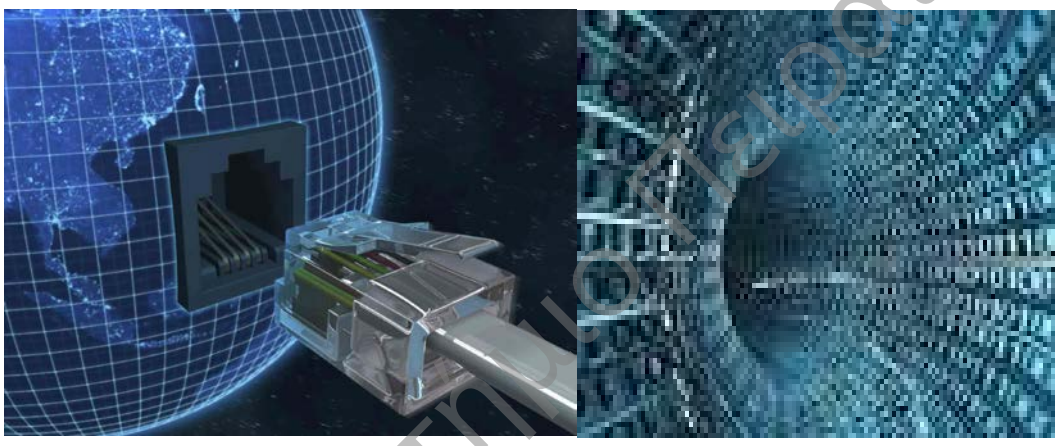
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΑ
ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
Π.Μ.Σ. ΨΗΦΙΑΚΕΣ ΕΠΙΚΟΙΝΩΝΙΕΣ ΚΑΙ ΔΙΚΤΥΑ



ΠΟΛΥΔΙΑΔΡΟΜΙΚΟ TCP- ΠΡΟΣΟΜΟΙΩΣΗ ΜΕ ΤΟ ΕΡΓΑΛΕΙΟ NS3

Διπλωματικη εργασια

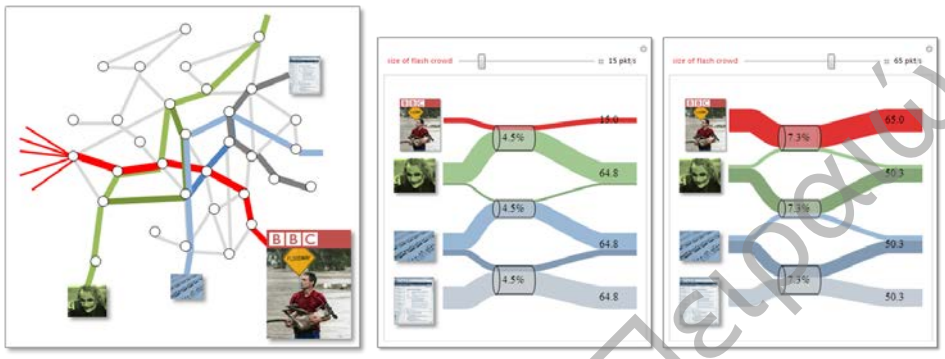
ΚΥΡΙΑΚΟΠΟΥΛΟΣ ΓΕΩΡΓΙΟΣ

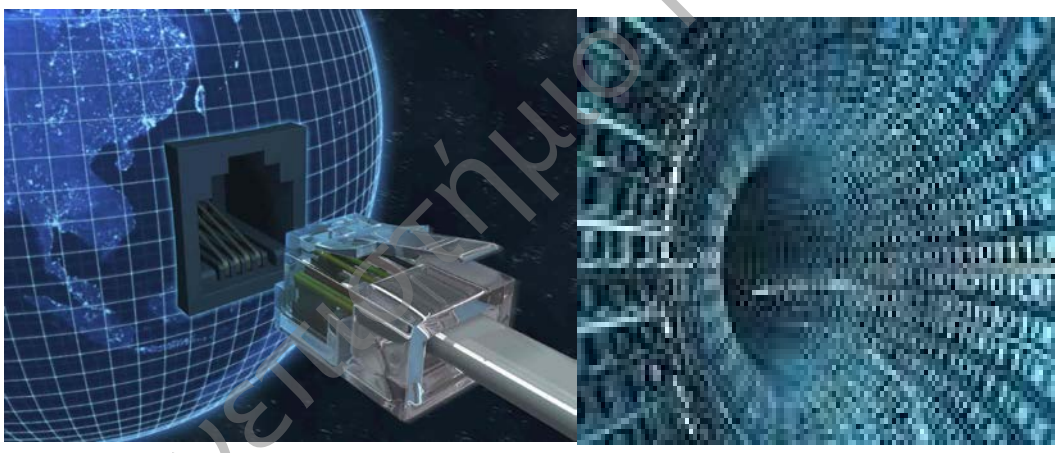
Α.Μ. ΜΕ10063

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ ***ΑΓΓΕΛΟΣ ΡΟΥΣΚΑΣ***

ΠΕΙΡΑΙΑΣ ΟΚΤΩΒΡΙΟΣ 2014

UNIVERSITY OF PIRAEUS
DEPARTMENT OF DIGITAL SYSTEMS
MASTER PROGRAM IN DIGITAL COMMUNICATIONS AND NETWORKS





MULTIPATH –TCP-SIMULATION WITH NS3

FINAL PROJECT

Author : GEORGE KYRIAKOPOULOS

A.M. 10063

SUPERVISOR ASSOCIATE PROFESSOR MR *ANGELOS ROUSKAS*

PIRAEUS OCTOBER 2014

Περίληψη

Στις μέρες μας οι κινητές συσκευές έχουν συχνά περισσότερες από μία διεπαφές δικτύου (network interfaces). Για παράδειγμα τα smartphones και tablet PCs μπορούν να έχουν πρόσβαση στο Internet είτε μέσω WiFi είτε μέσω του δικτύου κινητής τηλεφωνίας.

Οι συσκευές που είναι συνδεδεμένες ταυτόχρονα σε διαφορετικά δίκτυα λέγονται Multihomed. Η χρήση πολλαπλών διευθύνσεων (είτε στην ίδια, είτε σε πολλαπλές διεπαφές) έχει ως στόχο να καταστεί δυνατή η πρόσβαση στο διαδίκτυο μόνιμα και με όσο το δυνατόν μεγαλύτερη ανοχή σε τυχόν σφάλματα. Ιδιαίτερα μάλιστα στην περίπτωση των κινητών τερματικών στα οποία η πιθανότητα ξαφνικής διακοπής της υπάρχουσας σύνδεσης είναι σαφώς μεγαλύτερη. Η χρήση πολλαπλών μονοπατιών ταυτόχρονα ονομάζεται Multipath (routing).

Σχετικά πρόσφατα δημιουργήθηκε μια ομάδα εργασίας του IETF για τον καθορισμό ενός πρωτόκολλου πολλαπλών διαδρομών στο στρώμα μεταφορών το οποίο ονομάζεται Multi-path TCP (MPTCP). Το MPTCP είναι επέκταση του πρωτοκόλλου ελέγχου μετάδοσης (TCP), το πρωτόκολλο μεταφοράς που χρησιμοποιείται περισσότερο στο διαδίκτυο, με στόχο να μπορεί να χειρίζεται πολλαπλές υποσυνδέσεις σε διαφορετικά μονοπάτια ανάμεσα σε δύο άκρα.

Στα πλαίσια της εργασίας αυτής γίνεται μελέτη του πρωτοκόλλου MPTCP, και προσομοίωση με τη χρήση του προσομοιωτή NS-3. Γίνεται χρήση του NS3 για την προσομοίωση σεναρίων και την εξαγωγή αριθμητικών αποτελεσμάτων και συμπερασμάτων.

SUMMARY

These days, mobile phones often have more than one network interfaces. For example smartphones and tablet PCs can access the internet either from connecting via WiFi or via a mobile network.

Devices that are simultaneously connected in different networks are called Multihomed. The ability to use many addresses (whether that is in the same or in many interfaces) makes internet access possible and with bigger tolerance towards errors that might occur. Especially in the case of mobile terminals where the possibility of a sudden internet disconnection is bigger. The usage of multiple paths simultaneously is called Multipath (routing).

Recently, a work group from IETF was created, so they could determine a multipath protocol in the transport layer called Multi-path TCP (MPTCP). MPTCP is an extension of the transmission control protocol (TCP), which is the most often used transfer protocol on the internet, aiming to handle multiple hyperlinks on different paths between two ends.

In this thesis we study the MPTCP protocol, and simulate it using the NS-3 simulator. For simulating different scenarios and exporting numerical results and conclusion we use NS3.

Περιεχόμενα

Πίνακας Σχημάτων

ΕΥΧΑΡΙΣΤΕΙΕΣ

Με την ευκαιρια της ολοκληρωσης της διπλωματικης μου θα ηθελα να απευθυνω ενα μεγαλο ευχαριστω στο Κ. Αγγελο Ρουσκα ως υπευθυνο και καθοδηγητη μου για την εργασια αυτη. Η βοηθεια του σταθηκε πολυτιμη σε οτι και εαν του ζητησα πανω στην εργασια και κυριως για την επιμονη και υπομονη που υπεδειξε.

Θα θελα να ευχαριστησω τους γονεις μου για την στηριξη ψυχικη-ηθικη και οικονομικη καθ ολη τη διαρκεια του προγραμματος αλλα και των φοιτητικων μου χρονων.

1. Εισαγωγή

Στις μέρες μας οι κινητές συσκευές έχουν συχνά περισσότερες από μία διεπαφές δικτύου (network interfaces). Για παράδειγμα , οι φορητοί υπολογιστές έχουν συνήθως τουλάχιστον τόσο ενσύρματη (Ethernet) και μια ασύρματη (WiFi) κάρτα δικτύου. Ομοίως τα smartphones και tablet PCs μπορούν να έχουν πρόσβαση στο Internet είτε μέσω WiFi είτε μέσω του δικτύου κινητής τηλεφωνίας (UMTS ή 3G +).

Ένα άλλο γεγονός είναι ότι οι φορείς εκμετάλλευσης δικτύων έχουν συνήθως εις διπλούν τουλάχιστον τις σημαντικές / κεντρικές ζεύξεις δικτύου και τον αντίστοιχο εξοπλισμό με σκοπό την προστασία των δικτύων τους από αστοχίες υλικού. Επιπλέον, τα δίκτυα κορμού έχουν σε γενικές γραμμές τοπολογία mesh, δηλαδή είναι διασυνδεδεμένα όλα μεταξύ τους. Στο πλαίσιο αυτό, είναι πολύ πιθανό να υπάρχουν πολλά μονοπάτια ανάμεσα σε δύο άκρα του δικτύου. Συνεπώς προέκυψε η ιδέα να χρησιμοποιούνται ταυτόχρονα πολλά μονοπάτια, με σκοπό να βελτιωθεί η αντοχή και απόδοση των συνδέσεων από άκρο σε άκρο. Τέτοιες συνδέσεις πολλαπλών διαδρομών μπορεί να πετύχουν εξισορρόπηση του φορτίου ανάμεσα σε διαφορετικά μονοπάτια (load balancing), μεταφέροντας δυναμικά και αυτόματα την δικτυακή κίνηση από συμφορημένες ή γενικότερα προβληματικές συνδέσεις σε άλλες χωρίς προβλήματα.

Οι συσκευές που είναι συνδεδεμένες ταυτόχρονα σε διαφορετικά δίκτυα λέγονται Multihomed. Η χρήση πολλαπλών διευθύνσεων (είτε στην ίδια, είτε σε πολλαπλές διεπαφές) έχει ως στόχο να καταστεί δυνατή η πρόσβαση στο διαδίκτυο μόνιμα και με όσο το δυνατόν μεγαλύτερη ανοχή σε τυχόν σφάλματα. Ιδιαίτερα μάλιστα στην περίπτωση των κινητών τερματικών στα οποία η πιθανότητα ξαφνικής διακοπής της υπάρχουσας σύνδεσης είναι σαφώς μεγαλύτερη. Η χρήση πολλαπλών μονοπατιών ταυτόχρονα ονομάζεται Multipath (routing).

Πολλές μελέτες έχουν εξετάσει την χρήση πολλαπλών διαδρομών και μάλιστα σε διαφορετικά στρώματα του δικτύου: σε επίπεδο εφαρμογής [5], στο στρώμα

μεταφοράς [1][6][13][17][16][20], αλλά και σε άλλα στρώματα [19]. Το στρώμα μεταφοράς όμως θεωρείται από τους περισσότερους το καταλληλότερο για την εφαρμογή Multipath.

Σε αυτό το στρώμα, τα συστήματα μπορούν να συλλέξουν πληροφορίες σχετικά με κάθε διαδρομή που χρησιμοποιείται ελέγχοντας μεγέθη όπως η χωρητικότητα , η καθυστέρηση (latency) και η συμφόρηση του δικτύου. Αυτές οι πληροφορίες μπορούν στη συνέχεια να χρησιμοποιηθούν για την ανάδραση σε γεγονότα συμφόρησης στο δίκτυο επιλέγοντας αποσυμφορημένα μονοπάτια. Σχετικά πρόσφατα δημιουργήθηκε μια ομάδα εργασίας του IETF για τον καθορισμό ενός πρωτόκολλου πολλαπλών διαδρομών στο στρώμα μεταφορών το οποίο ονομάζεται Multi-path TCP [3] (MPTCP). Πιο συγκεκριμένα , αυτή η ομάδα εργασίας αναπτύσσει επεκτάσεις του πρωτοκόλλου ελέγχου μετάδοσης (TCP), το πρωτόκολλο μεταφοράς που χρησιμοποιείται περισσότερο στο διαδίκτυο, με στόχο να μπορεί να χειρίζεται πολλαπλές υποσυνδέσεις σε διαφορετικά μονοπάτια ανάμεσα σε δύο άκρα.

Μάλιστα το MPTCP χρησιμοποιήθηκε πρόσφατα από την Apple για την μεταφορά δεδομένων του Siri, ενός ψηφιακού οδηγού που είναι εγκατεστημένος στο iPhone. Ο χρήστης μπορεί να ρωτήσει κάτι το siri, η ερώτησή του ηχογραφείται, στέλνεται κεντρικά με την μορφή Mp3 και λαμβάνεται πάλι μια απάντηση σε Mp3. Η ανταλλαγή των ηχητικών μηνυμάτων γίνεται πλέον με χρήση MPTCP, εφόσον το τηλέφωνο είναι συνδεδεμένο σε Wifi και σε δίκτυο κινητής τηλεφωνίας [21].

Υπάρχουν δυο σημαντικές μελέτες στην προσομοίωση του πρωτοκόλλου MPTCP [7] [22]. Στη δημοσίευση [7] οι ερευνητές επικεντρώθηκαν στους μηχανισμούς ελέγχου συμφόρησης, χωρίς να προχωρήσουν στην εφαρμογή άλλων τμημάτων του MPTCP . Ο περιορισμός αυτός κληρονομήθηκε από τον προσομοιωτή που χρησιμοποιήθηκε τον htsim, ο οποίος δεν επιτρέπει τη δημιουργία ενός ακριβούς μοντέλου του προσομοιωμένου πρωτοκόλλου. Αντίθετα στη δημοσίευση [22] οι ερευνητές παρουσιάζουν ένα πληρέστερο μοντέλο του MPCTP με τη χρήση του προσομοιωτή NS-3 [12].

Ο NS-3 ανήκει στην κατηγορία των προσομοιωτών δικτύων διακριτών γεγονότων (discrete event network simulator). Οι προσομοιωτές της κατηγορίας αυτής, μοντελοποιούν τη λειτουργία ενός συστήματος, ως μια ακολουθία διακριτών γεγονότων στο χρόνο, με κάθε γεγονός να συμβάλει στην διαμόρφωση της κατάστασης του συστήματος. Στόχος του ns-3 είναι να παρέχει ένα ανοιχτό περιβάλλον προσομοίωσης που θα προτιμάται από την ερευνητική κοινότητα, παρέχοντας υποστήριξη για τις σύγχρονες ανάγκες της προσομοίωσης δικτύων και ελευθερία στην ανάπτυξη και επέκταση του προσομοιωτή.

Στα πλαίσια της εργασίας αυτής γίνεται μελέτη του πρωτοκόλλου MPTCP, και προσομοίωση με τη χρήση του προσομοιωτή NS-3. Τα πειράματα προσομοίωσης βασίζονται στον κώδικα που προέκυψε από την δημοσίευση [22] και είναι διαθέσιμος στον σύνδεσμο [23]. Στα πλαίσια της εργασίας, θα γίνει χρήση του NS3 και του παραπάνω κώδικα για την προσομοίωση σεναρίων και την εξαγωγή αριθμητικών αποτελεσμάτων και συμπερασμάτων.

Το υπόλοιπο τμήμα της εργασίας έχει την παρακάτω δομή: Στο κεφάλαιο 2 παρουσιάζονται τα πρωτόκολλα TCP και MPTCP. Στο κεφάλαιο 3 γίνεται μια συνοπτική παρουσίαση του προσομοιωτή NS-3 και των βασικών δυνατοτήτων του. Στο τέταρτο κεφάλαιο παρουσιάζονται τα σενάρια προσομοίωσης και τα αναλύονται τα αριθμητικά αποτελέσματα. Η εργασία ολοκληρώνεται στο πέμπτο κεφάλαιο με τα συμπεράσματα.

2. TCP και Multipath TCP

2.1. Το πρωτόκολλο TCP

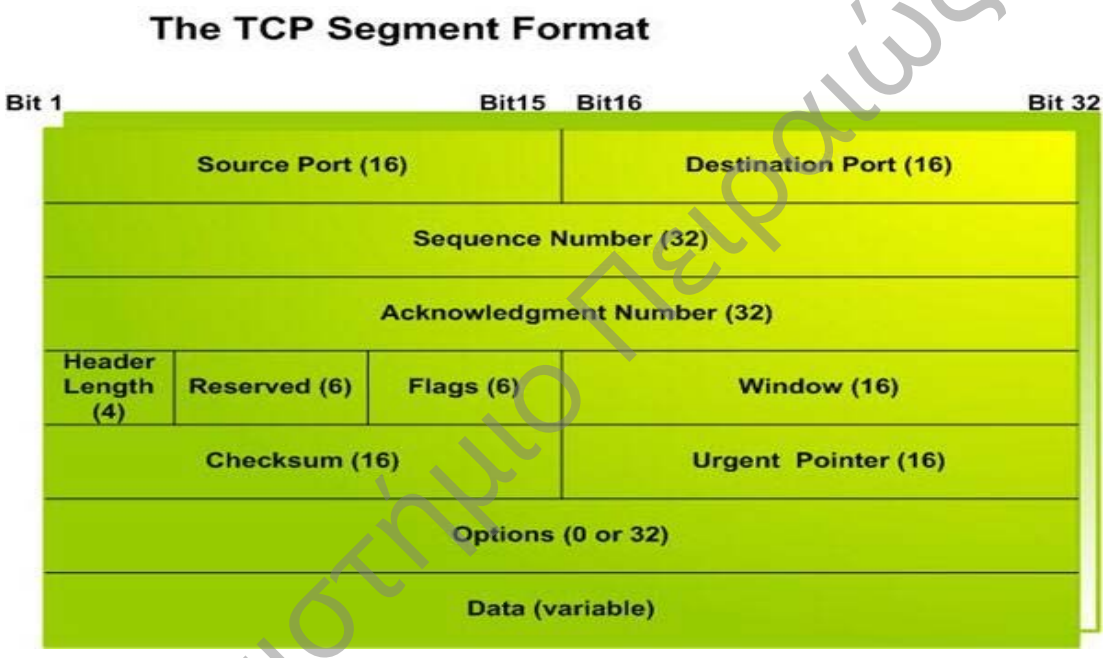
Το TCP (Transmission Control Protocol)είναι ένα πρωτόκολλο 4ου επιπέδου μεταφοράς του ISO/OSI και είναι υπεύθυνο για την αξιόπιστη μετάδοση πληροφοριών μέσα σε ένα δίκτυο. Το TCP εντάσσεται σε μια πολύ-επίπεδη αρχιτεκτονική πρωτοκόλλου ακριβώς πάνω από ένα βασικό πρωτόκολλο Internet, το οποίο παρέχει έναν τρόπο έτσι ώστε το TCP να μπορεί να δέχεται και να στέλνει μεταβλητού μήκους πληροφορίες που περικλείονται μέσα σε Internet datagram φακέλους. Σκοπός του datagram είναι η εύρεση της πηγής προέλευσης και προορισμού των TCPS's μέσα σε διαφορετικά δίκτυα. Οι πληροφορίες που στέλνονται είναι ομαδοποιημένες σε οκτάδες (bytes) και φτάνουν στον παραλήπτη με την ίδια σειρά που στάλθηκαν.

Η αποστολή πακέτων, δηλαδή πληροφοριών χωρισμένων σε οκτάδες, μπορεί να είναι αμφίδρομη (full-duplex) οπότε στην ουσία να επιτρέπεται μετάδοση πακέτων ταυτόχρονα και από τις δύο κατευθύνσεις, και από τον αποστολέα και τον παραλήπτη. Το TCP καθιστά ένα πολύ αξιόπιστο πρωτόκολλο μεταφοράς καθώς αναμεταδίδει τα πακέτα και στέλνει αναφορές παράδοσης στον αποστολέα για να είναι σίγουρη η λήψη τους και να μην υπάρχουν τυχόν σφάλματα.

Για να αποφευχθεί τυχόν συμφόρηση στο δίκτυο και ενδεχομένως μεταγενέστερη κατάρρευσή του, λόγω της μαζικής αποστολής πολλών πακέτων, το πρωτόκολλο TCP χρησιμοποιεί έναν αλγόριθμο που ελέγχει το ποσοστό της συμφόρησης που δημιουργείται. Όταν η χωρητικότητα πακέτων στο δίκτυο φτάσει στο ανώτερο όριο της, οι ενταμιευτές (buffers) των δρομολογητών δεν είναι ικανοί να λάβουν και να στείλουν άλλα πακέτα στον προορισμό τους και έτσι τα πακέτα απορρίπτονται και ξεκινάει η διαδικασία αναμετάδοσης τους. Με αυτή την

διαδικασία, ο αποστολέας ενημερώνεται για την απώλεια (αποτυχία αποστολής του πακέτου πληροφοριών) και έτσι μειώνει τον όγκο δεδομένων που μεταδίδει, βοηθώντας στην αποφόρτωση του δικτύου.

Ο συγκεκριμένος αλγόριθμος του TCP βοηθάει στην ομαλή λειτουργία του δικτύου και στην αξιόπιστη μετάδοση πληροφοριών στους σωστούς παραλήπτες σε άμεσο χρόνο.



Σχήμα 1 Η μορφή ενός πακέτου TCP

2.1.1. Ο μηχανισμός Ελέγχου Συμφόρησης στο TCP

Ο έλεγχος που πραγματοποιεί ο παραπάνω αλγόριθμος που χρησιμοποιεί ο TCP για την αποφυγή συμφόρησης δικτύου λόγω του φόρτου πακέτων ονομάζεται μηχανισμός ελέγχου συμφόρησης και αποτελείται από τα εξής στάδια :

- Αρχής (Slow Start)
- Αποφυγής Συμφόρησης (Congestion Avoidance)
- Ταχείας Αναμετάδοσης (Fast Retransmit)
- Ταχείας Ανάκαμψης (Fast Recovery)

Εξαιτίας της όλο και αυξανόμενης ποσότητας πακέτων που αποστέλλονται η επέκταση δικτύου είναι απαραίτητη για να μπορέσει να εξυπηρετήσει τον μεγάλο αριθμό πελατών. Για να γίνει αυτό αρμονικά ο έλεγχος της κίνησης και συμφόρησης είναι απαραίτητος ανά πάσα στιγμή. Για τον λόγο αυτό επιδιώκεται η προσαρμογή του όγκου των δεδομένων των οποίων η λήψη δεν έχει επιβεβαιωθεί και η αναμετάδοση των πακέτων πληροφοριών (segment) που χάθηκαν και απορρίφθηκαν από το δίκτυο.

Ένα επίσης βασικό ρόλο στον μηχανισμό ελέγχου συμφόρησης παίζουν τα μηνύματα επιβεβαίωσης παράδοσης των πακέτων (ACKs) που αποστέλλονται στον αποστολέα όταν γίνει η λήψη ολόκληρου του περιεχομένου του πακέτου από τον παραλήπτη. Η διαδικασία που πραγματοποιείται είναι η εξής.

Αρχικά κάθε πακέτο πληροφοριών έχει ένα μοναδικό σειριακό αριθμό (sequence number) τον οποίο παίρνει από το πρωτόκολλο με την αποστολή του πακέτου, ο κωδικός αυτός δεν μπορεί να εμφανιστεί παραπάνω της μιας φοράς στα πακέτα του δικτύου, στην ουσία αποτελεί μια ταυτότητα του πακέτου. Όταν αυτό το πακέτο φτάσει στον προορισμό του, γίνεται αποστολή ενός μηνύματος πάλι πίσω στον αποστολέα και έτσι γίνεται η επιβεβαίωση της σωστής αποστολής του πακέτου. Το μήνυμα επιβεβαίωσης περιέχει έναν αριθμό που αντιστοιχεί σε αυτόν που θα έπαιρνε το επόμενο σε σειρά πακέτο.

Μήνυμα παράδοσης λαμβάνει επίσης ο αποστολέας και στην περίπτωση που καταφτάσουν περισσότερα του ενός πακέτα με διαφορετική σειρά, δηλαδή με το μη αναμενόμενο αριθμό σειράς. Στην περίπτωση αυτή το μήνυμα παράδοσης που θα λάβει επιβεβαιώνει την λήψη του τελευταίου πακέτου που έφτασε με την σωστή σειρά.

Η μεταβλητή timeout είναι αυτή που καθορίζει τον χρόνο που θα κάνει ο αποστολέας να λάβει το μήνυμα παράδοσης του πακέτου. Αν ο χρόνος παράδοσης της επιβεβαίωσης είναι μεγαλύτερος από τον καθορισμένο χρόνο αποστολής της μεταβλητής, ο αποστολέας αναμεταδίδει το πακέτο, διότι υποθέτει ότι χάθηκε ή απορρίφθηκε.

Υπάρχουν δύο μεταβλητές που βοηθούν στον έλεγχο συμφόρησης. Η πρώτη είναι το παράθυρο συμφόρησης (congestion window), το οποίο είναι η μικρότερη ποσότητα δεδομένων που μπορεί να στείλει ο αποστολέας πριν λάβει μια επιβεβαίωση παράδοσης (ack). Η τιμή του παραθύρου ωστόσο δεν είναι ανεξάρτητη. Ορίζεται από την τιμή του παραθύρου που θα θέσει ο παραλήπτης και μπορεί να αυξομειώνεται καθ' όλη την διάρκεια της αποστολής δεδομένων αναλόγως με το ποσοστό συμφόρησης.

Η δεύτερη μεταβλητή είναι το κατώφλι της Αργοπορημένης εκκίνησης (slow-start). Σκοπός της είναι να ελέγχει ποιος αλγόριθμος χρησιμοποιείται για την ρύθμιση τιμής του παραθύρου συμφόρησης που αναφέρθηκε παραπάνω. Αν η τιμή του παραθύρου συμφόρησης είναι πιο μικρή από το κατώφλι της Αργοπορημένης εκκίνησης τότε ο συγκεκριμένος αλγόριθμος ξεκινάει προσπάθεια αύξησης της τιμής. Στην περίπτωση που η τιμή του παραθύρου συμφόρησης είναι μεγαλύτερη από το κατώφλι τότε χρησιμοποιείται ο αλγόριθμος της Αποφυγής συμφόρησης για την αύξηση του. Η αρχική τιμή του κατωφλίου Αργής Αρχής είναι ίση με το παράθυρο που γνωστοποιεί ο παραλήπτης.

Ο αποστολέας θα μπορέσει να χρησιμοποιήσει στο μέγιστο το εύρος ζώνης του δικτύου αν το παράθυρο του (sender's window), κάποια στιγμή γίνει ίσο με το

γινόμενο της καθυστέρησης επί το εύρος ζώνης τα σύνδεσης (delay-bandwidth product). Η παραπάνω τιμή δείχνει ποια θα μπορούσε να είναι η χωρητικότητα των δεδομένων που έχουν σταλθεί αλλά δεν έχουν ληφθεί ακόμα από τον παραλήπτη.

Η χρήση του μέγιστου της απόδοσης του στενωπού του εύρους ζώνης, πραγματοποιείται όταν ο χρήστης στέλνει το μέγιστο αριθμό δεδομένων που δεν έχουν ληφθεί. Στην περίπτωση που ο παραλήπτης των δεδομένων, δεν έχει ορίσει τιμή παραθύρου ίση με το γινόμενο καθυστέρησης επί του εύρους ζώνης της σύνδεσης τότε δεν θα μπορούν να εισαχθεί ο μέγιστος όγκος δεδομένων στο δίκτυο.

2.1.2. Βασικές εκδόσεις του TCP

2.1.2.1. TCP Tahoe

Οι σύγχρονες εφαρμογές TCP έχουν αλγορίθμους που ελέγχουν την συμφόρηση που προκαλείται στο δίκτυο ενώ παράλληλα είναι υπεύθυνοι για την διατήρηση της καλής απόδοσης των χρηστών. Οι παλιότερες εκδόσεις του TCP χρησιμοποιούσαν ένα μοντέλο επανάκλασης , το οποίο μετά από συγκεκριμένο χρονικό διάστημα αν δεν είχε δεχτεί μήνυμα παράδοσης, ξαναέστελνε τα δεδομένα. Αυτή η προσέγγιση δεν βοηθούσε και πολύ την συμφόρηση δικτύου.

Ο TCP Tahoe έχει νέους αλγορίθμους και βελτιώσεις σε σχέση με τις προηγούμενες εκδόσεις του. Οι αλγόριθμοι αυτοί είναι οι Αργοπορημένη εκκίνηση (Slow-Start) , Αποφυγή συμφόρησης (Congestion Avoidance) και Ταχεία αναμετάδοση (Fast Retransmit).

Αρχικά όταν θέλει να πραγματοποιηθεί μια σύνδεση, το παράθυρο συμφόρησης (cwnd) είναι ίσο σε μέγεθος με το αρχικό παράθυρο (initial window -

iw) και ξεκινάει ο αλγόριθμος Αργοπορημένη εκκίνηση, κατά τον οποίο το παράθυρο συμφόρησης αυξάνεται αναλόγως με το μέγιστο αριθμό πακέτων (maximum segment size) για κάθε επιβεβαίωση παράδοσης που καταφτάνει.

Το μέγεθος του παραθύρου αυξάνεται εκθετικά ενώ ταυτόχρονα διαπλασιάζεται ανά χρόνο μετάδοσης με επιστροφή (rtt). Η αύξηση του παραθύρου βοηθάει στην γρήγορη μετάδοση πακέτων, αποτρέπωντας με αυτό τον τρόπο την συμφόρηση του δικτύου. Όταν το παράθυρο συμφόρησης, γίνει το ίδιο σε μέγεθος με αυτό του κατωφλίου της αργοπορημένης εκκίνησης, ολοκληρώνεται η φάση της αργοπορημένης εκκίνησης.

Έπειτα ξεκινάει ο αλγόριθμος Αποφυγής συμφόρησης. Σε αυτό το στάδιο, το παράθυρο συμφόρησης αυξάνεται γραμμικά και συγκεκριμένα κατά ένα μέγιστο μέγεθος πακέτου (mss) ανά χρόνο μετάδοσης με επιστροφή (rtt). Σε αυτή τη φάση γίνεται προσπάθεια αύξησης του ρυθμού μετάδοσης, έτσι ώστε να προσεγγίσει όσο το δυνατό γίνεται το ρυθμό εισόδου πακέτων για να αποφευχθεί η συμφόρηση.

Το επόμενο στάδιο είναι η έναρξη του αλγορίθμου Ταχείας Αναμετάδοσης (Fast retransmit) που είναι υπεύθυνη για τον εντοπισμό απώλειας πακέτων και αποτελεί την βασικότερη διαφορά του TCP Tahoe με τις προηγούμενες του εκδόσεις. όταν λήξη ο χρομομετρητής (rto) και ο αποστολέας δεν έχει λάβει μήνυμα επιβεβαίωσης παράδοσης ο αλγόριθμος αντιλαμβάνεται ότι χάθηκαν κάποιες πληροφορίες. Τότε το παράθυρο συμφόρησης γίνεται ίσο με ένα μέγιστο μέγεθος πακέτου (1MSS) ενώ το κατώφλι γίνεται το μισό του μεγέθους των δεδομένων που δεν έχουν αποσταλεί και του διπλάσιου του μέγιστου μεγέθους πακέτου (2MSS). Όταν φτάσει και τρίτη επιβεβαίωση απανωτά για το ίδιο πακέτο, ο αλγόριθμος ξανά θέτει σε λειτουργία τον αλγόριθμο της Αργοπορημένης εκκίνησης.

Οι αλγόριθμοι ελέγχου συμφόρησης του TCP Tahoe φαίνονται στον Πίνακα

1.

Ορίστε τις ακόλουθες μεταβλητές:

cw : 0..sw
μέγεθος παραθύρου συμφόρησης.
Πραγματικός αριθμός που κυμαίνεται από το 0 μέχρι το **sw** (μέγεθος παραθύρου).

sst : 0..MAXW
Όριο(threshold) αργοπορημένης εκκίνησης (slow-start)

rtt
τελευταία μέτρηση χρόνου roundtrip(χρόνος μετάβασης με επιστροφή)
rttAvg
Εκθετικά αυξανόμενοι χρόνοι roundtrip κατά μέσο όρο.
Κυμαίνεται από το 0 έως το MAX_RTT.

rttDev
Αποκλίσεις εκθετικά αυξανόμενων χρόνων roundtrip

rttErr
Απόκλιση τελευταίου μετρήσιμου χρόνου roundtrip

rto : 0..MAX_RTO
Τιμή Timeout.

rtoTimer: {STOPPED, RUNNING}
Becomes RUNNING when it is started or restarted.
Παίρνει την τιμή RUNNING όταν ξεκινάει ή όταν γίνεται επανεκίνηση.
Σαν τιμή default έχει την STOPPED.
Γίνεται timeout όταν ο χρονομετρητής (rto)ξεπεράσει το όριο του από την τελευταία (επανε)κίνηση.

ΑΛΓΟΡΙΘΜΟΣ:

```
Acceptance of new data from local user {
   if (data is within congestion window size)    // τα δεδομένα
μπορούν να σταλθούν
      then { send data ;
             if (rtoTimer is off)       // nothing previously
outstanding
                then start rtoTimer ;
           }
}

Reception of "new" ACK (i.e., ack sequence number > na) {
                                        // ανανέωση της μεταβλητής cw
   if (cw < sst)
      then cw := cw + 1                          // αύξηση slow-start
```

```
      else if (cw >= sst)
           then cw := min(sw, cw + (1/cw)) ;     // γραμμική αύξηση

      // δεν γίνονται αλλαγές στο sst

      // rtt ισούται με τον χρόνο της τελευαίας (επανε)κίνησης του
    // rtoTimer
      // ανανέωση μεταβλητών σχετικών με το rtt αν η επιβεβαίωση (ack)
   // δεν είναι από πακέτο που έχει ξανασταλθεί
      if ( ack is for data that was transmitted only once ) then {
           rttAvg := (1-x)*rttAvg + x*rtt ;     // στο περίπου το x
   είναι 1/8
           rttErr := magnitude(rttAvg - rtt) ;
           rttDev := (1-y)*rttDev + y*rttErr ;  // στο περίπου το y
   είναι 1/8 or 1/4
           rto := rttAvg + z*rttDev ;           // στο περίπου το z
   είναι 2 or 4
      }

      send any packets that have entered congestion window due to this
   ack ;

   // Η συγκεκριμένη επιβεβαίωση (ack) έχει αυξήσει ταυτόχρονα
   // και το μέγεθος(cw) το κατώφλι (na) του παραθύρου συμφόρησης
   // (congestion window)

      if there is outstanding data
           then  restart rtoTimer
           else  stop rtoTimer
   }

   RtoTimer Timeout {
      // μεγάλη ποσότητα δεδομένων και έχει περάσει ο χρόνος rto από
   // την αποστολή της τελευταίας αποστολής
      // ΠΡΟΣΟΧΗ: Αν το rto παραμείνει στο MAX_RTO for MAX_RETRY (= 3
   // συνήθως)
      // διαδοχικά timeouts, τότε ο TCP σταματάει.

      // ανανέωση cw, stt; είσοδος slow start
      stt := ceiling(cw/2);
      cw :=  1;

      //ανανέωση μεταβλητών σχετικών με το rtt
      rto := min( 2 * rto , MAX_RTO );    //διπλό rto

      send packets in congestion window ;  // πακέτα cw
      restart rtoTimer ;
   }
```

Πίνακας 1 Ο αλγόριθμος ελέγχου συμφόρησης του TCP Tahoe

2.1.2.2. TCP Reno

Το γεγονός ότι φτάνουν συνεχόμενες επιβεβαιώσεις δείχνει ότι η ροή των πακέτων ήταν κανονική παρά το γεγονός ότι χάθηκαν ορισμένα από αυτά. Η σωστή ροή των πακέτων μειώνει το ποσοστό συμφόρησης και έτσι δεν χρειάζεται να μειωθεί δραματικά το παράθυρο συμφόρησης και η επανεκκίνηση της Αργοπορημένης Εκκίνησης.

Ο TCP Reno χρησιμοποιώντας τον αλγόριθμο Ταχείας ανάκαμψης (Fast recovery) βοηθάει την μείωση συμφόρησης όταν αντιληφθεί την επαναλαμβανόμενη άφιξη επιβεβαιώσεων.

Όταν ο αλγόριθμος καταλάβει ότι χαθεί κάποιο πακέτο, μετατρέπει το κατώφλι στο μισό του μεγέθους των δεδομένων που δεν έχουν ληφθεί ακόμα και στο διπλάσιου του μέγιστου μεγέθους πακέτου (2MSS). Στην συνέχεια ξαναστέλνετε μέχρι να να λάβει νέα επιβεβαίωση (και όχι επανάληψης) και τότε το παράθυρο συμφόρησης γίνεται ίσο με το κατώφλι και η μετάδοση συνεχίζεται από τον αλγόριθμο Αποφυγής Συμφόρησης.

Ο αλγόριθμος ελέγχου συμφόρησης του TCP Reno φαίνονται στον Πίνακα 2.

```
nDupAck : 0..3
Αριθμός διπλών επιβεβαιώσεων (acks) που έχουν
ληφθεί.  Αρχική τιμή = 0.


Reception of duplicate ACK (i.e., ack seq
number = na) {
    if (nDupAck < 2) then  nDupAck := nDupAck
+ 1 else {
        // λήψη 3 διπλών επιβεβαιώσεων (acks)
        nDupAck := 0 ;

        //ταχεία ανάκαμψη (fast
// recovery):μείωση κατά του ήμιση sst και
 // παράληψη αργοπορημένης εκκίνησης (slow-
```

```
// start)
        sst := ceiling(cw/2) ;
        cw := sst ;

        //ταχεία επανάκλαση (fast
// retransmit):μη αναμονή υπέρβασης χρονικού
//ορίου (timeout)
 send requested packet ; // το χαμηλότερο
// πακέτο στο παράθυρο συμφόρησης

        restart rtoTimer ;
    }
}

Reception of "new" ACK (i.e., ack sequence
number > na) {
    As in TCP Tahoe except that nDupAck is
zeroed ;
}
```

Πίνακας 2 Οι αλγόριθμοι ελέγχου συμφόρησης του TCP Reno

2.1.2.3. TCP NewReno

Ο TCP NewReno παρουσιάζει κάποιες διαφορές σε σχέση με τον TCP Reno όσον αφορά την συμπεριφορά του αποστολέα στην φάση της Ταχείας ανάκαμψης (fast recovery) όταν λαμβάνονται επιβεβαιώσεις παράδοσης για ορισμένα από τα πακέτα που στάλθηκαν.

Ενώ στον TCP Reno οι «μισές» επιβεβαιώσεις σταματούν την λειτουργία της Ταχείας ανάκαμψης, με το να μικραίνουν το μέγεθος του παραθύρου στο μέγεθος του παραθύρου συμφόρησης, στην αναβαθμισμένη έκδοση NewReno η λήψη αυτής της βεβαίωσης δεν δηλώνει την απώλεια ολόκληρου του πακέτου αλλά του αμέσως επόμενου προγραμματισμένου πακέτου και ο αλγόριθμος δέχεται την εντολή να

ξαναστείλει μόνο από το χαμένο κομμάτι και μετά και όχι ολόκληρη την πληροφορία.

Οπότε όταν χαθούν πολλά πακέτα από μια σειρά πληροφοριών, ο NewReno μπορεί να ανακτήσει ένα χαμένο πακέτο σε κάθε roundtrip, χωρίς να περιμένει ο χρονομετρητής να περάσει το ανώτερο όριο του. Η διαδικασία αυτή πραγματοποιείται έως ότου ανακτηθούν όλα τα χαμένα πακέτα και σε όλη αυτή την διάρκεια ο αλγόριθμος βρίσκεται στο στάδιο Ταχείας ανάκαμψης.

2.2. To Multipath TCP

Όπως φαίνεται και από την ονομασία του, το πρωτόκολλο Multipath TCP αποτελεί επέκταση του TCP και κύριο χαρακτηριστικό του είναι ότι επιτρέπει την χρήση πολλών μονοπατιών μεταξύ δύο ή περισσότερων διεπαφών ταυτόχρονα. Επιπλέον χαρακτηριστικό του είναι ότι δεν χρειάζεται να γίνουν αλλαγές στις εφαρμογές που αρχικά χρησιμοποιούσαν το TCP ενώ τώρα το Multipath TCP.

2.2.1. Υπόβαθρο - Multihoming στο επίπεδο αναφοράς

Σε ένα δίκτυο το Multihoming μπορεί να ανακτήσει χαμένα δεδομένα σε κάποια τυχόν αποτυχία αποστολής του πακέτου. Για να μπορέσει όμως να αξιοποιήσει πολλές ζεύξεις σε μια μόνο σύνδεση μεταφοράς θα πρέπει να πραγματοποιηθούν τροποποιήσεις στο επίπεδο μεταφοράς και όχι στο επίπεδο δικτύου. Αυτό εν μέρη είναι καλό γιατί το τροποποιημένο πρωτόκολλο μεταφοράς μετέπειτα μπορεί να χρησιμοποιηθεί χωρίς κανένα πρόβλημα σε διάφορα πρωτόκολλα δικτύου όπως είναι το IPv4 και IPv6

Είχαν πραγματοποιηθεί πολλές απόπειρες για την δημιουργία ενός πολυδρομικού πρωτοκόλλου (πρωτόκολλο που βοηθάει στην αποτελεσματική εξισορρόπηση της κίνησης σε πολλαπλές ζεύξεις) έχοντας ως πρότυπο το TCP ωστόσο καμία δεν ήταν ολοκληρωμένη ούτε παρουσίαζε τα απαραίτητα χαρακτηριστικά για να χρησιμοποιηθεί ευρέως σαν πρωτόκολλο δρομολόγησης ενός δικτύου. Το Πρωτόκολλο Ελέγχου Ροής της Μετάδοσης (Stream Control Transmission Protocol – SCTP) βασίστηκε στην ιδέα του Multihoming σε συνδυασμό με την δυνατότητα επανα-προώθησης σε περίπτωση αποτυχίας.

Το πρωτόκολλο SCTP [27][28] λόγω της ικανότητας του να επιτρέπει στο σύστημα να χρησιμοποιεί πολλές διεπαφές ταυτόχρονα χρησιμοποιήθηκε σε πολλά συστήματα [τα προηγούμενα χρόνια [27]. Ωστόσο λόγο των μειονεκτημάτων που έχει δεν προτιμάται πλέον στις μέρες μας. Μερικά από τα βασικότερα μειονεκτήματα που είχε και εμπόδιζαν την ομαλή λειτουργία του δικτύου είναι ότι θα έπρεπε να γίνουν κάποιες αλλαγές σε διάφορες εφαρμογές έτσι ώστε να μπορούν αν υποστηρίζουν το συγκεκριμένο πρωτόκολλο μεταφοράς και κατά δεύτερον οι ενδιάμεσοι κόμβοι (middleboxes) όπως το Firewall, που χρησιμοποιείται στην πλειοψηφία των προσωπικών υπολογιστών, δεν αναγνωρίζουν το συγκεκριμένο πρωτόκολλο με αποτέλεσμα να μπλοκάρουν τα πακέτα που καταφθάνουν χρησιμοποιώντας το.

Εξάλλου, το MPTCP έχει σχεδιαστεί με τρεις κύριους στόχους:

- Τη βελτίωση της απόδοσης (throughput): η απόδοση μιας multipath ροής πρέπει να είναι τουλάχιστον τόσο καλή όσο αυτή μιας απλής ροής TCP στο καλύτερο μονοπάτι από αυτά που είναι διαθέσιμα.
- Δικαιοσύνη: μία multi-path ροή δεν θα πρέπει να καταλάβει περισσότερη χωρητικότητα σε κάθε μία από τις διαδρομές της, από ότι οι κλασικές ροές TCP που ακολουθούν την ίδια διαδρομή.
- Εξισορρόπηση συμφόρησης: μια multipath ροή πρέπει να επιλέξει να δρομολογήσει όσο το δυνατόν περισσότερη κίνηση μακριά από τα πιο συμφορημένα μονοπάτια.

2.2.2. Κύριοι μηχανισμοί του MPTCP

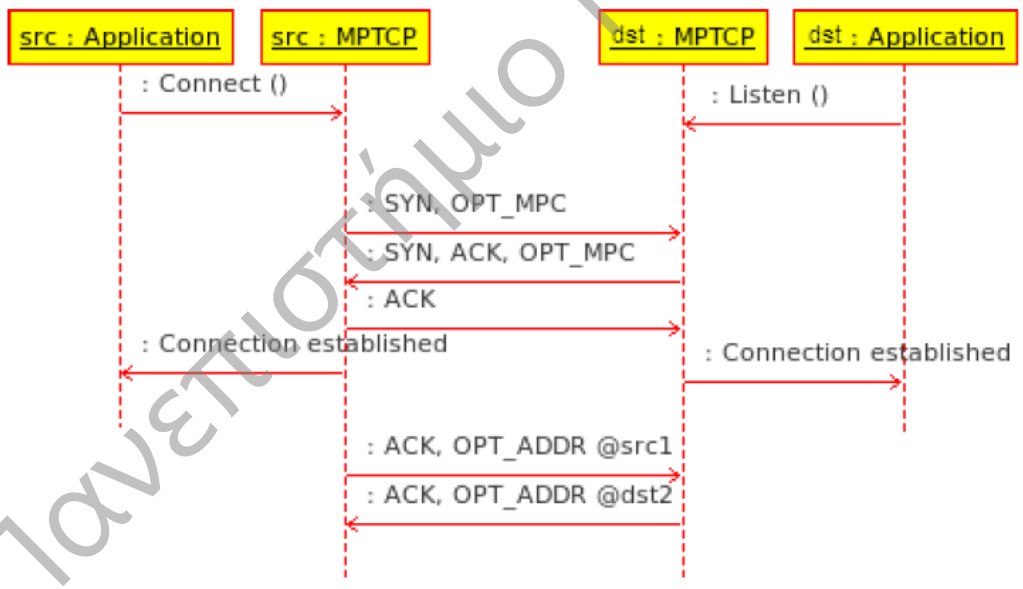
Στο MPTCP, το στρώμα μεταφοράς χωρίζεται σε δύο υποστρώματα. Το ανώτερο συγκεντρώνει τις λειτουργίες για τη διαχείρισης της σύνδεσης (δημιουργία σύνδεσης, ταξινόμηση πακέτων που λαμβάνονται κλπ…). Το χαμηλότερο υπόστρωμα διαχειρίζεται μια σειρά από υπό-ροές που κάθε μια είναι μια κλασική ροή TCP. Στο MPTCP έχουμε δυο διαφορετικά είδη από αριθμητικές ακολουθίες , μια για καθένα υπόστρωμα. Κάθε υπό-ροή έχει τη δική της αριθμητική ακολουθία, που είναι παρόμοια με το πρότυπο αύξοντα αριθμό του TCP, για τον προσδιορισμό της σειράς των πακέτων μέσα σε μια υπο-ροή. Σε επίπεδο σύνδεσης, χρησιμοποιείται μια ακόμα ακολουθία για την διάταξη των πακέτων TCP από τις διαφορετικές υπό-ροές πριν την αποστολή τους στο στρώμα εφαρμογής.

Το πρωτόκολλο MPTCP χρησιμοποιεί τις παρακάτω δηλώσεις για την ανταλλαγή πληροφοριών σηματοδότησης μεταξύ των δυο άκρων:

- MPC (Ικανότητα Multipath): χρησιμοποιείται κατά τη διάρκεια της τριμερούς χειραψίας για τη δημιουργία μιας σύνδεση TCP πολλαπλών διαδρομών.
- DATA FIN: χρησιμοποιείται για να ενημερώσει τον απομακρυσμένο σταθμό ότι η αποστολή των δεδομένων έχει ολοκληρωθεί και για να κλείσε τη σύνδεση TCP πολλαπλών διαδρομών.
- ADD / REMOVE Διεύθυνση (IPv4): χρησιμοποιούνται για την ενημέρωση του άλλου άκρου για τη διαθεσιμότητα μιας νέας διεύθυνσης ή για την αφαίρεση/αγνόηση μιας υπάρχουσας.
- JOIN: χρησιμοποιείται για να ξεκινήσει μια νέα υπό-ροή (TCP) μεταξύ δύο διευθύνσεων που δεν χρησιμοποιούνται ήδη.
- DSN: (Data Sequence Number): χρησιμοποιείται για την αντιστοίχηση των πακέτων ανάμεσα στα δυο υποστρώματα του MPTCP.

2.2.2.1. Δημιουργία Σύνδεσης

Το Σχήμα 2 απεικονίζει τη διαδικασία εγκατάστασης μιας MPTCP σύνδεσης. Μετά τη σύνδεση, η εφαρμογή-πηγή στέλνει μία Connect() κλήση, το στρώμα μεταφοράς δημιουργεί μια σύνδεση με τον server που περίμενε τη λήψη αιτήσεων σύνδεσης. Η δημιουργία της σύνδεσης είναι παρόμοια με το TCP (τριμερής χειραψία) με τη χρήση της επιλογής MPC ώστε να ενημερωθεί ο καλούμενος ότι ο ιδρυτής είναι σε θέση να ανταλλάσσει δεδομένα χρησιμοποιώντας Multipath TCP. Για να ξεκινήσει μια νέα υπό-ροή, τα δυο άκρα πρέπει να ανταλλάξουν πρόσθετες διευθύνσεις IP. Χρησιμοποιείται το option ADDR (Πρόσθεσε διεύθυνση) προκειμένου να αποσταλούν αμέσως μετά την επιτυχή εγκατάσταση του MPTCP σύνδεσης.

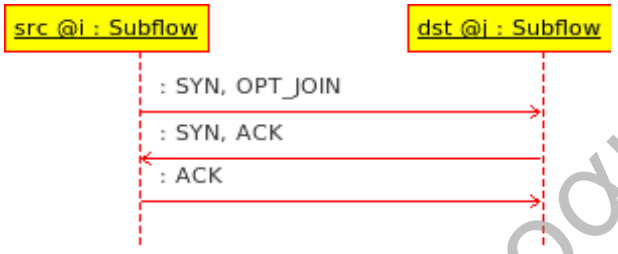


Σχήμα 2

Συνεπώς, χρησιμοποιούνται ειδικά TCP Options για το Multipath TCP, τα οποία απαιτούνται να ληφθούν υπόψη μόνο από το σύστημα προορισμού και όχι από το δίκτυο. Αν ένα εκ των δύο τελικών συστημάτων δεν υποστηρίζει το Multipath TCP, τότε το πρωτόκολλο συμπεριφέρεται ακριβώς όπως το TCP.

2.2.2.2. Αρχικοποίηση Υπό-ροής

Το Σχήμα 3 δείχνει την έναρξη μιας νέας υπό-ροής και την παρουσία ενός JOIN σε ένα tcp SYN segment. Για να μεγιστοποιηθεί η πιθανότητα η νέα υπό-ροή να ακολουθεί μια διαδρομή που είναι όσο το δυνατόν ασυσχέτιστη με την ή τις υπάρχουσες υπό-ροές, κάθε διεύθυνση IP χρησιμοποιείται μόνο από μια υπό-ροή.



Σχήμα 3

Τέλος, σημειώνεται ότι οι υπο-ροές MPTCP μπορούν να εγκατασταθούν χρησιμοποιώντας οποιαδήποτε έκδοση του Πρωτοκόλλου Διαδικτύου (IPv4 ή IPv6). Αυτή είναι μια ιδιαίτερα σημαντική ιδιότητα του Multipath TCP, καθώς διευκολύνει την μετάβαση από το IPv4 στο IPv6 και επιτρέπει σε συστήματα που διαθέτουν και τις δύο εκδόσεις να τις χρησιμοποιούν ταυτόχρονα.

2.2.3. Διαχείριση κίνησης

Το MPTCP ορίζει ξανά κάποιους μηχανισμούς TCP, έτσι ώστε να ταιριάζουν στο πλαίσιο πολλαπλών μονοπατιών. Ο έλεγχος συμφόρησης επιτρέπει στον αποστολέα να ρυθμίσει τον ρυθμό αποστολής δεδομένων σύμφωνα με τους διαθέσιμες πόρους του δικτύου. Με το MPTCP, ο έλεγχος συμφόρησης εκτελείται σε επίπεδο υπό-ροής. Κάθε υπό-ροή έχει το δικό της παράθυρο συμφόρησης. Ωστόσο τα παράθυρα συμφόρησης της κάθε υπό-ροής μιας δεδομένης σύνδεσης MPTCP μπορούν να συζευχθούν ώστε να βελτιώσουν συνολικά τις επιδόσεις. Εκτός

αυτού, στο άνω υπόστρωμα, ο δέκτης MPTCP έχει ένα ενιαίο παράθυρο λήψης το οποίο κατανέμεται μεταξύ του συνόλου των υπό-ροών που έχουν εγκατασταθεί.

Ο στόχος του MPTCP είναι να μην περιορίζεται η ταχύτητα κάποιας επιμέρους υπό-ροής. Έχουν προταθεί τέσσερις διαφορετικοί αλγόριθμοι από τους Raiciu et al [14], με στόχο την σύζευξη με διάφορους τρόπους των παράθυρων συμφόρησης των ενεργών υπό-ροών:

- Uncoupled
- Fully coupled
- Linked Increase
- RTT Compensator

Θεωρούν μια απλή επέκταση του προτύπου μηχανισμού ελέγχου συμφόρησης του TCP Reno στην περίπτωση που ο χρόνος μετά επιστροφής (Round Trip Time RTT) είναι ο ίδιος για όλα τα διαθέσιμα μονοπάτια r = 1, … , N.

Με τον αλγόριθμο Uncoupled δεν υπάρχει σύζευξη ανάμεσα στις διαφορετικές υπό-ροές, συνεπώς κάθε υπό-ροή συμπεριφέρεται σαν μια κλασική ροή TCP.

Έστω ότι w_r είναι το παράθυρο συμφόρησης στο μονοπάτι r και $W = \Sigma_r w_r$.

Αλγόριθμος Fully coupled:

$$w_r = w_r + \frac{1}{w} \text{ per ACK}$$

$$w_r = max(w_r - \frac{w}{2}, 1) \text{ per loss event}$$

Συνήθως, με τον αλγόριθμο αυτό χρησιμοποιείται είτε το ένα μονοπάτι είτε το άλλο, σπάνια και τα δυο. Αυτό το φαινόμενο ονομάζεται "flappiness". Πρακτικά, αν υπάρχουν απώλειες γίνονται συνεχώς εναλλαγές. Για να μειωθεί το φαινόμενο αυτό, προτείνεται ο παρακάτω αλγόριθμος:

Αλγόριθμος **Linked Increases:**

$$w_r = w_r + \frac{a}{w} \text{ per ACK}$$

$$w_r = \frac{w_r}{2} \text{ per loss event}$$

Για μεγαλύτερη πληρότητα, όταν το Round Trip Time δεν είναι το ίδιο σε όλα τα μονοπάτια, (γεγονός που αναμένεται να ισχύει στην γενική περίπτωση) , ο προηγούμενος αλγόριθμος βελτιώνεται ως εξής:

RTT Compensator

$$w_r = w_r + min(\frac{a}{w}, \frac{1}{w}) \text{ per ACK}$$

$$w_r = \frac{w_r}{2} \text{ per loss event}$$

2.2.4. Αναδιάταξη πακέτων

Με το κλασικό TCP, σε δίκτυα με μεγάλο jitter, δηλαδή όταν η καθυστέρηση από άκρο σε άκρο έχει μεγάλη διακύμανση, τα πακέτα πιθανότατα θα φτάσουν στον προορισμό με λάθος σειρά. Αυτό μπορεί για παράδειγμα να συμβαίνει σε ασύρματα δίκτυα όπου κινητές συσκευές μπορεί να αλλάζουν hotspot για την πρόσβαση στο Internet. Η αναδιάταξη ενός πακέτου κάνει τον δέκτη να αποκρίνεται με διπλές αναγνωρίσεις (Duplicate ACK), και αυτό μπορεί να προκαλέσει τον αποστολέα να καταλήξει στο λανθασμένο συμπέρασμα ότι υπάρχει απώλεια πακέτων.

Για να αποφύγει κανείς αυτό το πρόβλημα και να διαχωρίσει σαφώς τις απώλειες πακέτων λόγω συμφόρησης στο δίκτυο από την αναδιάταξη πακέτων λόγω jitter, πολλοί μηχανισμοί έχουν προταθεί για το TCP [29]. Μερικοί από αυτούς τους μηχανισμών προέρχονται από το IETF [31][32][33], και μερικοί από αυτά εφαρμόζονται στα λειτουργικά συστήματα [33] (ειδικά στα πιο διαδεδομένα όπως τα Windows και το Linux), όπου μπορούν να είναι ενεργά από προεπιλογή, ή όχι.

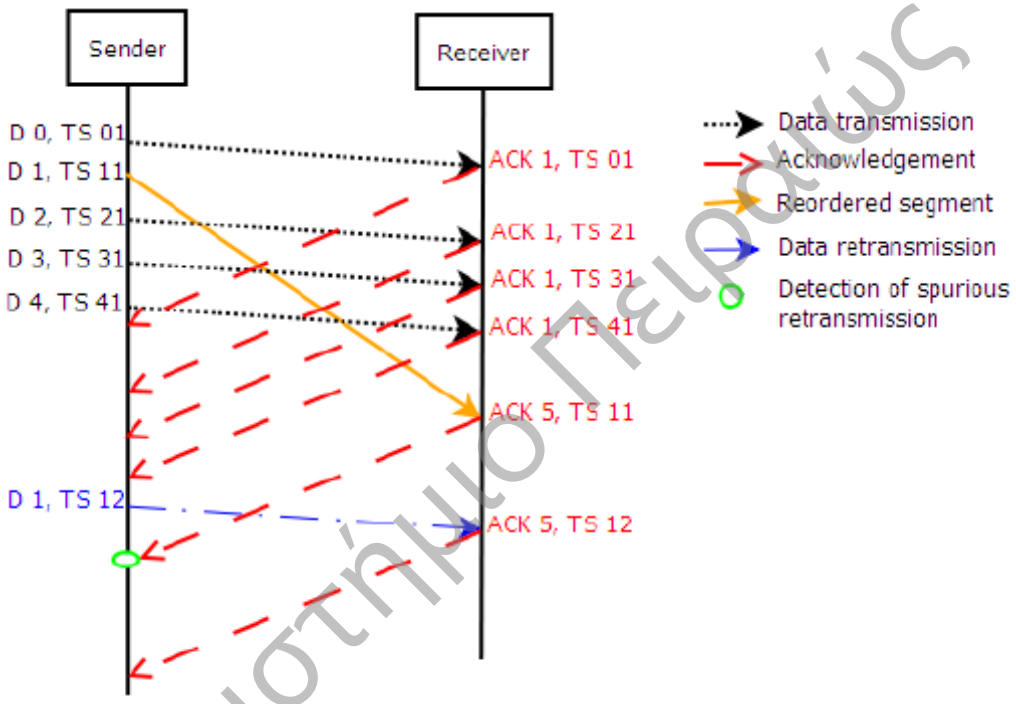
Στα πλαίσια χρήστης πολλαπλών μονοπατιών, τα πακέτα μπορούν επίσης να φθάνουν με λάθος σειρά καθώς τα διαφορετικά μονοπάτια πιθανότατα έχουν διαφορετικά χαρακτηριστικά (ειδικά η καθυστέρηση από άκρο σε άκρο), ή να παρουσιάζουν διαφορετική συμφόρηση (και συνεπώς διαφορετικές καθυστερήσεις).

Η άφιξη των πακέτων με λάθος σειρά δημιουργεί πρόβλημα στο MPTCP κατά την αναδιάταξη των πακέτων στο επίπεδο σύνδεσης, και όχι σε επίπεδο υπό-ροής καθώς οι υπό-ροές είναι ανεξάρτητες. Στο Multipath TCP προέκυψε ένας δεύτερος χώρος αριθμών ακολουθίας. Ένας χώρος αριθμών ακολουθίας σχετίζεται με τους αριθμούς ακολουθίας της υπό-ροής. Κάθε υπό-ροή διατηρεί το δικό της χώρο αριθμών ακολουθίας, και τα σημειώνει στο ίδιο πεδίο με εκείνο που χρησιμοποιείται στο TCP. Ο δεύτερος χώρος αριθμών ακολουθίας αποθηκεύεται σε ένα TCP Option, ονόματι DSS (Data Sequence Signal), που έχει οριστεί στο [FRHB12] για την μεταφορά των αριθμών ακολουθίας δεδομένων.

Στο MPTCP χρησιμοποιούνται παρόμοιοι μηχανισμοί για την αναδιάταξη των πακέτων με αυτούς που χρησιμοποιούνται στο πρωτόκολλο TCP σε ασύρματα δίκτυα. Στη συνέχεια θα παρουσιαστούν δυο από τους αλγορίθμους, ο Eifel [30][31] και ο DSACK [32].

2.2.4.1. Eifel Algorithm

Το Σχήμα 4 απεικονίζει πώς ο Eifel αλγόριθμος λειτουργεί [30][31]. Ο αποστολέας εισάγει μια επιλογή timestamp TCP σε κάθε τμήμα που αποστέλλει, και ο δέκτης εισάγει την τιμή της χρονικής σφραγίδας (timestamp) του λαμβανόμενου τμήματος στο αντίστοιχο ACK.

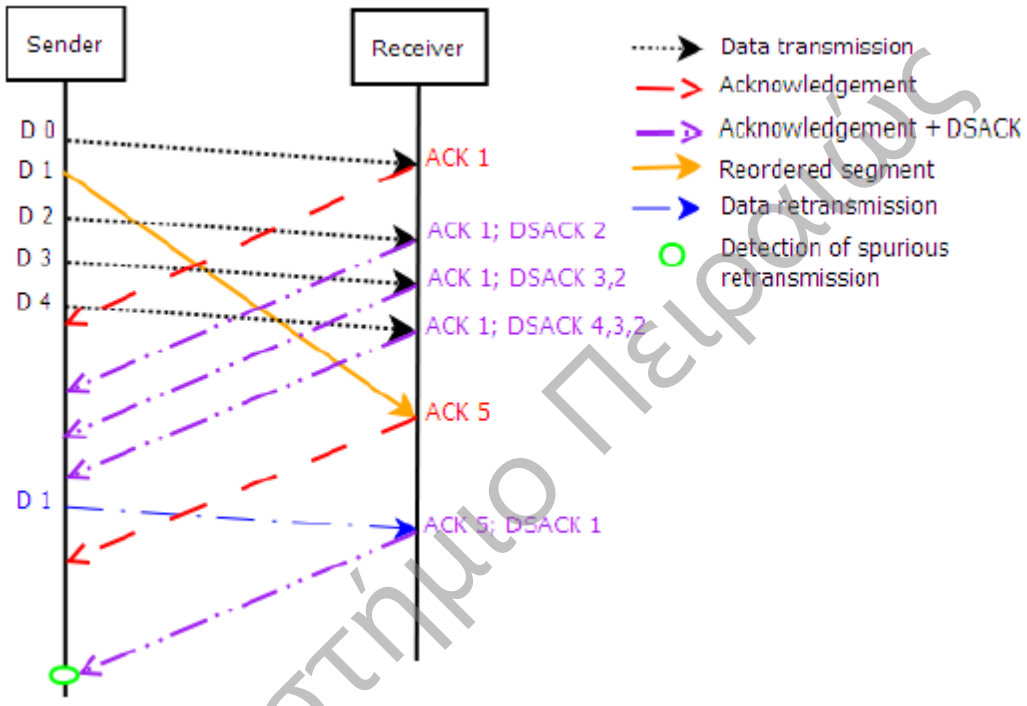


Σχήμα 4 Αλγόριθμος Eiffel

Σε περίπτωση απώλειας, ο αποστολέας αποθηκεύει τις τιμές του τρέχοντος παράθυρου συμφόρησης (cwnd) και του όριου αργής εκκίνησης (ssthresh). Στη συνέχεια, ο αποστολέας αναμεταδίδει το πακέτο που λείπει και αποθηκεύει την αξία της τρέχουσας χρονοσφραγίδας. Όταν ο αποστολέας λαμβάνει ένα ACK για πακέτο αναμετάδοσης, συγκρίνει την αποθηκευμένη τιμή timestamp με αυτήν που υπάρχει στο ACK. Εάν η πρώτη είναι μεγαλύτερη, τότε η αναμετάδοση θεωρείται ότι έγινε χωρίς λόγο και οι τιμές των cwnd και ssthresh αποκαθίστανται.

2.2.4.2. Αλγόριθμος DSACK

Το Σχήμα 5 απεικονίζεται η λειτουργία του αλγόριθμου DSACK. Αυτός ο αλγόριθμος βασίζεται στην Επιλεκτική Αναγνώριση λήψης - SACK (Selective Acknowledgement).



Σχήμα 5 Αλγόριθμος DSACK

3. Προσομοιωτής Δικτύων NS-3

3.1. Εισαγωγή

Το θέμα της διπλωματικής εργασίας αναφέρεται κυρίως στον εξομοιωτή δικτύων (network simulator) NS-3. Ο NS-3 είναι ο τρίτος στην σειρά εξομοιωτής δικτύων διακριτών γεγονότων (discrete event network simulator) και σκοπός του είναι, όπως και ο σκοπός των προηγούμενων εξομοιωτών, να μετατρέπει την λειτουργία ενός συστήματος σε μια ακολουθία διακριτών γεγονότων στο χρόνο.

Οι δημιουργοί του NS-3 ήθελαν να αναπτύξουν ένα περιβάλλον προσομοίωσης ελεύθερου λογισμικού για όλους που θα βοηθούσε να καλυφθούν οι σύγχρονες ανάγκες προσομοίωσης δικτύων, ενώ παράλληλα θα ήταν εφικτή η τροποποίηση και επέκταση του αρχικού κώδικα έτσι ώστε να μπορεί ο προσομοιωτής να προσαρμόζεται ανάλογα με το τι χρειάζονται οι χρήστες του. Αυτή την στιγμή μια μεγάλη κοινότητα ασχολείται με το ανοιχτό κώδικα του προσομοιωτή και προσπαθεί να τον βελτιώσει.

Τα νεότερα μοντέλα NS δεν αποτελούν μετεξέλιξη των προηγούμενων μοντέλων, έτσι δεν υπάρχουν και πολλές ομοιότητες μεταξύ τους. Αυτό γίνεται φανερό αν κοιτάξει κανείς τον πηγαίο κώδικα του NS-3 και NS-2, όπου και θα παρατηρήσει ότι είναι τελείως διαφορετικοί. Αυτή η διαφοροποίηση όμως δεν αποτρέπει την μεταφορά (porting) μοντέλων που έχουν ήδη δημιουργηθεί με την χρήση του NS-2 στον νέο προσομοιωτή.

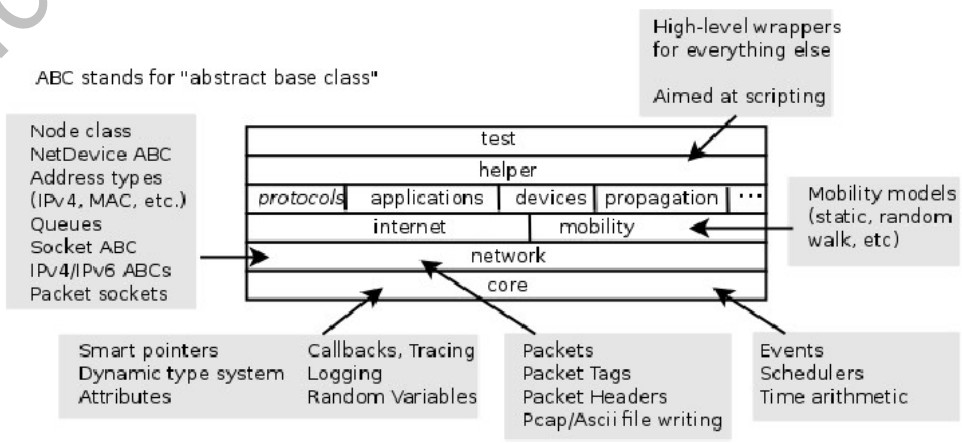
Μιας και ο NS-3 δεν βρίσκεται πολύ καιρό στην αγορά δεν βρίσκεται ακόμα στα επίπεδα ολοκλήρωσης που είναι ο NS-2, ωστόσο εξελίσσεται με γρήγορους ρυθμούς και έτσι καλύπτονται πολύ γρήγορα τα μειονεκτήματα του. Προτέρημα

είναι το ότι έχει κάποιες περαιτέρω δυνατότητες σε σχέση με τον προκάτοχο του, όπως για παράδειγμα :

- Σωστή υποστήριξη πολλαπλών διεπαφών (interfaces) σε ένα κόμβο.
- Χρήση IP και συμβατότητα με πρωτόκολλα του Internet.
- Υποστήριξη λεπτομερέστερων μοντέλων για ασύρματα δίκτυα (802.11).

3.2. Η δομή του NS-3

Ο NS-3 καθώς και τα μοντέλα που υποστηρίζει είναι γραμμένα σε C++. Είναι δομημένος σαν μια βιβλιοθήκη και μπορεί να συνδεθεί στατικά ή δυναμικά με ένα κύριο πρόγραμμα C++ που καθορίζει την τοπολογία προσομοίωσης και έτσι ξεκινά η λειτουργία του προσομοιωτή. Πέρα από αυτό το μεγαλύτερο μέρος των API (Application programming interface) που υποστηρίζει ο προσομοιωτής είναι γραμμένα σε Python.



Σχήμα 6 Η δομή του ns-3.

Συνοπτικά τα κυριότερα χαρακτηριστικά του NS-3 είναι :

Χρήση C++ και Python: οι περισσότεροι προσομοιωτές είναι γραμμένοι σε μια μόνο γλώσσα προγραμματισμού και δέχονται μοντέλα γραμμένα μόνο σε αυτή την γλώσσα. Ο NS-3 όπως αναφέρθηκε και προηγουμένως χρησιμοποιεί δύο γλώσσες προγραμματισμού, με αποτέλεσμα να επωφελείται από τα θετικά και των δύο προγραμματιστικών γλωσσών και έτσι καθίστανται πιο ευέλικτος και εύκολος στην χρήση.

Χρήση επανάκλησης (callback) : τα γεγονότα προσομοίωσης είναι κλήσεις συναρτήσεων-λειτουργιών και εκτελούνται σε ένα συγκεκριμένο χρονικό σημείο της προσομοίωσης. Σε κάθε γεγονός μπορεί να υπάρχει και μια συνάρτηση, η οποία να έχει δημιουργηθεί με την συνάρτηση επανάκλησης (μια συνάρτηση περνάει σαν παράμετρος σε μια άλλη συνάρτηση). Με την χρήση των επανακλήσεων μειώνεται ο χρόνος εξάρτησης του χρόνου μεταγλώττισης μεταξύ των αντικειμένων προσομοίωσης.

Ευέλικτος πυρήνας με βοηθητικά επίπεδα : ο NS-3 αποτελείται από ένα εύκολο προς χρήση API, επιτρέποντας έτσι στους χρήστες να τον ρυθμίσουν ανάλογα με το τι χρειάζονται. Επίσης στην κορυφή των επιπέδων, έχει ένα σύνολο βοηθητικών API που παρέχουν ευκολότερες συναρτήσεις προς χρήση.

Εξομοίωση : ο τρόπος σχεδιασμού του NS-3 καθιστά δυνατή την προσομοίωση αντικειμένων ικανών να αλληλεπιδράσουν με τον εξωτερικό κόσμο. Αυτά αποθηκεύονται ως packet byte buffers και μπορούν να σταλούν σε πραγματικά δίκτυα.

Ομοιότητες με πραγματικά μοντέλα : οι κόμβοι του NS-3 ακολουθούν την δομή δικτύωσης των Linux και έτσι παρουσιάζουν πάρα πολλές ομοιότητες στον τρόπο χρήσης και εμφάνισης. Αυτό απλοποιεί κατά πολύ τον κώδικα καθιστώντας τον πιο χρήσιμο και εφαρμόσιμο μιας και δημιουργεί πιο ρεαλιστικά μοντέλα τα οποία μπορούν να συγκριθούν με πραγματικά συστήματα.

Παραμετροποιήσεις: στον προσομοιωτή υπάρχει ένα ενσωματωμένο σύστημα παραμετροποίησης, δηλαδή ενός συστήματος που αναθέτει συγκεκριμένες

προκαθορισμένες τιμές στις παραμέτρους που θα χρησιμοποιηθούν ανάλογα με την φύση του προβλήματος. Το σύστημα αυτό αποτελείται από μια γραμμή εντολών, την τεκμηρίωση Doxygen, ένα XML-based και ένα GTK-based υποσύστημα παραμετροποίησης.

Μη ύπαρξη περιβάλλοντος ανάπτυξης (integrated Development Environment – IDE) : Ο NS-3 δεν έχει ενσωματωμένο κάποιο IDE, για να μπορεί να χρήστης να χειρίζεται, να ελέγχει, να εκτελεί, να εμφανίζει και γενικά να αλληλεπιδρά με την προσομοίωση σε ένα ενιαίο παράθυρο εφαρμογής, όπως έχουν οι προηγούμενοι προσομοιωτές.

Η τελική έκδοση του NS-3 που κυκλοφορεί είναι η NS-3.19, ωστόσο στην παρούσα εργασία θα χρησιμοποιηθεί η έκδοση 3.16. τους λόγους για αυτή την επιλογή θα τους αναλύσουμε αργότερα.

Για την ανάπτυξη του NS-3 χρησιμοποιείται το πρόγραμμα mercurial, μέσα από το οποίο ο χρήστης μπορεί να κάνει οποιοδήποτε αλλαγές και τροποποιήσεις θέλει στον πηγαίο κώδικα του NS-3. Για κάθε νέα έκδοση ή προσθήκη νέων χαρακτηριστικών διατηρείται και ένα development repository (NS-3-dev) πάνω στο οποίο γίνονται οι βασικές αλλαγές τις οποίες θα έχει η νέα έκδοση. Αξίζει επίσης να ειπωθεί ότι στην έκδοση ns-3.11, έγινε αλλαγή του τρόπου μετάφρασης του προσομοιωτή και αντίστοιχη οργάνωση του πηγαίου κώδικα, από μονολιθική βιβλιοθήκη σε modular.

Η διανομή του κώδικα κάθε έκδοσης, γίνεται με τα καθιερωμένα tarballs ή με τη χρήση του mercurial, όπου μπορεί να γίνει clone οποιουδήποτε από τα διαθέσιμα repositories.

Ο κώδικας των modules του προσομοιωτή βρίσκεται στον φάκελο src/ . Για παράδειγμα, όσον αφορά την τελευταία έκδοση ns-allinone-3.16 του NS-3 ο πηγαίος κώδικας των modules θα βρίσκεται στην διαδρομή ‘’ns-allinone-3.16/ns-3.16/src’’

Η διανομή ns-allinone-3.16, πέρα από τον NS-3, περιλαμβάνει επίσης τα παρακάτω εργαλεία:

- NetAnim (netanim-3.103/) - παραθυρικό περιβάλλον γραμμένο σε Qt, που χρησιμοποιεί xml trace αρχεία και βοηθάει στην αναπαράσταση των τοπολογιών, των πακέτων και γενικά την οπτική αναπαράσταση όλων των αντικειμένων του δικτύου.
- Network Simulation Cradle (nsc-0.5.3) – βιβλιοθήκη που επιτρέπει την χρήση TCP/IP stacks.
- PyBindGen (pybindgen-0.15.0.809) - βιβλιοθήκε γραμμένη σε Python. Παράγει Python bindings για το API του ns-3.
- Repository: ns-3-allinone - έχει Python scripts, που βοηθούν στο clone οποιουδήποτε διαθέσιμου repository (download.py).
- build.py -Βοηθάει στην διαδικασία μετάφρασης μέσω του συστήματος waf. Έτσι κάνει ευκολότερη την κατανόηση του αλγορίθμου από τον χρήστη, αφού μπορεί να ρυθμίσει το είδος της μετάφρασης (debug, optimized), να επιλέξει οποιαδήποτε modules θέλει κ.λ.π.

3.3. Βασικές δομές του ns-3

Για την προσομοίωση ενός δικτύου υπολογιστών θα πρέπει να σχεδιαστεί η τοποθεσία του κάθε υπο-τμήματος του συγκεκριμένου δικτύου. Στην επιστήμη των Μαθηματικών απεικόνιση αυτή γίνεται μέσα από γράφους. Ο γράφος G=(V,E) αποτελείται από ένα σύνολο κόμβων (V) που συνδέονται μεταξύ τους με ένα σύνολο ακμών (E).

Στην απεικόνιση του δικτύου μέσα από έναν γράφο, ο γράφος αναπαριστά την τοπολογία, οι κόμβοι τους υπολογιστές και οι ακμές την σύνδεση μεταξύ των υπολογιστών, μέσω ενός καναλιού επικοινωνίας. Σκοπός του δικτύου είναι η

μεταφορά δεδομένων και πληροφοριών μεταξύ των υπολογιστών, άρα και των κόμβων, οπότε θα πρέπει να υπάρχει τρόπος αναπαράστασης της κίνησης στον γράφο όπως και στο δίκτυο.

Παρακάτω παρατίθενται κάποιες από τις βασικότερες δομές του NS-3 και τον καθιστούν ικανό να παραστήσει έναν γράφο.

Κόμβος (Node)

Αποτελεί την βασικότερη υπολογιστική συσκευή του NS-3 και μέσα από την C++ αναπαριστάται με την κλάση Node. Πάνω του, μπορούν να προστεθούν και άλλες δομές που περιγράφουν τη σύνδεση του στο δίκτυο, καθώς και τη λειτουργία του.

Κανάλι (Channel)

Αποτελεί το κανάλι επικοινωνίας πάνω στο οποίο μπορεί να συνδεθεί ένας κόμβος και αυτός με την σειρά του να συνδεθεί με ένα επόμενο και χρησιμοποιείται για την μετάδοση δεδομένων και πληροφοριών. Στην C++ η κλάση Channel περιλαμβάνει τρόπους διαχείρισης αντικειμένων και μεθόδους σύνδεσης των κόμβων. Μερικοί από τους βασικότερους τύπους καναλιών είναι οι :

- Point- to point
- Csma
- Wifi
- wimax

Συσκευή δικτύου (Net Device)

Αποτελεί τη συσκευή που συνδέει κάθε κόμβο με το κανάλι. Αυτό που κάνει είναι να εγκαθίστανται σε έναν κόμβο έτσι ώστε να μπορέσει να συνδεθεί με τους υπόλοιπους κόμβους μέσω του καναλιού. Δεν είναι ανάγκη κάθε κόμβος να συνδέεται με ένα μόνο κανάλι. Με την χρήση πολλαπλών συσκευών δικτύου είναι εφικτή η σύνδεση του με περισσότερα κανάλια. Συμπεριλαμβάνει, τη συσκευή που συνδέει έναν υπολογιστή στο δίκτυο (Network Interface Card - NIC), καθώς και τους απαραίτητους drivers της συσκευής που επιτρέπουν στο λειτουργικό σύστημα του

υπολογιστή να χρησιμοποιήσει τη συσκευή. Έτσι για κάθε τύπο καναλιού, υπάρχει το αντίστοιχο Net Device: PointToPointNetDevice, CsmaNetDevice, WifiNetDevice, WimaxNetDevice κ.α.

Εφαρμογή (Application)

Όπως φαίνεται και από το όνομα της αποτελεί την εφαρμογή που τρέχει και δημιουργεί τη δραστηριότητα, ή με άλλα λόγια την κίνηση και ανταλλαγή πληροφοριών ενός δικτύου, που πρόκειται να προσομοιωθεί.

Βοηθοί τοπολογίας (Topology Helpers)

Η προσομοίωση και απεικόνιση ενός δικτύου παρουσιάζει πολλές ενέργειες που είναι περίπλοκες. Δουλεία των βοηθών τοπολογίας είναι να παρέχουν στον χρήστη εφόδια και έτοιμες εργασίες για να τον διευκολύνουν και να κάνουν την έκβαση της προσομοίωσης πιο εύκολη. Μερικές από τις ενέργειες τους είναι :

- Κανονισμός σύνδεσης μεταξύ κόμβων.
- Κανονισμός σύνδεσης μεταξύ κόμβων και καναλιών
- Κανονισμός σύνδεσης μεταξύ κόμβων και συσκευών δικτύου.
- Διευθυνσιοδότηση

3.4. NS-3 modules

Η έκδοση 3.16 του ns-3, αποτελείται από 40 modules. Τα οποία είναι τα :

antenna	energy	propagation
aodv	flow-monitor	spectrum
applications	internet	stats
bridge	lte	tap-bridge
brite	mesh	test
buildings	mobility	tools
click	mpi	topology-read
config-store	netanim	uan
core	network	virtual-net-device
csma	nix-vector-routing	visualizer
csma-layout	olsr	wifi
dsdv	openflow	wimax
dsr	point-to-point	
emu	point-to-point-layout	

Σχήμα 7 ns-3 modules

Το core module, είναι ο πυρήνας του προσομοιωτή και παρέχει διάφορες σημαντικές λειτουργίες. Μερικές από τις βασικότερες είναι η παραγωγή τυχαίων αριθμών και η παρακολούθηση του χρόνου εξομοίωσης. Μια επίσης από τις σημαντικότερες δυνατότητες του είναι η ικανότητα του να λειτουργεί σε πραγματικό χρόνο και έτσι μπορεί και χρησιμοποιεί πραγματικά network stacks.

Από την άλλη υπάρχει και το network module που χρησιμοποιείται για την δημιουργία διάφορων τύπων πακέτων, κόμβων ,Net devices και ουρών αναμονής πακέτων (τύπου Drop Tail (FIFO) και Early Detection (RED)).

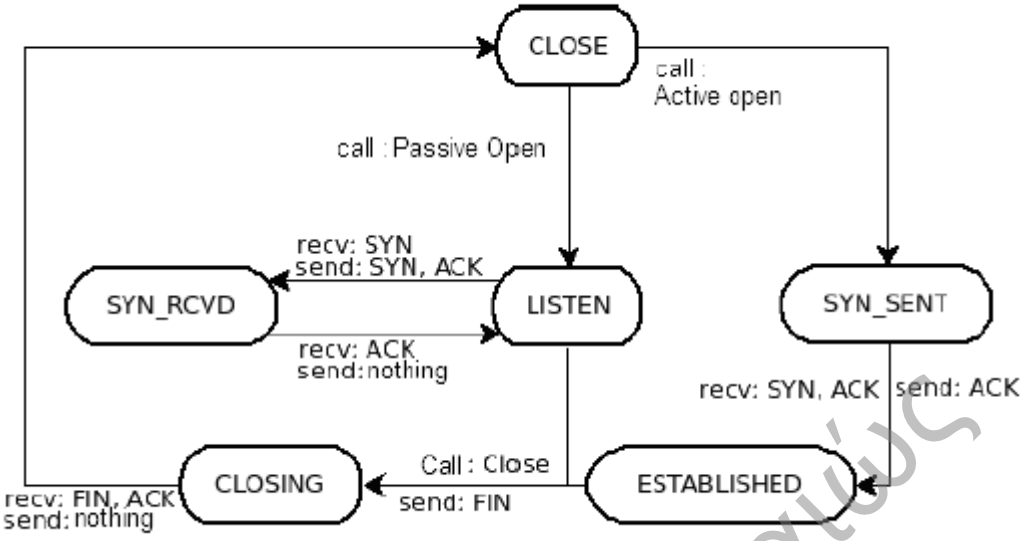
Πέρα από αυτά υπάρχουν πολλά ακόμα modules που το καθένα χρησιμοποιείται για ξεχωριστό σκοπό. Μερικά από αυτά είναι τα:

- Module point-to-point : βοηθάει στην δημιουργία point-to-point καναλιών
- Module point-to-point-layout : βοηθάει στην δημιουργία point-to-point τοπολογιών (dumbbell, grid)με ευκολότερο τρόπο.
- Module csma : βοηθάει στην δημιουργία csma καναλιών.
- Module bridge : βοηθάει στην σύνδεση πολλών csma καναλιών
- Module emu : βοηθάει στην αλληλεπίδραση του NS-3 με πραγματικά δίκτυα
- Module brite : βοηθάει στην δημιουργία ρεαλιστικών διαδικτυακών τοπολογιών με το περιβάλλον BRITE.
- Module antenna : βοηθάει στην υλοποίηση διαγραμμάτων ακτινοβολίας

3.5. Υλοποίηση του MPTCP στον NS-3

Στα πλαίσια της εργασίας αυτής, χρησιμοποιήσαμε τον κώδικα που περιγράφεται στο έργο [23]. Ο κώδικας αυτός έχει αναπτυχθεί στο πλαίσιο του έργου mptcp-NS3 και είναι συμβατός με το NS-3, έκδοση 3.6. Το έργο mptcp-ns3 επικεντρώθηκε στην ανάπτυξη μιας εφαρμογής των πολλαπλών TCP για ns-3 για ερευνητικούς σκοπούς και υλοποιεί ολόκληρο το στρώμα μεταφοράς του Multipath TCP σε ns-3.

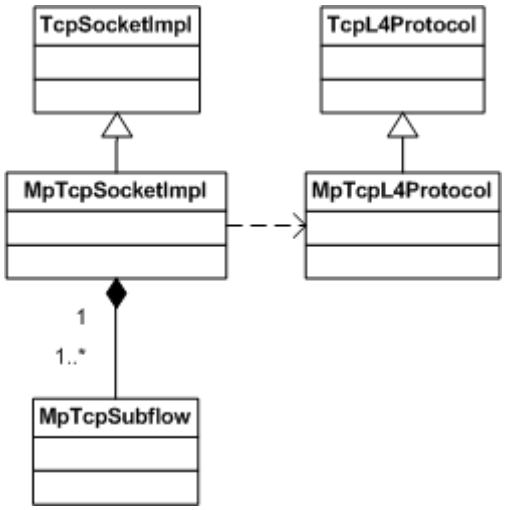
Πιο συγκεκριμένα, στο Module αυτό έχουν υλοποιηθεί οι καταστάσεις του MPTCP που εμφανίζονται στο παρακάτω σχήμα.



Σχήμα 8 Καταστάσεις σύνδεσης του MPTCP

Η δομή της υλοποίησης του MPTCP στον NS3 είναι η ακόλουθη:

- MpTcpSocketImpl είναι μια υποκλάση της κλάσης TopSocketImpl. Προσφέρει στην εφαρμογή στρώματος το MPTCP API (σύνδεση, bind, κ.λπ.) για τη διαχείριση πολλαπλών συνδέσεων TCP. Υλοποιεί επίσης τους αλγόριθμοθς επαναταξινόμησης πακέτων περιγράφηκαν στην προηγούμενη ενότητα.
- Η MpTcpL4Protocol είναι μια υποκλάση του NS-3 TcpL4Protocol. Πρόκειται για ένα Interface μεταξύ του στρώματος πολλαπλών διαδρομών μεταφοράς και του στρώματος δικτύου.
- Η MpTcpSubflow αντιπροσωπεύει μια υπό-ροή στο MPTCP
- Η MpTcpHeader είναι μια υποκλάση της TcpHeader. Ένα αντικείμενο αυτής της κλάσης είναι μια κεφαλίδα TCP που μπορεί να χειριστεί επιλογές όπως TimeStamp, MPC, κλπ.



Σχήμα 9 Διάγραμμα των βασικών κλάσεων του MPTCP

4. Σενάρια προσομοίωσης

4.1. Εισαγωγή

Για τις ανάγκες της εργασίας, δημιουργήσαμε ένα σενάριο από δύο κόμβους ανταλλαγή περιεχομένου πολυμέσων. Για τη διεξαγωγή των πειραμάτων, χρησιμοποιήσαμε τον κώδικα που περιγράφεται στο έργο [23]. Ο κώδικας παράγει ίχνη δικτύου (αρχεία pcap) που μπορούν να αναλυθούν με εξειδικευμένο λογισμικό, όπως το Wireshark [34]. Επιπλέον, ο κώδικας παράγει ένα αρχείο κειμένου που περιέχει πληροφορίες σχετικά με το μέγεθος του τρέχοντος παραθύρου σε κάθε υπό-ροή του MPTCP, το throughput σε κάθε ροή και το συνολικό throughput. Παρακάτω παρουσιάζουμε ένα απόσπασμα από αυτό το αρχείο.

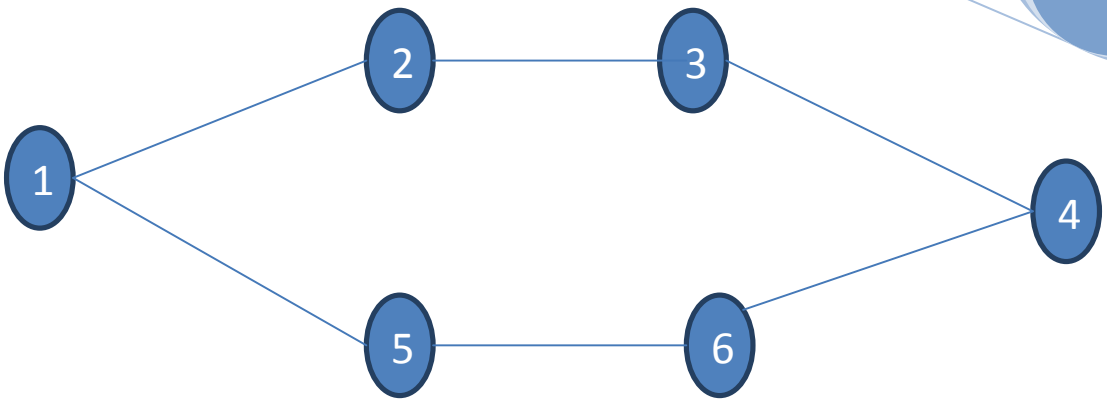
```
    1.20083 SentSegment ->   localToken 313 Subflow 0
DataTxSeq 1 SubflowTxSeq 734 SubflowRxSeq 598 Data 1400
unAcked Data 0 octets

    1.20083 SentSegment ->   localToken 313 Subflow 1
DataTxSeq 1401 SubflowTxSeq 10832 SubflowRxSeq 2 Data
1399 unAcked Data 1400 octets

    1.41286 Cumulative_ACK ->  localToken 313 Subflow 0
Data_ACK 1401 Subflow_ACK 2134 highestAckedData 733
maxSequenNumber 2133 AckedData 1400 unAckedData 1400

    1.41286 MpTcpSocketImpl ->  localToken 313 Subflow 0:
RTT 0.212032 Moving cwnd from 1 to 2 Throughput 105644
GlobalThroughput 113111 Efficacity 0.0754074 delay 0.1
alpha 0 Sum CWND (2)

    1.41286 SentSegment ->   localToken 313 Subflow 0
DataTxSeq 2800 SubflowTxSeq 2134 SubflowRxSeq 599 Data
1400 unAcked Data 0 octets
```

```
    1.41286 SentSegment ->  localToken 313 Subflow 0
DataTxSeq 4200 SubflowTxSeq 3534 SubflowRxSeq 599 Data
1400 unAcked Data 1400 octets

    1.42488 Cumulative_ACK ->  localToken 313 Subflow 1
Data_ACK 2800 Subflow_ACK 12231 highestAckedData 10830
maxSequenNumber 12230 AckedData 1400 unAckedData 1400

    1.42488 MpTcpSocketImpl ->  localToken 313 Subflow 1:
RTT 0.224048 Moving cwnd from 1 to 2 Throughput 99978.6
GlobalThroughput 205623 Efficacity 0.137082 delay 0.1
alpha 0 Sum CWND (3)

    1.42488 SentSegment ->  localToken 313 Subflow 1
DataTxSeq 5600 SubflowTxSeq 12231 SubflowRxSeq 3 Data
1400 unAcked Data 0 octets

    1.42488 SentSegment ->  localToken 313 Subflow 1
DataTxSeq 7000 SubflowTxSeq 13631 SubflowRxSeq 3 Data
1400 unAcked Data 1400 octets
```

4.2. Τοπολογία δικτύου και πειράματα

Η τοπολογία του δικτύου των πειραμάτων απεικονίζεται στο επόμενο σχήμα. Ο κόμβος 4 είναι ένας διακομιστής πολυμέσων και ο κόμβος 1 είναι ο πελάτης. Κάθε ένας από τις παραπάνω κόμβους έχει δύο διασυνδέσεις που μπορούν να χρησιμοποιηθούν ταυτόχρονα από το MPTCP. Ο κόμβος 1 έχει τα Interfaces 2 και 5 ενώ ο κόμβος 4 έχει τα Interfaces 3 και 6.

Σε κάθε σενάριο πρέπει να αποστέλλονται δεδομένα από τον διακομιστή στον πελάτη με ένα σταθερό ρυθμό δεδομένων Constant Bit Rate (CBR). Ο στόχος είναι να μοιράζεται η κίνηση έτσι από το MPTCP ανάμεσα στις δύο διαφορετικές διαδρομές κατά τέτοιο τρόπο ώστε να επιτυγχάνεται ο ζητούμενος συνολικός ρυθμός δεδομένων.



Σχήμα 10 Τοπολογία πειραμάτων

Στα πλαίσια της εργασίας, έγιναν πειράματα με διάφορες τιμές για το ζητούμενο CBR και με διαφορετική χωρητικότητα στα δυο διαφορετικά μονοπάτια. Επίσης, προκειμένου να εξεταστεί η λειτουργία του MPTCP όταν η κατάσταση του κάθε μονοπατιού δεν είναι σταθερή, σε διάφορες περιπτώσεις, εισάχθηκε μεταβλητό delay σε κάποια διαδρομή.

Συγκεντρωτικά, τα αποτελέσματα των πειραμάτων φαίνονται στον παρακάτω πίνακα. Στον πίνακα αυτό εμφανίζονται τα μέσα throughput για τις περιπτώσεις χωρίς μεταβλητή καθυστέρηση στις ζεύξεις δικτύου. Στις περιπτώσεις αυτές ο ρυθμός απόδοσης είχε πολύ μικρή διακύμανση γύρω από το μέσο όρο. Στις περιπτώσεις με μεταβλητό delay υπήρχε μεγάλη διακύμανση και η αποτύπωση του μέσου όρου δεν έχει νόημα.

Χωρητικότητα 1-2-3-4 (Mbps)	Χωρητικότητα 1-5-6-4 (Mbps)	Ζητούμενο CBR (Mbps)	Μέσο Throughput 1-2-3-4 (Mbps)	Μέσο Throughput 1-5-6-4 (Mbps)
1.0	0.5	0.25	0.125	0.125
1.0	0.5	0.50	0.255	0.255
1.0	0.5	0.75	0.375	0.375
1.0	0.5	1.00	0.5	0.5

1.0	0.5	1.25	0.96	0.29
1.0	0.5	1.50	0.96	0.45
1.0	0.75	0.25	0.125	0.125
1.0	0.75	0.50	0.255	0.255
1.0	0.75	0.75	0.375	0.375
1.0	0.75	1.00	0.5	0.5
1.0	0.75	1.25	0.625	0.625
1.0	0.75	1.50	0.78	0.72
1.0	1.0	0.25	0.125	0.125
1.0	1.0	0.50	0.255	0.255
1.0	1.0	0.75	0.375	0.375
1.0	1.0	1.00	0.5	0.5
1.0	1.0	1.25	0.625	0.625
1.0	1.0	1.50	0.75	0.75
1.0	1.25	0.25	0.125	0.125
1.0	1.25	0.50	0.255	0.255
1.0	1.25	0.75	0.375	0.375
1.0	1.25	1.00	0.5	0.5
1.0	1.25	1.25	0.625	0.625
1.0	1.25	1.50	0.75	0.75
1.0	1.50	0.25	0.125	0.125
1.0	1.50	0.50	0.255	0.255
1.0	1.50	0.75	0.375	0.375
1.0	1.50	1.00	0.5	0.5
1.0	1.50	1.25	0.625	0.625

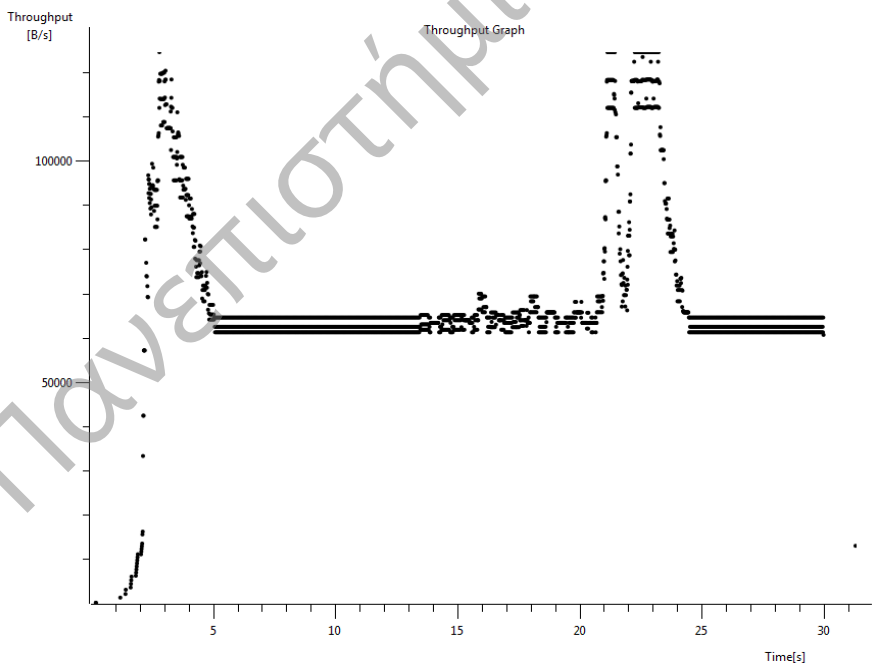
1.0	1.50	1.50	0.75	0.75

4.3. Χαρακτηριστικά σενάρια λειτουργίας

Στη συνέχεια θα παρουσιάσουμε αναλυτικά μερικά από τα παραπάνω σενάρια.

Στο πρώτο σενάριο, η διαδρομή 1-2-3-4 έχει χωρητικότητα 1.0Mbps, ενώ η διαδρομή 1-5-6-4 έχει 0.5Mbps χωρητικότητα. Ο ρυθμός δεδομένων που ζητείται είναι 1Mbps. Η κίνηση χωρίζεται σε δύο ροές περίπου 500Kbps η κάθε μία.

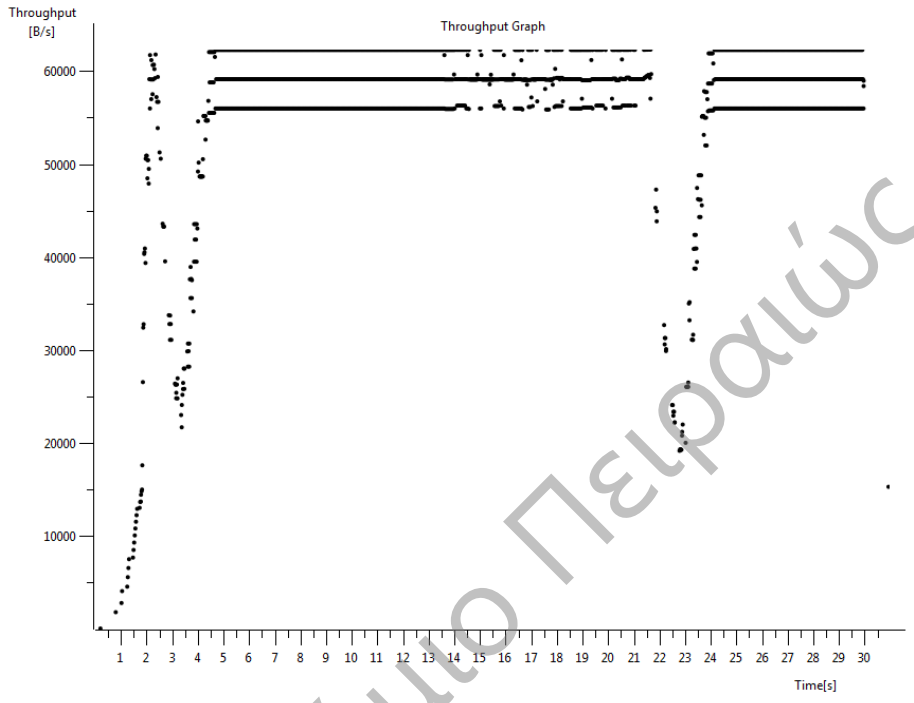
Παρακάτω παρουσιάζουμε το διάγραμμα απόδοσης της κάθε διαδρομής.



Σχήμα 11 Η ρυθμαπόδοση στο μονοπάτι 1-2-3-4

Κατά τη διάρκεια της μεταφοράς δεδομένων, το ποσό της κίνησης είναι ισορροπημένο μεταξύ των δύο διαφορετικών διαδρομών. Σε δύο περιπτώσεις, στη

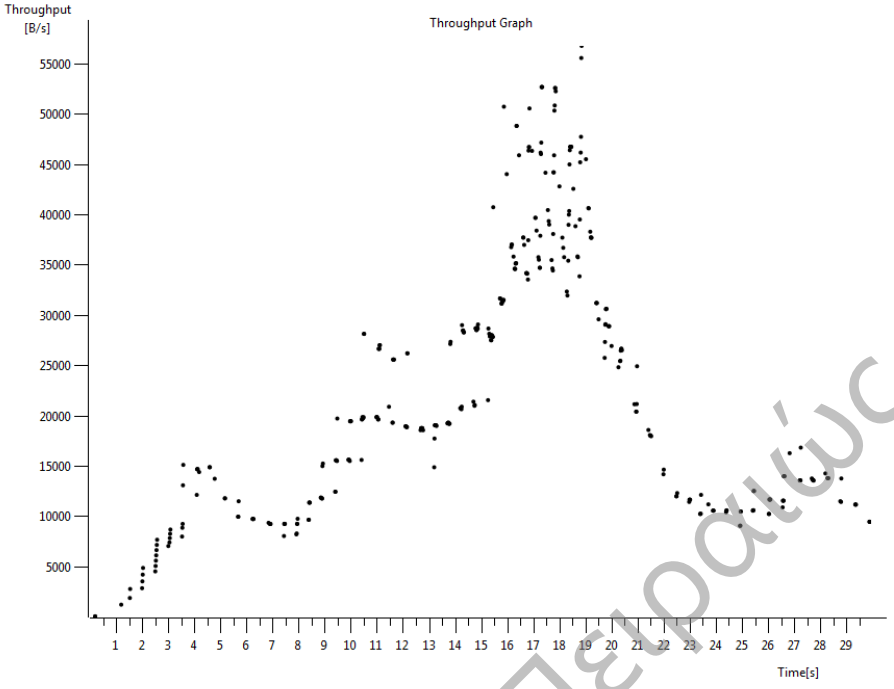
διαδρομή 1-5-6-4 συμβαίνει απώλεια πακέτων που έχει ως αποτέλεσμα τη μείωση της κυκλοφορίας που αποστέλλονται μέσω αυτής της διαδρομής και παράλληλα την αύξηση της κυκλοφορίας στην πρώτη διαδρομή, ούτως ώστε να αντισταθμιστεί η απώλεια και να διατηρηθεί σταθερό το συνολικό throughput.



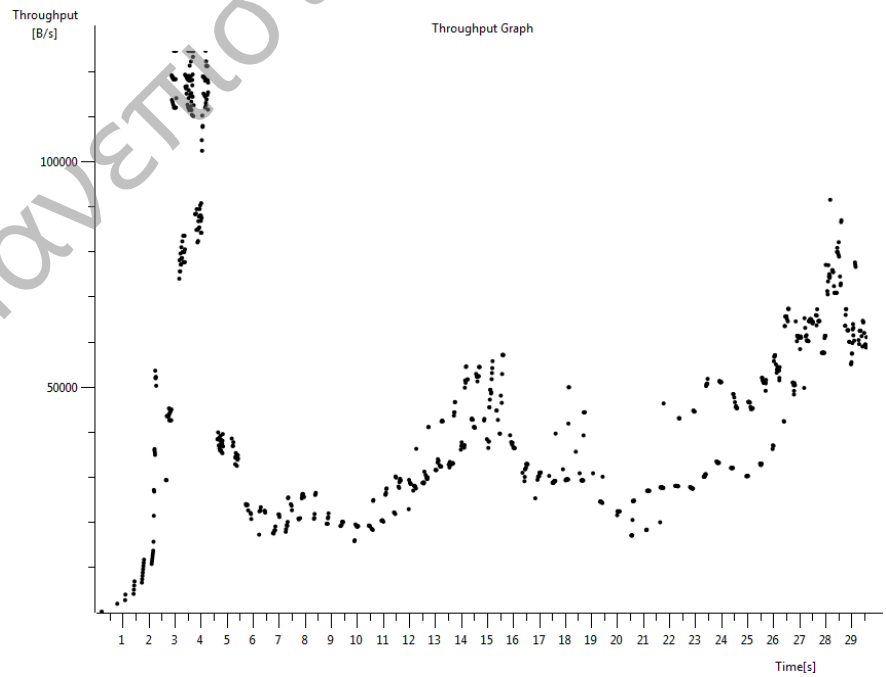
Σχήμα 12 Η ρυθμαπόδοση στο μονοπάτι 1-5-6-4

Στο δεύτερο σενάριο, ο απαιτούμενος ρυθμός δεδομένων είναι 1.2Mbps. Σε αυτό το σενάριο, η διαδρομή δικτύου 1-2-3-4 έχει 1.5Mbps χωρητικότητα, ενώ η διαδρομή 1-5-6-4 έχει 1.0Mbps χωρητικότητα. Η κίνηση θα πρέπει να χωριστεί σε δύο ροές περίπου 600Kbps. Ωστόσο, η διαδρομή του δικτύου 1-5-6-4 υπόκειται σε μεταβλητή καθυστέρηση του δικτύου και η αντίστοιχη υπό-ροή δεν μπορεί να διατηρήσει ένα σταθερό παράθυρο TCP.

Παρακάτω παρουσιάζουμε το διάγραμμα απόδοσης της κάθε διαδρομής. Παρατηρούμε ότι η κίνηση στις δύο διαφορετικές διαδρομές παρουσιάζει μεγάλες αποκλίσεις. Λόγω των μεταβαλλόμενων υποκείμενων χαρακτηριστικών κάθε μονοπατιού, η κατανομή της κυκλοφορίας ανάμεσα στις δύο υπο-ροές αλλάζει συνεχώς. Ωστόσο, η συνολική απόδοση είναι σταθερή, εξασφαλίζοντας ένα σταθερό ρυθμό δεδομένων μεταξύ των δύο άκρων.



Σχήμα 13 Η ρυθμαπόδοση στο μονοπάτι 1-2-3-4



Σχήμα 14 Η ρυθμαπόδοση στο μονοπάτι 1-5-6-4

5. Συμπεράσματα - Επίλογος

Στις μέρες μας οι κινητές συσκευές έχουν συχνά περισσότερες από μία διεπαφές δικτύου (network interfaces). Για παράδειγμα , οι φορητοί υπολογιστές έχουν συνήθως τουλάχιστον τόσο ενσύρματη (Ethernet) και μια ασύρματη (WiFi) κάρτα δικτύου. Ομοίως τα smartphones και tablet PCs μπορούν να έχουν πρόσβαση στο Internet είτε μέσω WiFi είτε μέσω του δικτύου κινητής τηλεφωνίας (UMTS ή 3G +).

Ένα άλλο γεγονός είναι ότι οι φορείς εκμετάλλευσης δικτύων έχουν συνήθως εις διπλούν τουλάχιστον τις σημαντικές / κεντρικές ζεύξεις δικτύου και τον αντίστοιχο εξοπλισμό με σκοπό την προστασία των δικτύων τους από αστοχίες υλικού. Επιπλέον, τα δίκτυα κορμού έχουν σε γενικές γραμμές τοπολογία mesh, δηλαδή είναι διασυνδεδεμένα όλα μεταξύ τους. Στο πλαίσιο αυτό, είναι πολύ πιθανό να υπάρχουν πολλά μονοπάτια ανάμεσα σε δύο άκρα του δικτύου. Συνεπώς προέκυψε η ιδέα να χρησιμοποιούνται ταυτόχρονα πολλά μονοπάτια, με σκοπό να βελτιωθεί η αντοχή και απόδοση των συνδέσεων από άκρο σε άκρο. Τέτοιες συνδέσεις πολλαπλών διαδρομών μπορεί να πετύχουν εξισορρόπηση του φορτίου ανάμεσα σε διαφορετικά μονοπάτια (load balancing), μεταφέροντας δυναμικά και αυτόματα την δικτυακή κίνηση από συμφορημένες ή γενικότερα προβληματικές συνδέσεις σε άλλες χωρίς προβλήματα.

Σχετικά πρόσφατα δημιουργήθηκε μια ομάδα εργασίας του IETF για τον καθορισμό ενός πρωτόκολλου πολλαπλών διαδρομών στο στρώμα μεταφορών το οποίο ονομάζεται Multi-path TCP (MPTCP). Το MPTCP είναι επέκταση του πρωτοκόλλου ελέγχου μετάδοσης (TCP), το πρωτόκολλο μεταφοράς που χρησιμοποιείται περισσότερο στο διαδίκτυο, με στόχο να μπορεί να χειρίζεται πολλαπλές υποσυνδέσεις σε διαφορετικά μονοπάτια ανάμεσα σε δύο άκρα.

Στα πλαίσια της εργασίας αυτής έγινε μελέτη του πρωτοκόλλου MPTCP, και προσομοίωση με τη χρήση του προσομοιωτή NS-3. Έγινε χρήση του NS3 για την προσομοίωση σεναρίων και την εξαγωγή αριθμητικών αποτελεσμάτων.

Μελετώντας τα αριθμητικά αποτελέσματα από τα πειράματα, καταλήγουμε στα παρακάτω συμπεράσματα:

Το MPTCP επιτρέπει την αντίστροφη πολυπλεξία των διαθέσιμων μονοπατιών και συνεπώς, αυξάνει την απόδοση του TCP στο άθροισμα όλων των διαθέσιμων ζεύξεων δικτύου. Αυτό έχει τα εξής πλεονεκτήματα:

- Προσφέρει network redudancy (ακόμα και αν πέσει κάποια ζεύξη, η κίνηση μεταφέρεται στις άλλες διαθέσιμες ζεύξεις έτσι ώστε να διατηρηθεί αν είναι δυνατόν ο ζητούμενος ρυθμός δεδομένων.
- Προσφέρει Load balancing ανάμεσα στις διάφορες ζεύξεις. Η κίνηση μοιράζεται ανάμεσα στις διαθέσιμες ζεύξεις. Το πρωτόκολλο έχει τάση να προτιμά τις ζεύξεις με λιγότερη συμφόρηση (μεγαλύτερο παράθυρο TCP).

Το MPTCP είναι ιδιαίτερα χρήσιμο στο πλαίσιο των ασύρματων δικτύων . Μια συσκευή που χρησιμοποιεί το Wi-Fi και το δίκτυο κινητής τηλεφωνίας είναι μία τυπική περίπτωση χρήσης. Πέρα από το θέμα του διαμοιρασμού του φορτίου, ζεύξεις δικτύου μπορούν να προστεθούν ή να πέσουν καθώς ο χρήστης κινείται μέσα ή έξω από την κάλυψη διαφορετικών δικτύων, χωρίς να διαταράσσουν τη σύνδεση και τη συνολική ρυθμαπόδοση από άκρο-προς-άκρο. Συνεπώς, το πρόβλημα του Link handover που παρουσιάζεται στις φορητές ασύρματες συσκευές, λύνεται αφαιρετικά στο στρώμα μεταφοράς, χωρίς ειδικούς μηχανισμούς στο δίκτυο ή επίπεδο σύνδεσης.

Εξάλλου, οι υπο-ροές MPTCP μπορούν να εγκατασταθούν χρησιμοποιώντας οποιαδήποτε έκδοση του Πρωτοκόλλου Διαδικτύου (IPv4 ή IPv6). Αυτή είναι μια ιδιαίτερα σημαντική ιδιότητα του Multipath TCP, καθώς διευκολύνει την μετάβαση από το IPv4 στο IPv6 και επιτρέπει σε συστήματα που διαθέτουν και τις δύο εκδόσεις να τις χρησιμοποιούν ταυτόχρονα.

Τέλος, ένα ιδιαίτερα σημαντικό πλεονέκτημα του MPTCP είναι ότι σε αντίθεση με παρόμοια πρωτόκολλα όπως το SCTPC, δεν απαιτεί αλλαγές στο δίκτυο. Μονάχα τα δυο άκρα αρκεί να υποστηρίζουν το MPTCP, κανένας ενδιάμεσος κόμβος δεν χρειάζεται αναβάθμιση, καθώς κάθε υπο-ροή συμπεριφέρεται ως μια κανονική σύνδεση TCP. Τέλος, αν κάποιο από τα δυο άκρα δεν υποστηρίζει MPTCP, τότε δημιουργείται μια απλή TCP σύνδεση χωρίς να προκύψει σφάλμα (backwards compatibility).

Παράρτημα - Κώδικας

```
#include <iostream>
#include <sstream>
#include <string>
#include <vector>

#include "ns3/core-module.h"
#include "ns3/simulator-module.h"
#include "ns3/node-module.h"
#include "ns3/helper-module.h"
#include "ns3/wifi-module.h"
#include "ns3/mobility-module.h"



#include "ns3/mp-internet-stack-helper.h"
#include "ns3/mp-tcp-packet-sink.h"
#include "ns3/mp-tcp-l4-protocol.h"
#include "ns3/mp-tcp-socket-impl.h"
#include "ns3/point-to-point-channel.h"
#include "ns3/point-to-point-net-device.h"

/* Multipath Network Topology

     lan 10.1.1.0
      ____________
     /            \
   n1              n2
     \____________/

     lan 10.1.2.0
*/
```

```
using namespace ns3;

NS_LOG_COMPONENT_DEFINE("FirstMultipathToplogy");

//maximum data rate
static const double dataRate = 1500000.0  ;
//path capacities
static const string path0Capacity = "1.0Mbps";
static const string path1Capacity = "1.5Mbps";

//simulation (data transmition) duration
static const double simDuration    = 30.0;
// The number of bytes to send in this simulation.
static const uint32_t totalTxBytes = 100000000;
static const uint32_t sendBufSize  = 14000; //2000000;
static const uint32_t recvBufSize  = 2000; //2000000;
static uint32_t currentTxBytes     = 0;



Ptr<Node> client;
Ptr<Node> server;



// Perform series of 1040 byte writes (this is a multiple
of 26 since
// we want to detect data splicing in the output stream)
static const uint32_t writeSize = sendBufSize;
uint8_t data[totalTxBytes];
Ptr<MpTcpSocketImpl> lSocket    = 0;
/*
PointToPointHelper fstP2Plink;
PointToPointHelper sndP2Plink;
PointToPointHelper trdP2Plink;
*/


void    StartFlow    (Ptr<MpTcpSocketImpl>,    Ipv4Address,
uint16_t);
void WriteUntilBufferFull (Ptr<Socket>, unsigned int);
```

```
    void connectionSucceeded(Ptr<Socket>);
    void connectionFailed(Ptr<Socket>);

    void HandlePeerClose (Ptr<Socket>);
    void HandlePeerError (Ptr<Socket>);
    void CloseConnection (Ptr<Socket>);

    int connect(Address &addr);
    void variateDelay(PointToPointHelper P2Plink);

    static void
    CwndTracer (double oldval, double newval)
    {
      NS_LOG_INFO ("Moving cwnd from " << oldval << " to " << newval);
    }

    void debug(string text) {
     string input;
     cout << text;
     getline(cin, input);

    }



    int main(int argc, char *argv[])
    {
        bool verbose;
        CongestionCtrl_t cc = Fully_Coupled;
        PacketReorder_t  pr = D_SACK;
        int arg1 = -1, arg2 = -1, arg3 = -1, arg4 = -1 ,arg5 = -1;
        int sf = 2; // number of subflows
     int vd=0;
        CommandLine cmd;
        cmd.AddValue("verbose", "Tell application to log if true", verbose);
```

```
        cmd.AddValue("level", "Tell application which log
level to use:\n \t - 0 = ERROR \n \t - 1 = WARN \n \t - 2 =
DEBUG \n \t - 3 = INFO \n \t - 4 = FUNCTION \n \t - 5 = LOGIC
\n \t - 6 = ALL", arg3);

        cmd.AddValue("cc", "Tell application which congestion
control algorithm to use:\n \t - 0 = Uncoupled_TCPs \n \t - 1
= Linked_Increases \n \t - 2 = RTT_Compensator \n \t - 3 =
Fully_Coupled", arg1);

        cmd.AddValue("pr", "Tell application which packet
reordering algorithm to use:\n \t - 0 = NoPR_Algo \n \t - 1 =
Eifel \n \t - 2 = TCP_DOOR \n \t - 3 = D_SACK \n \t - 4 =
F_RTO", arg2);

        cmd.AddValue("sf", "Tell application the number of
subflows to be established between endpoints", arg4);

     cmd.AddValue("vd", "Tell application whether to run
variateDelay or not", arg5);



        cmd.Parse (argc, argv);


        cc = (arg1==-1 ? Uncoupled_TCPs:(CongestionCtrl_t)
arg1);
        pr = (arg2==-1 ? D_SACK:(PacketReorder_t) arg2);
        sf = (arg4 = -1 ? 2: arg4);
     vd = (arg4 = -1 ? 0: arg5);



        LogComponentEnable("FirstMultipathToplogy",
LOG_LEVEL_ALL);
        //LogComponentEnable("TcpSocketFactory",
LOG_LEVEL_ALL);
        //LogComponentEnable("ApplicationContainer",
LOG_LEVEL_INFO);
        //LogComponentEnable("MpTcpL4Protocol",
LOG_LEVEL_ALL);
        //LogComponentEnable("TcpL4Protocol", LOG_LEVEL_INFO);
        //LogComponentEnable("Packet", LOG_LEVEL_ALL);
        //LogComponentEnable("Socket", LOG_LEVEL_ALL);
        if(arg3 == 2)
            LogComponentEnable("MpTcpSocketImpl", LOG_DEBUG);
        else if(arg3 == 6)
            LogComponentEnable("MpTcpSocketImpl",
LOG_LEVEL_ALL);
```

```
        else
            LogComponentEnable("MpTcpSocketImpl", LOG_WARN);
        //LogComponentEnable("TcpSocketImpl", LOG_LEVEL_ALL);
        LogComponentEnable("MpTcpPacketSink", LOG_WARN);
        LogComponentEnable("MpTcpHeader", LOG_WARN);
        //LogComponentEnable("TcpHeader", LOG_LEVEL_ALL);
        //LogComponentEnable("Ipv4L3Protocol", LOG_LEVEL_ALL);
        //LogComponentEnable("MpTcpTypeDefs", LOG_ERROR);
        //LogComponentEnable("RttEstimator", LOG_LEVEL_ALL);

        // Creation of the hosts
        NodeContainer nodes;
        nodes.Create(2);
        client = nodes.Get(0);
        server = nodes.Get(1);



        MpInternetStackHelper stack;
        stack.Install(nodes);

        vector<Ipv4InterfaceContainer> ipv4Ints;
        for(int i=0; i < sf; i++)
        {
            // Creation of the point to point link between
hots
            PointToPointHelper p2plink;
            if(i==0)
                p2plink.SetDeviceAttribute        ("DataRate",
StringValue(path0Capacity));
           else
                p2plink.SetDeviceAttribute        ("DataRate",
StringValue(path1Capacity));


           p2plink.SetChannelAttribute("Delay",
StringValue("100ms"));

            NetDeviceContainer netDevices;
            netDevices = p2plink.Install(nodes);
```

```
            // Attribution of the IP addresses
            std::stringstream netAddr;
            netAddr << "10.1." << (i+1) << ".0";
            string str = netAddr.str();

            Ipv4AddressHelper ipv4addr;
            ipv4addr.SetBase(str.c_str(), "255.255.255.0");
            Ipv4InterfaceContainer          interface          =
ipv4addr.Assign(netDevices);
            ipv4Ints.insert(ipv4Ints.end(), interface);

          //error model
          /*
          Ptr<RateErrorModel>                em                 =
CreateObjectWithAttributes<RateErrorModel> (
               "RanVar", RandomVariableValue (UniformVariable
(0., 1.)),
               "ErrorRate", DoubleValue (0.00001));
          netDevices.Get                        (1)->SetAttribute
("ReceiveErrorModel", PointerValue (em));
          */
        }

        // Enabling PCAP traces to be loged
        PointToPointHelper::EnablePcapAll("mptcp");


        // Configuration of the Client/Server application
        uint32_t servPort = 5000;
    NS_LOG_INFO ("address " << ipv4Ints[0].GetAddress (1));
        ObjectFactory m_sf;
        m_sf.SetTypeId("ns3::MpTcpPacketSink");
        m_sf.Set("Protocol",                          StringValue
("ns3::TcpSocketFactory"));
        m_sf.Set("Local",        AddressValue(InetSocketAddress
(ipv4Ints[0].GetAddress (1), servPort)));
        m_sf.Set("algopr", UintegerValue ((uint32_t) pr));
        Ptr<Application> sapp = m_sf.Create<Application> ();
        server->AddApplication(sapp);
```

```
        ApplicationContainer Apps;
        Apps.Add(sapp);
    //    Apps.Add(capp);


        //ApplicationContainer          serverApps          =
sink.Install(server);
        Apps.Start(Seconds(0.0));
        Apps.Stop(Seconds(simDuration));



        lSocket = new MpTcpSocketImpl (client);

        //lSocket->SetCongestionCtrlAlgo (Linked_Increases);
        //lSocket->SetCongestionCtrlAlgo (RTT_Compensator);
        //lSocket->SetCongestionCtrlAlgo (Uncoupled_TCPs);
     lSocket->SetCongestionCtrlAlgo (cc);

        lSocket->SetDataDistribAlgo (Round_Robin);

        //lSocket->SetPacketReorderAlgo (Eifel);
        lSocket->SetPacketReorderAlgo (pr);

        lSocket->Bind ();


        // Trace changes to the congestion window
        Config::ConnectWithoutContext
("/NodeList/0/$ns3::MpTcpSocketImpl/subflows/0/CongestionWindo
w", MakeCallback (&CwndTracer));


        // ...and schedule the sending "Application"; This is
similar to what an
        // ns3::Application subclass would do internally.
        Simulator::ScheduleNow      (&StartFlow,      lSocket,
ipv4Ints[0].GetAddress (1), servPort);
```

```
        // Finally, set up the simulator to run.  The 1000
second hard limit is a
        // failsafe in case some change above causes the
simulation to never end
        Simulator::Stop (Seconds(simDuration + 2.0));
        Simulator::Run ();
        Simulator::Destroy();
    NS_LOG_LOGIC("mpTopology:: simulation ended");
        return 0;
    }

    //----------------------------------------------------------
---------------------
    //----------------------------------------------------------
---------------------
    //----------------------------------------------------------
---------------------
    //begin implementation of sending "Application"
    void      StartFlow(Ptr<MpTcpSocketImpl>      localSocket,
Ipv4Address servAddress, uint16_t servPort)
    {
      NS_LOG_LOGIC("Starting flow at time " <<  Simulator::Now
().GetSeconds ());
      //localSocket->Connect  (InetSocketAddress  (servAddress,
servPort));//connect
      lSocket->SetMaxSubFlowNumber(5);
      lSocket->SetMinSubFlowNumber(3);
      lSocket->SetSourceAddress(Ipv4Address("10.1.1.1"));
      lSocket->allocateSendingBuffer(sendBufSize);
      lSocket->allocateRecvingBuffer(recvBufSize);
      // the following buffer is uesed by the received to hold
out of sequence data
      lSocket->SetunOrdBufMaxSize(50);


      int  connectionState  =  lSocket->Connect(  servAddress,
servPort);
    //NS_LOG_LOGIC("mpTopology:: connection request sent");
      //    tell    the    tcp    implementation    to    call
WriteUntilBufferFull again
      //  if  we  blocked  and  new  tx  buffer  space  becomes
available
```

```
      if(connectionState == 0)
      {
          lSocket->SetConnectCallback            (MakeCallback
(&connectionSucceeded), MakeCallback (&connectionFailed));
        //        lSocket->SetDataSentCallback    (MakeCallback
(&WriteUntilBufferFull));
          lSocket->SetCloseCallbacks             (MakeCallback
(&HandlePeerClose), MakeCallback(&HandlePeerError));
          lSocket->GetSubflow(0)->StartTracing
("CongestionWindow");
      }else
      {
          //localSocket->NotifyConnectionFailed();
          NS_LOG_LOGIC("mpTopology:: connection failed");
      }

    }

    void connectionSucceeded (Ptr<Socket> localSocket)
    {
        //NS_LOG_FUNCTION_NOARGS();
        NS_LOG_INFO("mpTopology: Connection request succeed");
        Simulator::Schedule          (Seconds          (1.0),
&WriteUntilBufferFull, lSocket, 0);
        Simulator::Schedule    (Seconds    (simDuration+1.0),
&CloseConnection, lSocket);

    }

    void connectionFailed (Ptr<Socket> localSocket)
    {
        //NS_LOG_FUNCTION_NOARGS();
        NS_LOG_INFO("mpTopology: Connection request failure");
        lSocket->Close();
    }

    void HandlePeerClose (Ptr<Socket> localSocket)
    {
        //NS_LOG_FUNCTION_NOARGS();
```

```
        NS_LOG_INFO("mpTopology: Connection closed by peer");
        lSocket->Close();
    }

    void HandlePeerError (Ptr<Socket> localSocket)
    {
        //NS_LOG_FUNCTION_NOARGS();
        NS_LOG_INFO("mpTopology: Connection closed by peer
error");
        lSocket->Close();
    }

    void CloseConnection (Ptr<Socket> localSocket)
    {
        lSocket->Close();
        NS_LOG_LOGIC("mpTopology:: currentTxBytes = " <<
currentTxBytes);
        NS_LOG_LOGIC("mpTopology:: totalTxBytes = " <<
totalTxBytes);
        NS_LOG_LOGIC("mpTopology:: connection to remote host
has been closed");
    }

    /*
    void variateDelay(PointToPointHelper P2Plink)
    {
        //NS_LOG_INFO ("variateDelay -> old delay == " <<
P2Plink.GetDelay());
        NS_LOG_INFO ("variateDelay");
        std::stringstream strDelay;
        int intDelay = rand() % 100;
        strDelay << intDelay << "ms";
        P2Plink.SetChannelAttribute("Delay",
StringValue(strDelay.str()));

        NS_LOG_INFO ("New delay == " << strDelay.str());
    }
    */
```

```
    void variateDelay (Ptr<Node> node)
    {
        //NS_LOG_INFO ("variateDelay");

        Ptr<Ipv4L3Protocol>        ipv4        =        node-
>GetObject<Ipv4L3Protocol> ();
        TimeValue delay;
        for(uint32_t i = 0; i < ipv4->GetNInterfaces(); i++)
        {
            //Ptr<NetDevice> device = m_node->GetDevice(i);
            Ptr<Ipv4Interface>      interface      =      ipv4-
>GetInterface(i);
            Ipv4InterfaceAddress  interfaceAddr  =  interface-
>GetAddress (0);
            // do not consider loopback addresses
            if(interfaceAddr.GetLocal()                    ==
Ipv4Address::GetLoopback())
            {
                // loopback interface has identifier equal to
zero
                continue;
            }

            Ptr<NetDevice> netDev =  interface->GetDevice();
            Ptr<Channel> P2Plink  =  netDev->GetChannel();
            P2Plink->GetAttribute(string("Delay"), delay);
            double oldDelay = delay.Get().GetSeconds();
            //NS_LOG_INFO ("Interface:" << i << " variateDelay
-> old delay == " << oldDelay);
            std::stringstream strDelay;
            double newDelay = (rand() % 100) * 0.01;
            double err = newDelay - oldDelay;
            strDelay << (0.95 * oldDelay + 0.05 * err) << "s";
            P2Plink->SetAttribute(string("Delay"),
StringValue(strDelay.str()));
            P2Plink->GetAttribute(string("Delay"), delay);
            NS_LOG_INFO ("Interface:" << i << " oldDelay == "
<< oldDelay << " neDelay == " << delay.Get().GetSeconds());
        }
    }
```

```
    void  ScheduleTx (int bytesSent, Ptr<Socket> localSocket){

     //debug("ScheduleTx");

     Time tNext (Seconds (bytesSent * 8 / dataRate + 0.00001
));
     //NS_LOG_INFO ("new tnext == " << tNext.GetSeconds());
     Time now =  ns3::Simulator::Now();
     if(     now.GetSeconds()        +   tNext.GetSeconds()   <
simDuration ) {
          Simulator::Schedule ( tNext, &WriteUntilBufferFull,
lSocket, 0);

     }
     else
          CloseConnection (localSocket);

    }



    void   WriteUntilBufferFull   (Ptr<Socket>   localSocket,
unsigned int txSpace)
    {



     //uint32_t txSpace = localSocket->GetTxAvailable ();
      if   (currentTxBytes   <   totalTxBytes   &&   lSocket-
>GetTxAvailable () > 0)
       {

          uint32_t   left          =   std::min(totalTxBytes   -
currentTxBytes, (uint32_t)3000);
          uint32_t   toWrite   =   std::min(writeSize,   lSocket-
>GetTxAvailable ());
          toWrite = std::min( toWrite , left );
```

```
     //NS_LOG_LOGIC("mpTopology::  data  already  sent  ("<<
currentTxBytes <<") data buffered ("<< toWrite << ") to be
sent subsequentlly");
          int     amountBuffered     =     lSocket->FillBuffer
(&data[currentTxBytes], toWrite);
          currentTxBytes += amountBuffered;


        //  ENTER Variate delay or not?
     variateDelay(client);



          lSocket->SendBufferedData();
       ScheduleTx (amountBuffered,localSocket);


      }
      else
       ScheduleTx (1,localSocket);



    }
```

Βιβλιογραφία

[1] Y. Dong, N. Pissinou and J. Wang. Concurrency Handling in TCP." 5th annual conference in Communication Networks and Services Research (CNSR'07),May 2007.

[2] S. Floyd, J. Mahdavi, M. Mathis, M. Podolsky. AnExtension to the Selective Acknowledgement (SACK) Option for TCP," IETF RFC 2883. July 2000.

[3] RFC 6824: TCP Extensions for Multipath Operation with Multiple Addresses http://datatracker.ietf.org/doc/rfc6824/?include_text=1

[4] GNU GPL (General Public License) version 2: http://www.gnu.org/licenses/gpl-2.0.html.

[5] T. Hacker and B. Athey, "The End-to-End Performance Effects of Parallel TCP Sockets on a Lossy Wide-Area Network ," in IEEE Parallel and Distributed Processing Symposium., Proceedings International (IPDPS), Florida, April 2002.

[6] Y. Hasegawa, I. Yamaguchi, T. Hama, H. Shimonishi and T. Murase. \Improved Data Distribution for Multipath TCP communication," IEEE Global Telecommunications Conference (Globecom), December 2005.

[7] M. Handley and C. Raiciu, Multipath TCP implementations, [last accessed Jan. 23, 2011] http://nrg.cs.ucl.ac.uk/mptcp/implementation.html

[8] Ka-Cheong Leung, Victor O.K. LI and D. Yang. "An Overview of Packet Reordering in Transmission Control Protocol (TCP) : Problems, Solutions, and Challenges". IEEE Transactions on Parallel and Distributed Systems, Vol. 18, No. 4, April 2007.

[9] R. Ludwig, M. Meyer, "The Eifel Detection Algorithm for TCP," IETF RFC 3522, April 2003.

[10] R. Ludwig, R. H.Katz. "The Eifel Algorithm: Making TCP Robust Against Spurious Retransmissions," ACM SIGCOMM Computer Comm. Review, vol. 30, no. 1, pp. 30-36, Jan 2000.

[11] MPTCP Code Project: http://code.google.com/p/mptcp-ns3/

[12] Network Simulator ns-3: http://www.nsnam.org/

[13] L. Ong, J. Yoakum, "An Introduction to the Stream Control Transmission Protocol (SCTP," IETF RFC 3286, May 2002.

[14] C. Raiciu, D. Wischik and M. Handley, "Practical Congestion Control for Multipath Transport Protocols," UCL Technical Report, 2009.

[15] C. Raiciu, M. Handley and D. Wischik, "Coupled Multipath-Aware Congestion Control", IETF draft draft-raiciu-mptcp-congestion-01, March 2010.

[16] K. Rojviboonchai and H. Aida. "An evaluation of multi-path Transmission Control Protocol (M/TCP) with robust acknowledgement schemes," Internet Conference (IC'02), Tokyo (Japan), October 2002.

[17] D. Sarkar. "A Concurrent Multipath TCP and Its Markov Model," IEEE International Conference on Communications (ICC), June 2006.

[18] P. Sarolahti, M. Kojo, K. Yamamoto, M. Hata. "Forward RTO-Recovery (F-RTO): An Algorithm for Detecting Spurious Retransmission Timeouts with TCP," IETF RFC 5682. September 2009.

[19] D. Wischik, M. Handley and M. Bagnulo Braun, "The Resource Pooling Principle", ACM SIGCOMM Computer Communication Review, Vol 38.5, pp 47-52, October 2008.

[20] M. Zhang, J. Lai, A. Krishnamurthy, L. Peterson and R. Wang. "A transport layer approach for improving end-to-end performance and robustness using redundant paths," Annual conference on USENIX (ATEC'04), Berkeley (CA, USA), June 2004.

[21] iOS: Multipath TCP Support in iOS 7 http://support.apple.com/kb/HT5977

[22] Bachir Chihani, Denis Collangem " A Multipath TCP model for ns-3 simulator", Workshop on ns-3 held in conjunction with SIMUTools 2011, Barcelona : Spain (2011).

[23] MPTCP Code Project για το NS-3: http://code.google.com/p/mptcp-ns3/.

[24] L. Magalhaes and R. Kravets. Transport Level Mechanisms for Bandwidth Aggregation on Mobile Hosts. In International Conference on Network Protocols (ICNP), 9th, σελίδες 165–171. IEEE Computer Society, Νοέμβριος 2001.

[25] Y. Hsieh and R. Sivakumar. pTCP: An End-to-End Transport Layer Protocol for Striped Connections. In *International Conference on Network Protocols (ICNP)*, 10th, σελίδες 24–33. IEEE, Νοέμβριος 2002.

[26] RFC Stream Control Transmission Protocol, http://www.ietf.org/rfc/rfc2960.txt

[27] J. Iyengar, P.D. Amer, and R.R. Stewart. Concurrent multipath transfer using SCTP multihoming over independent end-to-end paths. *IEEE/ACM Transactions on Networking (TON)*, 14(5):951–964, Οκτώβριος 2006.

[28] A. Abd El Al, T. N. Saadawi, and M. J. Lee. LS-SCTP: a bandwidth aggregation technique for stream control transmission protocol. Comput. Commun., 27(10):1012-1024, Ιούνιος 2004

[29] Ka-Cheong Leung, Victor O.K. LI and D. Yang. An Overview of Packet Reordering in Transmission Control Protocol (TCP) : Problems, Solutions, and Challenges". IEEE Transactions on Parallel and Distributed Systems, Vol. 18, No. 4, April 2007.

[30] R. Ludwig, M. Meyer, "The Eifel Detection Algorithm for TCP," IETF RFC 3522, April 2003.

[31] R. Ludwig, R. H.Katz. "The Eifel Algorithm: Making TCP Robust Against Spurious Retransmissions," ACM SIGCOMM Computer Comm. Review, vol. 30, no. 1, pp. 30-36, Jan 2000.

[32] S. Floyd, J. Mahdavi, M. Mathis, M. Podolsky. "An Extension to the Selective Acknowledgement (SACK) Option for TCP," IETF RFC 2883. July 2000.

[33] P. Sarolahti, M. Kojo, K. Yamamoto, M. Hata. "Forward RTO-Recovery (F-RTO): An Algorithm for Detecting Spurious Retransmission Timeouts with TCP," IETF RFC 5682. September 2009.

[34] http://www.wireshark.org/