between the Producer and Consumer regions. Therefore,

$$T_{int} = S - C \Leftrightarrow$$

$$S = T_{int} + C$$

By substituting S into Equation 3.8, we can arrive at Equation 3.10 the following way:

$$OR = \frac{S - R}{R} \Leftrightarrow$$

$$OR = \frac{S}{R} - 1 \Leftrightarrow$$

$$OR = \frac{T_{int} + C}{R} - 1 \Leftrightarrow$$

$$OR = \frac{1}{R} * T_{int} + \frac{C}{R} - 1 \Leftrightarrow$$

$$OR = \frac{1}{R} * T_{int} + \frac{C/S}{R/S} - 1 \Leftrightarrow$$

$$OR = \frac{1}{R} * T_{int} + \frac{h}{ISR} - 1 \Leftrightarrow$$

A.3 Average Delay

Another important network metric is the delay. The *Average Delay* (D_{avg}) for receiving a content object from intermittently-connected devices, can be calculated by using:

$$D_{avg} = (1 - \frac{h}{ISR}) * D_{p_{avg}} + (\frac{h}{ISR}) * D_{c_{avg}} \tag{A.2}$$

where h is the local (Consumer-side) *Cache Hit Ratio*, while $D_{p_{avg}}$ and $D_{c_{avg}}$ are the average delays for retrieving a content object from a remote Producer or local cache, respectively.

The *Average Delay* is affected by: (i) The respective number of content objects a Consumer receives either by a remote Producer or by a local cache and (ii) The corresponding delay for receiving a content object from either the remote Producer or local cache.

A cache can only store a limited number of content objects and satisfy a given number of requests. The latter, as a percentage of the total requests, is defined as the *Cache Hit Ratio* (h).

Without loss of generality, we can assume that all requests sent by NDN Consumers located in the same network, go through a NDN-over-DTN gateway which forwards them through the DTN face to remote NDN devices/networks and caches all content objects it receives. Therefore, h will subsequently refer to the NoD gateway *Cache Hit Ratio*.

Now, let S is the total number of Interests sent by a local Consumer, R the total number of Data packets received by the Consumer, C the number of Data packets retrieved from the local cache and P is the total number of packets retrieved by a remote Producer. In our experiments, the Interests received by the NDN-over-DTN gateway will be equal to S and the following equations are true:

$$R = P + C$$

$$h = \frac{C}{S}$$

$$ISR = \frac{R}{S}$$

Let D_{total} be the total content retrieval delay (for all retrieved content objects). D_{total} can be expressed as:

$$D_{total} = D_P + D_C$$

where D_P is the total delay of all packets received by a remote Producer and D_C the delay for all packets received by a local cache, which can be calculated by using the following equations:

$$D_P = \sum_{i=1}^{P} D_{p_i}$$

$$D_C = \sum_{j=1}^{C} D_{c_i}$$

The average value of the delay for each retrieved content object can be calculated by dividing the total delay for all objects by the number of retrieved content objects:

$$D_{avg} = \frac{1}{R} * D_{total}$$

In the same manner, we can calculate the average delay of remote Producer-retrieved content objects ($D_{P_{avg}}$) and local cache-retrieved content objects ($D_{C_{avg}}$) as follows:

$$D_{P_{avg}} = \frac{1}{P} * D_P$$

$$D_{C_{avg}} = \frac{1}{C} * D_C$$

Using the equations above, the form for D_{avg} presented in Section 3.5 can be derived as follows:

$$D_{avg} = \frac{1}{R} * D_{total} \Leftrightarrow$$

$$D_{avg} = \frac{1}{R} * (D_P + D_C) \Leftrightarrow$$

$$D_{avg} = \frac{1}{R} * (P * D_{P_{avg}} + C * D_{C_{avg}}) \Leftrightarrow$$

$$D_{avg} = \frac{P}{R} * D_{P_{avg}} + \frac{C}{R} * D_{C_{avg}} \Leftrightarrow$$

$$D_{avg} = \frac{R - C}{R} * D_{P_{avg}} + \frac{C}{R} * D_{C_{avg}} \Leftrightarrow$$

$$D_{avg} = (1 - \frac{C}{R}) * D_{P_{avg}} + (\frac{C}{R}) * D_{C_{avg}} \Leftrightarrow$$

$$D_{avg} = (1 - \frac{C/S}{R/S}) * D_{P_{avg}} + (\frac{C/S}{R/S}) * D_{C_{avg}} \Leftrightarrow$$

$$D_{avg} = (1 - \frac{h}{ISR}) * D_{P_{avg}} + (\frac{h}{ISR}) * D_{C_{avg}}$$

A.4 Average number of Round Trips

Let $n_{RT_{total}}$ be the total number of Round Trips needed to retrieve a certain number of content objects. The average number of Round Trips needed to retrieve each object can be calculated in two separate ways: either by only taking into account the objects received by the Producer, or by taking into account all objects, including the objects received by the local cache. All equations presented in Section 3.5 use the former version, denoted in this Appendix as $n_{RT_{avg}}(R)$. The latter version, denoted in this Appendix as $n_{RT_{avg}}(R)$, is only used to plot Figure 6.1 for comparison purposes.

The relationship between the two parameters is the following: Let $n_{RT_{avg}}(P)$ be the number of Round Trips when calculated only for Producer-received objects. In this case:

$$n_{RT_{avg}}(P) = \frac{n_{RT_{total}}}{P}$$

where P is the number of Producer-received objects.

Now, let $n_{RT_{avg}}(R)$ be the number of Round Trips when calculated for all received content objects. In this case:

$$n_{RT_{avg}}(R) = \frac{n_{RT_{total}}}{R}$$

where R is total number of received objects.

By combining the aforementioned equations, we can derive:

$$\frac{n_{RT_{avg}}(R)}{n_{RT_{avg}}(P)} = \frac{\frac{n_{RT_{total}}}{R}}{\frac{n_{RT_{total}}}{P}} \Leftrightarrow$$

$$\frac{n_{RT_{avg}}(R)}{n_{RT_{avg}}(P)} = \frac{P}{R} \Leftrightarrow$$

$$\frac{n_{RT_{avg}}(R)}{n_{RT_{avg}}(P)} = \frac{R-C}{R} \Leftrightarrow$$

$$\frac{n_{RT_{avg}}(R)}{n_{RT_{avg}}(P)} = 1 - \frac{C}{R} \Leftrightarrow$$

$$\frac{n_{RT_{avg}}(R)}{n_{RT_{avg}}(P)} = 1 - \frac{C/S}{R/S} \Leftrightarrow$$

$$\frac{n_{RT_{avg}}(R)}{n_{RT_{avg}}(P)} = 1 - \frac{h}{ISR} \Leftrightarrow$$

$$n_{RT_{avg}}(R) = (1 - \frac{h}{ISR}) * n_{RT_{avg}}(P)$$

Bibliography

[1] T. Barnett, S. Jain, U. Andra, and T. Khurana, "Cisco visual networking index (vni) complete forecast update, 2017–2022", *Americas/EMEAR Cisco Knowledge Network (CKN) Presentation*, pp. 1–30, 2018.

[2] G. Xylomenos, C. N. Ververidis, V. A. Siris, *et al.*, "A survey of information-centric networking research", *IEEE communications surveys & tutorials*, vol. 16, no. 2, pp. 1024–1049, 2013.

[3] L. Zhang, A. Afanasyev, J. Burke, *et al.*, "Named data networking", *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 66–73, 2014.

[4] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content", in *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '09, Rome, Italy: Association for Computing Machinery, 2009, pp. 1–12, ISBN: 9781605586366. DOI: 10.1145/1658939.1658941. [Online]. Available: https://doi.org/10.1145/1658939.1658941.

[5] K. Fall, "A delay-tolerant network architecture for challenged internets", in *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM '03, Karlsruhe, Germany: Association for Computing Machinery, 2003, pp. 27–34, ISBN: 1581137354. DOI: 10.1145/863955.863960. [Online]. Available: https://doi.org/10.1145/863955.863960.

[6] D. Han, A. Anand, F. Dogar, *et al.*, "Xia: Efficient support for evolvable internetworking", in *9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*, 2012, pp. 309–322.

[7] J. Day, I. Matta, and K. Mattar, "Networking is ipc: A guiding principle to a better internet", in *Proceedings of the 2008 ACM CoNEXT Conference*, 2008, pp. 1–6.

[8] T. Anderson, K. Birman, R. Broberg, *et al.*, "The nebula future internet architecture", in *The Future Internet Assembly*, Springer, 2013, pp. 16–26.

[9] J. Pan, S. Paul, and R. Jain, "A survey of the research on future internet architectures", *IEEE Communications Magazine*, vol. 49, no. 7, pp. 26–36, 2011.

[10] N. Laoutaris and C. Iordanou, "What do information centric networks, trusted execution environments, and digital watermarking have to do with privacy, the data economy, and their future?", *SIGCOMM Comput. Commun. Rev.*, vol. 51, no. 1, pp. 32–38, Mar. 2021, ISSN: 0146-4833. DOI: 10.1145/3457175.3457181. [Online]. Available: https://doi.org/10.1145/3457175.3457181.

[11] M. Gritter and D. R. Cheriton, "An architecture for content routing support in the internet", in *3rd USENIX Symposium on Internet Technologies and Systems (USITS 01)*, 2001.

[12] A. Carzaniga and A. L. Wolf, "Content-based networking: A new communication infrastructure", in *Workshop on Infrastruture for Mobile and Wireless Systems*, Springer, 2001, pp. 59–68.

[13] A. Carzaniga, M. J. Rutherford, and A. L. Wolf, "A routing scheme for content-based networking", in *IEEE INFOCOM 2004*, IEEE, vol. 2, 2004, pp. 918–928.

[14] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf, "Design and evaluation of a wide-area event notification service", *ACM Transactions on Computer Systems (TOCS)*, vol. 19, no. 3, pp. 332–383, 2001.

[15] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana, "Internet indirection infrastructure", in *Proceedings of the 2002 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 2002, pp. 73–86.

[16] I. Stoica, R. Morris, D. Liben-Nowell, *et al.*, "Chord: A scalable peer-to-peer lookup protocol for internet applications", *IEEE/ACM Transactions on networking*, vol. 11, no. 1, pp. 17–32, 2003.

[17] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, I. Stoica, and S. Shenker, "Rofl: Routing on flat labels", in *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, 2006, pp. 363–374.

[18] P. Francis and R. Gummadi, "Ipnl: A nat-extended internet architecture", *SIGCOMM Comput. Commun. Rev.*, vol. 31, no. 4, pp. 69–80, Aug. 2001, ISSN: 0146-4833. DOI: 10.1145/964723.383065. [Online]. Available: https://doi.org/10.1145/964723.383065.

[19] T. Koponen, M. Chawla, B.-G. Chun, *et al.*, "A data-oriented (and beyond) network architecture", in *Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, 2007, pp. 181–192.

[20] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content", in *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, 2009, pp. 1–12.

[21] L. Zhang, A. Afanasyev, J. Burke, *et al.*, "Named data networking", *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 66–73, 2014.

[22] S. Paul, R. Yates, D. Raychaudhuri, and J. Kurose, "The cache-and-forward network architecture for efficient mobile content delivery services in the future internet", in *2008 First ITU-T Kaleidoscope Academic Conference-Innovations in NGN: Future Network and Services*, IEEE, 2008, pp. 367–374.

[23] D. Raychaudhuri, K. Nagaraja, and A. Venkataramani, "Mobilityfirst: A robust and trustworthy mobility-centric architecture for the future internet", *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 16, no. 3, pp. 2–13, 2012.

[24] C. Dannewitz, D. Kutscher, B. Ohlman, S. Farrell, B. Ahlgren, and H. Karl, "Network of information (netinf)–an information-centric networking architecture", *Computer Communications*, vol. 36, no. 7, pp. 721–735, 2013.

[25] D. Trossen, M. J. Reed, J. Riihijärvi, M. Georgiades, N. Fotiou, and G. Xylomenos, "Ip over icn-the better ip?", in *2015 European conference on networks and communications (EuCNC)*, IEEE, 2015, pp. 413–417.

[26] G. Xylomenos, Y. Thomas, X. Vasilakos, *et al.*, "Ip over icn goes live", in *2018 European Conference on Networks and Communications (EuCNC)*, IEEE, 2018, pp. 319–323.

[27] G. Carofiglio, L. Muscariello, J. Augé, M. Papalini, M. Sardara, and A. Compagno, "Enabling icn in the internet protocol: Analysis and evaluation of the hybrid-icn architecture", in *Proceedings of the 6th ACM Conference on Information-Centric Networking*, 2019, pp. 55–66.

[28] Y. Thomas, N. Fotiou, S. Toumpis, and G. C. Polyzos, "Improving mobile ad hoc networks using hybrid ip-information centric networking", *Computer Communications*, vol. 156, pp. 25–34, 2020.

[29] E. Daniel and F. Tschorsch, "Ipfs and friends: A qualitative comparison of next generation peer-to-peer data networks", *IEEE Communications Surveys & Tutorials*, vol. 24, no. 1, pp. 31–52, 2022.

[30] J. Benet, "Ipfs-content addressed, versioned, p2p file system", *arXiv preprint arXiv:1407.3561*, 2014.

[31] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the xor metric", in *International Workshop on Peer-to-Peer Systems*, Springer, 2002, pp. 53–65.

[32] "Filecoin: A decentralized storage network", Protocol Labs, Tech. Rep., Jul. 2017. [Online]. Available: https://filecoin.io/filecoin.pdf.

[33] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking", *IEEE Communications Magazine*, vol. 50, no. 7, pp. 26–36, 2012.

[34] V. Jacobson, M. Mosko, D. Smetters, and J. Garcia-Luna-Aceves, "Content-centric networking", *Whitepaper, Palo Alto Research Center*, pp. 2–4, 2007.

[35] B. Wissingh, C. A. Wood, A. Afanasyev, L. Zhang, D. R. Oran, and C. Tschudin, *Information-centric networking (icn): Content-centric networking (ccnx) and named data networking (ndn) terminology*, RFC 8793, Jun. 2020. DOI: 10.17487/RFC8793. [Online]. Available: https://www.rfc-editor.org/info/rfc8793.

[36] A. Afanasyev, J. Burke, T. Refaei, L. Wang, B. Zhang, and L. Zhang, "A brief introduction to named data networking", in *MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM)*, IEEE, 2018, pp. 1–6.

[37] J. Shi, *What is a "face" in named data networking?*, May 2022. [Online]. Available: https://yoursunny.com/t/2021/NDN-face/.

[38] C. Yi, J. Abraham, A. Afanasyev, L. Wang, B. Zhang, and L. Zhang, "On the role of routing in named data networking", in *Proceedings of the 1st ACM Conference on Information-Centric Networking*, ser. ACM-ICN '14, Paris, France: Association for Computing Machinery, 2014, pp. 27–36, ISBN: 9781450332064. DOI: 10.1145/2660129.2660140. [Online]. Available: https://doi.org/10.1145/2660129.2660140.

[39] C. Yi, A. Afanasyev, I. Moiseenko, L. Wang, B. Zhang, and L. Zhang, "A case for stateful forwarding plane", *Computer Communications*, vol. 36, no. 7, pp. 779–791, 2013.

[40] A. M. Hoque, S. O. Amin, A. Alyyan, B. Zhang, L. Zhang, and L. Wang, "Nlsr: Named-data link state routing protocol", in *Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking*, 2013, pp. 15–20.

[41] T. Clausen, P. Jacquet, C. Adjih, *et al.*, *Optimized Link State Routing Protocol (OLSR)*, Network Working Group, 2003. [Online]. Available: https://hal.inria.fr/inria-00471712.

[42] V. Lehman, A. Gawande, B. Zhang, *et al.*, "An experimental investigation of hyperbolic routing with a smart forwarding plane in ndn", in *2016 IEEE/ACM 24th International Symposium on Quality of Service (IWQoS)*, IEEE, 2016, pp. 1–10.

[43] C. Ghali, G. Tsudik, E. Uzun, and C. A. Wood, "Closing the floodgate with stateless content-centric networking", in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, IEEE, 2017, pp. 1–10.

[44] M. Varvello, D. Perino, and L. Linguaglossa, "On the design and implementation of a wire-speed pending interest table", in *2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2013, pp. 369–374. DOI: 10.1109/INFCOMW.2013.6970719.

[45] J. Shi, D. Pesavento, and L. Benmohamed, "Ndn-dpdk: Ndn forwarding at 100 gbps on commodity hardware", in *Proceedings of the 7th ACM Conference on Information-Centric Networking*, ser. ICN '20, Virtual Event, Canada: Association for Computing Machinery, 2020, pp. 30–40, ISBN: 9781450380409. DOI: 10.1145/3405656.3418715. [Online]. Available: https://doi.org/10.1145/3405656.3418715.

[46] A. Afanasyev, J. Shi, B. Zhang, *et al.*, "Nfd developer's guide (revision 11)", *Dept. Comput. Sci., Univ. California, Los Angeles, Los Angeles, CA, USA, NDN, Technical Report NDN-0021, Revision 11, August 2021*, 2021.

[47] Z. Zhang, Y. Yu, H. Zhang, *et al.*, "An overview of security support in named data networking", *IEEE Communications Magazine*, vol. 56, no. 11, pp. 62–68, 2018.

[48] M. Dehghan, B. Jiang, A. Dabirmoghaddam, and D. Towsley, "On the analysis of caches with pending interest tables", in *Proceedings of the 2nd ACM Conference on Information-Centric Networking*, 2015, pp. 69–78.

[49] A. Dabirmoghaddam, M. Dehghan, and J. Garcia-Luna-Aceves, "Characterizing interest aggregation in content-centric networks", in *2016 IFIP Networking Conference (IFIP Networking) and Workshops*, IEEE, 2016, pp. 449–457.

[50] S. Burleigh, A. Hooke, L. Torgerson, *et al.*, "Delay-tolerant networking: An approach to interplanetary internet", *IEEE Communications Magazine*, vol. 41, no. 6, pp. 128–136, 2003. DOI: 10.1109/MCOM.2003.1204759.

[51] R. C. Shah, S. Roy, S. Jain, and W. Brunette, "Data mules: Modeling and analysis of a three-tier architecture for sparse sensor networks", *Ad Hoc Networks*, vol. 1, no. 2-3, pp. 215–233, 2003.

[52] W. Zhao, M. Ammar, and E. Zegura, "A message ferrying approach for data delivery in sparse mobile ad hoc networks", in *Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, 2004, pp. 187–198.

[53] A. Pentland, R. Fletcher, and A. Hasson, "Daknet: Rethinking connectivity in developing nations", *Computer*, vol. 37, no. 1, pp. 78–83, 2004.

[54] B. Sipos, M. Demmer, J. Ott, and P. Simon, "Delay-tolerant networking tcp convergence-layer protocol version 4, rfc 9174", Tech. Rep., 2022. DOI: DOI10.17487/RFC9174. [Online]. Available: https://www.rfc-editor.org/info/rfc9174.

[55] R. Ferreira, W. Moreira, P. Mendes, M. Gerla, and E. Cerqueira, "Improving the delivery rate of digital inclusion applications for amazon riverside communities by using an integrated bluetooth dtn architecture", *arXiv preprint arXiv:1405.7084*, 2014.

[56] J. Ronan, K. Walsh, and D. Long, "Evaluation of a dtn convergence layer for the ax. 25 network protocol", in *Proceedings of the Second International Workshop on Mobile Opportunistic Networking*, 2010, pp. 72–78.

[57] L. Zhang, D. Estrin, J. Burke, *et al.*, "Named data networking (ndn) project", *Relatório Técnico NDN-0001, Xerox Palo Alto Research Center-PARC*, vol. 157, p. 158, 2010.

[58] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan, "Building efficient wireless sensor networks with low-level naming", in *Proceedings of the eighteenth ACM symposium on Operating systems principles*, 2001, pp. 146–159.

[59] L. Clare, S. Burleigh, and K. Scott, "Endpoint naming for space delay/disruption tolerant networking", in *2010 IEEE Aerospace Conference*, IEEE, 2010, pp. 1–10.

[60] S. Trifunovic, S. T. Kouyoumdjieva, B. Distl, L. Pajevic, G. Karlsson, and B. Plattner, "A decade of research in opportunistic networks: Challenges, relevance, and future directions", *IEEE Communications Magazine*, vol. 55, no. 1, pp. 168–173, 2017.

[61] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and wait: An efficient routing scheme for intermittently connected mobile networks", in *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, 2005, pp. 252–259.

[62] S. Burleigh, K. Fall, and E. J. Birrane III, "Bundle protocol version 7, rfc 9171", Tech. Rep., 2022. DOI: 10.17487/RFC917. [Online]. Available: https://www.rfc-editor.org/info/rfc9171.

[63] K. Scott and S. Burleigh, "Bundle protocol specification", 2007.

[64] K. Fall, W. Hong, and S. Madden, "Custody transfer for reliable delivery in delay tolerant networks", *IRB-TR-03-030, July*, vol. 198, 2003.

[65] S. C. Burleigh, "Bundle-in-Bundle Encapsulation", Internet Engineering Task Force, Internet-Draft draft-ietf-dtn-bibect-03, Feb. 2020, Work in Progress, 14 pp. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-dtn-bibect-03.

[66] G. Araniti, N. Bezirgiannidis, E. Birrane, *et al.*, "Contact graph routing in dtn space networks: Overview, enhancements and performance", *IEEE Communications Magazine*, vol. 53, no. 3, pp. 38–46, 2015.

[67] S. Jain, K. Fall, and R. Patra, "Routing in a delay tolerant network", *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 4, pp. 145–158, Aug. 2004, ISSN: 0146-4833. DOI: 10.1145/1030194.1015484. [Online]. Available: https://doi.org/10.1145/1030194.1015484.

[68] J. A. Fraire, O. De Jonckere, and S. C. Burleigh, "Routing in the space internet: A contact graph routing tutorial", *Journal of Network and Computer Applications*, vol. 174, p. 102 884, 2021.

[69] S. Burleigh, C. Caini, J. J. Messina, and M. Rodolfi, "Toward a unified routing framework for delay-tolerant networking", in *2016 IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE)*, IEEE, 2016, pp. 82–86.

[70] A. Vahdat, D. Becker, *et al.*, *Epidemic routing for partially connected ad hoc networks*, 2000.

[71] A. Lindgren, A. Doria, and O. Schelén, "Probabilistic routing in intermittently connected networks", *ACM SIGMOBILE mobile computing and communications review*, vol. 7, no. 3, pp. 19–20, 2003.

[72] J. Burgess, B. Gallagher, D. D. Jensen, B. N. Levine, *et al.*, "Maxprop: Routing for vehicle-based disruption-tolerant networks.", in *Infocom*, Barcelona, Spain, vol. 6, 2006.

[73] P. Hui, A. Chaintreau, J. Scott, R. Gass, J. Crowcroft, and C. Diot, "Pocket switched networks and human mobility in conference environments", in *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, 2005, pp. 244–251.

[74] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott, "Impact of human mobility on opportunistic forwarding algorithms", *IEEE Transactions on Mobile Computing*, vol. 6, no. 6, pp. 606–620, 2007. DOI: 10.1109/TMC.2007.1060.

[75] P. Hui, J. Crowcroft, and E. Yoneki, "Bubble rap: Social-based forwarding in delay-tolerant networks", *IEEE transactions on mobile computing*, vol. 10, no. 11, pp. 1576–1589, 2010.

[76] W. Moreira, P. Mendes, and S. Sargento, "Opportunistic routing based on daily routines", in *2012 IEEE international symposium on a world of wireless, mobile and multimedia networks (WoWMoM)*, IEEE, 2012, pp. 1–6.

[77] I. Leontiadis and C. Mascolo, "Geopps: Geographical opportunistic routing for vehicular networks", in *2007 IEEE international symposium on a world of wireless, mobile and multimedia networks*, Ieee, 2007, pp. 1–6.

[78] Y. Cao, O. Kaiwartya, N. Aslam, *et al.*, "A trajectory-driven opportunistic routing protocol for vcps", *IEEE Transactions on Aerospace and Electronic Systems*, vol. 54, no. 6, pp. 2628–2642, 2018.

[79] Y. He, X. Tang, R. Zhang, X. Du, D. Zhou, and M. Guizani, "A course-aware opportunistic routing protocol for fanets", *IEEE Access*, vol. 7, pp. 144 303–144 312, 2019.

[80] M. A. Rahman, Y. Lee, and I. Koo, "Eecor: An energy-efficient cooperative opportunistic routing protocol for underwater acoustic sensor networks", *IEEE Access*, vol. 5, pp. 14 119–14 132, 2017.

[81] Q. Sang, H. Wu, L. Xing, H. Ma, and P. Xie, "An energy-efficient opportunistic routing protocol based on trajectory prediction for fanets", *IEEE Access*, vol. 8, pp. 192 009–192 020, 2020.

[82] D. K. Sharma, S. K. Dhurandher, I. Woungang, R. K. Srivastava, A. Mohananey, and J. J. P. C. Rodrigues, "A machine learning-based protocol for efficient routing in opportunistic networks", *IEEE Systems Journal*, vol. 12, no. 3, pp. 2207–2213, 2018. DOI: 10.1109/JSYST.2016.2630923.

[83] D. K. Sharma, S. K. Dhurandher, D. Agarwal, and K. Arora, "Krop: K-means clustering based routing protocol for opportunistic networks", *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 4, pp. 1289–1306, 2019.

[84] J. Ott and M. J. Pitkanen, "Dtn-based content storage and retrieval", in *2007 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, 2007, pp. 1–7. DOI: 10.1109/WOWMOM.2007.4351691.

[85] M. J. Pitkänen and J. Ott, "Redundancy and distributed caching in mobile dtns", in *Proceedings of 2nd ACM/IEEE International Workshop on Mobility in the Evolving Internet Architecture*, ser. MobiArch '07, Kyoto, Japan: Association for Computing Machinery, 2007, ISBN: 9781595937841. DOI: 10.1145/1366919.1366930. [Online]. Available: https://doi.org/10.1145/1366919.1366930.

[86] M. Chuah and P. Yang, "Performance evaluation of content-based information retrieval schemes for dtns", in *MILCOM 2007-IEEE Military Communications Conference*, IEEE, 2007, pp. 1–7.

[87] J. Greifenberg and D. Kutscher, "Efficient publish/subscribe-based multicast for opportunistic networking with self-organized resource utilization", in *22nd International Conference on Advanced Information Networking and Applications-Workshops (aina workshops 2008)*, IEEE, 2008, pp. 1708–1714.

[88] G. Sollazzo, M. Musolesi, and C. Mascolo, "Taco-dtn: A time-aware content-based dissemination system for delay tolerant networks", in *Proceedings of the 1st international MobiSys workshop on Mobile opportunistic networking*, 2007, pp. 83–90.

[89] J. Ott, E. Hyytiä, P. Lassila, T. Vaegs, and J. Kangasharju, "Floating content: Information sharing in urban areas", in *2011 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, IEEE, 2011, pp. 136–146.

[90] W. Gao, G. Cao, A. Iyengar, and M. Srivatsa, "Cooperative caching for efficient data access in disruption tolerant networks", *IEEE Transactions on Mobile Computing*, vol. 13, no. 3, pp. 611–625, 2013.

[91] R. Krishnan, P. Basu, J. M. Mikkelson, *et al.*, "The spindle disruption-tolerant networking system", in *MILCOM 2007-IEEE Military Communications Conference*, IEEE, 2007, pp. 1–7.

[92] S. Farrell, A. Lynch, D. Kutscher, and A. Lindgren, "Bundle protocol query extension block", *draft-farrell-dtnrg-bpq-0 1*, 2012.

[93] J. Scott, J. Crowcroft, P. Hui, and C. Diot, "Haggle: A networking architecture designed around mobile users", in *WONS 2006: Third Annual Conference on Wireless On-demand Network Systems and Services*, 2006, pp. 78–86.

[94] J. Su, J. Scott, P. Hui, *et al.*, "Haggle: Seamless networking for mobile applications", in *International Conference on Ubiquitous Computing*, Springer, 2007, pp. 391–408.

[95] G. Tyson, J. Bigham, and E. Bodanese, "Towards an information-centric delay-tolerant network", in *2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, IEEE, 2013, pp. 387–392.

[96] B. Kauffmann, J. Peltier, *et al.*, "Db 3 (d-3.3) final netinf architecture. deliverable d-3.3, version 1.1, sail eu fp7 project 257448, jan. 2013", FP7-ICT-2009-5-257448/DB 3, Tech. Rep.

[97] S. C. Nelson, G. Bhanage, and D. Raychaudhuri, "Gstar: Generalized storage-aware routing for mobilityfirst in the future mobile internet", in *Proceedings of the sixth international workshop on MobiArch*, 2011, pp. 19–24.

[98] D. Trossen, A. Sathiaseelan, and J. Ott, "Towards an information centric network architecture for universal internet access", *ACM SIGCOMM Computer Communication Review*, vol. 46, no. 1, pp. 44–49, 2016.

[99] H. Du, D. Trossen, and C. Theodorou, "D2. 3: Final component and interface specifications", 2015.

[100] T. Kärkkäinen, M. Pitkänen, P. Houghton, and J. Ott, "Scampi application platform", in *Proceedings of the seventh ACM international workshop on Challenged networks*, 2012, pp. 83–86.

[101] C. Anastasiades, A. Uruqi, and T. Braun, "Content discovery in opportunistic content-centric networks", in *37th Annual IEEE Conference on Local Computer Networks-Workshops*, IEEE, 2012, pp. 1044–1052.

[102] C. Anastasiades, T. Schmid, J. Weber, and T. Braun, "Opportunistic content-centric data transmission during short network contacts", *2014 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 2516–2521, 2014.

[103] C. Anastasiades, W. E. M. El Alami, and T. Braun, "Agent-based content retrieval for opportunistic content-centric networks", in *International Conference on Wired/Wireless Internet Communications*, Springer, 2014, pp. 175–188.

[104] C. Anastasiades, T. Schmid, J. Weber, and T. Braun, "Information-centric content retrieval for delay-tolerant networks", *Computer Networks*, vol. 107, pp. 194–207, 2016.

[105] B. Batista and P. Mendes, "Icon-an information centric architecture for opportunistic networks", *2nd IEEE NOMEN*, 2013.

[106] F. Neves dos Santos, B. Ertl, C. Barakat, T. Spyropoulos, and T. Turletti, "Cedo: Content-centric dissemination algorithm for delay-tolerant networks", in *Proceedings of the 16th ACM international conference on Modeling, analysis & simulation of wireless and mobile systems*, 2013, pp. 377–386.

[107] E. Monticelli, B. M. Schubert, M. Arumaithurai, X. Fu, and K. Ramakrishnan, "An information centric approach for communications in disaster situations", in *2014 IEEE 20th International Workshop on Local & Metropolitan Area Networks (LANMAN)*, IEEE, 2014, pp. 1–6.

[108] T. Yagyu and S. Maeda, "Demo overview: Reliable contents retrieval in fragmented icns for disaster scenario", in *Proceedings of the 1st ACM Conference on Information-Centric Networking*, 2014, pp. 193–194.

[109] P. Duarte, J. Macedo, A. D. Costa, M. J. Nicolau, and A. Santos, "A probabilistic interest forwarding protocol for named data delay tolerant networks", in *International conference on ad hoc networks*, Springer, 2015, pp. 94–107.

[110] Y. Lu, X. Li, Y.-T. Yu, and M. Gerla, "Information-centric delay-tolerant mobile ad-hoc networks", in *2014 IEEE conference on computer communications workshops (INFOCOM WKSHPS)*, IEEE, 2014, pp. 428–433.

[111] E. Borgia, R. Bruno, and A. Passarella, "Mobccn: A ccn-compliant protocol for data collection with opportunistic contacts in iot environments", in *Proceedings of the Eleventh ACM Workshop on Challenged Networks*, 2016, pp. 63–68.

[112] E. Borgia, R. Bruno, and A. Passarella, "Making opportunistic networks in iot environments ccn-ready: A performance evaluation of the mobccn protocol", *Computer Communications*, vol. 123, pp. 81–96, 2018.

[113] C. D. T. De Souza, D. L. Ferreira, C. A. V. Campos, A. C. D. O. Júnior, K. V. Cardoso, and W. Moreira, "Employing social cooperation to improve data discovery and retrieval in content-centric delay-tolerant networks", *IEEE Access*, vol. 7, pp. 137 930–137 944, 2019.

[114] J. Seedorf, A. Tagami, M. Arumaithurai, *et al.*, "The benefit of information centric networking for enabling communications in disaster scenarios", in *2015 IEEE Globecom Workshops (GC Wkshps)*, IEEE, 2015, pp. 1–7.

[115] J. Seedorf, D. Kutscher, and B. S. Gill, "Decentralised interest counter aggregation for icn in disaster scenarios", in *2016 IEEE Globecom Workshops (GC Wkshps)*, IEEE, 2016, pp. 1–6.

[116] T. Li, Z. Kong, and L. Zhang, "Supporting delay tolerant networking: A comparative study of epidemic routing and ndn", in *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*, IEEE, 2020, pp. 1–6.

[117] A. Demers, D. Greene, C. Hauser, *et al.*, "Epidemic algorithms for replicated database maintenance", in *Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*, 1987, pp. 1–12.

[118] T. Li, Z. Kong, S. Mastorakis, and L. Zhang, "Distributed dataset synchronization in disruptive networks", in *2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, IEEE, 2019, pp. 428–437.

[119] K. Russell and R. Simon, "Evaluation of a geo-region based architecture for information centric disruption tolerant networks", in *2019 International Conference on Computing, Networking and Communications (ICNC)*, 2019, pp. 911–917.

[120] K. Russell and R. Simon, "Efficient data advertisement in information centric disruption tolerant networks", in *2020 International Conference on Computing, Networking and Communications (ICNC)*, IEEE, 2020, pp. 112–117.

[121] N. van Adrichem, B. Wissingh, D. Ravesteijn, and L. D'Acunto, "Data muling in icn", in *Proceedings of the 4th ACM Conference on Information-Centric Networking*, 2017, pp. 206–207.

[122] S. Dynerowicz and P. Mendes, "Named-data networking in opportunistic network", in *Proceedings of the 4th ACM Conference on Information-Centric Networking*, 2017, pp. 220–221.

[123] C. Ghali, G. Tsudik, and E. Uzun, "Network-layer trust in named-data networking", *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 5, pp. 12–19, 2014.

[124] Y. Yu, Y. Li, X. Du, R. Chen, and B. Yang, "Content protection in named data networking: Challenges and potential solutions", *IEEE Communications Magazine*, vol. 56, no. 11, pp. 82–87, 2018.

[125] S. K. Ramani, R. Tourani, G. Torres, S. Misra, and A. Afanasyev, "Ndn-abs: Attribute-based signature scheme for named data networking", in *Proceedings of the 6th ACM Conference on Information-Centric Networking*, 2019, pp. 123–133.

[126] G. Bianchi, A. Detti, A. Caponi, and N. Blefari Melazzi, "Check before storing: What is the performance price of content integrity verification in lru caching?", *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 3, pp. 59–67, 2013.

[127] P. Gasti, G. Tsudik, E. Uzun, and L. Zhang, "Dos and ddos in named data networking", in *2013 22nd International Conference on Computer Communication and Networks (ICCCN)*, IEEE, 2013, pp. 1–7.

[128] Y. Wang, Z. Qi, K. Lei, B. Liu, and C. Tian, "Preventing" bad" content dispersal in named data networking", in *Proceedings of the ACM Turing 50th Celebration Conference-China*, 2017, pp. 1–8.

[129] W. Shang, J. Thompson, M. Cherkaoui, J. Burkey, and L. Zhang, "Ndn. js: A javascript client library for named data networking", in *2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, IEEE, 2013, pp. 399–404.

[130] T. Chen, K. Lei, and K. Xu, "An encryption and probability based access control model for named data networking", in *2014 IEEE 33rd International Performance Computing and Communications Conference (IPCCC)*, IEEE, 2014, pp. 1–8.

[131] R. Tourani, R. Stubbs, and S. Misra, "Tactic: Tag-based access control framework for the information-centric wireless edge networks", in *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, IEEE, 2018, pp. 456–466.

[132] A. Afanasyev, P. Mahadevan, I. Moiseenko, E. Uzun, and L. Zhang, "Interest flooding attack and countermeasures in named data networking", in *2013 IFIP Networking Conference*, IEEE, 2013, pp. 1–9.

[133] P. Gasti and G. Tsudik, "Content-centric and named-data networking security: The good, the bad and the rest", in *2018 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*, IEEE, 2018, pp. 1–6.

[134] C. Ghali, G. Tsudik, E. Uzun, and C. A. Wood, "Closing the floodgate with stateless content-centric networking", in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, IEEE, 2017, pp. 1–10.

[135] J. Liang, R. Kumar, Y. Xi, and K. W. Ross, "Pollution in p2p file sharing systems", in *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, IEEE, vol. 2, 2005, pp. 1174–1185.

[136] G. Montassier, T. Cholez, G. Doyen, R. Khatoun, I. Chrisment, and O. Festor, "Content pollution quantification in large p2p networks: A measurement study on kad", in *2011 IEEE International Conference on Peer-to-Peer Computing*, IEEE, 2011, pp. 30–33.

[137] C. Ghali, G. Tsudik, E. Uzun, *et al.*, "Needle in a haystack: Mitigating content poisoning in named-data networking", in *Proceedings of NDSS workshop on security of emerging networking technologies (SENT)*, 2014, pp. 1–10.

[138] ISO, "Iso/iec 23009-1:2019: Information technology–dynamic adaptive streaming over http (dash)–part 1: Media presentation description and segment formats", *Geneva, Switzerland: International Organization for Standardization*, 2019.

[139] C. Westphal, S. Lederer, C. Mueller, *et al.*, *Adaptive Video Streaming over Information-Centric Networking (ICN)*, RFC 7933, Aug. 2016. DOI: 10.17487/RFC7933. [Online]. Available: https://www.rfc-editor.org/info/rfc7933.

[140] T. Stockhammer, "Dynamic adaptive streaming over http– standards and design principles", in *Proceedings of the second annual ACM conference on Multimedia systems*, 2011, pp. 133–144.

[141] S. Lederer, C. Mueller, B. Rainer, C. Timmerer, and H. Hellwagner, "An experimental analysis of dynamic adaptive streaming over http in content centric networks", in *2013 IEEE International Conference on Multimedia and Expo (ICME)*, IEEE, 2013, pp. 1–6.

[142] Y. Liu, J. Geurts, J.-C. Point, *et al.*, "Dynamic adaptive streaming over ccn: A caching and overhead analysis", in *2013 IEEE international conference on communications (ICC)*, IEEE, 2013, pp. 3629–3633.

[143] S. Lederer, C. Mueller, B. Rainer, C. Timmerer, and H. Hellwagner, "Adaptive streaming over content centric networks in mobile networks using multiple links", in *2013 IEEE International Conference on Communications Workshops (ICC)*, IEEE, 2013, pp. 677–681.

[144] S. Lederer, C. Mueller, C. Timmerer, and H. Hellwagner, "Adaptive multimedia streaming in information-centric networks", *IEEE Network*, vol. 28, no. 6, pp. 91–96, 2014.

[145] M. Conti, R. Droms, M. Hassan, and C. Lal, "Fair-rtt-das: A robust and efficient dynamic adaptive streaming over icn", *Computer Communications*, vol. 129, pp. 209–225, 2018.

[146] M. Conti, R. Droms, M. Hassan, and S. Valle, "Qoe degradation attack in dynamic adaptive streaming over icn", in *2018 IEEE 19th International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*, IEEE, 2018, pp. 1–9.

[147] M. Hassan, H. Salat, M. Conti, F. H. Fitzek, and T. Strule, "Comon-das: A framework for efficient and robust dynamic adaptive streaming over ndn", in *2019 IEEE Symposium on Computers and Communications (ISCC)*, IEEE, 2019, pp. 1–7.

[148] S. Ruohomaa and L. Kutvonen, "Trust management survey", in *International Conference on Trust Management*, Springer, 2005, pp. 77–92.

[149] V. Shmatikov and C. Talcott, "Reputation-based trust management", *Journal of Computer Security*, vol. 13, no. 1, pp. 167–190, 2005.

[150] P. Bonatti, C. Duma, D. Olmedilla, and N. Shahmehri, "An integration of reputation-based and policy-based trust management", *networks*, vol. 2, no. 14, p. 10, 2007.

[151] E. Koutrouli and A. Tsalgatidou, "Reputation-based trust systems for p2p applications: Design issues and comparison framework", in *International conference on trust, privacy and security in digital business*, Springer, 2006, pp. 152–161.

[152] J. Sabater and C. Sierra, "Review on computational trust and reputation models", *Artificial intelligence review*, vol. 24, no. 1, pp. 33–60, 2005.

[153] L. Xiong and L. Liu, "Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities", *IEEE transactions on Knowledge and Data Engineering*, vol. 16, no. 7, pp. 843–857, 2004.

[154] J.-H. Cho, A. Swami, and R. Chen, "A survey on trust management for mobile ad hoc networks", *IEEE communications surveys & tutorials*, vol. 13, no. 4, pp. 562–583, 2010.

[155] T. Zahariadis, H. C. Leligou, P. Trakadas, and S. Voliotis, "Trust management in wireless sensor networks", *European Transactions on Telecommunications*, vol. 21, no. 4, pp. 386–395, 2010.

[156] S. Vavilis, M. Petković, and N. Zannone, "A reference model for reputation systems", *Decision Support Systems*, vol. 61, pp. 147–154, 2014.

[157] F. Hendrikx, K. Bubendorfer, and R. Chard, "Reputation systems: A survey and taxonomy", *Journal of Parallel and Distributed Computing*, vol. 75, pp. 184–197, 2015.

[158] Q. Zhang, T. Yu, and K. Irwin, "A classification scheme for trust functions in reputation-based trust management.", in *Trust@ ISWC*, Citeseer, 2004.

[159] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system", *Decentralized Business Review*, p. 21 260, 2008.

[160] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger", *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[161] H. A. Kalodner, M. Carlsten, P. Ellenbogen, J. Bonneau, and A. Narayanan, "An empirical study of namecoin and lessons for decentralized namespace design.", in *WEIS*, Citeseer, 2015.

[162] C. Fromknecht, D. Velicanu, and S. Yakoubov, "Certcoin: A namecoin based decentralized authentication system 6.857 class project", *Unpublished class project*, 2014.

[163] M. Ali, J. Nelson, R. Shea, and M. J. Freedman, "Blockstack: A global naming and storage system secured by blockchains", in *2016 {USENIX} annual technical conference ({USENIX}{ATC} 16)*, 2016, pp. 181–194.

[164] E. Damiani, D. C. di Vimercati, S. Paraboschi, P. Samarati, and F. Violante, "A reputation-based approach for choosing reliable resources in peer-to-peer networks", in *Proceedings of the 9th ACM conference on Computer and communications security*, 2002, pp. 207–216.

[165] K. Walsh and E. G. Sirer, "Experience with an object reputation system for peer-to-peer filesharing.", in *NSDI*, vol. 6, 2006, pp. 1–1.

[166] C. Costa and J. Almeida, "Reputation systems for fighting pollution in peer-to-peer file sharing systems", in *Seventh IEEE international conference on peer-to-peer computing (P2P 2007)*, IEEE, 2007, pp. 53–60.

[167] D. Fraga, Z. Bankovic, and J. M. Moya, "A taxonomy of trust and reputation system attacks", in *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*, IEEE, 2012, pp. 41–50.

[168] E. Bellini, Y. Iraqi, and E. Damiani, "Blockchain-based distributed trust and reputation management systems: A survey", *IEEE Access*, vol. 8, pp. 21 127–21 151, 2020.

[169] "Golem whitepaper", Tech. Rep., Nov. 2016. [Online]. Available: https://golem.network/doc/Golemwhitepaper.pdf.

[170] B. Guidi, A. Michienzi, and L. Ricci, "Steem blockchain: Mining the inner structure of the graph", *IEEE Access*, vol. 8, pp. 210 251–210 266, 2020.

[171] M. Król, A. Sonnino, M. Al-Bassam, A. G. Tasiopoulos, E. Rivière, and I. Psaras, "Proof-of-prestige: A useful work reward system for unverifiable tasks", *ACM Transactions on Internet Technology (TOIT)*, vol. 21, no. 2, pp. 1–27, 2021.

[172] J. Kwon and E. Buchman, "Cosmos whitepaper", *A Netw. Distrib. Ledgers*, 2019.

[173] R. L. Rivest and B. Lampson, "Sdsi-a simple distributed security infrastructure", Crypto, 1996.

[174] Z. Zhang, A. Afanasyev, and L. Zhang, "Ndncert: Universal usable trust management for ndn", in *Proceedings of the 4th ACM Conference on Information-Centric Networking*, 2017, pp. 178–179.

[175] Y. Yu, A. Afanasyev, D. Clark, K. Claffy, V. Jacobson, and L. Zhang, "Schematizing trust in named data networking", in *proceedings of the 2nd ACM Conference on Information-Centric Networking*, 2015, pp. 177–186.

[176] Z. Rezaeifar, J. Wang, and H. Oh, "A trust-based method for mitigating cache poisoning in name data networking", *Journal of Network and Computer Applications*, vol. 104, pp. 117–132, 2018.

[177] D. Wu, Z. Xu, B. Chen, and Y. Zhang, "What if routers are malicious? mitigating content poisoning attack in ndn", in *2016 IEEE Trustcom/BigDataSE/ISPA*, IEEE, 2016, pp. 481–488.

[178] N. Ntuli and S. Han, "Detecting router cache snooping in named data networking", in *2012 International Conference on ICT Convergence (ICTC)*, IEEE, 2012, pp. 714–718.

[179] S. Umeda, T. Kamimoto, Y. Ohata, and H. Shigeno, "Interest flow control method based on user reputation and content name prefixes in named data networking", in *2015 IEEE Trustcom/BigDataSE/ISPA*, IEEE, vol. 1, 2015, pp. 710–717.

[180] C.-A. Sarros, S. Diamantopoulos, S. Rene, *et al.*, "Connecting the edges: A universal, mobile-centric, and opportunistic communications architecture", *IEEE Communications Magazine*, vol. 56, no. 2, pp. 136–143, 2018.

[181] O. Ascigil, V. Sourlas, I. Psaras, and G. Pavlou, "Opportunistic off-path content discovery in information-centric networks", in *2016 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*, IEEE, 2016, pp. 1–7.

[182] I. Psaras, S. René, K. Katsaro, *et al.*, "Keyword-based mobile application sharing", in *Proceedings of the Workshop on Mobility in the Evolving Internet Architecture*, 2016, pp. 1–6.

[183] O. Aponte and P. Mendes, "Now@ content sharing application over ndn", in *Proceedings of the 4th ACM Conference on Information-Centric Networking*, 2017, pp. 196–197.

[184] P. Mendes, R. C. Sofia, J. Soares, V. Tsaoussidis, S. Diamantopoulos, and C.-A. Sarros, "Information-centric routing for opportunistic wireless networks", in *Proceedings of the 5th ACM Conference on Information-Centric Networking*, 2018, pp. 194–195.

[185] A. Lertsinsrubtavee, A. Ali, C. Molina-Jimenez, A. Sathiaseelan, and J. Crowcroft, "Picasso: A lightweight edge computing platform", in *2017 IEEE 6th International Conference on Cloud Networking (CloudNet)*, 2017, pp. 1–7. DOI: 10.1109/CloudNet.2017.8071529.

[186] M. Selimi, A. Lertsinsrubtavee, A. Sathiaseelan, L. Cerd-Alabern, and L. Navarro, "Picasso: Enabling information-centric multi-tenancy at the edge of community mesh networks", *Computer Networks*, vol. 164, p. 106 897, 2019, ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2019.106897. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128618312787.

[187] C.-A. Sarros, A. Lertsinsrubtavee, C. Molina-Jimenez, *et al.*, "Icn-based edge service deployment in challenged networks", in *Proceedings of the 4th ACM Conference on Information-Centric Networking*, 2017, pp. 210–211.

[188] C.-A. Sarros, V. Demiroglou, and V. Tsaoussidis, "Intermittently-connected iot devices: Experiments with an ndn-dtn architecture", *2021 18th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, 2021.

[189] V. Demiroglou, C.-A. Sarros, and V. Tsaoussidis, "Nod: A content retrieval scheme for intermittently-connected iot networks", *Ad Hoc Networks*, vol. 130, p. 102 825, 2022.

[190] K. Pentikousis, B. Ohlman, D. Corujo, *et al.*, *Information-Centric Networking: Baseline Scenarios*, RFC 7476, Mar. 2015. DOI: 10.17487/RFC7476. [Online]. Available: https://www.rfc-editor.org/info/rfc7476.

[191] J. Seedorf, M. Arumaithurai, A. Tagami, K. Ramakrishnan, and N. Blefari-Melazzi, *Research Directions for Using Information-Centric Networking (ICN) in Disaster Scenarios*, RFC 8884, Oct. 2020. DOI: 10.17487/RFC8884. [Online]. Available: https://rfc-editor.org/rfc/rfc8884.txt.

[192] A. Sathiaseelan, L. Wang, A. Aucinas, G. Tyson, and J. Crowcroft, "Scandex: Service centric networking for challenged decentralised networks", in *Proceedings of the 2015 Workshop on Do-it-yourself Networking: an Interdisciplinary Approach*, 2015, pp. 15–20.

[193] C. Natsis, C.-A. Sarros, and V. Tsaoussidis, "The impact of chunk size on named data networking performance", in *2020 3rd International Conference on Hot Information-Centric Networking (HotICN)*, IEEE, 2020, pp. 108–113.

[194] M. De Sanctis, E. Cianca, G. Araniti, I. Bisio, and R. Prasad, "Satellite communications supporting internet of remote things", *IEEE Internet of Things Journal*, vol. 3, no. 1, pp. 113–123, 2016. DOI: 10.1109/JIOT.2015.2487046.

[195] L. Baumgärtner, A. Penning, P. Lampe, B. Richerzhagen, R. Steinmetz, and B. Freisleben, "Environmental monitoring using low-cost hardware and infrastructureless wireless communication", in *2018 IEEE Global Humanitarian Technology Conference (GHTC)*, 2018, pp. 1–8. DOI: 10.1109/GHTC.2018.8601883.

[196] N. Ahmed, D. De, and I. Hussain, "Internet of things (iot) for smart precision agriculture and farming in rural areas", *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4890–4899, 2018. DOI: 10.1109/JIOT.2018.2879579.

[197] E. M. Migabo, K. D. Djouani, and A. M. Kurien, "The narrowband internet of things (nb-iot) resources management performance state of art, challenges, and opportunities", *IEEE Access*, vol. 8, pp. 97 658–97 675, 2020. DOI: 10.1109/ACCESS.2020.2995938.

[198] M. Zhang and X. Li, "Drone-enabled internet-of-things relay for environmental monitoring in remote areas without public networks", *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7648–7662, 2020. DOI: 10.1109/JIOT.2020.2988249.

[199] D. Palma and R. Birkeland, "Enabling the internet of arctic things with freely-drifting small-satellite swarms", *IEEE Access*, vol. 6, pp. 71 435–71 443, 2018. DOI: 10.1109/ACCESS.2018.2881088.

[200] J. Samain, G. Carofiglio, L. Muscariello, *et al.*, "Dynamic adaptive video streaming: Towards a systematic comparison of icn and tcp/ip", *IEEE Transactions on Multimedia*, vol. 19, no. 10, pp. 2166–2181, Oct. 2017, ISSN: 1520-9210. DOI: 10.1109/TMM.2017.2733340.

[201] Y. Liu, J. Geurts, J.-C. Point, *et al.*, "Dynamic adaptive streaming over ccn: A caching and overhead analysis", in *2013 IEEE international conference on communications (ICC)*, IEEE, 2013, pp. 3629–3633.

[202] M. Hassan, H. Salah, M. Conti, F. H. P. Fitzek, and T. Sturfe, " CoMon-DAS: A Framework for Efficient and Robust Dynamic Adaptive Streaming over NDN", *In proceedings of 24th IEEE Symposium on Computers and Communications (ISCC).*,

[203] D. Wang, T. Muller, Y. Liu, and J. Zhang, "Towards robust and effective trust management for security: A survey", in *2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications*, IEEE, 2014, pp. 511–518.

[204] I. A. Kapetanidou, M. Hassan, C.-A. Sarros, M. Conti, and V. Tsaoussidis, "Reputation-based trust: A robust mechanism for dynamic adaptive streaming over named data networking", in *2020 IEEE 14th International Conference on Big Data Science and Engineering (BigDataSE)*, IEEE, 2020, pp. 114–121.

[205] I. A. Kapetanidou, C.-A. Sarros, and V. Tsaoussidis, "Reputation-based trust approaches in named data networking", *Future Internet*, vol. 11, no. 11, p. 241, 2019.

[206] M. Król, A. Sonnino, M. Al-Bassam, A. Tasiopoulos, and I. Psaras, "Proof-of-prestige: A useful work reward system for unverifiable tasks", in *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, IEEE, 2019, pp. 293–301.

[207] S. Mastorakis, A. Afanasyev, Y. Yu, and L. Zhang, "Ntorrent: Peer-to-peer file sharing in named data networking", in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, IEEE, 2017, pp. 1–10.

[208] O. Ascigil, S. Reñé, M. Król, *et al.*, "Towards peer-to-peer content retrieval markets: Enhancing ipfs with icn", in *Proceedings of the 6th ACM Conference on Information-Centric Networking*, 2019, pp. 78–88.

[209] Q. T. Thai, N. Ko, S. H. Byun, and S.-M. Kim, "Design and implementation of ndn-based ethereum blockchain", *Journal of Network and Computer Applications*, vol. 200, p. 103 329, 2022, ISSN: 1084-8045. DOI: https://doi.org/10.1016/j.jnca.2021.103329. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804521003143.

[210] T. Jin, X. Zhang, Y. Liu, and K. Lei, "Blockndn: A bitcoin blockchain decentralized system over named data networking", in *2017 Ninth international conference on ubiquitous and future networks (ICUFN)*, IEEE, 2017, pp. 75–80.

[211] J. Guo, M. Wang, B. Chen, S. Yu, H. Zhang, and Y. Zhang, "Enabling blockchain applications over named data networking", in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, IEEE, 2019, pp. 1–6.

[212] I. Kapetanidou, S. Malagaris, and V. Tsaoussidis, "Avoiding notorious content sources: A content-poisoning attack mitigation approach", in *2022 IEEE Symposium on Computers and Communications (ISCC)*, IEEE, 2022.

[213] C.-A. Sarros, I. A. Kapetanidou, and V. Tsaoussidis, "Incentivising honest behaviour in p2p networks using blockchain-based reputation", in *2021 Eighth International Conference on Software Defined Systems (SDS)*, 2021, pp. 1–6. DOI: 10.1109/SDS54264.2021.9732162.

[214] S. Diamantopoulos *et al.*, "D3.2 umobile architecture report (2)", UMOBILE consortium, Tech. Rep., Aug. 2017. [Online]. Available: https://umobile-project.eu/deliverables.

[215] E. Baccelli, O. Hahm, M. Günes, M. Wählisch, and T. C. Schmidt, "Riot os: Towards an os for the internet of things", in *2013 IEEE conference on computer communications workshops (INFOCOM WKSHPS)*, IEEE, 2013, pp. 79–80.

[216] A. Afanasyev, I. Moiseenko, L. Zhang, *et al.*, "Ndnsim: Ndn simulator for ns-3", 2012.

[217] S. Mastorakis, A. Afanasyev, and L. Zhang, "On the evolution of ndnsim: An open-source simulator for ndn experimentation", *ACM SIGCOMM Computer Communication Review*, vol. 47, no. 3, pp. 19–33, 2017.

[218] S. Lederer, C. Mueller, C. Timmerer, and H. Hellwagner, "Adaptive multimedia streaming in information-centric networks", *IEEE Network*, vol. 28, no. 6, pp. 91–96, Nov. 2014, ISSN: 0890-8044. DOI: 10.1109/MNET.2014.6963810.

[219] A. Compagno, M. Conti, P. Gasti, L. V. Mancini, and G. Tsudik, "Violating consumer anonymity: Geo-locating nodes in named data networking", in *Applied Cryptography and Network Security: 13th International Conference, ACNS 2015, New York, NY, USA, June 2-5, 2015, Revised Selected Papers*, T. Malkin, V. Kolesnikov, A. B. Lewko, and M. Polychronakis, Eds. Cham: Springer International Publishing, 2015, pp. 243–262.

[220] G. Acs, M. Conti, P. Gasti, C. Ghali, G. Tsudik, and C. Wood, "Privacy-aware caching in information-centric networking", *IEEE Transactions on Dependable and Secure Computing*, vol. PP, no. 99, 2017.

[221] F. Zhang, W. He, X. Liu, and P. G. Bridges, "Inferring users' online activities through traffic analysis", in *Proceedings of the Fourth ACM Conference on Wireless Network Security*, ser. WiSec '11, Hamburg, Germany: ACM, 2011, pp. 59–70, ISBN: 978-1-4503-0692-8. DOI: 10.1145/1998412.1998425. [Online]. Available: http://doi.acm.org/10.1145/1998412.1998425.

[222] M. Liberatore and B. N. Levine, "Inferring the source of encrypted http connections", in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, ser. CCS '06, Alexandria, Virginia, USA: ACM, 2006, pp. 255–263, ISBN: 1-59593-518-5. DOI: 10.1145/1180405.1180437. [Online]. Available: http://doi.acm.org/10.1145/1180405.1180437.

[223] C. Kreuzberger, D. Posch, and H. Hellwagner, *Amust framework - adaptive multimedia streaming simulation framework for ns-3 and ndnsim*, 2016.

[224] C. Mueller, S. Lederer, J. Poecher, and C. Timmerer, "Demo paper: Libdash-an open source software library for the mpeg-dash standard", in *2013 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, IEEE, 2013, pp. 1–2.

[225] S. Lederer, C. Müller, and C. Timmerer, "Dynamic adaptive streaming over http dataset", in *Proceedings of the 3rd Multimedia Systems Conference*, ser. MMSys '12, Chapel Hill, North Carolina: ACM, 2012, pp. 89–94, ISBN: 978-1-4503-1131-1. DOI: 10.1145/2155555.2155570. [Online]. Available: http://doi.acm.org/10.1145/2155555.2155570.

[226] Y. Liu, S. Dey, F. Ulupinar, M. Luby, and Y. Mao, "Deriving and validating user experience model for dash video streaming", *IEEE Transactions on Broadcasting*, vol. 61, no. 4, pp. 651–665, 2015.

[227] D. Z. Rodriguez, R. L. Rosa, E. C. Alfaia, J. I. Abrahao, and G. Bressan, "Video quality metric for streaming service using dash standard", *IEEE Transactions on broadcasting*, vol. 62, no. 3, pp. 628–639, 2016.