ΔΗΜΟΚΡΙΤΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΡΑΚΗΣ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Ανάπτυξη Προσομοιώσεων και Αξιολόγηση Απόδοσης της NDN αρχιτεκτονικής μέσα από το εργαλείο mini-NDN

Αστέριος Παρλίτσης, 57166

Διπλωματική Εργασία

Επιβλέπων καθηγητής: Βασίλειος Τσαουσίδης

ΞΑΝΘΗ 2021

Ευχαριστίες

Θα ήθελα να εκφράσω τις ευχαριστίες μου στον επιβλέποντα καθηγητή της διπλωματικής μου εργασίας, κο Βασίλειο Τσαουσίδη, για την καθοδήγηση και υποστήριξή του στην πορεία των σπουδών μου και κυρίως στο τέλος κατά την εκπόνηση της εργασίας μου, η οποία έγινε κάτω από αντίξοες και απρόβλεπτες συνθήκες λόγω της πανδημίας. Δεν θα ήθελα επίσης να ξεχάσω τους υποψήφιους διδάκτορες, Αλέξανδρο Σάρρο και Ιωάννα Καπετανίδου, οι οποίοι με βοήθησαν και συνέβαλαν στην επιτυχή ολοκλήρωση της εργασίας μου. Τους ευχαριστώ θερμά.

Περίληψη

Στον σύγχρονο κόσμο όπου τεχνολογίες όπως το Internet of Things γνωρίζουν μεγάλη ανάπτυξη, προκύπτει η άμεση ανάγκη για μια εναλλακτική αρχιτεκτονική δικτύου πέρα από την IP/TCP. Η αλλαγή της βασικής λειτουργίας του Διαδικτύου από την απλή σύνδεση απομακρυσμένων κόμβων στην ανταλλαγή μεγάλου όγκου δεδομένων μεταξύ των χρηστών, οδήγησε στην ανάπτυξη νέων αρχιτεκτονικών όπως σε αυτή των ICN δικτύων που βασίζουν τη λειτουργία τους στην ονομασία πακέτων δεδομένων. Στην συγκεκριμένη εργασία θα συγκριθεί το μοντέλο NDN, που υπάγεται στα ICN δίκτυα, με την υπάρχουσα δομή του Internet, εξακριβώνοντας αυτά τα χαρακτηριστικά που το καθιστούν καταλληλότερο για την σύγχρονη content-oriented χρήση του. Ο στόχος είναι η ανάπτυξη κώδικα που προσομοιώνει την NDN λογική σε διαφορετικές τοπολογίες και σε διαφορετικά σενάρια με τη χρήση του εργαλείου mini-NDN. Τα αποτελέσματα της μοντελοποίησης επικεντρώνονται στα πλεονεκτήματα που προσφέρουν οι λειτουργίες της ενσωματωμένης ασφάλειας δικτύου στα πακέτα, της προσωρινής αποθήκευσης πακέτων ενδιάμεσα του δικτύου και του νέου πρωτοκόλλου δρομολόγησης NLSR. Παράλληλα, η συγκεκριμένη εργασία συμβάλει στην ανάπτυξη της βιβλιογραφίας του παραπάνω εργαλείου και παρέχει τυποποιημένες κλάσεις για την εξαγωγή αποτελεσμάτων.

Abstract

In today’s world where technologies such as the Internet of Things grow rapidly, there is an urgent need for an alternative network architecture beyond IP / TCP. The change in the basic function of the Internet from simply connecting remote nodes to exchanging large volumes of data between users, has led to the development of new architectures such as that of the ICN that uses the principle of naming the disseminating packets. In this paper we will compare the NDN model, which belongs to the ICN networks, with the existing structure of the Internet, identifying these features that make it more suitable for its modern content-oriented use. The goal is to develop code that runs the NDN model in different topologies and in different scenarios using the mini-NDN emulator. The results of the experimentation focus on the advantages offered by the functions of integrated network security in packets, the in-network caching and the NLSR routing protocol. At the same time, this work con-

tributes to the development of the bibliography of the above tool and provides standard classes for extracting data.

Πίνακας Περιεχομένων

Εισαγωγή

Με την εξάπλωση της χρήσης της τεχνολογίας σε ολοένα και μεγαλύτερο τμήμα του παγκόσμιου πληθυσμού, με την είσοδο περισσοτέρων συσκευών στο δίκτυο καθώς και λόγω της αύξησης της ταχύτητας και της μείωσης του κόστους αυτών, οι ανάγκες στρέφονται από την host-centric δομή του διαδικτύου σε ένα πιο content-oriented μοντέλο καθώς η χρήση του Internet πλέον αφορά περισσότερο την διάδοση πληροφορίας παρά την απλή σύνδεση απομακρυσμένων κόμβων. Τα ICN δίκτυα αναπτύσσονται για να καλύψουν αυτό το μειονέκτημα της αρχιτεκτονικής του Internet είτε σαν αυτοδύναμα μοντέλα είτε προσθετικά στα πρωτόκολλα διαδικτύου και δρομολόγησης που χρησιμοποιούνται. Μία συγκεκριμένη ICN αρχιτεκτονική είναι η NDN η οποία ακολουθεί τη βασική αρχή της ονομασίας των μεταδιδόμενων πακέτων.

Στην παρούσα εργασία θα μελετηθεί τη δομή και λειτουργία της NDN αρχιτεκτονικής, θα μοντελοποιηθούν διαφορετικά δίκτυα σε δικά μας εξειδικευμένα πειράματα διακοπτόμενης σύνδεσης και αποτυχίας πακέτων και θα εξαχθούν συμπεράσματα για τα χαρακτηριστικά και πλεονεκτήματα της συγκεκριμένης αρχιτεκτονικής στις συνθήκες προς μελέτη. Για τη προσομοίωση θα χρησιμοποιηθεί το πρόγραμμα mini-NDN μαζί με διάφορες εξωτερικές βιβλιοθήκες. Για την εξαγωγή και απεικόνιση αποτελεσμάτων θα γίνει χρήση του MATLAB και η συνολική πειραματική διαδικασία θα λάβει χώρο σε περιβάλλον linux, UBUNTU LTS 18.04.5 αξιοποιώντας τοπικές λειτουργίες και εγκατεστημένους compilers. Παράλληλος στόχος είναι η εξοικείωση και η συμβολή στην ανάπτυξη της βιβλιογραφίας και του εργαλείου mini-NDN το οποίο αποτελεί ένα σχετικά νέο εργαλείο ιδιαίτερα χρήσιμο στην προσομοίωση NDN δικτύων σε πραγματικό χρόνο.

Η εργασία χωρίζεται σε τρία βασικά κεφάλαια. Το πρώτο κεφάλαιο εξερευνά τις βασικές έννοιες που θα χρησιμοποιηθούν στην εργασία, αποτελεί δηλαδή ένα θεωρητικό υπόβαθρο το οποίο όμως επεκτείνεται και σε λεπτομερή εμβάθυνση των σημαντικών προς το αντικείμενο αρχιτεκτονικών δικτύου και λειτουργιών τους. Μετά από το πρώτο κεφάλαιο θα έχουν αποσαφηνιστεί οι κανόνες που διέπουν τα δίκτυα NDN, τα οποία θα μοντελοποιηθούν στο δεύτερο κεφάλαιο της εργασίας. Εκεί διευκρινίζονται οι τοπολογίες και οι υπόλοιπες παράμετροι των πειραμάτων, τίθενται οι στόχοι καθώς και τα συμβάντα που πρόκειται να τρέξουμε σε αυτά. Αναλύονται οι λειτουργίες του κώδικα, οι διάφορες διαδικασίες συλλογής δεδομένων και οι ρόλοι των κόμβων στο κάθε ξεχωριστό δίκτυο. Τέλος, το τρίτο κεφάλαιο της εργασίας αφορά την

αναπαράσταση και επεξεργασία των αποτελεσμάτων καθώς και τα τελικά συμπεράσματα της εργασίας.

Κεφάλαιο 1ο

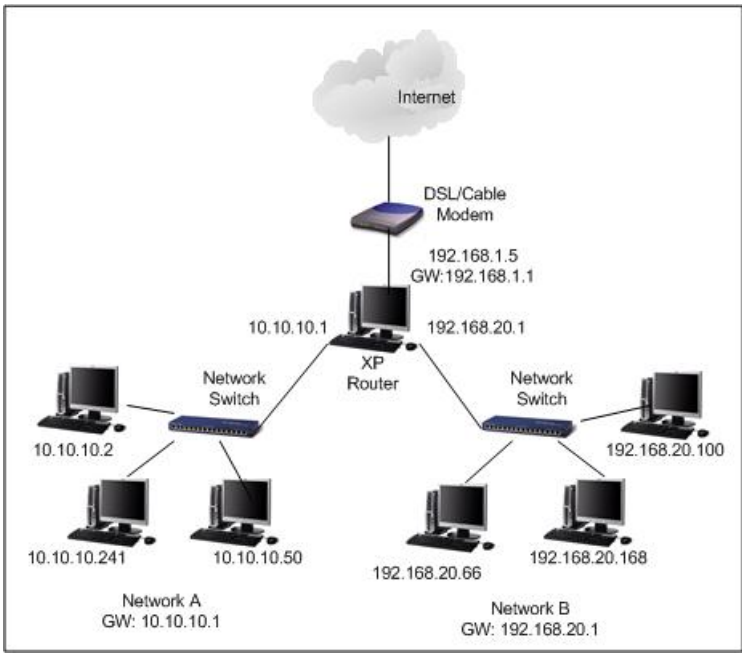
1.1 Πρωτόκολλο Διαδικτύου (Internet Protocol)

Το Internet είναι ένα σύστημα που αποτελείται από πολλά υποσύνολα δικτύων τα οποία επικοινωνούν μεταξύ τους για την ανταλλαγή δεδομένων. Ένα πρωτόκολλο δικτύου είναι ο σαφής, ορισμένος τρόπος για την επικοινωνία δύο ή περισσοτέρων κόμβων ή δικτύων χωρίς προβλήματα. Το κυριότερο πρωτόκολλο στο οποίο βασίζεται το Διαδίκτυο σήμερα είναι το IP (Internet Protocol) το οποίο επιτρέπει την αναφορά και σωστή δρομολόγηση πακέτων στους κόμβους καταναλωτές. Μαζί με το TCP (Transmission Control Protocol) και μερικά ακόμα πρωτόκολλα επικοινωνίας συνθέτουν το TCP/IP αρχιτεκτονικό μοντέλο, το οποίο χρησιμοποιείται σε πληθώρα δικτύων εκτός του Internet. Το IP παρουσιάστηκε από τους Vint Cerf και Bob Kahn το 1974 και οι πρώτες τρεις εκδόσεις του πρωτοκόλλου αποτέλεσαν πειραματικά στάδια αυτού που αργότερα θα ερχόταν να αποτελέσει τον βασικό πυλώνα του Διαδικτύου, το IPv4.

1.1.1 IP Διευθυνσιοδότηση

Το IP πρωτόκολλο βασίζεται στη χρήση διευθυνσιοδότησης για τον εντοπισμό δεδομένων και στην προσάρτηση κεφαλίδας στα πακέτα για τις διαδικασίες δρομολόγησης. Πακέτο ονομάζεται ένα σύνολο δεδομένων με κοινή κεφαλίδα που το δίκτυο μεταχειρίζεται σαν μία οντότητα.

Στα διάφορα υποσύνολα δικτύων (αυτόνομα συστήματα) που συνθέτουν το σημερινό Διαδίκτυο έχουν ανατεθεί συγκεκριμένα σύνολα IP διευθύνσεων. Έτσι, όταν ένα πακέτο φτάνει σε έναν δρομολογητή, ανάλογα με τη διεύθυνση αποστολής του προκύπτει το αυτόνομο δίκτυο στο οποίο πρέπει να προωθηθεί. Στη συνέχεια το πακέτο δρομολογείται κατά τον ίδιο τρόπο εσωτερικά στα μικρότερα υποδίκτυα μέχρι να φτάσει τον τελικό προορισμό του.



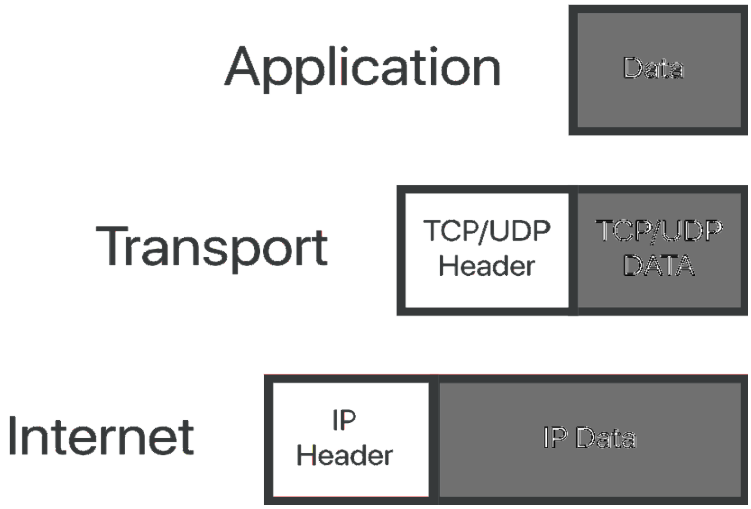
Εικόνα 1.1: Αναπαράσταση διασυνδεδεμένων δικτύων

Κάθε κόμβος που έχει τη δυνατότητα να αποθηκεύει δεδομένα αποκτά τη δική του μοναδική IP διεύθυνση και συμπερασματικά θα διευθυνσιοδοτηθεί οποιαδήποτε συνδεδεμένη στο δίκτυο συσκευή. Η IP διεύθυνση είναι μια σειρά από χαρακτήρες οι οποίοι όμως μπορούν να αντιστοιχηθούν μέσω DNS σε κανονικά ονόματα, γεγονός που διευκολύνει πολύ τους χρήστες καθώς μπορούν να διαχειρίζονται φυσικά ονόματα.

1.1.2 IP Ενθυλάκωση

Πληροφορίες ελέγχου που αφορούν το IP υπάρχουν προσαρτημένες σε κάθε δρομολογούμενο πακέτο οι οποίες χρησιμοποιούνται από τους δρομολογητές του δικτύου για τη σωστή πορεία του πακέτου. Η προσθήκη περιβλήματος στα δεδομένα που εμπεριέχει πληροφορίες ελέγχου ονομάζεται ενθυλάκωση (encaptulation) και αν η συγκεκριμένη διαδικασία πραγματοποιηθεί από το πρωτόκολλο IP τότε ονομάζουμε τα σύνολα των ομαδοποιημένων δεδομένων IP πακέτα. Σε κάθε IP κεφαλίδα υπάρχουν σημαντικές για τη δρομολόγηση πληροφορίες κωδικοποιημένες σε σειρές από bits. Διαιρεμένα σε συγκεκριμένα τμήματα, αυτά τα bits φέρουν πληροφορίες σχετικά με τον αποστολέα και παραλήπτη, το μέγεθος της ίδιας της κεφαλίδας, το μέγεθος του πακέτου, τον μέγιστο αριθμό αλμάτων που επιτρέπεται να κάνει το πακέτο πριν

απορριφθεί (TTL) και το είδος του πρωτοκόλλου μεταφοράς που χρησιμοποιείται. Στο IPv4 υπάρχουν 14 πεδία δεδομένων ελέγχου στην κεφαλίδα με ένα από αυτά να αποτελεί προαιρετικό τμήμα (Εικόνα 1.3).



Εικόνα 1.2: Επικεφαλίδες TCP και IP πακέτων

Version (4 bits)	IHL (4 bits)	Type of Service (8 bits)	Total Length (16 bits)	
Identification (16 bits)			Flags (3 bits)	Fragment Offset (13 bits)
Time to Live (8 bits)		Protocol (8 bits)	Header Checksum (16 bits)	
Source Address (32 bits)				
Destination Address (32 bits)				
Options and Padding (multiples of 32 bits)				

Εικόνα 1.3: Τμήματα IP κεφαλίδας του IPv4

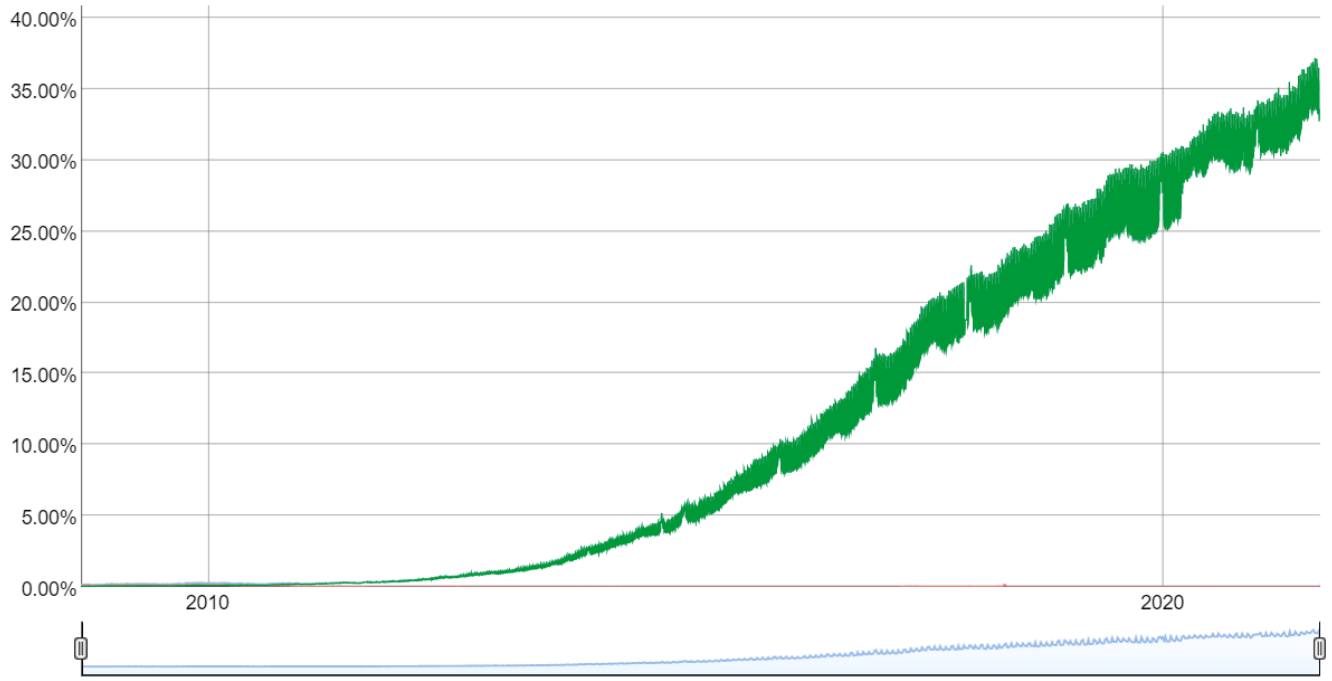
Το πρωτόκολλο IP διαθέτει μηχανισμούς για την αποφυγή προβλημάτων κατά την διάδοση πακέτων μέσα στο δίκτυο. Για τη δρομολόγηση των πακέτων χωρίς να προκύψει βρόγχος προώθησης (forwarding loop) το IP πρωτόκολλο χρησιμοποιεί την μία καλύτερη ή περισσότερες με ίδιο κόστος διαδρομές. Τα πακέτα που αποστέλλονται είναι πιθανό να ακολουθήσουν διαφορετικές διαδρομές

για να καταλήξουν εντέλει στον ίδιο τελικό κόμβο. Προκειμένου να επανασυνδεθούν δύο ή περισσότερα τμήματα του ίδιου πακέτου που ακολούθησαν διαφορετικές διαδρομές γίνεται χρήση της κεφαλίδας του κάθε τμήματος για την ταύτιση του ονόματος των πακέτων σε συνδυασμό με ένα δείκτη που ορίζει την σειρά με την οποία θα γίνει η συγχώνευση. Υπεύθυνο για την απόπλεξη και επανασύνδεση των τμημάτων αυτών είναι το πρωτόκολλο TCP που τρέχει “πάνω” στο IP. Μέσα σε ένα δίκτυο μπορούν να προκύψουν πολλαπλές αποπλέξεις.

1.1.3 Μειονεκτήματα του IP πρωτοκόλλου

Το end-to-end IP πρωτόκολλο αρχικά σχεδιάστηκε για την επικοινωνία μεταξύ ενός κλειστού κύκλου ινστιτούτων, πανεπιστημίων και ερευνητικών κέντρων [20]. Η είσοδος του ευρέως κοινού σε ένα διαδεδομένο, κοινό δίκτυο οδήγησε στην ανάπτυξη του World Wide Web, στις online συναλλαγές, στο streaming πολυμέσων και σε άλλες νέες δυνατότητες, αλλά παράλληλα βρήκαν χώρο ανάπτυξης η δημιουργία ιών, οι διαδικτυακές απάτες, οι παράνομες συναλλαγές κλπ. Ήδη έπρεπε να αναβαθμιστούν πτυχές όπως η αξιοπιστία του δικτύου, η προστασία των προσωπικών δεδομένων και η ασφάλεια πλοήγησης στον παγκόσμιο ιστό.

Στη σύγχρονη εποχή όμως, η μεγαλύτερη πρόκληση που αντιμετώπισε το IP μοντέλο ήταν ο κίνδυνος εξάντλησης του συνόλου των IP διευθύνσεων. Η τέταρτη έκδοση του πρωτοκόλλου IP (1983) αποτελούσε μέχρι πρόσφατα το επικρατέστερο IP μοντέλο στο Διαδίκτυο, αλλά λόγω των πεπερασμένων διευθύνσεων που ήταν δυνατό να δοθούν στους χρήστες εμφανίστηκε η ανάγκη για την εύρεση μιας καλύτερης λύσης. Το IPv6 ήρθε να αντικαταστήσει την τέταρτη έκδοση με τη βασική αλλαγή ότι το μέγεθος του συνόλου των διευθύνσεων πλέον περιγράφεται στην IP κεφαλίδα όχι από 32, αλλά από 128 bits. Από αρκετά νωρίς, οι περισσότερες συσκευές και domains μπορούσαν να υποστηρίξουν και τις δύο αυτές εκδόσεις. Παρακάτω (Εικόνα 1.4) αναπαρίσταται το σύγχρονο ποσοστό χρήσης των συνολικών διευθύνσεων του IPv6 όπως παρουσιάζεται από τα στατιστικά της Google [26]. Η κατάληψη της συνολικής χωρητικότητας των IP διευθύνσεων μπορεί να μεταβάλλεται μεταξύ των διαφορετικών χωρών, των διαφορετικών μηνών, ημερών και ωρών της ημέρας, αλλά παρόλα αυτά παρατηρείται μία μόνιμη, σταδιακή άνοδος.



Εικόνα 1.4: Ποσοστό χρήσης του συνόλου των IP διευθύνσεων συναρτήσει του χρόνου

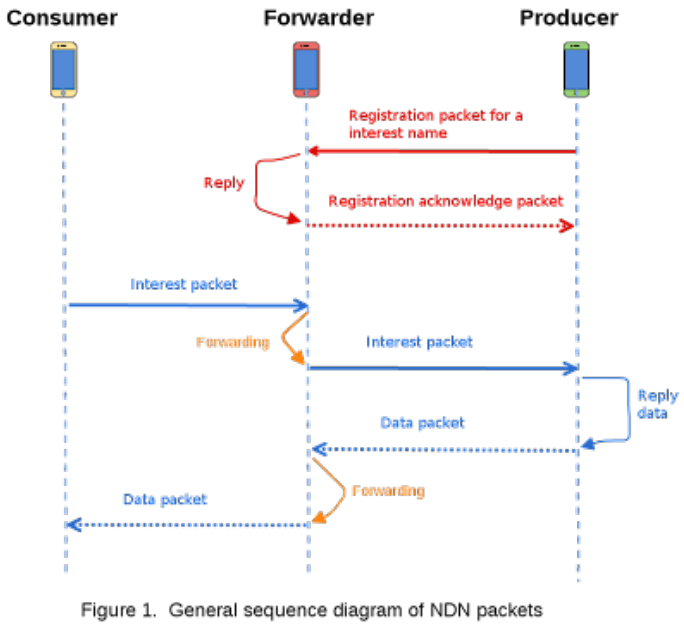
Το IPv6 είχε επιτυχία στο να προσφέρει μία λύση βιώσιμη για κάποια χρόνια, αλλά καθώς η ανάπτυξη της τεχνολογίας αυξάνεται με τομείς όπως το IoT να κερδίζουν συνεχώς έδαφος στην καθημερινότητά μας, συμπεραίνουμε ότι η ενώ μετάβαση στο συγκεκριμένο πρωτόκολλο ήταν η πλέον ασφαλής και ομαλή [25], δεν αντιμετώπισε ευθέως το πρόβλημα που φέρει το end-to-end μοντέλο στη σύγχρονη εποχή.

1.2 Αρχιτεκτονικό Μοντέλο Information Centric Networking

Το ICN είναι ένα νέο μοντέλο αρχιτεκτονικών δικτύων που στοχεύει να αντιμετωπίσει τις σύγχρονες απαιτήσεις και τα προβλήματα που προκύπτουν στις end-to-end επικοινωνίες στη μεγάλη κλίμακα του διαδικτύου. Η πλέον κυρίαρχη χρήση των δικτύων επικοινωνίας αφορά τη διάδοση πληροφορίας. Το ΤΙ μεταδίδεται σε ένα δίκτυο είναι συχνότερα πιο σημαντικό από το από ΠΟΥ μεταδίδεται. Έτσι, σχεδιάστηκε για να είναι κατάλληλο για την αυξανόμενη ανταλλαγή πληροφορίας και συστήνει την ονομασία των δεδομένων στο δίκτυο σαν βασική αρχή του διαδικτύου.

Ονομάζει τα ίδια τα πακέτα πληροφορίας με μοναδικές object IDs ανεξάρτητες από την τοποθεσία στην οποία βρίσκονται. Ξεπερνά την ανάγκη για επικοινωνία από έναν αρχικό σε έναν

τελικό κόμβο όπως συμβαίνει με το IP και χρησιμοποιεί αιτήματα περιεχομένου για την απόκτηση των επιθυμητών πακέτων. Η ικανοποίηση αυτών των αιτημάτων μπορεί να επέλθει από περισσότερους από έναν χρήστες που διαθέτουν αντίγραφο του ζητούμενου πακέτου. Έτσι αντιμετωπίζει παράλληλα προβλήματα όπως διακοπές στη σύνδεση κόμβων, flash-crowd συμβάντα κλπ. Η ασφάλεια πλέον έχει ενσωματωθεί στα μεταδεδομένα του κάθε πακέτου. Το ICN είναι το κατάλληλο μοντέλο για επικοινωνίες σε εφαρμογές όπως στο Internet of Things, στα 5G δίκτυα, στα τροχαία δίκτυα κ.α.



Εικόνα 1.5: Χρονοδιάγραμμα ενεργειών στην ICN ανταλλαγή πακέτων

Το ICN περιέχει φιλοσοφίες όπως αυτές των

- **P2P networking** (peer-to-peer) που προτείνει να αποσυμφορηθούν οι servers από τον μεγάλο όγκο πληροφορίας διαμοιράζοντας την μεταξύ των χρηστών (peers). Χαρακτηριστικά της είναι η προσαρμοστικότητα του δικτύου στις αλλαγές και η δυνατότητα ανάκτησης τμημάτων (chunks) δεδομένων από διαφορετικούς χρήστες.

- **CDN** (Content Delivery Networking) που ανακατευθύνει το αίτημα για κάποιο πακέτο στην επιθυμητή cache που το περιέχει. Η επιλογή της cache βασίζεται μόνο στην θέση της μνήμης στην τοπολογία του δικτύου.

1.2.1 Λειτουργία και ρόλοι κόμβων

Ο ρόλος ενός ICN κόμβου μπορεί να περιλαμβάνει καθήκοντα παραγωγού, καταναλωτή και δρομολογητή ταυτόχρονα. Οι κόμβοι διαθέτουν χώρο αποθήκευσης και μπορούν να προωθήσουν πακέτα ενδιαφέροντος ή δεδομένων με δυνατότητα υπολογισμού του επιθυμητού επόμενου δρομολογητή. Χαρακτηριστικό των ICN δικτύων είναι ότι οι μονάδες που αποτελούν το δίκτυο δεν γνωρίζουν άμεσα τη τοποθεσία των υπολοίπων μη γειτονικών μονάδων μέσα από δείκτες ή διευθύνσεις. Η τοπολογία του δικτύου η οποία μπορεί να μεταβάλλεται λόγω σύνδεσης νέων κόμβων και αποσύνδεση ή κατάρρευση άλλων, διευκρινίζεται για κάθε κόμβο από θεμελιώδεις διεργασίες του δικτύου που βασίζονται στην διαφήμιση περιεχομένου τρίτων, μη γειτονικών κόμβων. Όταν οι τελικοί σταθμοί των πακέτων είναι γειτονικοί κόμβοι, τότε η δρομολόγηση των πακέτων γίνονται άμεσα με μικρή εισαγόμενη καθυστέρηση από το δίκτυο.

1.2.2 Εφαρμογή των ICN δικτύων

Η προσπάθεια να δημιουργηθούν όλες οι απαραίτητες προϋποθέσεις για να μπορεί το ICN να σταθεί μόνο του σαν βασικό μοντέλο επικοινωνίας ονομάζεται clean-slate approach. Αυτή η απόλυτη αντικατάσταση του IP αποτελεί ένα εξαιρετικά σύνθετο εγχείρημα και φέρει πολλά ζητήματα που πρέπει να λυθούν καθώς ταυτόχρονα πρέπει να αναλογιστούν πολλοί τεχνικοί και οικονομικοί παράγοντες. Η προσωρινή ή μόνιμη αποθήκευση μέσα στο δίκτυο απαιτεί αποθηκευτικό χώρο, το κόστος της μετακίνησης λόγω αλλαγής της νοοτροπίας κάθε συστήματος είναι τεράστιο, και απαιτείται η προσαρμογή τόσο του software όσο και του hardware. Για τον λόγο αυτό η έρευνα επικεντρώνεται στην αρμονική συνύπαρξη των δύο φιλοσοφιών. Το ICN μοντέλο μπορεί να τροποποιηθεί κατά συγκεκριμένο τρόπο έτσι ώστε να λειτουργεί “πάνω” στην ήδη θεμελιωμένη IP δομή επηρεάζοντας για παράδειγμα το TCP πρωτόκολλο. Σε αυτά τα δίκτυα μπορεί να υπάρχουν κόμβοι που να επικοινωνούν χρησιμοποιώντας το μοντέλο του ICN ενώ άλλοι το IP πρωτόκολλο. Δεν είναι απίθανο επίσης να εγκατασταθεί δομή ICN η οποία φιλοξενεί λειτουργίες IP προκειμένου να καταστήσει εφικτή την επικοινωνία με τρίτα δίκτυα.

1.2.3 Σύγκριση IP με ICN

Εκτός από τα προφανή προβλήματα που τίθεται να λύσει το νέο μοντέλο δικτύων, προκύπτουν παράλληλα πολλά ακόμα πλεονεκτήματα και νέες δυνατότητες ανάπτυξης που τα end-to-end μοντέλα όπως το IP περιόριζαν. Είναι σημαντικό να μελετήσουμε τα δύο μοντέλα δικτύων σε σύγκριση για την ανάλυση των προαναφερθέντων πλεονεκτημάτων οπότε παρακάτω αναλύονται τα πιο σημαντικά από αυτά,

Εμπιστοσύνη μόνο στον παραγωγό

Ένα βασικό μειονέκτημα στο IP πρωτόκολλο είναι η ανάγκη για την εμπιστοσύνη του μέσου στο οποίο γίνεται η σύνδεση, αυξάνοντας έτσι τα μέτωπα εμπιστοσύνης στις κρυπτογραφημένες συνδέσεις. Εφόσον η ασφάλεια των ICN είναι προσαρμοσμένη στο ίδιο το πακέτο και ακολουθεί την αρχή της αποθήκευσης και προώθησης, δεν τίθεται ζήτημα εμπιστοσύνης τρίτου. Προκύπτει έτσι μεγαλύτερη αξιοπιστία στο δίκτυο αλλά και δίνεται η δυνατότητα για ανάπτυξη επιπλέον χρήσιμων λειτουργιών όπως οι παρακάτω.

Λειτουργίες caching και proxy

Σε αντίθεση με την κρυπτογραφημένη επικοινωνία του IP πρωτοκόλλου στην οποία η transparent caching λειτουργία δεν μπορεί να επιτευχθεί, τα ICN δίκτυα μπορούν να διατηρήσουν caching και proxy λειτουργίες ενδιάμεσα στους κόμβους της σύνδεσης. Αυτό συμβαίνει λόγω της ενσωματωμένης ασφάλειας στα πακέτα που διαδίδονται. Το γεγονός ότι τα πακέτα έχουν ενσωματωμένα τα ονόματά τους και τις κρυπτογραφημένες υπογραφές τους τα καθιστά ανεξάρτητα από τους παραγωγούς τους και από τους κόμβους που τα αποζητούν. Proxy ονομάζεται ένας διαμεσολαβητής server μεταξύ του δικτύου και ενός τελικού χρήστη με σκοπό το φιλτράρισμα, την απόκρυψη προσωπικών δεδομένων και άλλες λειτουργίες ασφαλείας. Caching ονομάζεται η διαδικασία κατά την οποία δεδομένα αποθηκεύονται προσωρινά σε κάποια δομή δεδομένων με σκοπό την ταχύτερη εξυπηρέτηση καταναλωτών σε πιθανό επόμενο αίτημα. Στα δίκτυα NDN αυτή η δομή δεδομένων είναι το Content Store των κόμβων που θα αναλυθεί περαιτέρω παρακάτω. Η

συγκεκριμένη δυνατότητα στα ICN δίκτυα θα μελετηθεί αρκετά κατά την διαμόρφωση των διαφόρων πειραμάτων στην συγκεκριμένη εργασία.

Δυνατότητα ανάπτυξης και βελτίωσης μοντέλων ελέγχου συμφόρησης

Το TCP congestion control έχει μελετηθεί και έχει φτάσει σε ένα σημείο-ταβάνι που δεν επιδέχεται περαιτέρω βελτίωση, ενώ αντίθετα το congestion control στο σύστημα όπου τα πακέτα είναι ονομασμένα δεν έχει αναπτυχθεί αρκετά και φαίνεται να έχει προοπτικές για μεγάλη βελτίωση απόδοσης throughput. Η δομή του νέου μοντέλου δικτύου από μόνη της συνεισφέρει στον έλεγχο του φόρτου δεδομένων όπως εξηγείται παρακάτω.

Καλύτερος έλεγχος συμφόρησης

Υπάρχει περίπτωση να υπάρξει υπερχείλιση του συστήματος με στιγμιαία δραστηριότητα μεγάλου πλήθους κόμβων (χρηστών). Συνήθως αυτό το πρόβλημα μπορεί να λυθεί με πρόβλεψη και ρύθμιση του συστήματος για την ανταπόκριση στις απαιτήσεις του δικτύου. Η λογική της διαφήμισης/ζήτησης πακέτων συνεισφέρει σε μεγάλο βαθμό στην απλή διαχείριση οποιουδήποτε αριθμού από αιτήματα χρηστών στο δίκτυο, καθώς μπορεί να οδηγηθεί από τους δέκτες των πακέτων ενδιαφέροντος.

1.3 Δίκτυα Named Data Networking (NDN)

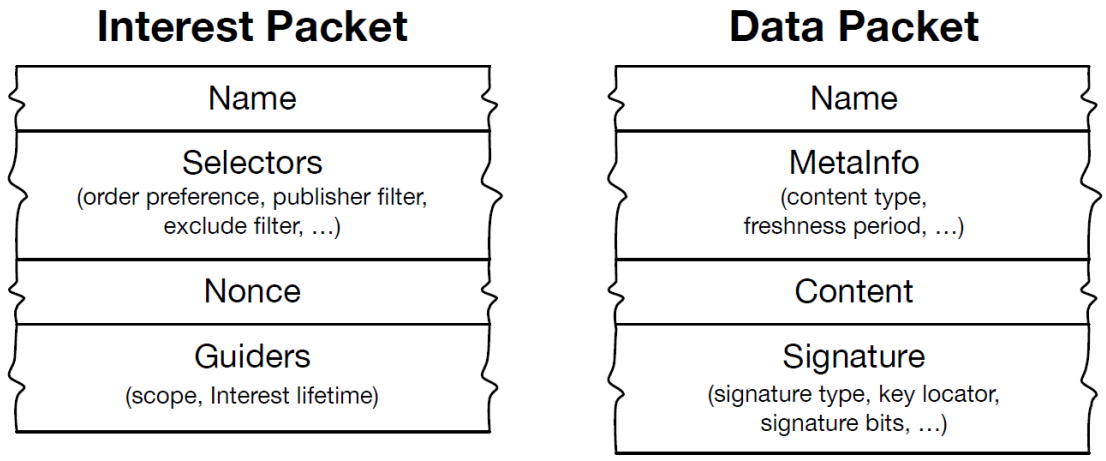
Το Named Data Networking (NDN) είναι μια consumer-driven αρχιτεκτονική που ανήκει στην κατηγορία των ICN δικτύων, δηλαδή ονομάζει δεδομένα έτσι ώστε να τα αναγνωρίζει και να τα ανακτά. Αποτελεί ένα από τα επιστημονικά έργα που δέχονται χρηματοδότηση από το Εθνικό Επιστημονικό Ίδρυμα (NSF) των ΗΠΑ υπό το εγχείρημα FIA (Future Internet Architectures) και η ανάπτυξή του απασχολεί πολλές επιστημονικές ομάδες σε πάνω από 15 ιδρύματα και πανεπιστήμια.

Η αρχή λειτουργίας του βασίζεται στην ICN ονομασία των μεταδιδόμενων πακέτων, στην ενσωματωμένη ασφάλεια του δικτύου στο ίδιο το πακέτο και στην δυνατότητα να αποθηκεύει προσωρινά αιτήματα δεδομένων προκειμένου να ενισχύσει αποδοτικές μεθόδους δρομολόγησης και αποφυγής προβλημάτων.

1.3.1 Ονομασία πακέτων δεδομένων

Τα ονόματα ID χρησιμοποιούνται απευθείας στην προώθηση και δρομολόγηση των πακέτων. Στη σωστή λειτουργία του δικτύου ένας χρήστης στέλνει ένα πακέτο-αίτημα (Interest Packet) και δέχεται πίσω ένα πακέτο δεδομένων (Data Packet) το οποίο περιέχει το όνομα του πακέτου που ζητήθηκε, τα επιθυμητά δεδομένα και την υπογραφή του αρχικού παραγωγού του πακέτου. Στην περίπτωση όπου δύο ή περισσότερα πακέτα δεδομένων μπορούν να εξυπηρετήσουν ένα αίτημα τότε το σύστημα επιστρέφει μόνο ένα εξ αυτών έτσι ώστε να αποφύγει ανούσια υπερφόρτωση του δικτύου.

Οι forwarders του δικτύου δημιουργούν μια συνθήκη για κάθε Interest Packet η οποία έχει ισχύ μέχρι να επιστραφεί μία φορά το ζητούμενο πακέτο δεδομένων ή να ξεπεραστεί ένα ορισμένο χρονικό διάστημα (Interest Lifetime). Η χρήση αυτής της τεχνικής προώθησης επιτρέπει την δρομολόγηση μέσα από πολλαπλές διαδρομές επειδή διατίθεται ενσωματωμένο σύστημα που εξαλείφει τους βρόγχους, καθώς και ισορροπία στη ροή δεδομένων. Αυτή η διεργασία ονομάζεται stateful forwarding.

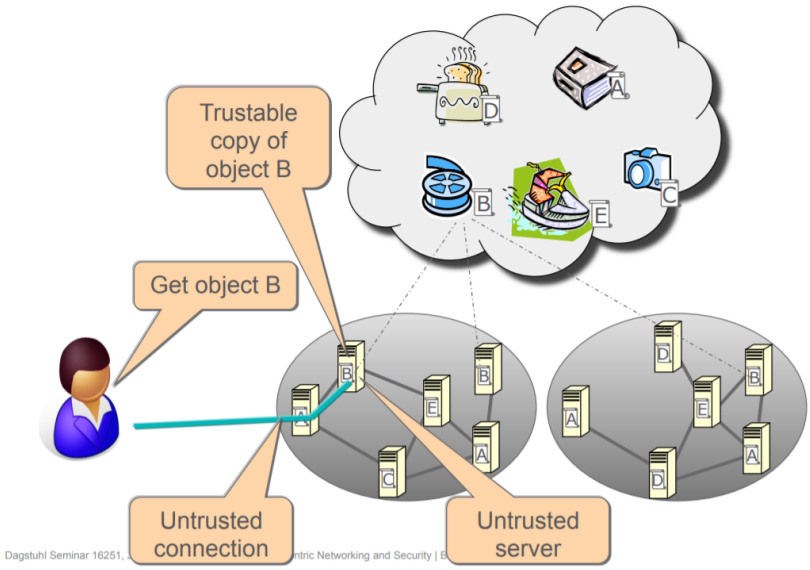


Εικόνα 1.6: Περιεχόμενα των πακέτων ενδιαφέροντος και δεδομένων στα NDN δίκτυα

1.3.2 Ασφάλεια Δικτύου

Η ασφάλεια του δικτύου ενσωματώνεται πλέον στα ίδια τα δεδομένα. Με τα πακέτα δεδομένων να διαθέτουν κρυπτογραφημένη υπογραφή από τον δημιουργό τους, καθώς και με

κρυπτογραφημένη επιπλέον ευαίσθητη πληροφορία, η αξιολόγηση της ασφάλειας του πακέτου ανεξαρτητοποιείται από την προέλευσή του και τον τρόπο με τον οποίο έφτασε στον παραλήπτη αλλά στρέφεται στα μεταδεδομένα (metadata) του ίδιου του πακέτου. Έτσι, ο παραλήπτης διαβεβαιώνεται πως το πακέτο που δέχθηκε είναι πράγματι από τον γνήσιο δημιουργό του και δεν έχει παρέμβει κάποιος τρίτος στο περιεχόμενό του κατά τη διαδικασία της δρομολόγησης. Για τη σύνδεση του ονόματος του κάθε πακέτου με το περιεχόμενο που περιγράφει χρησιμοποιείται συνήθως μια μοναδικά αντιστοιχισμένη κρυπτογραφημένη υπογραφή ή ένας κρυπτογραφικός κατακερματισμός των δεδομένων του πακέτου, είτε αυτό διαθέτει όνομα είτε όχι.



Εικόνα 1.7: Μείωση μετώπων εμπιστοσύνης – Ενσωματωμένη ασφάλεια στο πακέτο

Η συγκεκριμένη αρχιτεκτονική παρέχει εγκυρότητα για το περιεχόμενο, που σημαίνει ότι το ληφθέν πακέτο είναι ένα ολοκληρωμένο και μη αλλοιωμένο αντίγραφο του αρχικού πακέτου δεδομένων, για την προέλευση, ότι δηλαδή το περιεχόμενο έχει παραχθεί από μια αξιόπιστη πηγή και για τη συνάφεια του αιτήματος με το επιστρεφόμενο αποτέλεσμα. Βασικός παράγοντας που επιτρέπει τα παραπάνω είναι η αντιστοίχιση του ονόματος δεδομένων με συγκεκριμένο κλειδί-υπογραφή το οποίο δηλώνει από ποιόν παραγωγό προήλθε το πακέτο. Φυσικά η αυθεντικότητα του παραγωγού δεν ισοδυναμεί με αξιοπιστία των δεδομένων και για τον λόγο αυτό απαιτείται περαιτέρω έλεγχος για την καταλληλότητα του ονόματος σε σχέση με το περιεχόμενο των δεδομένων.

1.3.3 Δρομολόγηση (NLSR)

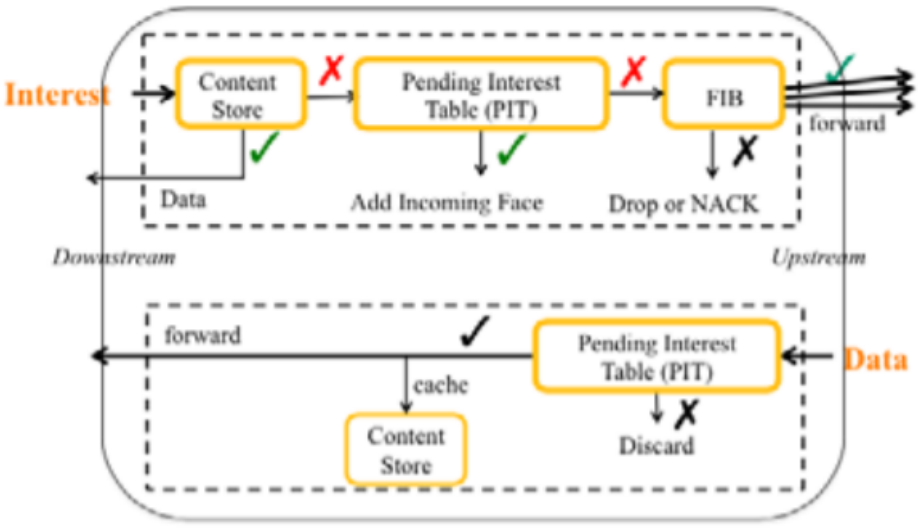
Καθώς η μετάδοση πακέτων βασίζεται στην ονομασία της πληροφορίας, αναγκαστικά απαιτείται και ένα νέο πρωτόκολλο δρομολόγησης. Το NLSR (Named Data Link State Routing Protocol) είναι το πρωτόκολλο δρομολόγησης για την αρχιτεκτονική NDN το οποίο εξασφαλίζει την πρόσβαση στα ID των πακέτων αντί στις IP διευθύνσεις τους. Για την ανταλλαγή μηνυμάτων μπορεί εν δυνάμει να χρησιμοποιήσει οποιοδήποτε διαθέσιμο κανάλι επικοινωνίας όπως το Ethernet και διάφορα IP ή TCP/UDP tunnels. Νέες ενημερώσεις για τη δρομολόγηση πακέτων διανέμονται μέσα από τα πακέτα αιτήματος και δεδομένων. Το συγκεκριμένο πρωτόκολλο ξεχωρίζει από τα υπόλοιπα καθώς εξασφαλίζει με υπογραφές και επαλήθευση την αυθεντικότητα των κόμβων της τοπολογίας και προσφέρει περισσότερες από μία δυνατές διαδρομές δρομολόγησης. Σε κάθε κόμβο, το NLSR φτιάχνει μια λίστα με κατανεμημένες κατά σειρά προτεραιότητας επιλογές δρομολόγησης για την κάθε επικεφαλίδα ονόματος πακέτου. Αυτή η ιεραρχική λίστα επιτρέπει την δυνατότητα για προσαρμοσμένες και έξυπνες στρατηγικές δρομολόγησης.

Η λειτουργία του συστήματος ακολουθεί την παρακάτω λογική. Κάθε κόμβος έχει τις δικές του τρεις δομές δεδομένων που ονομάζονται Forwarding Information Base (FIB), Pending Interest Table (PIT) και Content Store (CS). Στο FIB κάθε κόμβου γίνονται καταχωρήσεις που υποδεικνύουν την θέση συγκεκριμένων πακέτων και περιλαμβάνουν την επικεφαλίδα ονομασίας τους όπως και ένα ή περισσότερα επόμενα άλματα προς τη κατεύθυνσή τους (hops). Χρησιμοποιείται για να προωθηθούν τα πακέτα αιτημάτων στα ζητούμενα πακέτα δεδομένων. Στο PIT καταχωρούνται δεδομένα για όλα τα Interest packets που δρομολογήθηκαν αλλά δεν ικανοποιήθηκε το αίτημά τους και στο CS αποθηκεύονται πακέτα δεδομένων που έχει δεχτεί ο κόμβος.

Όταν ένας κόμβος επιθυμεί συγκεκριμένα δεδομένα, στέλνει ένα πακέτο αιτήματος όπως ήδη προαναφέρθηκε και περιμένει σαν απάντηση από το σύστημα το ζητούμενο πακέτο δεδομένων. Έτσι όταν το Interest πακέτο φτάνει σε έναν κόμβο το πρώτο πράγμα που συμβαίνει είναι να ελεγχθεί το CS για πιθανό πακέτο που ικανοποιεί το αίτημα. Εάν δεν υπάρχει κάποιο τέτοιο πακέτο, ελέγχεται το PIT για την πιθανότητα να υπάρχει ήδη κάποιο αίτημα για το ίδιο πακέτο που έχει προηγουμένως προωθηθεί. Σε ένα τέτοιο ενδεχόμενο είναι περιττό να σταλθεί δεύτερο ίδιο αίτημα, οπότε προστίθεται ο αριθμός του μετώπου σύνδεσης στην καταχώρηση που ήδη υπάρχει στο PIT έτσι ώστε να ικανοποιηθεί μαζί και το νέο αίτημα. Η συγκεκριμένη λειτουργία είναι αρκετά συνηθισμένη και αναφέρεται ως “interest aggregation”. Εάν αντίθετα δεν υπάρχει κάποια ίδια καταχώρηση στο PIT,

τότε σύμφωνα με το FIB, το Interest Packet δρομολογείται στον επόμενο κόμβο και ταυτόχρονα προστίθεται μια νέα καταχώρηση για το συγκεκριμένο αίτημα στο PIT. Αν στο FIB υπάρχουν περισσότερα από ένα επόμενα άλματα, τότε καλείται ο αλγόριθμος “forwarding strategy” για να καθοριστεί η βέλτιστη πορεία δρομολόγησης. Όταν τελικά βρεθεί το πακέτο δεδομένων που ζητείται, αυτό θα ακολουθήσει τη διαδρομή που υποδεικνύουν τα PITs ικανοποιώντας έτσι έναν ή και περισσότερους ενδιαφερόμενους κόμβους.

Ο αλγόριθμος forwarding strategy αποτελεί όπως δηλώνει και η ονομασία μια ορισμένη στρατηγική προώθησης πακέτων, αλλά δεν είναι πάντα η ίδια και απαιτεί σωστό σχεδιασμό για να είναι αποδοτική σε διαφορετικές περιστάσεις. Δέχεται δεδομένα από τις FIBs των διαθέσιμων κόμβων μαζί με άλλες παραμέτρους που μπορεί να ποικίλουν ανάλογα με το δίκτυο για να δημιουργηθεί η βέλτιστη κατά τον αλγόριθμο επιλογή για την πορεία δρομολόγησης. Ονομάζουμε upstream την πορεία δρομολόγησης όταν αυτή ακολουθεί τα Interest Packets ενώ αντίθετα μιλάμε για downstream forwarding όταν ακολουθεί τα πακέτα δεδομένων ή τα Interest NACKs. Μερικές από τις πιο διαδεδομένες στρατηγικές δρομολόγησης που θα αναλυθούν περαιτέρω στα πειράματα της εργασίας είναι η “Best Route”, η “Multicast” και η “Access”.



Εικόνα 1.8: Διαδικασία προώθησης Interest και Data σε δρομολογητές

1.3.4 Προσδιορισμός Τοπολογίας Δικτύου

Για τη σωστή λειτουργία του συστήματος πρέπει να καθίσταται πάντα γνωστή η τοπολογία δικτύου και σε αυτήν να εξασφαλίζεται η δυνατότητα ανίχνευσης της κάθε ζητούμενης ονομασίας πακέτου. Για αυτό το λόγο σε κάθε κόμβο του δικτύου υπάρχει μια επιπλέον δομή δεδομένων που χρησιμοποιείται από τον NLSR για την ανίχνευση πιθανών σφαλμάτων σύνδεσης, για την καθιέρωση γειτονικών συνδέσεων στους κόμβους και για την διάδοση των αλλαγών στα αποθηκευμένα τους πακέτα.

Αυτή ονομάζεται Link State Database (LSDB) και σε αυτήν καταχωρούνται οι πιο πρόσφατες εκδόσεις των LSAs. Ένα LSA (Link State Advertisement) διαδίδεται σε ολόκληρο το δίκτυο όταν ένας κόμβος ανιχνεύσει την πτώση ή επαναφορά της σύνδεσής του με άλλον κόμβο ή όταν προστεθεί ή αφαιρεθεί κάποια κεφαλίδα ονόματος στη βάση δεδομένων του. Με αυτό τον τρόπο ο NLSR καταφέρνει να καθιερώνει και να διατηρεί συνδέσεις μεταξύ των γειτονικών κόμβων. Από τη στιγμή που καθιερωθεί η τοπολογία του δικτύου, με εφαρμογή του αλγορίθμου του Dijkstra συνήθως υπολογίζονται τα πολλαπλά επόμενα άλματα για κάθε κόμβο. Από τα LSAs γνωρίζουμε τα ονόματα δεδομένων που είναι προσαρτημένα σε κάθε κόμβο οπότε είναι εφικτή η δημιουργία της FIB λίστας με διαδοχικά άλματα για την ανάκτηση οποιουδήποτε πακέτου στο δίκτυο. Σε κάθε κόμβο υπάρχει ένας τελεστής που διευκρινίζει τον αριθμό των διαφορετικών διαδρομών σε κάθε όνομα έτσι ώστε να υπάρχει βελτιωμένη διαχείριση του FIB όταν υπάρχουν πολλοί κόμβοι που γειτνιάζουν.

Για την ασφάλεια του δικτύου, κάθε LSA είναι επίσης υπογεγραμμένο και πιστοποιημένο από τον κόμβο από τον οποίο προήλθε έτσι ώστε να εξασφαλίζεται η αυθεντικότητα της προέλευσης. Η επικεφαλίδα του LSA ακολουθεί την παρακάτω μορφή /<network>/NLSR/LSA και προστίθεται το τμήμα /<site>/<router> για να ξεχωρίζουν τα LSA που προέρχονται από άλλους routers NLSR. Στην Εικόνα 1.9 φαίνεται ένα στιγμιότυπο από το FIB ενός κόμβου στο οποίο έχει αποθηκευτεί η πιο πρόσφατη έκδοση ενός LSA που ενημερώνει για τα διαθέσιμα επόμενα άλματα. Τέλος, καθώς στον ίδιο router μπορεί να υπάρχουν περισσότερα από ένα LSA, μπορούμε να εισάγουμε ένα επιπλέον τμήμα που τα ξεχωρίζει με έναν διαφορετικό ID αριθμό:

/<LSA-prefix>/<site>/<router>/LsType.1/<version>

/<LSA-prefix>/<site>/<router>/LsType.2/LsId.<ID>/<version>

```
FIB:
  /localhost/nfd/rib nexthops={faceid=258 (cost=0)}
  /ndn/d-site/%C1.Router/cs/d nexthops={faceid=264 (cost=20)}
  /ndn/c-site/%C1.Router/cs/c nexthops={faceid=266 (cost=10)}
  /localhop/ndn/nlsr/sync/%FD%08 nexthops={faceid=267 (cost=0), faceid=264 (cost=10), faceid=266 (cost=10)}
  /ndn/c-site/c nexthops={faceid=266 (cost=10)}
  /ndn/d-site/d nexthops={faceid=264 (cost=20)}
  /ndn/a-site/%C1.Router/cs/a/nlsr/KEY nexthops={faceid=267 (cost=0)}
  /localhop/ndn/nlsr/LSA nexthops={faceid=267 (cost=0), faceid=264 (cost=10), faceid=266 (cost=10)}
  /ndn/a-site/%C1.Router/cs/a/nlsr nexthops={faceid=267 (cost=0)}
  /ndn/b-site/%C1.Router/cs/b nexthops={faceid=264 (cost=10)}
```

Εικόνα 1.9: Δομή FIB στην οποία βρίσκεται αποθηκευμένο LSA πακέτο

Όταν ένα όνομα πακέτου απεγγράφεται από τη βάση δεδομένων ενός κόμβου, το αντίστοιχο LSA αποστέλλεται σε όλο το δίκτυο με το εξής αναγνωριστικό. Στην κεφαλίδα του LSA το πλαίσιο is-Valid αλλάζει σε 0, οπότε κάθε κόμβος που θα το δεχτεί διαγράφει το LSA με την αντίστοιχη ονομασία από το LSDB του και ανανεώνει το FIB.

1.3.5 Μηχανισμοί Αποφυγής Σφαλμάτων

Όσον αφορά το πλήθος νέων LSAs που μπορεί να προκύψουν ταυτόχρονα στο δίκτυο, ο κάθε δρομολογητής επιτρέπει νέα LSA μόνο όταν διαθέτει ελεύθερους CPU κύκλους. Υπάρχει δηλαδή ενημέρωση που οδηγείται από τον δέκτη. Ταυτόχρονα, αν κάποιο LSA μείνει για αρκετό χρονικό διάστημα αμετάβλητο και χωρίς κάποια νέα ενημέρωση σε κάποιον κόμβο, αυτό θα διαγραφεί καθώς διαθέτει συγκεκριμένο χρόνο ζωής. Έτσι, αν παρουσιαστεί κάποιο σφάλμα σύνδεσης σε κάποιον δρομολογητή, τα LSA που προήλθαν από αυτόν δεν θα καταλαμβάνουν άσκοπα χώρο στην LSDB των υπολοίπων κόμβων.

Για περαιτέρω έλεγχο πιθανής πτώσης σύνδεσης το NLSR διαθέτει έναν επιπλέον μηχανισμό ανταλλαγής πακέτων ενδιαφέροντος στους γειτονικούς κόμβους. Στην περίπτωση που δεν υπάρξει αποστολή νέας ενημέρωσης LSA εντός των προβλεπόμενων χρονικών ορίων πριν το timeout, το σύστημα εντοπίζει πιθανό πρόβλημα σύνδεσης και στέλνει πολλά διαδοχικά πακέτα με πολύ μικρή χρονική διαφορά μεταξύ τους. Αντιμετωπίζει έτσι την πιθανή απόρριψη πακέτων ενδιαφέροντος κατά τη μετάδοση και εάν δεν λάβει επιβεβαιώσεις για τα νέα πακέτα χαρακτηρίζει τη σύνδεση νεκρή. Ο κόμβος στη συνέχεια συνεχίζει να στέλνει πακέτα ενδιαφέροντος σε αραιά μεταξύ τους διαστήματα για να μην προκαλέσει φόρτο στο δίκτυο και όταν εντέλει λάβει επιβεβαίωση θεωρεί ότι η σύνδεση ανακτήθηκε και ανανεώνει την κατάσταση σύνδεσης στο LSA που στη συνέχεια θα διανείμει σε ολόκληρο το δίκτυο.

1.3.6 Προεκτάσεις NDN

Προκειμένου να αντιμετωπιστούν μειονεκτήματα που παρουσιάζει το NDN σε υψηλά μεταβαλλόμενα δίκτυα στα οποία προκύπτουν καθυστερήσεις, πτώσεις συνδέσεων κλπ., αναπτύσσονται παραλλαγές του βασικού μοντέλου αρχιτεκτονικής που παρουσιάζουν χαρακτηριστικά όπως ανεκτικότητα στην καθυστέρηση επιστροφής πακέτων και έχουν ιδιαίτερη χρησιμότητα σε IoT περιβάλλοντα και κινούμενα, ασύρματα δίκτυα [14]. Ένα παράδειγμα τέτοιου μοντέλου είναι το NDN-DTN που αποτελεί μια προσαρμογή της φιλοσοφίας του Delay Tolerant Networking μοντέλου στην NDN αρχιτεκτονική. Το Delay Tolerant Networking (DTN) επιδιώκει να αντιμετωπίσει τα τεχνικά ζητήματα που προκύπτουν σε ετερογενή δίκτυα στα οποία ενδέχεται να μην υπάρχει συνεχής συνδεσιμότητα.

Κεφάλαιο 2ο

2.1 Εισαγωγή

Το δεύτερο κεφάλαιο που αποτελεί το πειραματικό κομμάτι της εργασίας αφορά την ανάπτυξη διαφόρων σεναρίων δικτύων πάνω στα οποία θα πραγματοποιηθούν πειράματα. Θα αναλυθούν το εργαλείο mini-NDN με τις δυνατότητές του, οι τοπολογίες δικτύων στις οποίες θα βασιστούν τα πειράματα, οι κώδικες που αντιπροσωπεύουν τους ρόλους των διαφορετικών κόμβων στο δίκτυο, τα συμβάντα που λαμβάνουν χώρα στη πορεία της πειραματικής διαδικασίας και οι διαφορετικοί τρόποι κατά τους οποίους εξάγουμε τις επιθυμητές πληροφορίες από το σύστημα για την τελική ανάλυση.

2.2 Λογισμικά Μοντελοποίησης

2.2.1 Mini-NDN

Το miniNDN είναι ένα open-source εργαλείο για προσομοίωση δικτύων σε πραγματικό χρόνο, (emulator) το οποίο είναι ειδικά σχεδιασμένο για τον πειραματισμό σε δίκτυα με NDN αρχιτεκτονική. Είναι άμεσα συνδεδεμένο με βιβλιοθήκες του NDN όπως η ndn-cxx, με τα NLSR και NFD και άλλα προγράμματα που εμπλουτίζουν τις δυνατότητές του. Πιο συγκεκριμένα, το miniNDN είναι μια προέκταση του Mininet που προσομοιώνει αλληλεπιδράσεις μεταξύ κόμβων, στους οποίους εκτελούνται εξωτερικά προγράμματα, όπως είναι από προεπιλογή τα NFD και NLSR. Αντί να τρέχει μαθηματικά μοντέλα για προσομοίωση των δικτύων όπως το ndnSim, χρησιμοποιεί τον πυρήνα των linux για να εφαρμόσει πραγματικό NDN κώδικα και έτσι μπορούν να υπάρξουν πειραματισμοί και με την ίδια τη δομή της αρχιτεκτονικής.

Το πρόγραμμα συνεργάζεται στενά με τις διεργασίες των LINUX καθώς μπορεί πολύ εύκολα να αναθέσει σαν κόμβο του δικτύου το ίδιο το λειτουργικό σύστημα και κάνει ευρεία χρήση των προσωρινών directories (/temp/) για την καταγραφή αποτελεσμάτων και πληροφοριών των πειραμάτων. Βασίζεται στην Python 3 για την εκτέλεση των πειραματικών διαδικασιών, συνδέει τα αρχεία τοπολογιών με τα πειράματα από την γραμμή εντολών του terminal και προτείνει τη χρήση του ενσωματωμένου εργαλείου ./waf για την εγκατάσταση και μεταγλώττιση εξωτερικών προγραμμάτων. Στη συγκεκριμένη εργασία θα αναπτυχθούν κάποια εξωτερικά προγράμματα τα οποία, γραμμένα σε γλώσσα C++, θα προσαρμοστούν στους κόμβους του δικτύου για να καθορίσουν

τον ρόλο τους και τις ιδιαίτερες λειτουργίες τους. Παρόμοια τέτοια προγράμματα είναι τα NFD και NLSR που τρέχουν στους κόμβους από προεπιλογή.

2.2.2 NDN Forwarding Daemon

Το NDN Forwarding Daemon (NFD) είναι το πιο διαδεδομένο λογισμικό για router σε NDN δίκτυα. Περιέχει δομές όπως τα PIT, CS και FIB, είναι υπεύθυνο για τις στρατηγικές δρομολόγησης και διαχειρίζεται τη δημιουργία των faces μεταξύ των κόμβων. Προσφέρει μεγάλο εύρος πειραμάτων και είναι εύκολο στη χρήση. Μάλιστα το NFD έχει σαν στόχο η δομή του να προσφέρει πολλές δυνατότητες για τροποποιήσεις και βελτιστοποιήσεις όπως για παράδειγμα την ανάπτυξη πολιτικών cache εσωτερικά του δικτύου, την χρήση σε δίκτυα με κινητικότητα, την αναπτυγμένη ανταλλαγή πακέτων και την βελτίωση της απόδοσης του NDN μέσα από στοχευμένη εξαγωγή δεδομένων από το σύστημα [30].

Για τη χρήση του NFD ευκολότερα, όταν απαιτούνται απλές και γρήγορες ενέργειες στο σύστημα, έχει αναπτυχθεί ένα εργαλείο που είναι συμβατό με τα περισσότερα προγράμματα μοντελοποίησης NDN, το nfdc. Στο mini-NDN μπορεί να γίνει κλήση του συγκεκριμένου εργαλείου από τη γραμμή εντολών και ο ρόλος του είναι να διαχειρίζεται και να εξάγει δεδομένα από τα FIB, RIB και τις ενεργές στρατηγικές δρομολόγησης.

```
mini-ndn> midnoda nfdc status
General NFD status:
                version=0.7.1-commit-cabd980
              startTime=20210902T124811.297000
            currentTime=20210902T130319.701000
                 uptime=908 seconds
       nNameTreeEntries=281
            nFibEntries=69
            nPitEntries=3
   nMeasurementsEntries=0
             nCsEntries=512
          nInInterests=1021
         nOutInterests=1108
               nInData=586
              nOutData=776
              nInNacks=28
             nOutNacks=23
    nSatisfiedInterests=601
  nUnsatisfiedInterests=31
```

Εικόνα 2.1: Πληροφορίες για την κατάσταση του NFD

Στην Εικόνα 2.2 καλείται ο πίνακας που περιέχει την στρατηγική δρομολόγησης σε κάθε κόμβο του δικτύου. Είναι δυνατόν διαφορετικοί κόμβοι να τρέχουν διαφορετικές στρατηγικές δρομολόγησης. Παρατηρούμε πως στο συγκεκριμένο δίκτυο όλοι οι κόμβοι τρέχουν τον αλγόριθμο best-route που αναφέρεται στον υπολογισμό μίας βέλτιστης διαδρομής για την προώθηση των πακέτων ενδιαφέροντος. Οι στρατηγικές δρομολόγησης αφορούν κυρίως τα Interest Packets καθώς εάν υπάρξει απόκριση από το δίκτυο, τα πακέτα δεδομένων θα ακολουθήσουν την διαδρομή που τους σηματοδοτούν οι PIT καταχωρήσεις.

```
mini-ndn> midnoda nfdc strategy list
prefix=/ strategy=/localhost/nfd/strategy/best-route/%FD%05
prefix=/ndn/midnodh-site/KEY strategy=/localhost/nfd/strategy/best-route/%FD%05
prefix=/ndn/midnodk-site/KEY strategy=/localhost/nfd/strategy/best-route/%FD%05
prefix=/ndn/midnodi-site/%C1.Operator strategy=/localhost/nfd/strategy/best-route/%FD%05
prefix=/ndn/midnodj-site/%C1.Operator strategy=/localhost/nfd/strategy/best-route/%FD%05
prefix=/ndn/produceri-site/%C1.Router/cs/produceri/KEY strategy=/localhost/nfd/strategy/best-route/%FD%05
prefix=/ndn/midnodc-site/%C1.Router/cs/midnodc/nlsr/KEY strategy=/localhost/nfd/strategy/best-route/%FD%05
prefix=/ndn/producerb-site/%C1.Router/cs/producerb/nlsr/KEY strategy=/localhost/nfd/strategy/best-route/%FD%05
prefix=/ndn/midnode-site/%C1.Router/cs/midnode/nlsr/KEY strategy=/localhost/nfd/strategy/best-route/%FD%05
prefix=/ndn/midnodf-site/%C1.Router/cs/midnodf/KEY strategy=/localhost/nfd/strategy/best-route/%FD%05
prefix=/ndn/midnode-site/KEY strategy=/localhost/nfd/strategy/best-route/%FD%05
prefix=/ndn/midnodn-site/%C1.Router/cs/midnodn/nlsr/KEY strategy=/localhost/nfd/strategy/best-route/%FD%05
prefix=/ndn/midnodk-site/%C1.Router/cs/midnodk/KEY strategy=/localhost/nfd/strategy/best-route/%FD%05
prefix=/ndn/midnodb-site/%C1.Router/cs/midnodb/nlsr/KEY strategy=/localhost/nfd/strategy/best-route/%FD%05
prefix=/ndn/midnodi-site/KEY strategy=/localhost/nfd/strategy/best-route/%FD%05
prefix=/ndn/topnod-site/%C1.Router/cs/topnod/KEY strategy=/localhost/nfd/strategy/best-route/%FD%05
prefix=/ndn/midnodj-site/%C1.Router/cs/midnodj/KEY strategy=/localhost/nfd/strategy/best-route/%FD%05
prefix=/ndn/midnodl-site/%C1.Router/cs/midnodl/nlsr/KEY strategy=/localhost/nfd/strategy/best-route/%FD%05
```

Εικόνα 2.2: Πίνακας των στρατηγικών δρομολόγησης κάθε κόμβου όπως απεικονίζεται από το nfdc

Στην Εικόνα 2.3 καλείται ο πίνακας FIB ενός κόμβου μιας τοπολογίας δικτύου. Εκεί παρατηρούμε πως είναι καταχωρημένα τα βάρη των επιλεγμένων διαδρομών προς όλους τους κόμβους μαζί με το υπολογισμένο επόμενο άλμα για τον κάθε προορισμό. Ταυτόχρονα στο FIB περιέχονται τα διαφημισμένα ονόματα πακέτων στο δίκτυο (π.χ. */example/testApp8*) μαζί με το αντίστοιχο βάρος και επόμενο άλμα που τα συνοδεύουν. Ένα μειονέκτημα του NDN μοντέλου είναι η ραγδαία μεγέθυνση του υπολογιστικού φόρτου στο FIB όταν οι κόμβοι ενός δικτύου αυξάνονται. Για τον λόγο αυτό γίνεται μεγάλη έρευνα σε νέους “multipath” αλγορίθμους δρομολόγησης για την μείωση αυτού του υπολογιστικού φόρτου.

```
mini-ndn> midnoda nfdc fib
FIB:
  /ndn/producerg-site/%C1.Router/cs/producerg nexthops={faceid=267 (cost=100)}
  /ndn/midnodl-site/midnodl nexthops={faceid=267 (cost=80)}
  /example/testApp8 nexthops={faceid=267 (cost=100)}
  /ndn/producerd-site/producerd nexthops={faceid=271 (cost=60)}
  /localhost/nfd/rib nexthops={faceid=259 (cost=0)}
  /ndn/midnodg-site/%C1.Router/cs/midnodg nexthops={faceid=269 (cost=40)}
  /ndn/midnodi-site/midnodi nexthops={faceid=271 (cost=40)}
  /ndn/producerf-site/%C1.Router/cs/producerf nexthops={faceid=267 (cost=100)}
  /ndn/midnodc-site/midnodc nexthops={faceid=269 (cost=20)}
  /ndn/midnoda-site/%C1.Router/cs/midnoda/nlsr/KEY nexthops={faceid=272 (cost=0)}
  /ndn/producera-site/%C1.Router/cs/producera nexthops={faceid=269 (cost=60)}
  /example/testApp2 nexthops={faceid=269 (cost=60)}
  /ndn/midnodb-site/midnodb nexthops={faceid=267 (cost=40)}
  /ndn/midnodn-site/midnodn nexthops={faceid=267 (cost=80)}
  /ndn/producerh-site/%C1.Router/cs/producerh nexthops={faceid=267 (cost=100)}
  /ndn/midnode-site/midnode nexthops={faceid=267 (cost=60)}
  /localhop/ndn/nlsr/sync/%FD%08 nexthops={faceid=272 (cost=0), faceid=269 (cost=20), faceid=271 (cost=20), faceid=267 (cost=20)}
  /ndn/producerb-site/producerb nexthops={faceid=269 (cost=60)}
  /ndn/producerd-site/%C1.Router/cs/producerd nexthops={faceid=271 (cost=60)}
```

Εικόνα 2.3: Πίνακας FIB όπως απεικονίζεται από το nfdc

2.2.3 NLSR και NDN Routing Helper

Το λογισμικό του NLSR πρωτοκόλλου στοχεύει στη διαδικασία υπολογισμού των καταχωρήσεων των FIBs των κόμβων. Διαχειρίζεται τα LSAs για την αναγνώριση της τοπολογίας του δικτύου και τη σωστή ενημέρωση των κόμβων για αυτή. Διαθέτει επίσης ενσωματωμένη λειτουργία προκειμένου να αναγνωρίζει τους γειτονικούς κόμβους. Εάν ένας κόμβος διαθέτει κάποια δεδομένα και ένας γειτονικός κόμβος διαδώσει στο δίκτυο πακέτο ενδιαφέροντος για αυτά, τότε υπάρχει η δυνατότητα να του επιστρέψουν τα επιθυμητά πακέτα δεδομένων ακόμα και αν αυτά δεν έχουν διαφημιστεί.

Αντίστοιχα με το NFD, το NLSR διαθέτει επίσης το nfdc, ένα εργαλείο για την γρήγορη προσπέλαση της LSA βάσης δεδομένων (LSDB) και την διαφήμιση ή απόσυρση ονομάτων δεδομένων από έναν συγκεκριμένο κόμβο. Φυσικά το nfdc ακολουθεί τις λειτουργίες ασφαλείας που διέπουν τα NDN δίκτυα και για αυτό τον λόγο δεν μπορεί να προβεί σε καμία ενέργεια διαφήμισης ή απόσυρσης ονόματος εάν δεν υπάρχει η κατάλληλη πιστοποίηση αυθεντικότητας. Η συγκεκριμένη λειτουργία μπορεί να απενεργοποιηθεί σε δίκτυα στα οποία ο στόχος είναι απλά ο πειραματισμός με τις λειτουργίες του NDN.

Στην Εικόνα 2.4 καλείται η δομή δεδομένων στην οποία είναι αποθηκευμένα τα LSAs του δικτύου. Η κάθε καταχώρηση αφορά συγκεκριμένο κόμβο του δικτύου και φέρει το Name LSA που αναφέρεται στο όνομα του συγκεκριμένου router και το Adjacency LSA που ενημερώνει το δίκτυο για τους γειτονικούς κόμβους του router που αναγράφεται στο Name LSA και το βάρος των διαδρομών προς αυτούς. Παράλληλα διαθέτει ενσωματωμένη πληροφορία για το χρονικό όριο του κάθε LSA έτσι ώστε να γίνεται έλεγχος για αλλαγές στο δίκτυο ανά τακτά χρονικά διαστήματα.

```
mini-ndn> midnoda nlsrc lsdb
LSDB:
  OriginRouter: /ndn/topnod-site/%C1.Router/cs/topnod

    ADJACENCY LSA:
      Origin Router      : /ndn/topnod-site/%C1.Router/cs/topnod
      Sequence Number    : 1
      Expires in         : 3395132 milliseconds
      Adjacent(s):
        Adjacent 0: (name=/ndn/midnoda-site/%C1.Router/cs/midnoda, uri=udp4://10.0.0.2:6363, cost=20)
        Adjacent 1: (name=/ndn/midnodb-site/%C1.Router/cs/midnodb, uri=udp4://10.0.0.14:6363, cost=20)

    NAME LSA:
      Origin Router      : /ndn/topnod-site/%C1.Router/cs/topnod
      Sequence Number    : 1
      Expires in         : 3383119 milliseconds
      Names:
        Name 0: /ndn/topnod-site/topnod

  OriginRouter: /ndn/midnoda-site/%C1.Router/cs/midnoda

    ADJACENCY LSA:
      Origin Router      : /ndn/midnoda-site/%C1.Router/cs/midnoda
      Sequence Number    : 14
      Expires in         : 3395123 milliseconds
      Adjacent(s):
        Adjacent 0: (name=/ndn/topnod-site/%C1.Router/cs/topnod, uri=udp4://10.0.0.1:6363, cost=20)
        Adjacent 1: (name=/ndn/midnodc-site/%C1.Router/cs/midnodc, uri=udp4://10.0.0.6:6363, cost=20)
        Adjacent 2: (name=/ndn/midnodd-site/%C1.Router/cs/midnodd, uri=udp4://10.0.0.10:6363, cost=20)

    NAME LSA:
      Origin Router      : /ndn/midnoda-site/%C1.Router/cs/midnoda
      Sequence Number    : 12
      Expires in         : 3345116 milliseconds
      Names:
        Name 0: /ndn/midnoda-site/midnoda
```

Εικόνα 2.4: Πίνακας LSDB του nlsrc

Στην Εικόνα 2.5 αναγράφονται όλοι οι κόμβοι προορισμοί από έναν κόμβο σημείο μαζί με το ελάχιστο βάρος διαδρομής που όπως προαναφέρθηκε υπολογίζεται συνήθως από τον αλγόριθμο Dijkstra.

```
mini-ndn> midnoda nlsrc routing
Routing Table:
  Destination: /ndn/produceri-site/%C1.Router/cs/produceri
    NextHop(Uri: udp4://10.0.0.1:6363, Cost: 60)
  Destination: /ndn/midnodb-site/%C1.Router/cs/midnodb
    NextHop(Uri: udp4://10.0.0.1:6363, Cost: 40)
  Destination: /ndn/producerf-site/%C1.Router/cs/producerf
    NextHop(Uri: udp4://10.0.0.1:6363, Cost: 100)
  Destination: /ndn/midnodl-site/%C1.Router/cs/midnodl
    NextHop(Uri: udp4://10.0.0.1:6363, Cost: 80)
  Destination: /ndn/producerg-site/%C1.Router/cs/producerg
    NextHop(Uri: udp4://10.0.0.1:6363, Cost: 100)
  Destination: /ndn/midnodm-site/%C1.Router/cs/midnodm
    NextHop(Uri: udp4://10.0.0.1:6363, Cost: 80)
  Destination: /ndn/producerh-site/%C1.Router/cs/producerh
    NextHop(Uri: udp4://10.0.0.1:6363, Cost: 100)
  Destination: /ndn/midnodn-site/%C1.Router/cs/midnodn
    NextHop(Uri: udp4://10.0.0.1:6363, Cost: 80)
  Destination: /ndn/producere-site/%C1.Router/cs/producere
    NextHop(Uri: udp4://10.0.0.1:6363, Cost: 100)
  Destination: /ndn/midnodk-site/%C1.Router/cs/midnodk
    NextHop(Uri: udp4://10.0.0.1:6363, Cost: 80)
```

Εικόνα 2.5: Πίνακας Routing Table του nlsrc

Για την δρομολόγηση του δικτύου το miniNDN χρησιμοποιεί από προεπιλογή το NLSR το οποίο όμως μπορεί να αντικατασταθεί από την δεύτερη προεπιλογή, NDN Routing Helper. Ο συγκεκριμένος αλγόριθμος χρησιμοποιείται για να προσδώσουμε IP διευθύνσεις στους κατά τα άλλα NDN κόμβους, έτσι ώστε να πραγματοποιηθούν προσομοιώσεις στις οποίες συνυπάρχουν τα δύο μοντέλα με το NDN να αποτελεί τη βάση. Οι IP διευθύνσεις έχουν τη δυνατότητα να επιτυγχάνουν αλληλεπίδραση και των μη γειτονικών κόμβων μέσω IP routes χωρίς να απαιτείται το advertising του περιεχομένου τους. Φυσικά οι δύο αλγόριθμοι δρομολόγησης είναι αμοιβαία αποκλειόμενοι καθώς περιγράφουν διαφορετικά σενάρια δικτύων. Η λειτουργία του Routing Helper είναι να υπολογίζει τις διαδρομές της τοπολογίας και να τις αποθηκεύει στα FIB των κόμβων. Αυτή η λειτουργία ενδείκνυται για μεγαλύτερες τοπολογίες όπου δεν θέλουμε να υπάρχει η κεφαλίδα του NLSR στα πακέτα που αποστέλλονται και έτσι αποφεύγονται πιθανές καθυστερήσεις στο δίκτυο.

2.3 Μοντελοποίηση Δικτύων

Στο παρακάτω κεφάλαιο θα αναλύσουμε τα στάδια μοντελοποίησης των δικτύων με τα οποία θα ασχοληθούμε. Βασισμένα στο mini-NDN και στα υπόλοιπα προγράμματα που χρησιμοποιούνται θα αναλυθούν οι διάφορες εκδοχές των τοπολογιών, των ρόλων του καταναλωτή, παραγωγού και router που δίνονται σε συγκεκριμένους κόμβους και των λειτουργιών του πειράματος που θα μελετήσουμε.

2.3.1 Τοπολογίες πειράματος

Για τη διεξαγωγή των πειραμάτων θα χρησιμοποιήσουμε τις παρακάτω δύο τοπολογίες που η καθεμία βασίζεται σε διαφορετική οικογένεια τοπολογιών δικτύων. Όλες οι παρακάτω τοπολογίες με τις παραλλαγές τους έχουν γραφεί με συγκεκριμένο τρόπο ως configuration files τα οποία δέχεται το mini-NDN και πάνω στα οποία δημιουργεί τους εικονικούς κόμβους και τις λειτουργίες των πειραμάτων. Σε αυτά τα αρχεία ορίζονται οι ονομασίες όλων των κόμβων ξεχωριστά και οι μεταξύ τους συνδέσεις. Φυσικά ορίζονται επίσης και παράμετροι όπως η καθυστέρηση του κάθε link, το bandwidth, το ποσοστό των πακέτων που χάνονται κατά τη μετάδοση, καθώς και ρυθμίσεις σχετικά με τη δημιουργία log αρχείων για τα προγράμματα NFD και NLSR που τρέχουν στους κόμβους.

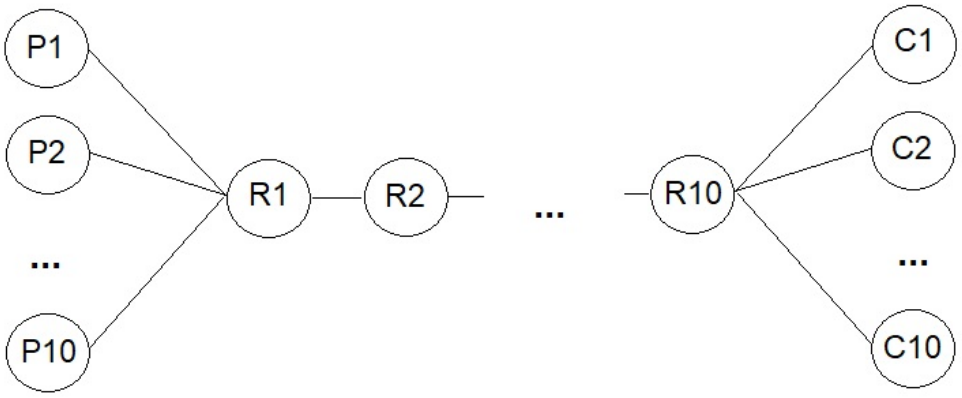
```
[nodes]
a: _
b: _
c: _
d: _
[links]
a:b delay=30ms
a:c delay=30ms
b:d delay=30ms
```

2.3.1.1 Dumbbell Topology

Η πρώτη τοπολογία είναι της λογικής dumbbell. Τα dumbbell δίκτυα αποτελούνται κατά κανόνα από καταναλωτές και παραγωγούς στα άκρα της τοπολογίας οι οποίοι συνδέονται μεταξύ τους με μία σειρά από δρομολογητές που εισάγουν συνήθως την μεγαλύτερη πηγή καθυστέρησης στο δίκτυο. Λόγω του σχήματος του δικτύου αλλά και της συγκεκριμένης συχνά εμφανιζόμενης καθυστέρησης, οι ενδιάμεσοι κόμβοι του δικτύου είναι αλλιώς γνωστοί και ως “bottleneck”.

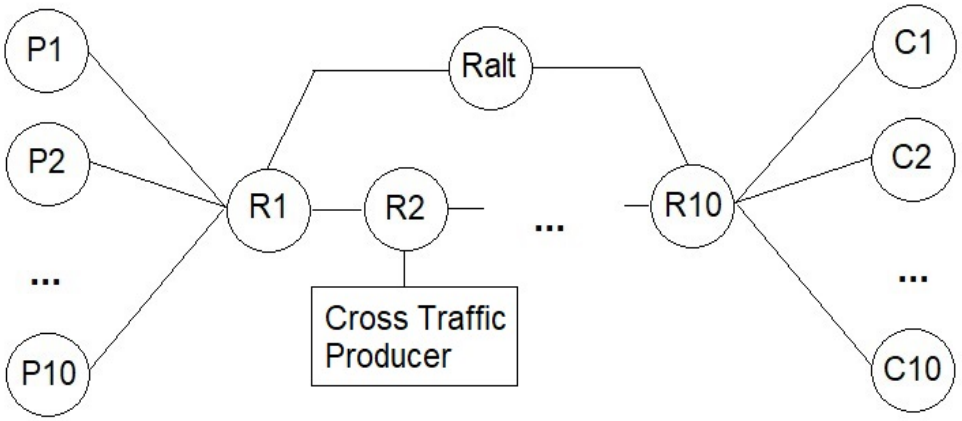
Η βασική λογική της πρώτης τοπολογίας είναι να δημιουργήσουμε δέκα καταναλωτές, δέκα παραγωγούς και δέκα ενδιάμεσους δρομολογητές όπως στην Εικόνα 2.6. Προκειμένου ένας καταναλωτής να δεχθεί ένα ζητούμενο πακέτο πρέπει πρώτα να αποστείλει ένα πακέτο ενδιαφέροντος που θα μεταδοθεί μέσα από όλους τους δρομολογητές στους παραγωγούς, και όταν το ενδιαφέρον εκδηλωθεί σε έναν παραγωγό που διαθέτει το ζητούμενο πακέτο, αυτό θα αποσταλεί ακολουθώντας την ακριβώς αντίστροφη διαδρομή.

Έτσι, εάν ο καταναλωτής C_x ζητήσει ένα πακέτο που διαθέτει ο παραγωγός P_y τότε η χρονική καθυστέρηση μέχρι να λάβει το πακέτο θα ισούται με δύο φορές την καθυστέρηση της διαδρομής C_x -R_{10} - R_9 - … - R_1 – P_y. Αυτό το γεγονός καθιστά εύκολη την πρόβλεψη καθυστέρησης και τον έλεγχο σφαλμάτων στο δίκτυο.



Εικόνα 2.6: Τοπολογία Dumbbell με ίσο αριθμό παραγωγών, καταναλωτών και δρομολογητών

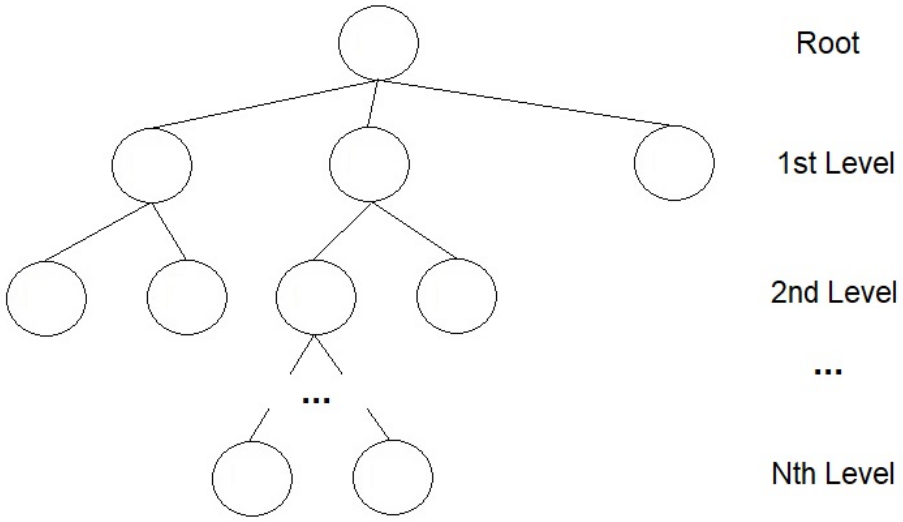
Στην παρούσα τοπολογία έχουμε προσαρμόσει ορισμένες παραλλαγές για την καλύτερη εκτέλεση των πειραμάτων και τη σαφέστερη παρουσίαση παραμέτρων και λοιπών ρυθμίσεων. Προσαρμόσαμε μία εναλλακτική ενδιάμεση διαδρομή με βάρος αρκετά μεγαλύτερο του συνολικού βάρους των υπολοίπων συνδέσεων των δρομολογητών έτσι ώστε να ελέγξουμε το δίκτυο σε συνθήκες για παράδειγμα κατάρρευσης της σύνδεσης των ενδιάμεσων κόμβων. Υπό άλλες συνθήκες σε ένα απλό dumbbell δίκτυο θα κατέρρεε η μοναδική διαδρομή των ενδιαφερόντων και πακέτων δεδομένων. Σε συνδυασμό με αυτή την αλλαγή, αναθέσαμε τυχαία σε ενδιάμεσους δρομολογητές, προγράμματα cross-traffic producer, που σημαίνει ότι λειτουργούν ταυτόχρονα ως δρομολογητές και παραγωγοί πακέτων.



Εικόνα 2.7: Τοπολογία Dumbbell με προσαρμοσμένο εναλλακτικό μονοπάτι και ενδιάμεσους παραγωγούς

2.3.1.2 Τοπολογία Δέντρου (Tree) ή Star-Bus

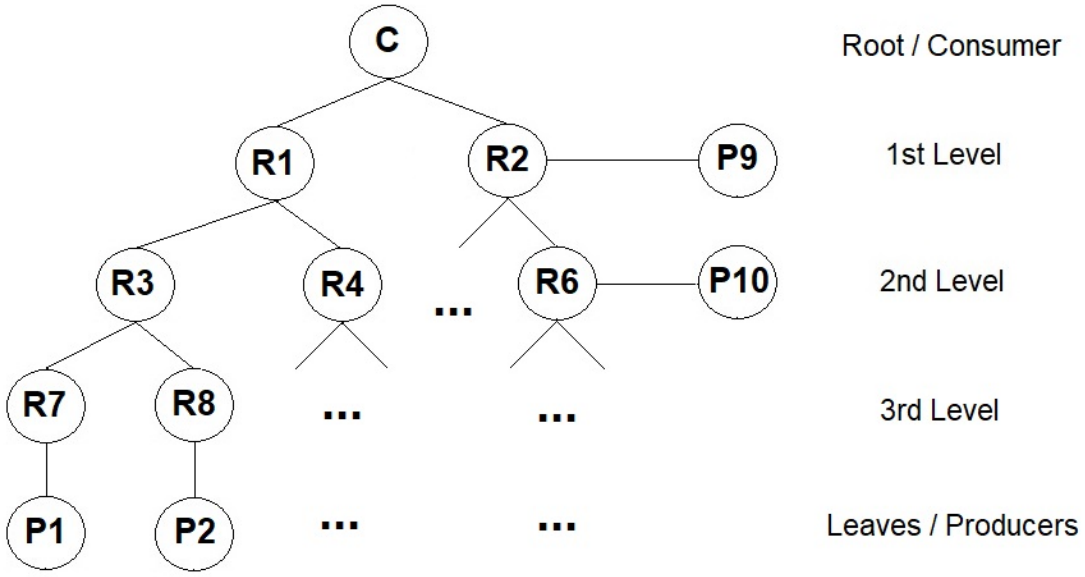
Η συγκεκριμένη τοπολογία πήρε την ονομασία της από το διάγραμμα των διασυνδεδεμένων κόμβων τα οποία ομοιάζουν με τα φύλλα, τα κλαδιά και τον κορμό ενός δέντρου. Δύο κόμβοι μπορούν να έχουν μόνο μία σύνδεση μεταξύ τους και έτσι σχηματίζεται φυσικά μία σχέση parent-child μεταξύ των κόμβων ή γενικότερα οι κόμβοι κατηγοριοποιούνται σε επίπεδα ανάλογα με το πόσα άλματα απέχουν από τη “ρίζα” της τοπολογίας. Η ρίζα μπορεί να ονομαστεί και Bus και είναι ο κόμβος που συνδέει μεταξύ τους τα μικρότερα υποδίκτυα που ορίζονται στα κλαδιά ή αλλιώς στους αστέρες.



Εικόνα 2.8: Γενική τοπολογία Δέντρου ή Star-Bus

Ένα μεγάλο πλεονέκτημα του συγκεκριμένου δικτύου εμφανίζεται σε καταστάσεις διακοπτόμενης σύνδεσης, όπου το σύστημα πλην του τμήματος που αποκόπτεται λόγω της πτώσης σύνδεσης, παραμένει ανεπηρέαστο. Αντίθετα, σε τοπολογίες dumbbell, μια πτώση σύνδεσης σε μόνο έναν ενδιάμεσο κόμβο σημαίνει κατάρρευση του δικτύου. Διαθέτει επίσης οργανωμένη ιεραρχική δομή στους κόμβους οι οποίοι μπορεί οι ίδιοι να συστήνουν στο δίκτυο ένα ή περισσότερα μικρότερα υποδίκτυα, οπότε δίνεται η δυνατότητα να βρεθεί γρήγορα και εύκολα κάποιο εν δυνάμει σφάλμα.

Γενικά οι κόμβοι δεν διαθέτουν συνήθως σταθερό αριθμό από θυγατρικά υποδίκτυα σε όλο το δίκτυο. Στη δικιά μας τοπολογία, ανεξάρτητα από τις τερματικές συσκευές που μπορεί να βρίσκονται συνδεδεμένες σε ενδιάμεσους κόμβους στο δίκτυο, κάθε μη τερματικός κόμβος διαθέτει σταθερά δύο θυγατρικές συνδέσεις προς άλλους routers. Μόνο στο τελευταίο, τρίτο επίπεδο δρομολογητών οι θυγατρικές σχέσεις αποτελούν μοναδικές συνδέσεις με τις τερματικές εφαρμογές των καταναλωτών. Ταυτόχρονα, για να εμπλουτίσουμε τα σενάρια των πειραμάτων μας συνδέουμε από έναν παραγωγό και σε κάθε ένα από τα παραπάνω ιεραρχικά επίπεδα κόμβων. Επιδιώκουμε το δίκτυό μας να είναι συμμετρικό όσον αφορά τις διακλαδώσεις και τα βάρη των διαδρομών έτσι ώστε η απόδοση του δικτύου να μην εξαρτάται από τα ονόματα των πακέτων που θα ζητούνται τυχαία. Τέλος, οι παραγωγοί σε αυτή τη τοπολογία παραμένουν σταθεροί σε αριθμό, οπότε η εναλλαγή αυτών θα γίνει μόνο σε τοπολογία dumbbell.



Εικόνα 2.9: Τοπολογία Δέντρου ή Star-Bus με ορισμένους ρόλους καταναλωτή και παραγωγών

2.3.2 Λειτουργίες Κόμβων

Σε αυτό το κεφάλαιο θα αναλυθούν οι λειτουργίες που εκτελούνται σε κάθε έναν κόμβο που έχει οριστεί ως παραγωγός, καταναλωτής ή “router” (εδώ η χρήση της λέξης router θα χρησιμοποιείται για να περιγράψει τους NDN routers με δυνατότητα in-network caching). Το mini-NDN αποτελεί ένα πρόγραμμα που λειτουργεί συνδέοντας βιβλιοθήκες και προγράμματα NDN σε σταθερές μονάδες κόμβων. Αυτό σημαίνει ότι λαμβάνοντας ένα .conf αρχείο τοπολογίας δημιουργεί τις συνδέσεις και τους κόμβους που θα χρησιμοποιηθούν, αλλά προκειμένου να προκαλέσει κάποια αλληλεπίδραση μεταξύ τους πρέπει να τρέξει διαφορετικά προγράμματα σε κάθε κόμβο. Τα προγράμματα που απαιτούνται για την προσομοίωση της συμπεριφοράς ενός απλού NDN δικτύου είναι τα NLSR και NFD για τη δρομολόγηση των πακέτων, αλλά προκειμένου να δημιουργήσουμε πιο εξελιγμένα και εξατομικευμένα πειράματα είναι αναγκαία η φόρτωση των κόμβων με επιπλέον λειτουργίες. Κάθε κόμβος που φέρει συγκεκριμένο ρόλο τρέχει και το δικό του ειδικά σχεδιασμένο πρόγραμμα. Η γλώσσα που χρησιμοποιείται στις βιβλιοθήκες NDN είναι η C++ και απαιτείται η

δημιουργία executables πριν τη χρήση τους, σε αντίθεση με τα προγράμματα πειράματος mini-NDN όπως θα δούμε στη συνέχεια.

2.3.2.1 Καταναλωτές

Όπως έχει ήδη αναλυθεί, στα NDN δίκτυα ένας καταναλωτής στέλνει πακέτα ενδιαφέροντος που φέρουν το όνομα του ζητούμενου πακέτου δεδομένων, και όταν βρεθεί παραγωγός με το συγκεκριμένο πακέτο στη μνήμη του, το αποστέλλει στην αντίστροφη διαδρομή. Προκειμένου να ορίσουμε την οντότητα ενός πακέτου ενδιαφέροντος δημιουργούμε ένα αντικείμενο Interest στο οποίο ορίζουμε παραμέτρους όπως το όνομα ενδιαφέροντος και τον χρόνο ζωής του (Interest Life-time). Στη συνέχεια με σκοπό την αποστολή του, δημιουργούμε ένα face που συνδέει το συγκεκριμένο executable με την υπόλοιπη εφαρμογή και πλέον ο καταναλωτής είναι ελεύθερος να εκφράσει το ενδιαφέρον του. Από τη στιγμή που ένα Interest αποσταλεί, το πρόγραμμα αναμένει μία εκ των τριών παρακάτω πιθανών αποκρίσεων.

1) Data Packet:

Όταν υπάρχει καταναλωτής που λάβει το ενδιαφέρον και στείλει πίσω το πακέτο δεδομένων του τότε η διαδικασία έχει πραγματοποιηθεί επιτυχώς και η απόκριση του δικτύου θα είναι το ίδιο το ζητούμενο πακέτο. Σε αυτή τη περίπτωση μπορούμε να επίσης να εξάγουμε πολλές πληροφορίες για το ίδιο το δίκτυο και τη πορεία δρομολόγησης του Interest.

2) Negative Acknowledgement (NACK):

Το NACK είναι ένα διαφορετικό είδος πακέτου δεδομένων που ειδοποιεί τον καταναλωτή ότι το ζητούμενο περιεχόμενο δεν υπάρχει σε κάποιο καταχωρημένο μονοπάτι. Περιλαμβάνει το όνομα του ίδιου του NACK, το όνομα του ζητούμενου πακέτου, την κρυπτογραφημένη υπογραφή και τμήμα κώδικα που αναλύει τον λόγο που δεν επέστρεψε το επιθυμητό περιεχόμενο. Συνήθεις λόγοι για να επιστρέψει ένα NACK είναι δημιουργία βρόγχου, μη καταχωρημένη διαδρομή προώθησης και συμφόρηση δικτύου.

Στην περίπτωση επιστροφής ενός NACK συμπεραίνουμε ότι υπήρξε κάποιο σφάλμα στον σχεδιασμό του δικτύου, οπότε είναι μια χρήσιμη λειτουργία για τη διόρθωση του κώδικα των πειραμάτων. Μία περίπτωση στην οποία θα είχαμε NACK σαν απόκριση του δικτύου θα ήταν άμα ο

καταναλωτής ζητούσε ένα πακέτο το οποίο δεν έχει προηγουμένως διαφημιστεί από κάποιον παραγωγό και άρα δεν υπάρχουν καταχωρήσεις στα FIB των routers προς αυτό.

3) Timeout

Στην τελευταία περίπτωση δεν έχουμε δεχθεί NACK σαν απόκριση οπότε το Interest έχει ακολουθήσει την πορεία που του υποδεικνύει η στρατηγική δρομολόγησης αλλά ο χρόνος αναμονής του καταναλωτή ξεπερνάει το Interest Lifetime και έτσι σημαίνει Timeout. Κάποιοι πιθανοί λόγοι για να έχουμε απόκριση Timeout είναι η συνολική καθυστέρηση των συνδέσεων στη διαδρομή του Interest να ξεπερνάει το Interest Lifetime ή το Interest να βρεθεί σε κόμβο του οποίου έχει καταρρεύσει η σύνδεση με τον επόμενο πριν προλάβει να ενημερωθεί το υπόλοιπο δίκτυο και συνεπώς τα FIBs.

```
Name interestName("/example/testApp/randomData");
interestName.appendVersion();

Interest interest(interestName);
interest.setCanBePrefix(false);
interest.setMustBeFresh(true);
interest.setInterestLifetime(6_s); // The default is 4 seconds

std::cout << "Sending Interest " << interest << std::endl;
m_face.expressInterest(interest,
                       bind(&Consumer::onData, this,  _1, _2),
                       bind(&Consumer::onNack, this, _1, _2),
                       bind(&Consumer::onTimeout, this, _1));

// processEvents will block until the requested data is received or a timeout occurs
m_face.processEvents();
```

Στον παραπάνω βασικό αλγόριθμο έχουμε προσθέσει επιπλέον κώδικα έτσι ώστε να επιτύχουμε εξατομικευμένες λειτουργίες όπως ειδικές μετρήσεις, ρύθμιση της ποσότητας και των ονομάτων των προς αποστολή Interest packets και εισαγωγή Buffer για την προσωρινή αποθήκευση Data packets που έχουν ήδη καταφθάσει στον κόμβο.

2.3.2.2 RTT timer (Καθυστέρηση Πακέτου Δεδομένων)

Η προσομοίωση του δικτύου γίνεται με την διαδοχική εκτέλεση των εντολών που εισάγει ο κώδικας του πειράματος και των εκτελέσιμων που τρέχουν σε κάθε κόμβο. Έτσι, εφόσον το mini-NDN αποτελεί έναν emulator δικτύου, οι καθυστερήσεις που εισάγονται θεωρητικά από το δίκτυο δεν αποτελούν μέρος μίας μαθηματικής εξίσωσης για εξαγωγή μετρήσεων, αλλά πραγματοποιούνται

φυσικά, ως καθυστερήσεις στον κώδικα του NDN. Προκειμένου να υπολογίσουμε τον χρόνο που απαιτείται από τη στιγμή που ο καταναλωτής στέλνει ένα Interest μέχρι τη στιγμή που θα λάβει το ζητούμενο Data Packet χρησιμοποιήσαμε δύο μεταβλητές που οριοθετούν χρονικά τις δύο παραπάνω δραστηριότητες.

```
auto now = time::steady_clock::now();

std::cout << "Sending Interest " << interest << std::endl;
```

Καλώντας και καταγράφοντας το ρολόι του συστήματος με ακρίβεια νανοδευτερολέπτων στις δύο στιγμές μπορούμε να βρούμε τον συνολικό RTT χρόνο από τη διαφορά τους.

```
onData(const Interest&, const Data& data, const time::steady_clock::TimePoint& sendTime, std::string sendName) const
{
  time::nanoseconds rtt = time::steady_clock::now() - sendTime;
```

Πράγματι, το αποτέλεσμα ισούται με την καθυστέρηση που εισάγουν οι συνδέσεις κόμβων όπως ορίζεται από το αρχείο τοπολογίας με μία μικρή απόκλιση που οφείλεται κυρίως στον χρόνο που απαιτείται για να τρέξουν όλες οι εντολές στους κόμβους από όπου διαδίδονται τα πακέτα ενδιαφέροντος και δεδομένων. Μπορούμε έτσι να ονομάσουμε την ορισμένη στο αρχείο τοπολογίας καθυστέρηση του δικτύου ως Transmission Delay ενώ την καθυστέρηση που εισάγεται από την εκτέλεση των εντολών του προγράμματος, προσεγγιστικά ως Proccessing Delay καθώς περιλαμβάνει τη συνολική καθυστέρηση από την προσπέλαση των FIB των κόμβων και άλλες αντίστοιχες διαδικασίες δρομολόγησης.

2.3.2.3 Δημιουργία και προγραμματισμός αποστολής των Interests

Τα πειράματα που θα τρέξουμε στην παρούσα εργασία απαιτούν την αποστολή πολλών πακέτων ενδιαφέροντος από τον ίδιο καταναλωτή και φυσικά με διαφορετικά ονόματα καθώς θα ήταν άσκοπο ο ίδιος κόμβος να ζητάει τα ίδια πακέτα ξανά και ξανά. Για τον λόγο αυτό προσαρμόσαμε κώδικα ο οποίος δέχεται σαν σταθερά το σύνολο των ονομάτων για τα οποία μπορεί να εκφράσει ενδιαφέρον ο καταναλωτής και τον αριθμό των πακέτων που στέλνει στο υπόλοιπο δίκτυο. Μία συνάρτηση παραγωγής τυχαίων αριθμών έχει ως έξοδο έναν αριθμό ο οποίος προστίθεται στο τελευταίο τμήμα της κεφαλίδας του κάθε Interest και έτσι επιτυγχάνεται ο διαχωρισμός μεταξύ των ονομάτων των πακέτων.

Αργότερα, αυτή η συνάρτηση δεν θα παράγει απλώς τυχαίους ακεραίους, αλλά η συχνότητα εμφάνισής τους θα ακολουθεί συγκεκριμένη κατανομή προκειμένου να πειραματιστούμε με διάφορες πολιτικές caching. Δημιουργήσαμε έναν αλγόριθμο τυχαίας παραγωγής αριθμών που ακολουθεί μια γραμμική κατανομή πιθανότητας εμφάνισης ενδιαφερόντων με τα ενδιαφέροντα με μεγαλύτερο ID αριθμό να εμφανίζονται συχνότερα από αυτά με μικρότερο ID. Πιο συγκεκριμένα, δημιουργήθηκε μία δομή δεδομένων, που αποτελεί το πεδίο των πιθανών τιμών της τυχαίας συνάρτησης, στην οποία καταχωρήθηκαν με γραμμική κατανομή οι διαφορετικές ID των πακέτων ενδιαφέροντος. Η συγκεκριμένη λειτουργία μπορεί εύκολα να τεθεί ανενεργή και αντίστροφα μέσω μίας λογικής σταθεράς στην αρχή του προγράμματος.

```
int randDivider;
srand(clock());
int randNum;

/////////////// Produce the Linear Probability Distribution
if (linearProb){
        int counter = 0;
            for (int i=1; i<=packetsNum; i++){
                    for (int j=0; j<i; j++){
                            linearDist[counter] = i;
                            counter++;
                    }
            }
         randDivider = counter;
         randNum = linearDist[(rand() % randDivider)];
 }
 else {
         randDivider = packetsNum;
         randNum = (rand() % randDivider) + 1;
 }
```

Για την αποστολή περισσοτέρων από ένα Interests κάνουμε χρήση του εργαλείου scheduler. Η συγκεκριμένη συνάρτηση οργανώνει στον χρόνο τα events ενός τμήματος κώδικα και έτσι μπορεί να λειτουργεί ανεξάρτητα από το υπόλοιπο πείραμα. Με τη χρήση της χρονικής καθυστέρησης στην αποστολή πακέτων μας δίνεται η δυνατότητα να ελέγξουμε τα αποτελέσματα του δικτύου σε μεταβαλλόμενες συνθήκες, όπως για παράδειγμα σε διακοπτόμενη σύνδεση. Έχουμε ορίσει ότι τα πακέτα θα στέλνονται διαδοχικά με χρονική καθυστέρηση τριών δευτερολέπτων για να μην υπάρχουν πιθανά επικαλυπτόμενα Interests στο δίκτυο και για τον καλύτερο έλεγχο και μελέτη των αποκρίσεων του δικτύου. Εδώ προσθέτουμε ότι καθώς λειτουργεί η συνάρτηση rand() για την

παραγωγή τυχαίων Interests, δεν αποκλείεται το σενάριο της αποστολής της ίδιας κεφαλίδας σε δύο διαδοχικά πακέτα ενδιαφέροντος.

2.3.2.4 In-Network Caching

Το mini-NDN είναι στραμμένο στη μοντελοποίηση της συμπεριφοράς ενός δικτύου, στη δρομολόγηση και στην ικανότητα των κόμβων να επικοινωνούν μεταξύ τους. Για τον λόγο αυτό δομές όπως το Content Store και το Pending Interest Table για πιο εξειδικευμένα πειράματα δεν είναι επαρκώς ανεπτυγμένες, παρά μόνο σε μορφή καταχωρήσεων σε .log αρχεία. Ο στόχος μας είναι να δημιουργήσουμε στους καταναλωτές έναν ενσωματωμένο buffer-Content Store χρησιμοποιώντας μια δομή δεδομένων έτσι ώστε σε κάθε αποστολή Interest να σαρώνονται τα περιεχόμενά της. Στην περίπτωση επανειλημμένης αποστολής πακέτου ενδιαφέροντος με κεφαλίδα που ταυτίζεται με κάποια επικεφαλίδα στον buffer, ο αλγόριθμος δεν συνεχίζει στην αποστολή του Interest αλλά ικανοποιεί το ενδιαφέρον μέσα από το Data packet του buffer του κόμβου. Έχουμε θέσει ως παραμέτρους του προγράμματος το μέγεθος του buffer καθώς και τις διαφορετικές πολιτικές προσωρινής αποθήκευσης.

Μία πολιτική προσωρινής αποθήκευσης (caching policy) καθορίζει τους κανόνες που τηρούνται για να αποφασιστεί το πότε και το πως θα αντικαθίστανται τα δεδομένα που είναι αποθηκευμένα σε κάποια μνήμη cache. Είναι λογικό να απορρίπτονται τιμές σύντομα έτσι ώστε να μπορούν να αποθηκευτούν νέες τιμές αλλά παράλληλα είναι συχνά χρήσιμο να διατηρούνται παλαιότερες καταχωρήσεις καθώς είναι πιθανό να ζητηθούν σε μελλοντικά Interests. Οι διαφορετικές πολιτικές διαφέρουν στον τρόπο που διαχειρίζονται τα δύο αυτά αντίθετα σενάρια. Οι πολιτικές που ακολουθούν λογική opportunistic caching πρέπει να είναι προσαρμοστικές, λαμβάνοντας υπόψη το κόστος της διαδρομής που απαιτείται για την λήψη συγκεκριμένων πακέτων, την χωρητικότητα του buffer σε συνάρτηση με το μέγεθος των πακέτων και τη συχνότητα εμφάνισης Interests για συγκεκριμένα ονόματα. Ο τελευταίος παράγοντας θα αποτελέσει και τη βασική προτεραιότητα για την επιλογή βέλτιστης opportunistic caching πολιτικής στα πειράματά μας.

Στον αλγόριθμο που σχεδιάσαμε η επιλεγμένη πολιτική τίθεται σε λειτουργία μόνο όταν τα συνολικά εισερχόμενα πακέτα δεδομένων ξεπεράσουν σε αριθμό τις θέσεις μνήμης του buffer. Η λίστα με τις διαφορετικές πολιτικές που μπορούν να χρησιμοποιηθούν δίνεται παρακάτω:

Admit-off

Η συγκεκριμένη πολιτική αποτελεί μια τεχνική που αποτρέπει την είσοδο νέων πακέτων στον buffer από τη στιγμή που αυτός γεμίσει. Αποτελεί δηλαδή μία πολιτική απόρριψης η οποία διατηρεί τα πακέτα που μπήκαν πρώτα στον buffer αποθηκευμένα απ αορίστων. Είναι μία πολιτική με μηδενική ευελιξία ειδικά αν η ζήτηση των πακέτων δεν είναι τυχαία αλλά ακολουθεί κάποιον κανόνα.

First In First Out (FIFO)

Η συγκεκριμένη στρατηγική χρησιμοποιείται ευρέως και σε άλλους κλάδους πέρα από τα δίκτυα και είναι σχεδιασμένη πάνω στη λογική ότι όταν γεμίσει ο buffer, το πρώτο πακέτο που εισήλθε θα είναι το πρώτο πακέτο που θα απορριφθεί. Έτσι, εφόσον η αποστολή των Interest Packets στο πείραμά μας είναι προγραμματισμένη ανά τακτά χρονικά διαστήματα, κάθε πακέτο δεδομένων θα έχει και δεδομένο “χρόνο ζωής” μέσα στον buffer.

Random Replacement (RR)

Σε κάθε νέο πακέτο δεδομένων που επιστρέφει στον καταναλωτή και εισάγεται στην προσωρινή μνήμη, ένα άλλο πακέτο θα απορρίπτεται από αυτήν. Στη συγκεκριμένη πολιτική το πακέτο που θα απορριφθεί δεν έχει να κάνει με τη σειρά με την οποία εισήλθε όπως στη FIFO, αλλά καθορίζεται τελείως τυχαία από μία συνάρτηση rand(). Το νέο πακέτο που εισέρχεται στη μνήμη δεν είναι σίγουρο ότι θα μπει στη τελευταία θέση αυτής, αλλά σε όποια θέση υπήρχε προηγουμένως το τυχαία απορριφθέν πακέτο.

```
if (counterCS < csLength){
        csArray[counterCS] = sendName;
        counterCS ++;
}
else if (cachePolicy == "FIFO"){
        for (int i=0; i<csLength-1; i++){
                csArray[i] = csArray[i+1];
        }
        csArray[csLength-1] = sendName;
        std::cout << "test FIFO works"<< std::endl;
}
else if (cachePolicy == "RR"){
     int randPlace = (rand() % csLength);
     csArray[randPlace] = sendName;
     std::cout << "test RR works"<< std::endl;
}
else if (cachePolicy == "Static"){
    std::cout << "test Static works"<< std::endl;
}
 else if (cachePolicy == "LFU"){
```

Least Frequently Used (LFU)

Η συγκεκριμένη πολιτική έχει στοιχεία προσαρμοστικότητας σε καταστάσεις όταν κάποιες κεφαλίδες δεδομένων είναι πιο δημοφιλείς από άλλες. Υπολογίζει σε κάθε νέα άφιξη πακέτου τις πιο συνήθεις κεφαλίδες και κρατάει στο buffer τις πιο συχνά εμφανιζόμενες. Αυτό σημαίνει ότι ένα νέο πακέτο δεδομένων θα εισέλθει στη μνήμη cache μόνο εάν η συχνότητα εμφάνισής του ξεπέρασε τη συχνότητα εμφάνισης κάποιου άλλου πακέτου που είναι ήδη αποθηκευμένο εκεί. Σε αυτή τη περίπτωση το εν λόγο πακέτο θα διαγραφεί και το νέο πακέτο θα πάρει τη θέση του.

```
////// LFU Function, creation of the Max Vector
if (cachePolicy == "LFU"){
    int placebo;

    originalLFU[randNum-1]++;
    for (int i=0; i<10; i++){
            copyLFU[i] = originalLFU[i];
            namesLFU[i] = i;
    }

    for (int i=1; i<10; i++){
            for (int j=10; j>i-1; j--){
                    if (copyLFU[j-1]<copyLFU[j]){
                            placebo = copyLFU[j];
                            copyLFU[j] = copyLFU[j-1];
                            copyLFU[j-1] = placebo;
                            placebo = namesLFU[j];
                            namesLFU[j] = namesLFU[j-1];
                            namesLFU[j-1] = placebo;
                    }
            }
    }
    std::ostringstream oss;
    for (int i=0; i<csLength; i++){
            oss << "/example/testApp" << namesLFU[i]+1 << "/randomData";
            csArray[i] = oss.str();
    }
```

2.3.2.5 Δρομολογητές

Ένας ακόμα ιδιαίτερος ρόλος που πρέπει να αναθέσουμε σε κάποιους κόμβους είναι αυτός του δρομολογητή (router). Ο βασικός ρόλος του δρομολογητή είναι να υπολογίζει με τον βέλτιστο τρόπο το επόμενο άλμα για να φτάσουν τα διάφορα πακέτα στον προορισμό τους. Ο υπολογισμός και η προώθηση των πακέτων στα επόμενα άλματα είναι λειτουργίες που πραγματοποιούνται ήδη στους αλγορίθμους των NFD και NLSR. Παρόλα αυτά αυτή δεν είναι η μόνη τους λειτουργία. Στα NDN δίκτυα που έχουν εξελιχθεί πέρα από τον περιορισμό των IP διευθύνσεων, η προσοχή έχει μετακινηθεί στο ίδιο το περιεχόμενο του πακέτου και η ασφάλεια δικτύου έχει προσαρμοστεί στο

συγκεκριμένο μοντέλο και έτσι το in-network caching έχει πλέον τις απαραίτητες βάσεις για να αναπτυχθεί. Αυτό προσφέρει τη δυνατότητα της προσωρινής αποθήκευσης πακέτων σε ενδιάμεσους κόμβους του δικτύου που δεν είναι οι ίδιοι παραγωγοί, έτσι ώστε να είναι δυνατόν να ικανοποιήσουν μελλοντικά Interests με μεγαλύτερη απόδοση.

Στο συγκεκριμένο ρόλο θα κάνουμε την εξής παραδοχή. Θεωρούμε πως σε κάθε κόμβο που τρέχει το πρόγραμμα του router έχουν ήδη προηγηθεί διελεύσεις πακέτων ενδιαφέροντος και δεδομένων και έτσι με την εκκίνηση του πειράματος έχουμε ήδη ένα γεμάτο Content Store με τυχαία πακέτα δεδομένων. Επίσης, η πολιτική της προσωρινής αποθήκευσης ακολουθεί την admission-off, στατική πολιτική, δηλαδή πακέτα που δρομολογούνται στον κόμβο δεν έχουν κάποια επίδραση στα περιεχόμενα του buffer. Η συγκεκριμένη προσέγγιση έγινε προκειμένου να μειώσουμε τον αριθμό των παραγόντων που επιδρούν στα τελικά αποτελέσματα των πειραμάτων μας, και να επικεντρωθούμε στις αλλαγές των παραμέτρων στους καταναλωτές και παραγωγούς.

Όσον αφορά την υλοποίηση του προγράμματος, ο κάθε κόμβος router ανατρέχει στο Content Store στο οποίο είναι αποθηκευμένα τα τυχαία πακέτα που διαθέτει και προβάλλει στη σύνδεση του με τους γειτονικούς του κόμβους ένα φίλτρο ενδιαφέροντος για την κεφαλίδα του κάθε πακέτου. Έτσι, ενώ συνεχίζει να λειτουργεί σαν απλός δρομολογητής πακέτων, όταν λάβει Interest του οποίου η κεφαλίδα ταυτίζεται με κάποιο από τα φίλτρα ενδιαφέροντός του, αυτόματα στέλνει αντίγραφο του πακέτου που έχει αποθηκευμένο στην μνήμη προσωρινής αποθήκευσης πίσω στον καταναλωτή. Με αυτόν τον τρόπο φυσικά μειώνεται το RTT του Interest-Data packet ανάλογα με την απόσταση που απέχει ο router από τον καταναλωτή. Στο τρίτο κεφάλαιο θα γίνει πιο λεπτομερής μελέτη της συγκεκριμένης δυνατότητας των NDN δικτύων.

2.3.2.6 Παραγωγοί

Οι παραγωγοί σε ένα δίκτυο αποτελούν τον προορισμό των πακέτων ενδιαφέροντος και ορίζουν το σύνολο των πακέτων δεδομένων που υπάρχουν στο δίκτυο. Στην εργασία χρησιμοποιήσαμε συγκεκριμένο αριθμό παραγωγών έτσι ώστε να ταυτίζονται με τον αριθμό των διαφορετικών prefixes των Interests. Πιο συγκεκριμένα, ο κάθε παραγωγός “υπογράφει” τα εισερχόμενα πακέτα με μοναδικό όνομα και έτσι δεν μπορούν να προκύψουν ίδια ονοματισμένα πακέτα από διαφορετικούς παραγωγούς. Παρόμοια με τους routers, οι παραγωγοί θέτουν ένα φίλτρο ενδιαφέροντος για την ονομασία του πακέτου που υπογράφουν και στην περίπτωση που

δρομολογηθεί πακέτο ενδιαφέροντος σε αυτούς, αυτοί αποστέλλουν ένα αντίγραφό του προς την αντίστροφη κατεύθυνση. Στον συγκεκριμένο αλγόριθμο δεν κάναμε καμία προσθήκη πέρα από τη διαφοροποίηση των Interest Filter για να ταυτίζονται με το πακέτο που υπογράφουν.

```
class Producer
{
public:
  void
  run()
  {
    m_face.setInterestFilter("/example/testApp1",
                             bind(&Producer::onInterest, this, _1, _2),
                             nullptr, // RegisterPrefixSuccessCallback is optional
                             bind(&Producer::onRegisterFailed, this, _1, _2));
    m_face.processEvents();
  }
```

2.3.2.7 Ενσωμάτωση των εκτελέσιμων στις τοπολογίες

Εφόσον είναι έτοιμα τα εκτελέσιμα προγράμματα μαζί με τις τοπολογίες, πρέπει να ορίσουμε τους κόμβους στους οποίους θα τρέχουν τα συγκεκριμένα προγράμματα. Εάν δεν ορίσουμε σε κάποιον κόμβο κάποιο από τα παραπάνω προγράμματα με εξαίρεση τα NFD και NLSR τότε αυτός θα έχει τις βασικές λειτουργίες δρομολόγησης χωρίς τη δυνατότητα caching, παραγωγής ή ζήτησης πακέτου. Είναι μόνο λογικό τα προγράμματα producer και consumer να τρέξουν στους παραγωγούς και καταναλωτές αντίστοιχα, ενώ για το πρόγραμμα router ορίζουμε τυχαία ενδιάμεσους κόμβους-δρομολογητές προκειμένου να τους προσδώσουμε μία επιπλέον δυνατότητα in-network caching.

2.3.3 Ανάπτυξη Πειράματος

Έχοντας αναλύσει τα βασικά εργαλεία που θα χρησιμοποιήσουμε στα πειράματα μας, πρέπει να τα συνδέσουμε με τέτοιο τρόπο έτσι ώστε να είναι λειτουργικά μεταξύ τους και να εξάγουν τα επιθυμητά προς την έρευνά μας αποτελέσματα. Παρακάτω θα αναλύσουμε τον κεντρικό κώδικα των πειραμάτων, γραμμένο σε γλώσσα Python και του οποίου η λειτουργία είναι να ορίσει τα προγράμματα που θα τρέξει ο κάθε κόμβος, να ελέγξει τη σωστή λειτουργία του δικτύου και να οργανώσει χρονικά τις δραστηριότητες των κόμβων για το πείραμά μας.

Ξεκινώντας το πρόγραμμα του πειράματος, καλούμε τη συνάρτηση του Mini-ndn να δεχθεί τις αρχικές παραμέτρους του δικτύου μας όπως το αρχείο τοπολογίας και δημιουργείται ένα αντικείμενο

“ndn” που είναι μία δομή δεδομένων με όλους τους κόμβους, τις παραμέτρους του δικτύου και άλλα. Με σκοπό οι κόμβοι να επικοινωνούν μεταξύ τους με τον επιθυμητό τρόπο, καλούμε σε κάθε κόμβο του δικτύου ξεχωριστά τα προγράμματα των NFD και NLSR. Μία σημαντική λειτουργία στο περιβάλλον του mini-NDN είναι η εξασφάλιση καταχώρησης αρχείων αναφοράς. Στα δύο παραπάνω προγράμματα υπάρχει η δυνατότητα log διαφορετικών επιπέδων λεπτομέρειας, με τα δύο πιο χρήσιμα: INFO και DEBUG. Παρακάτω ορίζουμε την στρατηγική δρομολόγησης σε κάθε κόμβο και αυτό αποτελεί τμήμα που θα μεταβάλλουμε κατά την εξαγωγή αποτελεσμάτων και συμπερασμάτων από το δίκτυο. Τέλος, έχουμε προσθέσει μερικές πρόσθετες γραμμές κώδικα για την δημιουργία πιο εύχρηστων NDN dump logs που θα αναλυθούν παρακάτω.

```
Minindn.cleanUp()
Minindn.verifyDependencies()
ndn = Minindn(parser=getParser())
args = ndn.args

ndn.start()
info('Starting NFD on nodes\n')
nfds = AppManager(ndn, ndn.net.hosts, Nfd, logLevel='INFO', )
info('Starting NLSR on nodes\n')
nlsrs = AppManager(ndn, ndn.net.hosts, Nlsr, logLevel='INFO')

for host in ndn.net.hosts:
    Nfdc.setStrategy(host, "/sync", Nfdc.STRATEGY_MULTICAST)

MyExp(Experiment, ndn, nfds, nlsrs, args)
for host in ndn.net.hosts:
        for intf in host.intfNames():
              ndnDumpOutputFile = "dump.%s" % intf
              host.cmd("sudo ndndump -i %s > %s &" % (intf, ndnDumpOutputFile))
MiniNDNCLI(ndn.net)
ndn.stop()
```

Ο αλγόριθμος του βασικού προγράμματος κάνει κλήση της συνάρτησης MyExp() η οποία αρχικά αναλαμβάνει να παρέχει επαρκή έλεγχο για τη συνδεσιμότητα μεταξύ των κόμβων της τοπολογίας. Με τη χρήση του εργαλείου checkConvergence() που βρίσκεται στην ενσωματωμένη κλάση Experiment ελέγχονται μέσω διαδοχικών pings οι συνδέσεις του δικτύου. Σε περίπτωση σφάλματος το πείραμα δεν ματαιώνεται, αλλά έχει ρυθμιστεί να δημιουργείται log με τη σημαία επιτυχίας του Convergence, στη προκειμένη περίπτωση με τη τιμή FALSE. Το επόμενο τμήμα κώδικα αφορά την εκτέλεση των ειδικών προγραμμάτων στους επιθυμητούς κόμβους και τη διαφήμιση των ονομάτων των διαθέσιμων πακέτων δεδομένων από τους παραγωγούς. Οι παραγωγοί διαφημίζουν

τα prefixes “/example/testApp#”, όπου “#” ο αριθμός ID των διαφορετικών Interests με σκοπό να υπολογιστούν τα μονοπάτια δρομολόγησης για τους μη γειτονικούς κόμβους και να γνωστοποιηθεί το περιεχόμενό τους. Μετά από κάθε ενέργεια ανάθεσης ρόλου και ειδικά διαφήμισης περιεχομένου είναι αναγκαίο να οριστεί ένα μικρό διάστημα αδράνειας προκειμένου να διαδοθούν οι αλλαγές στο σύνολο του δικτύου.

```
producerNum = 10 # Here I define the number of producers and produced packet names (both dumbbell and tree)

def MyExp(Experiment, ndn, nfds, nlsrs, args):

        Experiment.checkConvergence(ndn, ndn.net.hosts, args.ctime, quit=False)

        info('Advertising prefix...\n')

        char = "a"
        charCap = "A"
        myInt = ord(char[0])
        myIntCap = ord(charCap[0])

        for x in range(producerNum):
                char = chr(myInt)
                charCap = chr(myIntCap)
                stringName = "producer{}".format(char)
                stringAdvertise = "nlsrc advertise /example/testApp{}".format(x+1)
                stringRun = "/home/sergios/mini-ndn/ndn-src/ndn-cxx/build/examples/producer{} &> producer.log &".format(charCap)

                # Advertise the content of all the producers
                ndn.net[stringName].cmd(stringAdvertise)

                # Run the producer program in the defined nodes
                ndn.net[stringName].cmd(stringRun)

                myInt = myInt + 1
                myIntCap = myIntCap + 1
                time.sleep(1)

        # Run the router program in the defined nodes
        ndn.net['midnode'].cmd("/home/sergios/mini-ndn/ndn-src/ndn-cxx/build/examples/router &> router.log &")

        time.sleep(10)
        # Run the consumer program in the defined nodes (here 1 node, for Tree or Dumbbell it differs)
        ndn.net['topnod'].cmd("/home/sergios/mini-ndn/ndn-src/ndn-cxx/build/examples/consumer_RTT_rand_multi_test &> consumer.log &")
        info ('Sending Interests...\n')
        time.sleep(10)

        info('experiment run\n')
```

Ο παραπάνω κώδικας αποτελεί τον βασικό πυρήνα των πειραμάτων της εργασίας, πάνω στον οποίο θα συμβούν ορισμένες παραλλαγές. Μέσα στη συνάρτηση MyExp() έχουμε προσαρμόσει επιπλέον συμβάντα διακοπτόμενης σύνδεσης τα οποία θα τρέξουν μόνο σε ορισμένα πειράματα. Ο αλγόριθμος επιλέγει έναν κόμβο του οποίο θα σταματήσει τις λειτουργίες που προσφέρουν τα NFD και NLSR οπότε ουσιαστικά θα ρίξει τη σύνδεσή του με τους γειτονικούς του κόμβους. Δίνεται ο κατάλληλος χρόνος για να συμβεί η ενέργεια και να ανιχνευτεί από το σύστημα και στη συνέχεια ωθεί νέο κύμα από Interests να σταλούν στο δίκτυο. Για την ολοκλήρωση του πειράματος και για την βιβλιογραφική μελέτη της δυνατότητας του δικτύου να επανακάμψει, επαναφέρουμε τις λειτουργίες δρομολόγησης του επιλεγμένου κόμβου και μετά από αντίστοιχο χρόνο αδράνειας το σύστημα στέλνει το τελευταίο, τρίτο κύμα από Interests. Βασική λεπτομέρεια είναι η σωστή διαχείριση των

LSAs που περιγράφουν τις συνδέσεις των κόμβων καθώς στην περίπτωση που καταρρεύσει κάποια σύνδεση με τον παρακάτω τρόπο δεν υπάρχει αυτόματη άμεση ενημέρωση του δικτύου.

```
down = ndn.net['midnodi']
info ('Bringing down node midnodi...\n')
nlsrs[down.name].stop()
nfds[down.name].stop()
time.sleep(60)

ndn.net['consumerb'].cmd("/home/sergios/mini-ndn/ndn-src/ndn-cxx/build/examples/consumer_RTT_rand &> consumer.log &")

info ('Bringing up node midnodi...\n')
nlsrs[down.name].start()
nfds[down.name].start()
time.sleep(60)

ndn.net['consumerc'].cmd("/home/sergios/mini-ndn/ndn-src/ndn-cxx/build/examples/consumer_RTT_rand &> consumer.log &")
time.sleep(10)

Experiment.checkConvergence(ndn, ndn.net.hosts, args.ctime, quit=False)

info('experiment run\n')
```

Σημαντική παρατήρηση για το χρονοδιάγραμμα των ενεργειών του συστήματος είναι ότι οι ενέργειες που ενεργοποιούνται στα προγράμματα consumer, producer και router δεν σχετίζονται με τις υπόλοιπες λειτουργίες του βασικού αλγορίθμου του πειράματος. Πιο συγκεκριμένα, από τη στιγμή που κληθεί για παράδειγμα το πρόγραμμα του καταναλωτή, τα Interests δε θα σταματήσουν να στέλνονται ακόμα και αν το υπόλοιπο πρόγραμμα βρίσκεται προσωρινά σε κατάσταση αδράνειας. Περιγράφουν ανεξάρτητες δραστηριότητες, η μια αυτές που σχετίζονται με την έναρξη και παύση λειτουργιών στο σύστημα και η άλλη τις δραστηριότητες που πραγματοποιούν οι κόμβοι σαν μονάδες λόγω των ρόλων που τους έχουμε προσδώσει..

2.3.4 Λειτουργίες καταγραφής αποτελεσμάτων – Logs

Μία πρόκληση της συγκεκριμένης εργασίας ήταν η περιορισμένη ύπαρξη τυποποιημένων κλάσεων στο mini-NDN για συλλογή δεδομένων σε πειράματα ανταλλαγής Interest-Data πακέτων. Με τη χρήση του κατάλληλου κώδικα τόσο στο πείραμα όσο και στα προγράμματα που τρέχουν στους κόμβους, υπάρχει η δυνατότητα να δημιουργούνται αρχεία logs σε φακέλους-προορισμούς με τα ζητούμενα δεδομένα του δικτύου που μοντελοποιείται. Τα εξωτερικά προγράμματα καταναλωτή και παραγωγού που προσαρμόσαμε στο σύστημα παρουσιάζουν τα πιο ενδιαφέροντα προς εμάς αποτελέσματα και για αυτό το λόγο στη συγκεκριμένη εργασία χρησιμοποιούνται τρεις προσαρμοσμένες λειτουργίες δημιουργίας καταχωρήσεων.

Η πρώτη που ήδη παρουσιάστηκε στην ανάλυση του κώδικα είναι η NDN-dump καταχώρηση που δίνει πληροφορίες για τα πακέτα που διέρχονται από συγκεκριμένες συνδέσεις. Ελέγχει τόσο τα Interest και Data πακέτα, όσο και άλλα πακέτα που συμβάλλουν στη λειτουργία του NLSR πρωτοκόλλου. Από τις συγκεκριμένες καταχωρήσεις μπορούμε να διακρίνουμε την πορεία ορισμένων Interests χωρίς τη χρήση άλλων τεχνασμάτων όπως αυτού της ταύτισης του βάρους και της καθυστέρησης διάδοσης του πακέτου στο δίκτυο.

Εικόνα 2.10: Καταγραφή πακέτων που διέρχονται από το κανάλι συγκεκριμένης σύνδεσης κόμβων

Οι υπόλοιπες καταχωρήσεις συμβαίνουν από κώδικα προσαρμοσμένο στα προγράμματα consumer, producer και router. Για κάθε έναν κόμβο που τρέχει κάποιο από τα συγκεκριμένα προγράμματα δημιουργείται ένα προσωρινό log αρχείο στον */temp/* φάκελο των linux. Διαθέτουν τις πιο χρήσιμες πληροφορίες των πειραμάτων και είναι η βασική πηγή εξαγωγής δεδομένων για το σύστημα. Πιο συγκεκριμένα, στην εφαρμογή του παραγωγού δημιουργείται log που καταγράφει τα ονόματα των Interest που αποστέλλονται στο σύστημα, την απόκριση του συστήματος, πληροφορίες για τα χαρακτηριστικά του Interest πακέτου που στέλνεται και του πακέτου δεδομένων όταν αυτό επιστρέφει επιτυχώς, την RTT καθυστέρηση του δικτύου για το κάθε πακέτο, την εν δυνάμει επιτυχή λειτουργία του in-network caching και την caching πολιτική που χρησιμοποιείται, αφού γεμίσει ο buffer.

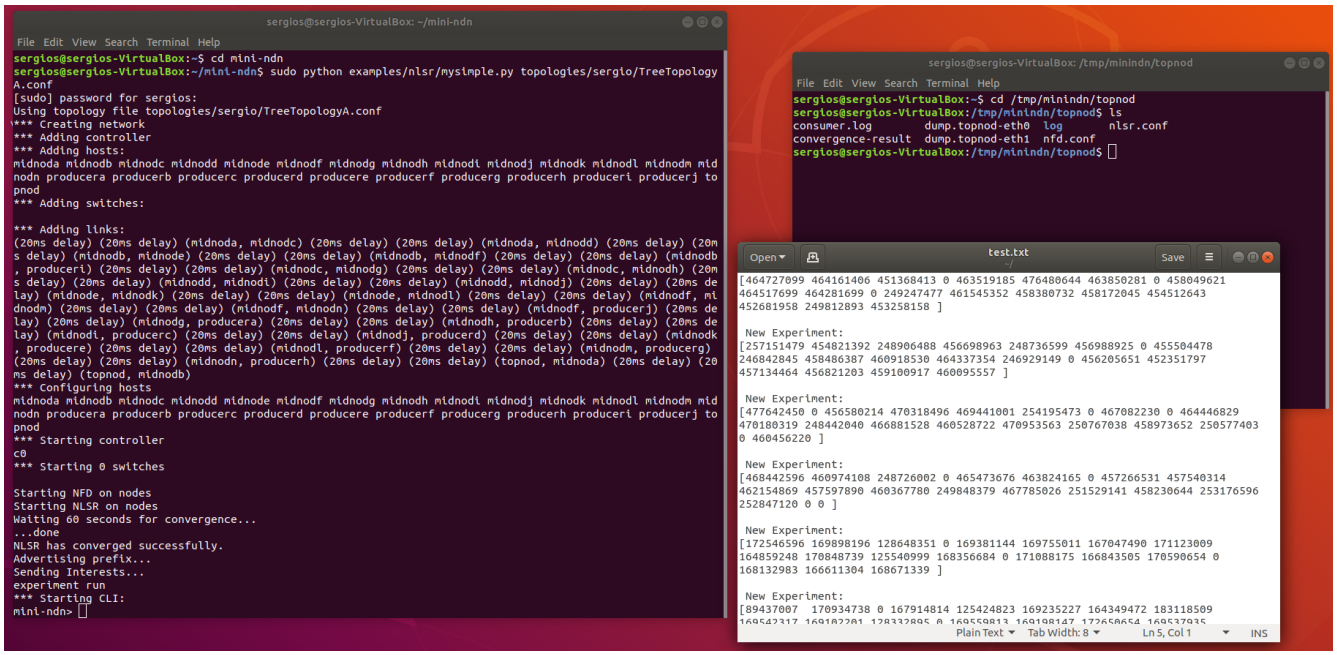
```
/example/testApp6/randomData
Sending Interest /example/testApp6/randomData/%FD%00%00%01%7B%B6%3E%9E%06?MustBeFresh&Lifetime=2000
Received Data Name: /example/testApp6/randomData/%FD%00%00%01%7B%B6%3E%9E%06
MetaInfo: [ContentType: 0, FreshnessPeriod: 10000 milliseconds]
Content: [13 bytes]
Signature: [type: SignatureSha256WithEcdsa, length: 70]

Total time: 171031067 nanoseconds
Received Content: Hello, world!

>>> One more Interest, delayed by the scheduler - Number 1
Node already has the packet in its Content Store (/example/testApp6/randomData)

>>> One more Interest, delayed by the scheduler - Number 2
Sending Interest /example/testApp8/randomData/%FD%00%00%01%7B%B6%3E%B5v?MustBeFresh&Lifetime=2000
Received Data Name: /example/testApp8/randomData/%FD%00%00%01%7B%B6%3E%B5v
MetaInfo: [ContentType: 0, FreshnessPeriod: 10000 milliseconds]
Content: [13 bytes]
Signature: [type: SignatureSha256WithEcdsa, length: 71]

Total time: 173795016 nanoseconds
Received Content: Hello, world!

>>> One more Interest, delayed by the scheduler - Number 3
Sending Interest /example/testApp5/randomData/%FD%00%00%01%7B%B6%3E%C1.?MustBeFresh&Lifetime=2000
Received Data Name: /example/testApp5/randomData/%FD%00%00%01%7B%B6%3E%C1.
MetaInfo: [ContentType: 0, FreshnessPeriod: 10000 milliseconds]
Content: [13 bytes]
Signature: [type: SignatureSha256WithEcdsa, length: 72]

Total time: 87721699 nanoseconds
Received Content: Hello, world!
test FIFO works
```

Εικόνα 2.11: Καταγραφή αποτελεσμάτων στον /tmp/minindn/consumer φάκελο

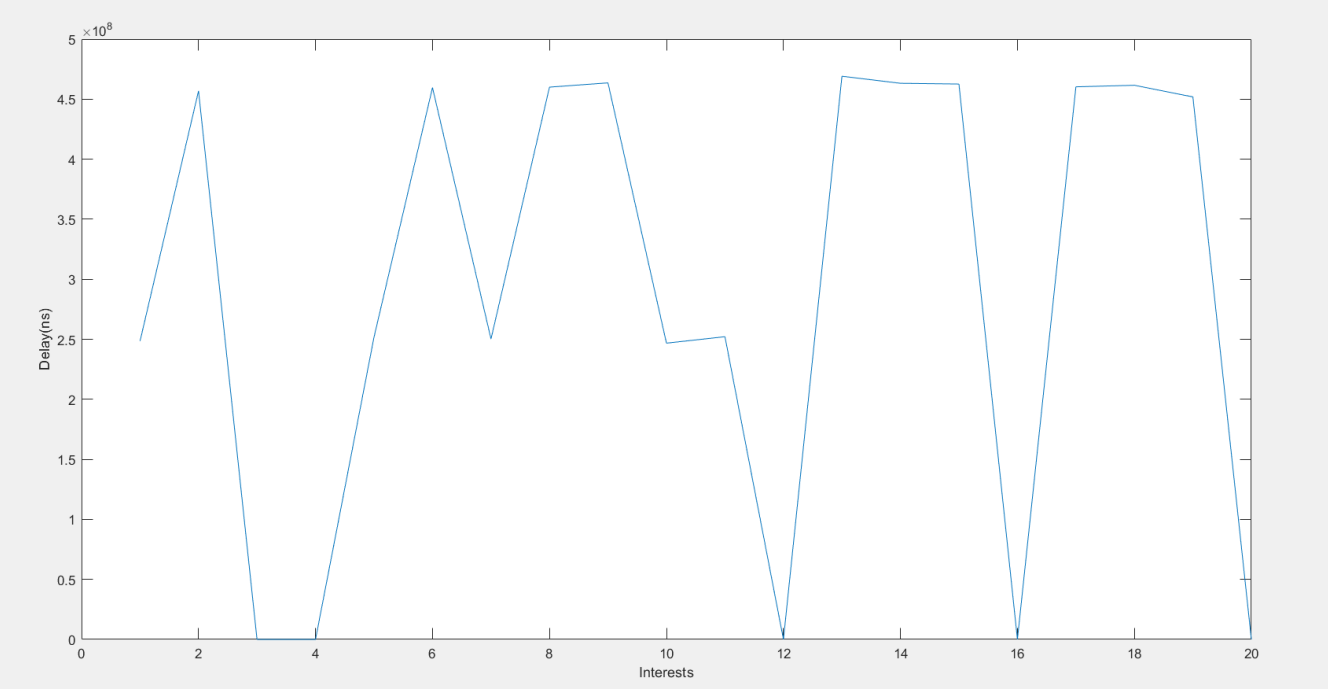
Για την εύκολη χρήση των κατά τα άλλα ανεπεξέργαστων δεδομένων, τα προγράμματα τρέχουν ένα τελευταίο τμήμα κώδικα το οποίο δημιουργεί νέες καταχωρήσεις σε ένα μόνιμα αποθηκευμένο text αρχείο. Οι καταχωρήσεις αφορούν τα RTT αποτελέσματα και τα ονόματα του κάθε ζητούμενου Interest που προκύπτουν σε κάθε κλήση του προγράμματος “consumer”. Καταχωρούνται κατά τέτοιο τρόπο έτσι ώστε να είναι έτοιμα προς χρήση στο περιβάλλον του MATLAB και διευκολύνουν τη δημιουργία γραφικών παραστάσεων σε κάθε πείραμα που δέχεται διαφορετικές παραμέτρους.

Στην Εικόνα 2.13 το γράφημα αποτελεί προϊόν της plot λειτουργίας του MATLAB όταν δίνεται ως είσοδος μία από τις αναγραφόμενες καταχωρήσεις του αρχείου “test.txt” (Εικόνα 2.12). Μπορούμε προσεγγιστικά να αναφερθούμε σε τρεις στάθμες καθυστέρησης στη συγκεκριμένη γραφική παράσταση και αυτό οφείλεται στις λειτουργίες caching στον καταναλωτή, in-network caching σε έναν router και στο ίδιο βάρος των διαδρομών προς τους παραγωγούς. Εάν έχουμε σαν δεδομένο ότι το συγκεκριμένο πείραμα διεξήχθη σε τοπολογία Dumbbell, τότε η υψηλότερη στάθμη υποδηλώνει ότι το Interest έφτασε μέχρι τον τελικό παραγωγό και υπήρχε απόκριση με το επιθυμητό πακέτο. Η μεσαία στάθμη υποδηλώνει ότι το αίτημα του καταναλωτή ικανοποιήθηκε ενδιάμεσα στο

δίκτυο πριν φτάσει στη τελική γραμμή των παραγωγών, οπότε λειτούργησε αποδοτικά η in-network προσωρινή αποθήκευση πακέτων. Τέλος, ο μηδενικός χρόνος υποδηλώνει ότι στον περιορισμένο χώρο αποθήκευσης του καταναλωτή υπήρχε ήδη αντίγραφο του ζητούμενου πακέτου, οπότε περαιτέρω προώθηση Interest θα ήταν άσκοπη. Παρατηρούμε ότι όσο περισσότερο γίνεται χρήση των caching λειτουργιών, η απόδοση του συστήματος μειώνεται δραστικά.



Εικόνα 2.12: Καταγραφή αποτελεσμάτων στο αρχείο home/test.txt



Εικόνα 2.13: Παράσταση της καθυστέρησης του συστήματος για είκοσι πακέτα ενδιαφέροντος

Κεφάλαιο 3ο

3.1 Παραδοχές

Πριν ξεκινήσουμε την πειραματική διαδικασία πρέπει να κάνουμε ορισμένες παραδοχές για να μειώσουμε τον αριθμό των παραμέτρων που επιδρούν στη μοντελοποίηση του δικτύου και συνεπώς στα αποτελέσματά μας. Οι παρακάτω παραδοχές θα συμβάλλουν επίσης στην μείωση των προγραμματισμένων ενεργειών του συστήματος και άρα του φόρτου και του χρόνου των υπολογιστικών διεργασιών.

1) Σε αυτή την εργασία δεν θα ασχοληθούμε με το εύρος Bandwidth καθώς τα πακέτα που στέλνονται έχουν αμελητέο μέγεθος δεδομένων και στέλνονται διαδοχικά ανά τρία δευτερόλεπτα, οπότε το σενάριο της υπερφόρτωσης του δικτύου δεν έχει βάση στα πειράματά μας.

2) Θεωρούμε ότι τα LSAs ενημερώνουν αμέσως το δίκτυο για τυχόν αλλαγές στη τοπολογία του, όπως για παράδειγμα σε κάποια πτώση σύνδεσης. Υπό άλλες συνθήκες θα έπρεπε να υπάρχει πολύ μεγάλη αναμονή σε κάθε πείραμα προκειμένου να λήξει ο ορισμένος χρόνος ζωής αυτών και να επανασταλούν ανά μεταξύ των κόμβων. Την θεωρητική ανανέωση των LSAs την επιτυγχάνουμε με την επανεκτέλεση του προγράμματος NLSR στους κόμβους που παραμένουν αμετάβλητοι, μετά από

οποιαδήποτε εξαναγκασμένη αλλαγή στη τοπολογία του συστήματος. Η διάδοση αυτής της πληροφορίας στο υπόλοιπο δίκτυο θα χρειαστεί κανονικά, πραγματικό χρόνο, οπότε η χρήση των εντολών αναμονής του δικτύου είναι αναγκαία.

3) Στην περίπτωση απόκρισης Timeout σε κάποιο πακέτο ενδιαφέροντος θεωρούμε ότι θα υπάρξει άμεση επαναποστολή του ίδιου πακέτου, η οποία όμως δεν θα επηρεάσει το χρονοδιάγραμμα του επόμενου Interest, εκτός εάν διευκρινιστεί διαφορετικά. Τα δύο συμβάντα είναι τελείως ανεξάρτητα και σε περίπτωση μεγάλης καθυστέρησης ενός πακέτου μπορούν να υπάρξουν δύο ή περισσότερα πακέτα διαδιδόμενα στο δίκτυο. Αν η επαναποστολή έχει ως αποτέλεσμα ξανά timeout θεωρούμε ότι η σύνδεση μεταξύ δύο κόμβων της διαδρομής έχει καταρρεύσει οπότε δεν έχει νόημα περαιτέρω προσπάθεια για ανάκτηση των συγκεκριμένων δεδομένων. Εάν φυσικά υπάρχει εναλλακτική διαδρομή, το σύστημα κάνει επαναποστολή των Interest από εκείνη για να αποφύγει περιττή καθυστέρηση.

3.2 Εκτέλεση πειραμάτων

Κάθε διαφορετικό πείραμα θα βασίζεται στη μεταβολή ξεχωριστών παραμέτρων για την εξαγωγή των επιθυμητών συμπερασμάτων. Στα προγράμματα των κόμβων καθώς και στο ίδιο το βασικό πείραμα έχουμε ορίσει συγκεκριμένες global σταθερές που αντιπροσωπεύουν τις προαναφερθέντες παραμέτρους. Η cachePolicy ορίζει την πολιτική της εισόδου πακέτων στην μνήμη cache, η csLength το μέγεθος της cache, η packetsNum τον αριθμό των διαφορετικών ονομάτων δεδομένων που υπάρχουν στο δίκτυο, η interestsSent τον αριθμό των Interest Packets που αποστέλλουν οι καταναλωτές και η linearProb ενεργοποιεί την γραμμική κατανομή πιθανοτήτων στα ονόματα των Interest Packets. Στα πειράματα που ακολουθούν θα γίνει πιο αναλυτική αναφορά στον ρόλο των σταθερών που επηρεάζουν την κάθε προσομοίωση.

```
#define cachePolicy "LFU" // Policies: "Static", "FIFO", "RR", "LFU"
#define csLength 2 //default value for cache length
#define packetsNum 10 //default value for the number of different requested packet names
#define interestsSent 20 //default value for the amount of interests sent by the consumer
#define linearProb true //Switch linear probability distribution for Interests ON/OFF
```

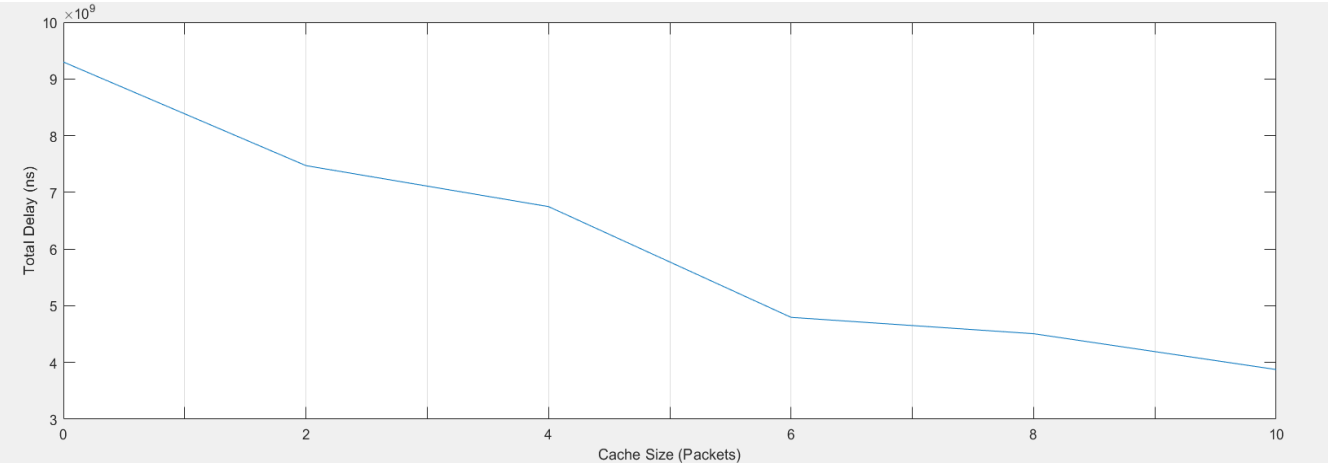
Για να προκύψουν αποτελέσματα τα οποία είναι αντιπροσωπευτικά των πειραμάτων που τρέχουμε, η κάθε μέτρηση επαναλαμβάνεται δέκα φορές του ίδιου πειράματος και σαν τιμή

παίρνουμε τον μέσο όρο των αποτελεσμάτων. Στις περιπτώσεις που θέλουμε να δείξουμε συγκεκριμένα συμβάντα από κάποιο πείραμα θα δειχθεί το πιο αντιπροσωπευτικό από τα δέκα, το κοντινότερο δηλαδή στον μέσο όρο των αποτελεσμάτων που υπολογίστηκαν. Στην προσομοίωση του δικτύου, ο αριθμός των κόμβων, των επαναλήψεων των πειραμάτων και των παραμέτρων είναι τέτοιος έτσι να περιορίζεται ο υπολογιστικός φόρτος του συστήματος ενώ ταυτόχρονα εξάγει αποτελέσματα που μπορούμε να αναγάγουμε σε μεγαλύτερα και περιπλοκότερα δίκτυα.

3.2.1 Πείραμα πρώτο – Μέγεθος Μνήμης Cache

Αρχικά θα μελετήσουμε την συμβολή της μνήμης cache του καταναλωτή στο throughput του δικτύου. Μεταβάλλοντας τον αποθηκευτικό χώρο του κόμβου που τρέχει το πρόγραμμα consumer μπορούμε να επιτύχουμε μηδενική καθυστέρηση διάδοσης σε όσα πακέτα δεδομένων βρίσκονται ήδη στον buffer. Η σταθερά που προσδιορίζει το μέγεθος του buffer είναι η csLength και επηρεάζει άμεσα λειτουργίες όπως τις πολιτικές caching. Το συγκεκριμένο πείραμα χρησιμοποίησε την απλή πολιτική FIFO για την προσωρινή αποθήκευση πακέτων χωρίς χαρακτηριστικά opportunistic caching, τα πακέτα που ζητούνταν ακολουθούσαν απολύτως τυχαία κατανομή και σε κάθε διαφορετικό σενάριο το μέγεθος του buffer υπέκυπτε σε αύξηση κατά δύο θέσεις. Για κάθε μέτρηση χρησιμοποιήθηκε η σταθερή τιμή των δέκα διαφορετικών ονομάτων πακέτων και των είκοσι αποστολών πακέτων ενδιαφέροντος.

Το συγκεκριμένο πείραμα έχει ως στόχο να εκλάβει τους καταναλωτές κόμβους ως οντότητες που συνδέουν άμεσα κάποια εφαρμογή καταναλωτή στο δίκτυο ή ακόμη και έμμεσα εξωτερικά δίκτυα που δρομολογούν το αίτημά τους στο δίκτυο προς μελέτη.



Σχήμα 3.1: Παράσταση της συνολικής καθυστέρησης του δικτύου συναρτήσει του μεγέθους cache

Στο Σχήμα 3.1 παρατηρούμε πως με την αύξηση του μεγέθους της cache μνήμης μειώνεται σχεδόν με σταθερή κλίση η καθυστέρηση το δικτύου να ικανοποιήσει και τα είκοσι πακέτα-αιτήματα. Όταν η χωρητικότητα της μνήμης μπορεί να εξυπηρετήσει δέκα πακέτα, δηλαδή τα μισά από όσα είναι τα διαδιδόμενα Interests στο δίκτυο, η καθυστέρηση μειώνεται σε τιμή μικρότερη του μισού της αρχικής και αυτό συμβαίνει γιατί η τυχαία αποστολή πακέτων δεν εγγυάται ισόποση εμφάνιση όλων των πακέτων ακόμα και μετά από πολλές επαναλήψεις του ίδιου πειράματος. Εφόσον ένα πακέτο δεδομένων ζητηθεί δύο ή περισσότερες φορές, η καθυστέρηση του δικτύου μειώνεται καθώς το νέο Interest εξυπηρετείται αμέσως. Η μικρή διακύμανση στη συχνότητα ζήτησης συγκεκριμένων πακέτων μας δίνει καλύτερο χρόνο εκτέλεσης του προγράμματος. Πιο συγκεκριμένα, τα στατιστικά εμφάνισης των διαφορετικών πακέτων στο πείραμα εμφανίζονται στον παρακάτω πίνακα.

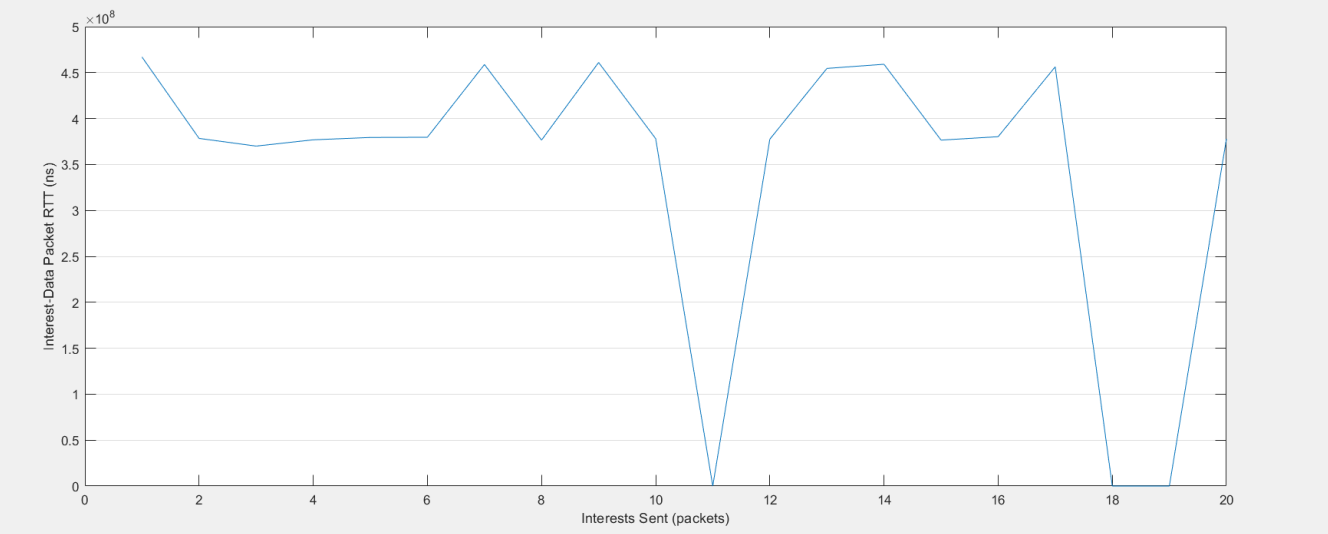
Interest	Συχνότητα Εμφάνισης (%)
“/testApp1”	13.33
“/testApp2”	5
“/testApp3”	10
“/testApp4”	8.33
“/testApp5”	10
“/testApp6”	10
“/testApp7”	13.33
“/testApp8”	10
“/testApp9”	11.66

Interest	Συχνότητα Εμφάνισης (%)
“/testApp10”	8.33

3.2.2 Πείραμα δεύτερο – In-Network Caching

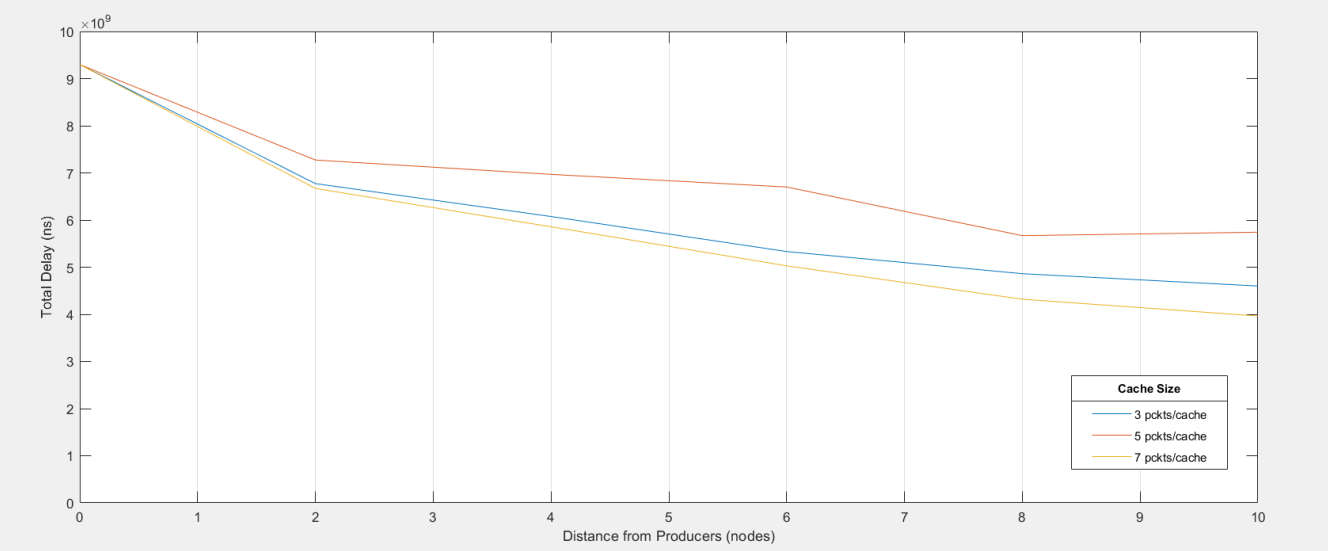
Το δεύτερο πείραμα που θα διεξάγουμε θα επικεντρωθεί στην προέκταση του προηγουμένου, στην επίδραση της in-network caching λειτουργίας στα αποτελέσματα καθυστερήσεως του δικτύου. Η συγκεκριμένη λειτουργία πραγματοποιείται από την ανάθεση του ρόλου “router” σε κάποιον ενδιάμεσο κόμβο – δρομολογητή. Όταν ένας κόμβος φιλοξενεί το πρόγραμμα του router τότε υιοθετεί λειτουργίες παρόμοιες με του παραγωγού με την έννοια ότι εάν βρεθεί σε αυτόν Interest που απευθύνεται σε κάποιο από τα αποθηκευμένα πακέτα δεδομένων του buffer του, τότε ο κόμβος επιστρέφει απευθείας αντίγραφο αυτού χωρίς περαιτέρω διαδικασίες προώθησης. Εδώ θα ελέγξουμε την απόδοση του συστήματος σε πειράματα όπου η θέση και ο αριθμός των routers αλλάζει. Η τοπολογία που διαλέξαμε είναι η Dumbbell έτσι ώστε να υπάρχει βεβαιότητα ότι εάν ζητηθεί ένα πακέτο στην cache μνήμη του δρομολογητή, τότε αυτός σίγουρα θα βρεθεί στη διαδρομή του πακέτου ενδιαφέροντος. Η πολιτική της προσωρινής αποθήκευσης είναι η FIFO, τέτοια ώστε να μην υπάρχει καμία προσαρμοστικότητα στα συμβάντα του πειράματος και η ποσότητα των Interests και των ονομάτων στο δίκτυο παραμένει ίδιο.

Όσο πιο κοντά βρίσκεται ο κόμβος με δυνατότητα caching στον καταναλωτή, τόσο χαμηλότερο θα είναι και το RTT της ανταλλαγής Interest-Data πακέτων. Αυτό συμβαίνει γιατί στη τοπολογία Dumbbell ο κάθε επιπλέον κόμβος που βρίσκεται στη διαδρομή κάποιας αποστολής προσθέτει επιπλέον καθυστέρηση στη μέτρηση. Στο παρακάτω Σχήμα 3.2 συμπεραίνουμε πως η μεσαία στάθμη καθυστερήσεων των πακέτων δημιουργείται λόγω κάποιου κόμβου με δυνατότητα προσωρινής αποθήκευσης, ο οποίος μάλιστα βρίσκεται κοντά στους υπόλοιπους παραγωγούς αφού η πτώση του RTT είναι μικρή. Αξίζει να αναφερθούμε στο γεγονός ότι η cache λειτουργία στον κόμβο του καταναλωτή είναι ενεργή οπότε θα έχει επίσης επίπτωση στα αποτελέσματα των πειραμάτων.



Σχήμα 3.2: Παράσταση της καθυστέρησης του συστήματος για είκοσι νέα πακέτα ενδιαφέροντος

Οι μετρήσεις του πειράματος έγιναν με διαδοχικές αλλαγές στη θέση του κόμβου router στο δίκτυο και στη συνέχεια με την αλλαγή της χωρητικότητας της μνήμης cache. Η προσαρμογή του πρώτου πειράματος στο συγκεκριμένο σενάριο έχει βάση, καθώς αυξάνοντας την πιθανότητα των cache hits γίνεται πιο ευδιάκριτη η συμβολή της θέσης του router στο throughput του δικτύου. Στο Σχήμα 3.3 φαίνεται η μείωση της συνολικής καθυστέρησης του δικτύου συναρτήσει της απόστασης της μνήμης cache από τους παραγωγούς. Ταυτόχρονα οι διαφορετικές καμπύλες υποδεικνύουν τα σενάρια διαφορετικών χωρητικοτήτων στη μνήμη. Η απόδοση του δικτύου αυξάνεται περίπου στο 25% όταν ο router με χωρητικότητα τριών πακέτων βρίσκεται στη μέση της διαδρομής και καταλήγει σε αύξηση περίπου 38% όταν βρίσκεται στον κοντινότερο δρομολογητή προς τον καταναλωτή. Τα νούμερα αυξάνουν για μεγαλύτερη χωρητικότητα μνήμης (εφτά πακέτα δεδομένων) σε 45% και 57% για τις δύο περιπτώσεις αντίστοιχα.



Σχήμα 3.3: Παράσταση της συνολικής καθυστέρησης του δικτύου συναρτήσει της απόστασης της in-network cache από τους παραγωγούς, για διαφορετικά μεγέθη cache

3.2.3 Πείραμα Τρίτο – Πολιτική Caching

Στο συγκεκριμένο πείραμα θα μελετήσουμε τη συμπεριφορά των διαφορετικών caching πολιτικών στο δίκτυο των πειραμάτων. Οι τέσσερις πολιτικές με τις οποίες θα ασχοληθούμε είναι οι “First In First Out”, “Admit Off”, “Random Replacement” και “Least Frequently Used”. Προκειμένου να εξάγουμε βάσιμα συμπεράσματα ενεργοποιούμε την γραμμική κατανομή της συχνότητας εμφάνισης των πακέτων, έτσι ώστε πλέον να υπάρχει διαφοροποίηση στη συχνότητα που θα εμφανίζονται συγκεκριμένα Interests. Η συχνότητα εμφάνισης του κάθε Interest όπως καταγράφηκε στα πειράματα αναφέρεται στον παρακάτω πίνακα. Το μέγεθος και άρα ο αποθηκευτικός χώρος που καταλαμβάνει το κάθε Interest παραμένει ίδιος σε όλα τα πακέτα και τα βάρη των διαδρομών για τον κάθε διαφορετικό παραγωγό είναι επίσης ίσα.

Interest	Συχνότητα Εμφάνισης (%)
“/testApp1”	2.5
“/testApp2”	3.75
“/testApp3”	2.5
“/testApp4”	10
“/testApp5”	7.5

Interest	Συχνότητα Εμφάνισης (%)
“/testApp6”	13.75
“/testApp7”	12.5
“/testApp8”	16.25
“/testApp9”	17.5
“/testApp10”	16.25

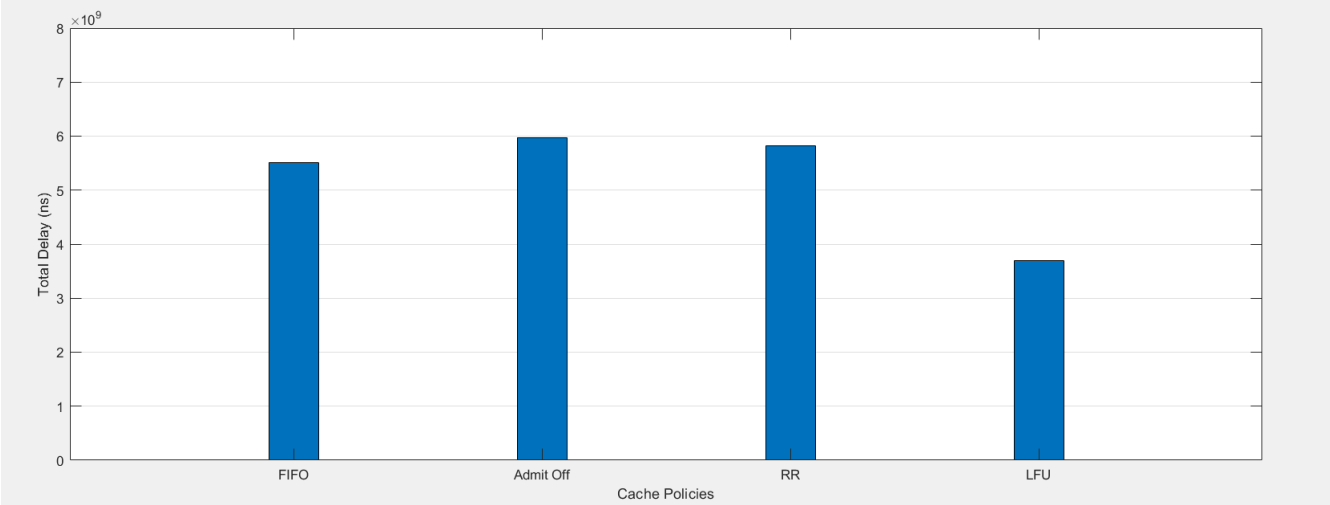
Το πείραμα διατηρεί τις προηγούμενες παραμέτρους σταθερές και πραγματοποιεί δέκα μετρήσεις για την κάθε πολιτική. Οι μετρήσεις αφορούν την συνολική καθυστέρηση του δικτύου για την εξυπηρέτηση όλων των Interests και το σύνολο των cache hits σε κάθε περίπτωση, από τις οποίες λαμβάνουμε τον μέσο όρο σαν αποτέλεσμα. Το Σχήμα 3.4 δείχνει την διαφορά των τεσσάρων πολιτικών όσον αφορά την επίδρασή τους στην καθυστέρηση του συστήματος, ενώ το Σχήμα 3.5 καταγράφει το ποσοστό των πακέτων που ικανοποιούνται από την μνήμη cache του δικτύου στη κάθε περίπτωση.

Μεταξύ των πολιτικών η Admit Off αποτελεί την λιγότερο προσαρμοστική, με μηδενική ευελιξία σε οποιοδήποτε συμβάν, καθώς λειτουργεί διατηρώντας τα πρώτα πακέτα που θα γεμίσουν τη μνήμη ες αεί. Η περίπτωση στην οποία τα πρώτα προς αποστολή Interests δεν έχουν μεγάλη συχνότητα εμφάνισης μπορεί να έχει ως αποτέλεσμα την αχρήστευση της λειτουργίας caching του NDN δικτύου στο συγκεκριμένο πείραμα. Για τον λόγο αυτό η συγκεκριμένη πολιτική επιφέρει και τη μεγαλύτερη καθυστέρηση στο δίκτυο.

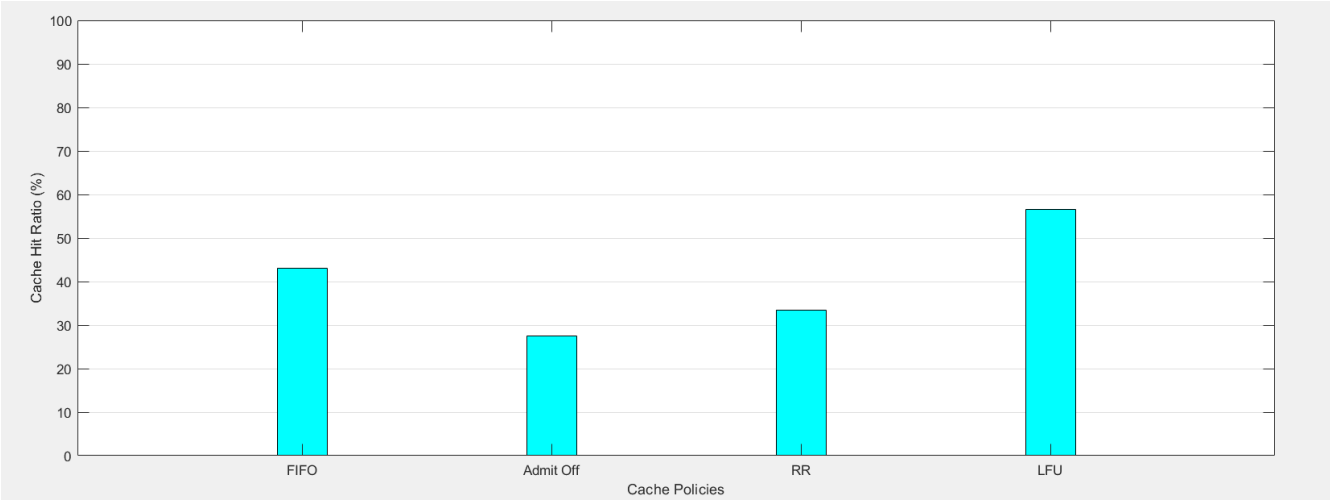
Με σχεδόν διπλάσιο ποσοστό επιτυχίας στην μνήμη, η πολιτική LFU εκφράζει έναν προσαρμοστικό αλγόριθμο όπου όσο αυξάνεται ο αριθμός των Interests και καθίσταται σαφές ποια πακέτα εμφανίζονται συχνότερα, τόσο γίνεται πιο αποδοτικός. Από τις παραμέτρους που επηρεάζουν την απόφαση μιας opportunistic πολιτικής, η μόνη που έχει ενεργό δράση στο συγκεκριμένο πείραμα είναι η συχνότητα εμφάνισης των πακέτων και αυτό καθιστά την LFU που κρατάει στη μνήμη τα πακέτα με τη μεγαλύτερη συχνότητα ζήτησης, την βέλτιστη επιλογή.

Οι FIFO και Random Replacement πολιτικές φαίνεται να έχουν κάποια απόκριση στις συνθήκες του πειράματος. Η FIFO εξυπηρετεί συγκεκριμένο αριθμό πακέτων για συγκεκριμένο διάστημα οπότε εάν υπάρχει μεγάλη συχνότητα εμφάνισης ενός πακέτου ενδιαφέροντος, είναι επίσης υψηλή η πιθανότητα να βρίσκεται αντίγραφο του ζητούμενου πακέτου δεδομένων ήδη μέσα στη μνήμη. Αντίθετα, η RR χρησιμοποιεί την τυχαία αντικατάσταση σαν νοοτροπία διαχείρισης της διαθέσιμης

μνήμης που σε συνθήκες άνισης κατανομής πιθανοτήτων εμφάνισης πακέτων δημιουργεί έναν ασταθή παράγοντα και δεν συμβάλει στην καλύτερη απόδοση του συστήματος.



Σχήμα 3.4: Παράσταση της συνολικής καθυστέρησης του δικτύου για διαφορετικές πολιτικές προσωρινής αποθήκευσης



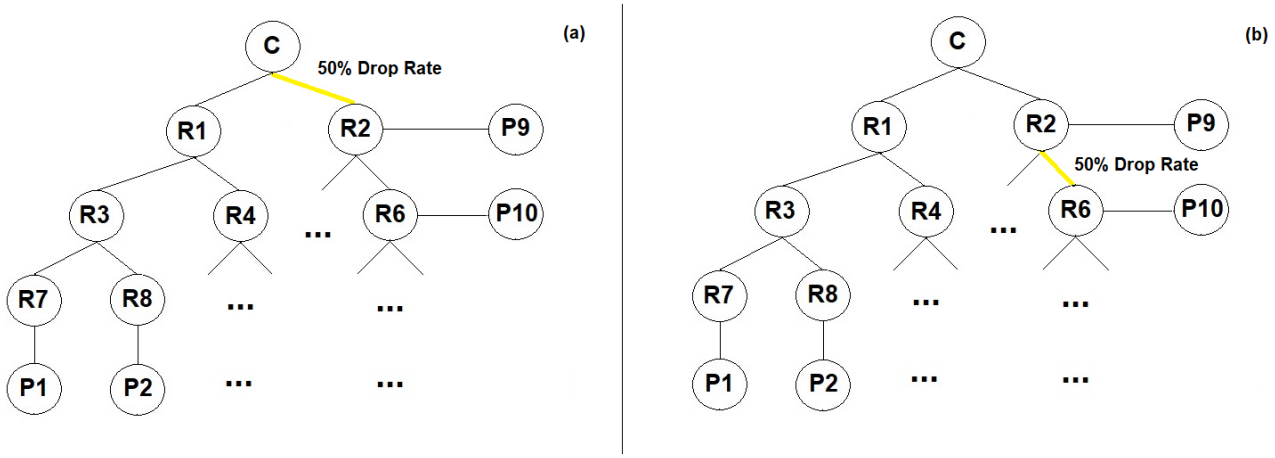
Σχήμα 3.5: Παράσταση του ποσοστού επί τοις εκατό των cache hits για διαφορετικές πολιτικές προσωρινής αποθήκευσης

3.2.4 Πείραμα Τέταρτο – Τοπολογίες δικτύου / Πτώση Πακέτου

Σε αυτό το πείραμα θα συγκρίνουμε τις δύο τοπολογίες των πειραμάτων στο σενάριο ότι σε κάποια σύνδεση ενδιάμεσου κόμβου τα πακέτα να απορρίπτονται με πιθανότητα 50%. Το αρχείο της τοπολογίας στο mini-NDN μας δίνει τη δυνατότητα να ορίσουμε την πιθανότητα απόρριψης πακέτου

προσθέτοντας το χαρακτηριστικό “loss=50” στην επιθυμητή καταχώρηση του τμήματος [links]. Η επανάληψη της κάθε μέτρησης γίνεται από δέκα φορές, η λειτουργία caching τίθεται ανενεργή και όλες οι υπόλοιπες παράμετροι παραμένουν ίδιες.

Εδώ θα εξάγουμε συμπεράσματα για τη χρησιμότητα των διαφορετικών τοπολογιών δικτύου μέσα από τις λειτουργίες που μας παρέχουν τα δίκτυα NDN. Ο σκοπός του πειράματος είναι παράλληλα να μοντελοποιηθεί επιτυχώς το σενάριο της απόρριψης πακέτων σε διαφορετικές τοπολογίες με αποδεκτή απόκριση του NDN δικτύου. Οι μετρήσεις θα γίνουν με τη ρύθμιση της απόρριψης πακέτων να εμφανίζεται σε διαφορετικούς κόμβους και πιο συγκεκριμένα σε κόμβους διαφορετικών επιπέδων στην τοπολογία Δέντρου ή Star-Bus. Στο δίκτυο Dumbbell η εναλλαγή του κόμβου στον οποίο γίνεται η απόρριψη πακέτων δεν έχει ουσιαστική διαφορά καθώς όλοι οι παραγωγοί συνδέονται από την ίδια διαδρομή με τους καταναλωτές. Το χαρακτηριστικό αυτό που διαχωρίζει τα δύο δίκτυα δίνει ένα πλεονέκτημα στις συνδεσμολογίες Star-Bus όσον αφορά την εξακρίβωση της θέσης του σφάλματος καθώς τα πακέτα που αποστέλλονται στα υποδίκτυα που δεν έχουν θυγατρική σχέση με τον κόμβο που εισάγει το συγκεκριμένο σφάλμα, δεν απορρίπτονται (Εικόνα 3.1). Για τον ίδιο λόγο η απόδοση του δικτύου στην περίπτωση ενός τέτοιου σεναρίου προκύπτει μεγαλύτερη από αυτή της τοπολογίας Dumbbell.

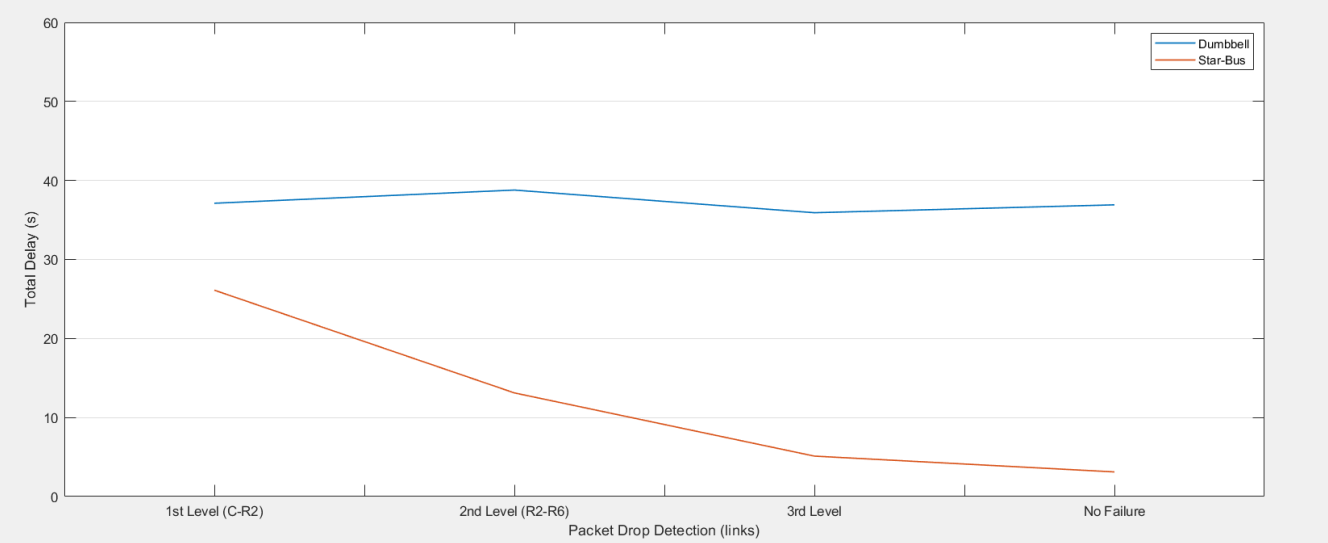


Εικόνα 3.1: Αναπαράσταση της θέσης του σφάλματος που απορρίπτει πακέτα με πιθανότητα 50%

Πράγματι, μέσα από τις μετρήσεις του πειράματος, στο Σχήμα 3.6 φαίνεται η μεγάλη χρονική διαφορά μεταξύ των δύο τοπολογιών. Ήδη από την πρώτη μέτρηση όπου το σφάλμα εμφανίζεται στη σύνδεση του καταναλωτή με το έναν δρομολογητή και περιλαμβάνει πολλούς θυγατρικούς κόμβους, η απόδοση του συστήματος έχει βελτιωθεί κατά σχεδόν 30% σε σχέση με την αντίστοιχη της Dumb-

bell τοπολογίας για σφάλμα σε οποιονδήποτε κόμβο της. Όταν το σφάλμα εμφανίζεται σε μία σύνδεση παραγωγού και δρομολογητή που βρίσκεται σε χαμηλό επίπεδο, η απόδοση του συστήματος βελτιώνεται έως 80%. Αξίζει να αναφερθεί ότι όταν ένα πακέτο απορρίπτεται από το δίκτυο, ο καταναλωτής επαναλαμβάνει την αποστολή του ίδιου ενδιαφέροντος το οποίο έχει τις ίδιες πιθανότητες να απορριφθεί. Η συνθήκη αυτή οδηγεί σε τόσο μεγάλες χρονικές διαφορές μεταξύ των πειραμάτων. Τέλος, στο σχήμα η καμπύλη της τοπολογίας Dumbbell δεν ανταποκρίνεται στον άξονα της απομάκρυνσης των σφαλμάτων από τον καταναλωτή, αλλά διαθέτει μία καμπύλη - σημείο αναφοράς με το σφάλμα σε τυχαίους κόμβους της τοπολογίας. Στο σενάριο όπου δεν υπάρχει σφάλμα στο δίκτυο οι καθυστερήσεις των δύο τοπολογιών είναι ίσες.

Συμπεραίνουμε πως στο συγκεκριμένο σενάριο η τοπολογία Star-Bus αποτελεί την βέλτιστη επιλογή αλλά υπάρχουν πολλά μειονεκτήματα που πρέπει να ληφθούν υπόψη. Σε μεγαλύτερα δίκτυα όπου απαιτούνται περισσότερα άλματα μεταξύ καταναλωτή και παραγωγού οι κόμβοι του δικτύου αυξάνονται εκθετικά οπότε υπάρχει αρνητική επίδραση στην απόδοση του δικτύου και δυσκολία στο configuration της τοπολογίας στο σύστημα. Ήδη σε αυτή την εργασία πετύχαμε με τον ίδιο αριθμό κόμβων 11 άλματα μεταξύ καταναλωτή και παραγωγού στην Dumbbell ενώ μόλις 4 στην Star-Bus. Για τη διεξαγωγή του συγκεκριμένου πειράματος χρησιμοποιήσαμε Dumbbell τοπολογία αντίστοιχης καθυστέρησης και ενδιάμεσων κόμβων με τη Star-Bus. Τέλος, αυτή η τοπολογία απαιτεί μεγάλο κόστος εγκατάστασης και το γεγονός ότι τα περισσότερα δεδομένα που διαδίδονται συσσωρεύονται στον κεντρικό κόμβο μπορεί να οδηγήσει σε κατάσταση υπερφόρτωσης δικτύου.



Σχήμα 3.6: Παράσταση της συνολικής καθυστέρησης του δικτύου συναρτήσει της θέσης του σφάλματος στην τοπολογία Δέντρου ή Star-Bus

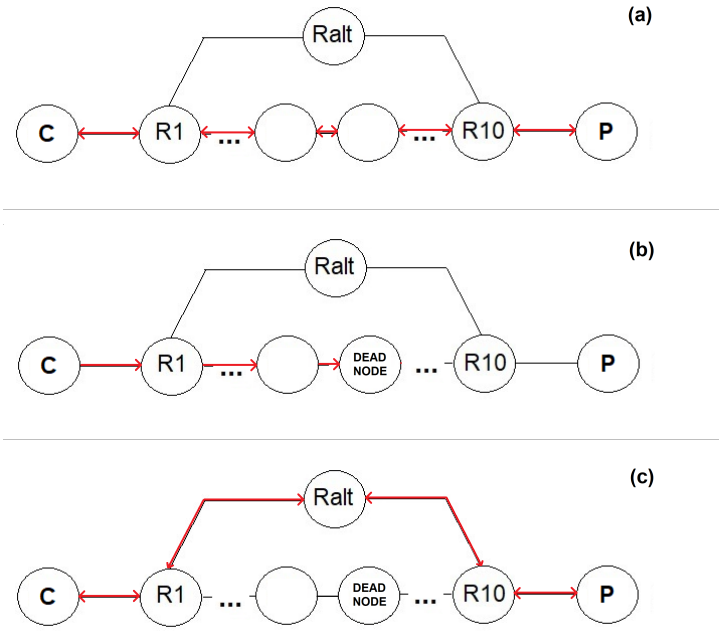
3.2.5 Πείραμα Πέμπτο – Στρατηγική Δρομολόγησης / Διακοπτόμενη Σύνδεση

Στο τελευταίο πείραμα θα αξιολογήσουμε τις διαφορετικές στρατηγικές δρομολόγησης των NDN δικτύων. Σαν τοπολογία θα χρησιμοποιηθεί η παραλλαγή του κλασσικού Dumbbell μοντέλου έτσι ώστε να δώσουμε στο δίκτυο περισσότερες από μια πιθανές διαδρομές για την προώθηση πακέτων. Στο συγκεκριμένο σενάριο δεν έχει νόημα να ασχοληθούμε επίσης με τις δυνατότητες προσωρινής αποθήκευσης οπότε έχουμε απενεργοποιήσει κάθε αντίστοιχη δραστηριότητα. Τέλος, για αυτό το πείραμα αναιρούμε το αυστηρό χρονοδιάγραμμα για αποστολή των Interests και ακολουθούμε μία back-to-back λογική αποστολής πακέτων, δηλαδή το κάθε Interest θα αποστέλλεται στο δίκτυο αμέσως μόλις ληφθεί το προηγούμενο ζητούμενο Data πακέτο και όχι ανά τρία δευτερόλεπτα όπως έχει ρυθμιστεί για τα υπόλοιπα πειράματα. Το κάθε πείραμα διαρκεί όσο χρειαστεί για να ικανοποιηθούν και τα είκοσι πακέτα ενδιαφέροντος των καταναλωτών

Το σενάριο που θα υπάρξει στο συγκεκριμένο πείραμα είναι ότι μετά την ικανοποίηση του δέκατου Interest θα υπάρξει διακοπή της σύνδεσης ενός κόμβου-δρομολογητή της συντομότερης διαδρομής. Η πτώση της σύνδεσης συμβαίνει ταυτόχρονα με την αποστολή του ενδέκατου πακέτου ενδιαφέροντος οπότε το σύστημα δεν προλαβαίνει να ενημερωθεί για την αλλαγή της τοπολογίας ακαριαία. Ακόμα και με την παραδοχή ότι τα LSAs ενημερώνονται αμέσως, χρειάζεται χρόνος προκειμένου να υπολογιστεί νέα διαδρομή προς τα διαφημιζόμενα πακέτα δεδομένων και να καταχωρηθεί στα FIBs όλων των κόμβων. Το mini-NDN δεν προωθεί πακέτα σε μη καταχωρημένες διαδρομές παρά μόνο μεταξύ γειτονικών κόμβων.

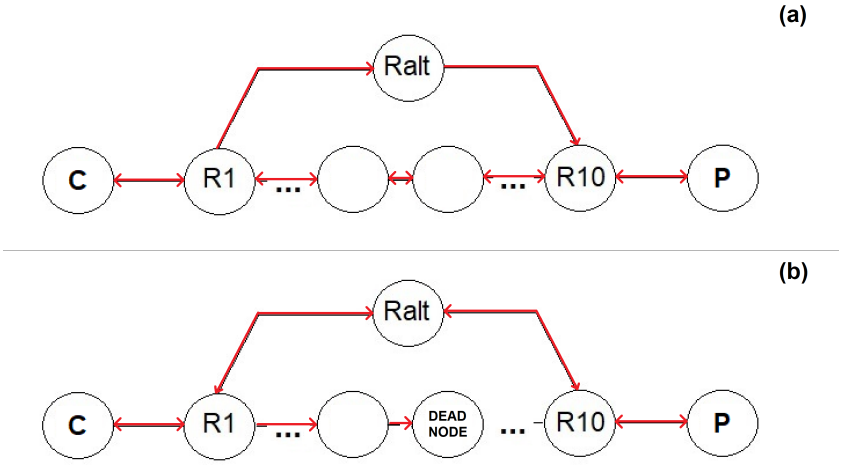
Οι στρατηγικές δρομολόγησης που θα αναλυθούν είναι οι “Best Route”, “Multicast” και “Access”. Η πρώτη υπολογίζει την βέλτιστη διαδρομή και στέλνει τα Interests μόνο από αυτή, η “Multicast” χρησιμοποιεί την τεχνική δρομολόγησης πολλαπλών διαδρομών, δηλαδή όσο δεν έχει βρεθεί το ζητούμενο Data Packet οι κόμβοι διαδίδουν το Interest στα FIBs των γειτόνων τους και η “Access” χρησιμοποιεί μία μίξη και των δύο προηγουμένων χρησιμοποιώντας την τεχνική πολλαπλών διαδρομών μόνο στο πρώτο πακέτο και στην συνέχεια ακολουθώντας τη βέλτιστη από αυτές για τα υπόλοιπα.

Στη τοπολογία του πειράματος θα συμβούν τα παρακάτω συμβάντα για τις τρεις στρατηγικές. Στην Εικόνα 3.2 φαίνεται η πορεία των γεγονότων στην “Best Route” στρατηγική. Η βέλτιστη διαδρομή αποτελεί αυτή με το μικρότερο βάρος οπότε τα δέκα πρώτα πακέτα ενδιαφέροντος θα σταλούν από εκείνη. Το πακέτο ενδιαφέροντος #11 θα σταλεί από την προηγουμένως υπολογισμένη βέλτιστη διαδρομή αλλά όταν φτάσει στη πτώση σύνδεσης θα απορριφθεί από το δίκτυο. Αυτό ωθεί το δίκτυο σε αδράνεια μέχρι να περάσει το χρονικό όριο του Timeout όπου και υπολογίζει ξανά τη νέα βέλτιστη διαδρομή. Πλέον τα πακέτα στέλνονται μέχρι να ικανοποιηθούν όλα από τη μοναδική πιθανή διαδρομή που ορίζει το μονοπάτι με το μεγαλύτερο βάρος.



Εικόνα 3.2: Best Route (a) Δρομολόγηση Interests πριν το σφάλμα, (b) Απόρριψη Interest λόγω σφάλματος, (c) Εύρεση νέας διαδρομής για δρομολόγηση των υπολοίπων Interest

Στην “Multicast” τα Interests διαδίδονται προς κάθε διαδρομή προκειμένου να ικανοποιηθεί έστω και ένα (Εικόνα 3.3α). Μέχρι το δέκατο πακέτο ικανοποιούνται πρώτα τα πακέτα ενδιαφέροντος που ακολουθούν τη διαδρομή με το μικρότερο βάρος ενώ από το ενδέκατο και μετά τα μόνα που επιστρέφουν δεδομένα είναι αυτά που ακολούθησαν την δεύτερη, μεγαλύτερη διαδρομή (Εικόνα 3.3β).

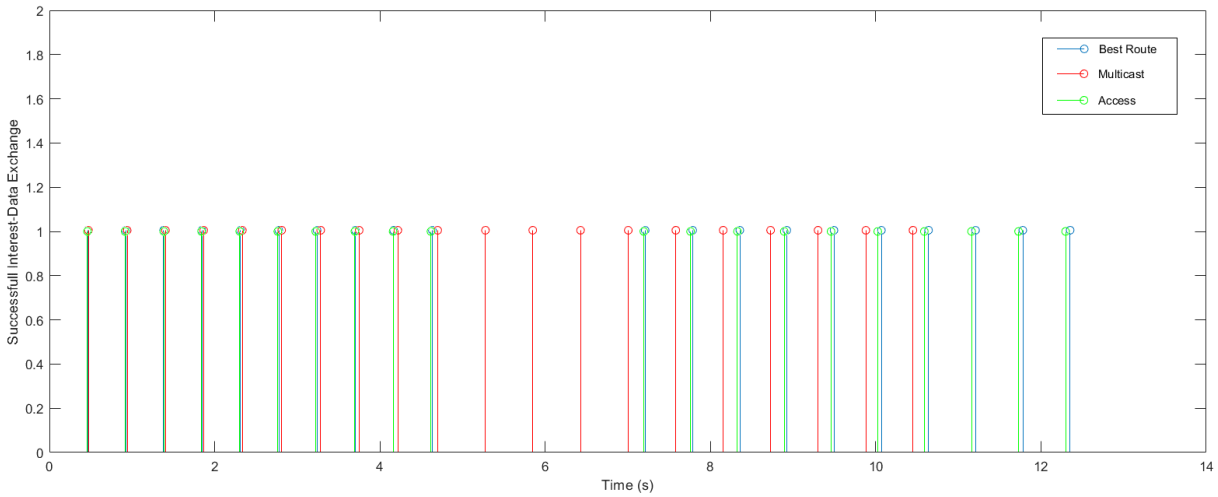


Εικόνα 3.3: Multicast (a) Επιστροφή δεδομένων από τη συντομότερη διαδρομή πριν το σφάλμα, (b) Επιστροφή δεδομένων από την άλλη διαθέσιμη διαδρομή μετά το σφάλμα

Τέλος, η “Access” χρησιμοποιεί την τεχνική multipath forwarding με το πρώτο Interest όπου και εκτιμάει την διαδρομή με το μικρότερο βάρος ως βέλτιστη και ενεργή. Στο πακέτο ενδιαφέροντος #11 το δίκτυο μένει αδρανές μέχρι το ορισμένο Timeout καθώς ο κόμβος που κατέρρευσε δημιουργεί φαινόμενο “μαύρης τρύπας” και ο καταναλωτής στέλνει ξανά το ίδιο Interest με την τεχνική πολλαπλών διαδρομών. Ενημερώνεται η νέα διαδρομή και τα υπόλοιπα Interests ακολουθούν αυτή μέχρι να ικανοποιηθούν όλα.

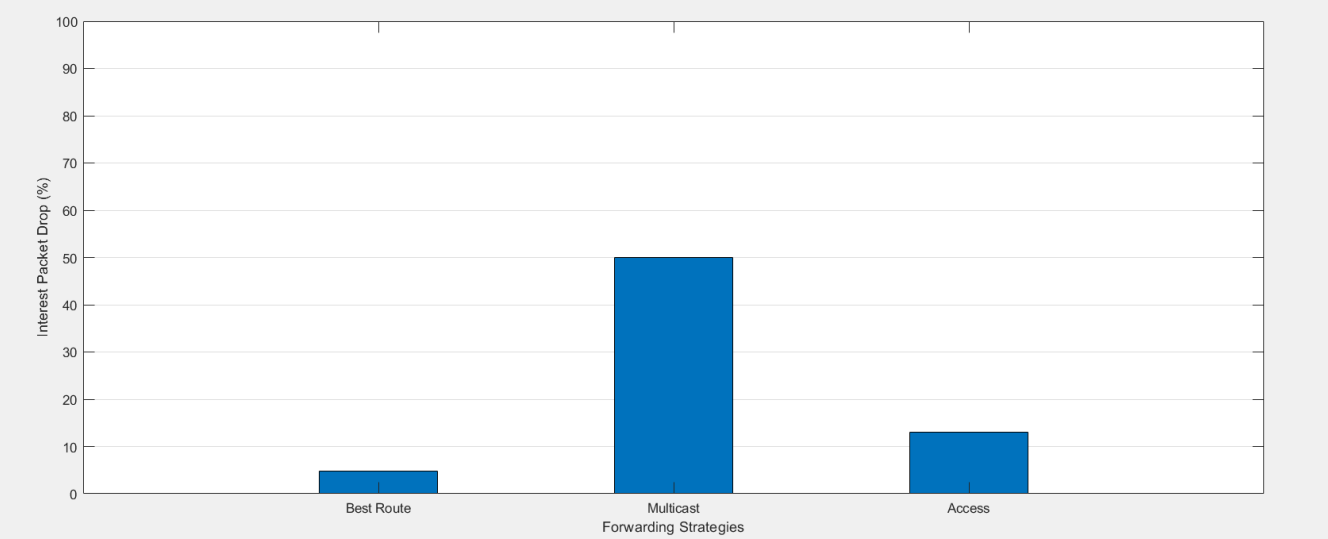
Στις μετρήσεις του πειράματος (Σχήμα 3.7) φαίνεται η χρονική διάρκεια των τριών στρατηγικών με την Multicast να οδηγεί σε ταχύτερη εκτέλεση του πειράματος από τις υπόλοιπες δύο. Η αιτία βρίσκεται στο γεγονός ότι οι υπόλοιπες δύο στρατηγικές υπέστησαν την καθυστέρηση των 2s που έχει οριστεί ως χρόνος Timeout. Στην περίπτωση που το σύστημα επέστρεφε NACK, το πείραμα στα δύο σενάρια των “Best Route” και “Access” θα επιβαρυνόταν μόνο την καθυστέρηση της διάδοσης του Interest και της επιστροφής του NACK, που έχει πολύ μικρότερη τιμή από αυτή του Timeout. Παρόλα αυτά στην περίπτωση που έχει συμβεί πτώση σύνδεσης χωρίς να προλάβει να

ενημερωθεί το υπόλοιπο δίκτυο δεν υπάρχει τρόπος να επέστρεφε απόκριση διαφορετική του Timeout. Τέλος, η μικρή χρονική διαφορά στην αποστολή πακέτων μεταξύ των τριών στρατηγικών οφείλεται στην είσοδο επιπλέον υπολογιστικού φόρτου στο σύστημα λόγω του μεγέθους της κεφαλίδας και της αντιγραφής και καταχώρησης των Interest στις δομές δεδομένων των κόμβων όταν χρησιμοποιείται η τεχνική της προώθησης πολλαπλών διαδρομών.



Σχήμα 3.7: Παράσταση της συνολικής χρονικής διάρκειας των διαφόρων στρατηγικών δρομολόγησης

Ενώ η “Multicast” στρατηγική φαίνεται να έχει τα καλύτερα χρονικά αποτελέσματα στο συγκεκριμένο σενάριο, έρχεται με μερικά μειονεκτήματα όπως το μεγάλο μέγεθος στις κεφαλίδες των Interests αλλά και το μεγάλο ποσοστό χαμένων πακέτων στο δίκτυο (Σχήμα 3.8). Τα συγκεκριμένα πακέτα καταλαμβάνουν μεγάλο μέρος του διαθέσιμου Bandwidth και σε μεγαλύτερα δίκτυα μπορεί να δημιουργήσουν προβλήματα συμφόρησης. Στην τοπολογία του πειράματος οι πιθανές διαδρομές ήταν μόνο δύο αλλά σε μεγαλύτερα συστήματα το ποσοστό των απορριφθέντων πακέτων ανεβαίνει ραγδαία. Όσον αφορά τη στρατηγική “Access” φαίνεται να είναι αποδοτική επιλογή όταν το πρόβλημα σε κάποια σύνδεση παρουσιάζεται πριν ξεκινήσουν να στέλνονται τα πακέτα ενδιαφέροντος, καθώς έχει ενσωματωμένη στρατηγική ανίχνευσης σφάλματος. Η “Best Route” στρατηγική που αποτελεί και την πιο διαδεδομένη στρατηγική στα NDN δίκτυα είναι αποδοτική στις τοπολογίες με σταθερή σύνδεση όπου ο σκοπός είναι να προωθηθούν τα πακέτα από την πιο σύντομη διαδρομή και με το μικρότερο δυνατό φόρτο.



Σχήμα 3.8: Παράσταση της απόρριψης πακέτων ενδιαφέροντος για διαφορετικές στρατηγικές δρομολόγησης

Συμπεράσματα

Συνοψίζοντας, στην παρούσα διπλωματική εργασία παρουσιάστηκε η λειτουργία των NDN δικτύων που αποτελούν μία από τις πιο κυρίαρχες αρχιτεκτονικές δικτύου του μέλλοντος και προσφέρει αποδοτική αντιμετώπιση των σύγχρονων προκλήσεων ταυτόχρονα με μεγάλο χώρο για περαιτέρω ανάπτυξη. Συγκρίθηκαν οι λειτουργίες τους με αυτές των IP και ορίστηκαν πλεονεκτήματα όπως η ενσωματωμένη ασφάλεια του δικτύου, η λειτουργία in-network caching και η content-oriented προσέγγιση της ανταλλαγής πληροφορίας. Έτσι δημιουργήσαμε τις δικές μας τοπολογίες, αναπτύξαμε προσαρμοσμένο κώδικα για τις λειτουργίες των κόμβων του δικτύου και για τις ενέργειες του δικτύου κατά τη διάρκεια των πειραμάτων και πραγματοποιήσαμε πειράματα στο mini-NDN emulator για την αξιολόγηση της απόδοσης και την εξαγωγή συμπερασμάτων για τις λειτουργίες και τα πλεονεκτήματα των NDN δικτύων.

Παράλληλα με την επιτυχή διεξαγωγή των πειραμάτων, η εργασία συνείσφερε στην αναβάθμιση του mini-NDN προγράμματος. Καθώς αυτό είναι λογισμικό προς ανάπτυξη, η εργασία υπέδειξε ορισμένες αδυναμίες και σφάλματα και συμπερασματικά μπορεί να λειτουργήσει σαν υλικό που εμπλουτίζει τη βιβλιογραφία του συγκεκριμένου εργαλείου. Τέλος, προσφέρει τυποποιημένο κώδικα για τη συλλογή και απεικόνιση αποτελεσμάτων και τα πειράματα που πραγματοποιήθηκαν μπορούν να αποτελέσουν βάση για ανάπτυξη νέου κώδικα με σκοπό την περαιτέρω μελέτη των NDN δικτύων.

Βιβλιογραφία

[1] J. Afanasyev, A., Yingdi , Y., Burke, j., Polterock, “The Second Named Data Networking Community Meeting (NDNcomm 2015),” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 46, no. 1, pp. 58–63, 2016.

[2] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, “A survey of information-centric networking,” *IEEE Commun. Mag.*, vol. 50, no. 7, pp. 26–36, 2012, doi: 10.1109/MCOM.2012.6231276.

[3] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, “A survey of information-centric networking,” *IEEE Commun. Mag.*, vol. 50, no. 7, pp. 26–36, 2012, doi: 10.1109/MCOM.2012.6231276.

[4] J. Bi, “Why ICN ? IP is already a great success How to Design an ICN Architecture ?,” pp. 1–4, 2015.

[5] N. Bigdeli and M. Haeri, “ARM-PFC an optimized AQM congestion controller in TCP/IP networks,” *Iran. J. Sci. Technol. Trans. B Eng.*, vol. 31, no. 6, pp. 663–678, 2007.

[6] F. A. Feltus and S. Ficklin, “Session 1 : NDN Trial Deployments Publishing Genomics Datasets into NDN Testbed and Integrating with Data-Centric Ecosystems for Large-scale Data-Intensive Science,” pp. 1–16.

[7] J. J. Garcia-Luna-Aceves and M. M. Barijough, “Efficient multicasting in content-centric networks using datagrams,” *2016 IEEE Glob. Commun. Conf. GLOBECOM 2016 - Proc.*, 2016, doi: 10.1109/GLOCOM.2016.7841781.

[8] A. K. M. M. Hoque, S. O. Amin, A. Alyyan, B. Zhang, L. Zhang, and L. Wang, “NLSR: Named-data Link State,” *Proc. 3rd ACM SIGCOMM Work. Information-centric Netw. - ICN ’13*, p. 15, 2013.

[9] E. Kalogeiton, T. Kolonko, and T. Braun, “A topology-oblivious routing protocol for NDN-VANETs,” *Ann. des Telecommun. Telecommun.*, vol. 73, no. 9–10, pp. 577–587, 2018, doi: 10.1007/s12243-018-0661-4.

[10] I. A. Kapetanidou, C. A. Sarros, and V. Tsaoussidis, “Reputation-based trust approaches in Named Data Networking,” *Futur. Internet*, vol. 11, no. 11, pp. 1–18, 2019, doi: 10.3390/fi11110241.

[11] I. Moiseenko and L. Zhang, “Consumer-producer API for named data networking,” *ICN 2014 - Proc. 1st Int. Conf. Information-Centric Netw.*, pp. 177–178, 2014, doi: 10.1145/2660129.2660158.

[12] R. Mortier and S. Riley, “Software Defined Networking - Computerphile,” *Computerphile*, 2016.

[13] B. Nour, F. Li, H. Khelifi, H. Moungla, and A. Ksentini, “Coexistence of ICN and IP networks: An NFV as a service approach,” *2019 IEEE Glob. Commun. Conf. GLOBECOM 2019 - Proc.*, 2019, doi: 10.1109/GLOBECOM38437.2019.9013881.

[14] C. A. Sarros, V. Demiroglou, and V. Tsaoussidis, “Intermittently-connected IoT devices: Experiments with an NDN-DTN architecture,” *2021 IEEE 18th Annu. Consum. Commun. Netw. Conf. CCNC 2021*, 2021, doi: 10.1109/CCNC49032.2021.9369578.

[15] C. A. Sarros *et al.*, “Connecting the Edges: A Universal, Mobile-Centric, and Opportunistic Communications Architecture,” *IEEE Commun. Mag.*, vol. 56, no. 2, pp. 136–143, 2018, doi: 10.1109/MCOM.2018.1700325.

[16] C. A. Sarros *et al.*, “ICN-based edge service deployment in challenged networks,” *ICN 2017 - Proc. 4th ACM Conf. Inf. Centric Netw.*, pp. 210–211, 2017, doi: 10.1145/3125719.3132096.

[17] D. W. Sudiharto, A. Herutomo, and Y. N. Rohmah, “The comparison of forwarding strategies between best route, multicast, and access on named data networking (NDN). Case study: A node compromised by the prefix hijack,” *J. Commun.*, vol. 12, no. 7, pp. 426–432, 2017, doi: 10.12720/jcm.12.7.426-432.

[18] D. Syrivelis *et al.*, “Pursuing a software defined information-centric network,” *Proc. - Eur. Work. Softw. Defin. Networks, EWSDN 2012*, no. October, pp. 103–108, 2012, doi: 10.1109/EWSDN.2012.20.

[19] D. Syrivelis *et al.*, “Pursuing a software defined information-centric network,” *Proc. - Eur. Work. Softw. Defin. Networks, EWSDN 2012*, no. October, pp. 103–108, 2012, doi: 10.1109/EWSDN.2012.20.

[20] S. Tibbits, “Designed for Change,” *Am. Sci.*, vol. 109, no. 5, p. 304, 2021, doi: 10.1511/2021.109.5.304.

[21] B. W. and C. A. W. and A. A. and L. Z. and D. R. O. and C. Tschudin, “RFC 8793 Information-Centric Networking (ICN): Content- Centric Networking (CCNx) and Named Data Networking (NDN) Terminology Abstract,” pp. 1–17, 2020.

[22] C. Yi, A. Afanasyev, L. Wang, B. Zhang, and L. Zhang, “Adaptive forwarding in named data networking,” *Comput. Commun. Rev.*, vol. 42, no. 3, pp. 62–67, 2012, doi: 10.1145/2317307.2317319.

[23] B. Zhang, “NFD Development Progress.”

[24] L. Zhang *et al.*, “Named data networking,” *Comput. Commun. Rev.*, vol. 44, no. 3, pp. 66–73, 2014, doi: 10.1145/2656877.2656887.

Επιπλέον Πηγές

[25] https://www.ietfjournal.org/ipv4ipv6-coexistence-and-transition/

[26] https://www.google.com/intl/en/ipv6/statistics.html

[27] https://www.nsf.gov/pubs/2010/nsf10528/nsf10528.htm

[28] https://tools.ietf.org/html/rfc8793

[29] https://api.deepai.org/publication-download-pdf/modifying-nfd-for-ndn-experimentation-a-review