ΔΗΜΟΚΡΙΤΕΙΟ ΠΑΝΕΠΙΣΤΗΜΕΙΟ ΘΡΑΚΗΣ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΛΟΓΙΣΜΙΚΟΥ ΚΑΙ ΑΝΑΠΤΥΞΗΣ ΕΦΑΡΜΟΓΩΝ

Μελέτη απόδοσης νέων δικτυακών αρχιτεκτονικών στο Διαδίκτυο των Αντικειμένων

Δεμίρογλου Βασίλειος
ΑΜ: 56774

Διπλωματική Εργασία

Επιβλέπων Καθηγητής: Τσαουσίδης Βασίλειος

Ξάνθη 2020

Ευχαριστίες

Η παρούσα διπλωματική εργασία διεκπεραιώθηκε στο πλαίσιο περάτωσης των προπτυχιακών μου σπουδών. Στο σημείο αυτό θα ήθελα να ευχαριστήσω τον επιβλέποντα της διπλωματικής μου εργασίας κ. Βασίλειο Τσαουσίδη, Καθηγητή στο τμήμα Η.Μ.Μ.Υ. του Δ.Π.Θ., για την εμπιστοσύνη που μου έδειξε, καθώς και για την επιστημονική στήριξη και καθοδήγηση που μου παρείχε σε όλη τη διάρκεια εκπόνησής της.

Ευχαριστίες οφείλω στα υπόλοιπα μέλη της εξεταστικής επιτροπής της εργασίας, στον κ. Νικόλαο Μητιανούδη, Αναπληρωτή Καθηγητή στο τμήμα Η.Μ.Μ.Υ. του Δ.Π.Θ., και στον κ. Παύλο Εφραιμίδη, Αναπληρωτή Καθηγητή στο τμήμα Η.Μ.Μ.Υ. του Δ.Π.Θ., για το χρόνο που αφιέρωσαν για την αξιολόγηση της διπλωματικής εργασίας μου.

Επίσης θερμές ευχαριστίες οφείλω στον κ. Χρήστο-Αλέξανδρο Σάρρο, υποψήφιο Διδάκτορα στο τμήμα Η.Μ.Μ.Υ. του Δ.Π.Θ., καθώς και στον κ. Χρήστο Χατζηανδρέογλου, Ερευνητή και Μηχανικό, για τις σημαντικές υποδείξεις και συμβουλές τους καθώς και για τη συμβολή τους στην επίλυση τεχνικών προβλημάτων.

Τέλος, οφείλω να ευχαριστήσω την οικογένεια μου που με υποστηρίζει καθημερινά.

Περιεχόμενα

ΚΕΦΑΛΑΙΟ 1: Εισαγωγή

1.1 Περιγραφή του θέματος

Το Διαδίκτυο των Αντικειμένων (Internet of Things, IoT) αποτελεί μια επέκταση του Διαδικτύου (Internet) στον πραγματικό κόσμο. Οι συνεχείς αλλαγές και εξελίξεις του IoT και η εμφάνιση όλο και περισσότερων κινητών, «έξυπνων» και διασυνδεδεμένων συσκευών δημιουργούν προκλήσεις τις οποίες η από άκρο σε άκρο αρχιτεκτονική του Διαδικτύου αδυνατεί να αντιμετωπίσει. Κατά συνέπεια, δημιουργείται η ανάγκη διερεύνησης νέων δικτυακών αρχιτεκτονικών ικανών να αντιμετωπίσουν τις καθυστερήσεις, τις διακοπές των συνδέσεων και την ετερογένεια των δικτύων. Η παρούσα εργασία εξετάζει και συγκρίνει την αποδοτικότητα της αρχιτεκτονικής DTN (Delay/Disruption-Tolerant Networking,) και της αρχιτεκτονικής UMOBILE. Στην αρχιτεκτονική DTN, ένας κόμβος παράγει δεδομένα, τα αποθηκεύει και τα αποστέλλει μόλις υπάρξει σύνδεση με κάποιον άλλο DTN κόμβο. Η αρχιτεκτονική UMOBILE αποτελεί μια ενοποίηση της Πληροφοριοκεντρικής αρχιτεκτονικής (Information Centric Networking, ICN) με την DTN. Στην αρχιτεκτονική ICN, κατονομάζονται τα δεδομένα και όχι οι τελικοί κόμβοι εστιάζοντας στο περιεχόμενο των δεδομένων και όχι στην τοποθεσία τους. Έτσι, στην αρχιτεκτονική UMOBILE, η πρόσβαση στα δεδομένα είναι ανεξάρτητη από την ύπαρξη από άκρο σε άκρο σύνδεσης.

1.2 Στόχοι της εργασίας

Βασικός στόχος της εργασίας είναι η εξέταση της απόδοσης των αρχιτεκτονικών DTN και UMOBILE σε πραγματικά IoT σενάρια, στα οποία οι κόμβοι αισθητήρων είναι τοποθετημένοι σε δυσπρόσιτες περιοχές και η μεταφορά των δεδομένων στο Διαδίκτυο γίνεται μέσω κινητών κόμβων. Για την επίτευξη του βασικού στόχου χρειάστηκε η υλοποίηση επιμέρους στόχων, όπως:

α) Η επιλογή των κατάλληλων εργαλείων και συμβατού υλικού,

β) η σχεδίαση των IoT σεναρίων και των αντίστοιχων τοπολογιών,

γ) η ανάπτυξη ισοδύναμων εφαρμογών για τις δύο αρχιτεκτονικές και

δ) η ρύθμιση και διεξαγωγή πειραμάτων που να εξομοιώνουν τα IoT σενάρια.

1.3 Μεθοδολογία και συμπεράσματα

Η εξέταση της απόδοσης των αρχιτεκτονικών DTN και UMOBILE σε πραγματικά IoT σενάρια προϋποθέτει τον σχεδιασμό αντίστοιχων εργαστηριακών πειραμάτων. Έτσι, επιλέχθηκε η δημιουργία δύο διαφορετικών περιοχών-δικτύων: στη μία περιοχή βρίσκονται οι αισθητήρες και ο κόμβος παραγωγός, ενώ στη δεύτερη βρίσκεται ο καταναλωτής της εφαρμογής. Η μεταφορά αφενός των αιτημάτων των καταναλωτών και αφετέρου των δεδομένων των αισθητήρων από τη μία περιοχή στην άλλη, πραγματοποιείται από έναν ενδιάμεσο κόμβο, ο οποίος μετακινείται μεταξύ των δυο περιοχών και συνδέεται διαδοχικά στα δύο δίκτυα.

Για την υλοποίηση αυτού του σεναρίου επιλέχθηκαν κατάλληλες πλατφόρμες, πρωτόκολλα δικτύωσης και τοπολογίες. Επιπλέον, αξιοποιήθηκε ο διαθέσιμος εργαστηριακός εξοπλισμός και αναπτύχθηκαν εφαρμογές για τη διεξαγωγή ισοδύναμων, για την εκάστοτε αρχιτεκτονική, πειραμάτων με κοινές μετρικές και παραμέτρους.

Βάσει των αποτελεσμάτων των πειραμάτων που διεξήχθησαν, η αρχιτεκτονική UMOBILE, λόγω του caching που εξασφαλίζεται από το NDN (Named Data Networking) stack, υπερτερεί της αρχιτεκτονικής DTN σε απόδοση. Η αρχιτεκτονική UMOBILE αποτελεί μια αποδοτική λύση για τη διασύνδεση απομακρυσμένων περιοχών που μπορεί να εφαρμοστεί με μεγάλη αποτελεσματικότητα στο Διαδίκτυο των Αντικειμένων.

1.4 Οργάνωση της εργασίας

Η παρούσα εργασία οργανώνεται σε έξι κεφάλαια. Στο δεύτερο κεφάλαιο παρουσιάζονται βασικές έννοιες που σχετίζονται με την εργασία. Αρχικά, περιγράφεται ο τρόπος λειτουργίας του Διαδικτύου και το όραμα του IoT. Κατόπιν, αναλύονται τα βασικά στοιχεία και η λειτουργία της DTN και NDN αρχιτεκτονικής.

Στο τρίτο κεφάλαιο αναλύονται τα οφέλη που προσφέρουν οι DTN και NDN αρχιτεκτονικές στο IoT, με αναφορές σε σχετικές έρευνες και μελέτες. Επιπλέον, περιγράφεται η αδυναμία της NDN αρχιτεκτονικής στα κινητά δίκτυα και η αντιμετώπιση της συγκεκριμένης αδυναμίας μέσω της UMOBILE αρχιτεκτονικής.

Στο τέταρτο κεφάλαιο γίνεται μια ανασκόπηση στις σχετικές πλατφόρμες κάθε αρχιτεκτονικής και αιτιολογείται η επιλογή των πιο κατάλληλων απ' αυτές.

Στο πέμπτο κεφάλαιο υποδεικνύονται τα σενάρια και η τοπολογία του δικτύου της εργασίας. Έπειτα, αναλύεται το υλικό και η λειτουργία κάθε κόμβου της τοπολογίας και επισημαίνονται οι τεχνικές δυσκολίες που υπήρξαν κατά τη ρύθμιση και τη διεξαγωγή των πειραμάτων.

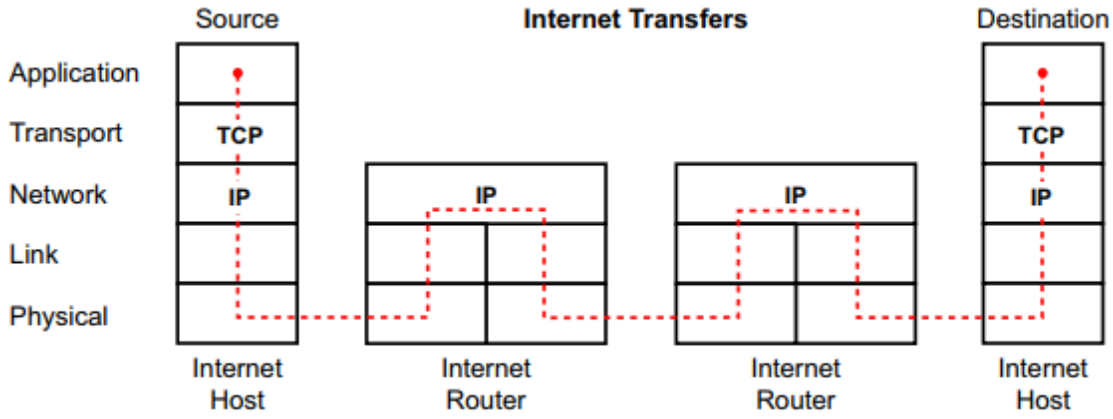
Στο έκτο κεφάλαιο αναλύονται τα πειράματα και σχολιάζονται τα αποτελέσματά τους. Στη συνέχεια, καταγράφονται τα κυριότερα συμπεράσματα και οι μελλοντικές προοπτικές της εργασίας. Τέλος, εκτίθενται η βιβλιογραφία και το παράρτημα.

ΚΕΦΑΛΑΙΟ 2: Θεωρητικό υπόβαθρο

2.1 Διαδίκτυο

Το Διαδίκτυο σχεδιάστηκε με σκοπό την παροχή από άκρο σε άκρο επικοινωνίας μεταξύ των υπολογιστών παντού – ακόμα και στο διάστημα [1]. Η ταχεία εξέλιξή του οδήγησε στην επέκταση της λειτουργίας του σε πολλές καθημερινές δραστηριότητες των ανθρώπων, οι οποίες πλέον πραγματοποιούνται διαδικτυακά (π.χ. εκπαίδευση, ψυχαγωγία, ενημέρωση, αγορές). Λόγω της μεγάλης χρησιμότητάς του θεωρείται σημαντική η συνεχής προσπάθεια βελτίωσής και αντιμετώπισης των δυσλειτουργιών του.

Οι κόμβοι του Διαδικτύου χρησιμοποιούν τα πρωτόκολλα TCP (Transmission Control Protocol) και IP (Internet Protocol). Το πρωτόκολλο TCP εκτελείται μόνο στους τελικούς κόμβους μιας διαδρομής εξασφαλίζοντας αξιόπιστη από άκρο σε άκρο επικοινωνία. Η επικοινωνία μεταξύ ενός αποστολέα και ενός παραλήπτη προϋποθέτει την εγκαθίδρυση μιας TCP σύνδεσης. Το πρωτόκολλο IP εκτελείται σε όλους τους κόμβους και δρομολογεί τα αυτοδύναμα πακέτα IP. Κάθε hop σε μια διαδρομή μπορεί να χρησιμοποιήσει ένα διαφορετικό πρωτόκολλο επιπέδου συνδέσμου και φυσικού επιπέδου.



Εικόνα 1. Μεταφορά πακέτων στο Διαδίκτυο [2]

Η μεταφορά ενός μηνύματος στο Διαδίκτυο ξεκινάει με τον κατακερματισμό του σε TCP segments. Κάθε TCP segment ενθυλακώνεται σε ένα πακέτο IP. Κάθε πακέτο IP αποστέλλεται μέσω δρομολογητών στον προορισμό του. Μόλις φθάσουν όλα τα πακέτα IP στον προορισμό, το πρωτόκολλο TCP τα τοποθετεί στη σωστή σειρά και ξαναδημιουργείται το αρχικό μήνυμα του αποστολέα.

2.2 Το Διαδίκτυο των Αντικειμένων

Το Διαδίκτυο των Αντικειμένων αναφέρεται στη διασύνδεση και την επικοινωνία «έξυπνων» συσκευών (αισθητήρων, «έξυπνων» τηλεφώνων και αυτοκινήτων, βιομηχανικών μηχανημάτων κ.τ.λ.) τόσο μεταξύ τους όσο και με το Διαδίκτυο. Με αυτόν τον τρόπο επεκτείνεται η χρήση του Διαδικτύου και αποκτά επίδραση στον χειρισμό, την παρακολούθηση και τον έλεγχο πραγματικών αντικείμενων που ενσωματώνουν συσκευές IoT. Αυτές οι διασυνδεδεμένες συσκευές συνδέουν τον φυσικό με τον ψηφιακό κόσμο, συμβάλλοντας τόσο στην απλοποίηση των υπαρχουσών όσο και στη δημιουργία νέων υπηρεσιών. Οι υπηρεσίες IoT συνεισφέρουν σε πολλούς τομείς, όπως στην παρακολούθηση της υγείας των ανθρώπων, στην προστασία του περιβάλλοντος, στο σχεδιασμό έξυπνων πόλεων, στην βελτίωση των βιομηχανικών αυτοματισμών και των μεταφορών.

Σε γενικές γραμμές, οι συσκευές IoT διακρίνονται σε συσκευές με αρκετούς πόρους (High-end IoT devices), όπως είναι τα smartphones, τα Raspberry pi computers κ.λπ., και σε συσκευές περιορισμένων πόρων (Low-end IoT devices) όπως μικροελεγκτές. Οι συσκευές περιορισμένων πόρων διαθέτουν μικρό μέγεθος, περιορισμένη υπολογιστική ισχύ, ελάχιστη μνήμη και τροφοδοτούνται από απλές μπαταρίες. Επίσης, αρκετές συσκευές είναι φορητές με αποτέλεσμα την δημιουργία δυναμικών τοπολογιών δικτύων με ευκαιριακές συνδέσεις. Η σύνδεση των συσκευών αυτών στο Διαδίκτυο προϋποθέτει την ύπαρξη ασφαλών συνδέσεων, καθώς η παραβίαση τους αποτελεί ένα σύνηθες φαινόμενο. Ακόμη λόγω της ανάγκης εξασφάλισης χαμηλής ενεργειακής κατανάλωσης

και του υψηλού κόστους σύνδεσης όλων των συσκευών στο Διαδίκτυο, πολλές συσκευές δε συνδέονται απευθείας σε αυτό αλλά συνδέονται σε κάποιο κόμβο-πύλη (που συνήθως είναι High-end IoT device). Μέσω του κόμβου-πύλης, οι συσκευές περιορισμένων πόρων προωθούν και λαμβάνουν πακέτα από το Διαδίκτυο. Λαμβάνοντας υπόψη αυτούς τους περιορισμούς, τον πολύ μεγάλο αριθμό διασυνδεδεμένων συσκευών και τις τεράστιες ποσότητες δεδομένων που μεταφέρουν, είναι επιτακτική η εύρεση αποδοτικών και αποτελεσματικών μεθόδων διασύνδεσης των συσκευών και έκθεσης τους στο Διαδίκτυο.

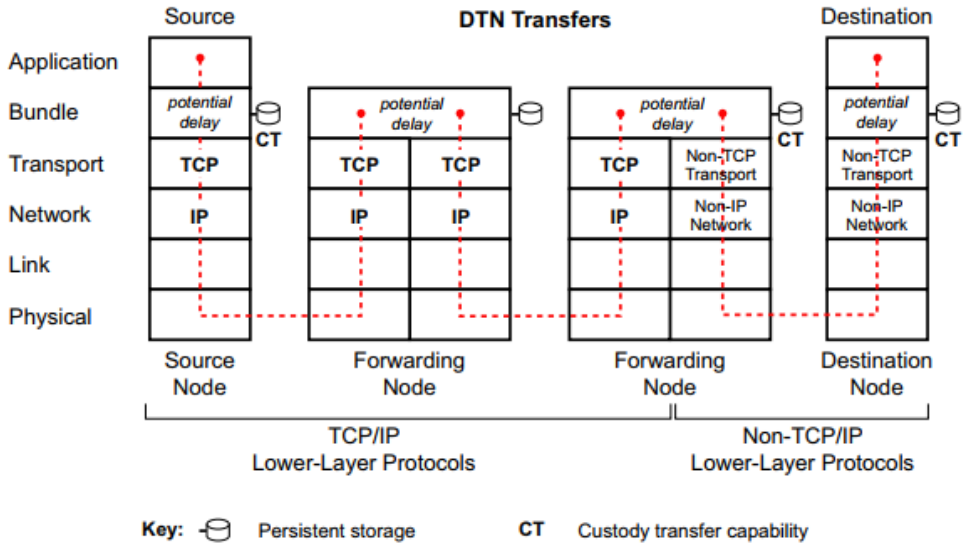
Το μοντέλο επικοινωνίας TCP/IP που χρησιμοποιείται από το Διαδίκτυο απαιτεί, τόσο ο αποστολέας όσο και ο παραλήπτης, να είναι συνδεδεμένοι ταυτόχρονα. Ωστόσο, σε σενάρια IoT, οι συσκευές περιορισμένων πόρων μπορεί συχνά είτε να εισέρχονται σε λειτουργία ύπνου (deep sleep mode) για εξοικονόμηση ενέργειας είτε να μετακινούνται στο χώρο. Η διακοπτόμενη επικοινωνία των συσκευών καθιστά συνήθως δύσκολη τη διατήρηση σταθερών συνδέσεων. Για αυτό το λόγο οι εφαρμογές IoT χρειάζεται να εφαρμόσουν νέες δικτυακές αρχιτεκτονικές, οι οποίες είναι ανεκτικές σε καθυστερήσεις και διακοπές.

2.3 DTN

Η αρχιτεκτονική δικτύων ανεκτικών σε καθυστερήσεις και διακοπές (Delay/Disruption Tolerant Networking, DTN) [3] έχει δημιουργηθεί για την αντιμετώπιση των προβλημάτων των απομακρυσμένων δικτύων, στα οποία η διασύνδεση μέσω του Διαδικτύου είναι ακατάλληλη. Αποτελεί μία αρχιτεκτονική επικάλυψης η οποία λειτουργεί πάνω από τις υπάρχουσες στοίβες πρωτοκόλλων διαφόρων δικτυακών αρχιτεκτονικών. Το DTN εισάγει τα επίπεδο δέσμης μεταξύ του επιπέδου μεταφοράς και του επιπέδου εφαρμογής, παρέχοντας τη λειτουργία της επίμονης αποθήκευσης μέσω του μηχανισμού αποθήκευσης και προώθησης (store and forward). Για την επίτευξη διαλειτουργικότητας μεταξύ ετερογενών δικτύων, η αρχιτεκτονική DTN χρησιμοποιεί μηχανισμό ονοματοδοσίας και διευθυνσιοδότησης, έτσι ώστε κάθε DTN κόμβος να

αναγνωρίζεται μέσω ενός αναγνωριστικού τελικού σημείου (Endpoint Identifier, EID). Οι κόμβοι μπορούν να αποθηκεύουν και να μεταφέρουν μηνύματα (που ονομάζονται bundles) στον προορισμό τους. Οι λεπτομέρειες της αρχιτεκτονικής DTN και του πρωτοκόλλου δέσμης (Bundle Protocol, BP) αναλύονται στα RFC (Request For Comments) 4838 [4] και RFC 5050 [5].

Ένας κόμβος DTN μπορεί συνήθως να αποτελείται από κάποια εφαρμογή χρήστη η οποία δημιουργεί, στέλνει και λαμβάνει δεδομένα χρησιμοποιώντας το πρωτόκολλο δέσμης. Το επίπεδο δέσμης υλοποιεί το πρωτόκολλο δέσμης καθώς και τη δρομολόγηση των δεσμών. Τα πακέτα αποθηκεύονται μέχρι να μεταφερθούν στον κατάλληλο DTN κόμβο. Τα επίπεδα πρωτοκόλλου χαμηλότερου επιπέδου καθορίζουν πότε είναι διαθέσιμη η σύνδεση και διασυνδέονται με το επίπεδο δέσμης χρησιμοποιώντας έναν προσαρμογέα επιπέδου σύγκλισης. Η στοίβα πρωτοκόλλων DTN ακολουθεί περίπου το μοντέλο στοίβας δικτύου OSI (Open Systems Interconnection). Σε ένα DTN, όλοι οι κόμβοι εφαρμόζουν τόσο το πρωτόκολλο δέσμης όσο και ένα πρωτόκολλο μεταφοράς κατώτερου στρώματος. Οι κόμβοι που προωθούν τα bundles μπορούν να εφαρμόσουν είτε τα ίδια είτε διαφορετικά πρωτόκολλα χαμηλότερου επιπέδου σε κάθε πλευρά της προώθησης [2].



Εικόνα 2. Μεταφορά bundle στο DTN [2]

Το πρωτόκολλο δέσμης είναι ένα πρωτόκολλο δικτύου το οποίο υλοποιεί όλες τις απαραίτητες λειτουργίες που απαιτούνται για την υποστήριξη της δικτύωσης, ανεκτικής σε καθυστερήσεις και διακοπές. Συνδέει διάφορα δίκτυα χρησιμοποιώντας μια μεθοδολογία αποθήκευσης, μεταφοράς και προώθησης. Συγκεκριμένα, το πρωτόκολλο δέσμης συγκεντρώνει τα δεδομένα της εφαρμογής, τα ενθυλακώνει σε bundles τα οποία αποθηκεύει και προωθεί σε άλλους διαθέσιμους DTN κόμβους. Τα βασικά χαρακτηριστικά του πρωτόκολλου δέσμης είναι τα εξής:

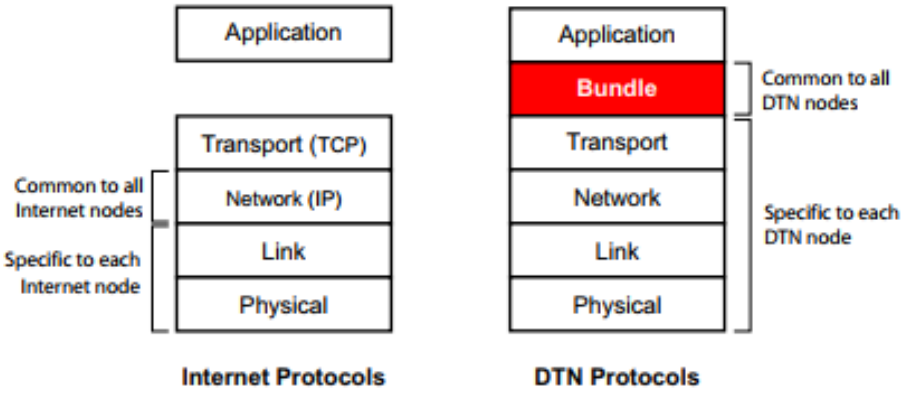
α) Αποθήκευση και προώθηση (Store-and-forward): Τα bundles προωθούνται σταδιακά και αποθηκεύονται σε κάθε ενδιάμεσο κόμβο μέχρι να φτάσουν στον προορισμό τους. Τα δεδομένα διαγράφονται από την μνήμη κάθε κόμβου, μόνο όταν προωθηθούν στον επόμενο παραλήπτη.

β) Μη-συνομιλητική ασύγχρονη επικοινωνία (Non-conversational Asynchronous Communication): Τα bundles, επιδιώκουν να συσσωρεύουν όσο το δυνατόν περισσότερες χρήσιμες πληροφορίες σε μια μονάδα δεδομένων ωφέλιμου φορτίου. Η διεργασία αυτή γίνεται πριν αποσταλούν στον αποδέκτη, ελαχιστοποιώντας τον αριθμό των ανταλλαγών επικοινωνίας που απαιτούνται για την ολοκλήρωση μιας συνεδρίας επικοινωνίας.

γ) Εναλλαγή εικονικών μηνυμάτων (Virtual message switching): Τα πακέτα είναι δομημένα ως μηνύματα μεταβλητού μήκους με θεωρητικά απεριόριστο μέγεθος. Κάθε μήνυμα μεταφέρεται ολόκληρο από κόμβο σε κόμβο. Έτσι οι ενδιάμεσοι δρομολογητές γνωρίζοντας το ακριβές ποσό των δεδομένων που πρέπει να μεταφέρουν, μπορούν να λαμβάνουν πιο ενημερωμένες αποφάσεις για τη δρομολόγηση, τον προγραμματισμό και την επιλογή διαδρομής.

δ) Αξιοπιστία και μεταφορά κηδεμονίας (Reliability and custody transfer): Κατά την έκδοση ενός bundle μπορεί να ζητηθεί αναφορά παράδοσης. Με την χρήση αυτού του μηχανισμού, μετά την παραλαβή της δέσμης από τον προορισμό της, ο αποδέκτης εκδίδει ένα bundle αναφοράς κατάστασης, το οποίο επιστρέφει στην πηγή, ενημερώνοντάς την ότι το bundle έχει παραδοθεί επιτυχώς. Παράλληλα η hop-by-hop αξιοπιστία της μετάδοσης παρέχεται μέσω ενός μηχανισμού μεταφοράς κηδεμονίας (custody transfer), ο οποίος επιτρέπει την ανάθεση ευθύνης αναμετάδοσης μεταξύ διαφορετικών κόμβων κατά μήκος της διαδρομής.

ε) Μηχανισμός ονοματοδοσίας και διευθυνσιοδότησης (Naming and Addressing Mechanism): Κάθε κόμβος στην αρχιτεκτονική DTN αναγνωρίζεται μέσω ενός EID. Ένα EID είναι ένα όνομα, που εκφράζεται χρησιμοποιώντας τη γενική σύνταξη των Ενιαίων Αναγνωριστικών Πόρων (Uniform Resource Identifiers, URI) [4], το οποίο χρησιμοποιείται για τον προσδιορισμό των τελικών σημείων προέλευσης και προορισμού ενός bundle καθώς και του τρέχοντα κηδεμόνα του bundle. Σε περίπτωση πολυεκπομπής, ένα EID μπορεί να αναφέρεται σε πολλαπλούς κόμβους DTN.



Εικόνα 3. Οι στοίβες πρωτοκόλλων Διαδικτύου και DTN [2]

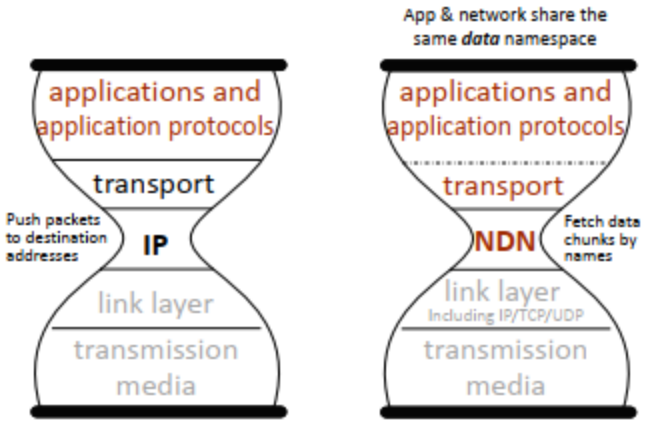
Κάθε bundle αποτελείται από τουλάχιστον δύο μπλοκ, ένα κύριο μπλοκ και ένα μπλοκ ωφέλιμου φορτίου. Το κύριο μπλοκ περιέχει πληροφορίες σχετικά με την πηγή του bundle, τον προορισμό, το χρόνο δημιουργίας και άλλες πληροφορίες επεξεργασίας. Το μπλοκ ωφέλιμου φορτίου περιέχει τα πραγματικά δεδομένα του bundle.

2.4 ICN και NDN

Η Πληροφοριοκεντρική δικτύωση έχει προσελκύσει μεγάλη προσοχή από την έρευνα δικτύων υπολογιστών και αποτελεί μία νέα δικτυακή αρχιτεκτονική η οποία στοχεύει να λύσει τα υπάρχοντα προβλήματα του σημερινού Διαδικτύου. Το Διαδίκτυο ακολουθεί μια δομή βασισμένη στο IP πρωτόκολλο, η οποία εξασφαλίζει την από άκρο σε

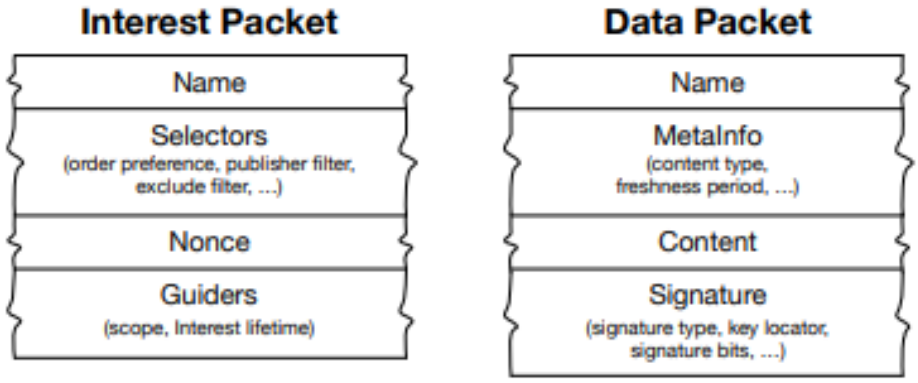
άκρο επικοινωνία μεταξύ των hosts. Η Πληροφοριοκεντρική δικτύωση, εστιάζει στο «τι» μεταφέρεται και όχι στο «που». Αυτό σημαίνει πως, όταν κάποιος χρήστης ζητήσει μια συγκεκριμένη πληροφορία, δεν θα ονομάσει το τελικό σημείο στο οποίο θεωρεί πως βρίσκονται αυτές οι πληροφορίες, αλλά θα ζητήσει το όνομα του συγκεκριμένου περιεχομένου. Το δίκτυο θα φροντίσει στη συνέχεια να δρομολογήσει το αίτημα με τέτοιο τρόπο ώστε ο χρήστης να λάβει ένα αντίγραφο του περιεχομένου. Βασικό όφελος της μετάβασης από το μοντέλο ονομασίας του κεντρικού υπολογιστή στο μοντέλο ονομασίας πληροφοριών είναι ότι η ανάκτηση πληροφοριών ξεκινά από τον δέκτη. Στο τρέχον Διαδίκτυο οι αποστολείς έχουν τον απόλυτο έλεγχο των δεδομένων που ανταλλάσσονται. Αντίθετα, στο ICN δεν μπορούν να ληφθούν δεδομένα, εκτός εάν ζητηθεί ρητά από τον παραλήπτη. Μετά την αποστολή ενός αιτήματος, το δίκτυο είναι υπεύθυνο για τον εντοπισμό της καλύτερης πηγής που μπορεί να παράσχει τις επιθυμητές πληροφορίες. Επομένως, η δρομολόγηση των αιτημάτων πληροφοριών επιδιώκει να βρει την καλύτερη πηγή πληροφοριών, βασισμένη σε ένα ανεξάρτητο από τη θέση όνομα [6].

Μια από τις πιο πρωτοποριακές υλοποιήσεις ICN είναι το πρωτόκολλο NDN (Named Data Networking) το οποίο έχει σχεδιαστεί για να συνδέει τον κόσμο των υπολογιστικών συσκευών μέσω της ονομασίας δεδομένων. Η στοίβα πρωτοκόλλων NDN διατηρεί το ίδιο σχήμα κλεψύδρας με το TCP/IP. Όπως το IP, το πρωτόκολλο δικτύου NDN εκτελεί παράδοση αυτοδύναμων πακέτων και τρέχει σε οποιοδήποτε μέσο μεταφοράς το οποίο μπορεί να μεταφέρει αυτοδύναμα πακέτα. Σε αντίθεση με το IP πρωτόκολλο στο οποίο μεταφέρονται IP πακέτα που περιέχουν τις διευθύνσεις του αποστολέα και του παραλήπτη, το NDN μεταφέρει ονοματισμένα κομμάτια δεδομένων.



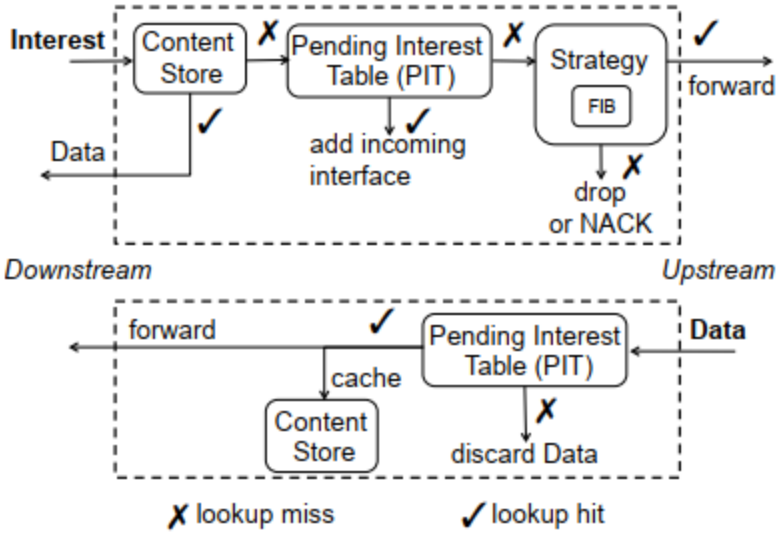
Εικόνα 4. Σύγκριση IP και NDN πρωτοκόλλου [7]

Η επικοινωνία στο NDN πραγματοποιείται με την χρήση δύο διαφορετικών πακέτων, των πακέτων «ενδιαφέροντος» (Interest packets) και των πακέτων Δεδομένων (Data packets). Τα NDN πακέτα μεταφέρουν ονόματα δεδομένων και όχι διευθύνσεις προορισμών. Και οι δύο τύποι πακέτων φέρουν ένα όνομα, το οποίο προσδιορίζει με μοναδικό τρόπο ένα κομμάτι δεδομένων που μπορεί να μεταφερθεί σε ένα πακέτο δεδομένων. Τα ονόματα δεδομένων στο NDN είναι ιεραρχικά δομημένα, για παράδειγμα home/room/humidity/.



Εικόνα 5. Τα πακέτα στην αρχιτεκτονική NDN [8]

Οι κόμβοι που ζητούν δεδομένα λέγονται καταναλωτές (consumers) ενώ οι κόμβοι που δημιουργούν δεδομένα ονομάζονται παραγωγοί (producers). Κάθε μονάδα προώθησης κόμβου NDN περιέχει τρία βασικά στοιχεία: το Content Store (CS), το Pending Interests Table (PIT) και το Forwarding Information Base (FIB).



Εικόνα 6. Διαδικασία προώθησης σε έναν NDN κόμβο [7].

Όταν ένας καταναλωτής επιθυμεί να λάβει κάποια δεδομένα αρχικά αποστέλλει ένα πακέτο Interest στο δίκτυο. Οι δρομολογητές που λαμβάνουν το πακέτο Interest το προωθούν βάσει του ονόματός του, μέχρι να φτάσει στον παραγωγό δεδομένων (ή σε κάποιον κόμβο που έχει αποθηκευμένα τα δεδομένα). Κατά τη λήψη ενός πακέτου Interest από κάθε κόμβο NDN, ο τελευταίος πρώτα ελέγχει αν τα ζητούμενα δεδομένα βρίσκονται στο CS του. Αν βρίσκονται, τα στέλνει πίσω στον κόμβο από τον οποίον έλαβε το αρχικό Interest. Αν τα δεδομένα δεν βρίσκονται στο CS, τότε ο NDN κόμβος ελέγχει αν υπάρχει η καταχώρηση του ονόματος στον PIT. Σε περίπτωση που βρεθεί παρόμοια καταχώρηση στον PIT, η εισερχόμενη διεπαφή του πακέτου Interest προστίθεται στο PIT, έτσι ώστε μόλις ληφθούν τα δεδομένα να τα προωθήσει στον καταναλωτή. Εάν δεν βρεθεί η καταχώρηση του ονόματος του πακέτου στον PIT, ο δρομολογητής καταγράφει τις εισερχόμενες και εξερχόμενες διεπαφές του Interest στο PIT, μαζί με ένα timestamp. Η διεπαφή εξόδου επιλέγεται από τη στρατηγική προώθησης του FIB και τη μετρούμενη απόδοση.

Μόλις ένα Interest φτάσει σε έναν κόμβο ο οποίος έχει ένα πακέτο δεδομένων με το αντίστοιχο όνομα, τότε το πακέτο δεδομένων αυτό, το οποίο περιέχει το όνομα, το περιεχόμενο και μια υπογραφή από το κλειδί του παραγωγού, προωθείται στην αντίθετη διαδρομή του Interest, hop-by-hop. Όταν λαμβάνεται ένα πακέτο δεδομένων από έναν

κόμβο, αυτός ελέγχει αν το όνομά του πακέτου δεδομένων υπάρχει σε κάποια καταχώρηση του PIT. Εάν εντοπιστεί μια αντιστοιχισμένη καταχώρηση PIT, ο δρομολογητής στέλνει το πακέτο δεδομένων στις διεπαφές από τις οποίες έλαβε το πακέτο Interest, αποθηκεύει τα δεδομένα στο Content Store και καταργεί την καταχώρηση PIT. Διαφορετικά, το πακέτο δεδομένων είναι ανεπιθύμητο και απορρίπτεται. Κάθε Interest έχει επίσης μια σχετική διάρκεια ζωής, η οποία όταν λήξει αφαιρείται η αντίστοιχη καταχώρηση PIT. Αυτή η ανταλλαγή πακέτων Interest και πακέτων δεδομένων δημιουργεί έναν κλειστό βρόχο σε κάθε hop, επιτρέποντας σε κάθε δρομολογητή να μετράει την απόδοση ανάκτησης δεδομένων, να αναφέρει προβλήματα προώθησης μέσω hop-by-hop NACK και να εκτελεί αποτελεσματικό έλεγχο συμφόρησης [7] [9].

ΚΕΦΑΛΑΙΟ 3: Σχετικές μελέτες/έρευνες

3.1 DTN και IoT

Οι κλασσικές λύσεις IoT βασίζονται στην επίμονη από άκρο σε άκρο συνδεσιμότητα. Η εφαρμογή της DTN αρχιτεκτονικής στο IoT μπορεί να υποστηρίξει σενάρια διακοπτόμενης σύνδεσης, κινητικότητας κόμβων και μεγάλων περιόδων εξοικονόμησης ενέργειας. Παρόλο που η κινητικότητα των κόμβων αποτελεί σημαντικό αντικείμενο μελέτης των Δικτύων υπολογιστών, οι προσωρινές χρονικές αποσυνδέσεις ή οι διακοπές έχουν μελετηθεί σε μικρότερο βαθμό [10]. Οι διακοπές είναι συνέπεια της μεγάλης κινητικότητας, κατά την οποία οι κόμβοι βγαίνουν εκτός εμβέλειας του δικτύου. Ωστόσο, διακοπές και αποσυνδέσεις συμβαίνουν και σε στατικούς κόμβους αισθητήρων που μπορεί να εισέλθουν σε λειτουργία ύπνου για εξοικονόμηση ενέργειας ή να απενεργοποιηθούν εντελώς λόγω έλλειψης ενέργειας.

Σε κλασσικές υλοποιήσεις IoT όταν η εφαρμογή ενημερώνεται ότι ένας συγκεκριμένος κόμβος δεν είναι διαθέσιμος, πρέπει να προσπαθήσει αργότερα την αποστολή του πακέτου. Με την εφαρμογή της αρχιτεκτονικής DTN γίνεται χρήση της στρατηγικής αποθήκευσης και προώθησης σε ένα hop (single-hop) και σε πολλαπλά hops (Multi-hop store and forward). Κατά τη single-hop αποθήκευση και προώθηση ένα αντίγραφο των δεδομένων μπορεί να παραμείνει στον κόμβο του αποστολέα μέχρις ότου δημιουργηθεί μια σταθερή διαδρομή με τον προορισμό. Κατά τη multi-hop αποθήκευση και προώθηση, η διαδικασία αποθήκευσης δεδομένων σε κόμβους μπορεί να συμβεί αρκετές φορές κατά μήκος της διαδρομής δεδομένων, δηλαδή κάθε ενδιάμεσος κόμβος να προωθήσει τα δεδομένα σε άλλους που μπορεί να φτάσουν στον προορισμό. Σε σενάρια κινητών κόμβων και μεγάλης διάρκειας αποσυνδέσεων, που η από άκρο σε άκρο σύνδεση μπορεί να μην συμβεί ποτέ, η χρήση multi-hop αποθήκευσης και προώθησης αποτελεί μια αποτελεσματική λύση. Η χρήση drone για τη συλλογή δεδομένων από αισθητήρες σε απομακρυσμένες περιοχές, η μεταφορά δεδομένων μέσω έξυπνων αυτοκινήτων και η διασύνδεση απομακρυσμένων περιοχών μέσω δορυφόρων, αποτελούν εφαρμογές που προϋποθέτουν την ύπαρξη ενδιάμεσων κόμβων (drones, αυτοκίνητα, δορυφόροι κ.τ.λ.).

Αυτοί οι ενδιάμεσοι κόμβοι, οι οποίοι λειτουργούν ως κινητές DTN πύλες και ονομάζονται data mules, θα πρέπει να λαμβάνουν δεδομένα από το ένα άκρο (πχ δίκτυο αισθητήρων), να τα αποθηκεύουν και, μόλις βρεθούν εντός εμβέλειας του προορισμού, να τα προωθούν στο άλλο άκρο (πχ κόμβος που είναι συνδεδεμένος στο Διαδίκτυο). Κανένα από τα υπάρχοντα IoT πρωτόκολλα δεν υποστηρίζει αυτόματα την multi-hop αποθήκευση και προώθηση. Αυτό καθιστά τη DTN αρχιτεκτονική κατάλληλη επιλογή για εφαρμογές στις οποίες η τοπολογία του δικτύου μεταβάλλεται διαρκώς και οι κόμβοι ενδέχεται να είναι αποσυνδεδεμένοι για μεγάλα χρονικά διαστήματα [10].

Η ενσωμάτωση της DTN αρχιτεκτονικής στο Διαδίκτυο των Αντικειμένων και γενικά στα Ασύρματα Δίκτυα Αισθητήρων έχει εφαρμοστεί σε διάφορα ερευνητικά προγράμματα. Στο ερευνητικό πρόγραμμα SENSKIN [11] πραγματοποιούνταν δραστηριότητες παρακολούθησης της καταπόνησης γεφυρών και, από πλευράς επικοινωνίας, εφαρμόστηκε ένα πλήρως κατανεμημένο και αυτόνομο ασύρματο δίκτυο αισθητήρων το οποίο χρησιμοποιεί τεχνολογίες DTN. Έτσι, οι μετρήσεις παραμόρφωσης θα λαμβάνονται από τον σταθμό βάσης ακόμη και σε ακραίες συνθήκες που διαταράσσονται οι κανονικές επικοινωνίες. Σε μία εργασία που παρουσιάστηκε μια πλατφόρμα αισθητήρων, υπολογισμών και επικοινωνιών για περιβαλλοντική παρακολούθηση [12], χρησιμοποιείται DTN για μη χρονικά κρίσιμες διεργασίες σε διαφορετικές ασύρματες συνδέσεις. Στη μελέτη που προτείνεται το πρωτόκολλο IoB-DTN [13], γίνεται χρήση του IoT σε συνδεδεμένα ποδήλατα τα οποία συλλέγουν, αποθηκεύουν και στέλνουν δεδομένα σε διαφορετικούς κόμβους μέχρι να φτάσουν στον τελικό παραλήπτη.

3.2 NDN και IoT

Ο σχεδιασμός μιας αρχιτεκτονικής δικτύωσης που διασυνδέει ένα τεράστιο οικοσύστημα, στο οποίο τα αντικείμενα μπορεί να είναι περιορισμένα σε πόρους και

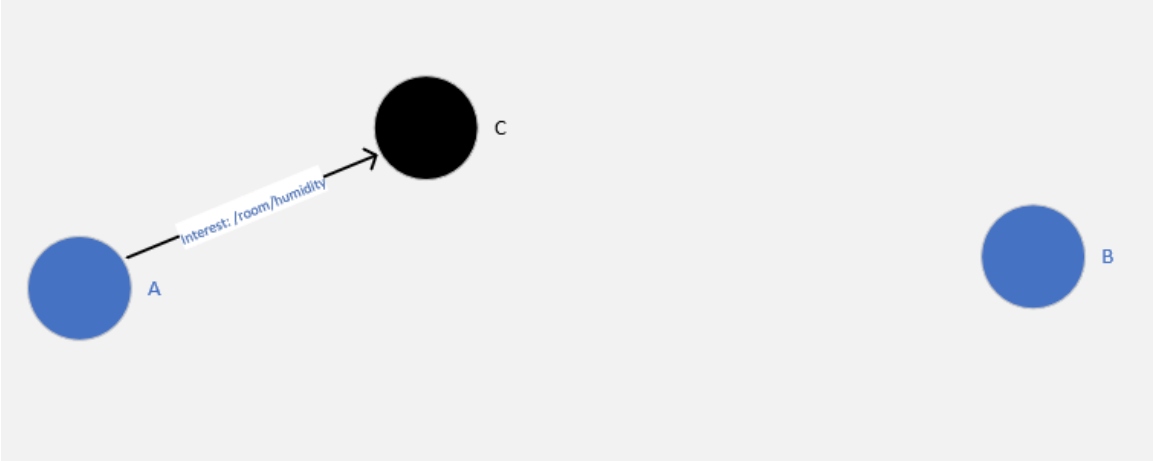
κινητά, δημιουργεί μεγάλες προκλήσεις. Η ασφάλεια, η δυνατότητα επέκτασης και η αξιοπιστία αποτελούν μερικές από τις απαιτήσεις του IoT.

Το NDN μπορεί να αντιμετωπίσει πολλές από αυτές τις απαιτήσεις καθώς προσφέρει εύκολη, ασφαλή και αξιόπιστη ανάκτηση δεδομένων. Η χρήση ονομάτων περιεχομένου αντικαθιστά τις διαδικασίες αντιστοίχισης διευθύνσεων IP και διευκολύνει την αναζήτηση και ανάκτηση περιεχομένου σε μεγάλα δίκτυα. Με αυτόν τον τρόπο, οι ενδιάμεσοι NDN δρομολογητές μπορούν να αναγνωρίσουν πολλαπλά πακέτα Interest για το ίδιο περιεχόμενο και να προωθήσουν μόνο ένα πακέτο Interest στον τελικό παραγωγό, μειώνοντας σημαντικά την κίνηση του δικτύου. Επιπρόσθετα, η προσωρινή αποθήκευση δεδομένων στο δίκτυο που υποστηρίζεται από NDN κόμβους αυξάνει την διαθεσιμότητα των δεδομένων και συμβάλλει στην καλύτερη αντιμετώπιση σύντομων διακοπών επικοινωνίας. Όλα αυτά τα χαρακτηριστικά βελτιώνουν σαφώς την ενεργειακή αποδοτικότητα των κόμβων του δικτύου, οι οποίοι μπορεί να είναι συσκευές περιορισμένων πόρων και ισχύος, και εξασφαλίζουν την συνολική μείωση της κίνησης του δικτύου. Η διασύνδεση των αντικειμένων στο Διαδίκτυο σε συνδυασμό με την σημαντικότητα της ορθής λειτουργίας τους δημιουργεί την ανάγκη εύρεσης και εφαρμογής μεθόδων ασφάλειας. Στο NDN, οι υπογραφές ανά πακέτο και η προαιρετική κρυπτογράφηση δεδομένων ενισχύουν την ασφάλεια του δικτύου. Η δρομολόγηση πολλαπλών μονοπατιών και η προσωρινή αποθήκευση εντός δικτύου συμβάλλουν στην αύξηση της αξιοπιστίας [14].

Έχουν εκπονηθεί αρκετές εργασίες σχετικά με το NDN για το IoT. Σε πειράματα που υλοποιήθηκαν με δεκάδες περιορισμένων πόρων IoT συσκευές σε πραγματικές συνθήκες [15] αποδείχθηκε ότι η εφαρμογή του NDN στο IoT μπορεί να συμβάλλει στην μείωση της καταναλισκόμενης ενέργειας και μνήμης. Επίσης, η εφαρμογή μοντέλων της ICN αρχιτεκτονικής για την υποστήριξη του IoT μέσω δορυφορικών δικτύων οδηγεί στη μείωση της κίνησης του δικτύου [16]. Σε μία σύγκριση των πρωτοκόλλων για το IoT [17] αποδείχθηκε ότι, σε αντίθεση με τα πρωτόκολλα που βασίζονται στο UDP, οι NDN αρχιτεκτονικές συμβάλλουν στην εξισορρόπηση των ροών και την αξιόπιστη μεταφορά των δεδομένων χωρίς την ανάγκη αξιοσημείωτων ρυθμών αναμετάδοσης.

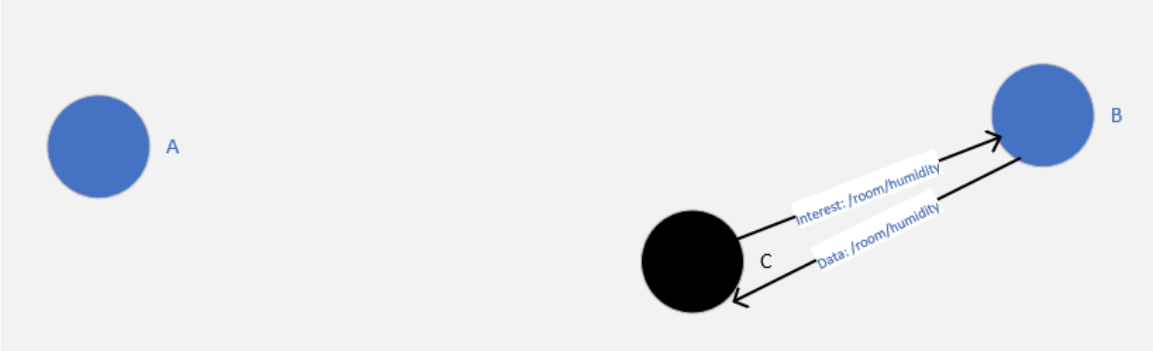
3.3 NDN πάνω από DTN

Παρά τα πλεονεκτήματα εφαρμογής του NDN στο IoT, η επιστροφή ενός πακέτου δεδομένων πρέπει να γίνει ακριβώς από το ίδιο μονοπάτι από το οποίο στάλθηκε το πακέτο Interest. Σε εφαρμογές με κινητούς κόμβους, μεγάλες καθυστερήσεις ή με μεταβλητές τοπολογίες λόγω περιορισμένης ισχύς των κόμβων, το μονοπάτι αυτό μπορεί να μην δημιουργηθεί ξανά καθώς ενδέχεται να μην εμφανιστεί ο ίδιος κινητός κόμβος. Έτσι, ακόμα και αν εμφανιστεί κάποιος άλλος κινητός κόμβος που έχει τα δεδομένα, δεν μπορεί να τα προωθήσει πίσω στον καταναλωτή. Για παράδειγμα, εάν ο κόμβος A θέλει να λάβει μια μέτρηση ενός αισθητήρα υγρασίας, στέλνει ένα πακέτο Interest σε όλους τους διαθέσιμους κόμβους (Εικόνα 6). Ο κινητός κόμβος C, λαμβάνει το πακέτο Interest και επειδή δεν έχει αποθηκευμένα στο Content Store τα ζητούμενα δεδομένα, δημιουργεί μια καινούρια εγγραφή στον PIT.



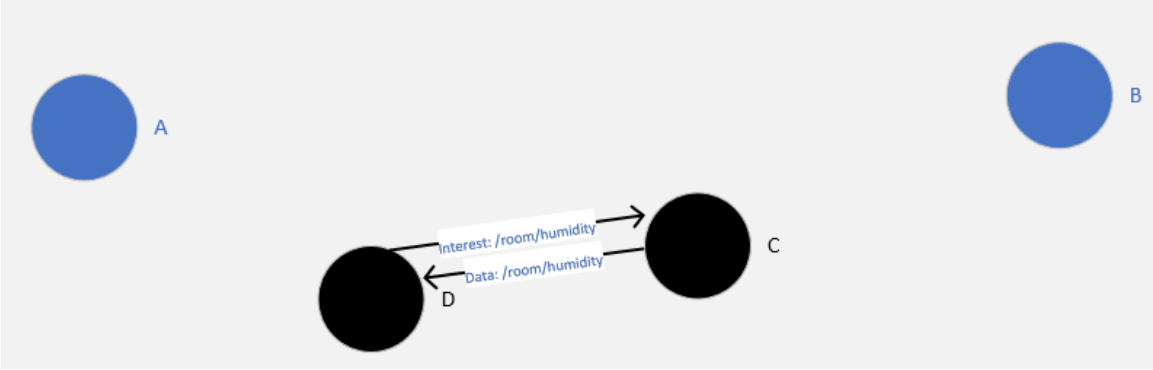
Εικόνα 7. Ο κόμβος A στέλνει ένα πακέτο Interest στον κόμβο C.

Όταν ο κινητός κόμβος C μετακινηθεί και βρεθεί εντός εμβέλειας του κόμβου B, του προωθεί το πακέτο Interest του A. Ο κόμβος B διαθέτει το ζητούμενο περιεχόμενο και το προωθεί αμέσως στον κόμβο C (Εικόνα 7).



Εικόνα 8. Ο κόμβος C λαμβάνει τα δεδομένα από τον κόμβο B

Ο κόμβος C απομακρύνεται από τον κόμβο B. Έπειτα εμφανίζεται ένας άλλος κινητός κόμβος D, ο οποίος του στέλνει ένα πακέτο Interest για την μέτρηση της υγρασίας. Ο κόμβος C διαθέτει το ζητούμενο περιεχόμενο στο Content Store του και έτσι το στέλνει αμέσως στον κόμβο D (Εικόνα 8).



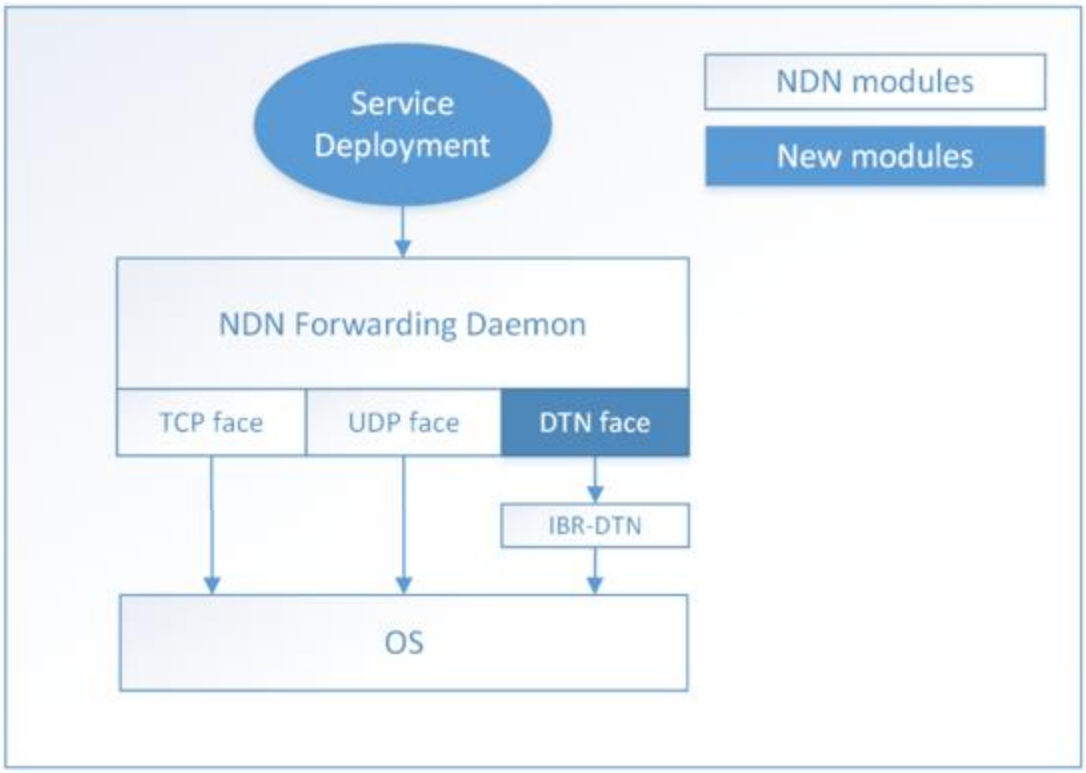
Εικόνα 9. Ο κόμβος D στέλνει ένα πακέτο Interest και λαμβάνει τα ζητούμενα δεδομένα από τον κόμβο C.

Στη συνέχεια ο κόμβος C απομακρύνεται από όλους τους κόμβους, ενώ ο κόμβος D βρίσκεται εντός εμβέλειας του κόμβου A. Ωστόσο, ο κόμβος D δεν μπορεί να μεταφέρει τα δεδομένα της μέτρησης που ζήτησε ο A, καθώς ο A δεν του έχει στείλει σχετικό πακέτο Interest (Εικόνα 9). Επίσης, το πακέτο δεδομένων που περιμένει ο A μπορεί να επιστρέψει μόνο από τον κόμβο C, δηλαδή μόνο από το ίδιο μονοπάτι από το οποίο στάλθηκε το πακέτο Interest. Έτσι, το πακέτο Interest του A ξεπερνά τη διάρκεια ζωής του και λήγει.

Εικόνα 10. Αδυναμία επιστροφής των δεδομένων από διαφορετικό μονοπάτι.

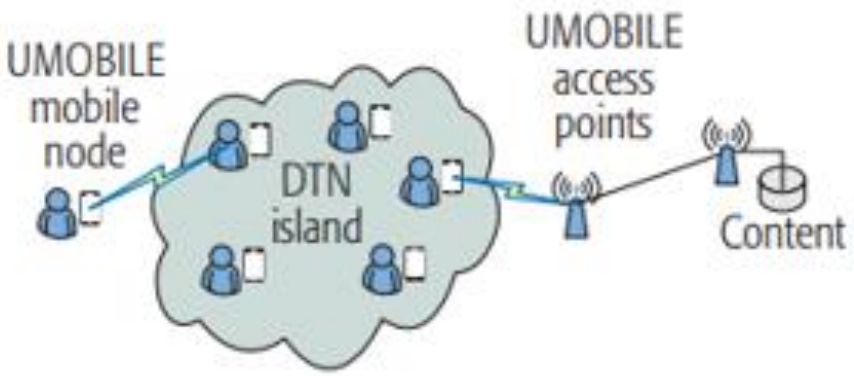
Όταν οι κόμβοι του δικτύου είναι DTN, τότε λύνεται το πρόβλημα της μεταφοράς των δεδομένων μέσω διαφορετικών ενδιάμεσων κόμβων. Αυτό συμβαίνει γιατί το DTN αξιοποιεί κάθε διαθέσιμο μονοπάτι (δηλαδή στο προηγούμενο παράδειγμα ο D θα μπορούσε να προωθήσει τα δεδομένα στον A). Ωστόσο, χάνουμε τα οφέλη της προσωρινής αποθήκευσης, με την οποία μπορούν να εξυπηρετηθούν μελλοντικά αιτήματα για τα ίδια δεδομένα από το σταθερό δίκτυο και να μειωθεί σημαντικά η καθυστέρηση. Σε μία παραπλήσια εφαρμογή με το προηγούμενο παράδειγμα, ένας κόμβος DTN χρειάζεται να στείλει ένα bundle για να ζητήσει μία μέτρηση από τον παραλήπτη και να λάβει ένα bundle από τον παραλήπτη με την ζητούμενη μέτρηση. Εάν τρεις DTN κόμβοι ζητήσουν να λάβουν την ίδια μέτρηση, θα έπρεπε να στείλουν τρεις διαφορετικές δέσμες με τις οποίες ο καθένας ζητά την μέτρηση από τον παραλήπτη. Έπειτα ο παραλήπτης, θα πρέπει να αποστείλει τρεις δέσμες με το ίδιο περιεχόμενο σε κάθε έναν από τους αποστολείς.

Προκειμένου να αξιοποιηθούν τα πλεονεκτήματα και των δύο αρχιτεκτονικών, στο UMOBILE [18] έργο εφαρμόστηκε μία νέα αρχιτεκτονική. Συγκεκριμένα, αναπτύχθηκε μια σημαντική δυνατότητα προώθησης, η σήραγγα DTN (DTN Tunneling), με την οποία τα NDN πακέτα διαβιβάζονται σε μια διεπαφή DTN (DTN face). Με αυτόν τον τρόπο επεκτείνεται η λειτουργία του NDN και εξασφαλίζεται προσβασιμότητα σε απομακρυσμένες περιοχές.

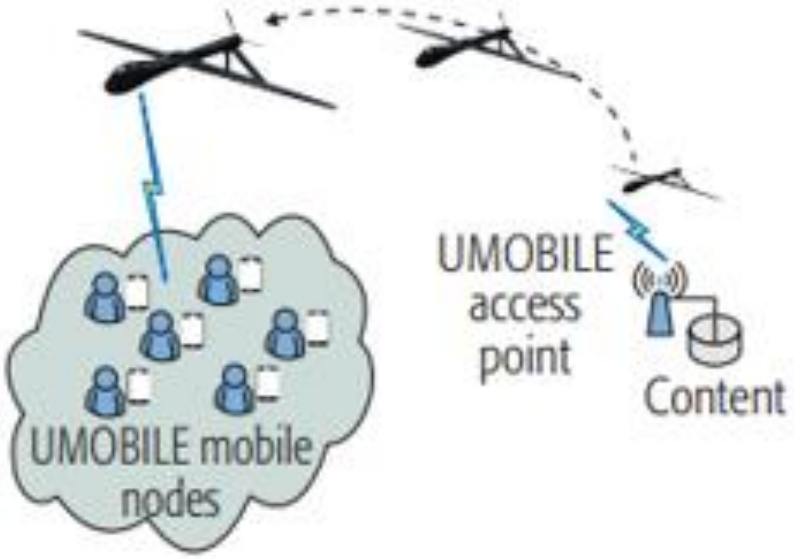


Εικόνα 11. Αρχιτεκτονική UMOBILE [18]

Η σήραγγα μέσω συσκευών με δυνατότητα DTN επιλύει το ζήτημα της επιστροφής του πακέτου δεδομένων NDN μόνο από το ίδιο μονοπάτι που ακολούθησε το πακέτο Interest. Ένας κόμβος UMOBILE (π.χ. ένα σημείο πρόσβασης) διατηρεί μία είσοδο FIB / PIT προς τον επόμενο κόμβο, ενώ τα πακέτα Interest και τα πακέτα δεδομένων ενθυλακώνονται σε bundles και προωθούνται από τους ενδιάμεσους κινητούς κόμβους DTN. Επιπλέον, το DTN μπορεί να χρησιμοποιηθεί για την παροχή ολόκληρων υπηρεσιών σε απομακρυσμένες περιοχές, μέσω των data mules.



Εικόνα 12. Λειτουργία της σήραγγας DTN [18]



Εικόνα 13. Λειτουργία data muling [18]

Με τη χρήση του NDN-DTN stack του UMOBILE, υπάρχει τόσο η δυνατότητα τοπικής εξυπηρέτησης των αιτημάτων από την cache, μειώνοντας σημαντικά την καθυστέρηση, όσο και η δυνατότητα προώθησης των πακέτων από οποιοδήποτε μονοπάτι. Επίσης υπάρχει και η δυνατότητα διαχείρισης των διακοπών στη συνδεσιμότητα, από το DTN stack.

Η εφαρμογή της αρχιτεκτονικής UMOBILE, μπορεί να αποτελέσει μια σημαντική βελτίωση της DTN αρχιτεκτονικής, ιδιαίτερα στον κλάδο του IoT, καθώς εξασφαλίζει τα οφέλη και των δύο αρχιτεκτονικών. Σε ασύρματα δίκτυα αισθητήρων τα οποία είναι εγκατεστημένα σε δυσπρόσιτες περιοχές, η απευθείας σύνδεση των συσκευών με το Διαδίκτυο είναι είτε ιδιαίτερα δαπανηρή είτε μη εφικτή. Έτσι, η χρήση των data mules, που μπορεί να είναι έξυπνα τηλέφωνα, έξυπνα αυτοκίνητα, drones, δορυφόροι κ.τ.λ., μπορεί να οδηγήσει στην διασύνδεση αυτών των περιοχών με το Διαδίκτυο. Ταυτόχρονα η ονομασία των δεδομένων, η προσωρινή αποθήκευσή τους και η ασφάλεια που παρέχει το NDN μπορούν να αυξήσουν την αποδοτικότητα και την αξιοπιστία ορισμένων εφαρμογών IoT.

ΚΕΦΑΛΑΙΟ 4: Πλατφόρμες-Εργαλεία

4.1 Πλατφόρμες DTN

Τα τελευταία χρόνια έχουν αναπτυχθεί αρκετές εφαρμογές λογισμικού οι οποίες υλοποιούν τα DTN πρωτόκολλα, ανάλογα με τις ειδικές ανάγκες του συστήματος και τις αρχιτεκτονικές επεξεργασίας. Η βασική πλατφόρμα η οποία χρησιμοποιείται σε διαστημικές αποστολές είναι το ION (Interplanetary Overlay Network ION) [19]. Το ION είναι μια πλατφόρμα DTN που έχει δημιουργηθεί από το Jet Propulsion Laboratory (JPL), ειδικά σχεδιασμένη για να χρησιμοποιηθεί για διαπλανητικές επικοινωνίες. Η επικοινωνία μεταξύ διαστημικών οχημάτων και Γης διαφέρει αρκετά από τις αντίστοιχες επίγειες επικοινωνίες. Τα ασύρματα δίκτυα στο Διάστημα είναι σχετικά πιο αργά και οι εκπομπές τους μπορεί να χρειαστούν αρκετά μεγάλο χρονικό διάστημα για να φτάσουν στο προορισμό τους. Επίσης, το χρησιμοποιούμενο hardware έχει σαφώς μεγαλύτερους περιορισμούς σε σχέση με το αντίστοιχο στις επίγειες επικοινωνίες, τόσο ως προς την υπολογιστική του ισχύ όσο και ως προς τις ενεργειακές του απαιτήσεις. Επιπρόσθετα, λόγω της φύσης της τροχιακής ή διαπλανητικής πτήσης και της περιστροφής της Γης, όπως και άλλων πλανητικών σωμάτων, τα παράθυρα επικοινωνιακής επαφής είναι περιορισμένα και αυστηρά καθορισμένα. Για τον λόγο αυτό, η δρομολόγηση δεδομένων στο ION γίνεται αποκλειστικά με τη χρήση ενός πίνακα προγραμματισμένων επαφών [20].

Το ION περιέχει όλες τις λειτουργίες προδιαγραφών BP, ωστόσο δεν είναι κατάλληλο για εφαρμογές, όπως ένα ασύρματο δίκτυο αισθητήρων. Δεν υποστηρίζει ένα μηχανισμό ανακάλυψης γειτονικών κόμβων, καθιστώντας τη δημιουργία κατ' απαίτηση δικτύων (ad hoc network) το λιγότερο προβληματική, ενώ δεν προβλέπεται η χρήση μηχανισμών δρομολόγησης για δίκτυα πλέγματος (mesh network routing). Πέραν της επέκτασης του ION με την προσθήκη αντίστοιχων δυνατοτήτων, μια εναλλακτική λύση θα είναι η χρησιμοποίηση μιας άλλης πλατφόρμας DTN δικτύωσης, η οποία να είναι συμβατή σε επίπεδο BP με αυτό. Μια τέτοια πλατφόρμα είναι το IBR-DTN. Το IBR-DTN [21] είναι μια ελαφριά και εξαιρετικά φορητή πλατφόρμα πρωτοκόλλων DTN σχεδιασμένη για ενσωματωμένα συστήματα. Περιλαμβάνει υποστήριξη αποστολής bundle

πάνω από γνωστά πρωτόκολλα επικοινωνιών όπως τα TCP, UDP και HTTP. Επιπλέον, υποστηρίζει την απευθείας αποστολή bundle μέσω του πρότυπου ασύρματης δικτύωσης IEEE 802.15.4 [22], η οποία μπορεί να είναι χρήσιμη για δίκτυα αισθητήρων εξαιρετικά χαμηλών ενεργειακών απαιτήσεων. Ο συμπεριλαμβανόμενος DTN δαίμονας παρέχει διαφορετικά συστήματα δρομολόγησης, καθώς και μηχανισμό ανακάλυψης γειτονικών κόμβων, ενώ είναι συμβατός με την προδιαγραφή ασφαλείας BSP [23]. Λόγω του γεγονότος ότι το IBR-DTN αναπτύχθηκε στοχεύοντας σε ενσωματωμένα συστήματα, οι εξαρτήσεις λογισμικού διατηρήθηκαν στο ελάχιστο και έτσι αποτελεί μια κατάλληλη επιλογή για την συγκεκριμένη μελέτη.

Ακόμη, υπάρχουν και πιο ελαφριές πλατφόρμες DTN, όπως το μDTN [24] ή το miniDTN [25]. Ωστόσο, επειδή δεν έχει εξεταστεί η συμβατότητα τους σε επίπεδο BP με το IBR-DTN, δεν χρησιμοποιήθηκαν σε αυτήν την εργασία.

4.2 Πλατφόρμες NDN

Τα πιο γνωστά πρωτόκολλα για το NDN είναι το CCNx και το Named Data Networking Forwarding Daemon (NFD) [26], τα οποία έχουν σχεδιαστεί με στόχο την εφαρμογή τους σε συσκευές υπολογιστών. Η εφαρμογή τους καθίσταται αδύνατη για συσκευές περιορισμένων πόρων. Για αυτό το λόγο έχουν αναπτυχθεί άλλες υλοποιήσεις που στοχεύουν σε Low-End IoT συσκευές [27]. Μία ελαφριά έκδοση του CCNx, το CCN-Lite, μεταφέρθηκε στο λειτουργικό σύστημα ανοιχτού κώδικα για συσκευές αισθητήρων IoT, RIOT [28]. Υλοποίηση του CCNx ενσωματώθηκε στο Contiki [29], το οποίο είναι ένα λειτουργικό σύστημα που χρησιμοποιείται για ασύρματα δίκτυα αισθητήρων. Επιπλέον, έχει υλοποιηθεί στο RIOT και η στοίβα πρωτοκόλλου NDN [30], η οποία υποστηρίζει τη βασική λογική προώθησης του NDN.

Η υλοποίηση του NFD ξεχωρίζει από τις υπόλοιπες λόγω της επεκτασιμότητας της και της ευκολίας που προσφέρει για την διεξαγωγή πειραμάτων με το NDN πρωτόκολλο. Στην παρούσα εργασία θα εφαρμοστεί η υλοποίηση του NFD, καθώς η UMOBILE

αρχιτεκτονική κάνει χρήση της υλοποίησης του NFD δαίμονα πάνω από την υλοποίηση του IBR-DTN. Επειδή οι προαναφερόμενες ελαφριές υλοποιήσεις δεν είναι συμβατές με την υλοποίηση του NFD, οι κόμβοι αισθητήρων δεν μπορούν να εφαρμόσουν κάποια από αυτές.

4.3 Λειτουργικό σύστημα RIOT

Για την υλοποίηση των πειραμάτων της εργασίας ήταν αναγκαία η εύρεση ενός απλοποιημένου τρόπου προγραμματισμού των μικροελεγκτών, με δυνατότητα εύκολης μεταφοράς της εφαρμογής σε μικροελεγκτές διαφορετικής αρχιτεκτονικής για μελλοντική χρήση. Η διεξαγωγή πειραμάτων με δίκτυα αισθητήρων προαπαιτεί τη χρήση πρότυπων επικοινωνίας και πρωτοκόλλων χαμηλής ενεργειακής κατανάλωσης, όπως το IEEE 802.15.4. Επίσης, η λήψη μετρήσεων από διαφορετικά είδη αισθητήρων προϋποθέτει την επιλογή μιας πλατφόρμας που να υποστηρίζει αρκετούς αισθητήρες.

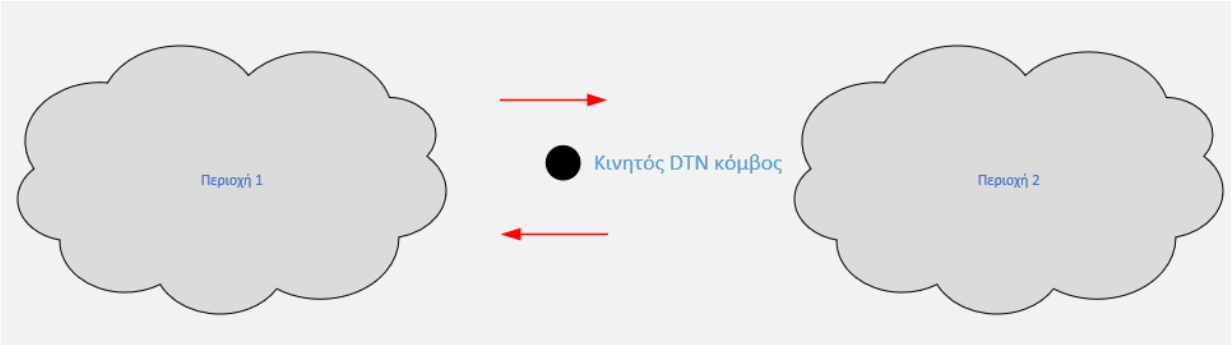
Για την εξυπηρέτηση των παραπάνω αναγκών στους κόμβους αισθητήρων επιλέχθηκε η χρήση του RIOT OS. Το συγκεκριμένο λειτουργικό σύστημα έχει σχεδιαστεί έτσι ώστε να εξασφαλίζει την ενεργειακή απόδοση της συσκευής, την δέσμευση όσο το δυνατόν λιγότερης μνήμης και την απλότητα στην ανάπτυξη εφαρμογών IoT. Επιπλέον, το RIOT υποστηρίζει πρωτόκολλα επιπέδου συνδέσμου, όπως το IEEE 802.15.4, το Bluetooth Low-Energy (BLE) και το LoRa, καθώς και πρωτόκολλα που επιτρέπουν την σύνδεση των μικροελεγκτών στο Διαδίκτυο, όπως το 6LowPAN, το IPv6 και το UDP.

ΚΕΦΑΛΑΙΟ 5: Μεθοδολογία

5.1 Σενάρια

Η παρούσα εργασία μελετά την απόδοση νέων δικτυακών αρχιτεκτονικών στο Διαδίκτυο των Αντικειμένων. Πιο συγκεκριμένα, εξετάζει την απόδοση εφαρμογής της DTN και της UMOBILE αρχιτεκτονικής, σε εφαρμογές του IoT, στις οποίες οι αισθητήρες είναι τοποθετημένοι σε δυσπρόσιτες περιοχές και η μεταφορά των δεδομένων γίνεται μέσω κινητών DTN κόμβων.

Για την πραγματοποίηση αυτής της μελέτης, χρειάστηκε η ύπαρξη δύο διαφορετικών περιοχών-δικτύων. Στην πρώτη περιοχή, βρίσκεται ο καταναλωτής της εφαρμογής, ο οποίος ζητάει κάποιες μετρήσεις από τους αισθητήρες μιας απομακρυσμένης περιοχής. Στη δεύτερη περιοχή βρίσκονται οι κόμβοι αισθητήρων, οι οποίοι στέλνουν της μετρήσεις τους σε έναν συλλέκτη δεδομένων. Ο συλλέκτης δεδομένων είναι κόμβος της δεύτερης περιοχής και λειτουργεί ως παραγωγός της εφαρμογής. Η απόσταση μεταξύ των περιοχών δεν επιτρέπει τη διασύνδεση τους. Προκειμένου να μεταφερθούν τα αιτήματα των καταναλωτών και τα δεδομένα των αισθητήρων από τη μία περιοχή στην άλλη, χρειάστηκε η ύπαρξη ενός ενδιάμεσου κόμβου, ο οποίος μετακινείται μεταξύ των δυο περιοχών και συνδέεται διαδοχικά στα δύο δίκτυα.



Εικόνα 14. Μια γενική τοπολογία

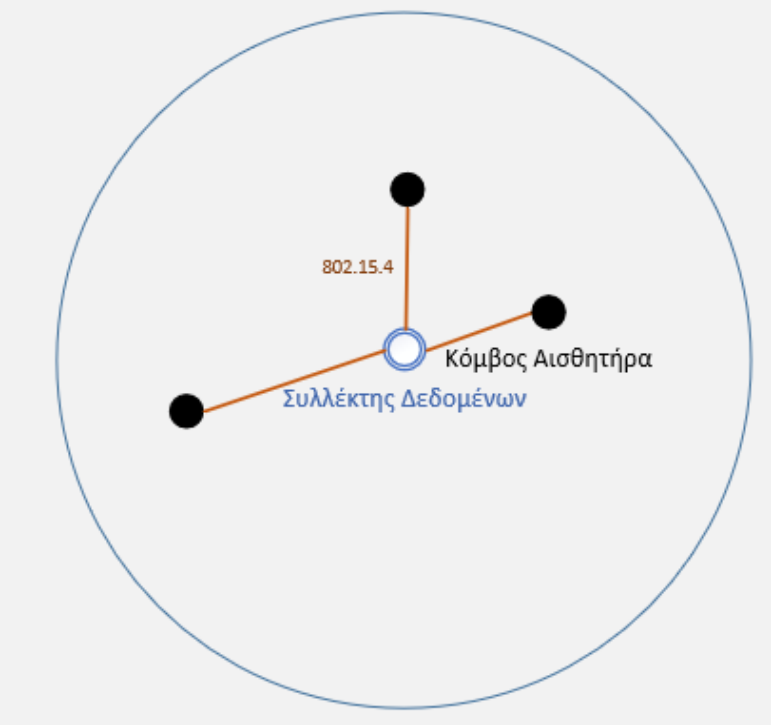
Αρχιτεκτονική DTN: Στην περίπτωση της DTN αρχιτεκτονικής, ο παραγωγός, ο καταναλωτής και ο κινητός κόμβος είναι DTN κόμβοι. Ο κόμβος καταναλωτής χρησιμοποιεί μια εφαρμογή η οποία στέλνει ένα bundle στον παραγωγό, μέσω του κινητού

κόμβου. Το περιεχόμενο του bundle είναι το όνομα μίας μέτρησης των αισθητήρων (π.χ. υγρασία, θερμοκρασία, κίνηση κ.τ.λ.). Η εφαρμογή του παραγωγού, λαμβάνει το περιεχόμενο του bundle που έστειλε ο καταναλωτής και αποστέλλει σε αυτόν, μέσω του κινητού κόμβου, ένα bundle του οποίου το περιεχόμενο είναι η ζητούμενη μέτρηση του αισθητήρα. Το παραπάνω σενάριο DTN έχει φτιαχτεί ώστε να διεξαχθεί μια ισοδύναμη σύγκριση με την UMOBILE αρχιτεκτονική. Σε άλλη περίπτωση ο κόμβος παραγωγός θα μπορούσε να αποστείλει απευθείας τις μετρήσεις των αισθητήρων, μόλις συνδέονταν με τον κινητό κόμβο και ο τελευταίος να τις προωθούσε στον κόμβο καταναλωτή, χωρίς να έχει αποστείλει αίτημα για αυτές.

Αρχιτεκτονική UMOBILE: Σε αυτήν την περίπτωση ο παραγωγός και ο καταναλωτής χρησιμοποιούν την UMOBILE αρχιτεκτονική (δηλαδή NDN στοίβα πάνω από την DTN) και ο κινητός κόμβος χρησιμοποιεί την DTN αρχιτεκτονική. Συγκεκριμένα, ο κόμβος καταναλωτής χρησιμοποιεί μια εφαρμογή του NDN πρωτοκόλλου, η οποία στέλνει ένα πακέτο Interest με όνομα μιας μέτρησης (π.χ. /example/movement1 ή /example/temperature5). Το πακέτο αυτό μετατρέπεται σε ένα bundle περνώντας από το DTN face. Μόλις υπάρξει σύνδεση με τον κινητό κόμβο DTN (και γενικά με οποιονδήποτε DTN κόμβο) ο καταναλωτής του προωθεί το bundle με σκοπό να προωθηθεί μελλοντικά στην περιοχή του παραγωγού. Όταν ο κινητός κόμβος βρεθεί εντός του δικτύου του παραγωγού, του προωθεί το bundle του καταναλωτή. Το DTN face του παραγωγού δέχεται τη δέσμη, την επεξεργάζεται και στη συνέχεια προωθεί το πακέτο Interest στον NFD δαίμονα. Έπειτα, το πακέτο Interest το παραλαμβάνει η εφαρμογή του παραγωγού του NDN. Στην εφαρμογή αυτή, δημιουργείται ένα πακέτο δεδομένων που περιέχει την πιο πρόσφατη ζητούμενη μέτρηση των αισθητήρων. Το πακέτο δεδομένων διέρχεται από το DTN face, γίνεται bundle και προωθείται στον κινητό κόμβο ο οποίος με τη σειρά του το προωθεί στον καταναλωτή, μόλις βρεθεί εντός της περιοχής του.

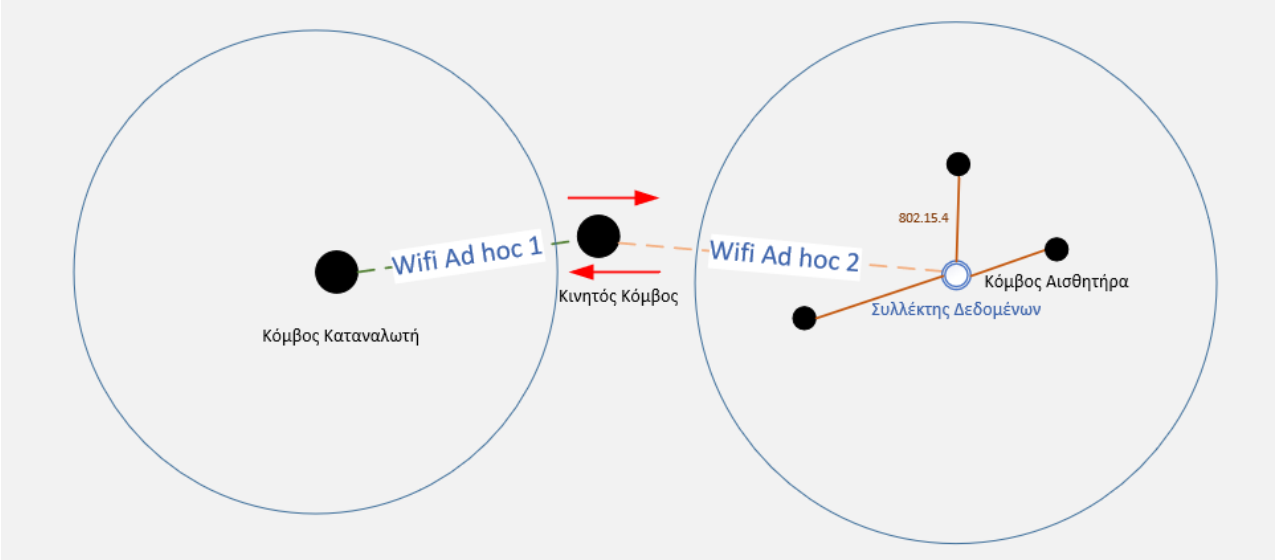
5.2 Τοπολογία Δικτύου

Για την υλοποίηση των προαναφερόμενων σεναρίων ήταν αναγκαία η εύρεση κατάλληλων μέσων και πρωτοκόλλων δικτύωσης των κόμβων. Οι κόμβοι αισθητήρων αποτελούν συνήθως μικρές συσκευές, χαμηλής υπολογιστικής ισχύος και περιορισμένων πόρων. Για να αξιοποιηθεί το πλεονέκτημα της χαμηλής ενεργειακής τους απαίτησης, οι συσκευές αυτές θα πρέπει να υποστηρίζουν την επικοινωνία με τον κόμβο συλλέκτη μέσω ενός κατάλληλου πρότυπου ασύρματης δικτύωσης. Για αυτόν τον λόγο χρησιμοποιήθηκε το πρότυπο ασύρματης δικτύωσης IEEE 802.15.4. Το εν λόγω πρότυπο, αποτελεί μια συνήθη επιλογή για δίκτυα αισθητήρων εξαιρετικά χαμηλών ενεργειακών απαιτήσεων καθώς είναι σχεδιασμένο για συσκευές μικρού ρυθμού μετάδοσης δεδομένων (έως 250 kbps), μειωμένης κατανάλωσης ισχύος και μικρού κόστους. Προκειμένου να υπάρξουν περιθώρια μελλοντικής διασύνδεσης των μικροελεγκτών στο Διαδίκτυο, χρησιμοποιήθηκε η στοίβα πρωτοκόλλου 6LoWPAN [24], η οποία εισάγει ένα στρώμα προσαρμογής μεταξύ της στοίβας IP και του επιπέδου ζεύξης δεδομένων, επιτρέποντας τη διαβίβαση πακέτων IPv6 μέσω του 802.15.4. Ως πρωτόκολλο επιπέδου μεταφοράς χρησιμοποιήθηκε το UDP (User Datagram Protocol), έτσι ώστε ο κάθε μικροελεγκτής να στέλνει μικρού μεγέθους UDP πακέτα σε μια γνωστή θύρα του κόμβου παραγωγού, αποφεύγοντας την εγκαθίδρυση σύνδεσης μεταξύ τους.



Εικόνα 15. Τοπολογία περιοχής παραγωγού

Επίσης, ο κόμβος παραγωγός και ο κόμβος καταναλωτής, θα πρέπει να δημιουργούν ο καθένας από ένα διαφορετικό δίκτυο με το οποίο θα συνδέονται με τον κινητό κόμβο. Έτσι, επιλέχθηκε η δημιουργία δυο ανεξάρτητων ad hoc WiFi δικτύων σε διαφορετικά κανάλια εκπομπής, προκειμένου να αποφευχθούν οι παρεμβολές. Η επιλογή του WiFi έγινε για την επίτευξη μεγαλύτερου ρυθμού μετάδοσης δεδομένων, καθώς οι συνδέσεις των κόμβων είναι ευκαιριακές και με μικρή διάρκεια.



Εικόνα 16. Τοπολογία Δικτύου

Με τη χρήση του WiFi interface, ο κινητός κόμβος θα μπορεί να συνδέεται εναλλακτικά είτε στο ένα ad hoc δίκτυο είτε στο άλλο είτε και σε κανένα από τα δύο, μεταβάλλοντας την εικονική του τοποθεσία. Η διάρκεια παραμονής σε κάθε κατάσταση (συνδεδεμένος στο ad hoc 1, αποσυνδεδεμένος, συνδεδεμένος στο ad hoc 2) εξομοιώνει την κίνηση του κόμβου.

5.3 Κόμβοι αισθητήρων

Οι κόμβοι αισθητήρων είναι συσκευές χαμηλού κόστους και περιορισμένων δυνατοτήτων, οι οποίοι βασίζονται σε μικροελεγκτές χαμηλής υπολογιστικής ισχύος και μειωμένων πόρων, όπως χωρητικότητα μνήμης και αποθηκευτικού χώρου. Το κυριότερο πλεονέκτημα τους είναι το μικρό μέγεθος και οι εξαιρετικά χαμηλές ενεργειακές απαιτήσεις τους. Οι συγκεκριμένοι κόμβοι συλλέγουν μετρήσεις (υγρασίας, θερμοκρασίας, κίνησης κτλ.) και τις προωθούν ασύρματα στον συλλέκτη. Θεωρούμε ότι, λόγω των περιορισμένων πόρων τους δεν μπορούν να υποστηρίξουν πλήρως τεχνολογίες DTN και αρκούνται σε άμεση αποστολή των δεδομένων τους. Το χαμηλό κόστος, το μικρό

μέγεθος και οι περιορισμένες ενεργειακές απαιτήσεις των κόμβων αυτών συγκαταλέγονται στα βασικά πλεονεκτήματά τους που καθιστούν δυνατή την μαζική τοποθέτησή τους. Επιπρόσθετα, όλοι οι κόμβοι αισθητήρων μπορούν να τίθενται σε κατάσταση βαθέως ύπνου (deep sleep mode) κατά την περίοδο που είναι ανενεργοί. Με αυτόν τον τρόπο μειώνουν περαιτέρω τον ενεργειακό φόρτο τους, σε αντίθεση με τον συλλέκτη που υποχρεούται να μένει σε λειτουργία για την δημιουργία ευκαιριακών ad hoc συνδέσεων. Το κυριότερο μειονέκτημα των κόμβων αισθητήρων αποτελεί η εξάρτηση τους από την ομαλή λειτουργία του συλλέκτη τόσο κατά την συλλογή, όσο και στην αποθήκευση και αποστολή δεδομένων.

Εικόνα 17. SAMR-21-XPRO

Για την υλοποίηση και περαιτέρω μελέτη ενός τέτοιου σεναρίου στο εργαστήριο, μπορούμε να χρησιμοποιήσουμε ως κόμβους αισθητήρων τα SAMR-21-XPRO. Το SAMR-21-XPRO βασίζεται σε έναν 32-bit ARM Cortex-M0+ επεξεργαστή, χρονισμένο στα 48MHz, με 32KB SRAM, αποθηκευτικό χώρο 256KB flash memory και ένα πλήρως συμβατό με το πρότυπο ασύρματης δικτύωσης IEEE 802.15.4 πομποδέκτη. Η συσκευή τρέχει μια απλή εφαρμογή συλλογής δεδομένων από αισθητήρες, πάνω από το ελαφρύ λειτουργικό σύστημα RIOT και κάνει άμεση αποστολή δεδομένων προς τον συλλέκτη μέσω του υποστηριζόμενου πρωτοκόλλου 6lowpan (ο κώδικας της εφαρμογής που χρησιμοποιήθηκε υπάρχει στο Παράρτημα). Θεωρητικά αυτός ο συνδυασμός συσκευής-

λειτουργικού συστήματος θα μπορούσε να υποστηρίξει μια ελαφριά πλατφόρμα DTN, όπως το μDTN ή το miniDTN, αλλά θα πρέπει να εξεταστεί η συμβατότητα τους σε επίπεδο BP με το IBR-DTN.

Κάθε κόμβος περιλαμβάνει έναν αισθητήρα κίνησης τύπου PIR (Passive Infrared sensor), έναν αισθητήρα φωτός τύπου LDR (Light Dependent Resistor) και έναν αισθητήρα φαινομένου Hall. Ο αισθητήρας PIR είναι ένας ηλεκτρονικός αισθητήρας που μετρά το υπέρυθρο φως που ακτινοβολεί από αντικείμενα στο οπτικό του πεδίο. Συχνά χρησιμοποιούνται σε ανιχνευτές κίνησης, συναγερμούς ασφαλείας και αυτόματες εφαρμογές φωτισμού. Η φωτοαντίσταση (LDR) είναι ένα ενεργό στοιχείο του οποίου η αντίσταση μεταβάλλεται σε σχέση με τα επίπεδα φωτεινότητας στην επιφάνεια του. Γενικά, η αντίσταση μιας φωτοαντίστασης μειώνεται με την αύξηση της έντασης του προσπίπτοντος φωτός. Ο αισθητήρας φαινομένου Hall χρησιμοποιείται για τη μέτρηση του μεγέθους ενός μαγνητικού πεδίου. Η τάση εξόδου του είναι άμεσα ανάλογη με την ισχύ του μαγνητικού πεδίου που το διαπερνά. Επίσης, στον κώδικα της εφαρμογής προσομοιώνεται η λήψη μετρήσεων από άλλους αισθητήρες (θερμοκρασίας, υγρασίας και ήχου).

Ο κόμβος αισθητήρων λαμβάνει συνέχεια μετρήσεις και τις αποστέλλει στον συλλέκτη δεδομένων. Μόλις λάβει μια καινούρια μέτρηση στέλνει ένα UDP πακέτο σε μία γνωστή θύρα του συλλέκτη δεδομένων (παραγωγού) και έπειτα συνεχίζει την λήψη της επόμενης μέτρησης. Οι μικροελεγκτές χρησιμοποιώντας το πρωτόκολλο 6LoWPAN έχουν μια IPv6 διεύθυνση με την οποία μπορούν να συνδεθούν μέσω του κόμβου παραγωγού στο Διαδίκτυο.

5.4 Κόμβος παραγωγός

Ο συλλέκτης δεδομένων-κόμβος παραγωγός πρέπει να είναι μια συσκευή με επαρκείς επιδόσεις και πόρους για την υποστήριξη μιας πλατφόρμας DTN λογισμικού και μιας πλατφόρμας NDN λογισμικού. Ο συλλέκτης είναι επιφορτισμένος με το να λαμβάνει

τις μετρήσεις από τους κόμβους αισθητήρων μέσω του 802.15.4 interface και να τις αποθηκεύει στη μνήμη του. Παράλληλα, είναι υπεύθυνος για την δημιουργία ενός ad hoc δικτύου μέσω WiFi interface με οποιονδήποτε DTN κόμβο βρεθεί εντός της εμβέλειάς του.

Ως συλλέκτης χρησιμοποιήθηκε ένα Raspberry Pi 2B. Το Raspberry Pi 2B βασίζεται σε έναν τετραπύρηνο ARM Cortex-A7 CPU στα 900MHz, παρέχει 1GB μνήμης RAM και χρησιμοποιεί μια αφαιρούμενη SD κάρτα ως αποθηκευτικό μέσο. Μπορεί να τρέξει την πλήρη έκδοση του IBR-DTN, πάνω από τον κλώνο του Linux Raspbian. Η συνδεσιμότητα του προς τους κόμβους αισθητήρων εξασφαλίζεται με τον πομποδέκτη MRF24J40MA, ο οποίος είναι συμβατός με το IEEE 802.15.4. Για τη δημιουργία των ad hoc συνδέσεων χρησιμοποιήθηκε το κλασσικό WiFi, με WiFi usb dongle με αποσπώμενη κεραία.

Στο σενάριο των DTN κόμβων, ο κόμβος παραγωγός λαμβάνει τις μετρήσεις των αισθητήρων από μια συγκεκριμένη θύρα, μέσω του εργαλείου γραμμής εντολών netcat, και τις αποθηκεύει σε txt αρχεία. Η παραπάνω λειτουργία υλοποιήθηκε με την βοήθεια bash scripts, προκειμένου να αυτοματοποιηθούν όλες οι αναγκαίες διεργασίες. Ταυτόχρονα, ο κόμβος παραγωγός εκτελεί και κάποια bash scripts με τα οποία μόλις ο IBR-DTN λάβει ένα bundle με συγκεκριμένο EID, σκανδαλίζει την εκτέλεση ενός άλλου bash script, το οποίο ανακτά την πιο πρόσφατη ζητούμενη μέτρηση και την αποστέλλει στον καταναλωτή σε μορφή bundle. Τα προαναφερόμενα bash scripts κάνουν χρήση των εργαλείων dtnsend, dtnrecv και dtntrigger του IBR-DTN και αναλύονται στο Παράρτημα.

Στο σενάριο των UMOBILE κόμβων, ο κόμβος παραγωγός εκτελεί μια εφαρμογή NDN παραγωγού η οποία κάνει χρήση της βιβλιοθήκης ndn-cxx. Η εφαρμογή αυτή μόλις λάβει κάποιο πακέτο Interest με συγκεκριμένο prefix δημιουργεί ένα πακέτο δεδομένων με την πιο πρόσφατη μέτρηση για το ζητούμενο prefix. Στη συνέχεια, ορίζει το freshness period του πακέτου και το αποστέλλει στον NFD. Ο NFD προωθεί το πακέτο δεδομένων στο face από το οποίο προήλθε το πακέτο Interest, δηλαδή στο DTN-face.

5.5 Κόμβος καταναλωτή

Ο κόμβος καταναλωτή απαιτεί να είναι μια συσκευή με επαρκείς επιδόσεις και πόρους για την υποστήριξη μιας πλατφόρμας DTN λογισμικού και μιας πλατφόρμας NDN λογισμικού. Ο καταναλωτής στέλνει πακέτα με τα οποία ζητάει από τον παραγωγό κάποιες μετρήσεις. Παράλληλα, είναι υπεύθυνος για την δημιουργία ad hoc δικτύου μέσω WiFi interface με οποιονδήποτε DTN κόμβο βρεθεί εντός της εμβέλειάς του.

Παρόμοια με τον κόμβο παραγωγό, ως κόμβος καταναλωτής χρησιμοποιήθηκε ένα Raspberry Pi 2B χωρίς τη χρήση του πομποδέκτη MRF24J40MA. Οι ad hoc συνδέσεις δημιουργήθηκαν με ένα WiFi USB dongle με αποσπώμενη κεραία.

Στο σενάριο των DTN κόμβων, κάθε φορά που ο IBR-DTN λάβει μία δέσμη με συγκεκριμένο EID (δηλαδή μόλις λάβει τα δεδομένα μίας μέτρησης), ο κόμβος καταναλωτή εκτελεί κάποια bash scripts με τα οποία αποστέλλει μια νέα δέσμη, ζητώντας τυχαία μια μέτρηση. Τα προαναφερόμενα bash scripts κάνουν χρήση των εργαλείων dtnsend, dtnrecv και dtntrigger του IBR-DTN.

Στο σενάριο των UMOBILE κόμβων, ο κόμβος καταναλωτή εκτελεί μια εφαρμογή NDN καταναλωτή η οποία κάνει χρήση της βιβλιοθήκης ndn-cxx. Η εφαρμογή αυτή αποστέλλει ορισμένα πακέτα Interest με συγκεκριμένο prefix και χρόνο ζωής. Κάθε φορά που επιστρέφει ένα πακέτο δεδομένων, η εφαρμογή στέλνει ένα νέο πακέτο Interest με τυχαίο prefix. Σε περίπτωση λήξης του πακέτου Interest, η εφαρμογή δοκιμάζει να ξαναστείλει ένα πακέτο Interest με ίδιο prefix. Κάθε πακέτο που στέλνει η εφαρμογή, προωθείται στον NFD και από εκεί οδηγείται στο DTN face.

5.6 Κινητός κόμβος

Η υλοποίηση του κινητού κόμβου στο εργαστήριο μπορεί να γίνει με χρήση της συσκευής Arietta G25. Η Arietta G25 βασίζεται στον AT91SAM9G25 της Microchip, χρονισμένο στα 400MHz, διαθέτει 128 MB μνήμης RAM και χρησιμοποιεί μια αφαιρούμενη SD κάρτα ως αποθηκευτικό μέσο. Η συνδεσιμότητα της εξασφαλίζεται από

την υποστήριξη ενός συμβατού με WiFi πομποδέκτη (WM1030WU). Η συσκευή τρέχει μια ελαφριά έκδοση του Linux, ειδικά φτιαγμένη για την Arietta με το εργαλείο buildroot, και υποστηρίζει πλήρως το IBR-DTN.

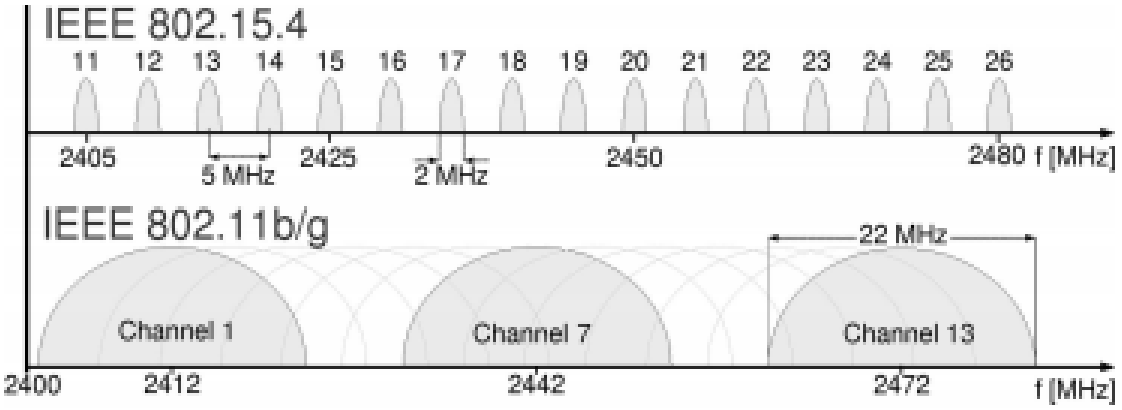
Εικόνα 18. Arietta G25

Η προσομοίωση της κίνησης του κόμβου στο εργαστήριο, πραγματοποιήθηκε με απλή ενεργοποίηση και απενεργοποίηση του WiFi interface με την χρήση ενός bash script. Με την εκτέλεση του bash script, ο κινητός κόμβος συνδέεται στο δίκτυο ad hoc 1 και παραλαμβάνει τα πακέτα του καταναλωτή για καθορισμένο χρόνο. Κατόπιν, ο κινητός κόμβος απενεργοποιεί το WiFi interface και παραμένει αποσυνδεδεμένος. Στη συνέχεια, συνδέεται στο ad hoc 2, προωθεί τα πακέτα του καταναλωτή και λαμβάνει τα πακέτα του παραγωγού. Όλοι οι προαναφερόμενοι κώδικες βρίσκονται στο Παράρτημα.

5.7 Τεχνικές δυσκολίες και περιορισμοί

Κατά την εκπόνηση της εργασίες ανέκυψαν ζητήματα σε σχέση με τη ρύθμιση του δικτύου και τη διεξαγωγή των πειραμάτων. Επειδή ο εντοπισμός και η επίλυση των συγκεκριμένων ζητημάτων αποτέλεσε σημαντικό μέρος της εργασίας, στο παρόν κεφάλαιο παρατίθεται σύντομη περιγραφή τους.

Αρχικά, επειδή τα πειράματα πραγματοποιήθηκαν στο χώρο του εργαστηρίου, τα δίκτυα WiFi και 802.15.4 συνυπήρχαν στον ίδιο χώρο. Η αμοιβαία παρεμβολή τους μπορεί να οδηγήσει σε αδυναμία επικοινωνίας, καθώς η χρήση των δύο τεχνολογιών γίνεται στη ζώνη ISM 2,4 GHz. Έτσι, ένα από τα προβλήματα που εμφανίστηκαν ήταν η αδυναμία λήψης πακέτων από το 802.15.4 δίκτυο. Το πρόβλημα αυτό επιλύθηκε διαλέγοντας μη επικαλυπτόμενα κανάλια.



Εικόνα 19. Απεικόνιση του φάσματος που χρησιμοποιείται από το 802.15.4 και το 802.11

Επίσης, η δημιουργία των δύο WiFi δικτύων, του κόμβου παραγωγού και του κόμβου καταναλωτή, δοκιμάστηκε με την ύπαρξη δύο διαφορετικών σημείων πρόσβασης (Access Points, AP). Ωστόσο, η διαδικασία εγκαθίδρυσης σύνδεσης είναι πολύ αργή και δεν βολεύει σε περιβάλλοντα με τακτές διακοπές. Για αυτό το λόγο, οι κόμβοι δημιουργούν από ένα ad hoc δίκτυο στο οποίο η διαδικασία σύνδεσης και αποσύνδεσης συμβαίνει πολύ γρήγορα.

Έτσι κάθε Raspberry Pi 2B με τη χρήση WiFi usb dongle θα πρέπει να δημιουργεί ένα ad hoc δίκτυο. Η ad hoc λειτουργία δεν υποστηρίζεται από όλα τα chipsets της Realtek [31]. Δοκιμάστηκαν αρκετά WiFi usb dongles και τελικά η ad hoc λειτουργία πραγματοποιήθηκε μόνο με αυτά που είχαν αποσπώμενη κεραία (συγκεκριμένα με ένα Crypto Airdata 54 Plus S2 και με ένα Tp-link tl-wn722n).

Ο κινητός κόμβος στα πειράματα συνδέεται και αποσυνδέεται διαρκώς μεταξύ των δύο ad hoc δικτύων. Με τη χρήση ενός bash script, επαναλαμβάνεται η εξής διαδικασία:

Αρχικά, ο κινητός κόμβος ενεργοποιεί το interface του WiFi και συνδέεται στο πρώτο δίκτυο. Έπειτα, απενεργοποιεί εντελώς το interface του WiFi και μετά ενεργοποιεί το interface του WiFi και συνδέεται στο δεύτερο δίκτυο. Ο κινητός κόμβος παραμένει σε κάθε μία από τις τρείς καταστάσεις (συνδεδεμένος στο ad hoc 1, αποσυνδεδεμένος, συνδεδεμένος στο ad hoc 2) για συγκεκριμένο χρονικό διάστημα. Κατά την δοκιμή μικρών χρονικών διαστημάτων (μικρότερων των είκοσι δευτερολέπτων) εμφανίζονταν σφάλματα και σταματούσε η εκτέλεση του bash script. Για αυτό το λόγο, στα πειράματα δοκιμάστηκαν χρόνοι μεγαλύτεροι από δέκα δευτερόλεπτα.

Σημαντικά προβλήματα παρατηρήθηκαν και κατά την λειτουργία της εφαρμογής IBR-DTN. Στα πειράματα χρησιμοποιήθηκε μόνο η δρομολόγηση τύπου flooding, καθώς με την υλοποίηση τύπου epidemic δεν αποστέλλονταν αρκετά πακέτα. Επίσης, η εφαρμογή στον κινητό κόμβο (Arietta G25) σταματούσε να λειτουργεί στην προσπάθεια διεξαγωγής πολλών πειραμάτων, με αποτέλεσμα την επανάληψη τους αρκετές φορές.

Τέλος, λόγω της μεγάλης χρονικής διάρκειας των πειραμάτων και της ύπαρξης όλων των προαναφερόμενων προβλημάτων δεν ήταν εφικτή η διεξαγωγή μεγάλου αριθμού ολοκληρωμένων επαναλήψεων των πειραμάτων.

ΚΕΦΑΛΑΙΟ 6: Πειραματικό μέρος

6.1 Εισαγωγή πειραματικού μέρους

Για τη μελέτη απόδοσης των δύο προαναφερόμενων σεναρίων χρειάζεται η διεξαγωγή ισοδύναμων πειραμάτων με κοινές μετρικές και παραμέτρους. Τα πειράματα που πραγματοποιήθηκαν αποτελούν μια «αλυσίδα», καθώς τα βέλτιστα αποτελέσματα των προηγουμένων εφαρμόζονται στα επόμενα, με σκοπό την εύρεση της αποδοτικότερης δικτυακής αρχιτεκτονικής και των ευνοϊκότερων παραμέτρων για τα συγκεκριμένα σενάρια. Οι μετρήσεις των πειραμάτων αναφέρονται αναλυτικά στο παράρτημα. Σε κάθε σύνδεση αποστελλόταν ένα μόνο bundle από τον κάθε κόμβο, ώστε τα αποτελέσματα των πειραμάτων να προκύπτουν λαμβάνοντας υπόψη την καλύτερη δυνατή περίπτωση. Η αποστολή περισσότερων πακέτων ανά σύνδεση και η εύρεση μεθόδων ελέγχου συμφόρησης αποτελεί μελλοντική εργασία.

Συνοπτικά, στο πείραμα 1 γίνεται σύγκριση της DTN αρχιτεκτονικής, αφενός με την UMOBILE αρχιτεκτονική με μικρό freshness και αφετέρου με την UMOBILE αρχιτεκτονική με μεγάλο freshness. Η παράμετρος που μεταβάλλεται σε αυτά τα πειράματα είναι ο χρόνος παραμονής του κινητού κόμβου σε κάθε κατάσταση. Οι μετρικές που συγκρίνονται είναι ο χρόνος μετάβασης και επιστροφής (Round-Trip Time, RTT) και οι διαδρομές του κινητού κόμβου (Round-Trips). Στο πείραμα 2 διατηρείται σταθερός ο χρόνος παραμονής του κινητού κόμβου σε κάθε κατάσταση και το freshness της UMOBILE αρχιτεκτονικής που έδωσε τα βέλτιστα αποτελέσματα στο πείραμα 1, ενώ μεταβάλουμε τον αριθμό των διαφορετικών ονομάτων. Οι μετρικές που συγκρίνονται είναι το RTT και το Cache hit ratio. Στο πείραμα 3 εξετάζεται η απόδοση της UMOBILE αρχιτεκτονικής, μεταβάλλοντας το lifetime των Interests και μετρώντας τη συνολική διάρκεια λήψης δέκα πακέτων μετρήσεων και το Success Ratio. Τέλος, στο πείραμα 4 υπολογίζεται η συνολική διάρκεια λήψης δέκα πακέτων μετρήσεων, το Success Ratio και το Cache hit ratio, μεταβάλλοντας το freshness των πακέτων.

6.2 Πείραμα 1

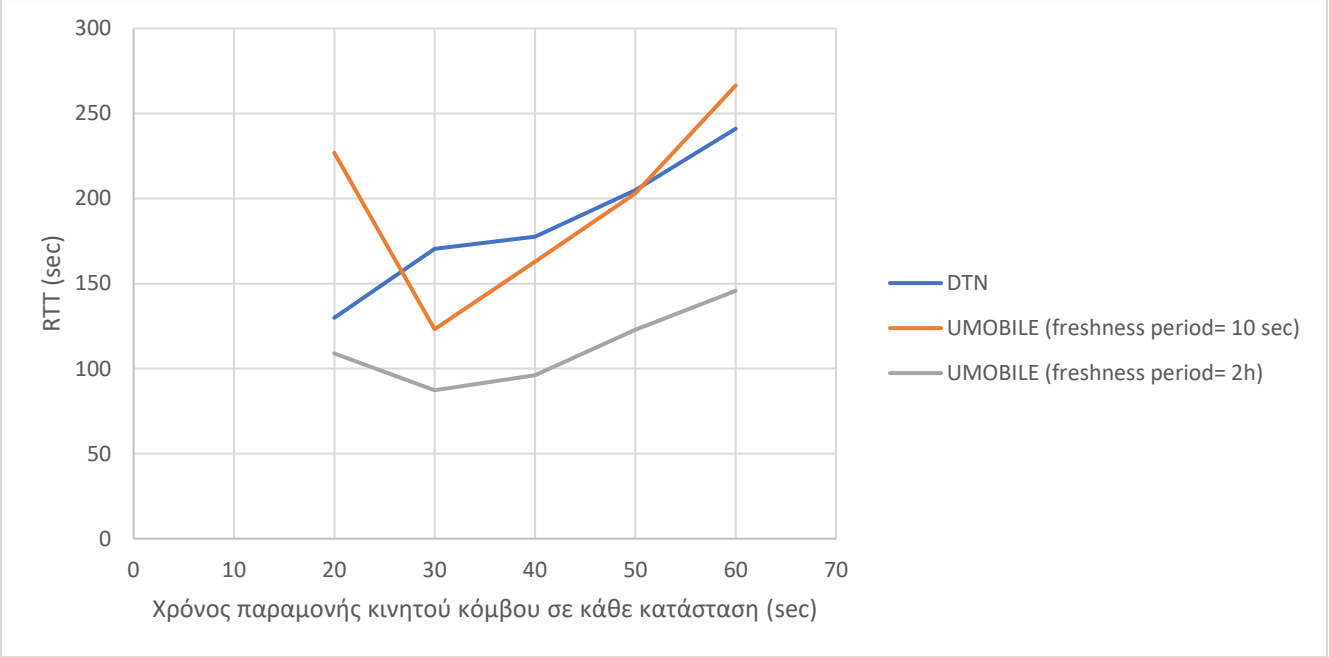
Σε αυτό το πείραμα γίνεται έλεγχος της απόδοσης του δικτύου μεταβάλλοντας το χρόνο παραμονής του κινητού κόμβου σε μια από τις τρεις καταστάσεις (συνδεδεμένος στο δίκτυο adhoc 1, αποσυνδεδεμένος, συνδεδεμένος στο δίκτυο adhoc 2). Ο κινητός κόμβος είναι DTN και λειτουργεί ως data mule. Σε κάθε περίπτωση υπάρχουν έξι διαφορετικά ονόματα (name prefix), δηλαδή όσες και οι διαφορετικές μετρήσεις που στέλνει ένας κόμβος αισθητήρων. Ο καταναλωτής σε κάθε επαφή με τον κόμβο ζητά με τυχαίο τρόπο ένα όνομα από τα έξι. Το πείραμα αυτό χωρίζεται σε τρεις υποκατηγορίες:

Πείραμα 1.1: Ο παραγωγός και ο καταναλωτής είναι DTN κόμβοι. Ο καταναλωτής πρέπει να στείλει ένα bundle ζητώντας από τον παραγωγό τα συγκεκριμένα δεδομένα. Έπειτα, ο data mule μεταφέρει το bundle στον παραγωγό, ο οποίος με την σειρά του προωθεί την ζητούμενη μέτρηση στέλνοντας ένα bundle στον καταναλωτή. Ο data mule λαμβάνει αυτό το bundle και το προωθεί στον καταναλωτή. Η παραπάνω διαδικασία επαναλαμβάνεται 10 φορές. Ο καταναλωτής στέλνει ένα bundle στον παραγωγό ζητώντας τα συγκεκριμένα δεδομένα (και δεν στέλνει ο παραγωγός από μόνος του τα δεδομένα των μετρήσεων), προκειμένου να γίνει η αντίστοιχη σύγκριση με το σενάριο της UMOBILE αρχιτεκτονικής.

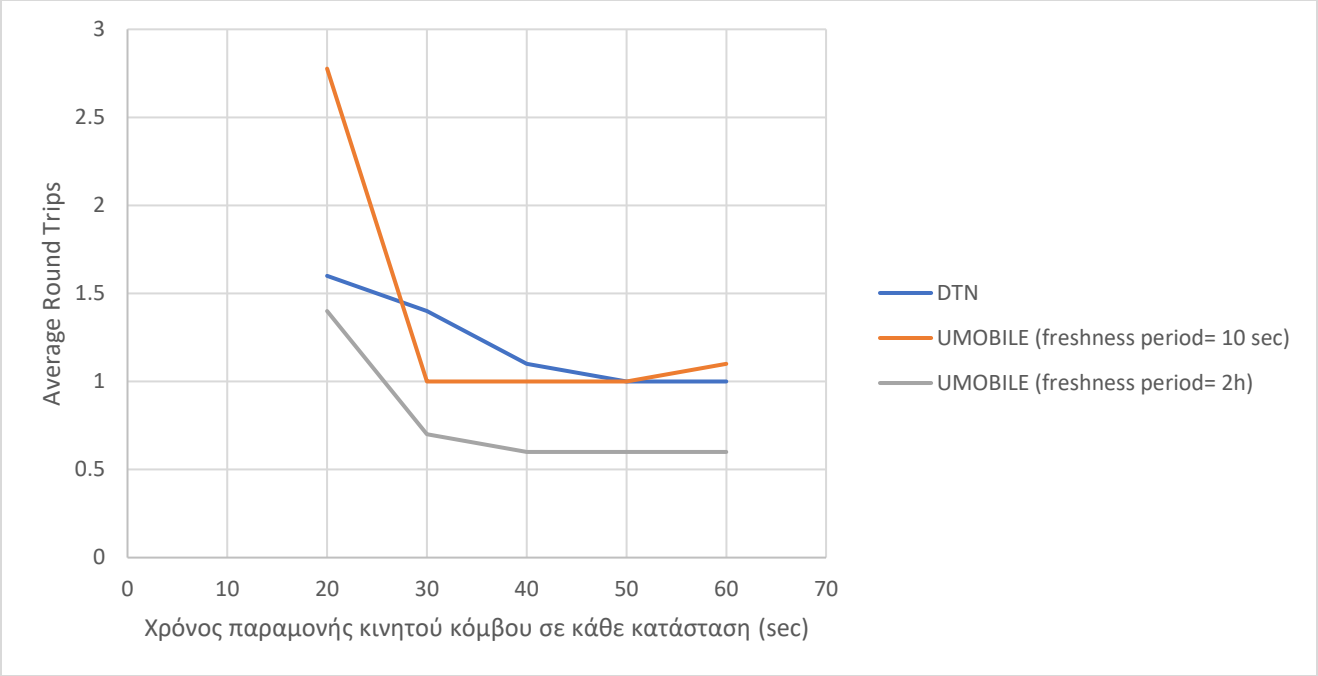
Πείραμα 1.2: Ο παραγωγός και ο καταναλωτής είναι UMOBILE κόμβοι. Ο καταναλωτής στέλνει ένα πακέτο Interest το οποίο μετατρέπεται σε bundle και προωθείται μέσω του data mule στον παραγωγό. Ο παραγωγός προωθεί το αντίστοιχο πακέτο δεδομένων στο DTN face. Το πακέτο αυτό ενθυλακώνεται σε ένα bundle πακέτο και μεταφέρεται μέσω του data mule στον καταναλωτή. Η παραπάνω διαδικασία επαναλαμβάνεται 10 φορές με την χρήση Freshness period 10 δευτερολέπτων. Έτσι, ο καταναλωτής μπορεί να λάβει κατευθείαν τα δεδομένα που ζητά από την cache μόνο αν στείλει διαδοχικά Interests με ίδιο prefix.

Πείραμα 1.3: Ο παραγωγός και ο καταναλωτής είναι UMOBILE κόμβοι. Το πείραμα αυτό διαφέρει από το πείραμα 1.2 διότι χρησιμοποιείται Freshness period 2 ωρών. Ο καταναλωτής μπορεί να λάβει κατευθείαν τα δεδομένα που ζητά από την cache (εφόσον τα δεδομένα βρίσκονται σε αυτή).

Στο πείραμα 1.1 χρησιμοποιήθηκε για το DTN μνήμη με default size και default lifetime των bundles. Στα πειράματα 1.2 και 1.3 χρησιμοποιήθηκαν οι ίδιες ρυθμίσεις με το πείραμα 1.1 για το DTN, ενώ για τα πακέτα Interest χρησιμοποιήθηκε lifetime=3600 sec.



Διάγραμμα 1



Διάγραμμα 2

Θεωρούμε ότι ο κινητός κόμβος συνδέεται αρχικά για χρόνο DelayState στο δίκτυο Adhoc1 (σύνδεση με τον καταναλωτή). Έπειτα, παραμένει αποσυνδεδεμένος για χρόνο DelayState, συνδέεται στο δίκτυο Adhoc2 (σύνδεση με τον παραγωγό) για χρόνο DelayState, παραμένει αποσυνδεδεμένος για χρόνο DelayState και τέλος συνδέεται στο δίκτυο Adhoc1 για χρόνο DelayState.

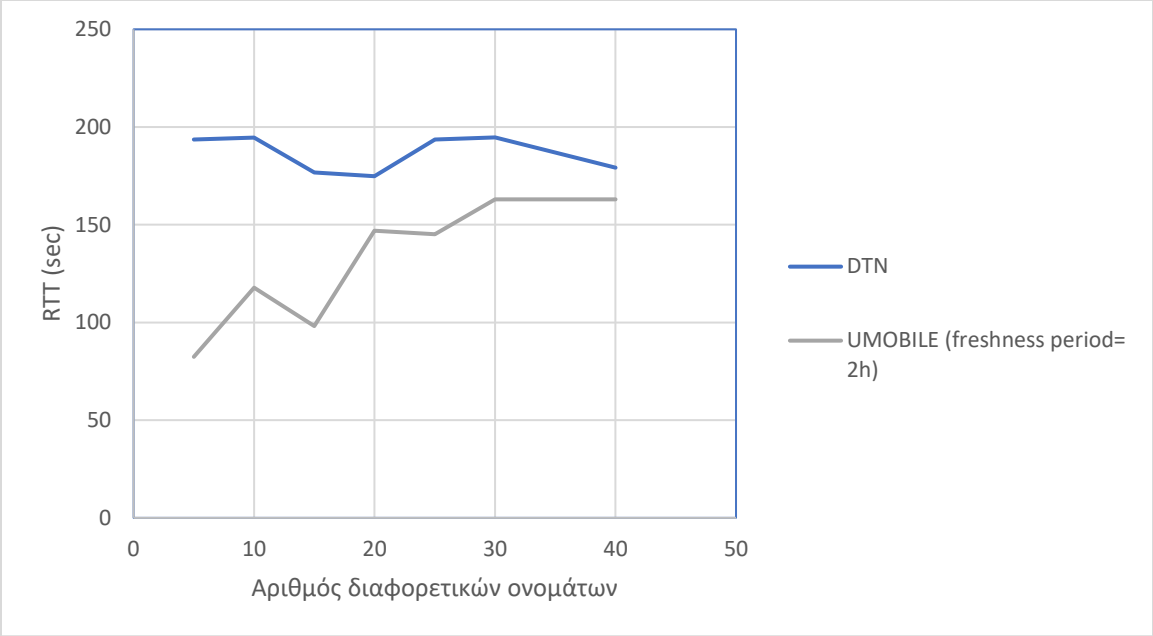
Ένα πακέτο μπορεί να σταλεί από τον ένα κόμβο στον άλλο από τη στιγμή της σύνδεσης τους μέχρι και λίγο πριν από την διακοπή της σύνδεσης. Συνεπώς, η χειρότερη περίπτωση ενός Round Trip είναι 5*DelayState. Ωστόσο, η χρήση μικρών χρόνων DelayState, το αυξανόμενο μέγεθος των μηνυμάτων, η ανάγκη ύπαρξης συγχρονισμού και η αυξανόμενη καθυστέρηση λόγω του συνδυασμού των δύο πρωτοκόλλων μπορούν να οδηγήσουν στην αδυναμία αποστολής ενός πακέτου μέσα σε χρόνο DelayState. Το τελευταίο έχει ως αποτέλεσμα την ανάγκη περισσότερων Round Trips του κινητού κόμβου. Για παράδειγμα, αν κατά την πρώτη σύνδεση του κινητού κόμβου με τον παραγωγό καταφέρει ο κινητός κόμβος να στείλει το bundle με το οποίο ο καταναλωτής ζητά δεδομένα από τον παραγωγό αλλά δεν καταφέρει να λάβει το αντίστοιχο πακέτο δεδομένων, τότε η επόμενη ευκαιρία να λάβει το bundle θα είναι μετά από χρόνο 4*DelayState. Έτσι, στην περίπτωση που χρειάζονται δύο επαφές του κινητού κόμβου με

τον παραγωγό, το συνολικό RTT θα είναι 5*DelayState + 4*DelayState = 9*DelayState. Με αυτόν τον τρόπο υπολογίζοντας τον κάθε φορά χρόνο επιστροφής είναι δυνατόν ο υπολογισμός των συνολικών Round Trips. Η σύγκριση της απόδοσης των τριών υποπειραμάτων του πρώτου πειράματος θα πρέπει να γίνει τόσο συναρτήσει του χρόνου RTT όσο και βάσει του αριθμού Round Trips.

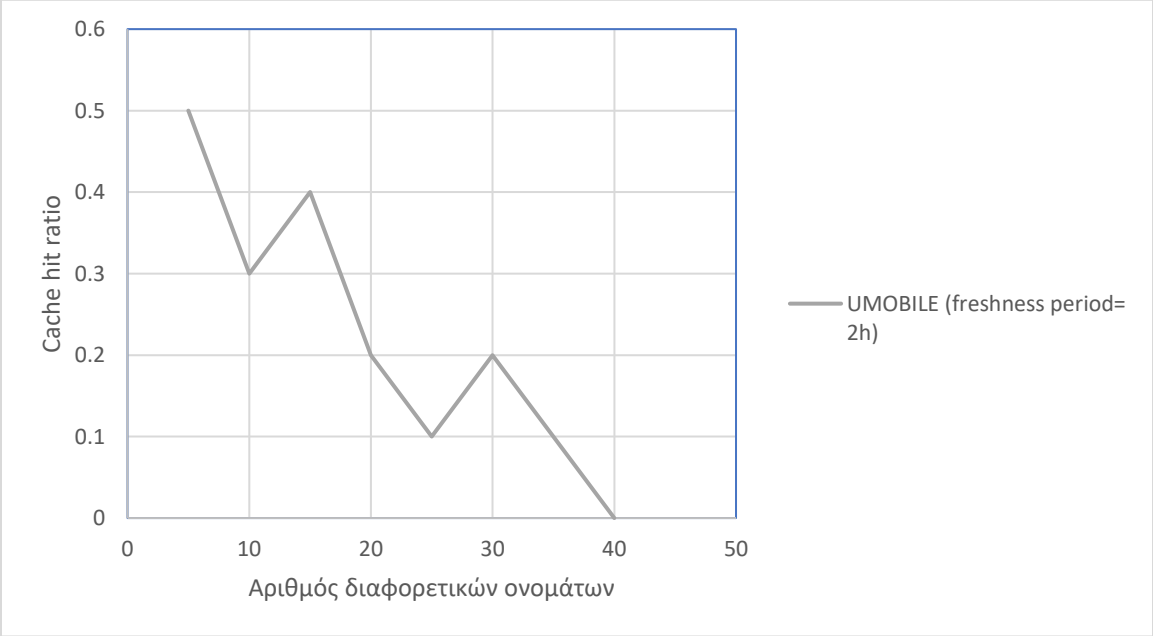
Σχολιασμός αποτελεσμάτων: Παρατηρείται ότι ο μικρότερος χρόνος DelayState στον οποίο επιτυγχάνεται μικρότερος Average RTT και Average Round Trips είναι ο DelayState= 40 sec. Για χρόνους DelayState μεγαλύτερους από 40 sec, ο μέσος αριθμός των Round Trips παραμένει σταθερός καθώς ο χρόνος επαρκεί για την ανταλλαγή των απαραίτητων πακέτων. Για χρόνους μικρότερους των 40 sec, ο μέσος αριθμός των Round Trips είναι μεγαλύτερος από την περίπτωση του DelayState =40 sec, καθώς ο χρόνος μιας σύνδεσης δεν επαρκεί για την ανταλλαγή των απαραίτητων πακέτων. Στην περίπτωση DelayState =10 sec δεν επιτεύχθηκε η ανταλλαγή πακέτων στα πειράματα 1.2 και 1.3. Αυτό οφείλεται τόσο στο μικρό διάστημα σύνδεσης των κόμβων όσο και στο ότι το μέγεθος του bundle που ενθυλακώνει ένα Interest (160 με 170 Bytes) και το μέγεθος του bundle που ενθυλακώνει ένα πακέτο δεδομένων (περίπου 513 Bytes) είναι αρκετά μεγαλύτερα από το μέγεθος ενός απλού bundle (149 με 160 Bytes). Επίσης παρατηρούμε ότι για freshness period = 2 ώρες μειώνεται σημαντικά το συνολικό RTT. **Συνεπώς επιλέγουμε για την υλοποίηση των επόμενων πειραμάτων DelayState= 40 sec και freshness= 2 ώρες.**

6.3 Πείραμα 2

Σε αυτό το πείραμα διατηρούμε σταθερό DelayState=40 sec και freshness period=2 ώρες (βάσει των αποτελεσμάτων του πειράματος 1) και μεταβάλουμε τον αριθμό των διαφορετικών ονομάτων (5, 10, 15, 20, 25, 30, 35, 40).



Διάγραμμα 3



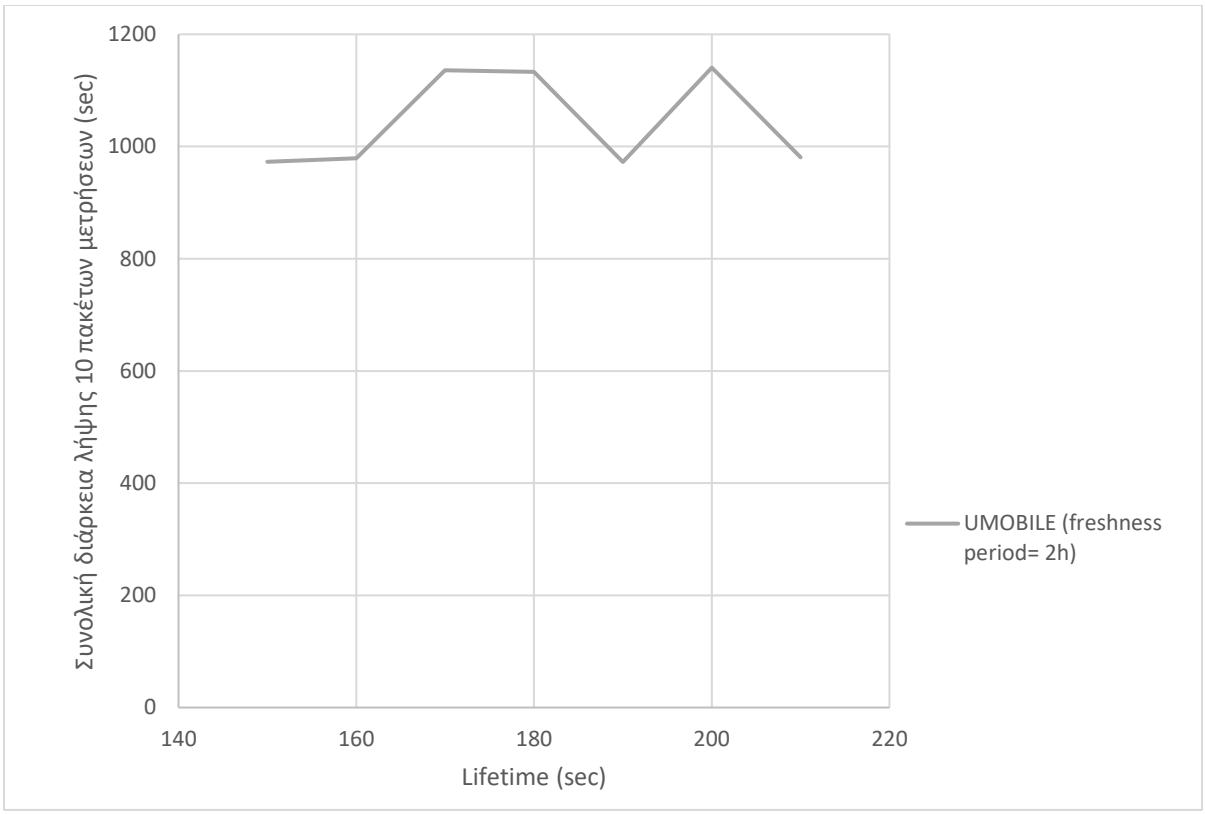
Διάγραμμα 4

Σχολιασμός αποτελεσμάτων: Παρατηρείται από το Διάγραμμα 3, ότι η αρχιτεκτονική DTN έχει σχεδόν σταθερές τιμές του RTT καθώς αλλάζει ο αριθμός των διαφορετικών ονομάτων, επειδή δεν κάνει χρήση caching. Η UMOBILE αρχιτεκτονική

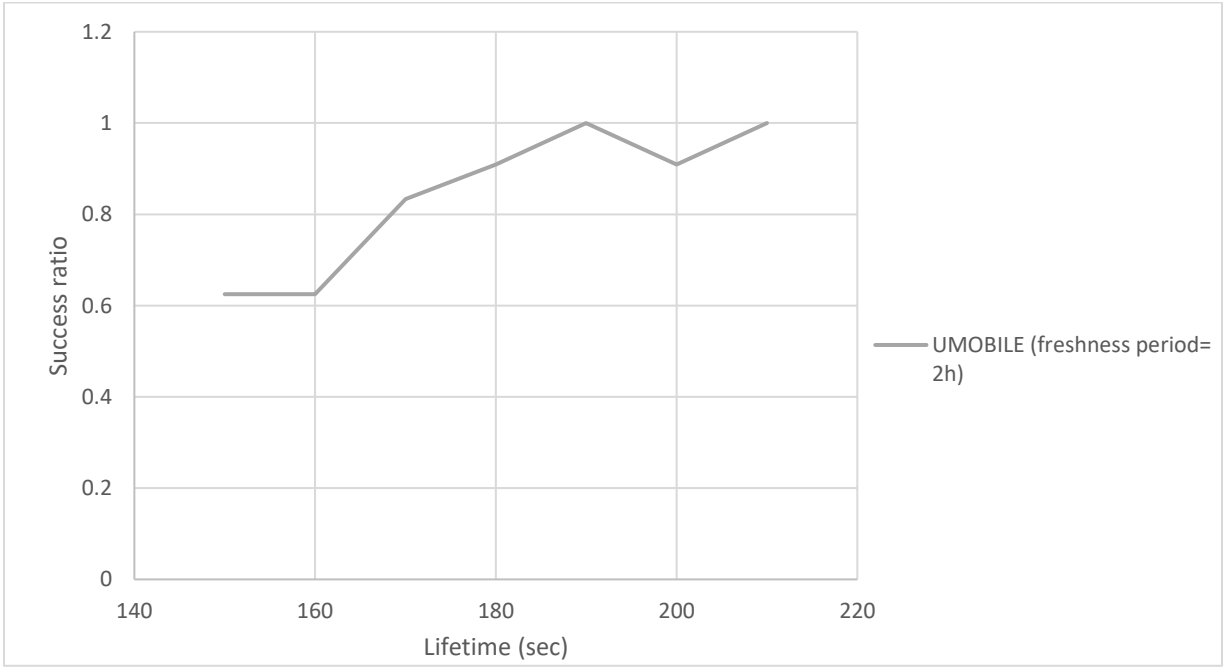
για μικρό αριθμό ονομάτων εμφανίζει σημαντικά μικρότερο RTT επειδή είναι μεγαλύτερη η πιθανότητα αποστολής Interests με ίδιο prefix. Σύμφωνα με το Διάγραμμα 4, για μικρό αριθμό διαφορετικών ονομάτων, το Cache hit ratio παίρνει μεγάλες τιμές. Αντίθετα, μειώνεται σημαντικά καθώς αυξάνεται ο αριθμός των διαφορετικών ονομάτων. **Επομένως, για να υπάρχουν αρκετά διαφορετικά ονόματα, μικρό RTT και να αξιοποιούνται οι δυνατότητες caching της UMOBILE αρχιτεκτονικής, στα επόμενα πειράματα επιλέγεται η χρήση 15 διαφορετικών ονομάτων (RTT=98.13 sec και Cache hit ratio=0.4). Επίσης, επειδή η UMOBILE αρχιτεκτονική εμφανίζει καλύτερα αποτελέσματα από την DTN αρχιτεκτονική, στα επόμενα πειράματα θα γίνει περαιτέρω μελέτη της απόδοσης της UMOBILE αρχιτεκτονικής.**

6.4 Πείραμα 3

Σε αυτό το πείραμα διατηρείται σταθερό DelayState=40 sec (βάσει των αποτελεσμάτων του πειράματος 1) και 15 διαφορετικά ονόματα (βάσει των αποτελεσμάτων του πειράματος 2) freshness period=2h, ενώ μεταβάλλεται το lifetime των Interests. Αρχικά υπολογίζεται η συνολική διάρκεια για την λήψη των δέκα μετρήσεων από τον καταναλωτή συναρτήσει του lifetime. Έπειτα, γίνεται σύγκριση του ποσοστού των επιτευχθέντων Interests για διαφορετικές τιμές του lifetime. Οι τιμές του Lifetime ξεκινούν από την καλύτερη περίπτωση μεταφοράς πακέτου -10 δευτερόλεπτα (δηλαδή 4*DelayState-10=150) και αυξάνονται κατά 10 έως την χειρότερη περίπτωση μεταφοράς πακέτου +10 δευτερόλεπτα (δηλαδή 5*DelayState+10=210).



Διάγραμμα 5

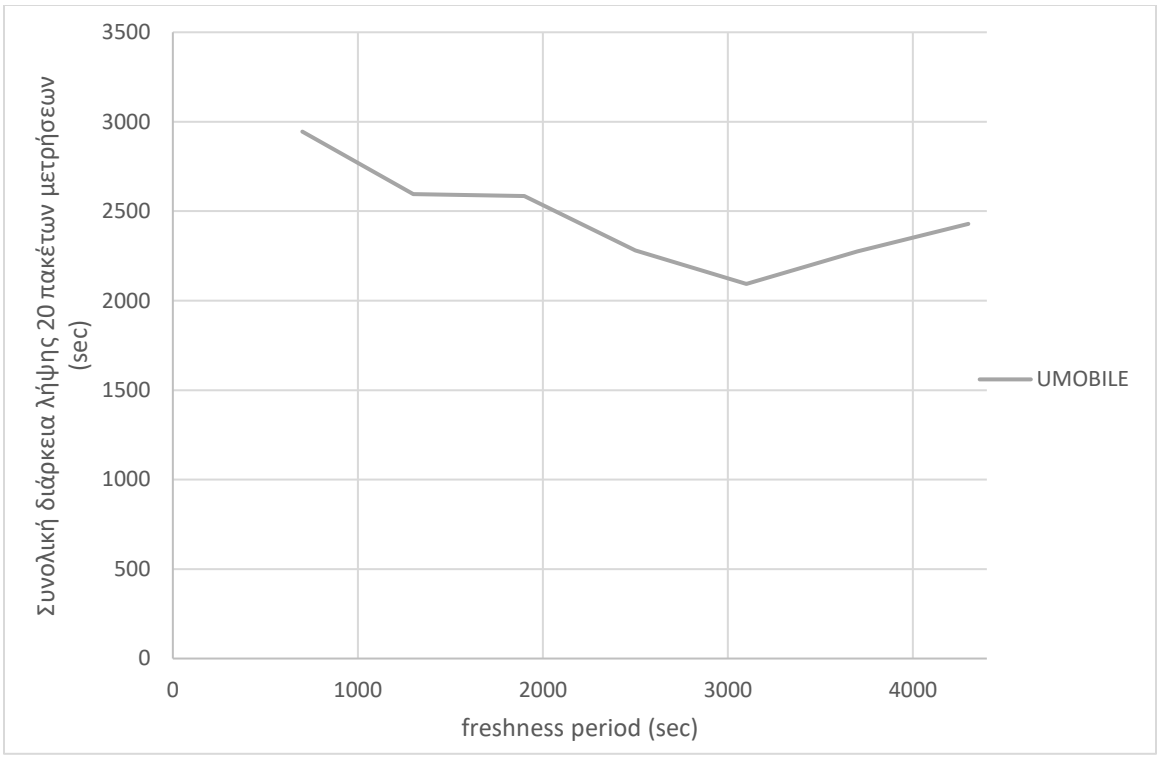


Διάγραμμα 6

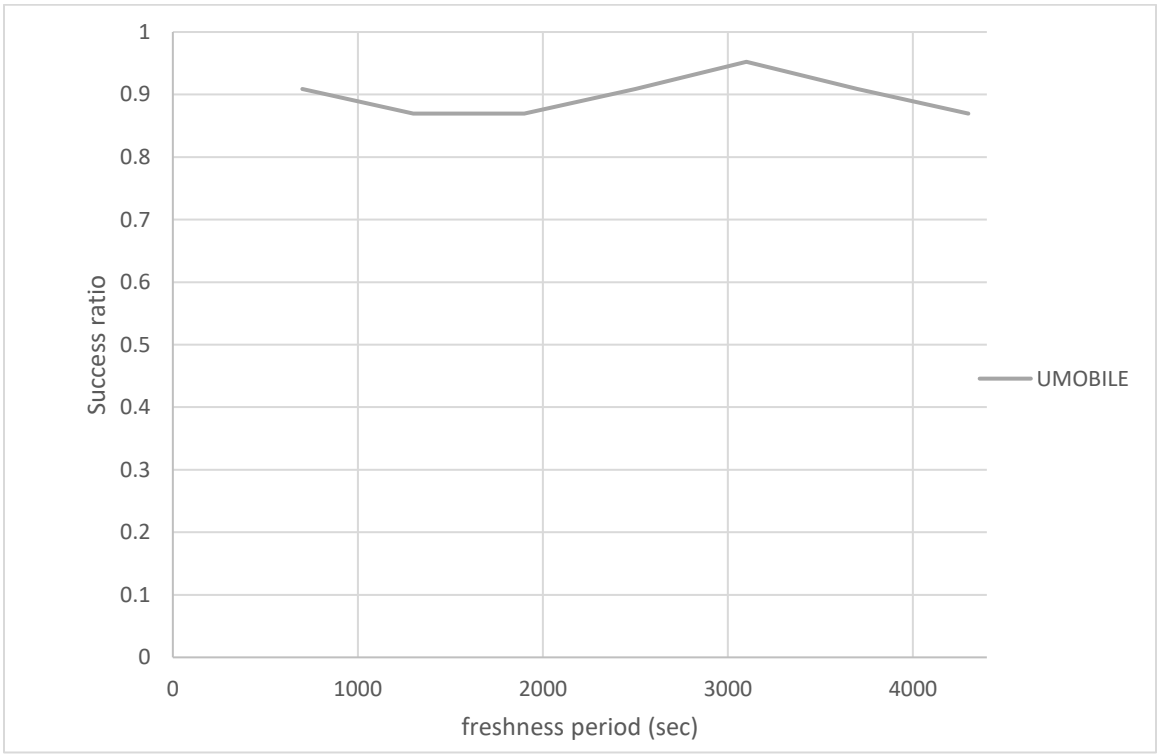
Σχολιασμός αποτελεσμάτων: Σχετικά με το Διάγραμμα 5 παρατηρείται ότι η διάρκεια λήψης δέκα μετρήσεων παραμένει σχετικά σταθερή. Αυτό οφείλεται στο ότι μόλις ένα πακέτο υπερβεί το lifetime του, ο καταναλωτής δοκιμάζει να ξαναστείλει ένα Interest με το ίδιο prefix. Όπως φαίνεται και στις μετρήσεις του πειράματος (στο Παράρτημα), σε πολλές περιπτώσεις που προκαλούνταν Timeout, το δεύτερο Interest που στελνόταν, λάμβανε πολύ σύντομα τα δεδομένα που ζητούσε και προκαλούσε ένα μικρό RTT. Αυτό συνέβαινε διότι ο κινητός κόμβος είχε το πακέτο δεδομένων που είχε ζητηθεί από το πρώτο Interest και συνδεόταν στον καταναλωτή λίγο μετά τη λήξη του πρώτου Interest (και άρα λίγα δευτερόλεπτα μετά την αποστολή του δεύτερου Interest). **Παρατηρήθηκε ότι ο μικρότερος δυνατός χρόνος lifetime ενός Interest με το μεγαλύτερο Success ratio είναι η περίπτωση με lifetime=190 sec.**

6.5 Πείραμα 4

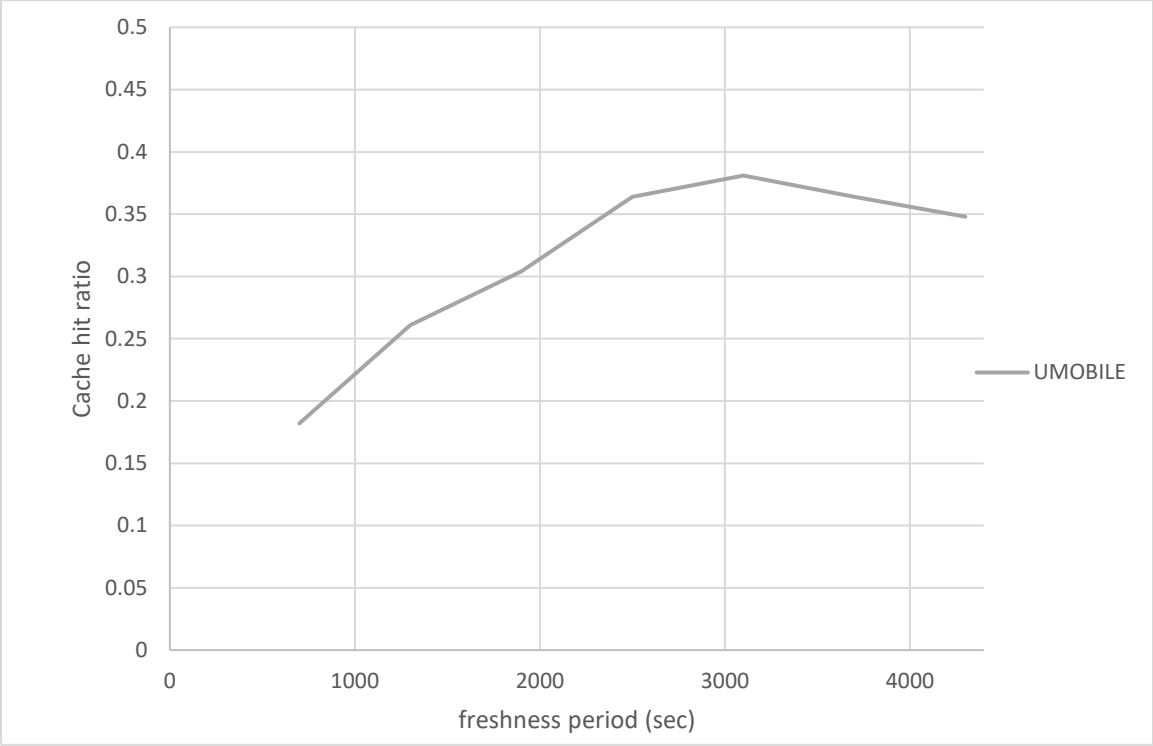
Σε αυτό το πείραμα διατηρείται σταθερό DelayState=40 sec (βάσει των αποτελεσμάτων του πειράματος 1) και 15 διαφορετικά ονόματα (βάσει των αποτελεσμάτων του πειράματος 2), lifetime των Interests=190 δευτερόλεπτα και μεταβάλλουμε το freshness period σε πολλαπλάσια του αναμενόμενου RTT. Προκειμένου να εξασφαλιστεί μια πιο σταθερή μέση τιμή του Success Ratio και να προκύψουν πιο αξιόπιστα αποτελέσματα, στο Πείραμα 4 πραγματοποιήθηκαν 20 επαναλήψεις ανά υποπείραμα**.** Για DelayState= 40 sec η χειρότερη περίπτωση RTT είναι 200 sec, ενώ η καλύτερη περίπτωση είναι 160 sec. Κατά την διεξαγωγή των προηγούμενων πειραμάτων σημειώθηκαν περιπτώσεις που χρειάστηκαν έως και τρία Round Trips του κινητού κόμβου. Για να αξιοποιηθεί το όφελος του caching ακόμα και στη χειρότερη περίπτωση (όπου το συνολικό RTT φτάνει τα 3*(5*DelayState)=600 δευτερόλεπτα) το freshness period ξεκινά με αφετηρία τα 700 δευτερόλεπτα και αυξάνεται κατά 3*(5*DelayState)=600 sec ανά υποπείραμα μέχρι τα 4300 δευτερόλεπτα.



Διάγραμμα 7



Διάγραμμα 8



Διάγραμμα 9

Σχολιασμός αποτελεσμάτων: Από το Διάγραμμα 7 παρατηρείται ότι υπάρχει σημαντική μείωση της διάρκειας λήψης είκοσι πακέτων καθώς αυξάνεται το freshness period των πακέτων. Αυτό οφείλεται κυρίως στην αύξηση του Cache hit ratio όπως φαίνεται στο Διάγραμμα 9, καθώς τα δεδομένα με μεγάλο freshness παραμένουν για μεγαλύτερο χρονικό διάστημα στο Content Store και άρα αυξάνεται η πιθανότητα εύρεσης των δεδομένων σε αυτό. Το Διάγραμμα 8 απεικονίζει σχεδόν σταθερό το Success Ratio με τιμή κοντά στα 0.9 και επαληθεύει ότι αυτά τα πειράματα διεξήχθησαν λαμβάνοντας σωστές παραμέτρους από τα προηγούμενα. Από τα παραπάνω διαγράμματα εξάγεται ως καλύτερος χρόνος freshness period τα 3000 δευτερόλεπτα, κατά τον οποίο εξασφαλίζεται η μικρότερη συνολική διάρκεια λήψης 10 πακέτων με το μεγαλύτερο cache hit ratio.

Συμπεράσματα

Τα κυριότερα συμπεράσματα της μελέτης μπορούν να συνοψιστούν ως εξής:

α) Από τα πειράματα 1 και 2 παρατηρείται ότι παρά την επιπλέον επιβάρυνση που εισάγει η UMOBILE αρχιτεκτονική (καθώς το μέγεθος των πακέτων είναι αρκετά μεγαλύτερο από το μέγεθος ενός απλού bundle και η χρήση δύο πρωτοκόλλων επιβαρύνει σημαντικά τον χρόνο επεξεργασίας των πακέτων) υπερέχει σε απόδοση έναντι της DTN αρχιτεκτονικής. Συγκεκριμένα, από το πείραμα 1 προκύπτει ότι τόσο η υλοποίηση IBR-DTN όσο και η υλοποίηση UMOBILE δεν λειτουργούν αποδοτικά για συνδέσεις διάρκειας μικρότερης των 20 δευτερολέπτων, ο καλύτερος χρόνος σύνδεσης και για τις δύο υλοποιήσεις είναι τα 40 δευτερόλεπτα και η UMOBILE αρχιτεκτονική με freshness period 2 ώρες είναι αποδοτικότερη τόσο της DTN αρχιτεκτονικής όσο και της UMOBILE με freshness period 10 δευτερόλεπτα. Στο πείραμα 2 αποδεικνύεται ότι η χρήση του caching καθιστά την UMOBILE αρχιτεκτονική πιο αποδοτική από την DTN αρχιτεκτονική.

β) Στα πειράματα 3 και 4, πραγματοποιήθηκε μελέτη της απόδοσης της UMOBILE αρχιτεκτονικής. Στο πείραμα 3, μελετήθηκε η παράμετρος lifetime των NDN πακέτων και προέκυψε ως πιο αποδοτική η διάρκεια lifetime 190 δευτερολέπτων για την συγκεκριμένη εφαρμογή. Στο πείραμα 4, μελετήθηκε η απόδοση της UMOBILE αρχιτεκτονικής σε σχέση με το freshness period των πακέτων και επιλέχθηκε ως βέλτιστη μια μεγάλη τιμή, προκειμένου να αξιοποιείται το caching.

Το γενικό συμπέρασμα το οποίο εξάγεται είναι ότι η UMOBILE αρχιτεκτονική αποτελεί μια αποδοτική λύση για τη διασύνδεση απομακρυσμένων περιοχών που μπορεί να εφαρμοστεί με μεγάλη αποτελεσματικότητα στο Διαδίκτυο των Αντικειμένων, κάνοντας χρήση των αποτελεσμάτων της παρούσας εργασίας.

Ανοιχτά ζητήματα και μελλοντική εργασία

Στην παρούσα διπλωματική εργασία μελετήθηκε η αποδοτικότητα των UMOBILE και DTN αρχιτεκτονικών με πειράματα που αποστέλλονταν ένα πακέτο από κάθε κόμβο ανά σύνδεση. Αναδείχθηκαν όμως ζητήματα που θα μπορούσαν να αποτελέσουν αντικείμενο μελλοντικών εργασιών.

Η μελέτη αυτή μπορεί να επεκταθεί σε σενάρια στα οποία να αποστέλλονται πολλά πακέτα σε κάθε σύνδεση καθώς και στη διερεύνηση μεθόδων ελέγχου συμφόρησης. Μια πιο ολοκληρωμένη μελέτη του κινητού δικτύου απαιτεί την προσθήκη περισσότερων κινητών κόμβων και η εφαρμογή κατάλληλων πρωτοκόλλων δρομολόγησης. Επιπλέον, η UMOBILE αρχιτεκτονική μπορεί να τροποποιηθεί, εφαρμόζοντας συσσώρευση περισσότερων NDN πακέτων σε ένα bundle με σκοπό την αύξηση της απόδοσής της. Τέλος, η εφαρμογή της UMOBILE αρχιτεκτονικής στο Διαδίκτυο των Αντικειμένων μπορεί να μελετηθεί περεταίρω, μεταφέροντάς την σε συσκευές περιορισμένων πόρων και συγκρίνοντάς την με άλλα υπάρχοντα IoT πρωτόκολλα.

Βιβλιογραφία

[1] V. Cerf *et al.*, "Space for Internet and Internet for Space", Ad Hoc Networks, Special Issue on New Research Challenges in Mobile, Opportunistic & Delay-Tolerant Networks, Elsevier, 2014.

[2] Delay- and Disruption-Tolerant Networks (DTNs): A Tutorial, Version 3.2, 2015.

[3] K. Fall, "A Delay-Tolerant Network Architecture for Challenged Internets," Proc. 2003 Conf. Applications, Technologies, Architectures and Protocols for Computer Commin., 2003, pp. 27-34.

[4] V. Cerf *et al.*, "Delay-Tolerant Networking Architecture," *RFC 4838,* 2007.

[5] K. Scott. and S. Burleigh, "Bundle Protocol Specification," *RFC 5050,* November 2007.

[6] G. Xylomenos *et al.*, "A Survey of Information-Centric Networking Research," *IEEE Commun. Surveys & Tutorials*, vol. 16, no. 2, 2014. pp. 1024-49.

[7] A. Afanasyev *et al.*, "A Brief Introduction to Named Data Networking," *in IEEE MILCOM 2018*.

[8] L. Zhang *et al.*, "Named Data Networking," *ACM Computer Communication Reviews*, Jun. 2014.

[9] C. Yi *et al*, "A case for stateful forwarding plane," *Computer Communications: ICN Special Issue,* vol. 36, no. 7, pp. 779-791, April 2013.

[10] A. Pereira da Silva, S. Burleigh and K. Obraczka, "DTN of Things" in *Delay and Disruption Tolerant Networks: Interplanetary and Earth-Bound Architecture, Protocols and Applications,* CRC Press, pp. 276-307, 2019.

[11] K. Loupos *et al.*, "Structural health monitoring system for bridges based on skin-like sensor," IOP Conf. Series: Materials Science and Engineering 236 012100, 2017.

[12] L. Baumgärtner *et al.*, "Environmental Monitoring Using Low-Cost Hardware and Infrastructureless Wireless Communication,"in *IEEE Global Humanitarian Technology Conference (GHTC)*, Oct. 2018.

[13] Y.Zguira *et al.*, "IoB-DTN: A lightweight DTN protocol for mobile IoT Applications to smart bike sharing systems," in *Proceedings of the 2018 Wireless Days (WD)*, April 2018.

[14] M. Amadeo *et al.*, "Named data networking for IoT: An architectural perspective," in *Proc. Eur. Conf. Netw. Commun. (EuCNC)*, pp. 1-5, June 2014.

[15] Emmanuel Baccelli *et al.*, "Information Centric Networking in the IoT: Experiments with NDN in the Wild," in *Proc. of 1st ACM Conf. on Information-Centric Networking* , 2014.

[16] V. A. Siris *et al.*, "Supporting the IoT over Integrated Satellite-Terrestrial Networks using Information-Centric Networking," in *8th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, 2016.

[17] C. Gündoğan *et al.*, "NDN, CoAP, and MQTT: A Comparative Measurement Study in the IoT," in *Proc. of ACM ICN*, 2018.

[18] C.A. Sarros *et al.*, "Connecting the Edges: A Universal, Mobile-Centric, and Opportunistic Communications Architecture," in *IEEE Communications Magazine*, February 2018.

[19] Jet Propulsion Laboratory. California Institute of Technology. CA, "Interplanetary Overlay Network (ION)".

[20] N. Bezirgiannidis *et al.*, "Towards Flexibility and Accuracy in Space DTN Communications," *in 8th ACM MobiCom Workshop on Challenged Networks, (CHANTS'13), Miami, Florida, USA*, 30 September 2013.

[21] M. Doering *et al.*, "IBR-DTN: an efficient implementation for embedded systems," in *Third ACM workshop on Challenged networks*, pp. 117-120, September 2008.

[22] "802.15.4-2011 - IEEE Standard for Local and metropolitan area networks--Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)".

[23] S. Symingtongro *et al.*, "Bundle Security Protocol Specification," *RFC 6257*, May 2011.

[24] G. Von Zengen *et al.*, "An Overview of µDTN: Unifying DTNs and WSNs," in *11th GI/ITG KuVS Fachgespräch Drahtlose Sensornetze,* 2012.

[25] S. Rottmann *et al.*, "miniDTN: A DTN Stack for 5€-WiFi-Nodes," *16th GI/ITG KuVS Fachgespräch Drahtlose Sensornetze,* pp. 1-4, September 2017.

[26] "NFD - Named Data Networking Forwarding Daemon," 2014.

[27] A. Aboodi *et al.*, "Survey on the Incorporation of NDN/CCN in IoT," in *IEEE Access*, 2019.

[28] E. Baccelli *et al.*, "RIOT OS: Towards an OS for the Internet of Things," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, pp. 79-80, April 2013.

[29] B. Saadallah *et al.*, "CCNx for contiki: Implementation details," Technical report, no. 432, Nov. 2012.

[30] W. Shang *et al.*, "The design and implementation of the NDN protocol stack for RIOT-OS," in *Proc. IEEE Globecom Workshops GC Wkshps*, pp. 1-6, December 2016.

[31] O. J. A. Contreras, "Performance Evaluation of an IEEE 802.11 Mobile AdHoc Network on the Raspberry Pi," October 2015.

Παράρτημα

Διαδικασία διεξαγωγής των πειραμάτων

Κόμβος Καταναλωτής και Παραγωγός

Αρχικά αλλάζουμε το αρχείο /etc/network/interfaces σε:

```
auto wlan0
iface wlan0 inet static
address 10.10.20.4
netmask 255.255.255.0
```

Κατόπιν εκτελούμε το bash script my_startwlan_adhoc.sh το οποίο είναι:

```
#!/bin/sh

/sbin/ifdown wlan0

sleep 1

iw dev wlan0 set type ibss

/sbin/ifup wlan0

sleep 1

#iw dev wlan0 ibss join adhoc1 2412 02:ca:ff:ee:ba:be

iw dev wlan0 ibss join adhoc2 2437 02:ca:ff:ee:ba:ff # channel 6 ->2437
```

Με το παραπάνω bash script απενεργοποιούμε και στη συνέχεια ενεργοποιούμε το wlan0 interface (δηλαδή το interface του WiFi). Με την τελευταία γραμμή ρυθμίζουμε το Raspberry έτσι ώστε να δημιουργήσει ένα Ad-hoc δίκτυο με όνομα adhoc2 στο κανάλι 6. Στον άλλο κόμβο θα πρέπει να τρέξουμε την προτελευταία σχολιασμένη εντολή ώστε το Raspberry pi να δημιουργήσει ένα διαφορετικό Ad-hoc δίκτυο με όνομα adhoc1 στο κανάλι 1. Έπειτα προκειμένου να τρέξει ο δαίμονας του IBR-DTN με συγκεκριμένο configuration εκτελούμε την εντολή:

```
sudo dtnd -v -c my_config.conf
```

Το configuration αρχείο που χρησιμοποιήθηκε είναι το εξής:

```
local_uri = dtn://consumer # or dtn://producer
storage_path = /home/pi/Desktop/bundles/

net_interfaces = lan0
net_lan0_type = tcp
net_lan0_interface = wlan0
net_lan0_port = 4556

routing = flooding
routing_forwarding = yes
routing_prefer_direct = yes

time_reference = no
time_synchronize = yes
time_discovery_announcements = yes
time_set_clock = yes
```

Για τα πειράματα με την DTN αρχιτεκτονική, ο κόμβος του καταναλωτή εκτελεί το παρακάτω bash script:

```
#!/bin/bash

filename='names.txt'

selected_names='selected_names.txt'

n=6 #n = how many lines to get from .txt file

shuf -n $n $filename > $selected_names


temp=$(shuf -n 1 $selected_names)

echo $temp | dtnsend dtn://producer/sensors
```

Με το παραπάνω bash script ο καταναλωτής διαλέγει ορισμένα ονόματα μετρήσεων (με τα οποία θα τρέξει το πείραμα) και κατόπιν στέλνει ένα bundle με ένα από τα επιλεγμένα ονόματα μετρήσεων (ζητώντας έτσι την αντίστοιχη μέτρηση από τον παραγωγό). Τέλος εκτελείτε η εντολή:

```
dtntrigger dtnConsumer ./dtn_consumer.sh
```

Έτσι ώστε μόλις ο καταναλωτής λάβει ένα bundle με το πεδίο dtnConsumer να εκτελέσει το bash script dtn_consumer.sh με το οποίο διαλέγει τυχαία ένα όνομα από τα επιλεγμένα και στέλνει ένα καινούριο bundle με αυτό.

```
#!/bin/bash

selected_names='selected_names.txt'

temp=$(shuf -n 1 $selected_names)

echo $temp | dtnsend dtn://producer/sensors
```

Για τα πειράματα με την DTN αρχιτεκτονική, ο κόμβος του παραγωγός εκτελεί τις παρακάτω εντολές με τις οποίες ρυθμίζει και ενεργοποιεί τα interfaces wpan0 και lowpan0, δημιουργεί ένα UDP server που ακούει για εισερχόμενες συνδέσεις στη θύρα 8808 και στέλνει τα λαμβανόμενα πακέτα σε ένα txt αρχείο, οργανώνει τις μετρήσεις ανάλογα με το είδος τους σε διαφορετικά txt αρχεία, ενεργοποιεί το Ad hoc δίκτυο του και ενεργοποιεί το δαίμονα του IBR-DTN.

```
sudo iwpan phy phy0 set tx_power 80

ip link set wpan0 down
iwpan phy phy0 set channel 0 26
iwpan dev wpan0 set pan_id 35
iwpan dev wpan0 set short_addr 69
iwpan phy phy0 set tx_power 50
iwpan phy phy0 set tx_power 80
iwpan phy phy0 set tx_power 0 #0dbm
ip link set wpan0 up

## Shutdown any already active links
ip link set lowpan0 down
ip link set wpan0 down

## WPAN configuration section
iwpan phy phy0 set channel 0 26

#pan_id is a 2bytes  number (0-65536) or (0-FFFF)
#pan_id is the respective network name of an 802.11 network
#all nodes participating in a the WSN need to have the same pan_id
iwpan dev wpan0 set pan_id 35

#short_addr is the actual address of each 802.15.4 node
#short_addr is also a 2bytes number (0-65536)
iwpan dev wpan0 set short_addr 69
```

```
#Configure transmission power in dBm
iwpan phy phy0 set tx_power 0

## LOWPAN configuration section
ip link add link wpan0 name lowpan0 type lowpan

#IPv6 address of the lowpan interface
ip addr add 2001:db8::100/64 dev lowpan0

ip link set wpan0 up
ip link set lowpan0 up

nc -6ul 8808 -k |& tee -a  Desktop/NDN-DTN/ndn-cxx_umobile/build/examples/out.txt
./tail_nc10.sh &
sudo ./my_startwlan_adhoc
sudo dtnd -v -c test_3.conf
dtntrigger sensors ./dtn_producer2.sh
```

To bash script tail_nc10.sh είναι το παρακάτω:

```
#!/bin/bash
filename='Desktop/NDN-DTN/ndn-cxx_umobile/build/examples/out.txt'
temp='temp.txt'
temp1='temp1.txt'
temp2='temp2.txt'
#…  measurements
hum='hum.txt'
hum1='hum1.txt'
#… measurements
while true; do
 after=`tail -n 1 $filename`;
 if [ ! -z "$after" ]
 then
        if [ ${after:0:4} == "temp" ]        #Temperature measurements
        then
                if [ ${after:0:5} == "temp1" ]
                then
                        echo "$after" > $temp1;
                        echo "$after" > $temp;
                elif [ ${after:0:5} == "temp2" ]
                then
                        echo "$after" > $temp2;
                        echo "$after" > $temp;
                else
                        echo "$after" > $temp;
                fi
        elif [ ${after:0:3} == "hum" ]       #Humidity measurements
        then
                if [ ${after:0:4} == "hum1" ]
```

```
then
                        echo "$after" > $hum1;
                        echo "$after" > $hum;
                else
                        echo "$after" > $hum;
                fi
        else
         echo "Unknown";
        fi
 echo "" >> $filename;
 fi
done
```

To bash script dtn_producer2.sh είναι το παρακάτω:

```
#!/bin/bash
temp='temp.txt'
temp1='temp1.txt'
temp2='temp2.txt'
…
hum='hum.txt'
hum1='hum1.txt'
…
MAXPREVIEW=250
src=$1
toprint=$(head -c $MAXPREVIEW $payload  | strings -n 1 -e S )
echo $toprint
```

```
#if producer receives bundle "temp"
#then producer sends the last temperature measurement (temp.txt)
if [ ${toprint:0:4} == "temp" ]
        then
        if [ ${toprint:0:5} == "temp1" ] #if producer receives bundle temp1
                then #the he sends the last temperature measurement from sensor1
                after=`tail -n 1 $temp1`
        elif [ ${toprint:0:5} == "temp2" ]
                then
                after=`tail -n 1 $temp2`
        …
        else
                after=`tail -n 1 $temp`
        fi
        echo "$after" | dtnsend dtn://consumer/dtnConsumer
elif [ ${toprint:0:3} == "hum" ]
        then

if [ ${toprint:0:4} == "hum1" ]
                then
                after=`tail -n 1 $hum1`
        else
                after=`tail -n 1 $hum`
        fi
        echo "$after" | dtnsend dtn://consumer/dtnConsumer
        …
else
        echo "Unknown";
fi
```

Για τα πειράματα με την UMOBILE αρχιτεκτονική, ο κόμβος καταναλωτής εκτελεί τις παρακάτω εντολές με τις οποίες δημιουργεί ένα ad hoc δίκτυο, ενεργοποιεί τον δαίμονα του IBR-DTN, ενεργοποιεί τον δαίμονα NFD, κάνει τις απαραίτητες εγγραφές και εκτελεί την εφαρμογή του consumer:

```
sudo ./my_startwlan_adhoc

sudo dtnd -v -c my_conf.conf

nfd-start

nfdc register /example/temp dtn://producer/nfd

nfdc register /example/temp1 dtn://producer/nfd

nfdc register /example/temp2 dtn://producer/nfd

nfdc register /example/hum dtn://producer/nfd

nfdc register /example/hum1 dtn://producer/nfd

…

sudo ./Desktop/NDN-DTN/ndn-cxx_umobile/build/examples/my_consumer11
```

Ο κώδικας my_consumer11.cpp είναι ο παρακάτω:

```
#include "face.hpp"
#include <util/time.hpp>
#include <util/random.hpp>
ndn::time::system_clock::time_point sentTime;
int total_interests=0;
int satisfied_interests=0;
double total_RTT=0;
bool retry = 1;
int lastsend;
char uri[105][20] = {                    "/example/temp",
                                         "/example/hum",
                                         "/example/sound",
                                         "/example/light",
                                         "/example/movement",
                                         "/example/hall",
                                        "/example/temp1",
                                         "/example/hum1",
                                    …
                              };
namespace examples {
class Consumer : noncopyable
{
public:
 void
 run()
 {
        int i;
        int n=15; // number of different name prefixes
```

```
        //The rand() % 100 will give you a random number between 0 and 100, and the
        //probability of it being under 75 is 75%.
        i=(rand() % n);
        std::cout << "n= " << n << std::endl;
        std::cout << "Random number is: " << i << std::endl;
        lastsend=i;

        Interest interest(uri[i]);
        interest.setInterestLifetime(time::milliseconds(190000));
        interest.setMustBeFresh(true);
        m_face.expressInterest(interest,
                bind(&Consumer::onData, this,  _1, _2),
                bind(&Consumer::my_onTimeout, this, _1));
        std::cout << "Sending " << interest << std::endl;
        total_interests++;
        std::cout << "\t Total interests :" << total_interests <<"\n\n"<< std::endl;
        //Calculate starting time
        sentTime=time::system_clock::now();
         // processEvents will block until the requested data received or timeout occurs
         m_face.processEvents();
 }
void
 my_onTimeout(const Interest& interest)
 {
        std::cout << "Timeout " << interest << std::endl;
        std::cout << "Retry lastsend= " << lastsend << std::endl;
        my_retry(lastsend);
 }
```

```
void
 my_retry(int last)
 {
      //Retry
      Interest interest_retry(uri[last]);
      interest_retry.setInterestLifetime(time::milliseconds(190000));
      interest_retry.setMustBeFresh(true);
      my_face.expressInterest(interest_retry,
                  bind(&Consumer::onData, this,  _1, _2),
                  bind(&Consumer::my_onTimeout, this, _1));
      std::cout << "Sending " << interest_retry << std::endl;
      total_interests++;
      std::cout << "\t Total interests :" << total_interests <<"\n\n"<< std::endl;
      //Calculate starting time
      sentTime=time::system_clock::now();
      // processEvents will block until the requested data received or timeout occurs
      my_face.processEvents();
 }
private:
 void
 onData(const Interest& interest, const Data& data)
 {
          std::cout << data << std::endl;
          double roundTripTime;
          satisfied_interests++;
          std::cout << "From interest: " << interest << std::endl;
          roundTripTime = (time::system_clock::now() -  sentTime).count() / 1000000.0;
          std::cout <<"\t RTT:" <<roundTripTime << "\n\n\n\n"<< std::endl;
          total_RTT=total_RTT + roundTripTime;
```

```
retry=0;
  }
private:
  Face m_face;
  Face my_face;
};
} // namespace examples
} // namespace ndn

int
main(int argc, char** argv)
{
  ndn::examples::Consumer consumer;
 const int rep= 20; //number of repetitions
 for (int i=0; i<rep; i++){
                        try {
                                consumer.run();
                        }
                        catch (const std::exception& e) {
                                std::cerr << "ERROR: " << e.what() << std::endl;
                        }
            }
std::cout <<"\t Total RTT:" <<total_RTT/satisfied_interests << std::endl;
std::cout <<"\t Success Ratio:" <<(double)satisfied_interests/total_interests << std::endl;
std::cout <<"\t Satisfied interests:" <<satisfied_interests << std::endl;

  return 0;
}
```

Για τα πειράματα με την UMOBILE αρχιτεκτονική, ο κόμβος παραγωγός εκτελεί τις παρακάτω εντολές με τις οποίες δημιουργεί ένα ad hoc δίκτυο, ενεργοποιεί τον δαίμονα του IBR-DTN, ενεργοποιεί τον δαίμονα NFD, κάνει τις απαραίτητες εγγραφές και εκτελεί την εφαρμογή του producer:

```
sudo ./my_startwlan_adhoc

sudo dtnd -v -c test_3.conf

#Change /urs/local/etc/ndn/nfd.conf

nfd-start

#nfdc register /example/temp dtn://pi2/nfd

#nfdc register /example/hum dtn://pi2/nfd

./nfdc_register_producer.sh

sudo ./Desktop/NDN-DTN/ndn-cxx_umobile/build/examples/my_producer6
```

Ο κώδικας της εφαρμογής του producer (my_producer6.cpp) είναι ο παρακάτω:

```
#include "face.hpp"
#include "security/key-chain.hpp"
#include <iostream>
#include <fstream>
#include <string>

namespace ndn {
namespace examples {

class Producer : noncopyable
{
public:
  void
  run()
  {
    m_face.setInterestFilter("/example",
                             bind(&Producer::onInterest, this, _1, _2),
                             RegisterPrefixSuccessCallback(),
                             bind(&Producer::onRegisterFailed, this, _1, _2));
    m_face.processEvents();
  }
private:
  void
  onInterest(const InterestFilter& filter, const Interest& interest)
  {
    std::cout << "<< I: " << interest << std::endl;
    // Create new name, based on Interest's name
    Name dataName(interest.getName());
    dataName
```

```
    .append("fresh") // add "testApp" component to Interest name
    .appendVersion();  // add "version" component (current UNIX timestamp in milliseconds)
std::string interestName=  (interest.getName()).toUri();
std::string sensor= interestName.substr(9,10); //starting at position 9 with 8 chars
std::string line;

if (sensor.substr(0,4) == "temp"){
        if(sensor.substr(0,5) == "temp1"){
                //Reading .txt file
                 std::ifstream myfile ("temp1.txt");

                 if (myfile.is_open())
                 {
                         while ( getline (myfile,line) )
                         {
                                 if(line.at(0) =='t') break;
                         }
                         myfile.close();
                 }
                 else std::cout << "Unable to open file\n";
        }
        else if(sensor.substr(0,5) == "temp2"){
                //Reading .txt file
                 std::ifstream myfile ("temp2.txt");

                 if (myfile.is_open())
                 {
                         while ( getline (myfile,line) )
                         {
```

```
                        if(line.at(0) =='t') break;
                        }
                        myfile.close();
                    }
                    else std::cout << "Unable to open file\n";
            }
            else if(sensor.substr(0,5) == "temp3"){
                    …
}
std::string content = line.substr(0,line.size());
std::cout << "Content :"<<content << std::endl;
   // Create Data packet
   shared_ptr<Data> data = make_shared<Data>();
   data->setName(dataName);
   data->setFreshnessPeriod(time::seconds(10));
   data->setContent(reinterpret_cast<const uint8_t*>(content.c_str()), content.size());
   // Sign Data packet with default identity
   m_keyChain.sign(*data);
  // Return Data packet to the requester
   std::cout << ">> D: " << *data << std::endl;
   m_face.put(*data);
  }
  void
  onRegisterFailed(const Name& prefix, const std::string& reason)
  {
   std::cerr << "ERROR: Failed to register prefix \""
          << prefix << "\" in local hub's daemon (" << reason << ")"
          << std::endl;
   m_face.shutdown();
```

```
  }
private:
  Face m_face;
  KeyChain m_keyChain;
};
}
}

int
main(int argc, char** argv)
{
  ndn::examples::Producer producer;
  try {
    producer.run();
  }
  catch (const std::exception& e) {
    std::cerr << "ERROR: " << e.what() << std::endl;
  }
  return 0;
}
```

Κινητός κόμβος

Με το παρακάτω bash script ο κινητός κόμβος συνδέεται στο δίκτυο ad hoc 1 έπειτα απενεργοποιεί το WiFi interface του και στη συνέχεια συνδέεται στο δίκτυο ad hoc 2. Η παραπάνω διαδικασία επαναλαμβάνεται διαρκώς.

```
#! /bin/bash
/sbin/ifdown wlan0
iw dev wlan0 set type ibss
ip -4 addr flush dev wlan0
ip -6 addr flush dev wlan0
sleep 10

while true; do

# State 1: Connected to ad hoc2
ip a add 10.10.21.3/255.255.255.0 dev wlan0
ip link set wlan0 up
iw dev wlan0 ibss join adhoc2 2437 02:ca:ff:ee:ba:ff #channel 6-->2437
printf "Connected to adhoc2 \n\n"
ifconfig
sleep 40

ip link set wlan0 down
ip -4 addr flush dev wlan0
printf "Disconnected from adhoc2 \n\n"


# State 2: Not connected
ifconfig
sleep 40

# State 3: Connected to ad hoc 1
ip a add 10.10.20.3/255.255.255.0 dev wlan0
ip link set wlan0 up
iw dev wlan0 ibss join adhoc1 2412 02:ca:ff:ee:ba:be
```

```
printf "Connected to ad hoc 1! \n\n"
ifconfig
sleep 40

ip link set wlan0 down
ip -4 addr flush dev wlan0
printf "Disconnected from senskin \n\n"

ifconfig
sleep 40

done
```

Κόμβος Αισθητήρων

Με τον παρακάτω κώδικα ο κόμβος αισθητήρων παράγει και αποστέλλει μετρήσεις στον κόμβο συλλέκτη.

```
#include <stdio.h>
#include "board.h"

#include <inttypes.h>
#include "net/gnrc.h"
#include "net/gnrc/ipv6.h"
#include "net/gnrc/netif.h"
#include "net/gnrc/netif/hdr.h"
#include "net/gnrc/udp.h"
#include "net/gnrc/pktdump.h"
#include "timex.h"
#include "utlist.h"
#include "xtimer.h"

//ADC
#include "timex.h"
#include "periph/adc.h"
#define RES             ADC_RES_12BIT
#define DELAY           (10000LU * US_PER_MS) /* 10000 ms */

//PIR
#define PIR_GPIO                GPIO_PIN(PA, 13)
#include "pir.h"

static void send(char *addr_str, char *port_str, char *data, unsigned int num,unsigned int delay)
                        {
                                int iface;
                                uint16_t port;
```

```
        ipv6_addr_t addr;

        /* get interface, if available */
        iface = ipv6_addr_split_iface(addr_str);
        if ((iface < 0) && (gnrc_netif_numof() == 1)) {
                iface = gnrc_netif_iter(NULL)->pid;
        }
        /* parse destination address */
        if (ipv6_addr_from_str(&addr, addr_str) == NULL) {
                puts("Error: unable to parse destination address");
                return;
        }
        /* parse port */
        port = atoi(port_str);
        if (port == 0) {
                puts("Error: unable to parse destination port");
                return;
        }

        for (unsigned int i = 0; i < num; i++) {
                gnrc_pktsnip_t *payload, *udp, *ip;
                unsigned payload_size;
                /* allocate payload */
payload = gnrc_pktbuf_add(NULL, data, strlen(data), GNRC_NETTYPE_UNDEF);
                if (payload == NULL) {
                puts("Error: unable to copy data to packet buffer");
                        return;
                }
                /* store size for output */
```

```
payload_size = (unsigned)payload->size;
                                /* allocate UDP header, set source port := destination port */
                                        udp = gnrc_udp_hdr_build(payload, port, port);
                                        if (udp == NULL) {
                                                puts("Error: unable to allocate UDP header");
                                                gnrc_pktbuf_release(payload);
                                                return;
                                        }
                                        /* allocate IPv6 header */
                                        ip = gnrc_ipv6_hdr_build(udp, NULL, &addr);
                                        if (ip == NULL) {
                                                puts("Error: unable to allocate IPv6 header");
                                                gnrc_pktbuf_release(udp);
                                                return;
                                        }
                                        /* add netif header, if interface was given */
                                        if (iface > 0) {
                        gnrc_pktsnip_t *netif = gnrc_netif_hdr_build(NULL, 0, NULL, 0);
                        ((gnrc_netif_hdr_t *)netif->data)->if_pid = (kernel_pid_t)iface;
                                                LL_PREPEND(ip, netif);
                                        }
                                        /* send packet */
                                if(!gnrc_netapi_dispatch_send(GNRC_NETTYPE_UDP,
GNRC_NETREG_DEMUX_CTX_ALL, ip)) {
                                                puts("Error: unable to locate UDP thread");
                                                gnrc_pktbuf_release(ip);
                                                return;
```

```
                                        }
/* access to `payload` was implicitly given up with the send operation above
                                                * => use temporary variable for output */
printf("Success: sent %u byte(s) to [%s]:%u\n", payload_size, addr_str,  port);
                                                xtimer_usleep(delay);
                                        }
                                }


int main(void)
{
        uint32_t num = 1;
        uint32_t delay = 1000000;
        char addr[1][26]={"fe80::e830:9f2a:a5f7:6fcd"};
        char *a=&addr[0][0];
        char port[1][5]={"8808"};
        char *p=&port[0][0];
        char content[10][3]={"23","22", "20","18","19","18","20","18","19"};
        char *s =&content[0][0];
        int sizecont=sizeof(content[0]);
        int count=0;
        int rows=sizeof(content)/sizeof(content[0]);
        int hum=35;
    xtimer_ticks32_t last = xtimer_now();
    int sample = 0;
    char c_sample[13] ;
    /* initialize all available ADC lines */
    for (unsigned i = 0; i < ADC_NUMOF; i++) {
```

```
        if (adc_init(ADC_LINE(i)) < 0) {
            printf("Initialization of ADC_LINE(%u) failed\n", i);
            return 1;
        } else {
            printf("Successfully initialized ADC_LINE(%u)\n", i);
        }
    }
            xtimer_usleep(1000 * 1000);
//PIR
            static pir_t dev;
    puts("PIR motion sensor test application\n");
    printf("Initializing PIR sensor at GPIO_%ld... ", (long)PIR_GPIO);
    if (pir_init(&dev, PIR_GPIO) == 0) {
        puts("[OK]\n");
    }
    else {
        puts("[Failed]");
        return 1;
    }
            //Hall sensor
            gpio_init(GPIO_PIN(PA, 23), GPIO_IN);

    while (1) {
                        //TEMP
                        sprintf(c_sample,"temp1=%s", s);
                        printf("\n %s \n", c_sample);
                        send(a, p, c_sample, num, delay);
                        xtimer_periodic_wakeup(&last, DELAY);

                        for(int i=0; i<sizecont;i++){
```

```
                s++;
                }
        count++;
        if(count==rows) {
                s =&content[0][0];
                count=0;
        }

        //HUM
        sprintf(c_sample,"hum1=%d", hum);
        printf("\n %s \n", c_sample);
        send(a, p, c_sample, num, delay);
        xtimer_periodic_wakeup(&last, DELAY);
        hum ++;
        if(hum==50) hum=35;

        //ADC
for (unsigned i = 0; i < ADC_NUMOF; i++) {
   sample = adc_sample(ADC_LINE(i), RES);
   if (sample < 0) {
      printf("ADC_LINE(%u): selected resolution not applicable\n", i);
   } else {
                        if (i==0){ //Pin PA06
                                sprintf(c_sample,"light1=%d", sample);
                // printf("Light: ADC_LINE(%u): %i\n", i, sample);
                                printf("\n %s \n", c_sample);
                        }
                        else{ //Pin PA07
                                sprintf(c_sample,"sound1=%d", sample);
```

```
                                        //          printf("Sound: ADC_LINE(%u): %i\n", i,
sample);
                                        printf("\n %s \n", c_sample);
                                    }
                                    send(a, p, c_sample, num, delay);
                            xtimer_periodic_wakeup(&last, DELAY);

            }
        }

            printf("PIR Status: %s\n", pir_get_status(&dev) == PIR_STATUS_LO ? "lo" : "hi");
sprintf(c_sample,"movement1=%s", pir_get_status(&dev) == PIR_STATUS_LO ? "lo" : "hi");
                    printf("\n %s \n", c_sample);
                    send(a, p, c_sample , num, delay);
                    xtimer_periodic_wakeup(&last, DELAY);

                    //Hall sensor
sprintf(c_sample,"hall1=%d",gpio_read(GPIO_PIN(PA, 23)) ); //Hall=0 (magnet is near)
                    printf("\n %s \n", c_sample);      //Hall==1 (no magnet)
                    send(a, p, c_sample , num, delay);

        xtimer_periodic_wakeup(&last, DELAY);
    }
            /* should be never reached */
    return 0;
}
```

Μετρήσεις πειράματος 1.1

Χρόνος παραμονής= 20 sec			
RTT(sec)	Average RTT(sec)	Round Trips	Average Round Trips
92.236	129.88	1	1.6
156.682		2	
95.054		1	
78.664		1	
150.672		2	
162.731		2	
173.682		2	
156.23		2	
148.629		2	
84.214		1	

Χρόνος παραμονής= 30 sec			
RTT(sec)	Average RTT(sec)	Round Trips	Average Round Trips
119.867	170.3969	1	1.4
119.863		1	
125.103		1	
247.116		2	
118.77		1	
141.796		1	
351.69		3	
119.511		1	
239.931		2	
120.322		1	

Χρόνος παραμονής= 40 sec			
RTT(sec)	Average RTT(sec)	Round Trips	Average Round Trips
167.303	177.5681	1	1.1
154.907		1	
154.773		1	
326.178		2	
162.869		1	
155.142		1	
155.005		1	
169.873		1	
169.743		1	
159.888		1	

Χρόνος παραμονής= 50 sec			
RTT(sec)	Average RTT(sec)	Round Trips	Average Round Trips
201.261	204.9434	1	1
205.269		1	
203.987		1	
201.078		1	
229.065		1	
204.053		1	
200.086		1	
201.283		1	
199.302		1	
204.05		1	

Χρόνος παραμονής= 60 sec			
RTT(sec)	Average RTT(sec)	Round Trips	Average Round Trips
237.054	241.073	1	1
240.092		1	
245.097		1	
239.994		1	
241.121		1	
243.05		1	
252.192		1	
227.998		1	
245.032		1	
239.1		1	

Μετρήσεις πειράματος 1.2

Χρόνος παραμονής= 20 sec			
RTT(sec)	Average RTT(sec)	Round Trips	Average Round Trips
166.386	226.772	2	2.777
78.881		1	
245.364		3	
80.545		1	
248.744		3	
321.981		4	
574.984		7	

173.883	2
150.182	2
X	

Χρόνος παραμονής= 30 sec			
RTT(sec)	**Average RTT(sec)**	**Round Trips**	**Average Round Trips**
137.776	123.153	1	1
125.116		1	
114.678		1	
122.598		1	
128.001		1	
116.436		1	
123.561		1	
120		1	
124.42		1	
118.936		1	

Χρόνος παραμονής= 40 sec			
RTT(sec)	**Average RTT(sec)**	**Round Trips**	**Average Round Trips**
179.919	162.907	1	1
153.086		1	
171.97		1	
158.309		1	
155.116		1	
160.263		1	
165.022		1	
165.164		1	
160.068		1	
160.15		1	

Χρόνος παραμονής= 50 sec			
RTT(sec)	**Average RTT(sec)**	**Round Trips**	**Average Round Trips**
208.824	202.929	1	1
204.17		1	
200.248		1	
200.099		1	
205.147		1	
200.176		1	
200.259		1	
209.089		1	

195.172	1
206.107	1

Χρόνος παραμονής= 60 sec			
RTT(sec)	Average RTT(sec)	Round Trips	Average Round Trips
245.064	266.492	1	1.1
251.53		1	
232.844		1	
240.353		1	
248.012		1	
476.428		2	
245.966		1	
240.352		1	
249.964		1	
234.411		1	

Μετρήσεις πειράματος 1.3

Χρόνος παραμονής= 20 sec			
RTT(sec)	Average RTT(sec)	Round Trips	Average Round Trips
264.862	108.938	4	1.4
331.179		4	
162.682		2	
1.19599		0	
76.1335		1	
1.22099		0	
1.50177		0	
248.231		3	
1.22698		0	
1.15583		0	

Χρόνος παραμονής= 30 sec			
RTT(sec)	Average RTT(sec)	Round Trips	Average Round Trips
267.976	87.283	2	0.7
115.868		1	
120.003		1	
1.0837		0	
120.015		1	
1.00729		0	
0.865625		0	

244.105	2
1.06448	0
0.844115	0

Χρόνος παραμονής= 40 sec			
RTT(sec)	**Average RTT(sec)**	**Round Trips**	**Average Round Trips**
333.674	96.093	2	0.6
163.093		1	
165.128		1	
1.1324		0	
150.224		1	
1.72411		0	
1.34729		0	
141.335		1	
1.81219		0	
1.46391		0	

Χρόνος παραμονής= 50 sec			
RTT(sec)	**Average RTT(sec)**	**Round Trips**	**Average Round Trips**
418.64	122.841	2	0.6
200.121		1	
200.146		1	
1.07448		0	
200.157		1	
1.0301		0	
0.862187		0	
204.4203		1	
1.10922		0	
0.854584		0	

Χρόνος παραμονής= 60 sec			
RTT(sec)	**Average RTT(sec)**	**Round Trips**	**Average Round Trips**
240.342	145.745	1	0.6
242.22		1	
240.217		1	
1.27729		0	
490.293		2	
1.22974		0	
0.918854		0	
238.211		1	

1.45917	0
1.28469	0

Μετρήσεις πειράματος 2.1 (DTN)

Αριθμός διαφορετικών ονομάτων= 5	
RTT(sec)	**Average RTT(sec)**
320.115	193.5367
165.038	
164.923	
325.1	
163.212	
167.893	
156.385	
152.761	
158.963	
160.977	

Αριθμός διαφορετικών ονομάτων= 10	
RTT(sec)	**Average RTT(sec)**
165.041	194.5814
165.042	
160.039	
165.328	
163.33	
320.172	
164.971	
319.894	
160.53	
161.467	

Αριθμός διαφορετικών ονομάτων= 15	
RTT(sec)	**Average RTT(sec)**
158.971	176.6992
159.984	
167.722	
312.413	
164.201	
169.91	
149.979	
159.991	
163.938	
159.883	

Αριθμός διαφορετικών ονομάτων= 20	
RTT(sec)	**Average RTT(sec)**
163.089	174.8349
160.161	
160.845	
160.77	
165.242	
160.039	
320.12	
164.992	
133.176	
159.915	

Αριθμός διαφορετικών ονομάτων= 25	
RTT(sec)	**Average RTT(sec)**
165.554	193.5722
163.09	
320.632	
159.522	
166.745	
313.344	
163.32	

Αριθμός διαφορετικών ονομάτων= 30	
RTT(sec)	**Average RTT(sec)**
159.103	194.6949
324.98	
158.966	
165.008	
164.11	
178.69	
301.419	

159.548
165.091
158.876

164.878
164.753
165.042

Αριθμός διαφορετικών ονομάτων= 40	
RTT(sec)	**Average RTT(sec)**
164.888	179.1899
166.875	
164.887	
165.748	
165.553	
164.833	
154.024	
173.278	
316.829	
154.984	

Μετρήσεις πειράματος 2.2 (UMOBILE)

Αριθμός διαφορετικών ονομάτων= 5			
RTT(sec)	**Average RTT(sec)**	**Cache hit**	**Cache hit ratio**
175.616	82.404664	0	0.5
168.092		0	
150.174		0	
172.017		0	
0.0014		1	
0.001		1	
0.0009		1	
0.00114		1	
158.142		0	
0.0012		1	

Αριθμός διαφορετικών ονομάτων= 10			
RTT(sec)	**Average RTT(sec)**	**Cache hit**	**Cache hit ratio**
172.98	117.76605	0	0.3
174.637		0	
176.33		0	
163.238		0	
0.0012		1	
0.001		1	

0.0013	1
163.997	0
159.092	0
167.383	0

Αριθμός διαφορετικών ονομάτων= 15			
RTT(sec)	**Average RTT(sec)**	**Cache hit**	**Cache hit ratio**
172.685	98.135355	0	0.4
167.658		0	
157.604		0	
159.155		0	
160.116		0	
0.0015		1	
0.0021		1	
0.0024		1	
164.128		0	
0.00155		1	

Αριθμός διαφορετικών ονομάτων= 20			
RTT(sec)	**Average RTT(sec)**	**Cache hit**	**Cache hit ratio**
179.164	146.87373	0	0.2
160.138		0	
159.118		0	
325.336		0	
164.622		0	
0.0012		1	
0.0011		1	
159.108		0	
161.072		0	
160.177		0	

Αριθμός διαφορετικών ονομάτων= 25			
RTT(sec)	**Average RTT(sec)**	**Cache hit**	**Cache hit ratio**
176.955	145.1486	0	0.1
158.229		0	
160.236		0	
160.023		0	
164.137		0	
160.125		0	
0.001		1	

160.162	0
165.11	0
146.508	0

Αριθμός διαφορετικών ονομάτων= 30			
RTT(sec)	Average RTT(sec)	Cache hit	Cache hit ratio
334.195	162.92223	0	0.2
325.25		0	
159.232		0	
165.06		0	
165.129		0	
0.0013		1	
0.001		1	
155.149		0	
160.098		0	
165.107		0	

Αριθμός διαφορετικών ονομάτων= 40			
RTT(sec)	Average RTT(sec)	Cache hit	Cache hit ratio
170.815	162.9914	0	0
164.111		0	
165.126		0	
155.12		0	
170.132		0	
160.121		0	
155.097		0	
165.193		0	
160.075		0	
164.124		0	

Μετρήσεις πειράματος 3

lifetime= 150				
RTT(sec)	Συνολική διάρκεια λήψης 10 πακέτων μετρήσεων (sec)	Total Interests	Satisfied Interests	Success Ratio
Timeout	972.86496	16	10	0.625
12.5451				
Timeout				
17.9479				
Timeout				

10.1671
Timeout
10.0757
Timeout
11.7426
0.0016
0.0013
0.0016
Timeout
10.3787
0.00336

lifetime= 160				
RTT(sec)	**Συνολική διάρκεια λήψης 10 πακέτων μετρήσεων (sec)**	**Total Interests**	**Satisfied Interests**	**Success Ratio**
Timeout	978.92205	16	10	0.625
13.2865				
Timeout				
0.099				
Timeout				
0.111527				
Timeout				
2.398				
Timeout				
2.935				
0.00122				
0.00092				
0.00096				
Timeout				
0.08769				
0.001233				

lifetime= 170				
RTT(sec)	**Συνολική διάρκεια λήψης 10 πακέτων μετρήσεων (sec)**	**Total Interests**	**Satisfied Interests**	**Success Ratio**
Timeout	1136.17053	12	10	0.8333
160.607				
160.063				
160.125				
Timeout				

1.9333
153.332
0.0012
0.00098
0.00885
160.098
0.0012

lifetime= 180				
RTT(sec)	Συνολική διάρκεια λήψης 10 πακέτων μετρήσεων (sec)	Total Interests	Satisfied Interests	Success Ratio
Timeout	1133.092	11	10	0.909
142.47				
165.127				
165.12				
160.254				
167.244				
0.003				
0.0011				
0.0009				
152.869				
0.003				

lifetime= 190				
RTT(sec)	Συνολική διάρκεια λήψης 10 πακέτων μετρήσεων (sec)	Total Interests	Satisfied Interests	Success Ratio
166.832	972.4532	10	10	1
160.152				
156.05				
159.346				
164.957				
0.0012				
0.001				
0.0009				
165.112				
0.0011				

lifetime= 200				
RTT(sec)	Συνολική διάρκεια λήψης 10 πακέτων μετρήσεων (sec)	Total Interests	Satisfied Interests	Success Ratio
Timeout	1140.641	11	10	0.909

123.969
164.499
159.095
165.138
190.479
0.0013
0.0011
0.0008
137.457
0.0012

lifetime= 210				
RTT(sec)	Συνολική διάρκεια λήψης 10 πακέτων μετρήσεων (sec)	Total Interests	Satisfied Interests	Success Ratio
168.178	980.6037	10	10	1
159.868				
165.133				
160.156				
160.113				
0.0012				
0.0011				
0.0014				
167.15				
0.002				

Μετρήσεις πειράματος 4

freshness period= 700						
RTT(sec)	Συνολική διάρκεια λήψης 10 πακέτων μετρήσεων (sec)	Total Interests	Satisfied Interests	Success Ratio	Cache hits	Cache hit ratio
157.971	2944.825	22	20	0.9091	4	0.182
160.039						
Timeout						
129.21						
162.151						
163.061						
0.00105						
0.00077						
0.00072						
165.13						
157.154						

163.108
167.546
152.994
Timeout
161.56
159.13
167.537
172.166
158.832
0.0001
167.233

freshness period= 1300						
RTT(sec)	Συνολική διάρκεια λήψης 10 πακέτων μετρήσεων (sec)	Total Interests	Satisfied Interests	Success Ratio	Cache hits	Cache hit ratio
158.394	2596.04	23	20	0.869565	6	0.261
Timeout						
131.086						
161.342						
Timeout						
138.265						
165.103						
0.0017						
0.0016						
0.0015						
159.132						
0.00193						
160.126						
160.104						
165.147						
Timeout						
135.281						
0.00114						
167.957						
152.268						
171.823						
0.00142						
0.001						

freshness period= 1900

RTT(sec)	Συνολική διάρκεια λήψης 10 πακέτων μετρήσεων (sec)	Total Interests	Satisfied Interests	Success Ratio	Cache hits	Cache hit ratio
163.177	2584.21	23	20	0.869565	7	0.304
Timeout						
128.361						
162.015						
165.122						
165.113						
0.00114						
0.00098						
0.00087						
160.126						
0.00221						
155.223						
165.035						
Timeout						
134.249						
160.13						
0.0013						
0.0011						
Timeout						
134.26						
161.131						
0.0011						
160.263						

freshness period= 2500						
RTT(sec)	Συνολική διάρκεια λήψης 10 πακέτων μετρήσεων (sec)	Total Interests	Satisfied Interests	Success Ratio	Cache hits	Cache hit ratio
Timeout	2280.97	22	20	0.9091	8	0.364
137.874						
160.108						
165.134						
Timeout						
133.893						
160.502						
0.0028						
0.002						
0.0014						
161.629						
0.0014						

163.613
165.228
155.06
165.129
0.0016
0.00114
165.126
167.661
0.0012
0.00116

freshness period= 3100						
RTT(sec)	Συνολική διάρκεια λήψης 10 πακέτων μετρήσεων (sec)	Total Interests	Satisfied Interests	Success Ratio	Cache hits	Cache hit ratio
Timeout	2093.544	21	20	0.952381	8	0.381
123.444						
155.854						
164.131						
172.546						
152.692						
0.0056						
0.001						
0.0008						
160.11						
0.0011						
160.116						
165.123						
160.12						
165.139						
0.0011						
0.0009						
179.023						
145.234						
0.001						
0.0008						

freshness period= 3700						
RTT(sec)	Συνολική διάρκεια λήψης 10 πακέτων μετρήσεων (sec)	Total Interests	Satisfied Interests	Success Ratio	Cache hits	Cache hit ratio
Timeout	2275.64	22	20	0.9091	8	0.364
149.021						

160.142
160.117
160.124
167.327
0.0015
0.0017
0.0017
153.15
0.0012
164.932
160.145
165.163
160.106
0.0013
0.0013
160.2
Timeout
135.196
0.003
0.0011

freshness period= 4300						
RTT(sec)	**Συνολική διάρκεια λήψης 10 πακέτων μετρήσεων (sec)**	**Total Interests**	**Satisfied Interests**	**Success Ratio**	**Cache hits**	**Cache hit ratio**
Timeout	2428.966	23	20	0.869565	8	0.348
139.717						
159.137						
167.105						
155.677						
167.148						
0.0018						
0.0012						
0.0012						
Timeout						
127.294						
0.0011						
Timeout						
152.792						
144.553						
165.145						
166.742						

0.0012
0.0011
158.527
155.121
0.0013
0.0012