ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

ΣΧΟΛΗ ΕΠΙΣΤΗΜΩΝ & ΤΕΧΝΟΛΟΓΙΑΣ ΤΗΣ ΠΛΗΡΟΦΟΡΙΑΣ
SCHOOL OF INFORMATION SCIENCES & TECHNOLOGY
ΜΕΤΑΠΤΥΧΙΑΚΟ ΕΠΙΣΤΗΜΗ ΤΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
MSc IN COMPUTER SCIENCE

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΜΣ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Διπλωματική Εργασία
Μεταπτυχιακού Διπλώματος Ειδίκευσης

«*A Hybrid In-Network Caching Policy for Information-Centric Networks*»

Παπαευσταθίου Ανάργυρος

ΕΥ1305

Επιβλέπων: Ξυλωμένος Γεώργιος

ΑΘΗΝΑ, ΙΟΥΛΙΟΣ 2015

Πίνακας Περιεχομένων

Λίστα Εικόνων

Λίστα Πινάκων

Λίστα Συντομογραφιών και Συμβόλων

CCN	Content-Centric Networking
CDN	Content Delivery Network
CR	Content Router
CS	Content Store
DoS	Denial of Service
DPI	Deep Packet-Inspection
DRAM	Dynamic Random-Access Memory
FIB	Forwarding Information Base
FIFO	First In First Out
ICN	Information-Centric Networking
ISP	Internet Service Provider
L1, L2	Layer 1, Layer 2 ευρετήρια
LFBL	Listen First Broacast Later
LRU	Least Recently Used
MT	Master thesis
MTU	Maximum Transfer Unit
NDN	Named-Data Networking
OPC	Object-Oriented Packet Cache
PIT	Pending Interest Table
SRAM	Static Random-Access Memory
URL	Uniform Resource Locator

Περίληψη

Ένα από τα πιο δημοφιλή χαρακτηριστικά που προσφέρουν οι αρχιτεκτονικές Information-Centric Networking (ICN) είναι η δυνατότητά τους να υποστηρίζουν packet-level caching σε κάθε κόμβο του δικτύου. Κατονομάζοντας ξεχωριστά κάθε πακέτο πληροφορίας, οι αρχιτεκτονικές αυτές επιτρέπουν στους δρομολογητές να μετατρέψουν τους ενταμιευτές ουρών (queuing buffers) που κατέχουν σε packet caches, εκμεταλλεύοντας έτσι τους υπάρχοντες αποθηκευτικούς πόρους του δικτύου. Ωστόσο η απόδοση των packet caches περιορίζεται από μια σειρά από αδυναμίες. Οι αδυναμίες αυτές ουσιαστικά απορρέουν από την μικρή χωρητικότητα της γρήγορης μνήμης SRAM των δρομολογητών, στην οποία αποθηκεύεται το ευρετήριο για τα πακέτα που είναι αποθηκευμένα στην αργή DRAM μνήμη. Έτσι λοιπόν, προτείνουμε το Object-oriented Packet Caching (OPC), μια καινοτόμο πολιτική caching για τα δίκτυα ICN, η οποία καταφέρνει και ξεπερνά τα προβλήματα που δημιουργεί ο περιορισμός της μικρής μνήμης SRAM, συνδυάζοντας δεικτοδότηση επιπέδου αντικειμένου (object-level indexing) στην SRAM με αποθήκευση επιπέδου πακέτων (packet-level storage) στην DRAM.

Στην παρούσα διπλωματική εργασία, υλοποιήσαμε την OPC πολιτική και βάση των πειραμάτων που πραγματοποιήσαμε, δείξαμε ότι η προσέγγισή μας αποδίδει καλύτερα σε σχέση τις κλασσικές πολιτικές packet-level caching (π.χ LRU), αυξάνοντας σημαντικά την απόδοση όσον αφορά το cache hit ratio και ταυτόχρονα μειώνοντας τόσο τον φόρτο του δικτύου όσο και τον φόρτο του server.

Επιπλέον, υιοθετούμε την απόφαση να μην έχουμε cache σε κάθε κόμβο του δικτύου, καθώς με βάση τα πειράματά μας βρήκαμε ότι έχουμε την ίδια ή και, αρκετές φορές, καλύτερη απόδοση όταν κάνουμε caching σε ένα υποσύνολο των κόμβων (access nodes ή central nodes).

1 Εισαγωγή

1.1 Η αρχιτεκτονική του Ίντερνετ σήμερα

Η αρχιτεκτονική του Ίντερνετ που εφαρμόζεται μέχρι και σήμερα, σχεδιάστηκε πάνω σε ένα μοντέλο επικοινωνίας επικεντρωμένο στους υπολογιστές των χρηστών, έτσι ώστε να καλύπτει τις ανάγκες που είχαν στα πρώτα χρόνια χρησιμοποίησής τους. Τα προβλήματα που αντιμετωπίζει το Ίντερνετ σήμερα είναι απόρροια αυτής της αρχιτεκτονικής, η οποία σχεδιάστηκε για να ικανοποιήσει τις ανάγκες επικοινωνίας εκείνης της εποχής. Δηλαδή, στόχος ήταν η δημιουργία ενός δικτύου με σκοπό την κατανομή και το μοίρασμα σπάνιων και ακριβών πόρων όπως περιφερειακές συσκευές, κεντρικοί υπολογιστές, και συνδέσεις μεγάλης απόστασης. Ωστόσο η χρήση του Διαδικτύου έχει πλέον εξελιχθεί και συνεχίζει να εξελίσσεται, με τους περισσότερους χρήστες να ενδιαφέρονται για πρόσβαση σε τεράστιες ποσότητες πληροφορίας και περιεχομένου, χωρίς να τους ενδιαφέρει η φυσική θέση στην οποία βρίσκονται οι πληροφορίες αυτές. Παράλληλα η τεράστια ανάπτυξη του Ίντερνετ και η εισαγωγή νέων καινοτόμων εφαρμογών και υπηρεσιών, απαιτούν πλέον μια αρχιτεκτονική ικανή να υποστηρίξει μεγαλύτερη ασφάλεια, φορητότητα και επεκτασιμότητα όσον αφορά την διανομή περιεχομένου. Η αδυναμία της αρχιτεκτονικής του Ίντερνετ να ικανοποιήσει τέτοιες απαιτήσεις, οδήγησε στην εμφάνιση μιας σειράς από "μπαλώματα" ή "πατέντες" (για παράδειγμα Mobile IP) έτσι ώστε να μπορέσει να εξελιχθεί. Οι περισσότερες από αυτές τις πατέντες, εκτός ότι αύξησαν την συνολική πολυπλοκότητα της αρχιτεκτονικής, αποδείχθηκαν μόνο προσωρινές λύσεις. Όλα τα παραπάνω οδήγησαν τους ερευνητές στην αναζήτηση μιας νέας αρχιτεκτονικής Διαδικτύου η οποία να αποτελεί μια ριζική αλλαγή στην μορφή του Internet που γνωρίζουμε μέχρι σήμερα. Έτσι έκαναν την εμφάνισή τους αρχιτεκτονικές όπως CCN (Content-Centric Networking) και ICN(Information-Centric Networking) οι οποίες ευνοούν την ανάπτυξη caching μέσα στο δίκυτο (in-network caching), καθώς και την ανάπτυξη μηχανισμών πολυδιανομής (multicasting), διευκολύνοντας έτσι την αποτελεσματική και έγκαιρη διανομή πληροφοριών προς τους χρήστες.

1.2 Caching

Πρόσφατες έρευνες υποστηρίζουν ότι το World Wide Web (WWW), όπως προηγουμένως αναφέραμε, υπόκειται σιγά σιγά μια μεταμόρφωση. Η μεταμόρφωση αυτή συνοψίζεται στο γεγονός ότι οι χρήστες έχουν μετατραπεί από αποκλειστικά καταναλωτές περιεχομένου σε παρόχους περιεχομένου, και συνεπώς η φύση και ο όγκος των δεδομένων προς μετάδοση έχει σημαντικά τροποποιηθεί [1], [2]. Αναλύσεις από την Cisco δείχνουν ότι τα τελευταία 5 χρόνια η παγόσμια κίνηση IP έχει αυξηθεί περισσότερο από 500%, και θα συνεχίζει να αυξάνεται με τέτοιο ρυθμό έτσι ώστε να τριπλασιαστεί στα επόμενα 5 χρόνια, για να φτάσει τελικά τα 1.6 zettabytes τον χρόνο στο 2018 [3]. Αυτό το φαινόμενο επιβάλλει αρκετά απαιτητικές ανάγκες επεκτασιμότητας για διάφορες δικτυακές υπηρεσίες, όπως Web caches, traffic monitoring, archival systems, και Content Delivery Networks (CDN).

Πολυάριθμες μελέτες και έρευνες έχουν ήδη εξετάσει τον χαρακτήρα του Internet traffic, τονίζοντας την σπουδαιότητα του caching με σκοπό την μείωση του δικτυακού φόρτου, το οποίο επιτυγχάνεται μέσω ανίχνευσης επαναλαμβανόμενων μεταδόσεων [4], [5], [6], [7]. Τα τελευταία χρόνια, οι HTTP και FTP caches υπήρξαν αναπόσπαστο κομμάτι ζωτικής σημασίας των δικτύων, φέρνοντας δημοφιλές πε-

ριεχόμενο πιο κοντά στον χρήστη, συνεισφέροντας έτσι σε ταχύτερη διανομή περιεχομένου και μειώνοντας τον φόρτο του δικτύου και του εξυπηρετητή, τόσο στους Παρόχους Υπηρεσιών Ίντερνετ (Internet Service Providers ή ISPs) όσο και σε μεγάλα δίκτυα κορμού.

Ωστόσο, αρκετές πρόσφατες μελέτες αμφισβητούν την αποτελεσματικότητα των παραπάνω συστημάτων [8], [9] και προτείνουν καινοτόμες λύσεις οι οποίες επιτρέπουν ανίχνευση πλεονασμού (redundancy detection) σε μικρότερα μεγέθη, όπως για παράδειγμα πακέτα, αντί για ολόκληρα αντικείμενα. Αυτές οι προσεγγίσεις, γνωστές ως packet-caches, είναι σε μεγάλο βαθμό αποτελεσματικότερες στην εξάλειψη επαναλαμβανόμενου περιεχομένου, καθώς ανιχνεύουν πλεονασμό (redundancy) μέσα στα πακέτα αντί σε ολόκληρα τα αντικείμενα. Παρόλα αυτά, οι packet-caches παρουσιάζουν σημαντικά ζητήματα επεκτασιμότητας και ευελιξίας, όπως διαχείριση μεγάλων ευρετηρίων αναζήτησης, εκτέλεση αναζήτησης ανά πακέτο σε wire-speed καθώς και ανάγκη για συγχρονισμένα ευρετήρια αναζήτησης.

Για τους παραπάνω λόγους στην παρούσα διπλωματική εργασία προτείνουμε μια καινοτόμα πολιτική διαχείρισης για την μνήμη cache στο ICN η οποία λειτουργεί σε επίπεδο αντικειμένου (Object-level) μειώνοντας το μέγεθος των ευρετηρίων, αποφεύγοντας τα μικρά hit-ratios στους κεντρικούς κόμβους του δικτύου (core nodes) και παρέχοντας καλύτερο έλεγχο συμφόρησης. Η πολιτική αυτή ονομάστηκε Object-level Packet Caching (OPC) και θα παρουσιαστεί στην συνέχεια.

2 Θεωρητικό υπόβαθρο

Στην ενότητα αυτή θα παρουσιαστούν οι αρχιτεκτονικές οι οποίες αποτελούν το απαραίτητο υπόβαθρο για την παρούσα διπλωματική εργασία.

2.1 Πληροφοριοκεντρικά Δίκτυα (ICN – Information Centric Networking)

2.1.1 Σύντομη Επισκόπηση

Το ICN είναι μια νέα και πολλά υποσχόμενη αρχιτεκτονική δικτύου η οποία είναι υποψήφια για την αρχιτεκτονική του μελλοντικού Ίντερνετ. Εμπνευσμένη από το γεγονός ότι το Ίντερνετ σήμερα χρησιμοποιείται όλο και περισσότερο για μετάδοση και διάδοση πληροφορίας, παρά για επικοινωνία μεταξύ 2 άκρων, το ICN στοχεύει να ικανοποιήσει τις παρούσες και τις μελλοντικές ανάγκες πιο αποτελεσματικά από την αρχιτεκτονική που στηρίζεται το Ίντερνετ σήμερα. Κατονομάζοντας την πληροφορία στο Επίπεδο Δικτύου (Network Layer), το ICN ευνοεί την ανάπτυξη caching μέσα στο δίκτυο (in-network caching), καθώς και μηχανισμούς πολυδιανομής (multicasting), διευκολύνοντας έτσι την αποτελεσματική και έγκαιρη διανομή πληροφοριών προς τους χρήστες. Το ICN έχει επίσης την δυνατότητα να αντιμετωπίσει άλλος περιορισμούς στους οποίους υπόκειται η παρούσα αρχιτεκτονική του Ίντερνετ, όπως διαχείρηση κινητικότητας (mobility management), και επιβολή ασφάλειας (security enforcement). Παρακάτω παρουσιάζονται τα 4 στοιχεία στα οποία δίνει έμφαση η αρχιτεκτονική αυτή.

2.1.2 Ονομασία Πληροφορίας (Information Naming)

Το Ίντερνετ έχει μετατραπεί από ένα ακαδημαϊκό δίκτυο σε μία πλέον παγκόσμια υποδομή η οποία χρησιμοποιείται για να υποστηρίζει την διανομή πληροφορίας και περιεχομένου μεγάλης κλίμακας. Οι χρήστες ενδιαφέρονται πλέον να λαμβάνουν πληροφορία, περιεχόμενο ή δεδομένα, οπουδήποτε μπορεί αυτά να βρίσκονται, χωρίς να χρειάζεται να έχουν πρόσβαση σε ένα συγκεκριμένο υπολογιστικό περιβάλλον (υπολογιστή υπηρεσίας – host ή εξυπηρετητή – server). Ωστόσο το Ίντερνετ σήμερα είναι σχεδιασμένο έτσι ώστε ο χρήστης να χρειάζεται να καθορίσει σε κάθε αίτηση όχι μόνο την πληροφορία που θέλει να λάβει, αλλά και τον συγκεκριμένο εξυπηρετητή ο οποίος κατέχει την πληροφορία αυτή. Εκτός και αν χρησιμοποιηθούν επιπρόσθετες λειτουργίες, οι μηχανισμοί του επιπέδου δικτύου του Ίντερνετ είναι ανίκανοι να εντοπίσουν και να φέρουν την πληροφορία από τη βέλτιστη τοποθεσία στην οποία αυτή φιλοξενείται.

Η προσέγγιση του ICN ουσιαστικά αποσυνδέει την πληροφορία από την πηγή της, με την έννοια ότι η πληροφορία έχει κατονομαστεί, διευθυνσιοδοτηθεί και αντιστοιχηθεί ανεξάρτητα από την τοποθεσία στην οποία βρίσκεται [10], [11]. Στην αρχιτεκτονική ICN κατονομάζεται ένα κομμάτι πληροφορίας, αντί να καθορίζεται ένα ζευγάρι προέλευσης – προορισμού επικοινωνίας. Η μετακίνηση αυτή από το μοντέλο ονομασίας κόμβων (host naming) στο μοντέλο ονομασίας πληροφορίας (information naming) έχει ως θετική επίπτωση η ανάκτηση πληροφορίας να είναι καθοδηγούμενη από τον παραλήπτη. Σε αντίθεση με το Ίντερνετ σήμερα, στο οποίο οι αποστολείς έχουν αποκλειστικό έλεγχο πάνω στα δεδο-

μένα προς ανταλλαγή, στο ICN τα δεδομένα μπορούν να ληφθούν μόνο εφόσον ζητηθούν ρητά από τον παραλήπτη. Στο ICN, έπειτα από την αποστολή ενός αιτήματος, το δίκτυο είναι υπεύθυνο για να εντοπίσει την καλύτερη πηγή η οποία μπορεί να παρέχει την πληροφορία που ζητήθηκε. Έτσι η λειτουργία δρομολόγησης των αιτήσεων πληροφορίας συνοψίζεται στην εύρεση της καλύτερης (πιο κοντινής) πηγής που κατέχει την πληροφορία αυτή, βασισμένη σε ένα όνομα το οποίο είναι ανεξάρτητο από το που βρίσκεται η πληροφορία αυτή.

2.1.3 Παράδοση Πληροφορίας (Information Delivery)

Η στροφή προς εφαρμογές κεντρικοποιημένες στο περιεχόμενο (content-centric), απαιτεί από το Ίντερνετ την αποτελεσματική διανομή τεράστιων ποσοτήτων πληροφοριών, καθώς και την διαχείριση απρόβλεπτων εκρηκτικών κυμάτων κίνησης, τα οποία είναι γνωστά ως flash crowds. Ωστόσο, το Ίντερνετ σήμερα δεν παρέχει από μόνο του μηχανισμούς για να μπορέσει να χειριστεί flash crowd γεγονότα, καθώς και να διανείμει αποτελεσματικά την πληροφορία. Σύμφωνα με την παρούσα αρχιτεκτονική του Ίντερνετ, τα δεδομένα προς μεταφορά αντιμετωπίζονται ως μια σειρά από bytes τα οποία πρέπει να μεταφερθούν από μια προέλευση σε έναν προορισμό. Έτσι λοιπόν το δίκτυο δεν γνωρίζει τίποτα για τα δεδομένα τα οποία μεταφέρει, και ως εκ τούτου δεν μπορεί να αντιληφθεί τυχόν βελτιστοποιήσεις οι οποίες διαφορετικά θα ήταν δυνατόν να συμβούν (π.χ in-network caching, information replication, information-aware traffic engineering). Το Ίντερνετ για να μπορέσει να εκμεταλλευτεί όλες τις δυνατότητες αποθήκευσης μέσα στο δίκτυο (in-network storage), θα πρέπει να επεκταθεί με μηχανισμούς πληροφορίας οι οποίοι να μπορούν να αναγνωρίζουν και να ανακτούν την πληροφορία από την βέλτιστη τοποθεσία [12]. Η εμφάνιση τέτοιων τεχνικών στο επίπεδο εφαρμογής (π.χ Web caching) επιβεβαιώνει ότι όλες αυτές οι ιδέες αποτέλεσαν μια μεταγενέστερη σκέψη για το Ίντερνετ.

Οι τεχνικές αυτές εφαρμόστηκαν από τα Δίκτυα Διανομής Περιεχομένου ή Content Delivery Networks (CDNs), στο επίπεδο εφαρμογής. Παρόλα αυτά, στην περίπτωση των flash crowds η ποσότητα, η τοποθεσία καθώς και ο προορισμός της κίνησης δεν μπορούν να προβλεφθούν. Επιπρόσθετα, η επένδυση για να υπάρχει ένα CDN που να περιλαμβάνει όλες αυτές τις πιθανές περιπτώσεις είναι σίγουρα μη εφικτή από οικονομικής άποψης, ειδικά εάν λάβουμε υπόψη την σταθερή αύξηση που έχουν οι πληροφορίες που δημιουργούνται από τους χρήστες.

Για όλους τους παραπάνω λόγους, θα ήταν προτιμότερο να έχουμε μια νέα ευρεία υποδομή Ίντερνετ η οποία να υποστηρίζει μηχανισμούς μέσα στο ίδιο το δίκτυο, για αποτελεσματική ανάκτηση πληροφορίας. Έτσι λοιπόν, παρουσιάζοντας την ICN αρχιτεκτονική, το δίκτυο μπορεί να ικανοποιεί πλέον μια αίτηση για λήψη πληροφορίας όχι μόνο μέσω του εντοπισμού της αρχικής πηγής, αλλά επίσης με την χρησιμοποίηση πιθανά πολλαπλών κρυφών μνημών (in-network caches) οι οποίες κρατάνε αντίγραφα της επιθυμητής πληροφορίας. Αυτό μπορεί να πραγματοποιηθεί χωρίς να χρειάζεται να καταφύγουμε σε επιπρόσθετες δαπανηρές λύσεις όπως τα CDNs, διότι το επίπεδο δικτύου στο ICN λειτουργεί άμεσα πάνω σε πληροφορία που έχει κατονομαστεί.

Τα πακέτα δεδομένων στην αρχιτεκτονική ICN παίρνουν το όνομά τους ανάλογα με την πληροφορία που κατέχουν. Έτσι λοιπόν μπορούν να αποθηκευτούν σε κρυφές μνήμες (caches) και να ανακτηθούν πολύ εύκολα, αντίθετα με το κλασσικό Ίντερνετ στο οποίο απαιτούνται δαπανηρές τεχνικές όπως Deep Packet Inspection (DPI) [13], [14], [15]. Επιπλέον, δίνοντας όνομα στην πληροφορία το ICN επιτρέπει

την ομαδοποίηση των αιτήσεων που αφορούν την ίδια πληροφορία, διευκολύνοντας έτσι την διανομή στους αντίστοιχους ενδιαφερόμενους προορισμούς μέσω multicast forwarding.

2.1.4 Κινητικότητα (Mobility)

Η υπάρχουσα αρχιτεκτονική Ίντερνετ όπως έχουμε ήδη αναφέρει, σχεδιάστηκε με γνώμονα την από σημείο σε σημείο επικοινωνία μεταξύ δύο κόμβων, όπου το αναγνωριστικό του κόμβου (διεύθυνση IP) είναι το κεντρικό στοιχείο. Επομένως, μιλάμε για σταθερούς υπολογιστές υπηρεσίας (hosts), των οποίων οι διευθύνσεις IP ανήκουν σε ένα δίκτυο. Ωστόσο στατιστικές έρευνες δείχνουν μια συνεχόμενη αύξηση στον αριθμό των μη σταθερών κόμβων που έχουν πρόσβαση στο Ίντερνετ, και προβλέπουν ότι μέχρι τα τέλη του 2015, η κίνηση που προέρχεται από ασύρματα τερματικά θα ξεπεράσει την κίνηση που προέρχεται από τα ενσύρματα [16]. Οι ασύρματες και κινητές συσκευές, ενώ μπορούν να πολύ εύκολα να μεταβούν από ένα δίκτυο σε ένα άλλο αλλάζοντας τις IP διευθύνσεις τους, αδυνατούν να επιτύχουν συνεχής συνδεσιμότητα καθώς βρίσκονται σε κίνηση, ζήτημα το οποίο αποτελεί πλέον μια όλο και σημαντικότερη απαίτηση για το μελλοντικό Ίντερνετ. Λύσεις όπως το Mobile IP, που έχουν προταθεί για να αντιμετωπίσουν το πρόβλημα αυτό κρίθηκαν προσωρινές και μη αποτελεσματικές, λόγω προβλημάτων όπως η τριγωνική δρομολόγηση (triangular routing), η δρομολόγηση κοιλάδας (valley routing) και η παραβίαση της πολιτικής εξόδου (exit policy violation).

Στο ICN, το πρόβλημα κινητικότητας των κόμβων αντιμετωπίζεται με το μοντέλο επικοινωνίας έκδοσης-εγγραφής (publish-subscribe) [17], στο οποίο οι χρήστες που ενδιαφέρονται για μια πληροφορία εγγράφονται (*subscribe)* με την έννοια ότι δηλώνουν το ενδιαφέρον τους για την πληροφορία αυτή, ενώ οι χρήστες που προσφέρουν την πληροφορία την εκδίδουν (*publish)*, διαφημίζοντας την έτσι στο δίκτυο. Μέσα στο δίκτυο οι μεσάζοντς (*brokers)* είναι υπεύθυνοι να ταιριάξουν τις εγγραφές (*subscriptions*) με τις εκδόσεις (*publications*), μέσω μιας λειτουργίας *rendezvous*. Η δύναμη αυτού του μοντέλου επικοινωνίας πηγάζει από το γεγονός ότι η επικοινωνία μεταξύ publishers και subscribers είναι ασύγχρονη. Ο publisher μπορεί να εκδώσει μια πληροφορία πριν την ζητήσει κάποιος subscriber και οι subscribers μπορούν να αρχίσουν να πραγματοποιούν αιτήσεις για μια πληροφορία, αμέσως μετά την ανακοίνωση ότι η πληροφορία αυτή είναι διαθέσιμη. Επίσης, συνήθως δεν υπάρχει κάποιου είδους αναφορά μεταξύ publishers και subscribers και έτσι οι publishers δεν γνωρίζουν πόσοι subscribers λαμβάνουν μια δημοσίευση για μια συγκεκριμένη πληροφορία. Αντίστοιχα, οι subscribers δεν γνωρίζουν πόσοι διαφορετικοί publishers παρέχουν την πληροφορία για την οποία ενδιαφέρονται [17]. Αυτές οι ιδιότητες αποτελούν το κλειδί για το πρόβλημα της κινητικότητας των κόμβων, καθώς οι κινητοί κόμβοι μπορούν απλά να δηλώσουν ξανά το ενδιαφέρον τους στέλνοντας μηνύματα *interest* για την πληροφορία που θέλουν μετά από μια μετακίνηση (handoff), και το δίκτυο μπορεί να ανακατευθύνει αυτά τα subscriptions σε κοντινές κρυφές μνήμες (caches) αντί να τα ζητήσει από τον αρχικό publisher.

2.1.5 Ασφάλεια (Security)

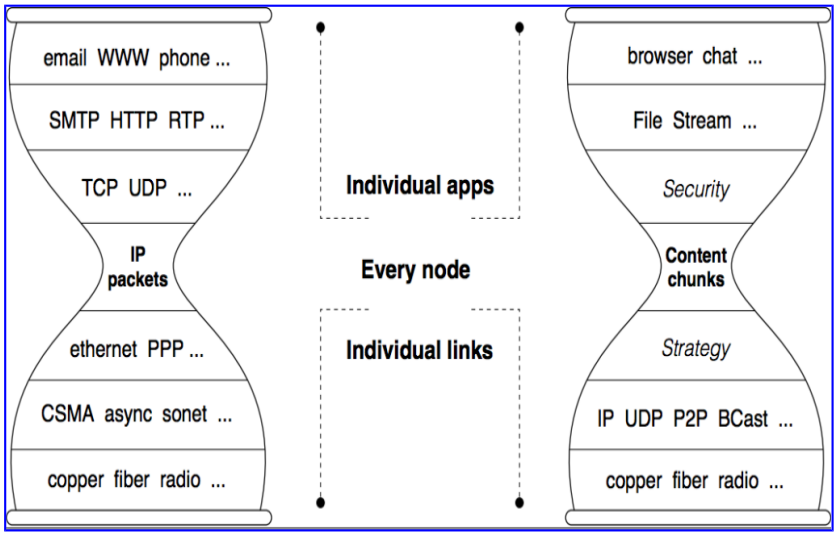
Το Ίντερνετ σχεδιάστηκε για να λειτουργεί σχεδόν ουτοπικά, σε ένα εντελώς αξιόπιστο περιβάλλον όπου έννοιες και ενέργειες όπως η αυθεντικοποίηση χρηστών και δεδομένων, ακεραιότητα και ιδιωτικότητα, δεν περιλαμβάνονταν στις αρχικές απαιτήσεις. Επιπρόσθετα, το Ίντερνετ είχε σχεδιαστεί για να προωθεί οποιαδήποτε κίνηση εισερχόταν στο δίκτυο, γεγονός που οδηγούσε σε μια ανισορροπία δύνα-

μης μεταξύ αποστολέων και παραληπτών. Τα χαρακτηριστικά αυτά επέτρεψαν σε spammers, hackers και γενικά κακόβουλους χρήστες να πραγματοποιούν επιθέσεις Denial of Service (DoS) καλύπτοντας ταυτόχρονα τα ίχνη τους. Για την αντιμετώπιση τέτοιων κακόβουλων επιθέσεων αναπτύχθηκαν μηχανισμοί όπως firewalls, spam filters, κτλ, καθώς και νέα πρωτόκολλα ασφάλειας (π.χ IPSec, DNSSec) τα οποία συμπλήρωναν κάποια άλλα που υπήρχαν ήδη. Ωστόσο, τέτοιες λύσεις δεν καταφέρνουν να διεισδύσουν βαθιά στο δίκτυο και κακόβουλα δεδομένα καταφέρνουν ακόμα και προωθούνται. Η από άκρη-σε άκρη (end-to-end) φιλοσοφία του Ίντερνετ, έχει μέχρι τώρα εμποδίσει την εγκατάσταση μηχανισμών ασφάλειας και αξιοπιστίας βαθύτερα μέσα στο δίκτυο, εκεί όπου θα ήταν πιο αποτελεσματικοί, τόσο στην αποφυγή ή αναγνώριση, όσο και στην αναχαίτιση κακόβουλων επιθέσεων.

Στην αρχιτεκτονική ICN, δεν υπάρχει ροή δεδομένων, εκτός και αν ένας χρήστης έχει ζητήσει ρητά να λάβει ένα συγκεκριμένο κομμάτι πληροφορίας. Έτσι μειώνεται σημαντικά η ποσότητα των ανεπιθύμητων (spam) μεταφορών δεδομένων. Επιπρόσθετα, σε κάποιες ICN αρχιτεκτονικές οι οποίες πιστοποιούν τα ονόματα της πληροφορίας, μπορεί να επιτευχθεί κακόβουλο φιλτράρισμα δεδομένων (malicious data filtering) ακόμα και από μηχανισμούς που βρίσκονται μέσα στο δίκτυο (in-network mechanisms). Τέλος η αποσύνδεση της επικοινωνίας που επιτυγχάνουν οι αρχιτεκτονικές ICN μεταξύ των χρηστών που αιτούνται να λάβουν μια πληροφορία και μεταξύ εκείνων που επεξεργάζονται μια πληροφορία, αποτελεί από μόνη της ένα βήμα προς την αποτελεσματικότερη αντιμετώπιση των επιθέσεων denial of service (DoS). Προσθέτοντας ένα ενδιάμεσο σημείο ή σημείο μεσολάβηση (indirection point) στο οποίο γίνεται ο διαχωρισμός μεταξύ των δύο οντοτήτων, το ICN μπορεί να αξιολογήσει εκεί τις αιτήσεις για συγκεκριμένα κομμάτια πληροφορίας, πριν αυτά φτάσουν στον τελικό τους προορισμό. Αυτή η προσέγγιση, γνωστή ως *indirection,* εκτός από άμυνα κατά των DoS επιθέσεων, μπορεί επίσης να προσφέρει και στην υποστήριξη της ιδιωτικότητας του χρήστη, καθώς ο publisher δεν χρειάζεται να γνωρίζει σχετικά με τις ταυτότητες που έχουν οι subscribers.

2.2 Το σύστημα NDN (Named Data Networking)

2.2.1 Σύντομη Επισκόπηση



Εικ. 1: Αρχιτεκτονικές κλεψύδρας του Ίντερνετ και του NDN
Πηγή: http://named-data.net/project/execsummary/

Το Content Centric Networking (CCN) [18] είναι μια πρωτοπόρα ολοκληρωμένη ICN αρχιτεκτονική, η οποία αντικατοπτρίζεται πλήρως στο Named Data Networking (NDN) [19] σύστημα. Το NDN οραματίζεται την πλήρη αναδιαμόρφωση της στοίβας του πρωτοκόλλου του Ίντερνετ. Στο παραπάνω σχήμα (Εικ. 1), βλέπουμε ότι η στοίβα του NDN διαφέρει από εκείνη της κλασσικής αρχιτεκτονικής του Ίντερνετ που όλοι γνωρίζουμε σήμερα, τοποθετώντας στη λεπτή μέση της αρχιτεκτονικής του Ίντερνετ την ανταλλαγή των ονομαζόμενων δεδομένων (named data ή content chunks) και χρησιμοποιώντας διάφορες δικτυακές τεχνολογίες κάτω από τη μέση για διασύνδεση με το IP καθώς και με άλλα πρωτόκολλα. Ανάμεσα στο επίπεδο Ονομασίας Δεδομένων (Named Data) και στο επίπεδο των Δικτυακών Τεχνολογιών βρίσκεται το επίπεδο Στρατηγικής (Strategy), το οποίο έχει σαν στόχο την βελτιστοποίηση της χρήσης των πόρων (π.χ επιλογή καναλιού σε ένα κόμβο πολλαπλών συνδέσεων). Ακριβώς από πάνω από το επίπεδο named data βρίσκεται το επίπεδο Ασφάλειας (Security), το οποίο εφαρμόζει απευθείας λειτουργίες ασφάλειας πάνω στα κομμάτια περιεχομένου (content chunks).

2.2.2 Ονομασία (Naming)

Τα ονόματα στο NDN ακολουθούν ιεραρχική δομή και η μορφή τους μοιάζει με εκείνη ενός URL, για παράδειγμα /aueb.gr/ai/main.html. Ωστόσο, τα ονόματα αυτά δεν είναι απαραίτητα URLs, καθώς το πρώτο μέρος τους δεν αποτελείται από ένα DNS όνομα ούτε από μια IP διεύθυνση. Αντίθετα δεν χρειάζεται καν να είναι σε ανθρώπινη αναγνώσιμη μορφή (human-readable), επιτρέποντας σε κάθε κομμάτι ονόματος να μπορεί να είναι οτιδήποτε, είτε μία human-readable συμβολοσειρά, είτε μια τιμή κατακερματισμού. Στο NDN ένα αίτημα (request) για ένα όνομα ταιριάζει με οποιοδήποτε κομμάτι πληροφορίας, η οποία έχει το ζητούμενο όνομα ως πρόθεμα. Για παράδειγμα, το /aueb.gr/ai/main.html ταιριάζει με ένα αντικείμενο πληροφορίας με όνομα το /aueb.gr/ai/main.html/_v1/_s1, το οποίο υποδηλώνει το 1ο κομμάτι (segment - s1) της πρώτης έκδοσης (version - v1) του ζητούμενου κομματιού πληροφορίας. Αντίστοιχα μπορεί να ζητηθεί είτε το 2ο segment ως /aueb.gr/ai/main.html/_v1/_s2, είτε

κάποιο άλλο version, π.χ /aueb.gr/ai/main.html/_v2. Ο τρόπος με τον οποίο τα αντικείμενα πληροφορίας είναι κατακερματισμένα, είναι γνωστός στην εφαρμογή του subscriber, έτσι η αντιστοίχηση προθέματος (prefix matching) επιτρέπει στην εφαρμογή να ανακαλύπτει ποια κομμάτια πληροφορίας είναι διαθέσιμα. Επιπλέον, επιτρέπει στον subscriber να μπορεί να ζητήσει δεδομένα τα οποία δεν έχουν παραχθεί ακόμα με τον εξής τρόπο: ο publisher διαφημίζει ότι μπορεί να ικανοποιήσει αιτήσεις για ένα συγκεκριμένο πρόθεμα και έπειτα επιστρέφει αντικείμενα πληροφορίας με ολοκληρωμένα NDN ονόματα, καθιστώντας έτσι δυνατή την υλοποίηση ποικίλων εφαρμογών (π.χ τηλεδιάσκεψη - voice conferencing [20]) όπου τα αντικείμενα πληροφορίας παράγονται δυναμικά και άρα τα πλήρη ονόματά τους δεν μπορεί να είναι γνωστά εκ των προτέρων.

2.2.3 Επίλυση ονομάτων και δρομολόγηση δεδομένων (Name Resolution and Data Routing)

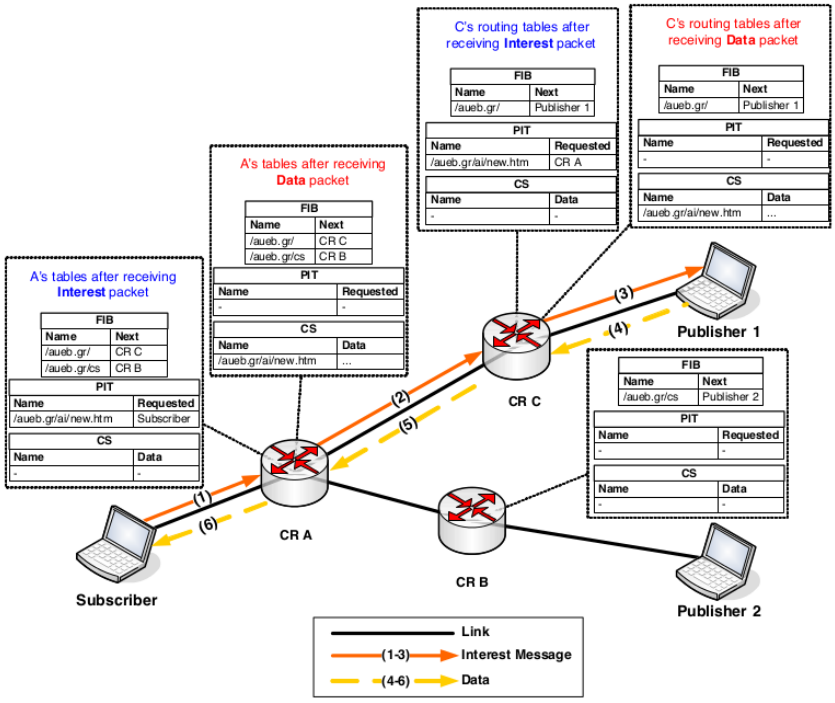
Στο NDN υπάρχουν δύο είδους μηνύματα, τα INTEREST και τα DATA, καθένα από τα οποία έχει το όνομα της αιτούμενης ή μεταφερόμενης πληροφορίας αντίστοιχα. Τα INTEREST μηνύματα αποστέλλονται από τους subscribers για την αίτηση των αντικειμένων πληροφορίας, τα οποία με τη σειρά τους παραδίδονται από τους publishers στους subscribers με την μορφή των DATA μηνυμάτων. Όπως φαίνεται στην παρακάτω εικόνα (Εικ. 2) όλα τα μηνύματα προωθούνται απ' άλμα σε άλμα (hop by hop) από τα Content Routers (CRs), με κάθε CR να διατηρεί 3 δομές δεδομένων:

- Το Forwarding Information Base (FIB) που αντιστοιχεί τα ονόματα πληροφορίας με τις διεπαφές εξόδου που πρέπει να χρησιμοποιηθούν για την προώθηση των INTEREST μηνυμάτων,
- Το Pending Interest Table (PIT) το οποίο σημειώνει από ποια εισερχόμενη διεπαφή έχουν φτάσει τα INTEREST μηνύματα που εκκρεμούν και
- Το Content Store (CS), το οποίο λειτουργεί ως μια τοπική cache για τα αντικείμενα πληροφορίας που έχουν περάσει από τον συγκεκριμένο CR.

Η λειτουργία του Data Routing συνοψίζεται ως εξής:

Όταν φτάνει ένα INTEREST μήνυμα, ο CR αποσπά το όνομα της πληροφορίας και ψάχνει να δει στο CS του αν υπάρχει αντικείμενο πληροφορίας του οποίου το όνομα να ταιριάζει με το πρόθεμα της αιτούμενης πληροφορίας. Αν βρεθεί τέτοιο αντικείμενο, τότε το CS το αποστέλλει με ένα DATA μήνυμα απευθείας από την διεπαφή που ήρθε το μήνυμα INTEREST και το INTEREST διαγράφεται. Σε αντίθετη περίπτωση, δηλαδή αν δεν βρεθεί αντικείμενο που να ταιριάζει, τότε ο router (CR) πραγματοποιεί μια αναζήτηση με τον κανόνα του longest prefix match στο FIB του, έτσι ώστε να αποφασίσει προς τα πού θα προωθήσει το συγκεκριμένο INTEREST. Αν βρεθεί κάποια εγγραφή στο FIB με το longest prefix match, τότε ο CR καταγράφει την εισερχόμενη διεπαφή του INTEREST στο PIT και έπειτα προωθεί το INTEREST προς εκείνο το CR που υποδεικνύει το FIB. Στο σχήμα Εικ. 2 φαίνονται οι παραπάνω λειτουργίες.

Όταν βρεθεί ένα αντικείμενο πληροφορίας το οποίο ταιριάζει με το ζητούμενο όνομασε έναν κόμβο publisher ή σε ένα CS, τότε η πληροφορία επιστρέφεται μέσα σε ένα DATA μήνυμα και το INTEREST μήνυμα απορρίπτεται. Το DATA μήνυμα προωθείται πίσω στον subscriber hop-by-hop, βασισμένο στην κατάσταση που είναι καταγεγραμμένη στα PITs. Συγκεκριμένα όταν το CR λαμβάνει ένα μήνυμαDATA, τότε πρώτα αποθηκεύει το αντίστοιχο αντικείμενο πληροφορίας στο CS και έπειτα πραγματοποιεί



Εικ. 2: Η αρχιτεκτονική του NDN. CR - Content Router, FIB - Forwarding Information Base, PIT - Pending Interest Table και CS - Content Store.
Πηγή: A survey of Information-Centric Networking Research [21]

longest prefix match στο PIT του, έτσι ώστε να εντοπίσει μια εγγραφή που να ταιριάζει με το DATA πακέτο. Στην περίπτωση που η εγγραφή στο PIT περιλαμβάνει πολλαπλές διεπαφές, τότε το DATA μήνυμα αναπαράγεται, επιτρέποντας έτσι διανομή με πολυεκπομπή (multicast delivery). Τέλος, το CR προωθεί το DATA πακέτο σε αυτές τις διεπαφές οι οποίες εμπεριέχονται στην λίστα του PIT και έπειτα διαγράφει την εγγραφή από το PIT. Στην περίπτωση που δεν υπάρχουν εγγραφές που να ταιριάζουν στο PIT, τότε ο CR απορρίπτει το πακέτο DATA ως διπλότυπο.

2.2.4 Caching

Το NDN υποστηρίζει αποθήκευση των πακέτων σε κρυφή μνήμη πάνω στο μονοπάτι αποστολής, ή με ορολογία on-path caching (γεγονός στο οποίο στηρίζεται και η υλοποίηση του simulator της παρούσας διπλωματικής την οποία θα δούμε στο 2ο κεφάλαιο), καθώς κάθε CR όποτε λαμβάνει ένα INTEREST συμβουλεύεται πρώτα το CS του και έπειτα αποθηκεύει τοπικά (caching) όλα τα αντικείμενα πληροφορίας που περιέχονται στα DATA μηνύματα. Το CS μπορεί να χρησιμοποιήσει πολιτική LRU ή οποιαδήποτε άλλη πολιτική αντικατάστασης. Το CS είναι κυρίως χρήσιμο για ανάνηψη από απώλειες πακέτων και για διαχείριση γεγονότων flash crowd, όπου πολλοί χρήστες ζητούν τα ίδια δεδομένα συχνά και διαδοχικά. Ωστόσο, ρεαλιστικά, παρουσιάζει κάποιες αδυναμίες τις οποίες εμείς αντιμετωπίζουμε με την OPC (Object Level Caching) πολιτική την οποία θα δούμε παρακάτω.

Επίσης, το NDN υποστηρίζει και off-path caching, δηλαδή αποθήκευση των πακέτων σε κρυφή μνήμη έξω από το μονοπάτι αποστολής, διανέμοντας ένα INTEREST σε οποιαδήποτε πηγή δεδομένων

που φιλοξενεί το ζητούμενο αντικείμενο πληροφορίας. Το επίπεδο Strategy μπορεί να ανακατευθύνει το INTEREST σε έναν CDN εξυπηρετητή παρά στον publisher. Ωστόσο, αυτό προσθέτει έξτρα πολυπλοκότητα στην αρχιτεκτονική.

2.2.5 Κινητικότητα (Mobility)

Όταν ένας subscriber κινείται σε ένα δίκτυο NDN, μπορεί πολύ απλά να εκδώσει ξανά νέα μηνύματα INTEREST για τα αντικείμενα πληροφορίας που δεν έχει ακόμα λάβει, από την τοποθεσία στην οποία βρίσκεται εκείνη την στιγμή. Τα μηνύματα αυτά θα καταναλωθούν από το PIT του πρώτου κοινού CS και στα δύο μονοπάτια διανομής, δηλαδή τα μονοπάτια πριν και μετά από την μετακίνηση (handshake) αντίστοιχα. Με αυτόν τον τρόπο βέβαια, τα αντίστοιχα αντικείμενα πληροφορίας θα αποσταλούν επίσης και στην παλιά τοποθεσία. Στην περίπτωση του publisher, όταν αυτός κινείται, πρέπει να ενημερωθούν τα FIBs που δείχνουν σε αυτόν. Μια τέτοια ενέργεια απαιτεί ξανά την διαφήμιση των προθεμάτων των ονομάτων για την πληροφορία που ο publisher κατέχει, μέσω του πρωτοκόλλου δρομολόγησης. Καθώς η ενέργεια αυτή προκαλεί αρκετά υψηλό overhead σε λύσεις με μεγάλη κινητικότητα, το NDN χρησιμοποιεί το Listen First Broadcast Later (LFBL) πρωτόκολλο [22], έτσι ώστε να υλοποιήσει την κινητικότητα σε ad-hoc/opportunistic δίκτυα. Στο πρωτόκολλο αυτό χρησιμοποιείται η τεχνική της πλημμύρας, δηλαδή τα INTEREST μηνύματα πλημμυρίζουν το δίκτυο. Όταν μια πηγή για μια ζητούμενη πληροφορία λάβει ένα INTEREST, τότε ακούει το ασύρματο κανάλι για να ανακαλύψει αν κάποιος άλλος κόμβος έχει ήδη στείλει ένα DATA μήνυμα που να ταιριάζει το συγκεκριμένο INTEREST. Αν κάτι τέτοιο δεν συμβαίνει, τότε στέλνει εκείνη το μήνυμα DATA στον subscriber, διαφορετικά αφήνει τον άλλο κόμβο ο οποίος έχει αντιστοιχίσει το μήνυμα DATA με το INTEREST να στείλει την πληροφορία.

2.2.6 Ασφάλεια (Security)

Το NDN υποστηρίζει τον συσχετισμό των human-readable ιεραρχικών ονομάτων πληροφορίας με τα αντίστοιχα αντικείμενα πληροφορίας με τρόπο ο οποίος μπορεί να επαληθευτεί [23]. Κάθε μήνυμα DATA περιέχει μια υπογραφή πάνω στο όνομα και την πληροφορία που περιλαμβάνεται στο μήνυμα, όπως επίσης και πληροφορία σχετικά με το κλειδί που χρησιμοποιείται για να παραχθεί αυτή η υπογραφή, π.χ το δημόσιο κλειδί, ένα πιστοποιητικό για το κλειδί αυτό ή έναν δείκτη προς το κλειδί και το πιστοποιητικό. Με αυτόν τον τρόπο το NDN επιτρέπει σε κάθε κόμβο, συμπεριλαμβανομένων των CRs, να μπορεί να επαληθεύσει την σύνδεση μεταξύ του human-readable ονόματος του πακέτου και της αντίστοιχης πληροφορίας. Για να επιβεβαιώσει ότι η πληροφορία έρχεται από μία εξουσιοδοτημένη πηγή, ο subscriber πρέπει να εμπιστευτεί τον ιδιοκτήτη του δημοσίου κλειδιού. Η ιεραρχική δομή των ονομάτων απλοποιεί το χτίσιμο σχέσεων εμπιστοσύνης, καθώς για παράδειγμα, το όνομα /aueb.gr/ai/main.html μπορεί να υπογραφεί από τον ιδιοκτήτη του /aueb.gr/ai domain, του οποίου το κλειδί μπορεί να πιστοποιηθεί από τον ιδιοκτήτη του /aueb.gr domain.

3 Σχετική έρευνα

3.1 Packet-caches στο IP

Το Packet-level caching στα δίκτυα IP απαιτεί την ανίχνευση πλεονασμού σε αυθαίρετα πακέτα σε ταχύτητες ανάλογες με αυτές των ενσύρματων συνδέσεων (wire-speeds). Το υπολογιστικό κόστος όμως για μια σειρά από γεγονότα όπως ο εντοπισμός των επαναλαμβανόμενων δεδομένων, μέσω ας πούμε, της καταστολής τους[24], το deep packet inspection [25] και το delta coding [26], έχουν εμποδίσει τις Web Caches να λειτουργούν όπως πραγματικά θα θέλαμε. Το ενδιαφέρον για το packet-level caching αναζωπυρώθηκε με την εμφάνιση μιας τεχνικής γνωστής ως Rabin fingerprints [27], η οποία σχεδιάστηκε για την εύρεση πλεονασμού σε Web traffic, εντοπίζοντας παρόμοιες αλλά όχι πανομοιότυπες μεταφορές δεδομένων πληροφορίας σε πραγματικό χρόνο. Παρόλο που αυτή η τεχνική μπορεί να εξολοθρεύσει πλεονασμό ανάμεσα σε διαφορετικές υπηρεσίες, καλύπτει ένα περιορισμένο πεδίο εφαρμογής, αφού απαιτεί την τοποθέτηση σημείων caching ανά ζευγάρια, με τα δύο σημεία να τοποθετούνται στα αντίθετα άκρα του φυσικού καναλιού. Επίσης, απαραίτητη προϋπόθεση είναι τα δύο σημεία να κρατάνε συγχρονισμένα τα ευρετήρια αναζήτησής (look up indexes) τους. Κάποια χρόνια αργότερα, η τεχνική αυτή δοκιμάστηκε και αξιολογήθηκε. Τα αποτελέσματα οδήγησαν τους ερευνητές στο συμπέρασμα ότι, παρόλο που η τεχνική οδήγησε σε καλύτερα αποτελέσματα σε σχέση με μια συνηθισμένη object-cache, μπορεί να εφαρμοστεί μόνο σε υλοποιήσεις περιορισμένης κλίμακας σε συγκεκριμένα δικτυακά κανάλια. Οι συγγραφείς υποστήριξαν ότι η χρησιμότητα της τεχνικής αυτής θα μπορούσε να ενισχυθεί με την υιοθέτηση νέων δικτυακών πρωτοκόλλων τα οποία θα εστίαζαν στην εξάλειψη του πλεονασμού σε επίπεδο καναλιού.

3.2 Packet-caches στο ICN

Όπως έχουμε ήδη αναφέρει, το ξεχωριστό χαρακτηριστικό της ICN αρχιτεκτονικής είναι ότι θεωρεί την πληροφορία ως το κέντρο όλων των λειτουργιών του δικτύου, χωρίς να χρειάζεται να γνωρίζει διευθύνσεις IP, σε αντίθεση με τα IP δίκτυα που εστιάζουν στις από άκρο σε άκρο συνδέσεις. Στις περισσότερες προσεγγίσεις ICN, η πληροφορία ταξιδεύει μέσα στο δίκτυο σαν ένα σύνολο από κομμάτια δεδομένων (data chunks[1]) τα οποία φέρουν ένα μοναδικό αναγνωριστικό. Το αναγνωριστικό αυτό, το οποίο συνήθως είναι το όνομα του περιεχομένου ακολουθούμενο από τον αριθμό σειράς (rank) του πακέτου, τοποθετείται στην επικεφαλίδα του πακέτου, ανακουφίζοντας έτσι τους κόμβους του ICN δικτύου από το υπολογιστικό κόστος που απαιτείται για την ανίχνευση ίδιων πακέτων. Στο ICN αν δύο πακέτα φέρουν το ίδιο αναγνωριστικό, τότε θα πρέπει (στατιστικά) να περιέχουν και το ίδιο περιεχόμενο.

Τα πρωτόκολλα μεταφοράς του ICN είναι κυρίως καθοδηγούμενα από τον παραλήπτη (receiver-driven) [28], [29], ολοκληρώνοντας μια μετάδοση μέσω πολλαπλών ανεξάρτητων μεταδόσεων chunk. Κάθε μετάδοση πυροδοτείται από μια συγκεκριμένη αίτηση πακέτου (packet request) και ολοκληρώνεται από την μετάδοση του αντίστοιχου πακέτου δεδομένων (data packet).

[1] Στην μεγαλύτερη πλειοψηφία των ICN ερευνών, ο όρος chunk αναφέρεται στο Maximum Transfer Unit(MTU), το οποίο είναι το μεγαλύτερο σε μέγεθος πακέτο που επιτρέπεται, και έτσι θεωρούμε τους όρους packet και chunk πανομοιότυπους.

Το μοντέλο αυτό επιτρέπει την εκμετάλλευση και χρησιμοποίηση on-path caches, δηλαδή κρυφές μνήμες ή αλλιώς ενταμιευτές ουρών (buffer queues), οι οποίες τοποθετούνται πάνω στους ICN routers ως προσωρινές αποθήκες, δίνοντας έτσι την δυνατότητα να μπορούν να εξυπηρετούν απευθείας αιτήματα για πακέτα τα οποία έχουν αποθηκευμένα.

Το ICN έχει μεγάλες δυνατότητες για την εκμετάλλευση των packet-level caches, και επομένως πολλοί ερευνητές έχουν εξετάσει τα πλεονεκτήματα που προσφέρει το "πανταχού παρών"(ubiquitous) caching [30], [31], [32], [33]. Οι συγγραφείς αυτών των ερευνών προσπαθούν να αυξήσουν την απόδοση του caching, συναθροίζοντας τις δυνατότητες κάθε cache σε κάθε on-path router. Ωστόσο η εμπειρία έχει δείξει ότι η κατανομή περιεχομένου δεν είναι ομοιόμορφη. Έτσι, κάποιοι συγγραφείς υποστηρίζουν ότι κάνοντας caching σε κάποιους "ιδιαίτερους" κόμβους (super-nodes) που βρίσκονται στις άκρες του δικτύου (edge nodes) είναι δυνατόν να επιτευχθεί σχεδόν ίδια απόδοση με το να κάνουμε caching σε όλους τους κόμβους [34]. Άλλοι πάλι υποστηρίζουν ότι το caching πρέπει να γίνεται σε ένα υποσύνολο κόμβων οι οποίοι ικανοποιούν κάποιες συγκεκριμένες απαιτήσεις (centrality requirements), οι οποίες ορίζουν πόσο κεντρικός είναι ο κόμβος [35].

Από όσο γνωρίζουμε, υπάρχει μόνο μία έρευνα στην βιβλιογραφία η οποία ασχολείται με τις λεπτομέρειες της ICN packet cache δομής [36]. Αυτή η έρευνα προτείνει ένα μοντέλο cache δύο επιπέδων με σκοπό την βελτίωση του χρόνου απόκρισης. Συγκεκριμένα, προτείνει ότι ομάδες από chunks πρέπει να προσκομίζονται από μία αργή μνήμη (SRAM) σε μία γρήγορη (DRAM), έτσι ώστε να έχουμε καλύτερο χρόνο απόκρισης σε συνεχόμενες διαδοχικές αιτήσεις. Ωστόσο οι συγγραφείς προτείνουν αυτήν την σχεδίαση μόνο για edge routers, λόγω των απαιτήσεων αποθήκευσης και του στατικού καταλόγου περιεχομένου που απαιτεί. Για τους in-network routers υποστηρίζουν ότι τόσο η SRAM όσο και η DRAM θα πρέπει να χρησιμοποιηθούν για λειτουργία σε wire-speed. Το μεγαλύτερο ποσοστό των διαφορετικών προσεγγίσεων απλά θεωρούν μια Least Recently Used (LRU) πολιτική αντικατάστασης [31], [32], [34], [35] ή άλλες καινοτόμες πολιτικές για την ομοιόμορφη κατανομή του αποθηκευμένου περιεχομένου (cached content) κατά μήκος του μονοπατιού [30], [33], [37], χωρίς να λαμβάνουν όμως υπόψη τους ότι η απόδοση της cache των δρομολογητών μπορεί να περιοριστεί από το μέγεθος της γρήγορης μνήμης τους.

4 Object-level Packet Caching (OPC)

4.1 Σύντομη Επισκόπηση

Στην παρούσα διπλωματική εργασία ασχολούμαστε με μια πρωτότυπη πολιτική caching στο ICN, η οποία ονομάζεται Object-level Packet-Caching (OPC). Το OPC αντιμετωπίζει αποτελεσματικά τις αδυναμίες που εμφανίζονται στις packet-caches στο ICN, πληρώντας τις απαιτήσεις των ICN δικτύων, όπως για παράδειγμα in-network caching σε wire speeds. Η αρχιτεκτονική OPC πραγματοποιείται σε επίπεδο αντικειμένου (object-level), με την έννοια ότι αποθηκεύει συνεχόμενες ομάδες από πακέτα βελτιώνοντας έτσι την λειτουργία του ελέγχου συμφόρησης (RTT-based congestion control), ενώ ταυτόχρονα μειώνει σημαντικά τις απαιτήσεις για δεικτοδότηση (indexing) με την δομή δύο επιπέδων που διαθέτει. Επιπρόσθετα προορίζεται τόσο για access nodes, δηλαδή κόμβους οι οποίοι βρίσκονται στην άκρη του δικτύου, όσο και για κόμβους οι οποίοι έχουν υψηλό centrality, το οποίο είναι ένας αριθμός που δείχνει πόσο κεντρικός είναι ο κόμβος, δηλαδή αν από αυτόν διέρχονται πολλά ή λίγα μονοπάτια [35]. Με αυτόν τον τρόπο η προσέγγιση OPC καταφέρνει να αποφύγει τα χαμηλά hit-ratios στους κεντρικούς κόμβους (core nodes) του δικτύου, ενώ ταυτόχρονα εξοικονομεί πόρους σε σχέση με άλλες caching πολιτικές οι οποίες ακολουθούν την προσέγγιση να κάνουν caching σε όλους τους κόμβους, λαμβάνοντας εφάμιλλα, ή και πολλές φορές καλύτερα αποτελέσματα, όπως θα δούμε παρακάτω στην αξιολόγηση επίδοσης.

4.2 Αδυναμίες των ICN packet-caches

Το κίνητρο για την υιοθέτηση της αρχιτεκτονικής OPC είναι η αντιμετώπιση των αδυναμιών που παρουσιάζουν οι packet-caches στο ICN. Αυτές είναι κυρίως 4 και είναι οι ακόλουθες:

4.2.1 Περιορισμένοι πόροι μνήμης

Μια λογική απαίτηση για τις packet-level caches όπως έχουμε ήδη αναφέρει, είναι η λειτουργία τους σε wire-speed. Συνήθως ο σχεδιασμός σε μια cache πολιτική, υλοποιείται μέσω μιας δομής πίνακα κατακερματισμού (hash-table) η οποία απλώνεται στην αργή και γρήγορη μνήμη του συστήματος. Ο πίνακας κατακερματισμού, αποθηκεύεται στην γρήγορη και ακριβή μνήμη του συστήματος, αντιστοιχίζοντας ένα κατακερματισμένο αναγνωριστικό σε ένα δείκτη, ο οποίος δείχνει στα δεδομένα του πακέτου στην αργή, αλλά φτηνή μνήμη του συστήματος [33], [38]. Οι περισσότεροι σχεδιασμοί πολιτικών cache υποθέτουν ότι έχουν chunks μεγέθους 1500 byte και εγγραφές LRU μεγέθους τουλάχιστον 32 byte [38]. Όταν η αναλογία καταχωρίσεων μεταξύ γρήγορης και αργής μνήμης είναι ένα προς ένα, το γεγονός αυτό οδηγεί στην απαίτηση ο λόγος (ratio) μεταξύ του μεγέθους της γρήγορης και αργής μνήμης, να είναι περίπου ίσος με 1:46.

Το μεγαλύτερο μέγεθος single-chip SRAM μνήμης που βρέθηκε σε τρέχοντες δρομολογητές είναι 210 Mbits [39], επομένως μπορεί να δεικτοδοτήσει σχεδόν 1.2 Gbytes από chunks μεγέθους 1500 byte (περίπου 800K chunks). Η DRAM μνήμη ενός δρομολογητή έχει μέγεθος 10 Gbytes[2], και άρα το 88%

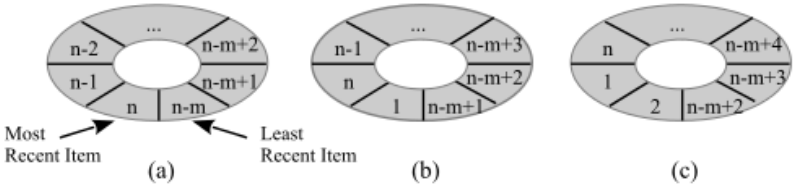
[2] Indexing potential size = (SRAM_size/LRU_entry)*chunks_size = (((210×(10^6))÷8)÷(32×8))×(1500×8) ~= 1.2 Gbytes

του διαθέσιμου αποθηκευτικού χώρου του δικτύου δεν μπορεί να καταχωρηθεί στο packet-level. Μια λύση για το πρόβλημα αυτό θα ήταν να μεγαλώσει το μέγεθος των chunks, έτσι ώστε οι hardware προδιαγραφές των δρομολογητών να μην επηρεάζουν την απόδοση του caching. Αυτή η λύση όμως κρίθηκε ανέφικτη, αφού όχι μόνο θα επέβαλλε σοβαρές κυρώσεις στο caching granularity [27], [40], αλλά θα απαιτούσε και την αλλαγή της Maximum Transfer Unit (MTU) έτσι ώστε να καταφέρει να διατηρήσει την αυτο-αναγνώριση (self-identification) των δικτυακών μονάδων.

Μία άλλη λύση θα ήταν να χρησιμοποιηθεί η DRAM για ευρετηρίαση των αποθηκευμένων πακέτων. Ωστόσο, αυτός ο σχεδιασμός απαιτεί μία ανάγνωση της αργής μνήμης για κάθε εισερχόμενο αίτημα, ακόμα και χωρίς cache hits, καθιστώντας έτσι αμφισβητήσιμη την wire-speed λειτουργία.

4.2.2 Κυκλική αντικατάσταση

Αντίθετα με τις object-caches, οι packet-caches ανάλογα με την πολιτική αντικατάστασης και το μοτίβο πρόσβασης (access pattern), μπορεί να περιέχουν μόνο ένα κομμάτι ενός αντικειμένου. Το γεγονός αυτό μπορεί να αποδειχθεί τόσο ευεργετικό όσο και καταστροφικό ανάλογα με την περίσταση. Στις περισσότερες εφαρμογές τα πακέτα ενός αντικειμένου ζητούνται σε αύξουσα ακολουθιακή σειρά, το οποίο σημαίνει ότι σε μια cache με LRU πολιτική αντικατάστασης, τα πρώτα πακέτα του αντικειμένου διαγράφονται από την cache πριν τα τελευταία, εφόσον υπάρχουν αποθηκευμένα για περισσότερο χρόνο μέσα στην μνήμη. Ας φανταστούμε λοιπόν ένα παράδειγμα με ένα αντικείμενο το οποίο περιέχει n πακέτα, και μια cache μνήμη η οποία μπορεί κάθε στιγμή να αποθηκεύσει m πακέτα, όπου n>m. Στο παράδειγμα αυτό, μια object-cache δεν θα μπορούσε να κάνει cache ολόκληρο το αντικείμενο, ενώ μια packet-cache θα μπορούσε να κάνει cache κάποια από τα πακέτα του αντικειμένου. Ωστόσο, αν τα πακέτα του αντικειμένου εξετάζονται σε ακολουθιακή σειρά, τότε μετά από την αποθήκευση των πρώτων m πακέτων στην cache, το m+1-οστό πακέτο θα αντικαταστήσει το 1ο πακέτο, το m+2-οστό το 2ο και ούτω καθεξής, μέχρι να ολοκληρωθεί η μεταφορά του αντικειμένου (σχήμα Εικ. 3(a)). Στην συνέχεια όταν το αντικείμενο ζητηθεί ξανά, το 1ο πακέτο δεν θα βρεθεί στην cache και άρα θα ανακτηθεί από την πηγή, αντικαθιστώντας το πακέτο του αντικειμένου που χρησιμοποιήθηκε λιγότερο πρόσφατα. Η διαδικασία αυτή θα επαναλαμβάνεται συνέχεια μέχρι να ανακτηθεί ολόκληρο το αντικείμενο ξανά, χωρίς να έχουμε κανένα cache hit (σχήμα Εικ.3(b) και (c)). Το φαινόμενο αυτό ονομάζεται κυκλική αντικατάσταση (looped replacement) και μπορεί να εμφανιστεί σε οποιαδήποτε μεγέθους cache, εφόσον χρησιμοποιείται η LRU πολιτική αντικατάστασης, δεδομένου ότι τα πακέτα του αντικειμένου εξετάζονται ακολουθιακά και τα αιτήματα για το ίδιο αντικείμενο δεν είναι τόσο συχνά.



Εικ. 3: Μία LRU cache χωρητικότητας m πακέτων, εμφανιζόμενη σαν ένας κυκλικός ενταμιευτής (circular buffer). Στο (a) μόλις ανακτήθηκε ένα αντικείμενο με n πακέτα (n > m), στο (b) και (c) το 1ο και 2ο πακέτο του ίδιου αντικειμένου, ζητούνται ξανά.
Πηγή: Object-oriented Packet Caching for ICN

4.2.3 Παρεμπόδιση του ελέγχου συμφόρησης μέσω RTT

Μια δεύτερη αδυναμία που παρουσιάζουν οι packet caches είναι ότι μπορούν να προκαλέσουν σημαντική βλάβη σε πρωτόκολλα ελέγχου κίνησης τα οποία χρησιμοποιούν τα RTTs σαν δείκτες συμφόρησης. Όταν στην cache αποθηκεύονται τυχαία πακέτα περιεχομένου, τότε εμφανίζεται μια σημαντική διακύμανση στο RTT, η οποία εξαρτάται από το αν ένα συγκεκριμένο πακέτο που ζητήθηκε έφτασε από την cache ή από την αρχική πηγή. Οι διακυμάνσεις αυτές προκαλούν ψευδή λήξεις χρονομέτρων (timeouts) και μειώνουν την απόδοση του επιπέδου μεταφοράς (transport layer). Αυτό το φαινόμενο δεν εμφανίζεται στις object-caches, διότι σε αυτές τις αρχιτεκτονικές, η κρυφή μνήμη είτε κρατάει ολόκληρο το αντικείμενο, είτε το παίρνει από την αρχική πηγή, χωρίς έτσι να προκαλούνται διακυμάνσεις στο RTT. Οι συγγραφείς στην έρευνα [2] βρίσκουν ότι περίπου το 2-3% των Web αιτήσεων περιεχομένου ματαιώνονται πριν ολοκληρωθεί πλήρως η λήψη τους, με τον συνολικό όγκο των ληφθέντων bytes μέχρι την ματαίωση να αγγίζει σε μερικές περιπτώσεις, περίπου το 30% ολόκληρης της κίνησης. Επομένως σε μια packet-level cache προσέγγιση, περιμένουμε διάφορα αντικείμενα να βρίσκονται μερικώς αποθηκευμένα μέσα σε κάθε cache, προκαλώντας έτσι τέτοια ζητήματα που σχετίζονται με το RTT.

4.2.4 Δηλητηρίαση από μεγάλα αντικείμενα

Μια άλλη σημαντική πρόκληση που καλούνται να αντιμετωπίσουν οι μικρές in-network caches είναι η διαχείριση μεγάλων, αλλά μη δημοφιλών αντικειμένων. Μια οποιαδήποτε σχεδίαση cache, είτε σε επίπεδο πακέτου, είτε σε επίπεδο αντικειμένου που χρησιμοποιεί LRU πολιτική αντικατάστασης, αποθηκεύει όλα τα chunks κάθε αντικειμένου ανεξάρτητα από το πόσο δημοφιλή είναι. Αυτό μπορεί να μειώσει σημαντικά την απόδοση της cache, ειδικά σε περιπτώσεις όπου έχουμε μεγάλα αρχεία τα οποία απασχολούν ένα σημαντικό ποσό χώρου της μνήμης. Το φαινόμενο αυτό προκαλεί σπατάλη σημαντικών πόρων της cache, αφού εκείνη αποθηκεύει περιεχόμενα τα οποία δεν προσφέρουν στην πραγματικότητα αληθινό κέρδος, εφόσον ζητούνται πάρα πολύ σπάνια.

4.3 Σχεδίαση του OPC

Για να μπορέσουμε να αντιμετωπίσουμε τις παραπάνω αδυναμίες που παρουσιάζουν οι προσεγγίσεις caching που βασίζονται σε επίπεδο πακέτου, σχεδιάσαμε το Object Oriented Packet Caching ή OPC [41], αρχιτεκτονική η οποία συνδυάζει την αποτελεσματικότητα του packet-level caching με την αποδοτικότητα όσον αφορά την διαχείριση πόρων που παρουσιάζει το object-level caching. Ο σχεδιασμός του OPC εστιάζει και αντιμετωπίζει αποτελεσματικά τις αδυναμίες των ICN packet caches. Συγκεκριμένα, αυξάνει την χρησιμοποίηση μνήμης, βοηθάει στον έλεγχο συμφόρησης που βασίζεται στο RTT και εμποδίζει την ύπαρξη μεγάλων αρχείων τα οποία ως μη δημοφιλή, μειώνουν την απόδοση της cache όπως περιγράψαμε στην ενότητα 4.2.4. Η καινοτομία έγκειται στο γεγονός ότι το OPC κατορθώνει να επιτύχει τους παραπάνω στόχους χωρίς να απαιτείται περισσότερη μνήμη ή υπολογιστικοί πόροι από αυτούς που απαιτεί μια συνηθισμένη LRU cache.

Το OPC συνδυάζει επίσης την αποτελεσματικότητα των cache αναζητήσεων των object-level caches χρησιμοποιώντας πολιτική αντικατάστασης σε επίπεδο πακέτου. Βασισμένοι στην παρατήρηση ότι οι περισσότερες εφαρμογές ζητούν τα πακέτα ενός αντικειμένου που φέρει μια πληροφορία σε αύξουσα ακολουθιακή σειρά, επιλέξαμε στο OPC να αποθηκεύεται στην κρυφή μνήμη πάντα το αρχικό κομμάτι

ενός αντικειμένου, από το 1ο μέχρι το n-οστό πακέτο χωρίς να υπάρχουν κενά. Επομένως οποιαδήποτε μερικώς αποθηκευμένα στην κρυφή μνήμη αντικείμενα, αντιπροσωπεύονται πάντα από τα πρώτα n πακέτα τους.

Το ευρετήριο αναζήτησης στο OPC περιέχει το όνομα του αντικειμένου και έναν μετρητή (counter) για κάθε αποθηκευμένο αντικείμενο. Ο μετρητής αυτός αναπαριστά τον αριθμό των chunks που έχουν αποθηκευτεί στην κρυφή μνήμη. Λόγω της ακολουθιακής σειράς των πακέτων, αυτός ο μετρητής λέγεται ή είναι και last chunk id, δηλώνοντας έτσι το αναγνωριστικό του τελευταίου πακέτου του αντικειμένου που έχει αποθηκευτεί στην κρυφή μνήμη. Επομένως μπορούμε να καταλάβουμε, ότι αν για παράδειγμα έχουμε την εγγραφή *file/a, 45* τότε αυτό σημαίνει ότι η cache κρατάει αποθηκευμένα τα πρώτα 45 chunks του αντικειμένου file/a χωρίς κενά ενδιάμεσα. Αν φτάσει ένα αίτημα για ένα αντικείμενο με αριθμό σειράς ή rank μικρότερο ή ίσο με το last chunk id, τότε αυτό σημαίνει ότι η cache έχει αποθηκευμένο αυτό το πακέτο και μπορεί να εξυπηρετήσει κατευθείαν το αίτημα αυτό. Σε αντίθετη περίπτωση, όταν δηλαδή φτάσει ένα αίτημα με υψηλότερο chunk rank από το last chunk id, τότε η cache προωθεί το αίτημα στον πραγματικό προορισμό έτσι ώστε να εξυπηρετηθεί από εκεί. Η τεχνική αυτή μειώνει το κόστος δεικτοδότησης σε μία εγγραφή για κάθε (μερικώς) αποθηκευμένο αρχείο, ή περίπου κατά μέσο όρο τόσες φορές λιγότερο όσο είναι το μέγεθος του αρχείου σε σύγκριση με μία συνηθισμένη LRU packet cache.

4.4 Αρχιτεκτονική και δομές δεδομένων του OPC

Ένας κόμβος OPC διατηρεί δύο δομές δεδομένων για εισαγωγές και αναζητήσεις chunks, και μία για διαγραφή. Οι δύο πρώτες δομές ονομάζονται Layer 1 (L1) και Layer 2 (L2) ευρετήρια (βλ. Εικ. 4) και οργανώνουν τα δεδομένα σε object-level και cache-level αντίστοιχα. Πιο συγκεκριμένα, το ευρετήριο L1 αποθηκεύεται στην γρήγορη μνήμη (π.χ SRAM) και υλοποιείται ως ένας πίνακας κατακερματισμού (hash table) σταθερού μεγέθους, με μία εγγραφή για κάθε αντικείμενο αποθηκευμένο στην cache. Κάθε εγγραφή στο L1 αντιστοιχίζει ένα αναγνωριστικό περιεχομένου με ένα ζευγάρι από τιμές: το rank/order του τελευταίου αποθηκευμένου chunk (last chunk id) του συγκεκριμένου αντικειμένου και έναν pointer ο οποίος δείχνει στο τελευταίο chunk του αντικειμένου που βρίσκεται αποθηκευμένο στο L2 ευρετήριο (Ptr_{mem}). Το ευρετήριο L2 από την άλλη, αποτελεί ουσιαστικά έναν πίνακα στην αργή μνήμη (π.χ DRAM) ο οποίος περιέχει τα αποθηκευμένα στην cache chunks κάθε αντικειμένου σε ακολουθιακή σειρά.

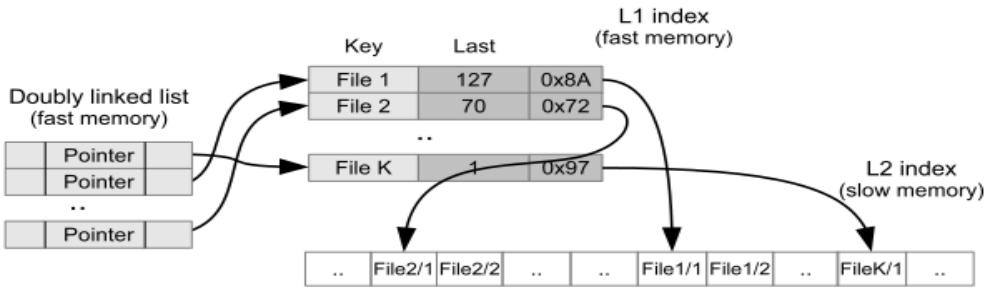
Μόλις το OPC λάβει ένα αίτημα για ένα chunk πληροφορίας αντικειμένου, τότε χρησιμοποιεί το αναγνωριστικό που βρίσκει στην επικεφαλίδα του αιτήματος για το συγκεκριμένο chunk και μέσω του ευρετηρίου L1 ελέγχει να δει αν υπάρχουν αποθηκευμένα στην cache chunks για το αντικείμενο αυτό. Αν βρεθούν τέτοια chunks και η αναζήτηση επιστρέψει ένα last chunk id μεγαλύτερο ή ίσο από το rank/order του ζητούμενου chunk, τότε το chunk αυτό μπορεί να ανακτηθεί κατευθείαν από την μνήμη της cache και συγκεκριμένα από την διεύθυνση $Ptr_{mem} - (\text{chunk id} - \text{id}) * \text{MSS}$, όπου το id είναι το rank/order του ζητούμενου chunk και το MSS είναι το maximum segment size που μπορεί να έχει ένα chunk δεδομένων. Διαφορετικά, αν δηλαδή το last chunk id είναι μικρότερο από το rank του ζητούμενου chunk, τότε το αίτημα προωθείται στον αρχικό προορισμό του και τα δεδομένα ανακτούνται από εκεί. Στο σημείο αυτό πρέπει να σημειώσουμε ότι ο πίνακας μνήμης χρησιμοποιεί τόσα bytes ανά

chunk όσο το μέγεθος του MSS. Αυτό συμβαίνει, έτσι ώστε να έχουμε γρηγορότερες αναζητήσεις, ανεξάρτητα από το μέγεθος των chunks.

Όταν ληφθεί ένα καινούριο chunk δεδομένων, πάλι με τη βοήθεια του L1 index, βλέπουμε αν το αντικείμενο για το chunk αυτό είναι αποθηκευμένο στην cache και το chunk αποτελεί το ακριβώς επόμενο στην ακολουθία. Αν αυτό συμβαίνει, τότε το αποθηκεύουμε στο ευρετήριο L1 και αυξάνουμε το last chunk id κατά 1. Αν τώρα το αντικείμενο δεν είναι αποθηκευμένο και το chunk είναι το πρώτο chunk του αντικειμένου, τότε αποθηκεύουμε το chunk στο ευρετήριο L2 και δημιουργούμε μια καινούρια εγγραφή στο ευρετήριο L1, με το last chunk id να είναι ίσο με 1 και το Ptr_{mem} να δείχνει στο chunk που αποθηκεύτηκε στο L2. Σε διαφορετική περίπτωση αγνοούμε το chunk.

Η τρίτη δομή δεδομένων στο OPC είναι μια διπλά συνδεδεμένη λίστα η οποία χρησιμοποιείται για να κατατάξει τα αντικείμενα ανάλογα με την πολιτική αντικατάστασης που υιοθετείται. Η λίστα αυτή επίσης αποθηκεύεται στην γρήγορη μνήμη και δείχνει, το λιγότερο σημαντικό (ανάλογα με την εκάστοτε πολιτική) αντικείμενο που βρίσκεται αποθηκευμένο στην OPC cache. Το αντικείμενο αυτό απομακρύνεται όταν η cache είναι γεμάτη και χρειάζεται να απελευθερωθεί έξτρα χώρος. Στην δικιά μας υλοποίηση η πολιτική αντικατάστασης που χρησιμοποιείται είναι η LRU και επομένως τα αντικείμενα κατατάσσονται από την διπλά συνδεδεμένη λίστα ανάλογα με το ποιο έχει χρησιμοποιηθεί πιο πρόσφατα, απομακρύνοντας έτσι από την cache τα αντικείμενα εκείνα που έχουν να χρησιμοποιηθούν περισσότερο καιρό. Ωστόσο ο σχεδιασμός της OPC δεν περιορίζεται μόνο σε μια πολιτική αντικατάστασης και έτσι τα αποθηκευμένα περιεχόμενα μπορούν να οργανωθούν είτε με LRU, LFU ή FIFO δομή. Αν η απομάκρυνση ενός αντικειμένου από την cache πρέπει να γίνει λόγω έλλειψης χώρου στο ευρετήριο L1, τότε ανακτάται τόσο η εγγραφή στο L1, όσο και όλα τα chunks στα οποία εκείνη δείχνει και βρίσκονται στο ευρετήριο L2. Αν η απομάκρυνση γίνεται λόγω έλλειψης χώρου στο L2 ευρετήριο, τότε ανακτάται μόνο το τελευταίο chunk της επιλεγμένης εγγραφής και η εγγραφή L1 ενημερώνεται με την μείωση του last chunk id κατά 1.

Παρακάτω βλέπουμε ένα στιγμιότυπο των δομών που χρησιμοποιεί η σχεδίαση OPC.



Εικ. 4: Δομές δεδομένων του OPC
Πηγή: Object-oriented Packet Caching for ICN

4.5 Αλγόριθμος του OPC

Για να βεβαιώσουμε ότι το OPC κρατάει πάντα αποθηκευμένο το αρχικό κομμάτι ενός αρχείου, εισάγουμε έναν καινοτόμο αλγόριθμο αντικατάστασης πακέτων:

Το OPC εισάγει ένα chunk με rank/order i στην cache μνήμη, εάν είναι είτε το 1ο chunk του αντικειμένου, είτε αν υπάρχει ήδη αποθηκευμένο το (i – 1)-οστό (δηλαδή το προηγούμενο) chunk. Στην πρώτη περίπτωση, δηλαδή όταν το chunk i είναι το πρώτο chunk του αντικειμένου, δημιουργείται μια νέα εγγραφή ευρετηρίου για το αντίστοιχο περιεχόμενο. Η δεύτερη περίπτωση, δηλαδή ότι έχουμε ήδη αποθηκευμένο το chunk i – 1 για ένα συγκεκριμένο αντικείμενο σημαίνει ότι το last chunk id για το αντικείμενο αυτό είναι ίσο με i – 1. Η ιδιότητα αυτή εγγυάται ότι σε οποιαδήποτε χρονική στιγμή, η cache θα κρατάει πάντα αποθηκευμένο το πρώτο κομμάτι κάθε αντικειμένου, χωρίς να υπάρχουν κενά ενδιάμεσα. Αν δεν υπάρχει χώρος στην αργή μνήμη για την αποθήκευση ενός νέου chunk, τότε χρησιμοποιείται μια object-level LRU λίστα η οποία επιλέγει ένα αντικείμενο στην cache και διαγράφει το τελευταίο chunk που είχε αποθηκευτεί σε αυτήν, έτσι ώστε και πάλι η cache μνήμη να κρατάει αποθηκευμένα τα πρώτα chunks ενός αντικειμένου χωρίς κενά ενδιάμεσα. Αν τώρα δεν υπάρχει χώρος στην γρήγορη μνήμη για ένα νέο αντικείμενο, τότε η εγγραφή ευρετηρίου για το αντικείμενο αυτό στην ουρά/τέλος της LRU διαγράφεται μαζί με τα αντίστοιχα chunks του αντικειμένου στην slow μνήμη.[3]

4.6 Αντιμετώπιση των αδυναμιών

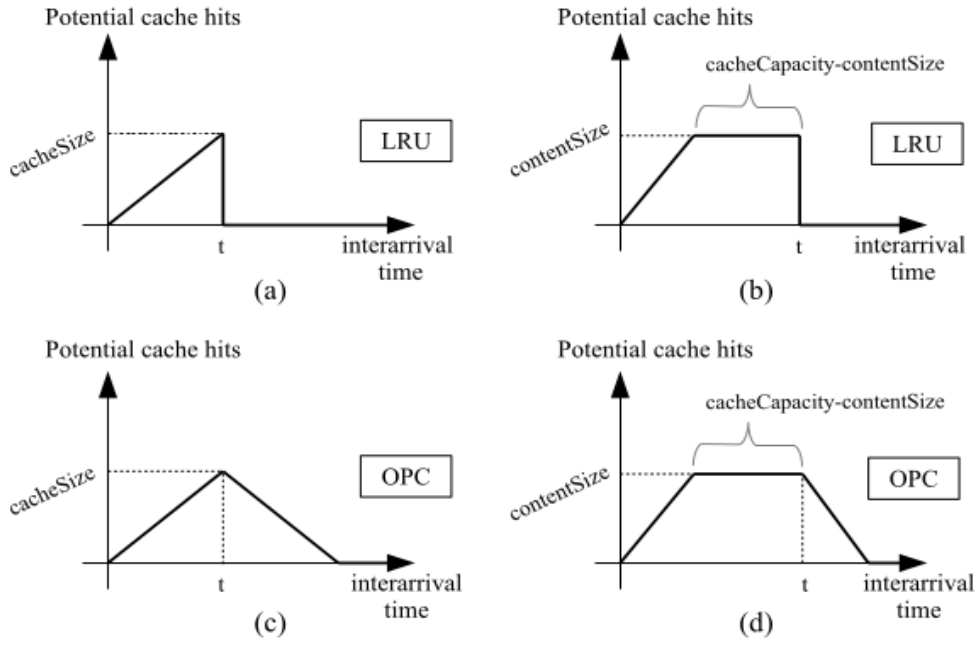
Σε αυτήν την ενότητα θα εξηγήσουμε πως το OPC αντιμετωπίζει αποτελεσματικά τις αδυναμίες που περιγράφτηκαν στην ενότητα 4.2.

Καταρχήν η ευρετηριακή δομή δύο επιπέδων του OPC βελτιστοποιεί την χρήση τόσο της γρήγορης όσο και της αργής μνήμης. Το ευρετήριο L1, που βρίσκεται όπως είπαμε στην γρήγορη μνήμη, χρησιμοποιεί μια εγγραφή για κάθε αντικείμενο, έναντι της μιας εγγραφής για κάθε chunk που χρησιμοποιούν οι προηγούμενες ICN packet-cached αρχιτεκτονικές. Έτσι λοιπόν, ως αποτέλεσμα το L1 ευρετήριο είναι μικρού μεγέθους, γεγονός που το κάνει να χωράει να αποθηκευτεί στην μικρή, αλλά γρήγορη μνήμη, επιταχύνοντας έτσι τις αναζητήσεις. Επίσης με αυτόν τον τρόπο αυξάνεται σημαντικά ο όγκος των δεδομένων που μπορεί να δεικτοδοτηθεί στο L2, σε σύγκριση με απλούστερες τεχνικές όπως LRU και FIFO, λύνοντας έτσι το πρόβλημα των περιορισμένων αποθηκευτικών πόρων που περιγράψαμε στην ενότητα 4.2.1.

Κατά δεύτερον, για να αποφύγει την εμφάνιση του φαινομένου της κυκλικής αντικατάστασης (ενότητα 4.2.2), το OPC κρατάει πάντα αποθηκευμένα μόνο τα πρώτα chunks ενός αντικειμένου, εισάγοντάς τα με αύξουσα ακολουθιακή σειρά και απομακρύνοντάς τα με την αντίθετη σειρά. Η λογική αυτή λειτουργεί βάση της υπόθεσης ότι τα chunks ζητούνται σε αύξουσα ακολουθιακή σειρά (όπως και στην περίπτωση της έρευνας [36]) και έτσι καταφέρνει και επεκτείνει τον χρόνο μέσα στον οποίο μπορούμε να εκμεταλλευτούμε ένα αντικείμενο που υπάρχει αποθηκευμένο στην cache. Αυτό αυτομάτως σημαίνει υψηλότερο cache hit rate. Για να γίνει αυτό περισσότερο κατανοητό, στην Εικ. 5 παρουσιάζουμε τα cache hits που μπορούμε να έχουμε για δύο αιτήσεις του ίδιου αντικειμένου (y-axis) σε μια LRU και σε μια OPC cache αντίστοιχα, σε σχέση με τον χρόνο άφιξης μεταξύ των δύο αυτών αιτημάτων (x-

[3] Ο πίνακας hash μπορεί να χρησιμοποιήσει linear probing, double hashing, ή οποιαδήποτε άλλη τεχνική η οποία δεν απαιτεί επιπρόσθετη μνήμη για να μπορεί να διαχειριστεί τυχόν συγκρούσεις.

axis). Κατά γενική ομολογία, καθώς τα chunks ζητούνται ακολουθιακά, ο αριθμός αυτών που μπορούν να αποθηκευτούν στην cache αυξάνεται, άρα μεγαλώνει και η πιθανότητα για περισσότερα cache hits. Στα σχήματα Εικ. 5 (a), (c), το μέγεθος της cache είναι μικρότερο από το μέγεθος του αντικειμένου και επομένως από το σημείο που η cache γεμίσει (χρόνος t), δεν μπορούμε να έχουμε περισσότερα cache hits. Ωστόσο, σε μια LRU cache (Εικ .5 (a)), το φαινόμενο της κυκλικής αντικατάστασης προκαλεί την αντικατάσταση των πρώτων chunks του αντικειμένου από τα νέα chunks που ζητήθηκαν, καταλήγοντας έτσι απευθείας σε μηδενικά cache hits, δεδομένου μιας νέας ακολουθιακής αίτησης για το ίδιο αντικείμενο. Αντίθετα, στο OPC (Εικ. 5 (c)), τα chunks απομακρύνονται από την cache μόνο από το τέλος του αντικειμένου, κρατώντας τα πρώτα ακόμα μέσα σε αυτήν. Επομένως η δυνατότητα για cache hits μειώνεται σταδιακά, μέχρι να αντικατασταθούν όλα τα chunks, χωρίς να έχουμε μία πτώση απευθείας στο 0. Παρόμοια συμπεριφορά έχουμε επίσης όταν το μέγεθος της cache είναι μεγαλύτερο από το μέγεθος του αντικειμένου (Εικ. .5 (b), (d)).Στο στιγμιότυπο αυτό, αφού έχει αποθηκευτεί στην cache ολόκληρο το αντικείμενο, η δυνατότητα για cache hits παραμένει σταθερή. Όταν την χρονική στιγμή t, τα chunks αρχίσουν να απομακρύνονται από την cache, στην περίπτωση της LRU (Εικ. 5 (b)) έχουμε μια απότομη πτώση στο 0, αφού έχουν διαγραφεί τα πρώτα chunks του αντικειμένου, ενώ στην περίπτωση του OPC (Εικ. .5 (d)) βλέπουμε μια σταδιακή μείωση.



Εικ. 5: Δυνητικά cache-hits 2 αιτήσεων για το ίδιο αντικείμενο σε μία LRU και σε μία OPC cache. Στο (a) και (c) το content size ξεπερνά το cache size, ενώ το αντιθετο ισχύει για τα (b) και (d).
Πηγή: Object-oriented Packet Caching for ICN

Τρίτον, το OPC αντιμετωπίζει το πρόβλημα του ελέγχου συμφόρησης μέσω RTT (ενότητα 4.2.3), μειώνοντας τις διακυμάνσεις του RTT οι οποίες συμβαίνουν λόγω των cache απαντήσεων που παίρνουμε στην περίπτωση αιτήσεων για τυχαία αποθηκευμένα πακέτα περιεχομένου. Το OPC πετυχαίνει αυτή την μείωση, αποθηκεύοντας πάντα το αρχικό κομμάτι του αντικειμένου, δηλαδή τα πρώτα chunks, δια-

χωρίζοντας έτσι την μετάδοση του αντικειμένου αυτού σε δύο διακριτές φάσεις. Στην πρώτη φάση, τα chunks δεδομένων λαμβάνονται από την cache, και έτσι ο RTT estimator αντανακλά την κατάσταση συμφόρησης που υπάρχει στο μονοπάτι από την cache μέχρι τον παραλήπτη. Αφού χρησιμοποιηθούν όλα τα chunks που βρίσκονται αποθηκευμένα στην cache, τα υπόλοιπα θα ληφθούν από την αρχική πηγή περιεχομένου, προκαλώντας πιθανότατα μια λήξη χρονομέτρου (time-out). Αυτό το timeout σηματοδοτεί την μετάβαση στην δεύτερη φάση, με την έναρξη της οποίας ο RTT estimator θα προσαρμόσει την κατάσταση συμφόρησης του μονοπατιού ανάμεσα στον παραλήπτη και την αρχική πηγή. Επομένως για να ανακεφαλαιώσουμε, το OPC ενισχύει και βελτιώνει τον ρυθμό μετάδοσης, αποστέλλοντας cached απαντήσεις κατά την πρώτη φάση, χωρίς να διαταράσσει τα time-outs ούτε στην πρώτη ούτε στην δεύτερη φάση.

Τέλος το OPC αντιμετωπίζει το φαινόμενο της δηλητηρίασης από μεγάλα αντικείμενα (ενότητα 4.2.4) εφαρμόζοντας φιλτράρισμα επιπέδου αντικειμένου σε στατιστικά δημοτικότητας. Πιο συγκεκριμένα, ένα L1 object-level ευρετήριο ακολουθεί την LRU πολιτική, εισάγοντας ένα αντικείμενο στην κορυφή/κεφάλι της LRU λίστας όταν έχουμε εισαγωγή νέου chunk ενός αντικειμένου, ή όταν συμβαίνει cache hit για οποιοδήποτε από τα chunks του αντικειμένου αυτού. Σε αντίθεση λοιπόν με μια packet-level LRU, κάθε εισαγόμενο chunk τοποθετείται στην κορυφή της LRU λίστας, και το ίδιο γίνεται για κάθε hit που συμβαίνει και αφορά το συγκεκριμένο αντικείμενο. Συνεπώς, στο OPC, η απομάκρυνση ενός αντικειμένου από την cache, εξαρτάται από την δημοτικότητά αυτού του αντικειμένου ως ολότητα, ενώ σε οποιεσδήποτε packet-cache αρχιτεκτονικές βασισμένες σε LRU πολιτική αντικατάστασης, τα πολλά ξεχωριστά chunks ενός αντικειμένου γεμίζουν την LRU λίστα, δυσκολεύοντας έτσι την συγκράτηση των δημοφιλών αντικειμένων μέσα στην cache. Με τον τρόπο αυτόν το OPC επιτυγχάνει περισσότερα cache hits αφού αποθηκεύει τα chunks που ζητούνται περισσότερο.

5 Προσομοιωτής NS3

5.1 Σύντομη επισκόπηση

Το mobile multimedia lab (mmlab) του Οικονομικού Πανεπιστημίου Αθηνών, έχει υλοποιήσει τα τελευταία χρόνια μία προσομοίωση ενός NDN δικτύου. Την υλοποίηση αυτή την χρησιμοποιήσαμε για να προσομοιώσουμε και να αξιολογήσουμε την OPC πολιτική μας, σε σχέση με άλλες πολιτικές (π.χ LRU όπως θα δούμε παρακάτω στα πειράματα. Ο κορμός της υλοποίησης βασίστηκε στον NS3 Simulator[4], ο οποίος αποτελεί έναν προσομοιωτή δικτύου γραμμένο στην γλώσσα C++, που χρησιμοποιείται κυρίως για ερευνητικούς και εκπαιδευτικούς λόγους. Στόχος του ns-3 project, είναι η ανάπτυξη ενός κατάλληλου, ανάλογα με τις εκάστοτε ανάγκες, περιβάλλοντος προσομοίωσης για έρευνα πάνω σε δίκτυα.

Παρακάτω παρουσιάζεται η υλοποίηση του project με use-case UML διαγράμματα τα οποία απεικονίζουν την διάρθρωση και λειτουργία του κώδικα που αναπτύχθηκε για την NDN λειτουργικότητα και τα caching policies πάνω στα οποία τρέχουμε τα πειράματά μας.

5.2 Υλοποίηση

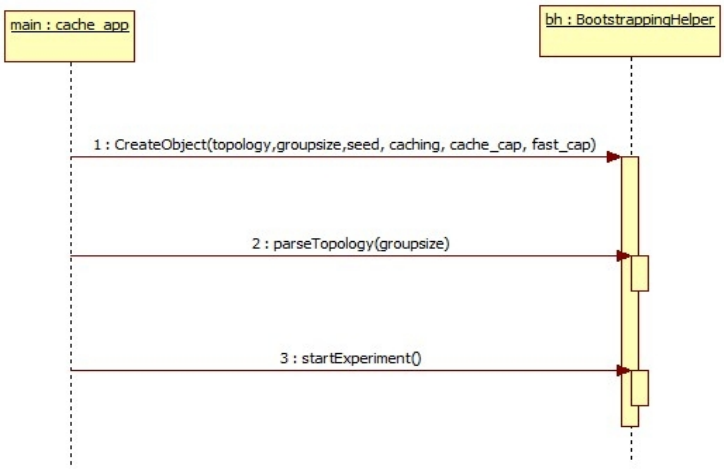
Η υλοποίηση του project ξεκινά από την συνάρτηση main που βρίσκεται στο αρχείο cache_app.cc. Αρχικά λοιπόν όπως μπορούμε να δούμε (Εικ. 6), δημιουργείται ένα αντικείμενο της τάξης BootStrappingHelper με παραμέτρους:

- το αρχείο της τοπολογίας δικτύου που χρησιμοποιούμε (topology),
- το groupsize, το οποίο είναι μια παράμετρος η οποία δείχνει τον αριθμό των host nodes που θα εισαχθούν σε κάθε access node,
- το seed, το οποίο είναι μια παράμετρος που χρησιμοποιεί ο ns-3 simulator έτσι ώστε για κάθε εκτέλεση του προγράμματος να λαμβάνουμε πανομοιότυπα αποτελέσματα έτσι ώστε να μπορούμε να τα συγκρίνουμε,
- και τέλος οι χωρητικότητες της αργής (cache_cap) και γρήγορης (fast_cap) μνήμης αντίστοιχα.

Στη συνέχεια καλείται η μέθοδος parseTopology του BootstrappingHelper η οποία διαβάζει το αρχείο της τοπολογίας και δημιουργεί ένα ns-3 δίκτυο. Έπειτα προσθέτει ένα CCN module σε κάθε κόμβο του δικτύου. Τέλος καλείται η μέθοδος startExpirement η οποία είναι η μέθοδος που ξεκινά το πείραμα.

Στην παρούσα ενότητα παρουσιάζουμε μόνο το αρχικό διάγραμμα (cache_app). Τα υπόλοιπα διαγράμματα προστέθηκαν στο Παράρτημα όπου μπορεί κάποιος να δει την πλήρη λειτουργία του κώδικα της εφαρμογής από την αρχή μέχρι το τέλος του πειράματος.

[4] Available at http://www.nsnam.org/



Εικ. 6: Cache_app
Πηγή: Προσωπική δημιουργία

6 Πειράματα

6.1 Προεργασία

Στην παρούσα διπλωματική εργασία υλοποιήσαμε την λειτουργικότητα ενός CCN/NDN δικτύου, καθώς και την LRU πολιτική cache σε packet-level. Επίσης υλοποιήσαμε την δικιά μας προσέγγιση (OPC) με σκοπό την σύγκριση μεταξύ των δύο πολιτικών. Η υλοποίηση πραγματοποιήθηκε χρησιμοποιώντας τον NS-3 network simulator [42].

Συνοπτικά, εξετάσαμε 10 scale-free τοπολογίες μεγέθους 100 κόμβων η κάθε μία, οι οποίες δημιουργήθηκαν χρησιμοποιώντας την μέθοδο των A.-L. Barabasi και R. Albert, που παρουσιάζεται στην έρευνα [43]. Αποφύγαμε να εξετάσουμε πως συμπεριφέρεται η πολιτική μας σχετικά με τις επιπτώσεις του RTT-based congestion control, υποθέτωντας ένα πρωτόκολλο παύσης και αναμονής (stop-and-wait protocol), έτσι ώστε να έχουμε πιο ξεκάθαρη και σαφέστερη εικόνα για τα υπόλοιπα σημεία αξιολόγησης.

Χρησιμοποιήσαμε το GlobeTraff traffic trace generator [44], για παραγωγή και χρησιμοποίηση ρεαλιστικού workload, με αρχεία μεταβλητού μεγέθους, διαφορετικού τύπου κίνησης (Web, P2P, Video, Other) και διαφορετικών τιμών δημοφιλίας. Το workload μοιράστηκε τυχαία στους receivers της εφαρμογής, οι οποίοι άρχισαν όλοι να αιτούνται περιεχόμενο ταυτόχρονα. Τα χαρακτηριστικά του workload φαίνονται στον πίνακα Πιν. 1.

		Web	P2P	Video	Other	Total
#Objects		195386	1	176	10485	206048
#Chunks	Total	1513081	687167	1328168	41940	3570356
	median	6	687168	8133	4	
	max	19929	687167	16997	5120	
	std. dev	56.6	0	5261.2	0	
#Requests	Total	658686	2	326	22352	681366
	max	10984	2	17	1106	
	std. dev	53.8	0	2.33	15.3	

Πιν. 1: Χαρακτηριστικά του Workload
Πηγή: Προσωπική δημιουργία

Συγκρίναμε την απόδοση μεταξύ της πολιτικής LRU και του OPC για 3 διαφορετικές πολιτικές τοποθέτησης της cache χρησιμοποιώντας παραμέτρους χωρητικότητας για κάθε μία από τις δύο μνήμες (SRAM, DRAM) τις τιμές που φαίνονται στον πίνακα Πιν. 2. Οι τιμές αυτές αντιπροσωπεύουν το 10% του μεγέθους του workload. Ο λόγος που χρησιμοποιήσαμε 10% του workload δεν είναι τυχαίος. Σύμφωνα με την έρευνα [35], η πολιτική τοποθέτησης ανάλογα το betweeness [35] δεν λειτουργεί καλά με μικρότερα μεγέθη cache. Έτσι λοιπόν αποφασίσαμε να ακολουθήσουμε το δικό τους πειραματικό setup. Παρόλα αυτά όπως θα δούμε αργότερα η OPC πολιτική δεν παρουσιάζει τόσο καλά αποτελέσματα με

τόσο μεγάλη χωρητικότητα. Αντίθετα δείχνει αυξημένη απόδοση με χωρητικότητες πολύ μικρότερες, της τάξης του 0.1% του workload.

Στον επόμενο πίνακα βλέπουμε τις παραμέτρους χωρητικότητας των δύο μνημών που χρησιμοποιήθηκαν στα πειράματα για κάθε πολιτική. Ακολουθώντας τα χαρακτηριστικά του hardware των μνημών που παρουσιάζονται στην έρευνα [39], υποθέσαμε ότι η μεγαλύτερη χωρητικότητα για έναν caching δρομολογητή περιλαμβάνει 210 Mbits SRAM μνήμης και 10 GB DRAM μνήμης. Επίσης, υποθέσαμε πως έχουμε chunks μεγέθους 1500 byte και επιπλέον πως κάθε εγγραφή LRU έχει μέγεθος 40 bytes [34], [35], [39], ενώ κάθε εγγραφή στην γρήγορη μνήμη του OPC χρειάζεται δύο bytes επιπλέον, δηλαδή 42 bytes. Αυτό πρακτικά σημαίνει ότι η LRU μπορεί να ευρετηριάσει μέχρι και 688,128 αντικείμενα στην γρήγορη μνήμη, ενώ το OPC 655,360. Ωστόσο στην LRU απαιτείται έχουμε μια εγγραφή ευρετηρίου για κάθε πακέτο και επομένως η αναλογία μεταξύ γρήγορης και αργής μνήμης είναι 1:1. Αντίθετα στο OPC έχουμε μια εγγραφή ευρετηρίου για πολλά πακέτα, καταλήγοντας σύμφωνα με τα παραπάνω μεγέθη των SRAM, DRAM, σε μια αναλογία περίπου 1:11, (δηλαδή 1 εγγραφή για κάθε 11 chunks).

	LRU	OPC
SRAM (fast)	357036	340034
DRAM (slow)	357036	3706369

Πιν. 2: Cache Capacities Parameters
Πηγή: Προσωπική δημιουργία

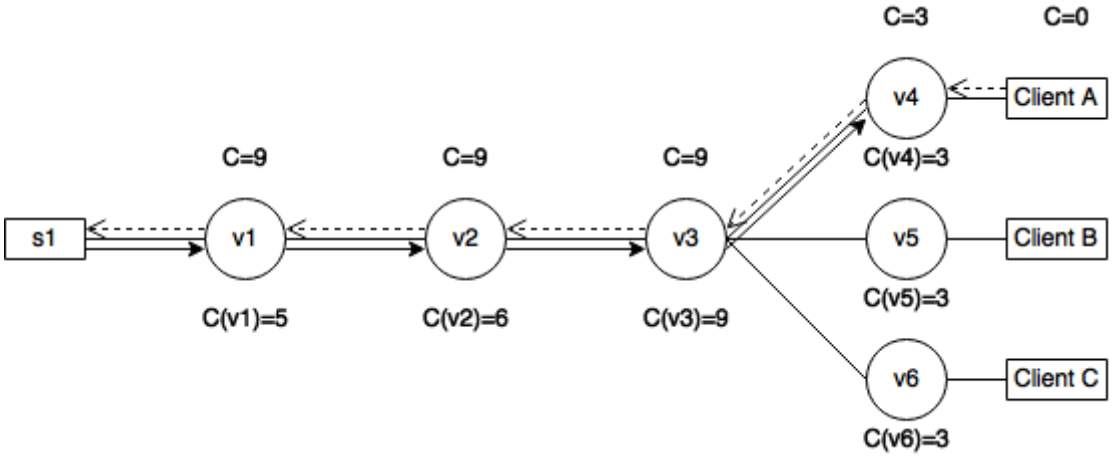
6.1.1 Τοποθέτηση των caches

Όπως αναφέραμε και προηγουμένως, εξετάζουμε 3 διαφορετικές περιπτώσεις σχετικά με το που θα βάλουμε caches πάνω στο δίκτυό μας.

1. *Universal caching*:
   Στην πρώτη περίπτωση η ιδέα είναι να κάνουμε caching παντού, δηλαδή σε κάθε κόμβο του δικτύου, χρησιμοποιώντας έτσι όλους τους διαθέσιμους πόρους που μας προσφέρει ένα ICN δίκτυο.
2. *Caching in edge nodes*:
   Η δεύτερη περίπτωση, αποτελεί μια διαφορετική προσέγγιση [34] η οποία υποστηρίζει πως δεν είναι απαραίτητο να κάνουμε caching παντού, αλλά μόνο στους κόμβους οι οποίοι συνδέονται με τους χρήστες και ονομάζονται edge nodes ή access nodes. Η ιδέα αυτή βασίζεται στο γεγονός ότι το μεγαλύτερο μέρος της κίνησης του δικτύου αφορά Web traffic requests. Δεδομένου ότι, όπως γνωρίζουμε, το Web traffic ακολουθεί μία κατανομή Zipf, είναι πολύ πιθανότερο να έχουμε περισσότερα cache hits στους κόμβους αυτούς, αντίθετα με τους υπόλοιπους κόμβους που βρίσκονται στον κορμό του δικτύου (core nodes).
3. *Caching based in Betweeness-Centrality*:
   Η τρίτη και τελευταία περίπτωση πάλι βασίζεται στην ιδέα ότι με λιγότερες caches σε επιλεγμένους κόμβους μπορούμε να πετύχουμε τουλάχιστον ίδια αποτελέσματα όσον αφορά την απόδοση, χωρίς το κόστος που απαιτεί το universal caching. Συγκεκριμένα, όπως παρουσιάζεται στην έρευνα [35], τα πακέτα αποθηκεύονται σε caches που βρίσκονται σε κόμβους με το

υψηλότερο βαθμό centrality. Ο βαθμός αυτός, δείχνει ουσιαστικά πόσο κεντρικός είναι ένας συγκεκριμένος κόμβος και αναλυτικότερα αντιπροσωπεύει πόσες φορές ο κόμβος αυτός βρίσκεται πάνω στο μονοπάτι διανομής περιεχομένου μεταξύ όλων των ζευγαριών κόμβων που υπάρχουν στο δίκτυο. Επομένως λοιπόν, είναι πιο πιθανό να έχουμε περισσότερα cache-hits σε κόμβους με υψηλό βαθμό centrality.

Στην παρακάτω εικόνα (Εικ. 7) μπορούμε να δούμε τον αλγόριθμο που χρησιμοποιείται στην έρευνα [35], ο οποίος βρίσκει τον κόμβο/-ους με το υψηλότερο centrality degree. Στον αλγόριθμο αυτόν, όταν για παράδειγμα ο Client A ζητάει ένα κομμάτι περιεχομένου από τον s1 προωθεί το αίτημα στον επόμενο κόμβο εισάγοντας στην επικεφαλίδα του το centrality degree που στην αρχή έχει την τιμή 0.Εκεί συγκρίνει την τιμή αυτή με το centrality του κόμβου (στην συγκεκριμένη περίπτωση τον v4). Αν το centrality του κόμβου C(v4) είναι μεγαλύτερο από το centrality C στην επικεφαλίδα του request τότε κάνει το C ίσο με την τιμή του C(v4), δηλαδή C=3. Αν είναι μικρότερο ή ίσο δεν αλλάζει την τιμή. Και στις δύο περιπτώσεις, προωθεί στην συνέχεια το αίτημα στον επόμενο κόμβο και επαναλαμβάνεται η ίδια διαδικασία. Έτσι λοιπόν στο συγκεκριμένο παράδειγμα βρήκαμε ότι το on-path centrality degree είναι το 9. Επομένως κατά την μετάδοση των δεδομένων από τον s1, πάλι συγκρίνεται το on-path centrality degree με το centrality του κάθε κόμβου μέχρι να βρεθεί ο κόμβος ο οποίος έχει centrality degree ίσο με το on-path centrality degree. Αυτός στο συγκεκριμένο παράδειγμα είναι ο v9 και επομένως εκεί θα αποθηκευτεί στην cache το πακέτο δεδομένων.



Εικ. 7: Αλγόριθμος Betweeness Centrality
Το διακεκκομένο βέλος αναπαριστά την λειτουργία προώθησης όταν ο Client A αιτείται ένα περιεχόμενο στον s1.
Το κανονικό βέλος αναπαριστά την μεταφορά δεδομένων από τον s1 στον ClientA.
Πηγή: Προσωπική δημιουργία

6.2 Αξιολόγηση

Στα πειράματά μας συγκρίναμε την LRU πολιτική με το OPC ως προς:

α) το cache-hit ratio, δηλαδή τον λόγο των cache-hits ως προς τα συνολικά cache requests.

β) το network load (φόρτος δικτύου), δηλαδή τον λόγο των INTEREST μηνυμάτων που στέλνονται συνολικά στο δίκτυο όταν δεν έχουμε cache προς τα INTEREST μηνύματα που στέλνονται όταν έχουμε cache.

γ) και το server load (φόρτος εξυπηρετητή), δηλαδή τον λόγο των INTEREST μηνυμάτων που λαμβάνει ο server/publisher όταν δεν έχουμε cache προς τα INTEREST μηνύματα που λαμβάνει όταν έχουμε cache.

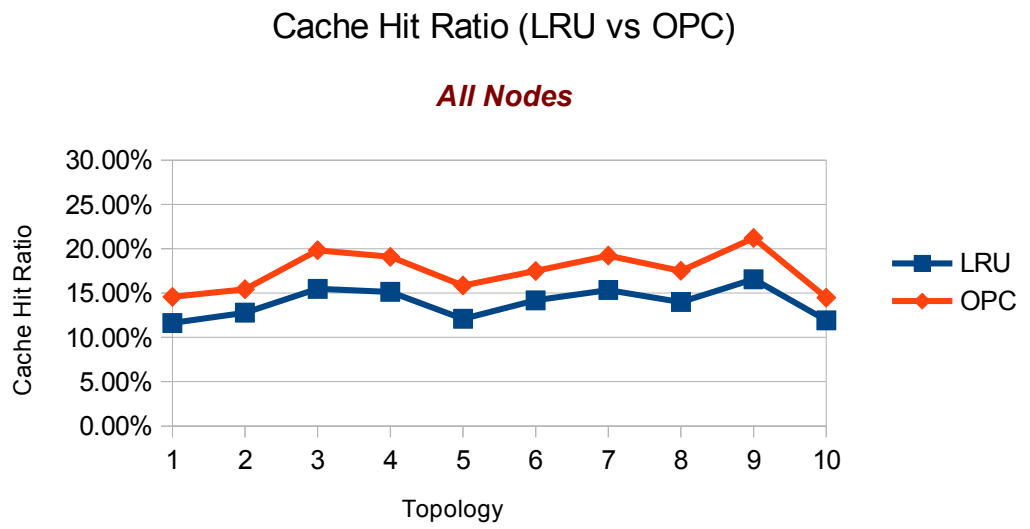
Τα πειράματα αυτά επαναλήφθηκαν για κάθε μία από τις 3 περιπτώσεις τοποθέτησης των caches (Universal caching, Caching in edge nodes, Caching in nodes based on Centrality Betweeness). Όπως βλέπουμε στα επόμενα διαγράμματα (Εικ. 8 – Εικ. 16) το OPC παρουσιάζει καλύτερη απόδοση από το LRU τόσο όσον αφορά το cache-hit ratio, όσο και για τον φόρτο δικτύου και εξυπηρετητή, εισάγοντας caches σε όλους τους κόμβους.

Όταν η τοποθέτηση των caches γίνεται στους κόμβους με υψηλό centrality degree (Betweeness centrality), τότε βλέπουμε πως το OPC παρουσιάζει στις περισσότερες των περιπτώσεων καλύτερη απόδοση από το LRU. Παρόλα αυτά η απόδοση των δύο τεχνικών είναι αρκετά κοντά. Ειδικά σε κάποιες τοπολογίες και συγκεκριμένα στην 5 και στην 9 βλέπουμε ότι η απόδοση του OPC και του LRU συμπίπτουν. Το γεγονός αυτό εξηγείται από την μορφή των τοπολογιών αυτών, οι οποίες είναι "φαρδιές" - ουσιαστικά αντίθετες από την μορφή ενός δέντρου. Για τις scale-free τοπολογίες αυτό σημαίνει ότι έχουν μεγαλύτερο power law. Σε τέτοιες περιπτώσεις η απόδοση του OPC μειώνεται τόσο στα cache-hits όσο και στον φόρτο δικτύου και εξυπηρετητή.

Όταν η τοποθέτηση των caches γίνεται στους edge nodes τότε βλέπουμε πως το LRU και το OPC έχουν ακριβώς την ίδια απόδοση. Αυτό οφείλεται στο γεγονός ότι η παράμετρος 10% του μεγέθους του workload για την χωρητικότητα της cache αποτελεί ένα αρκετά μεγάλο ποσοστό. Στα πειράματά μας το workload μοιράζεται τυχαία στους edge nodes. Αυτό σημαίνει ότι κάθε τέτοιος κόμβος έχει ένα κομμάτι και όχι ολόκληρο το workload. Ας φανταστούμε επίσης ότι έχουμε για παράδειγμα 10 edge nodes. Τότε η χωρητικότητα των caches στους edge nodes επαρκεί για το LRU έτσι ώστε να καλύψει όλο το workload. Αυτό πρακτικά σημαίνει ότι το LRU υλοποιείται με βέλτιστο τρόπο εφόσον όλα τα requests χωράνε να αποθηκευτούν στο ευρετήριο της γρήγορης μνήμης της LRU. Έτσι λοιπόν όταν οι caches έχουν μέγεθος ένα ποσοστό τέτοιου μεγέθους, το OPC και το LRU έχουν ίδια απόδοση. Το OPC κερδίζει κατά πολύ, όταν χρησιμοποιούνται μικρότερα μεγέθη cache, της τάξης του 0.1%. Παρόλα αυτά εξηγήσαμε ήδη γιατί αποφασίσαμε να χρησιμοποιήσουμε το 10% στην ενότητα 6.1. Για να αποδείξουμε πως το παραπάνω επιχείρημά μας στέκει, πραγματοποιήσαμε και πειράματα με cache στους edge nodes με χωρητικότητα cache 0.1% του workload. Στα σχήματα Εικ. 17, Εικ. 18, Εικ. 19, βλέπουμε πράγματι πως με παράμετρο 0.1% το OPC παρουσιάζει πολύ καλύτερα αποτελέσματα τόσο στο cache-hit ratio όσο και στο network και server load. Συγκεκριμένα το OPC επιτυγχάνει ένα cache-hit ratio μεγαλύτερο του 20% και για τις 10 τοπολογίες, την ίδια στιγμή που το LRU επιτυγχάνει γύρω στο 12%. Επίσης το OPC επιτυγχάνει περίπου 14% μείωση στον φόρτο του δικτύου και 16% μείωση στον φόρτο του εξυπη-

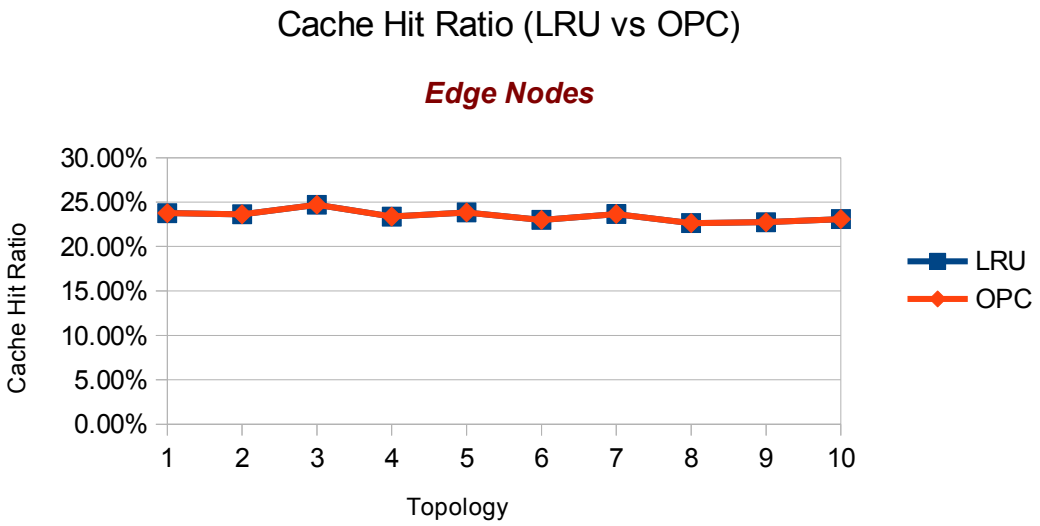
ρετητή, ενώ το LRU επιτυγχάνει περίπου 7% μείωση και για τα δύο. Αυτό πρακτικά σημαίνει ότι λαμβάνουμε περίπου 200% κέρδος όταν χρησιμοποιείται η τεχνική του OPC.

Παρακάτω παρουσιάζονται τα διαγράμματα για κάθε μία από τις περιπτώσεις που αναλύθηκαν παραπάνω:



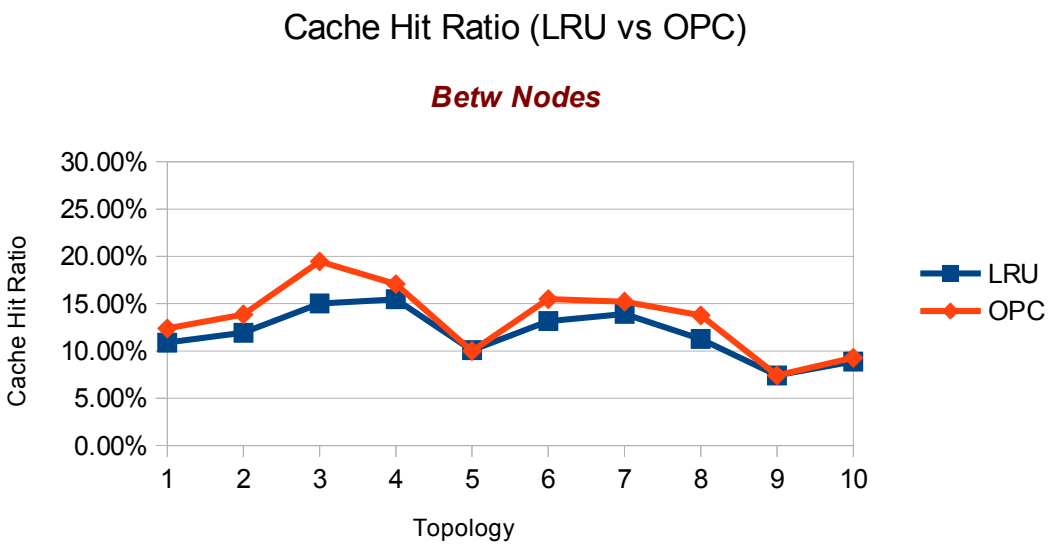
Εικ. 8:Cache-Hit Ratio - LRU vs OPC

Τοποθέτηση cache:All nodes
Πηγή: Προσωπική δημιουργία



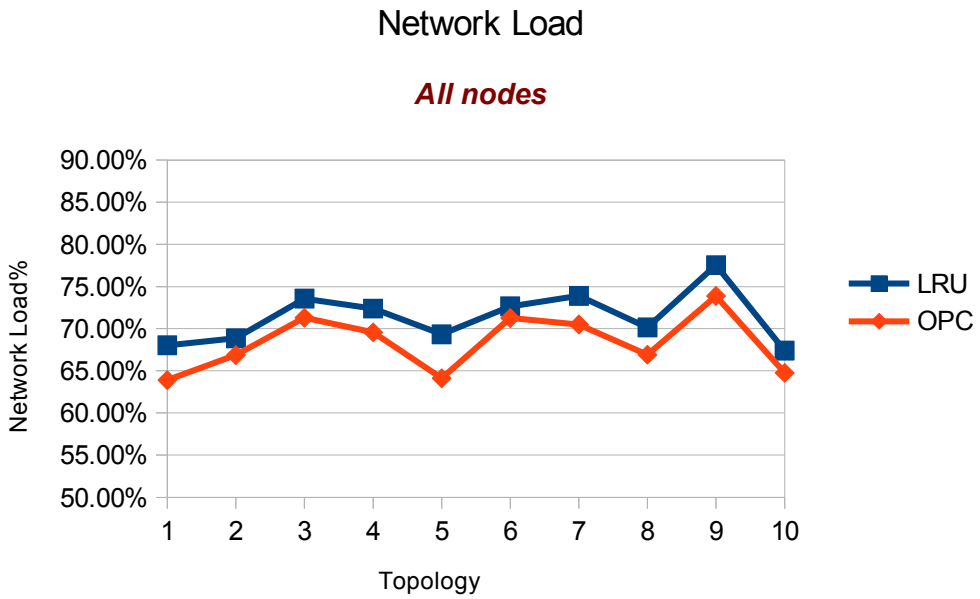
Εικ. 9:Cache-Hit Ratio - LRU vs OPC

Τοποθέτηση cache:Edge nodes
Πηγή: Προσωπική δημιουργία



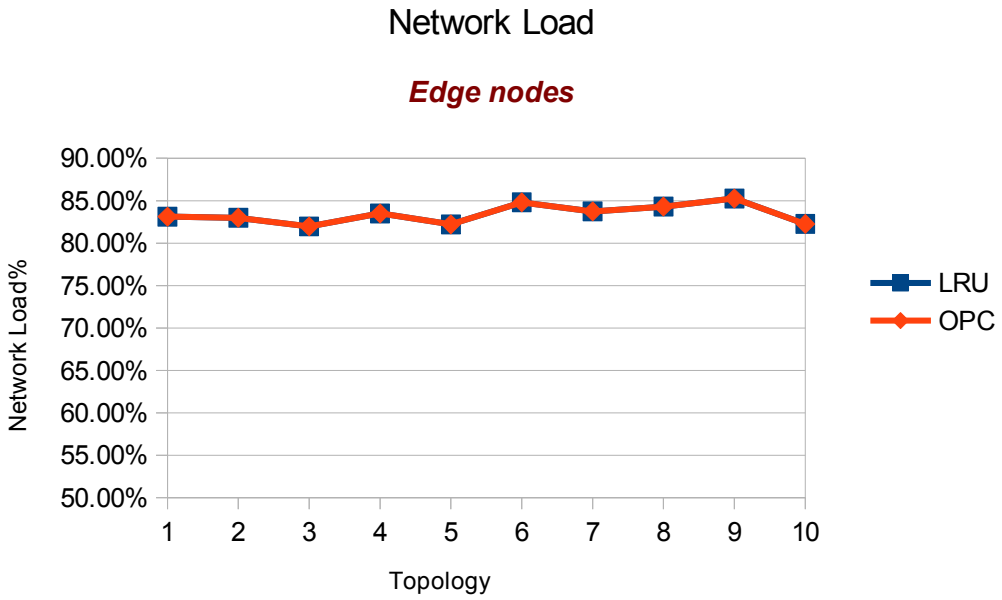
Εικ. 10:Cache-Hit Ratio - LRU vs OPC

Τοποθέτηση cache:Betweeness nodes
Πηγή: Προσωπική δημιουργία



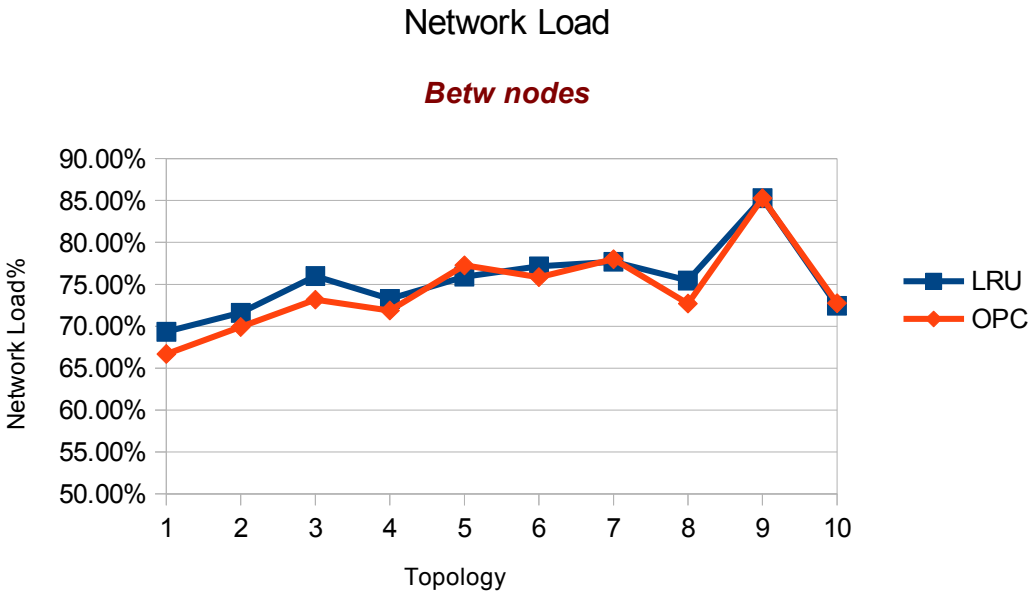
Εικ. 11:Network Load - LRU vs OPC

Τοποθέτηση cache:All nodes
Πηγή: Προσωπική δημιουργία



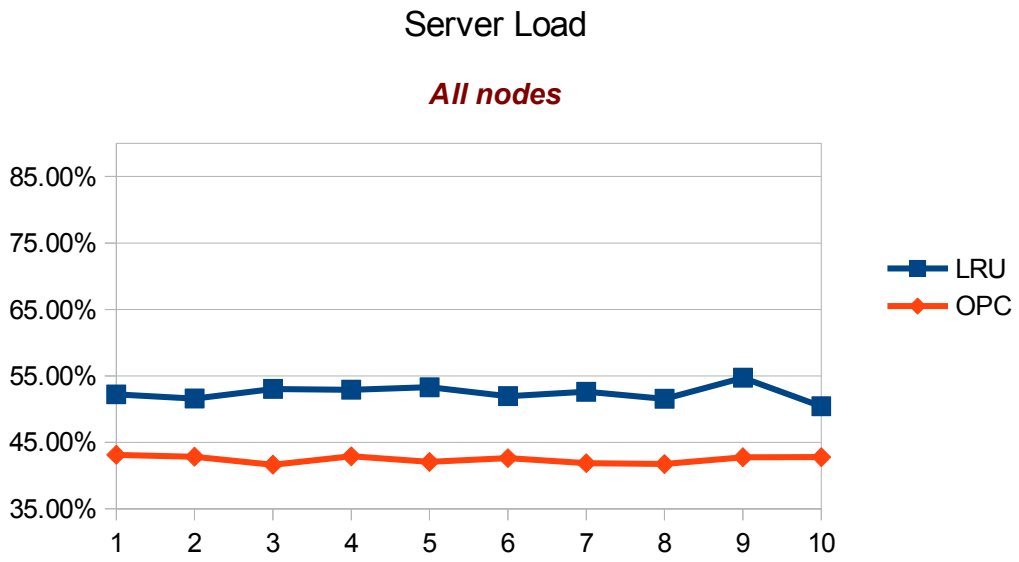
Εικ. 12:Network Load - LRU vs OPC

Τοποθέτηση cache:Edge nodes
Πηγή: Προσωπική δημιουργία



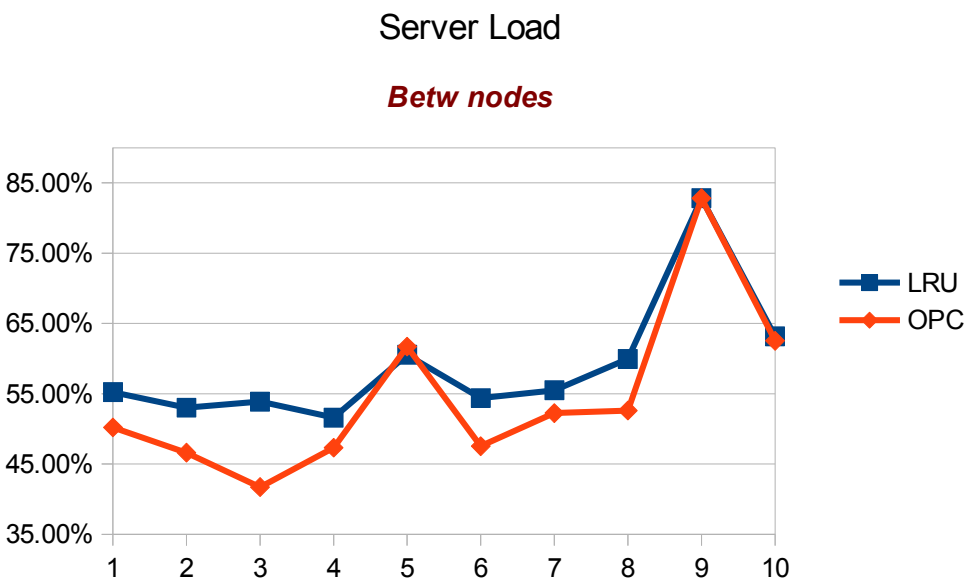
Εικ. 13:Network Load - LRU vs OPC

Τοποθέτηση cache:Betweeness nodes
Πηγή: Προσωπική δημιουργία



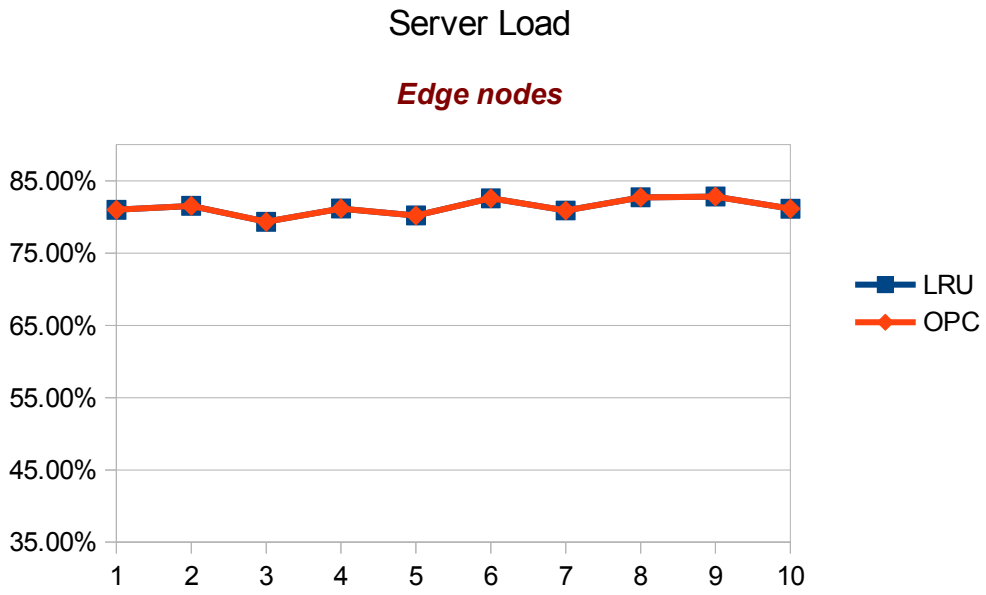
Εικ. 14:Server Load - LRU vs OPC

Τοποθέτηση cache:All nodes
Πηγή: Προσωπική δημιουργία



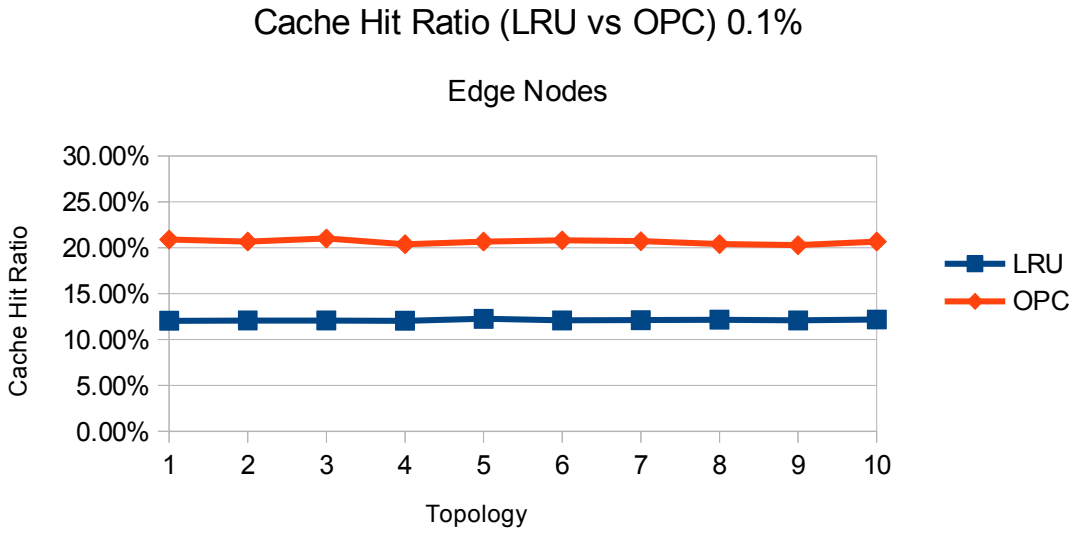
Εικ. 15:Server Load - LRU vs OPC

Τοποθέτηση cache:Betweeness nodes
Πηγή: Προσωπική δημιουργία



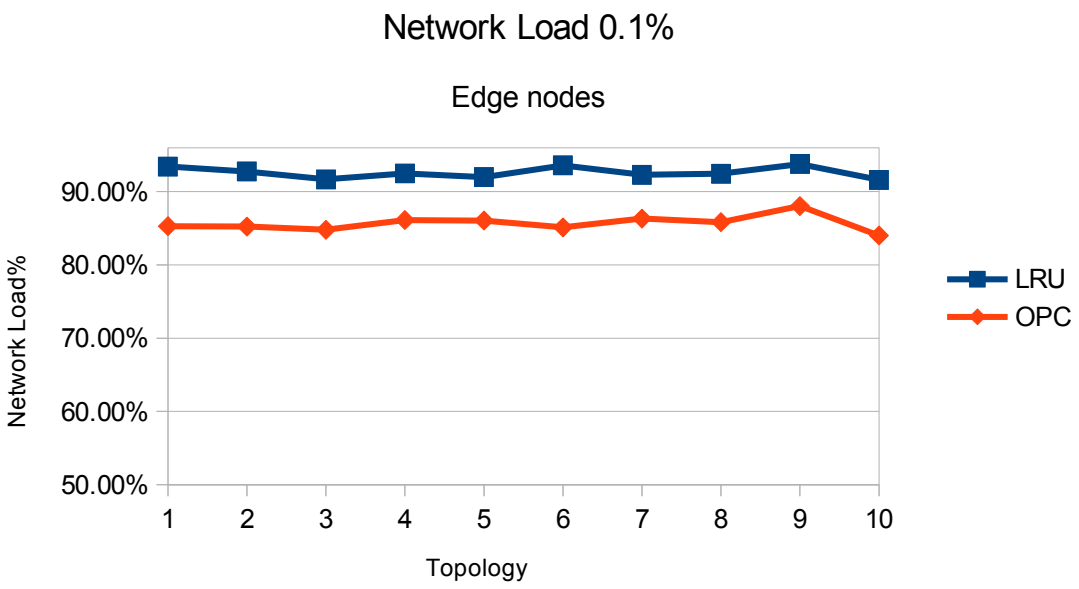
Εικ. 16:Server Load - LRU vs OPC

Τοποθέτηση cache:Edge nodes
Πηγή: Προσωπική δημιουργία



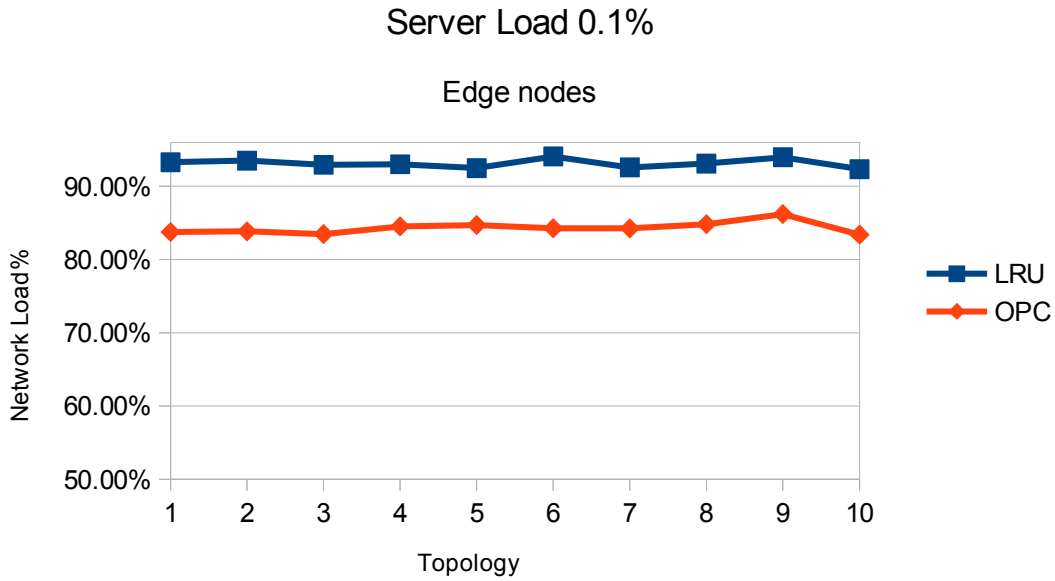
Εικ. 17:Cache-hit Ratio - LRU vs OPC (cache size=0.1%)

Τοποθέτηση cache:Edge nodes
Πηγή: Προσωπική δημιουργία



Εικ. 18:Network Load - LRU vs OPC (cache size=0.1%)

Τοποθέτηση cache:Edge nodes
Πηγή: Προσωπική δημιουργία



Εικ. 19:Server Load - LRU vs OPC (cache size=0.1%)

Τοποθέτηση cache:Edge nodes
Πηγή: Προσωπική δημιουργία

7 Συμπεράσματα

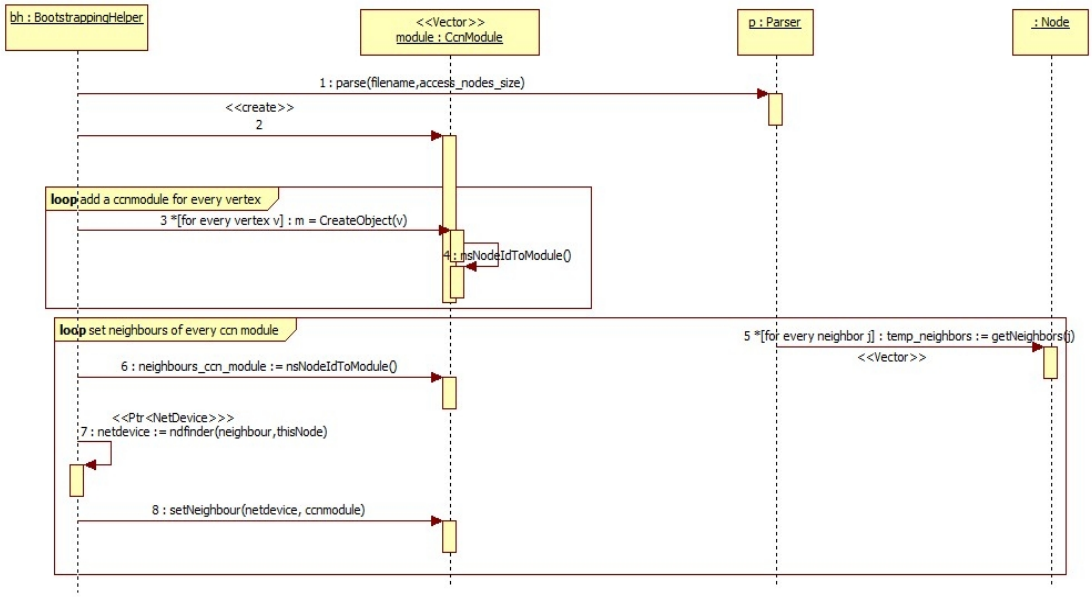
Στην παρούσα διπλωματική εργασία, παρουσιάσαμε μία καινοτόμα πολιτική caching για τα δίκτυα ICN. Το Object-oriented Packet Caching (OPC), με την σχεδίαση των δύο επιπέδων του, καταφέρνει να εκμεταλλευτεί αποτελεσματικά τόσο την γρήγορη όσο και την αργή μνήμη των δρομολογητών για caching. Παρουσιάσαμε τις αδυναμίες των συνηθισμένων packet-level caches στα ICN δίκτυα και ταυτόχρονα το κίνητρο και τους στόχους που θέσαμε για την σχεδίαση αυτής της αρχιτεκτονικής. Αναλύσαμε πως το OPC μπορεί να αυξήσει τον αποθηκευτικό χώρο σε επίπεδο πακέτου (packet-level) ο οποίος περιορίζεται από την μικρή σε μέγεθος SRAM μνήμη, όπως επίσης και την βελτίωση της αποτελεσματικότητας του caching. Επίσης εξηγήσαμε πως το OPC δεν εμποδίζει την λειτουργία του RTT-based congestion control. Επιπρόσθετα, αναγνωρίζοντας το looped replacement effect (ενότητα 4.2.2) και το large object poisoning (ενότητα 4.2.4) σαν δύο σημαντικά προβλήματα που αντιμετωπίζουν οι packet-level caches, παρουσιάσαμε έναν αποτελεσματικό αλγόριθμο για αναζήτηση, εισαγωγή και απομάκρυνση πακέτων από την cache. Στην συνέχεια αξιολογήσαμε την απόδοση του OPC σε σχέση με μια LRU packet-level cache με την προσομοίωση ενός NDN δικτύου, χρησιμοποιώντας scale-free τοπολογίες και ρεαλιστικό workload, για 3 διαφορετικές περιπτώσεις τοποθέτησης των caches. Από τα πειράματα συμπεράναμε πως το OPC επιτυγχάνει καλύτερα αποτελέσματα από την LRU packet-level cache, ειδικά για μικρά μεγέθη χωρητικότητας των caches, τόσο μειώνοντας τον δικτυακό φόρτο και τον φόρτο του εξυπηρετητή, όσο και αυξάνοντας το cache-hit ratio.

8 Παράρτημα

8.1 NS-3 Simulator – Code sequence diagrams

8.1.1 Μέθοδος ParseTopology

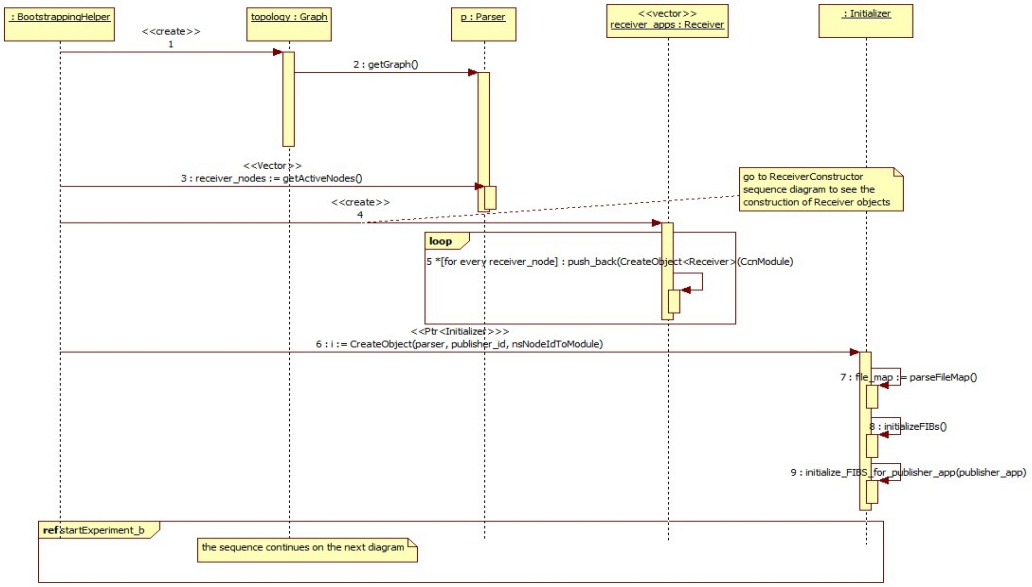
Η μέθοδος αυτή διαβάζει το αρχείο της τοπολογίας και δημιουργεί ένα ns-3 δίκτυο. Έπειτα προσθέτει ένα CCN module σε κάθε κόμβο του δικτύου, και τέλος ορίζει τους γείτονες κάθε CCN module. Το CCN module είναι το αντίστοιχο του Content Router (CR), στην NDN αρχιτεκτονική η οποία αναλύεται στην ενότητα 2.2..



Εικ. 20: ParseTopology
Πηγή: Προσωπική δημιουργία

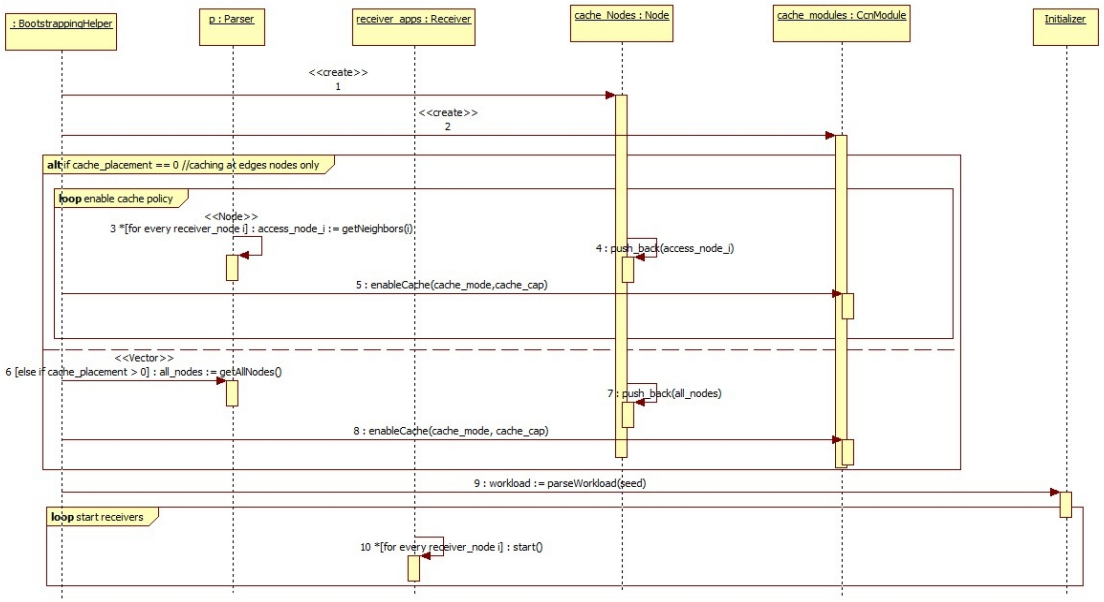
8.1.2 Μέθοδος StartExpirement

Η μέθοδος StartExpirement είναι η μέθοδος με την οποία ξεκινάει η λειτουργία της εφαρμογής. Αρχικά δημιουργεί ένα Vector με τους κόμβους οι οποίοι είναι οι παραλήπτες (receivers) από το αντικείμενο topology[5]. Έπειτα για κάθε κόμβο receiver δημιουργεί και εισάγει ένα CCN module τύπου Receiver[6] και



Εικ. 21: StartExpirement (a)
Πηγή: Προσωπική δημιουργία

έπειτα αρχικοποιεί το Forwarding Interest Base (FIB). Στο 2ο διάγραμμα (Εικ. 22) ενεργοποιείται η cache και εισάγεται ανάλογα με την εκάστοτε πολιτική cache placement.



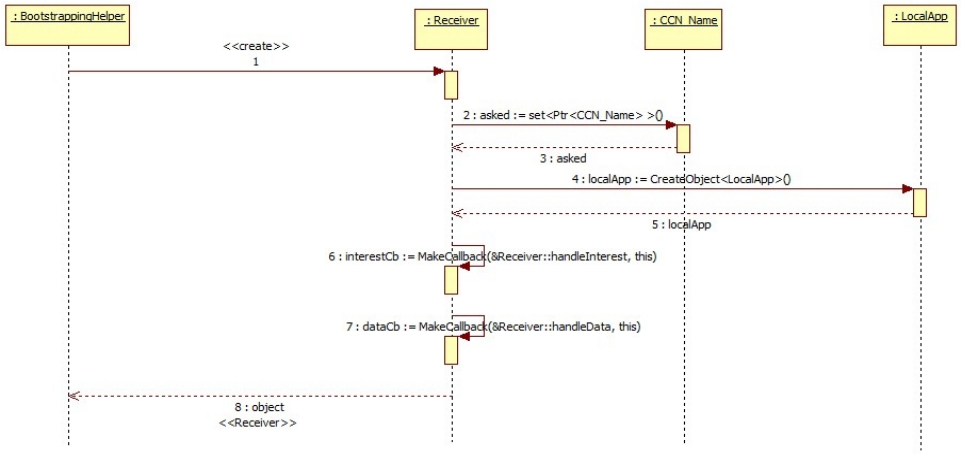
Εικ. 22: StartExpirement (b)
Πηγή: Προσωπική δημιουργία

[5]Το αντικείμενο topology έχει ήδη δημιουργηθεί από την μέθοδο ParseTopology, αλλά περιλαμβάνεται για ευκολότερη κατανόη - ση.

[6]Η κατασκευή ενός Receiver αντικειμένου περιγράφεται παρακάτω σε διαφορετικό διάγραμμα.

8.1.3 Μέθοδος Receiver Constructor

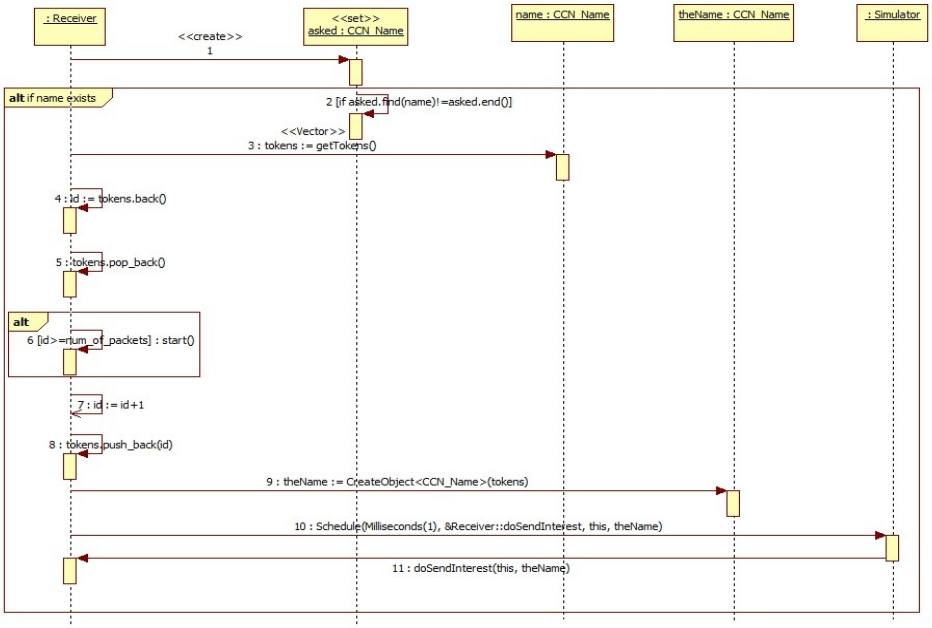
Εδώ βλέπουμε τον constructor για τα αντικείμενα τύπου Receiver ο οποίος ουσιαστικά ορίζει τα INTEREST και τα DATA μηνύματα. Τα μηνύματα INTEREST προφανώς δεν λαμβάνονται στους Receivers. Τα μηνύματα DATA αναφέρονται στα πραγματικά δεδομένα που αντιστοιχούν σε INTEREST μηνύματα για μια συγκεκριμένη πληροφορία.



Εικ. 23: Receiver Constructor
Πηγή: Προσωπική δημιουργία

8.1.4 Μέθοδος HandleData

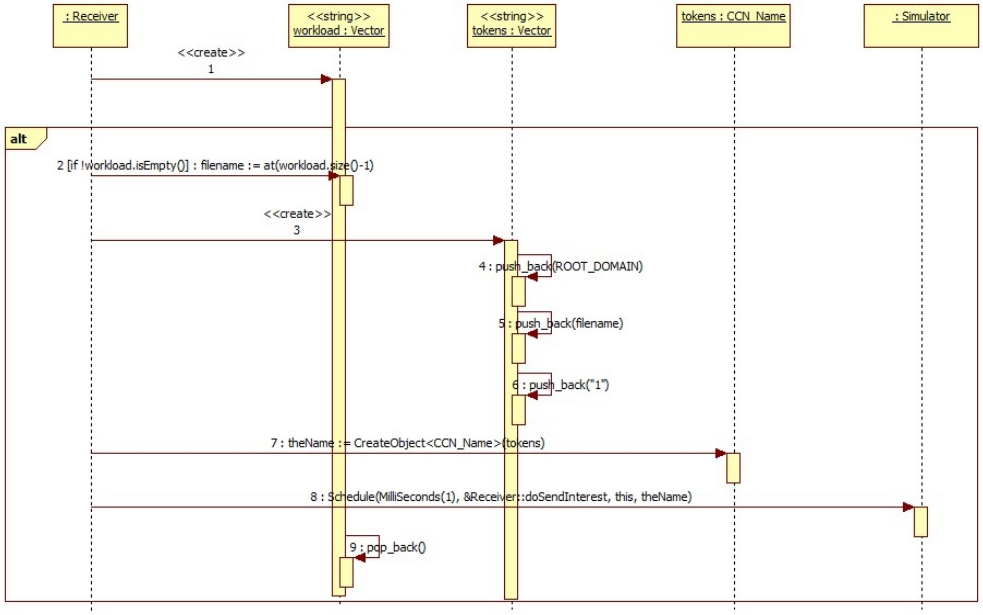
Η μέθοδος handleData καλεί την μέθοδο start (Εικ. 25) για να λάβει τα δεδομένα από το workload το οποίο είναι ένας Vector με όλα τα requests για τα πακέτα πληροφορίας.



Εικ. 24: handledata
Πηγή: Προσωπική δημιουργία

8.1.5 Μέθοδος Start

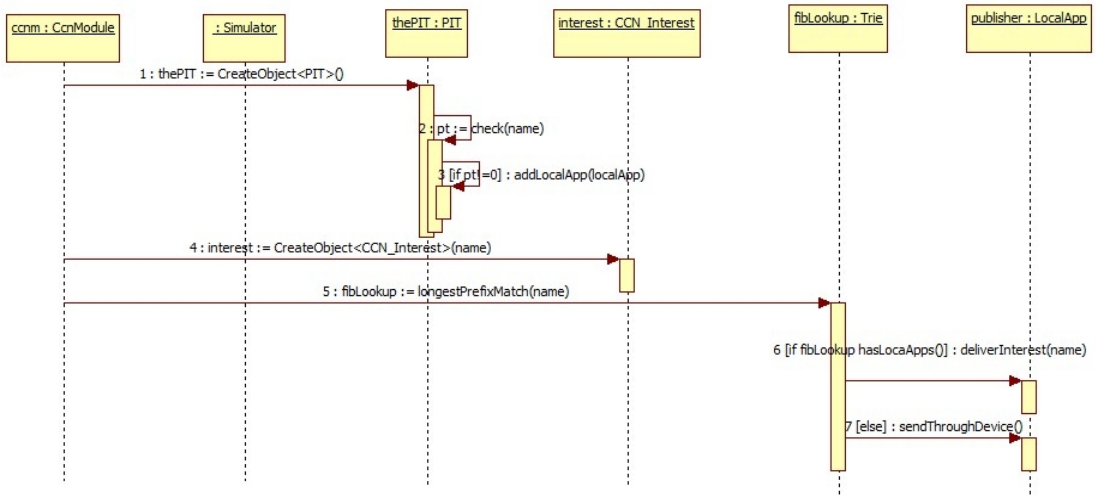
Η μέθοδος start παίρνει το πρώτο πακέτο του τελευταίου αρχείου στο workload, το οποίο στην συνέχεια απομακρύνεται από τον workload Vector, και καλεί την συνάρτηση doSendInterest για να στείλει ένα INTEREST στην localApp για το συγκεκριμένο όνομα αρχείου πληροφορίας που έχει ζητηθεί.



Εικ. 25: start
Πηγή: Προσωπική δημιουργία

8.1.6 Μέθοδος DoSendInterest

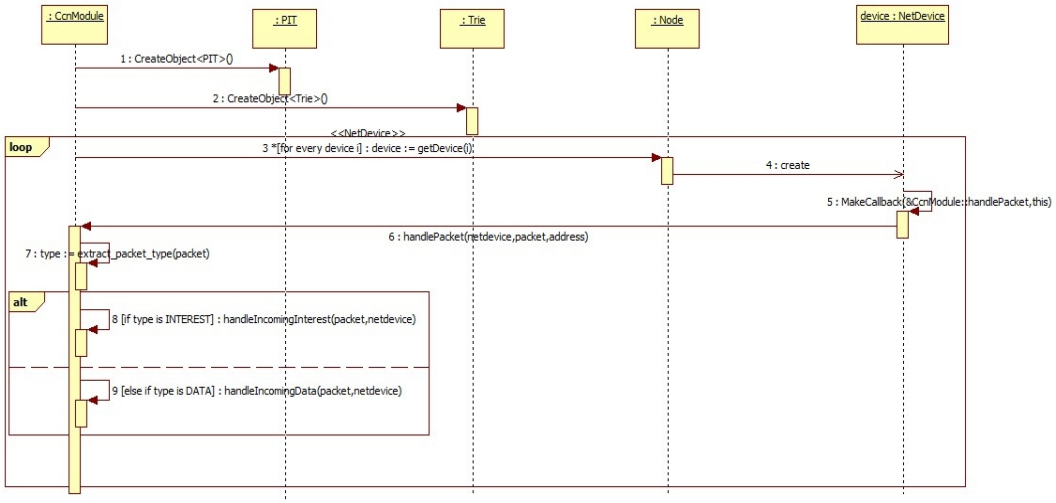
Η doSendInterest μέθοδος δημιουργεί από το CCN module το PIT table, και στη συνέχεια ελέγχει και κάνει match το μήνυμα INTEREST με το DATA με longest prefix match και έπειτα προωθεί το INTEREST.



Εικ. 26: DoSendInterest
Πηγή: Προσωπική δημιουργία

8.1.7 Μέθοδος CCNModule

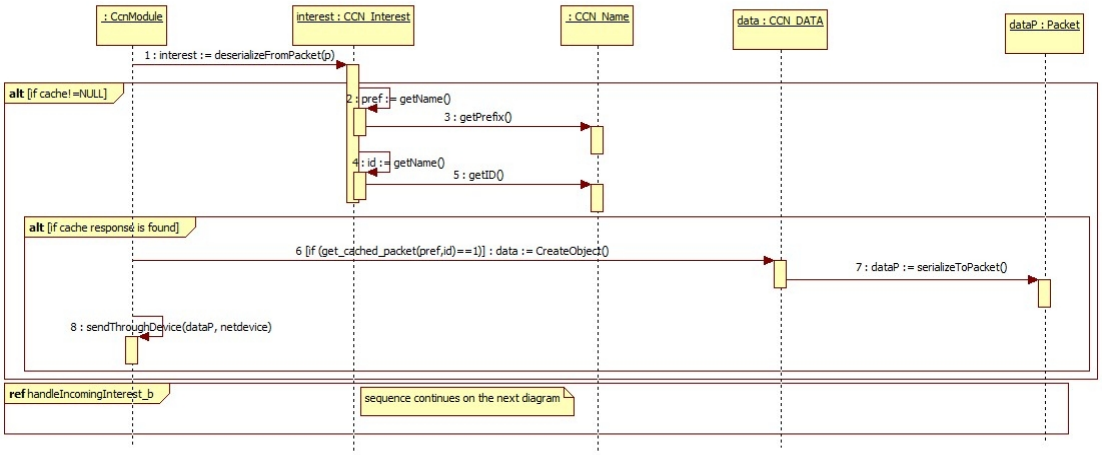
Σε αυτόν τον constructor βλέπουμε πως κατασκευάζεται ένα CCN module. Αρχικά δημιουγείται ο πίνακας PIT και έπειτα για κάθε network device καλείται η μέθοδος handlePacket στην οποία, ανάλογα με το αν πρόκειται για μήνυμα INTEREST ή DATA, καλούνται αντίστοιχα οι μέθοδοι handleIncomingInterest και handleIncomingData.



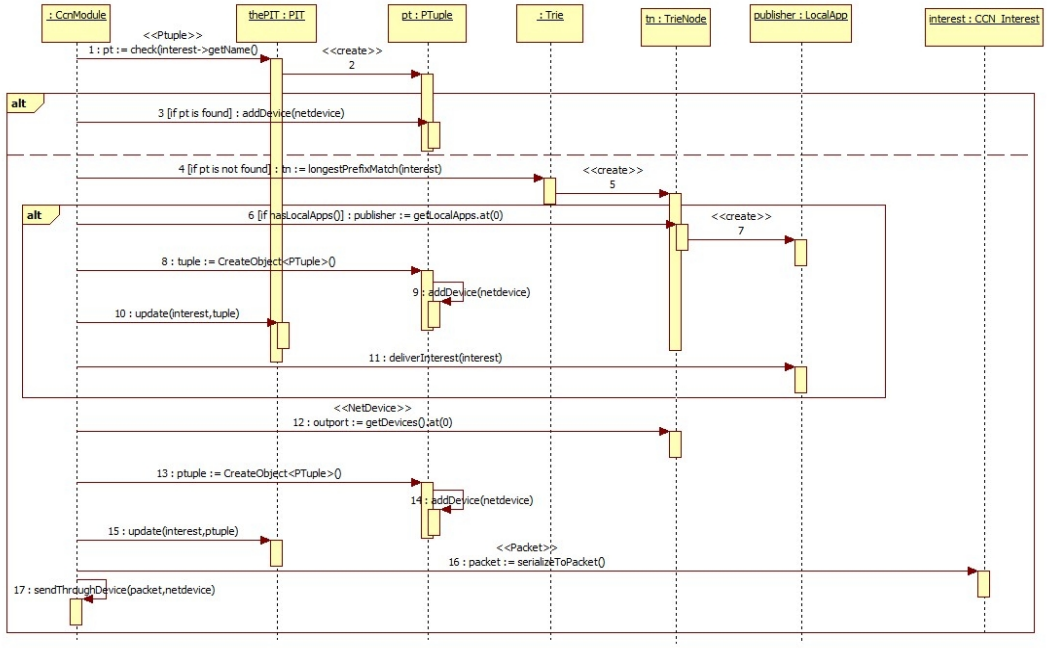
Εικ. 27: CCNModule Constructor
Πηγή: Προσωπική δημιουργία

8.1.8 Μέθοδος HandleIncomingInterest

Η handleIncomingInterest μέθοδος καλείται όταν ο τύπος του πακέτου αφορά INTEREST μήνυμα. Αν βρεθεί πακέτο το οποίο υπάρχει στην cache τότε στέλνει DATA. Διαφορετικά προωθεί το INTEREST αφού συμβουλευτεί το PIT και το κάνει deliver στον publisher. Η λειτουργία αυτή παρουσιάζεται στα επόμενα 2 διαγράμματα.



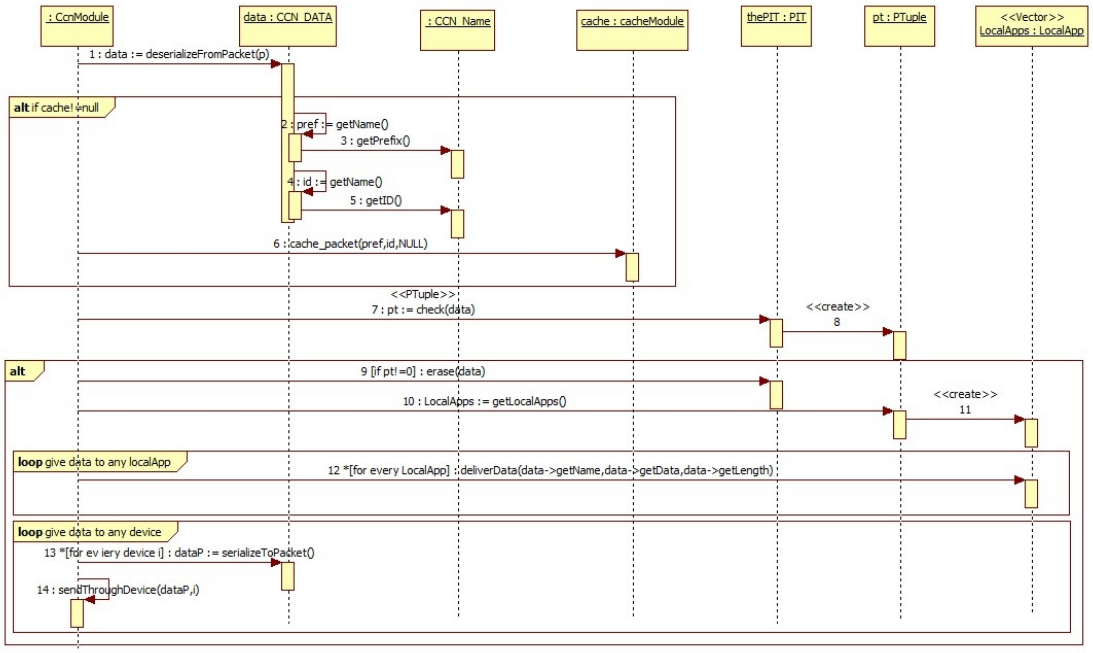
Εικ. 28: handleIncomingInterest(a)
Πηγή: Προσωπική δημιουργία



Εικ. 29:handleIncomingInterest(b)
Πηγή: Προσωπική δημιουργία

8.1.9 Μέθοδος HandleIncomingData

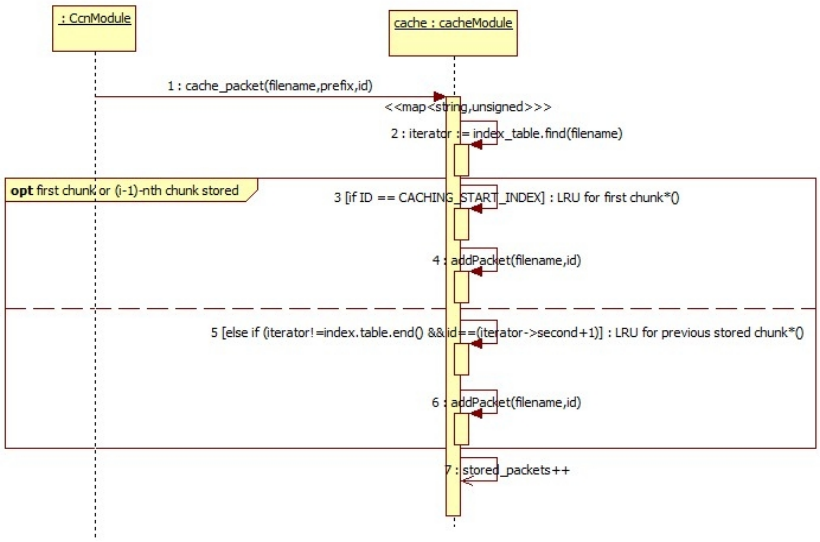
Αντίστοιχα με την handleIncomingInterest μέθοδο, η handleIncomingData καλείται όταν ο τύπος του πακέτου είναι DATA. Αν υπάρχει cache και το πακέτο υπάρχει μέσα σε αυτήν, τότε καλείται η συνάρτηση cache_packet. Διαφορετικά προωθεί τα δεδομένα σε κάθε localApp και device.



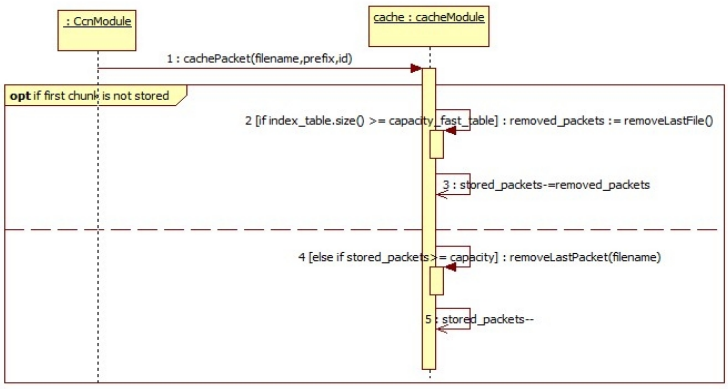
Εικ. 30:handleIncomingData
Πηγή: Προσωπική δημιουργία

8.1.10 Μέθοδος CachePacket

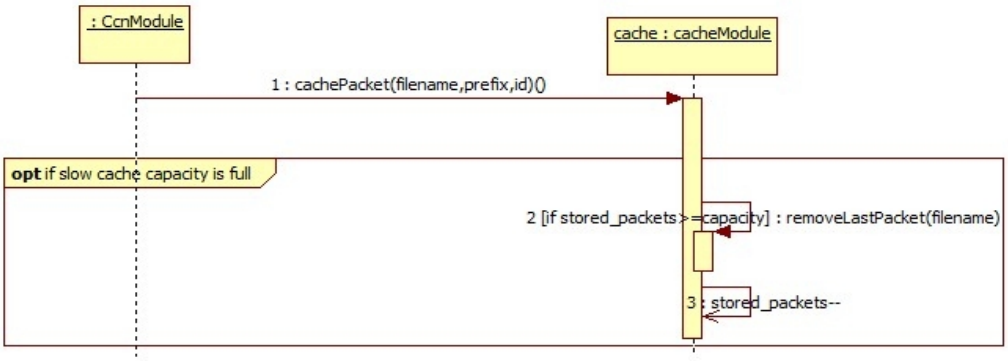
Η μέθοδος cachePacket καλείται όταν λαμβάνεται ένα data πακέτο. Ουσιαστικά αποφασίζει αν το πακέτο πρέπει να αποθηκευτεί στην cache ή όχι. Η μέθοδος αυτή αποτελεί υλοποίηση της παρούσας διπλωματικής και αναλύεται στην 3η παράγραφος της ενότητας 4.4. Συνοπτικά η διαδικασία έχει ως εξής: Αν το αντικείμενο που αφορά το συγκεκριμένο πακέτο δεδομένων που ζητήθηκε είναι αποθηκευμένο στην cache και το πακέτο αυτό είναι ακριβώς το επόμενο πακέτο στην ακολουθία (δηλαδή υπάρχει αποθηκευμένο στην cache ακριβώς το προηγούμενο πακέτο) τότε το αποθηκεύουμε στο L1 ευρετήριο και αυξάνουμε το last_chunk_id κατά 1. Διαφορετικά αν το αντικείμενο δεν είναι αποθηκευμένο και το πακέτο αποτελεί το πρώτο πακέτο για αυτό το αντικείμενο, τότε το αποθηκεύουμε στο ευρετήριο L2. Στην συνέχεια δημιουργούμε μια νέα εγγραφή στο ευρετήριο L1 με τον δείκτη Ptr_{mem} να δείχνει στο πακέτο που αποθηκεύσαμε στο L2, και κάνουμε το last_chunk_id ίσο με 1. Σε κάθε άλλη περίπτωση αγνοούμε το πακέτο δεδομένων. Στα επόμενο διάγραμμα παρουσιάζεται η λειτουργία αυτή για κάθε μία από τις δύο περιπτώσεις. Για κάθε περίπτωση ακολουθούν άλλα 2 διαγράμματα τα οποία παρουσιάζουν την LRU πολιτική αντικατάστασης που χρησιμοποιείται εκάστοτε σε κάθε μία από τις δύο περιπτώσεις. Δηλαδή όταν το πακέτο είναι το πρώτο του αντικειμένου (Εικ. 32) και όταν υπάρχει αποθηκευμένο στην cache το προηγούμενο πακέτο αντίστοιχα (Εικ. 33). Στην πρώτη περίπτωση (Εικ. 32), αν το ευρετήριο L1 είναι μεγαλύτερο από την χωρητικότητα της γρήγορης μνήμης. Αν αυτό συμβαίνει, τότε απομακρύνουμε από την cache το τελευταίο αρχείο μαζί με όλα τα πακέτα του. Αλλιώς, ελέγχουμε αν ο αριθμός των αποθηκευμένων πακέτων ξεπερνά την χωρητικότητα της αργής μνήμης. Αν ισχύει η περίπτωση αυτή, τότε απομακρύνουμε από την cache το τελευταίο πακέτο του αρχείου. Αντίστοιχα συμβαίνει και η λειτουργία στο σχήμα Εικ. 33, όπου ελέγχεται μόνο η χωρητικότητα της αργής μνήμης.



Εικ. 31:cachePacket
Πηγή: Προσωπική δημιουργία



Εικ. 32:LRU for first case(first packet and object not stored)
Πηγή: Προσωπική δημιουργία



Εικ. 33:LRU for second case(previous packet stored)
Πηγή: Προσωπική δημιουργία

9 Παραπομπές

[1] A. Finamore, M. Mellia, M. Meo, M. M. Munafo, and D. Rossi, “Experiences of internet traffic monitoring with tstat, ” Network, IEEE, vol. 25, no. 3, pp. 8–14, 2011.

[2] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, “Youtube traffic characterization: a view from the edge,” in Proceedings of the 7th ACM SIGCOMM conference on Internet measurement. ACM, 2007, pp. 15–28.

[3] Cisco. (2014) Visual networking index: Forecast and methodology. [Online]. Available: http://www.cisco.com/c/en/us/solutions/service-provider/visual-networking-index-vni/index.html

[4] S. Ihm and V. S. Pai, “Towards understanding modern web traffic,” in Proc. of the 2011 ACM Internet Measurement Conference, 2011, pp. 295–312.

[5] G. Maier, A. Feldmann, V. Paxson, and M. Allman, “On dominant characteristics of residential broadband internet traffic,” in Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference. ACM, 2009, pp. 90–102.

[6] B. Ager, F. Schneider, J. Kim, and A. Feldmann, “Revisiting cacheability in times of user generated content,” in Proc. of the IEEE Global Internet Symposium, 2010.

[7] S. Gadde, J. Chase, and M. Rabinovich, “Web caching and content distribution: A view from the interior,” Computer Communications, vol. 24, no. 2, pp. 222–231, 2001.

[8] N. T. Spring and D. Wetherall, “A protocol-independent technique for eliminating redundant network traffic,” ACM SIGCOMM Computer Communication Review, vol. 30, no. 4, pp. 87–95, 2000.

[9] A. Anand, A. Gupta, A. Akella, S. Seshan, and S. Shenker, “Packet caches on routers: the implications of universal redundant traffic elimination, ” in ACM SIGCOMM Computer Communication Review, vol. 38, no. 4, 2008, pp. 219–230.

[10] S. Arianfar, P. Nikander, and J. Ott, “On content-centric router design and implications,” in ACM ReARCH, 2010.

[11] M. Diallo, S. Fdida, V. Sourlas, P. Flegkas, and L. Tassiulas, “Leveraging caching for Internet-scale content-based publish/subscribe networks,” in IEEE ICC, 2011, pp. 1–5.

[12] European Community Future Internet Architecture (FIArch) Experts Group. (2011, March) Fundamental limitations of current Internet and the path to future Internet.[Online].Available: http://www.future-internet.eu/publications/view/article/fundamental-limitations-of-current-internet.html

[13] Z. Wang, Z. Qian, Q. Xu, Z. Mao, and M. Zhang, “An untold story of middleboxes in cellular networks,” in ACM SIGCOMM, 2011, pp. 374–385.

[14] A. Anand, A. Gupta, A. Akella, S. Seshan, and S. Shenker, “Packet caches on routers: the implications of universal redundant traffic elimination,” in ACM SIGCOMM, 2008, pp. 219–230.

[15] A. Anand, V. Sekar, and A. Akella, “SmartRE: an architecture for coordinated network-wide redundancy elimination,” in ACM SIGCOMM, 209, pp. 87–98.

[16] Cisco. (2011, June) Visual networking index: Forecast and methodology, 2010-2015. White Paper. [Online]. Available:

http://www.cisco.com/go/vni
[17] P. T. Eugster, P. A. Felber, R. Guerraoui, and A. Kermarrec, “The many faces of publish/subscribe,” ACM Computing Surveys, vol. 35, no. 2, pp.114–131, June 2003.
[18] Content Centric Networking project. [Online]. Available: http://www.ccnx.org/
[19] NSF Named Data Networking project. [Online]. Available: http://www.named-data.net/
[20] V. Jacobson, D. K. Smetters, N. H. Briggs, M. F. Plass, P. Stewart, J. D. Thornton, and R. L. Braynard, “VoCCN: Voice over content-centric networks,” in ACM ReArch Workshop, 2009.
[21] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V.Katsaros, and G. C. Polyzos, “A survey of information-centric networking research,” IEEE Communications Surveys Tutorials, vol. 16, no. 2, pp. 1024–1049, 2014.
[22] M. Meisel, V. Pappas, and L. Zhang, “Ad hoc networking via named data,” in ACM MobiArch, 2010.
[23] D. Smetters and V. Jacobson, “Securing network content,” PARC, Tech. Rep. TR-2009-01, October 2009.
[24] J. R. Santos and D. Wetherall, “Increasing effective link bandwidth by supressing replicated data.” in Proc. of the USENIX Annual Technical Conference, no. 98, 1998.
[25] S. Kumar, S. Dharmapurikar, F. Yu, P. Crowley, and J. Turner, “Algorithms to accelerate multiple regular expressions matching for deep packet inspection,” Proc. of the ACM SIGCOMM, pp. 339–350, 2006.
[26] J. C. Mogul, F. Douglis, A. Feldmann, and B. Krishnamurthy, “Potential benefits of delta encoding and data compression for http,” in Proc. of the ACM SIGCOMM, 1997, pp. 181–194.
[27] N. T. Spring and D. Wetherall, “A protocol-independent technique for eliminating redundant network traffic,” ACM SIGCOMM Computer Communication Review, vol. 30, no. 4, pp. 87–95, 2000.
[28] G. Carofiglio, M. Gallo, and L. Muscariello, “Icp: Design and evaluation of an interest control protocol for content-centric networking,” in Proc. of the IEEE INFOCOM NOMEN Workshop, 2012, pp. 304–309.
[29] Y. Thomas, C. Tsilopoulos, G. Xylomenos, and G. C. Polyzos, “Accelerating file downloads in publish subscribe internetworking with multisource and multipath transfers,” in Proc. of the World Telecommunications Congress (WTC), 2014.
[30] Z. Ming, M. Xu, and D. Wang, “Age-based cooperative caching in information-centric networks,” in Proc. of the IEEE INFOCOM NOMEN Workshop, 2012, pp. 268–273.
[31] S. Saha, A. Lukyanenko, and A. Yla-Jaaski, “Cooperative caching through routing control in information-centric networks,” in Proc. of the IEEE INFOCOM, 2013, pp. 100–104.
[32] I. Psaras, W. K. Chai, and G. Pavlou, “Probabilistic in-network caching for information-centric networks,” in Proc. of the ACM SIGCOMM ICN Workshop, 2012, pp. 55–60.
[33] S. Arianfar, P. Nikander, and J. Ott, “Packet-level caching for information-centric networking,” in Proc. of the ACM ReArch Workshop, 2010.
[34] S. K. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B. Maggs, K. Ng, V. Sekar, and S. Shenker, “Less pain, most of the gain: Incrementally deployable ICN,” in ACM SIGCOMM Computer Communication Review, vol. 43, no. 4, 2013, pp. 147–158.

[35] W. K. Chai, D. He, I. Psaras, and G. Pavlou, “Cache less for more in information-centric networks,” in Proc. of the IFIP Networking Conference, 2012, pp. 27–40.
[36] G. Rossini, D. Rossi, M. Garetto, and E. Leonardi, “Multi-terabyte and multi-gbps information centric routers,” in Proc. of the IEEE INFOCOM, 2014, pp. 181–189.
[37] M. Badov, A. Seetharam, J. Kurose, V. Firoiu, and S. Nanda, “Congestion-aware caching and search in information-centric networks,” in Proc. of the ACM ICN Conference, 2014, pp. 37–46.
[38] A. Badam, K. Park, V. S. Pai, and L. L. Peterson, “Hashcache: Cache storage for the next billion.” in Proc. of the USENIX Symposium on Networked Systems Design and Implementation (NSDI), vol. 9, 2009, pp. 123–136.
[39] D. Perino and M. Varvello, “A reality check for content centric networking,” in Proc. of the ACM SIGCOMM ICN Workshop, 2011, pp. 44–49.
[40] A. Anand, A. Gupta, A. Akella, S. Seshan, and S. Shenker, “Packet caches on routers: the implications of universal redundant traffic elimination,” in ACM SIGCOMM Computer Communication Review, vol. 38, no. 4, 2008, pp. 219–230.
[41] Y. Thomas and G. Xylomenos, “Towards improving the efficiency of ICN packet-caches,” in Proc. of the International Workshop on Quality, Reliability, and Security in ICN (Q-ICN), 2014.
[42] NS-3 Network Simulator. Available at: https://www.nsnam.org/
[43] A.-L. Barab´si and R. Albert, “Emergence of scaling in random networks,” Science, vol. 286, no. 5439, pp. 509–512, 1999.
[44] K. V. Katsaros, G. Xylomenos, and G. C.Polyzos, “GlobeTraff: a traffic workload generator for the performance evaluation of future internet architectures,” in Proc. of the International Conference on New Technologies, Mobility and Security (NTMS), 2012, pp. 1–5.