ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Τμήμα Ψηφιακών Συστημάτων

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

Κατεύθυνση

ΨΗΦΙΑΚΕΣ ΕΠΙΚΟΙΝΩΝΙΕΣ ΚΑΙ ΔΙΚΤΥΑ

ΠΟΛΥΔΙΑΔΡΟΜΙΚΟ TCP

Μεταπτυχιακός φοιτητής: Σιόλος Δ. Νικόλαος

Επιβλέπων: Αναπληρωτής Καθηγητής Ρούσκας Άγγελος

Διπλωματική Εργασία υποβληθείσα στο Τμήμα Ψηφιακών Συστημάτων του Πανεπιστήμιου Πειραιώς ως μέρος των απαιτήσεων για την απόκτηση Μεταπτυχιακού Διπλώματος Ειδίκευσης στις Ψηφιακές Επικοινωνίες και Δίκτυα.

Πειραιάς, Δεκέμβριος 2015

UNIVERSITY OF PIRAEUS

DEPARTMENT OF DIGITAL SYSTEMS

MASTER PROGRAM IN

DIGITAL COMMUNICATIONS

AND NETWORKS

MULTIPATH TCP

Author's Name: Nick Siolos

Supervisor: Associate Professor Angelos Rouskas

Master Thesis submitted to the Department Digital Systems of the University of Piraeus in partial fulfillment of the requirements for the degree of Master Program in Digital Systems And Networks.

Piraeus, Greece, December 2015

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Πειραιάς 2015

Η Ιθάκη σ’ έδωσε τ’ ωραίο ταξίδι.

Χωρίς αυτήν δεν θα 'βγαινες στον δρόμο.

Άλλα δεν έχει να σε δώσει πια.

Κωνσταντίνος Καβάφης

Το άτομο το οποίο εκπονεί την Διπλωματική Εργασία φέρει ολόκληρη την ευθύνη προσδιορισμού της δίκαιης χρήσης του υλικού, η οποία ορίζεται στην βάση των εξής παραγόντων: του σκοπού και χαρακτήρα της χρήσης (εμπορικός, μη κερδοσκοπικός ή εκπαιδευτικός), της φύσης του υλικού, που χρησιμοποιεί (τμήμα του κειμένου, πίνακες, σχήματα, εικόνες ή χάρτες), του ποσοστού και της σημαντικότητας του τμήματος, που χρησιμοποιεί σε σχέση με το όλο κείμενο υπό copyright, και των πιθανών συνεπειών της χρήσης αυτής στην αγορά ή στη γενικότερη αξία του υπό copyright κειμένου

Περίληψη

Από την αρχή της δημιουργίας των δικτύων εμφανίστηκε η ανάγκη για μεγαλύτερη αξιοπιστία και μεγαλύτερη ταχύτητα. Έτσι αναπτύχτηκαν διάφορες τεχνικές για την εξασφάλιση της μεταφοράς των δεδομένων και ταυτόχρονα επενδύθηκαν περισσότερα χρήματα ούτως ώστε τα συστήματα να έχουν την δυνατότητα εναλλακτικών διαδρομών σε περίπτωση βλαβών. Το TCP είναι ένα πρωτόκολλο επικοινωνίας που μας δίνει ακόμα και τώρα τη δυνατότητα να ανταλλάσουμε πακέτα μεταξύ χρηστών με αποτελεσματικότητα. Η αύξηση όμως των χρηστών δημιούργησε το πρόβλημα της διευθυνσιοδότησης αφού οι IP σε IPv4 εξαντλήθηκαν και η απαίτηση για μεγαλύτερες ταχύτητες και αδιάλειπτο internet αποτέλεσαν την γέννηση τεχνικών Multihoming. Επιπλέον, η δυνατότητα χρήσης Wi-Fi αλλά και 3G/4G ενέτεινε το πρόβλημα. Αν και δόθηκε η λύση στην διευθυνσιοδότηση με το IPv6 το Multihoming παρέμενε ένα πολύπλοκο και πολυσύνθετο πρόβλημα λαμβάνοντας υπ όψιν πως η κίνηση TCP ακολουθεί μοναδική διαδρομή (single path). Όλα αυτά τα χρόνια παρουσιάστηκαν αρκετές λύσεις multihoming, όπως το Link aggregation, το Shim6 και μια πιο επιτυχημένη το SCTP, οι οποίες όμως είτε χρησιμοποιούσαν τεχνικές που κάλυπταν προβλήματα και δημιουργούσαν άλλα είτε δεν μπορούσαν να είναι εφαρμόσιμα καθώς δεν υποστηρίζονταν από τις υπάρχουσες εφαρμογές. Το Multipath TCP είναι μια εφαρμογή η οποία εκμεταλλεύεται την υπάρχουσα δομή του internet και κάνει πράξη τις πολυδιαδρομικές συνδέσεις επεκτείνοντας τις επιλογές του TCP, ξεπερνώντας τις μέχρι τώρα γνωστές λύσεις multihoming και έχοντας πάντα κατά νου τα προβλήματα που δημιουργούνται με τα middleboxes.

Στην παρούσα διπλωματική εργασία θα δούμε τις βασικές αρχές και δομές του MPTCP καθώς και την εφαρμογή του σε πραγματικό και εικονικό περιβάλλον.

Συγκεκριμένα,

Στο **Κεφάλαιο 1** περιγράφεται η δομή και η ορολογία του Multipath TCP.

Το **Κεφάλαιο 2** είναι ένα από τα σημαντικότερα, καθώς αναλύεται ο τρόπος με τον οποίο γίνεται η εγκατάσταση του πρώτου αλλά και των επιμέρους υποροών (subflows), συνεπώς αποτελεί το πρώτο σκαλοπάτι για μια συγχρονισμένη και ασφαλή μετάδοση δεδομένων.

Στο **Κεφάλαιο 3** εμβαθύνουμε στον τρόπο με τον οποίο γίνεται η επιλογή υποροής για την μεταφορά δεδομένων, τη διασφάλιση της αναμετάδοσης των χαμένων πακέτων αλλά και τη διασφάλιση δικαιοσύνης ανάμεσα στις υπόλοιπες ροές μέσα στο ίδιο bottleneck.

Στο **Κεφάλαιο 4** γίνεται εγκατάσταση ρύθμιση και εκτέλεση του MPTCP σε πραγματικό περιβάλλον με την χρήση λειτουργικού Linux, και

Στο **Κεφάλαιο 5** γίνεται εγκατάσταση ρύθμιση και εκτέλεση του MPTCP σε εικονικό περιβάλλον με την χρήση λειτουργικού VMware και Linux.

Στο **Κεφάλαιο 6** γίνονται μετρήσεις με διαφορετικούς αλγορίθμους congestion control.

Στο **Κεφάλαιο 7** παρουσιάζονται τα συνολικά συμπεράσματα της εργασίας.

Λέξεις Κλειδιά

Multipath TCP, MPTCP, Multipath Transport, Path Manager, Congestion Control, Scheduler, Subflows, Meta-Socket, Sub-Socket.

Abstract

The need for greater reliability and greater speed was evident even since the appearance of early networks. Various techniques for ensuring data transfers have been developed, and more money was invested for systems to have the possibility of alternative paths in case of failures. TCP is a communication protocol that provides for the exchange of packets between users efficiently. However, the increase of users has created the problem of IPv4 addresses shortage and the demand for higher speeds and uninterrupted internet access gave birth to Multihoming techniques. In addition, Wi-Fi and use 3G / 4G mobile internet penetration intensified the problem. Although IPv6 solved the addressing problem, multihoming remained a complex issue taking into account that TCP is single path oriented. Over the years several multihoming solutions have been presented, like Link aggregation, Shim6 and SCTP, but these techniques are not always applicable because they are not supported by existing applications. Multipath TCP (MPTCP) is an application which is based on the existing internet structure and creates multipath connections extending TCP options, surpassing the known Multihoming solutions and taking into account the problems created by middleboxes.

In this thesis, we will see the basic principles and structures of MPTCP and it implementation in a real and virtual environment.

Specifically,

Chapter 1 describes the structure and terminology of the Multipath TCP.

Chapter 2 describes how the installation of the first and the individual subflows is done, which is the first step for a synchronized and secure data transmission.

Chapter 3 describes how the selection of a subflow for data transfer is performed, ensuring the retransmission of lost packets and justice among the remaining flows of the same bottleneck.

In Chapter 4, the installation, adjustment and implementation of MPTCP in a real environment using Linux operating system is described.

In Chapter 5, the installation, adjustment and implementation of MPTCP in a virtual environment using VMware and Linux operating systems is described.

In Chapter 6, we performed measurements using different congestion control algorithms.

In Chapter 7 we present the overall conclusions reached.

Keywords

Multipath TCP, MPTCP, Multipath Transport, Path Manager, Congestion Control, Scheduler, Subflows, Meta-Socket, Sub-Socket.

Ακρωνύμια

CPU: Central Processing Unit

HMAC: Hash-based Message Authentication Code

IP: Internet Protocol

IPv4: Internet Protocol version 4

IPv6: Internet Protocol version 6

MSS: Maximum Segment Size

MPTCP: MultiPath Transmission Control Protocol

NAT: Network Address Translation

NIC: Network Interface Card

RTT: Round Trip Time

SSTHRES: Slow start threshold

SRTT: Smoothed Round Trip Time

TCP: Transmission Control Protocol

TSO: TCP Segmentation Offload

Ευχαριστίες

Θερμές ευχαριστίες στους γονείς μου και τον αδελφό μου για την στήριξη τους, τον καθηγητή μου κύριο Άγγελο Ρούσκα για τις χρήσιμες συμβουλές του καθώς και σε όλους τους φίλους για τις χρήσιμες πληροφορίες τους.

Περιεχόμενα

Κατάλος Σχημάτων

Κατάλογος Εικόνων

1. Κεφάλαιο - MPTCP

1.1. Η θεωρία του MPTCP

Το Multipath TCP πετυχαίνει την συγκέντρωση πόρων ούτως ώστε να αυξήσει την ανοχή στην αποτυχία των συνδέσεων, κατανέμοντας την κίνηση σε όλες τις πιθανές εναλλακτικές διαδρομές.

Τα επιτεύγματα του Multipath TCP θα μπορούσαν να συνοψιστούν στα εξής:

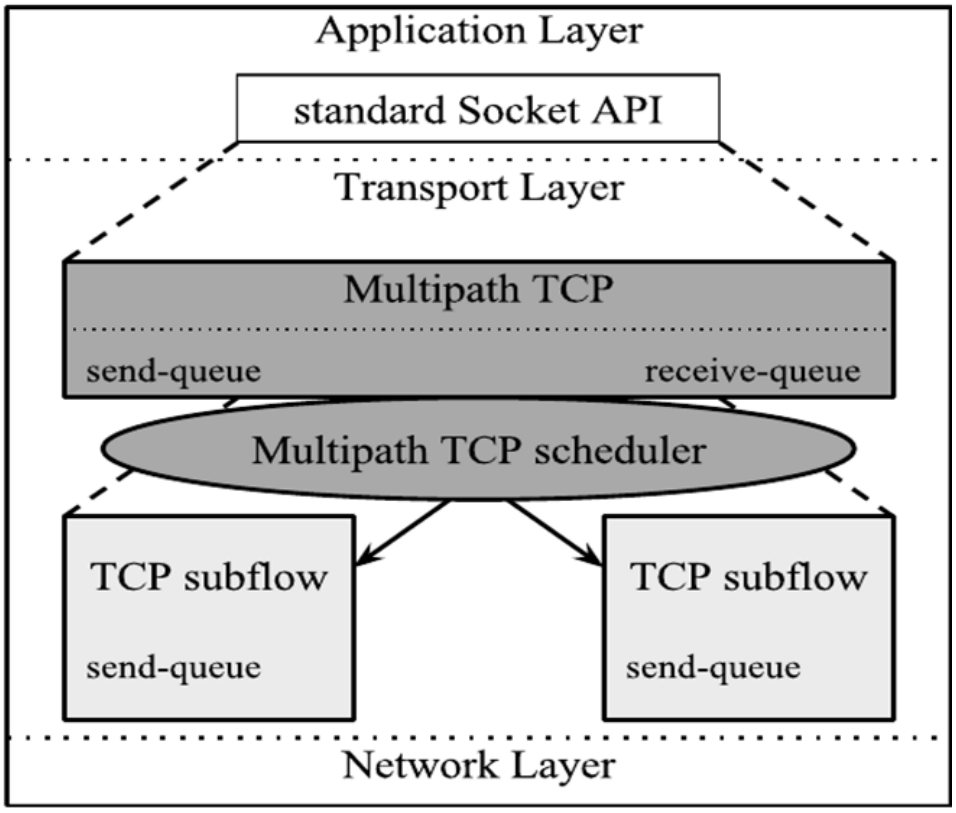
- Δυνατότητα χρήσης πολλαπλών διαδρόμων δικτύου με μια μόνο σύνδεση
- Η χρήση των πολλαπλών διαδρομών δικτύου θα πρέπει να είναι τουλάχιστον στο ίδιο επίπεδο με το TCP χωρίς όμως αυτό να γίνεται σε βάρος του TCP
- Είναι άμεσα συμβατό με τις υπάρχουσες εφαρμογές
- Η ενεργοποίηση του δεν εμποδίζει συνδέσεις που ήδη ενεργούν με μια TCP σύνδεση.

Για να μπορεί το Multipath TCP να είναι χρηστικό στις υπάρχουσες εφαρμογές όπως το TCP θα πρέπει να παρέχει ένα αξιόπιστο και δομημένο μηχανισμό για την μεταφορά των δεδομένων.

Έτσι το MPTCP πρωτόκολλο περιέχει την φάση εγκατάστασης σύνδεσης μεταξύ τερματικών, έναν μηχανισμό αναγνώρισης (acknowledgement) που δίνει την δυνατότητα μιας αξιόπιστης σύνδεσης, βάζοντας στη σειρά τα δεδομένα που καταφθάνουν στον παραλήπτη ή ζητά αυτά που δεν πήγαν, έναν μηχανισμό διαχείρισης ροής (flow control) κατά τον οποίο ο λήπτης προφυλάσσεται από υπερβολική λήψη δεδομένων από τον αποστολέα με αποτέλεσμα να εξαντλούνται οι buffers και τέλος για να μπορέσει να σηματοδοτηθεί το τέλος ενός byte stream υπάρχει ο κατάλληλος τερματισμός.

Ένας απλός τρόπος για να εφαρμοστεί το MPTCP θα ήταν να παίρναμε τα segments από την έξοδο ενός απλού TCP stack και να τα μοιράζαμε σε όλα τα διαθέσιμα μονοπάτια. Για να μπορούσε αυτό να είναι λειτουργικό ο αποστολέας θα χρειαζόταν να γνωρίζει ποιά μονοπάτια λειτουργούν και ποια όχι (RTT check για τον έλεγχο απωλειών). Έτσι ο αποστολέας θα έπρεπε να θυμάται ποια segment στάλθηκαν, σε ποιο path και αν αυτά έφθασαν στον προορισμό τους. Με αυτές τις πληροφορίες ο αποστολέας θα είχε την δυνατότητα να κάνει retransmissions σε περίπτωση απώλειας ανεξάρτητα σε κάθε path διατηρώντας τον έλεγχο της συμφόρησης (congestion control).

Όμως με αυτόν τον τρόπο για κάθε path θα δημιουργείτο μια ασυνέχεια στο TCP bytestream με αποτέλεσμα να δημιουργείται σύγχυση στα middleboxes και αυτά να ρίχνουν τις συνδέσεις! (Τα middleboxes είναι συσκευές δικτύου οι οποίες διαχειρίζονται την κίνηση για διαφορετικούς λόγους εκτός από packet forwarding. Συνήθως είναι firewalls και network address translators - NAT) [RPBF+]



Σχήμα 1*: Αρχιτεκτονική υλοποίησης πρωτοκόλλου MPTCP*

Το Multipath TCP δημιουργεί ένα TCP subflow για κάθε path, ούτως ώστε ο scheduler να κάνει την διανομή των δεδομένων σε αυτά. Στο application layer υπάρχει μόνο ένα standard stream socket interface. Από κάτω το MPTCP διαπραγματεύεται με καινούργιες επιλογές του TCP που υπάρχουν στα SYN πακέτα των TCP συνδέσεων. Για να είναι εφικτή η μετάδοση μεταξύ διαφορετικών διαδρομών, δημιουργείται ένα TCP subflow για κάθε μια από αυτές τις διαδρομές. Αυτές μοιάζουν με απλές TCP συνδέσεις που συμπεριλαμβάνουν το 3-way handshake για την εγκατάσταση, κατάλληλης ακολουθίας αριθμών (sequence number) για τα retransmissions και ένα 4 -way handshake για τον τερματισμό. Αυτά τα subflows συνδέονται μεταξύ τους για να σχηματίσουν μια MPTCP σύνδεση και να μεταφέρουν δεδομένα μεταξύ τελικών χρηστών. Ο Multipath TCP Scheduler έχει ως αρμοδιότητα να διανέμει δεδομένα μεταξύ των διαφορετικών subflows αφήνοντας να κάνει την επιλογή των πηγών για κάθε subflow path. Κάθε subflow χρησιμοποιεί το δικό του sequence number space για να ελέγχει τις απώλειες και να ζητά retransmission. Το Multipath TCP προσθέτει connection-level sequence numbers που του επιτρέπουν reordering κατά τη λήψη. Τέλος χρησιμοποιούνται connection-level acknowledgements για να υπάρχει κατάλληλο flow control.

Θα μπορούσαμε να πούμε λοιπόν πως το MPTCP χαρακτηρίζεται κυρίως από δυο κομμάτια. Το control plane, το οποίο είναι υπεύθυνο να δημιουργεί και να καταστρέφει subflows και να σηματοδοτεί connection-level πληροφορίες ελέγχου και το data plane, το οποίο μεταδίδει τα δεδομένα μεταξύ των τελικών χρηστών.

Ας δούμε όμως πρώτα την γενική αρχιτεκτονική του Multipath TCP.

1.2. Multipath TCP

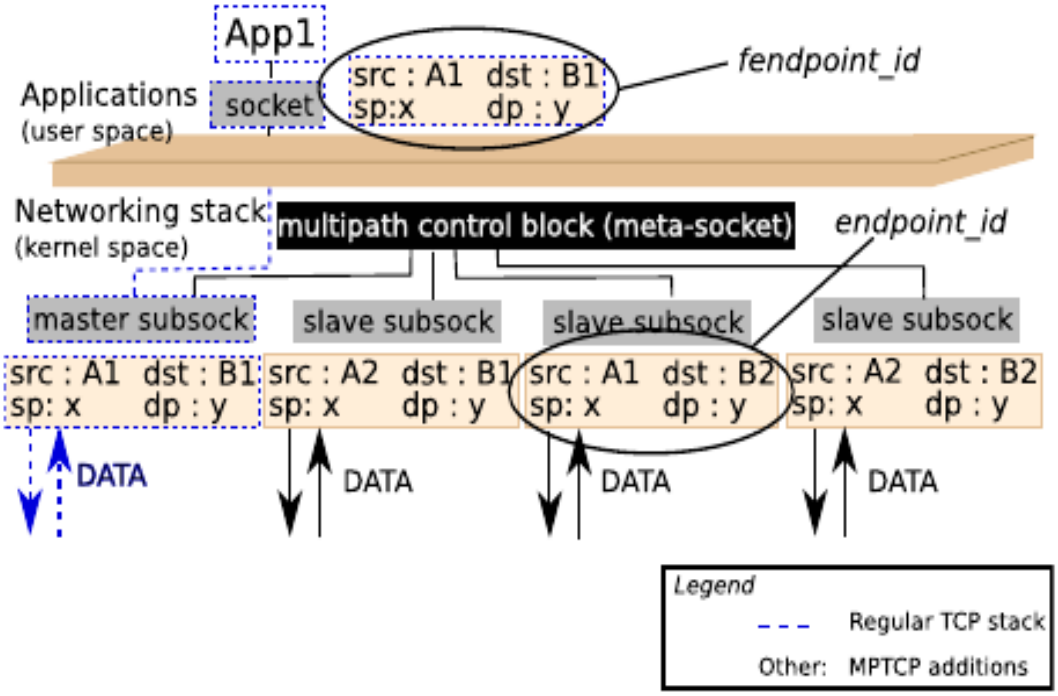
Όπως προαναφέραμε το Multipath TCP είναι μια σημαντική επέκταση του TCP που επιτρέπει την ταυτόχρονη χρήση πολλαπλών διαδρομών [RFC 6824]. Σε αυτή την διπλωματική θα δούμε την αρχιτεκτονική του MPTCP έτσι όπως αναπτύχθηκε σε λειτουργικό περιβάλλον Linux. Ο κώδικας υπάρχει ως open-source στην σελίδα http://www.multipath-tcp.org/ .

1.3. Ορολογία

Για να γίνει κατανοητή η εφαρμογή του Linux MPTCP θα ήταν χρήσιμο να ερμηνεύσουμε κάποιους όρους οι οποίοι εμφανίζονται στο *Σχήμα 2* .

- **Meta-socket:** Μια δομή socket που χρησιμοποιείται για την αναδιάταξη των εισερχομένων δεδομένων στο επίπεδο σύνδεσης (connection level) και σχεδιάζει τα εξερχόμενα δεδομένα στις υποροές (subflows).
- **Master subsocket:** Η socket δομή που είναι εμφανής στο application. Όταν μια απλή TCP σύνδεση είναι ενεργή τότε είναι το μόνο ενεργό socket, ενώ εάν χρησιμοποιείται το MPTCP τότε αυτό το socket αντιστοιχεί στο πρώτο subflow.
- **Slave subsocket:** Κάθε socket που δημιουργείται από τον kernel για να παρέχει πρόσθετο subflow. Αυτά τα socket δεν είναι εμφανή από το application.
- **Endpoint ID:** Endpoint Identifier. Ομάδα στοιχείων που προσδιορίζουν ένα συγκεκριμένο subflow και ως εκ τούτου ένα συγκεκριμένο subsocket: (saddr, sport, daddr, dport).
- **Fendpint ID:** First Endpoint identifier. Το endpoint id του Master sub socket.
- **Connection ID η Token:** Ένας τοπικά μοναδικός αριθμός που επιτρέπει την εύρεση μιας σύνδεσης κατά τη δημιουργία νέων subflows.
- **local_addr_table**: Πίνακας τοπικών διευθύνσεων. Αποθηκεύει το σύνολο των τοπικών διευθύνσεων, ανά σύνδεση, καθώς μια σύνδεση MPTCP μπορεί να χρησιμοποιηθεί για τα subflows.

- **remote_addr_table:** Πίνακας με απομακρυσμένες διευθύνσεις. Αποθηκεύει, ανά σύνδεση, το σύνολο των απομακρυσμένων διευθύνσεων που μια σύνδεση MPTCP έχει μάθει από τους συμμετέχοντες, είτε μέσω της επιλογής ADD_ADDRESS του MPTCP, είτε μέσω SYNS που αποστέλλονται από τους συμμετέχοντες χρησιμοποιώντας νέες διευθύνσεις.



Σχήμα 2: *Επισκόπηση της multipath αρχιτεκτονικής*

1.4. Αρχιτεκτονική Multipath Transport

Στο [RFC 6182] περιγράφεται η λειτουργική δομή του MPTCP σε τέσσερις οντότητες, το **Path Management**, το **Packet Scheduling**, το **Subflow Interface** και το **Congestion Control**.

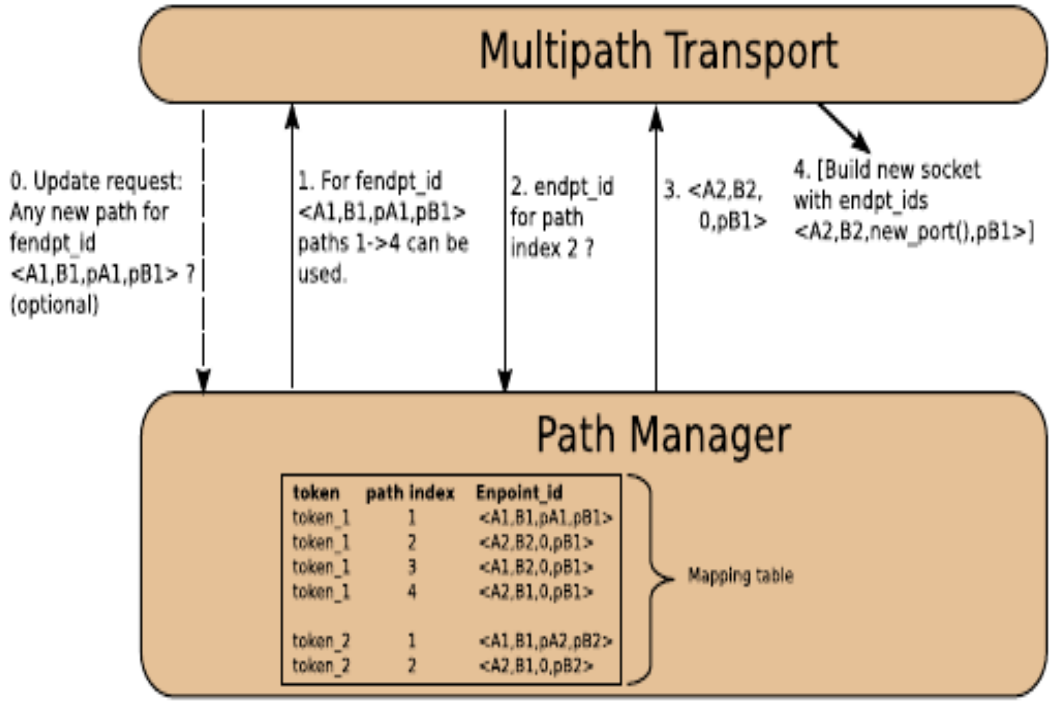
Αυτές οι οντότητες θα μπορούσαν να ομαδοποιηθούν ανάλογα με το επίπεδο το οποίο εργάζονται:

- **Transport layer:** περιλαμβάνει το Packet Scheduling, το Subflow Interface και το Congestion Control και ομαδοποιούνται κάτω από τον όρο “Multipath Transport (MT)”.

- **Transport layer and below:** Path management. Το Path management μπορεί να γίνει στο transport layer, στην περίπτωση που είναι ενσωματωμένος ο Path Manager (PM). Ο PM ανακαλύπτει διαδρομές μέσα από την ανταλλαγή των TCP options όπως το ADD_ADDR ή κατά τη λήψη ενός SYN για ένα νέο ζευγάρι διευθύνσεων, και ορίζει τη διαδρομή ως ένα endpoint id (saddr, sport, daddr, dport). Αλλά, γενικότερα, ένα PM θα μπορούσε να είναι οποιαδήποτε μονάδα είναι σε θέση να εκθέσει πολλαπλές διαδρομές σε ένα MPTCP, και να βρίσκεται είτε στον kernel είτε στο user space, και να ενεργεί σε κάθε OSI layer.

1.5. Αρχιτεκτονική του MPTCP

Στην *Σχήμα 3* μπορούμε να δούμε τον PM και τον τρόπο με τον οποίο αλληλεπιδρά με το MPTCP, που ασχέτως αν είναι built-in μπορεί να φανεί οργανωμένος ως Path Manager και Multipath Transport Layer.



***Σχήμα 3**: Λειτουργικός διαχωρισμός του MPTCP στο transport layer*

Ο **Path Manager** ανακοινώνει στο Multipath Transport ποιές διαδρομές μπορούν να χρησιμοποιηθούν μέσω δεικτών διαδρομής MPTCP σύνδεσης, που προσδιορίζονται από το fendpoint ID (first end point ID). Το fendpoint ID ειναι μια τετράδα στοιχείων (saddr, sport, daddr, dport) που θεωρείται από το application ότι προσδιορίζει μοναδικά μια MPTCP σύνδεση.

Ο **Path Manager** διατηρεί την χαρτογράφηση μεταξύ του path_index και του endpoint ID.

Το **endpoint ID** είναι επίσης μια τετράδα στοιχείων (saddr, sport, daddr, dport) που χρησιμοποιείτε για το αντίστοιχο path index.

Στο παραπάνω σχήμα (*Σχήμα 3)* μπορούμε να δούμε ένα παράδειγμα για το πως αλληλεπιδρούν μεταξύ τους το MT με τον PM κατά την εκκίνηση μιας ανταλλαγής. Όταν το MT ξεκινά μια νέα σύνδεση ζητά από τον PM να κάνει μια ανανέωση της βάσης του για πιθανές εναλλακτικές διαδρομές (βήμα 0). Ο PM μπορεί αυτόματα να ανανεώσει το MT σε οποιαδήποτε χρονικό διάστημα (βήμα 1). Στο παράδειγμά μας τέσσερις διαδρομές μπορούν να επιλεγούν και οι τρεις από αυτές είναι νέες. Το MT βασίζεται στην ανανέωση της βάσης για να μπορέσει να αποφασίσει πότε να δημιουργήσει ένα νέο subflow. Εν προκειμένω το MT αποφασίζει να δημιουργήσει ένα subflow και στέλνει μια αίτηση για endpoint ID στον PM (βήμα 2). Στο 3^{o} βήμα δίνεται η απάντηση: <A2, B2, 0, pB2>. Η source port δεν καθορίζεται για να δοθεί η δυνατότητα στο MT να διασφαλίσει την μοναδικότητα στο νέο endpoint ID, χάρη στο new_port() (βήμα 4).

Οι παρακάτω επιλογές διαχειρίζονται οπό το Multipath Transport:

- MULTIPATH CAPABLE (MP CAPABLE): Λέει στους μετέχοντες πως υποστηρίζεται το MPTCP και ανακοινώνει το local token.
- MP JOIN/MP AUTH: Ξεκινά ένα καινούργιο subflow.
- DATA SEQUENCE NUMBER (DSN MAP): Αναγνωρίζει τη θέση των new byte στο meta-flow.

- DATA ACK: Αναγνωρίζει τα δεδομένα στο connection level (subflow level Acknowledgements are contained in the normal TCP header).
- DATA FIN (DFIN): Τερματίζει τη σύνδεση.
- MP PRIO: Ανανέωση του backup status ενός subflow. Δεν υποστηρίζεται ακόμα.
- MP FAIL: Checksum failed at connection-level. Δεν υποστηρίζεται ακόμα.

Ο Path Manager εφαρμόζει μια συγκεκριμένη τεχνολογία για να δώσει στον MT τη δυνατότητα να χρησιμοποιεί διάφορες διαδρομές. Ο built-in MPTCP Path Manager χρησιμοποιεί πολλαπλές IPv4/v6 διευθύνσεις για να μπορέσει να προωθήσει τα πακέτα στο Internet. Όταν ο MT ξεκινά μια νέα σύνδεση, διαλέγει ένα token για να μπορέσει να αναγνωρίσει την σύνδεση. Αυτό είναι απαραίτητο για να δώσει την δυνατότητα σε νέα subflow-establishment SYNS να επισυναφθούν στη σωστή σύνδεση.

Στο παράδειγμά μας (**Σχήμα 4**) έχουμε ενεργές δυο MPTCP συνδέσεις. Η μία έχει αναγνωριστικό token_1 και η άλλη token_2. Από τη στιγμή που το endpoint ID μπορεί να αλλάξει από το ένα subflow στο άλλο η επισύναψη των νέων εισερχόμενων subflows στη σωστή σύνδεση επιτυγχάνεται χάρις στο τοπικά σημαντικό token.

Οι παρακάτω επιλογές περιλαμβάνονται στον built-in Path Manager:

- **Add Address (ADD ADDR):** Ανακοινώνει μια νέα διεύθυνση που μας ανήκει.
- **Remove Address (REMOVE ADDR):** Αποσύρει μια παλιά διεύθυνση.

Οι επιλογές αυτές βασίζονται στην δήλωση IP διευθύνσεων και μεταφέρουν πληροφορίες ελέγχου στις επιλογές TCP.

1.6. Δομή του Multipath Transport

Το Multipath Transport διαχειρίζεται τριών ειδών sockets (*Σχήμα 2, σελ.5*):

- **Master subsocket:** Το πρώτο socket που είναι σε χρήση όταν μια σύνδεση ξεκινά (TCP η MPTCP), όπως επίσης και η μοναδική που μπορεί να χρησιμοποιηθεί σε περίπτωση που χρειαστεί να επιστρέψουμε σε κανονική TCP σύνδεση. Αυτό το socket ενεργοποιείται από το application μέσω του **system call socket**. Αμέσως μετά την δημιουργία ενός καινούργιου master subsocket, το MPTCP capability ενεργοποιείται από τη δημιουργία του meta-socket.

- **Meta-socket:** Διατηρεί το multipath control block, και λειτουργεί σαν το connection level socket. Σαν data source, διατηρεί το βασικό buffer των απεσταλμένων, ενώ σαν data sink, διατηρεί την σειρά των εισερχομένων στο connection-level και των out-of-order queue ούτως ώστε να υπάρξει reordering. Ακόμη διατηρεί τις κατάλληλες σειρές για τα απεσταλμένα / ληφθέντα δεδομένα.

- **Slave subsocket:** Κάθε άλλο subflow που δημιουργείται από το MPTCP, σε αντίθεση με το πρώτο. Δημιουργείται από τον kernel και μαζί με το master subsocket φτιάχνουν ένα pool διαθέσιμων subflows όπου ο MPTCP Packet Scheduler (καλείται από το meta-socket) χρησιμοποιεί για να στείλει πακέτα.

1.7. Δομή του Path Manager

Ο Path Manager διατηρεί τον mapping table και ανανεώνει το Multipath Transport όταν ο mapping table δέχεται αλλαγές.

event	**action**
master_sk bound: ενεργοποιείται είτε από το bind () / connect () system call, η όταν δημιουργείται ένα νέο socket στην πλευρά του server	Ανακαλύπτει τις νέες τοπικές διευθύνσεις και αυτές στη συνέχεια διατηρούνται στο local_addr_table
ADDR or SYN + MP_JOIN received on new address	Ανανεώνει αντίστοιχα τον remote_addr_table
local/remote addr table updated	Ανανεώνει το mapping_table προσθέτοντας κάθε νέο συνδυασμό διευθύνσεων ή αφαιρώντας αυτές που εξαφανίστηκαν. Σε κάθε ζευγάρι διευθύνσεων δίνετε ένα path index
Mapping table updated (*Σχήμα 3*, msg 1)	Στέλνει μια ειδοποίηση στο Multipath Transport (μήνυμα 1 στο *Σχήμα 3*)
Endpoint ID request received fromMT (*Σχήμα 3*, msg 2)	Ανακτά τα endpoint IDs από το αντίστοιχο path index από τον mapping table και τα επιστρέφει στο MT (*Σχήμα 3*, msg 3)

2. Κεφάλαιο - Control Plane / Handshake

Οι πληροφορίες ελέγχου μεταξύ δυο MPTCP χρηστών μεταφέρονται εντός του TCP option space. Το MPTCP χρησιμοποιεί μια επιλογή TCP και διαφοροποιείται την πληροφορία ελέγχου με υποεπιλογές. Στη συνέχεια του κεφαλαίου θα δούμε πώς το MPTCP δημιουργεί ή καταστρέφει subflows.

2.1. Αρχικό handshake

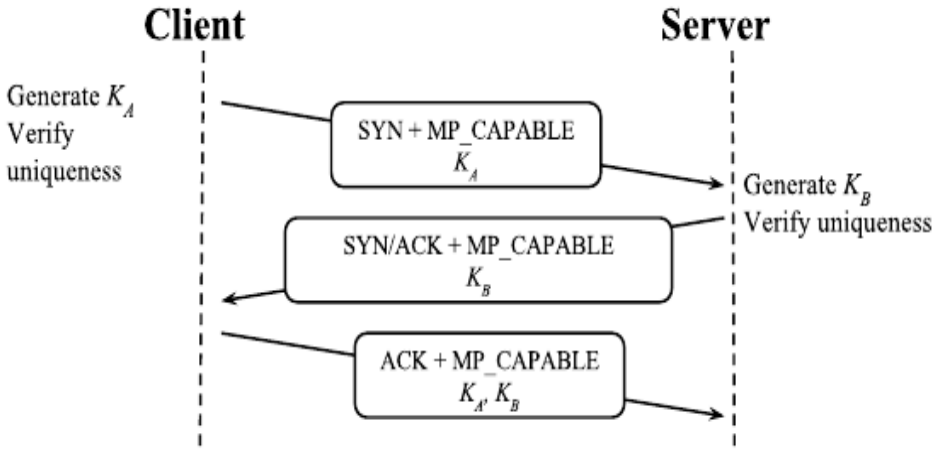
Στο TCP το three-way handshake χρησιμοποιείται για να συγχρονίσει τις καταστάσεις μεταξύ του client και του server. Ανταλλάσσονται sequence numbers και αναγνωρίζονται ενώ TCP options μεταφέρονται μέσα στα SYN και τα SYN/ACK πακέτα για να υπάρξει μια λειτουργική συμφωνία όπως το maximum segment size (MSS) ή την υποστήριξη χρονοσφραγίδων TCP [RFC 1323]. Στο MPTCP ο χρήστης μπορεί να αναγνωρίσει εάν η άλλη πλευρά υποστηρίζει το Multipath και ανταλλάσσονται μεταβλητές τριών επιπέδων για τη σύνδεση :

1. Το TCP αναγνωρίζει μια σύνδεση χάρις σε μιας πεντάδα στοιχείων των πακέτων (source ip, port number, destination ip, port number, protocol_in_use). Το MPTCP συνδυάζει πολλαπλά TCP subflows σε μια κοινή σύνδεση όπου κάθε μια από αυτές έχει διαφορετική πεντάδα. Έτσι κάθε φορά που ένα καινούριο subflow συμμετέχει στο MPTCP αναγνωρίζεται με ένα 32-bit διακριτικό, token.

2. Για να επιτευχθεί η μετάδοση των δεδομένων χρειάζεται ένας αριθμός για να διατηρεί στη σειρά τα δεδομένα. Αυτός είναι ο data sequence number όπου καθορίζει τη θέση ενός segment μέσα σε ένα data stream. Και οι δύο πλευρές θα πρέπει να συμφωνούν στον ίδιο αρχικό αριθμό. (Initial Data Sequence Number / IDSN)

3. Το MPTCP χρειάζεται για να αναγνωρίζει πως ο χρήστης που ανοίγει ένα νέο subflow, είναι ο ίδιος που έχει ανοίξει και τα προηγούμενα. Για αυτό τον λόγο χρησιμοποιείται ένα 64-bit κλειδί.

Τα παραπάνω στοιχειά (token, IDSN, key) ανταλλάσσονται κατά την διάρκεια του handshake του αρχικού subflow.



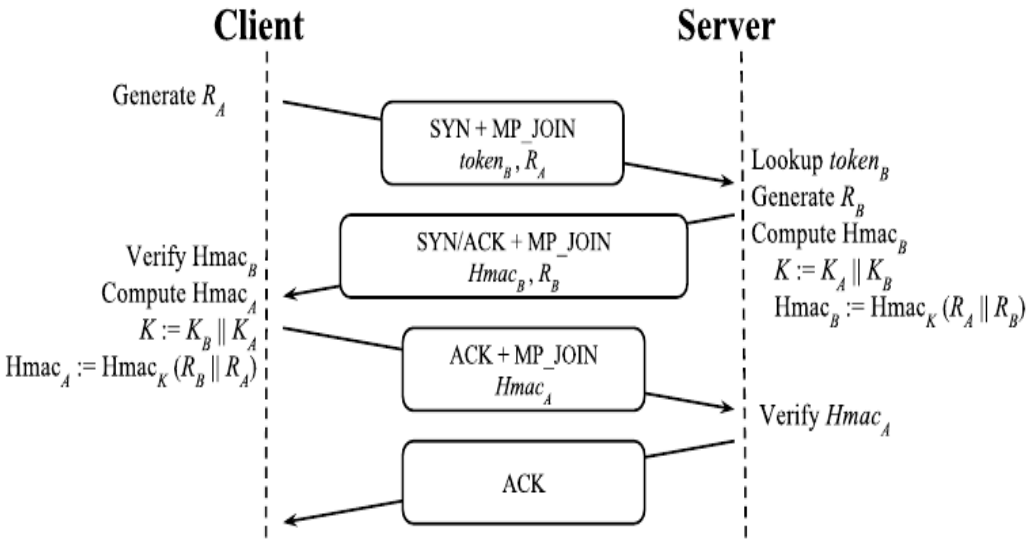
Σχήμα 5: *Handshake του αρχικού subflow*

Το handshake για το αρχικό MPTCP subflow χρησιμοποιεί το MP_CAPABLE option για να μεταφέρει τις κατάλληλες πληροφορίες για μια Multipath σύνδεση. Το MPTCP ,σύμφωνα με το *Σχήμα 4*, ξεκινά να εγκαταστήσει ένα αρχικό TCP subflow με ένα κλασσικό TCP 3-way handshake. Χρησιμοποιεί τα TCP options για να προσθέσει σηματοδοσία στο 3-way handshake έτσι ώστε να ανιχνεύσει εάν υποστηρίζεται το Multipath και να ανταλλάξει πληροφορίες για τα παραπάνω τρία στοιχεία. Όταν αντιλαμβάνεται πως υποστηρίζεται το Multipath TCP τότε προστίθεται ένα MP_CAPABLE μέσα στα SYN πακέτα. Τα κλειδιά στέλνονται μέσω του MP_CAPABLE option και επαναλαμβάνονται προς τα πίσω στο 3ο handshake.

2.2. Handshake των επιμέρους subflows

Με την προσθήκη ενός νέου subflow σε μια σύνδεση Multipath δημιουργούνται δυο προβλήματα. Το πρώτο είναι πως η ύπαρξη των Middle Boxes (πχ NAT) δεν βοηθά το νέο subflow που θέλει να συνδεθεί με μια υπάρχουσα Multipath σύνδεση με την πεντάδα των πληροφοριών (IPs, ports) του και δεύτερο το MPTCP θα πρέπει να είναι ισχυρό απέναντι σε επιθέσεις που θα προσπαθήσουν να προσθέσουν ένα δικό τους subflow σε μια υπάρχουσα σύνδεση. Το MPTCP λύνει αυτά τα προβλήματα

χρησιμοποιώντας ένα τοπικά μοναδικό token για το πρώτο και υπολογίζοντας και πιστοποιώντας ένα Hmac για το δεύτερο.

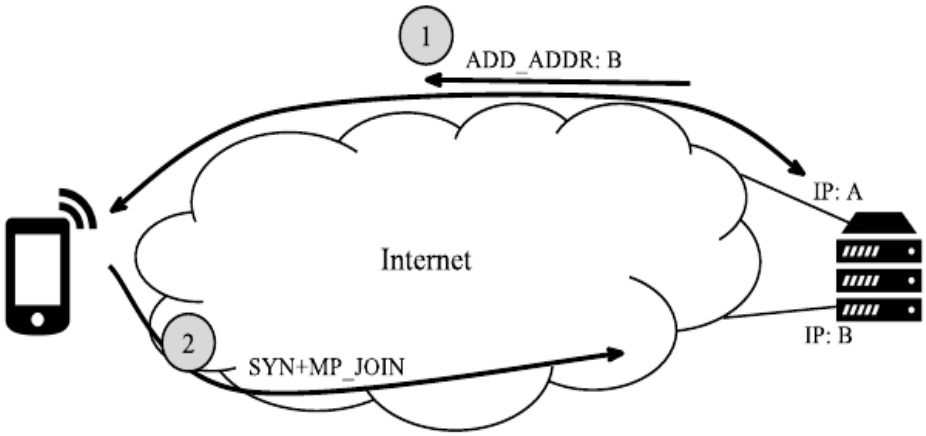


Σχήμα 6: *Handshake των επιμέρους subflows*

Το token χρησιμοποιείται ως διακριτικό σύνδεσης ενώ η ανταλλαγή HMAC πιστοποιεί τους τελικούς χρήστες και προσθέτει νέα subflows. Για να ανοίξει ένα νέο subflow το MPTCP εκτελεί μια νέα ανταλλαγή SYN χρησιμοποιώντας τις διευθύνσεις και τα ports που επιθυμεί. Ένα TCP option, MP_join, προστίθεται στο SYN. Το token που περιλαμβάνεται στο SYN δίνει την δυνατότητα στο MPTCP να καταλάβει σε ποια σύνδεση ανήκει το subflow, ενώ 32 random bits (RA) παίρνουν μέρος στον υπολογισμό του HMAC. Ο server υπολογίζει τον HMACB που ειναι βασισμένος σε δυο τυχαίους αριθμούς RA και RB και τα κλειδιά τα οποία έχουν ήδη ανταλλαγεί στο αρχικό handshake. Από τη στιγμή που θα λάβει ο client το SYN/ACK μπορεί να πιστοποιήσει την ορθότητα του HMACB κάτι που αποδεικνύει πως ο server γνωρίζει τα κλειδιά KA και KB και έτσι συμμετέχει στο αρχικό handshake. Στην συνέχεια ο client δημιουργεί ένα καινούργιο HMACA το οποίο περιέχεται μέσα στο 3^{ο} ACK.

Αυτό με τη σειρά του ενεργοποιεί τον server να αναγνωρίσει πως ο client είναι ο ίδιος με αυτόν που έκανε το πρωταρχικό handshake. Τέλος ο server στέλνει ένα διπλό acknowledgement στον client για να σηματοδοτήσει την λήψη του 3^{ου} ACK. Ο client θα αρχίσει να στέλνει δεδομένα μόνο μετά την λήψη του 4^{ου} ACK!

2.3. Address agility



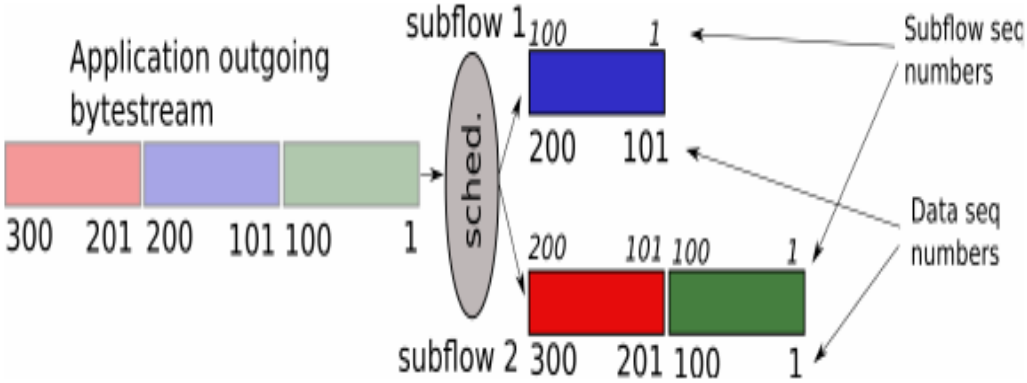
Σχήμα 7:*Address agility*

Ο server ανακοινώνει για επιπλέον διευθύνσεις που διαθέτει μέσω ενός subflow και ο client εγκαθιστά ένα επιπλέον subflow βάσει της διεύθυνσης που του ανακοινώθηκε. Εάν ένας client υποστηρίζει πολλά δίκτυα τότε μπορεί να ενεργοποιήσει νέα subflows από κάθε ξεχωριστή διεύθυνση. Αν όμως μόνο ο server έχει πολλαπλά δίκτυα τότε γίνεται αδύνατο ένα νέο SYN να παραληφθεί από τον client λόγω της μεγάλης χρήσης των NAT. Η λύση για το Multipath TCP server είναι να ενημερώνει τον client για τις επιπλέον διευθύνσεις που διαθέτει στέλνοντας ένα ADD_ADDR option σε ένα segment σε ένα από τα υπάρχοντα subflows (*Σχήμα 6*). Έτσι μπορεί ένα path να επιλεγεί ως back up σε περίπτωση προβλήματος του πρώτου.

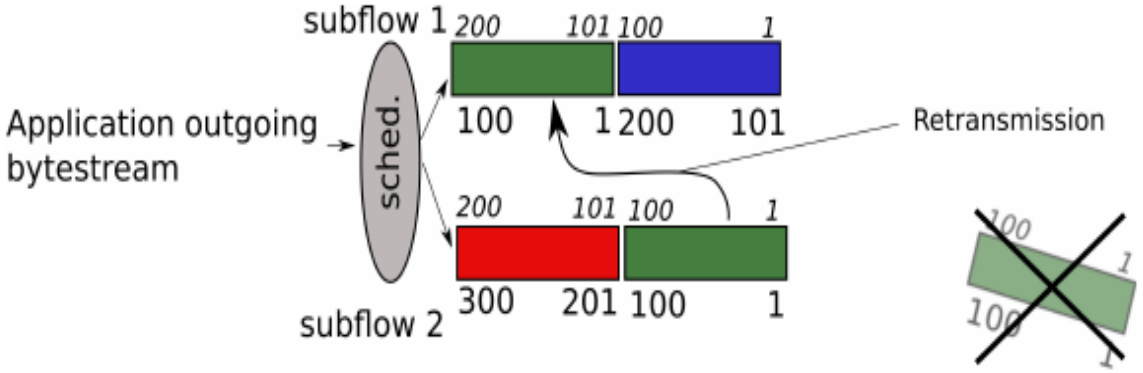
Το ADD_ADDRESS option είναι ένας επιπλέον τύπος TCP επιλογών που έχει διατηρηθεί για το MPTCP. Έτσι ο client μπορεί να ενεργοποιήσει το νέο subflow. Αυτή η διαδικασία δεν έχει κάποιον τεχνικό περιορισμό και έτσι μπορεί να γίνει εκατέρωθεν. Οι διευθύνσεις που ανταλλάσσονται μπορούν να είναι είτε IPv4 είτε IPv6. Έτσι μπορούν να υπάρξουν data streams και με τα δυο version IP ταυτόχρονα. Κάθε διεύθυνση εκχωρεί στο γνωστό address-ID, έναν 8-bit ακέραιο που προσδιορίζει τοπικά την IP-address. Χρησιμοποιείται στα πλαίσια του MP_JOIN option για να γνωρίζει ο host ποια ζευγάρια address-IDs αντιστοιχούν σε κάθε subflow, και επίσης αποτελεί κομμάτι του ADD_ADDR option. Όμως η κύρια χρήση του address-ID είναι το REMOVE_ADDR option. Όταν μια διεύθυνση δεν είναι διαθέσιμη τότε αφαιρείται και κλείνει και το αντίστοιχο subflow που χρησιμοποιεί αυτή την διεύθυνση.

3. Κεφάλαιο - Ανταλλαγή δεδομένων μεταξύ πολλαπλών ροών.

Για να γίνει η ανταλλαγή των δεδομένων μεταξύ των πολλαπλών subflows έχει εγκατασταθεί στον αποστολέα ένας scheduler (*Σχήμα 7* και *Σχήμα 8*) ο οποίος αποφασίζει από ποιο subflow θα σταλούν τα bytes των δεδομένων (βλ. 3.1). Αυτό δημιουργεί εκ των πραγμάτων ένα πρόβλημα καθώς δεν τηρείται η αρχική σειρά που έχει δοθεί από το application στο bytestream, με αποτέλεσμα να δημιουργούνται κενά, και με τη σειρά του αυτό οδηγεί στην ανάγκη δημιουργίας ενός δεύτερου sequence number space. Έτσι έχουμε το subflow sequence number το οποίο διατηρείται ξεχωριστά από κάθε subflow και το data sequence number το οποίο είναι κοινό και για τα δυο subflows.



Σχήμα 8: *MPTCP Data Sequence Numbers (DSNs)*

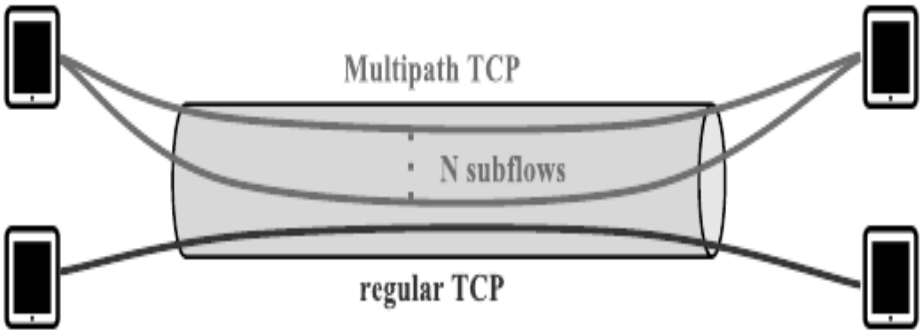


Σχήμα 9*: MPTCP retransmission*

Όταν ανιχνευτεί μια απώλεια (ξεχωριστά για κάθε subflow) μια έξυπνη επιλογή είναι να γίνει αναμετάδοση από άλλο subflow και όχι το ίδιο. Αυτό γίνεται με την επανατοποθέτηση των data sequence numbers του segment που χάθηκε σε νέα subflow sequence numbers του νέου subflow που έχει επιλεγεί (*Σχήμα 8*).

Όσον αναφορά τις επιβεβαιώσεις των απεσταλμένων / ληφθέντων δεδομένων (acknowledgements), θα μπορούσαμε να πούμε πως αφού κάθε subflow λειτουργεί σαν ένα απλό TCP subflow τότε θεωρητικά τα acknowledgements θα ανταλλάσσονται κανονικά μεταξύ των δύο άκρων. Όμως υπάρχουν δυο πρακτικά ζητήματα που δεν επιτρέπουν αυτήν την λειτουργία στο MPTCP. Το ένα είναι πως στον σχεδιασμό των δικτύων υπάρχουν διάφορα μηχανήματα που έχουν προστεθεί για να βελτιστοποιήσουν την συμπεριφορά του TCP (Perfomance Enchancing Proxies / middleboxes). Έτσι αφενός υπάρχει περίπτωση να χαθούν πακέτα εάν μια διαδρομή πέσει, αφετέρου το PEP δεν έχει την δυνατότητα να κάνει retransmission (λόγω απελευθέρωσης buffer). Το δεύτερο είναι ότι στο MPTCP δεν υπάρχει συγκεκριμένο receive window στο subflow.

Για να λυθεί αυτό το πρόβλημα ορίστηκε ένα νέο option, το data acknowledgement option. Έτσι το Data Acknowledgement μπορεί να μεταδοθεί μέσα από οποιαδήποτε subflow και να οριστεί το data level receive window. Οι αλγόριθμοι για τον έλεγχο της συμφόρησης (congestion control) στο TCP προσπαθούν να διαχειριστούν δίκαια την διαθέσιμη χωρητικότητα ενός διαύλου (βλ. 3.2). Αν όμως κρατούσαμε στο MPTCP το ίδιο μοντέλο αλγόριθμου με το TCP τότε δεν θα υπήρχε δικαιοσύνη ανάμεσα σε δυο πχ. χρήστες που χρησιμοποιούν το ίδιο bottleneck link (*Σχήμα 9*).

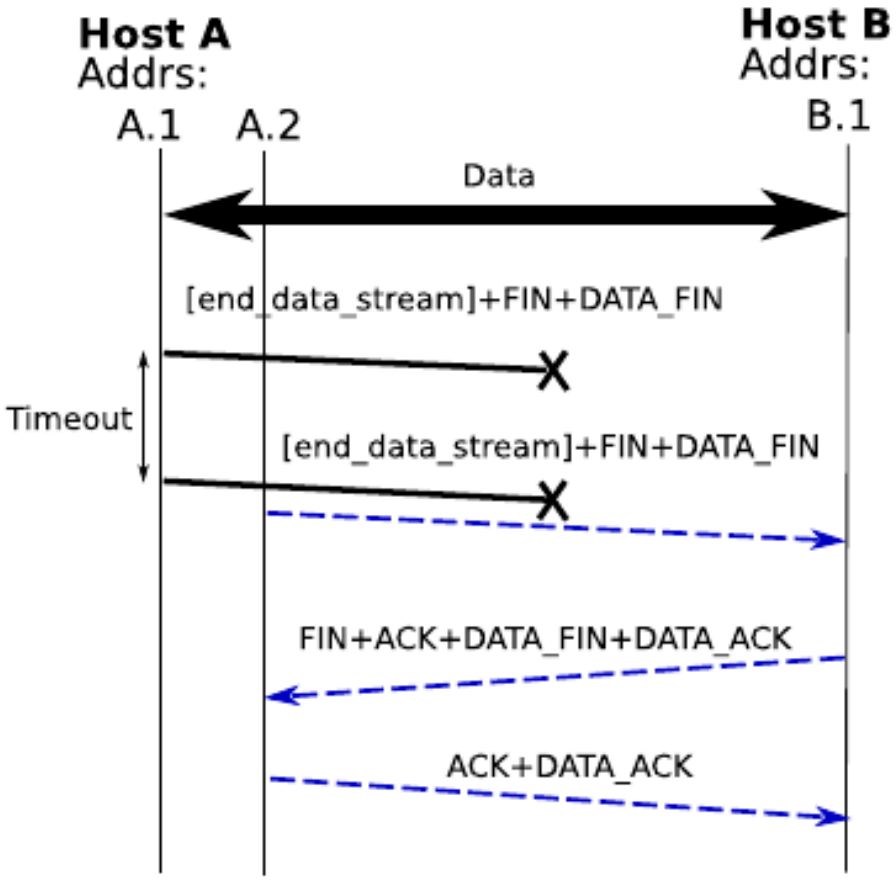


Σχήμα 10: *Δύο χρήστες χρησιμοποιούν το ίδιο bottleneck link.*

Αυτό το οποίο είναι επιθυμητό σε ένα multipath σενάριο είναι να χρησιμοποιούνται τα μονοπάτια με την λιγότερη συμφόρηση αντί να μοιράζεται η κίνηση ισόποσα σε όλα τα διαθέσιμα μονοπάτια. Έτσι ένα multipath flow θα πρέπει να δίνει τουλάχιστον τόσο throughput όσο θα είχε ένα απλό TCP σε μια σύνδεση στο καλύτερο μονοπάτι που διαθέτει. Αυτό διασφαλίζει το κίνητρο για την ανάπτυξη πολλαπλών διαδρομών. Επίσης ένα multipath flow δεν θα πρέπει να έχει παραπάνω χωρητικότητα σε οποιαδήποτε από τα μονοπάτια του ή σε συνδυασμό μονοπατιών από όση χωρητικότητα θα είχε ένα απλό TCP flow χρησιμοποιώντας τα καλύτερα από αυτά μονοπάτια. Εδώ διασφαλίζεται πως το multipath θα είναι δίκαιο απέναντι σε ένα άλλο TCP flow που θα μοιράζεται το ίδιο bottleneck link.

Αφού γίνει η ανταλλαγή των δεδομένων η σύνδεση πρέπει να τερματιστεί και να απελευθερωθούν οι πόροι. Όμως ένα απλό FIN σε κάθε subflow δεν θα ήταν αρκετό καθώς υπάρχει η περίπτωση να μην λειτουργεί κάποιο subflow. Για αυτό το σκοπό έχει προστεθεί ένα DATA FIN στο τελευταίο block στο retransmission queue το οποίο μπορεί να μεταδοθεί σε κάθε subflow (βλ. 3.3). Το παράδειγμα στο *Σχήμα 10* μας δείχνει τι συμβαίνει σε περίπτωση που αποτύχει ένα subflow.

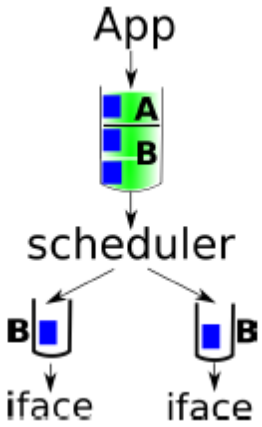
Ας υποθέσουμε πως το subflow <A1 , B1> έχει αποτύχει και το application στέλνει ένα close() system call για να τερματίσει την σύνδεση. Από τη στιγμή που το MPTCP έχει το DATA FIN ξέρει πως μπορεί να τερματίσει τα subflows. Στην περίπτωση μας όμως όπου το subflow <A1,B1> έχει αποτύχει το segment χάνεται και μετά από ένα time out προσπαθεί να το ξανά στείλει. Το MPTCP όμως έχει την δυνατότητα να κάνει το retransmission σε κάθε subflow που ειναι ενεργό. Έτσι το retransmission γίνεται από το <A2,B2> subflow. Από την πλευρά του application έχει ολοκληρωθεί ο τερματισμός και από την πλευρά του MPTCP θα γίνει εκ νέου retransmission του FIN segment μέχρι να γίνει backoff και τελικά να τερματιστεί μετά από timeout, μια διαδικασία η οποία εκτελείται ερήμην του application.



Σχήμα 11: *Παράδειγμα τερματισμού μιας σύνδεσης MPTCP*

3.1. Διαχείριση ροής δεδομένων.

Από τη στιγμή που το Multipath TCP έχει στη διάθεση του πολλαπλά subflows ένας scheduler είναι χρήσιμος για να αποφασίζει που θα αποστέλλεται κάθε byte δεδομένων. Στην παρούσα εκδοχή του MPTCP υιοθετήθηκε η άποψη πως τα data θα πρέπει να αποθηκεύονται κεντρικά στο Multipath Transport σε ένα buffer κοινής αποστολής (Σχήμα 11). Το scheduling τότε πραγματοποιείται κατά την διάρκεια της μετάδοσης και όταν οποιοδήποτε subflow ήταν έτοιμο να στείλει περισσότερα δεδομένα (σσ. αφού ληφθούν τα acknowledgments και ελευθερωθεί χώρος στο congestion window του subflow). Σε αυτή την περίπτωση τα subflows τραβούν segments από την share send queue όταν είναι έτοιμα. Στην περίπτωση που υπάρχουν πολλά subflows έτοιμα να τραβήξουν δεδομένα ταυτόχρονα τότε ο Packet Scheduler ενεργοποιείται και μπορεί να τα τροφοδοτήσει ταυτόχρονα.



***Σχήμα 12**: Δομή ροής αποστολής*

Έτσι κάθε subflow θα μπορεί να γεμίζει όσο υπάρχουν δεδομένα και ο scheduler δεν εφαρμόζει περιορισμούς. Εάν αποτύχει ένα subflow δεν θα δέχεται acknowledgments και ως εκ τούτου δεν θα τραβά data και σε περίπτωση που το subflow αυτό επανέλθει τα εκκρεμή segments που έχουν σχέση με το δικό του congestion window θα αναγνωριστούν και θα ελευθερωθεί χώρος για νέα segments. Στην παραπάνω περίπτωση αυτό που συμβαίνει είναι πως τα acknowledged data "πέφτουν" από τον παραλήπτη καθώς τα αντίστοιχα segments έχουν αναμεταδοθεί από άλλο subflow κατά την διάρκεια της βλάβης.

Στο Multipath Transport υπάρχει μια μοναδική ουρά από την οποία όλα τα subflows τραβούν segments. Στα Linux όμως για να επιτευχτεί καλύτερη απόδοση η διαδικασία της στοίχισης των δεδομένων σε μια ουρά διαχειρίζεται segments και όχι bytes. Η επιλογή του μεγέθους του segment ήταν ένα ζήτημα που απασχόλησε τον σχεδιασμό του MPTCP. Ως λύση προκρίθηκε η share send queue να περιέχει segments προκαθορισμένου μεγέθους και άρα είναι απαίτηση να χρησιμοποιείται το ίδιο MSS (Maximum Segment Size) για όλα τα subflows. Μετά από μια σειρά δοκιμών το MSS που επιλέχτηκε για το υπάρχον Internet είναι περίπου 1380 bytes [BPB]. Ένα ακόμη ζήτημα ήταν πως τα subflows τραβούν δεδομένα οπότε έχουν ελεύθερο χώρο στο congestion window τους, με αποτέλεσμα ο Packet Scheduler να τρέχει στιγμιαία και παράλληλα το σύστημα να δέχεται και acknowledgments. Μια διαδικασία που εκτελείται στο interrupt context με αποτέλεσμα να μην είναι δίκαια από πλευράς απόδοσης σε άλλες διαδικασίες του συστήματος, αλλά ελαφρώς αναποτελεσματική για τις υψηλότερης

ταχύτητας ροές. Μια λύση που θα μπορούσε να δοθεί είναι να υπήρχε μια μικρή καθορισμένη ουρά αποστολής ως subflow η οποία στην πραγματικότητα θα ήταν μια υβριδική αρχιτεκτονική μεταξύ pull και push προσέγγισης που θα δημιουργούσε περαιτέρω ζητήματα.

3.2. Σχεδιασμός δρομολόγησης δεδομένων.

Αφού στο MPTCP υπάρχουν πολλά subflow για την μετάδοση δεδομένων χρειάζεται να γνωρίζουμε ποια από αυτά είναι ενεργά. Ο μηχανισμός που κάνει αυτόν τον έλεγχο είναι ο congestion controller ο οποίος διατηρεί το congestion window για κάθε subflow. Ο στόχος για έναν multipath congestion controller είναι να γίνεται η μετάδοση των δεδομένων από ροές που θα υποφέρουν όσο το δυνατόν λιγότερο από συμφόρηση αλλά και να διασφαλίζει πως θα είναι δίκαιο απέναντι στις υπόλοιπες συνδέσεις σε ένα shared bottleneck.

Έχουν αναπτυχτεί αρκετοί αλγόριθμοι για τον υπολογισμό του congestion window. Μερικοί από αυτούς οι οποίοι λειτουργούν και στο MPTCP είναι ο Coupled Congestion Control (lia) [RFC 6356] και ο Opportunistic Linked-Increases Congestion Control Algorithm (Olia) [KHALI]. Όλοι τους βασίζονται σε μετρήσεις όπως το MSS, RTT και τον αριθμό των subflows.

Από τη στιγμή που ο Congestion controller επιτρέψει να αποσταλούν νέα πακέτα έστω και σε ένα subflow ο Packet Scheduler εκτελεί τις διαδικασίες βασιζόμενος σε τρεις αποφάσεις που πρέπει να πάρει.

Πρώτον επιλέγει σε ποιο subflow θα στείλει νέα data. Αυτό εξαρτάται από το εάν θα χρησιμοποιηθούν όλα τα subflow ταυτόχρονα ή θα διατηρηθούν paths μόνο για λόγους back up ή εάν χρειαστεί και νέο subflow στην περίπτωση που τα υπόλοιπα έχουν πλήρες το congestion window τους και δεν μπορούν να δεχτούν άλλα data.

Δεύτερον θα πρέπει να επιλέξει με ποια σειρά θα γεμίσει τα subflows σε περίπτωση που αυτά το ζητήσουν ταυτόχρονα. Για αυτόν το σκοπό ορίστηκε μια παράμετρος που ονομάστηκε time - distance και εκφράζει τον χρόνο που χρειάζεται ένα πακέτο να φτάσει στον αποδέκτη. Αυτό προσδιορίζεται από το RTT, το bandwidth και το queue size (σε bytes):

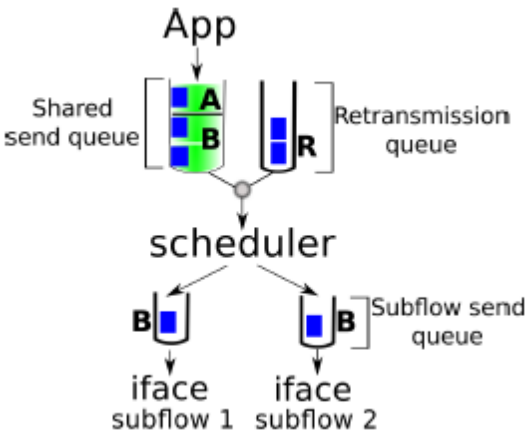
$$\text{time distance}_i = \text{queue size}_i/\text{bw}_i + \text{RTT}_i$$

Έτσι το subflow-specific queue size δεν μπορεί να υπερβεί το congestion window και το time distance$_i$ γίνεται περίπου ισο με RTT_i.

Τρίτον θα πρέπει να υπολογιστεί πόσα data θα αποδοθούν σε κάθε subflow. Χρησιμοποιώντας μεγάλες μονάδες θα υπήρχε μεγαλύτερη υποστήριξη του TCP Segmentation Offload (TSO). Αυτό θα έδινε την δυνατότητα να αθροίζονται αρκετά MSS σε ένα segment μειώνοντας την χρήση μνήμης και CPU και αφήνοντας την εργασία του fragmentation στις κάρτες των interfaces (NIC). Σε αντίθεση η χρήση μικρών μονάδων θα επέτρεπε μια ομοιόμορφη χρήση των subflows σε εφαρμογές μικρής κίνησης οι οποίες όμως θα μπορούσαν να είχαν εκτελεστεί με μια και μόνο μεγάλη μονάδα δεδομένων. Έτσι λόγω και της μη υποστήριξης του TSO στο MPTCP, υπολογίζει τα δεδομένα ανά MSS.

3.3. Αναμετάδοση - Retransmissions

Η αναμετάδοση δεδομένων είναι πιο περίπλοκη στο MPTCP σε σύγκριση με το απλό TCP και αυτό γιατί ένα χαμένο segment μπορεί να μεταδοθεί εκ νέου από οποιοδήποτε subflow. Στο σχήμα 12 μπορούμε να δούμε τον μηχανισμό μετάδοσης των δεδομένων. Μια συγκεκριμένη ουρά χρησιμοποιείται για τα retransmissions και βρίσκεται πάνω από το scheduler για να μπορεί να αναμεταδίδει τα χαμένα πακέτα σε οποιοδήποτε subflow. Από τη στιγμή που ένα retransmission μπορεί να καθυστερήσει την μεταφορά των δεδομένων , ο scheduler σταματά να στέλνει νέα πακέτα από την κοινή ουρά αποστολής μέχρι η κοινή ουρά αναμετάδοσης να αδειάσει. Με τη βοήθεια του σχήματος 12 θα προσπαθήσουμε να δούμε πως γίνεται αυτό.



Σχήμα 13: Μηχανισμός Retransmission

Ας υποθέσουμε πως έχουμε ένα host με δύο subflows. Το subflow 1 λαμβάνει ένα acknowledgement, ανανεώνει το congestion window και ζητά από τον scheduler νέα πακέτα. Επιπρόσθετα υποθέτουμε πως το retransmission queue είναι άδειο και έτσι ο scheduler παραχωρεί στο subflow 1 ένα segment μεγέθους S. Το segment αντιγράφεται αυτούσιο και παραμένει στο shared send queue (μέχρι να γίνει acknowledge στο connection level) και μια νέα δομή (πχ S1) που δείχνει στο segment data δίνεται στην ουρά αποστολής του subflow 1. Το S1 παίρνει subflow sequence number από το sequence number space του subflow 1. Στο S1 περιέχεται και ένα bitmap (path_bitmap(S)) το οποίο ανανεώνεται για να γνωρίζει πως το S έχει αντιγραφεί μόνο στο path 1 και ως εκ τούτου μπορεί να αντιγραφεί και στο path 2 εάν χρειαστεί.

Υποθέτουμε πως για το S1 δεν λαμβάνουμε acknowledge, μετά από μια παύση του subflow γίνεται εκ νέου αποστολή. Όμως ένα timeout μπορεί να είναι μια πιθανή βλάβη στο subflow και έτσι θα ήταν καλύτερα η αναμετάδοση να γίνει μέσω του δεύτερου subflow. Έτσι με τη χρήση του αλγορίθμου που φαίνεται στο σχήμα 13 το S προσαρτάται στο retransmit καθώς στο subflow 2 δεν έχει ποτέ καταχωρηθεί στο S. Ο scheduler αναλαμβάνει ρόλο και για να γίνει το retransmission από το subflow 2. (Εάν υπάρχουν πάνω δυο subflows τότε ο scheduler θα επιλέξει σε ποιο από αυτά θα γίνει το retransmission.)

```
func timeout(subflow sf):
        for each segment S in unacked segments(sf):
                for each path i in available paths():
                        if i is not in path bitmap(S):
                          /* There is at least one possible alternative
                           * subflow for retransmission */
                          enqueue segment(clone(S), retransmit queue);
                        break;
        schedule();
return;
```

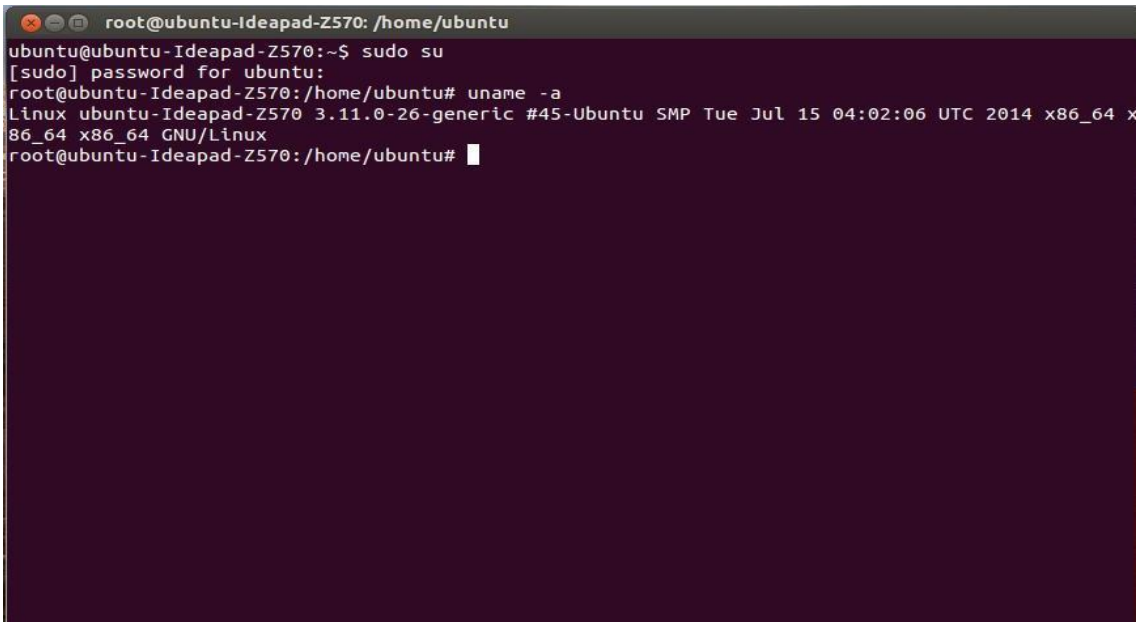
Σχήμα 14: *Retransmission algorithm*

Σε περίπτωση που το subflow 2 δεν δέχεται πακέτα , γιατί μπορεί να έχει στείλει ήδη ένα γεμάτο congestion window και περιμένει να πάρει acknowledgment ο scheduler περιμένει μέχρι το acknowledgment ανοίξει ελεύθερο χώρο στο congestion window του subflow 2 και αυτό με τη σειρά του να δεχτεί νέα πακέτα. Στη συνέχεια ο scheduler παίρνει data από το retransmission queue και μόνο όταν αυτό αδειάσει παίρνει νέα data από το send queue.

4. Κεφάλαιο - Εφαρμογή του Linux-MPTCP σε πραγματικό περιβάλλον

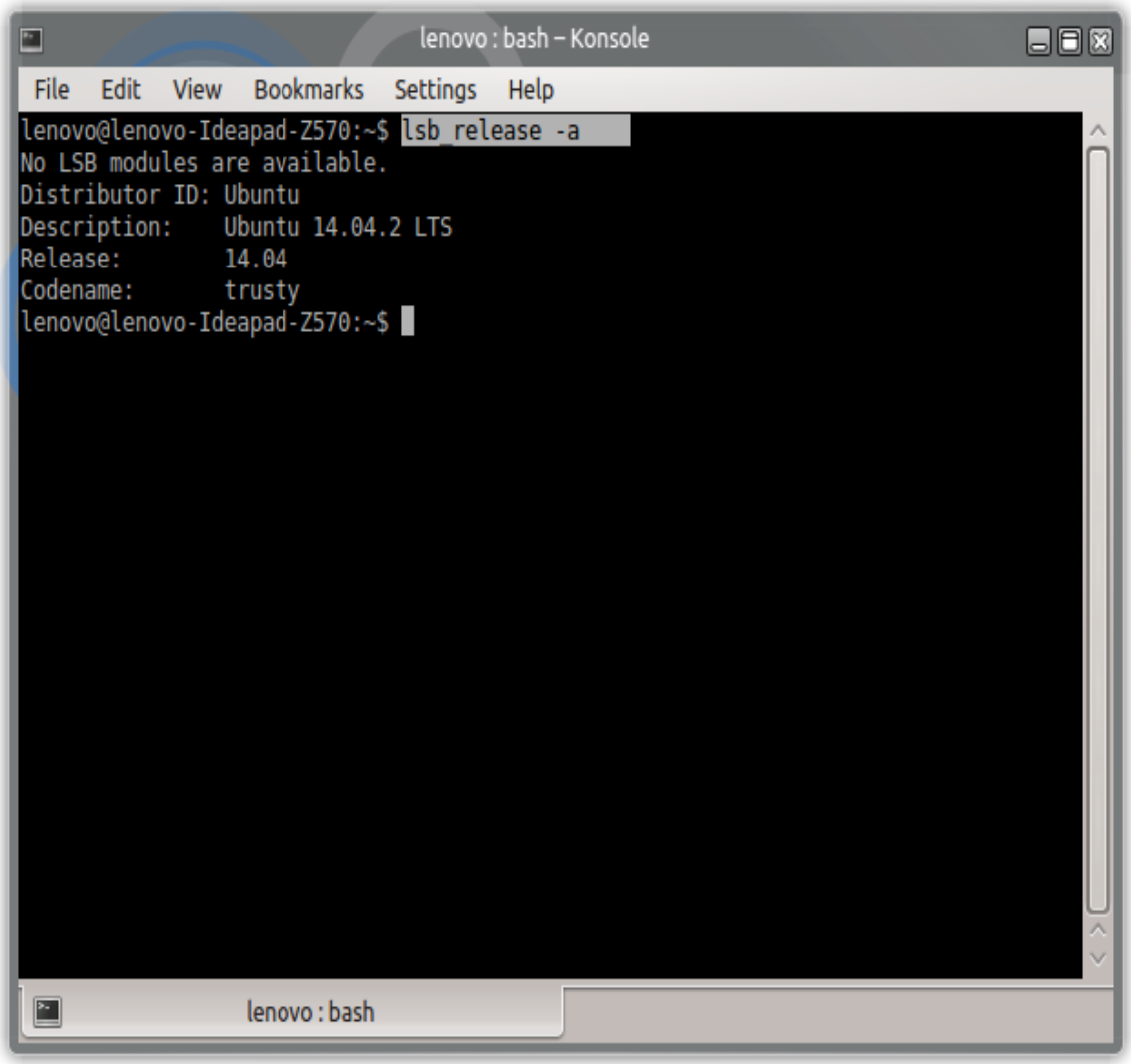
Για την εκτέλεση της multipath εφαρμογής σε πραγματικό περιβάλλον χρησιμοποιήθηκε υπολογιστής με λειτουργικό Linux Ubuntu 14.04 LTS στο οποίο έγινε εγκατάσταση του MPTCP kernel από το repository του Ip Networking Lab του Université catholique de Louvain (UCL) στην διεύθυνση http://multipath-tcp.org/. Στο σημείο αυτό θα πρέπει να αναφέρουμε ότι απαιτείται ιδιαίτερη προσοχή στην επιλογή του release των Linux που θα εγκαταστήσουμε καθώς το repository του mptcp είναι σχεδιασμένο για συγκεκριμένα releases. Όλη η διαδικασία της εγκατάστασης πραγματοποιείται στον terminal των Linux. Τέλος η εγκατάσταση μπορεί να γίνει είτε σε αυτόνομο περιβάλλον Linux, είτε σε παράλληλο με Windows άλλα ακόμη και σε virtual. Αφού λοιπόν επιλέξουμε το release των Linux (στο παρόν 14.01) και το εγκαταστήσουμε ανοίγουμε το **terminal**.

Χρησιμοποιώντας την εντολή **sudo su** παίρνουμε admin permissions και εκτελώντας την εντολή **uname -a** μπορούμε να δούμε πληροφορίες για το pc μας όπως, όνομα (kernel), λειτουργικό (Linux Ubuntu 14.04 LTS).



Εικόνα 1: *Έξοδος της εντολής uname -a*

Η εντολή **lsb_release –a**, μας δίνει το release της έκδοσης των linux που έχουμε



Εικόνα 2: *Έξοδος εντολής lsb_release -a*

Πριν απ όλα και κατά την διάρκεια της εγκατάστασης θα χρειαστεί να εκτελέσουμε αρκετές φόρες την εντολή **apt-get update** ή **sudo apt-get update** (αν δεν έχουμε τα permissions) για να ανανεωθούν οι λίστες των packages και να γίνουν update εάν χρειάζεται. Επιπλέον η εντολή **apt-get upgrade** ανανεώνει τα πακέτα που είναι εγκατεστημένα. Εκτελώντας την εντολή

wget -q -O - http://multipath-tcp.org/mptcp.gpg.key | sudo apt-key add -

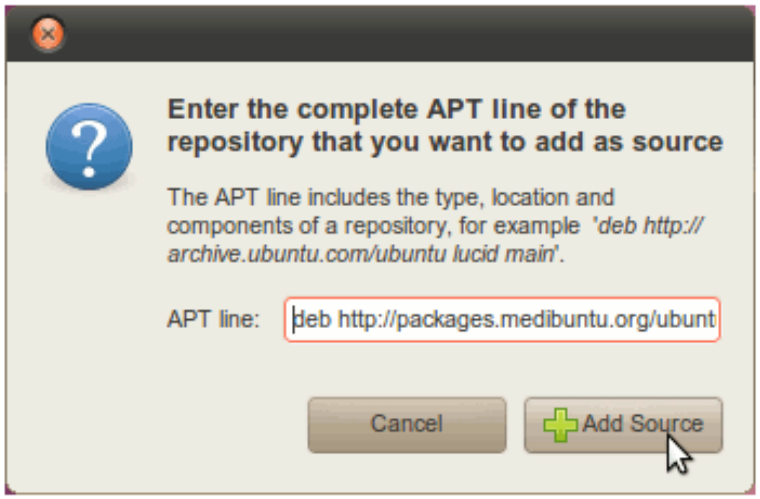
εγκαθιστούμε ένα **gpg-apt-key.**

Στη συνέχεια ανοίγουμε το Software and Updates (ή το *Synaptic Package*

Manager) και προσθέτουμε ένα νέο new software repository στον φάκελο /etc/apt/sources.list.d/mptcp.list με την παρακάτω εντολή

deb http://multipath-tcp.org/repos/apt/debian trusty main

(On an Ubuntu Trusty (14.04) for the newest v0.89-release)



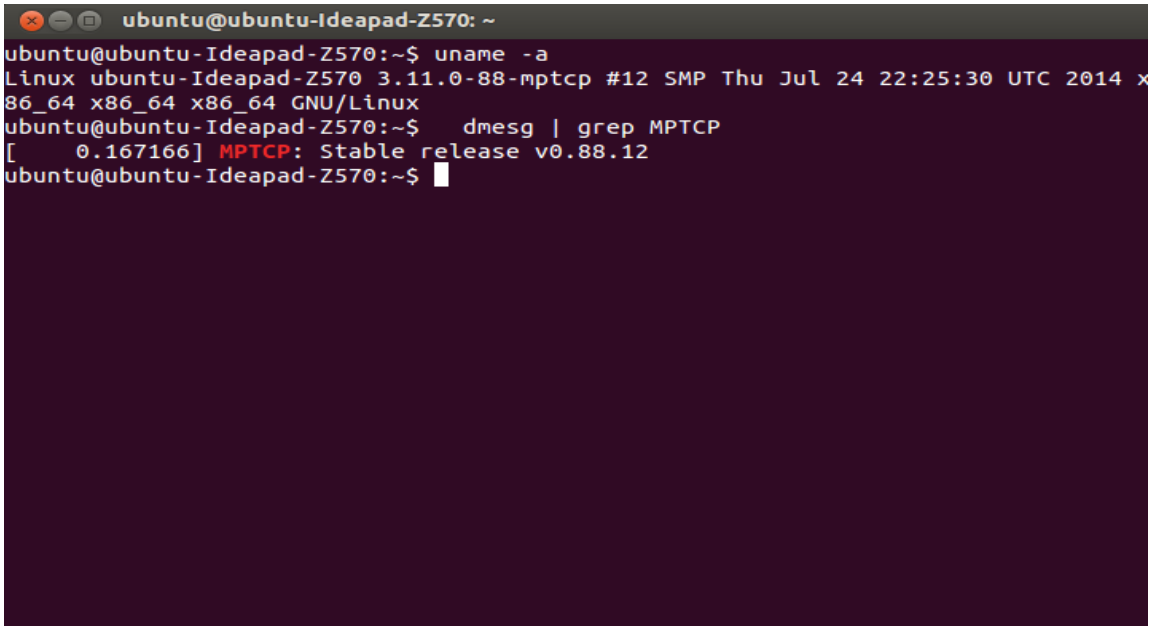
Εικόνα 3: *Εισαγωγή software repository περισσότερες πληροφορίες στο :https://help.ubuntu.com/community/Repositories/Ubuntu)*

Έπειτα εκτελούμε την εντολή **sudo apt-get update** και ακολουθούμε τις οδηγίες για να γίνει το **download των libraries**.

Ιδιαίτερη προσοχή απαιτείται στις ενημερώσεις ή την εγκατάσταση διάφορων επιπλέον libraries που θα χρειαστεί το σύστημα. Αν υπάρξει το οποιαδήποτε κόλλημα σε μη εγκατεστημένο πρόγραμμα ή library (libdecoration0, compiz-gnome, iproute2, compiz-core, compiz, libcompizconfig0, net-tools) απλώς κάνουμε copy το όνομα του και επαναλαμβάνουμε τη διαδικασία (πχ. **sudo apt-get install compiz-gnome**)

Μετά την ολοκλήρωση των παραπάνω, εκτελούμε την εκτολή **sudo apt-get install linux-mptcp** έτσι ώστε να εγκαταστήσουμε το **mptcp kernel.**

Εκτελώντας εκ νέου την εντολή **uname -a** βλέπουμε τον νέο kernel και με την εντολή **dmesg | grep MPTCP** βλέπουμε την έκδοση του mptcp.



***Εικόνα 4**: Έχουμε νέο kernel MPTCP με έκδοση v.0.88.12*

Από τη στιγμή που έχουμε πολλαπλά interfaces θα έχουμε και πολλαπλές διευθύνσεις και θέλουμε να έχουμε την δυνατότητα να πούμε στον kernel πως αν θέλουμε να χρησιμοποιήσουμε συγκεκριμένη διεύθυνση, να χρησιμοποιήσει συγκεκριμένο interface και gateway και όχι τα προεπιλεγμένα που υπάρχουν στο network config του linux. Αυτό επιτυγχάνεται με την διαμόρφωση ενός **routing table** ανά **εξερχόμενο interface.** Κάθε routing table προσδιορίζεται από έναν αριθμό.

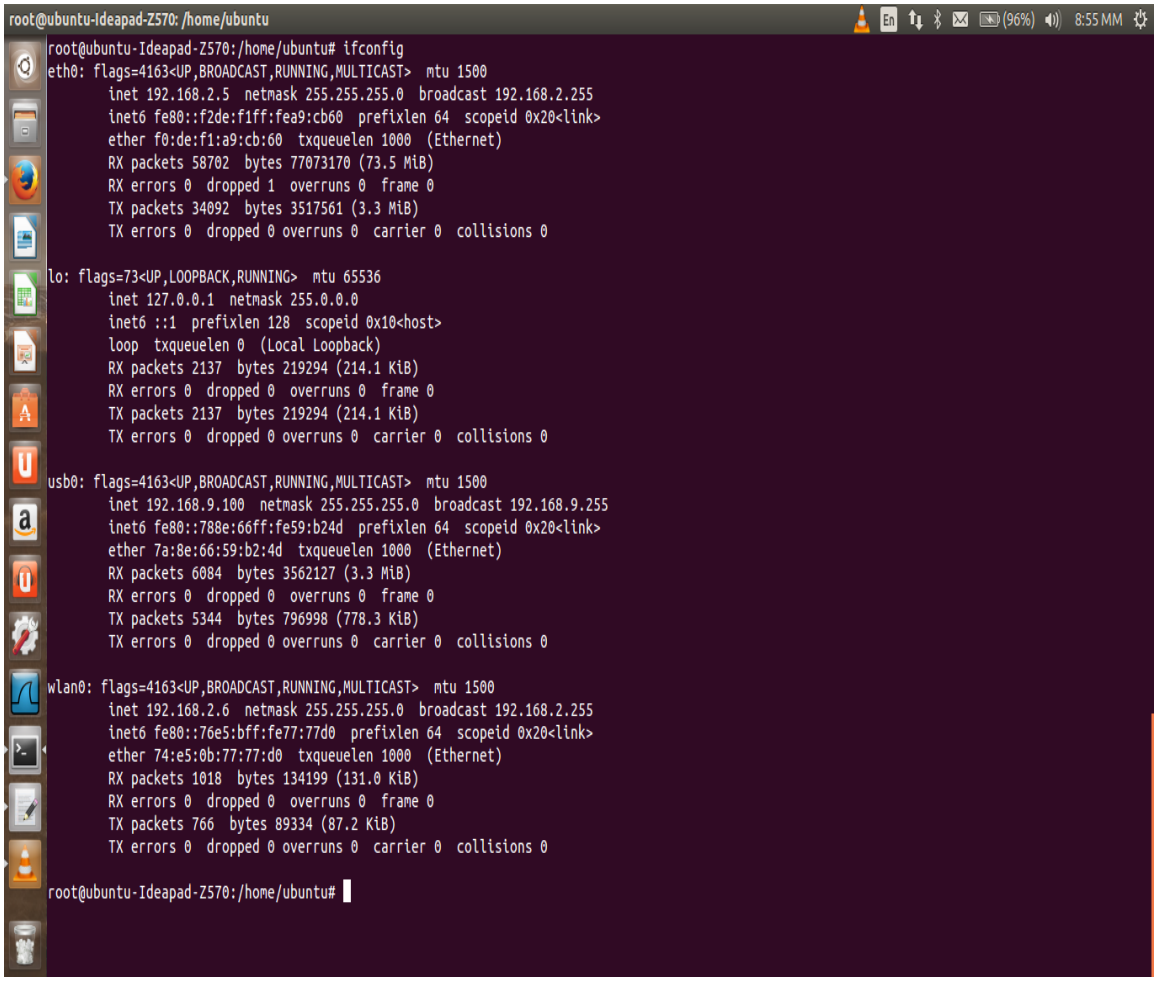
Η διαδικασία επιλογής διαδρομής συμβαίνει σε δύο φάσεις.

I. Πρώτα ο πυρήνας κάνει μια αναζήτηση στον policy table (ο οποίος πρέπει να ρυθμίστεί με τα ip rules). Στη δική μας περίπτωση τα policies, είναι, για y πρόθεμα IP πηγής, πηγαίνουμε στο routing table με αριθμό x.

II. Στη συνέχεια, ο αντίστοιχος routing table εξετάζει ποιο gateway θα επιλέξει με βάση τη διεύθυνση προορισμού.

Στην παρούσα περίπτωση έχουμε ένα σύστημα που διαθέτει ένα **ethernet interface**, ένα **Wi-Fi interface** και **ένα 3G dongle mobile interface**.

Με την εντολή **ifconfig** μπορούμε να δούμε τα interface του συστήματός μας (eth0 , wlan0 , usb0).



Εικόνα 5: *Τα interfaces τα οποία έχω είναι: eth0 για την Ethernet , wlan για το wireless και usb0 για το 3G. Το lo είναι η loopback του linux.*

Η κατασκευή των **routing policies** και των **routing tables** μπορεί να γίνει είτε εκτελώντας εντολές στο terminal είτε αυτόματα μέσω scripts που υπάρχουν στην κοινότητα.

Για εκπαιδευτικούς λόγους στην παρούσα εργασία επιλέξαμε τον πρώτο τρόπο αλλά θα πρέπει να επισημανθεί πως θα πρέπει να εκτελείται κάθε φορά που το σύστημα μας απενεργοποιείται ή κλείνουν τα interface καθώς οι πίνακες δεν αποθηκεύονται.

Έτσι έχουμε:

routing policies

ip rule add from 192.168.2.5 table 1

ip rule add from 192.168.2.6 table 2

ip rule add from 192.168.9.100 table 3

routing tables

ip route add 192.168.2.0/24 dev eth0 scope link table 1

ip route add default via 192.168.2.1 dev eth0 table 1

ip route add 192.168.2.0/24 dev wlan0 scope link table 2

ip route add default via 192.168.2.1 dev wlan0 table 2

ip route add 192.168.9.0/24 dev usb0 scope link table 3

ip route add default via 192.168.9.1 dev usb0 table 3

Θα παρατηρήσετε πως για το eth0 και το wlan0 χρησιμοποιήθηκε το ίδιο default way και αυτό γιατί στο σύστημα που δουλέψαμε το ethernet και το Wi-Fi κατέληγαν σε ένα οικιακό router.

Με τις παρακάτω εντολές μπορούμε να δούμε τα αποτελέσματα των όσων από τα παραπάνω έχουμε εκτελέσει για την κατασκευή των routing policies και routing table:

sudo ip rule show

sudo ip route

sudo ip route show table x (όπου x το νούμερο του πίνακα)

Στη συνέχεια με τις παρακάτω εντολές ενεργοποιούμε το mptcp για τα interfaces:

ip link set dev eth0 multipath on

ip link set dev wlan0 multipath on

ip link set dev usb0 multipath on

Για την απενεργοποίηση απλώς από on το κάνουμε off.

Προσοχή στην ενεργοποίηση και απενεργοποίηση των interfaces καθώς όταν απενεργοποιούνται χάνονται οι πινάκες δρομολόγησης γιατί δεν αποθηκεύονται. Η απενεργοποίηση του interface γίνεται με την εντολή **ip link set dev eth0 down** και ip link set dev eth0 up για να ανέβει πάλι. Αρκετές φορές βοηθά στο να καταλαβαίνει ο mptcp πότε ενεργοποιείται ένα interface για να το εντάξει στην διαδικασία multipath αλλά μόνο όταν χρησιμοποιούμε script για αυτόματη ρύθμιση routing table. Για την ενεργοποίηση και απενεργοποίηση των Interface στο MPTCP υπάρχουν οι εντολές:

ip link set dev (int) multipath on

ip link set dev (int) multipath off

Επίσης υπάρχει η δυνατότητα να γίνουν επιπλέον ρυθμίσεις του MPTCP όπως: **sysctl -w net.mptcp.mptcp_enabled=1 (default)** για την ενεργοποίηση του mptcp,

sysctl -w net.mptcp.mptcp_checksum=1 (default)

sysctl -w net.mptcp.mptcp_syn_retries=3 (default),

όπου καθορίζει το πόσες φορές θα κάνει retransmit ένα SYN με MP_CAPABLE-option

Για τον καθορισμό του Path manager οι εντολές είναι οι παρακάτω:

sysctl net.mptcp.mptcp_path_manager=fullmesh (default)

options: default

fullmesh

ndiffports

binder

Για τον καθορισμό του Scheduler:

sysctl net.mptcp.mptcp_scheduler=default (default)

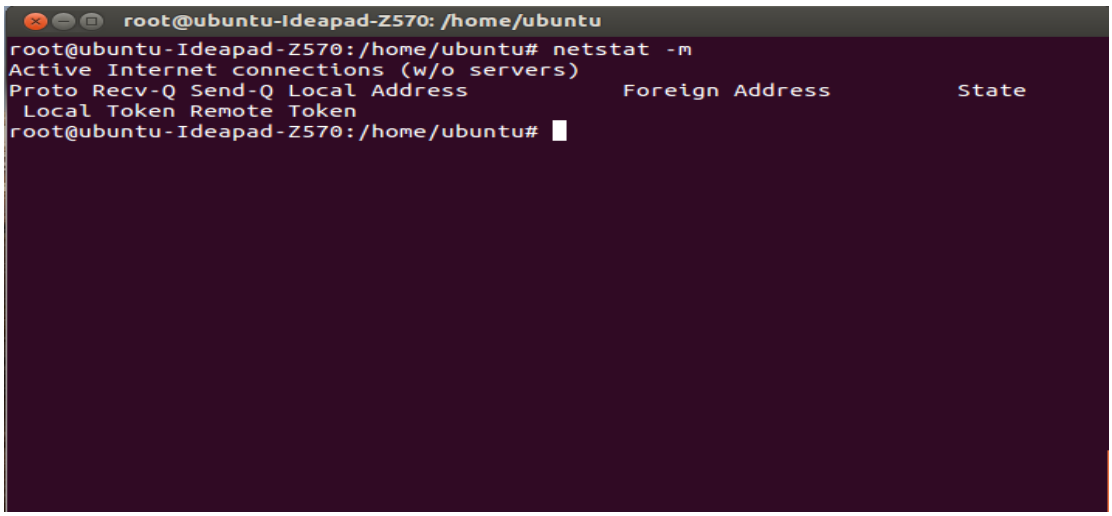
options: default

roundrobin

Για τον καθορισμό του Congestion control, υπάρχει η δυνατότητα έλεγχου του congestion control που χρησιμοποιείται με την εντολή

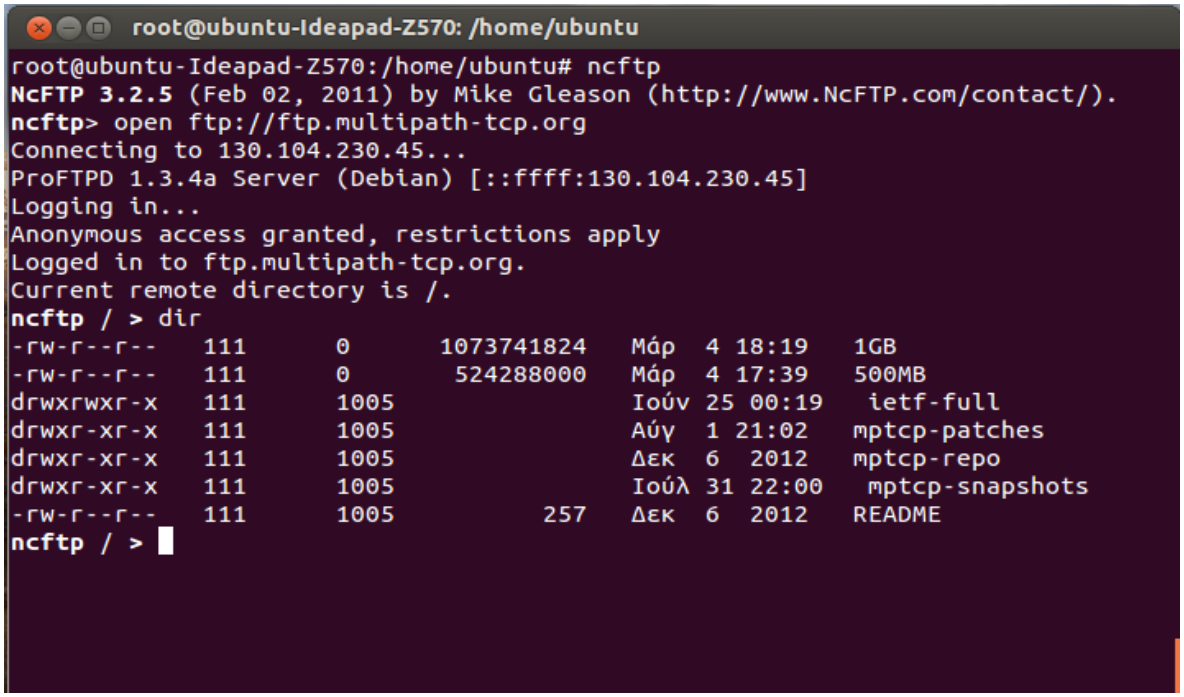
cat /proc/sys/net/ipv4/tcp_congestion_control

καθώς επίσης να δούμε ποιους αλγόριθμους έχουμε την δυνατότητα να χρησιμοποιήσουμε **cat /proc/sys/net/ipv4/tcp_available_congestion_control** και να τον αλλάξουμε **sysctl -w net.ipv4.tcp_congestion_control=[value]**



Εικόνα 6: *Με την εντολή netstat -m μπορούμε να δούμε εάν υπάρχει ενεργή multipath σύνδεση. Στην προκειμένη περίπτωση δεν υπάρχει καμιά.*

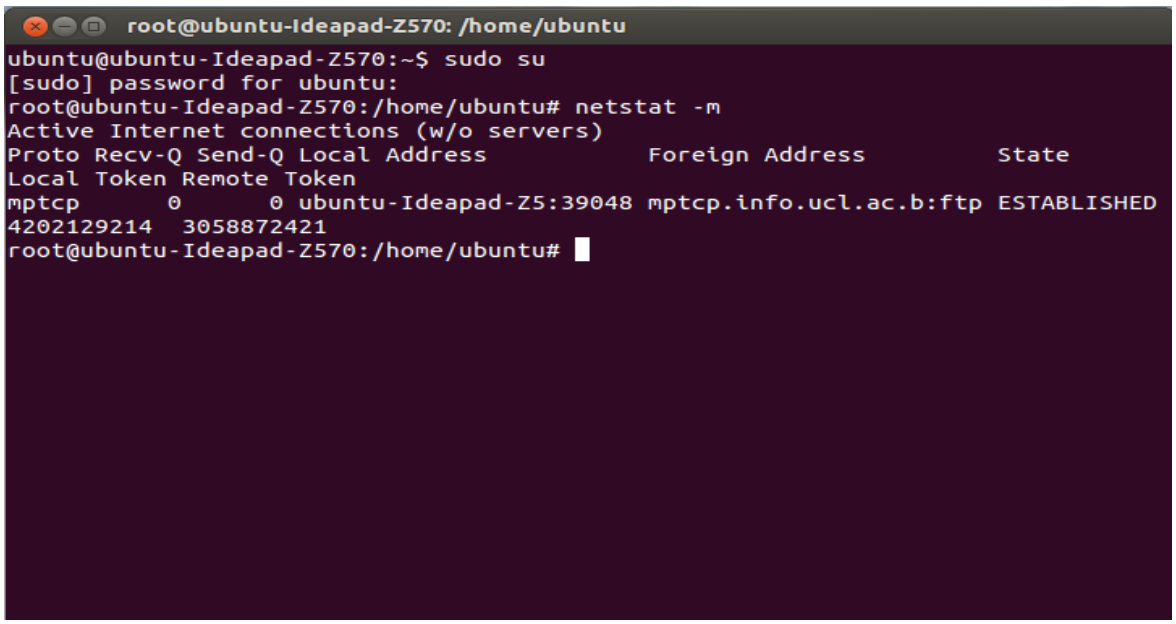
Μέσω της εφαρμογής ncftp μπορούμε να συνδεθούμε στον multipath ftp server ftp://ftp.multipath-tcp.org και να πλοηγηθούμε σε αυτόν και να κατεβάσουμε ένα



Εικόνα 7: *Εφαρμογή ncftp για ftp transfer.*

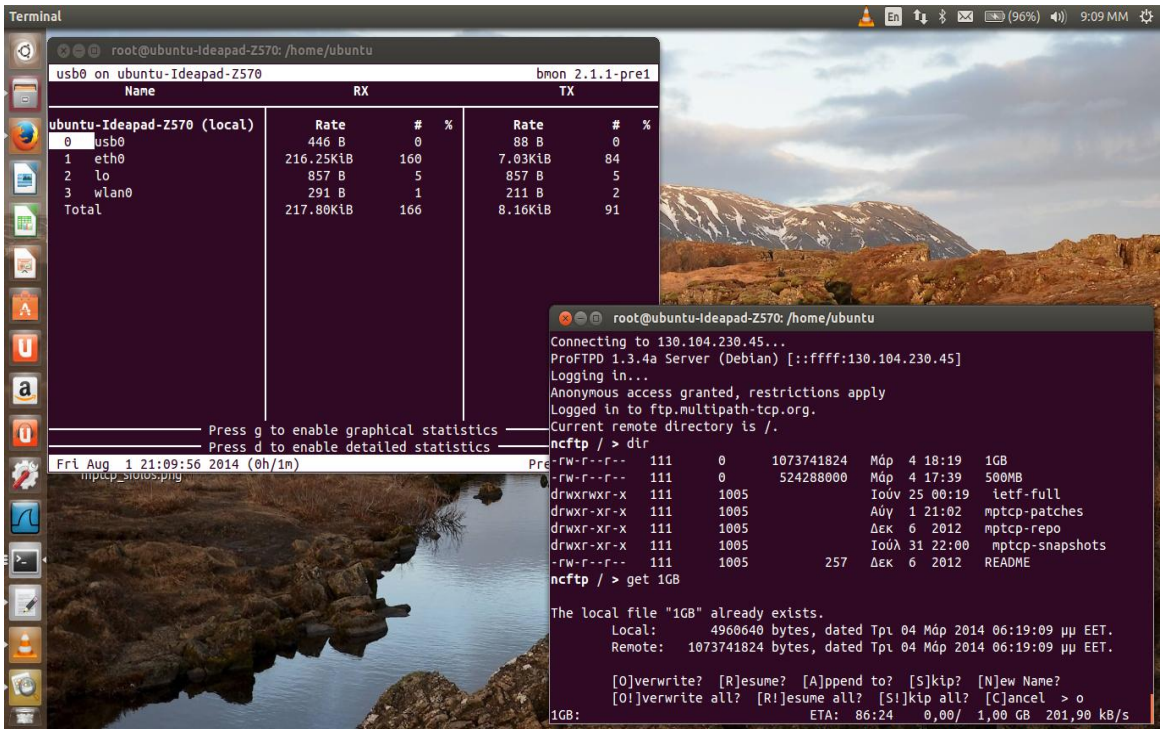
διαθέσιμο αρχείο.

Έτσι πάλι με την εντολή **netsat -m** βλέπουμε πως έχει γίνει η εγκατάσταση της σύνδεσης με τον multiptah ftp server.



Εικόνα 8: *mptcp network establishment*

Με το network monitoring bmon έχουμε τη δυνατότητα να βλέπουμε τα πακέτα που αποστέλλονται και λαμβάνονται από όλα μου τα interfaces!

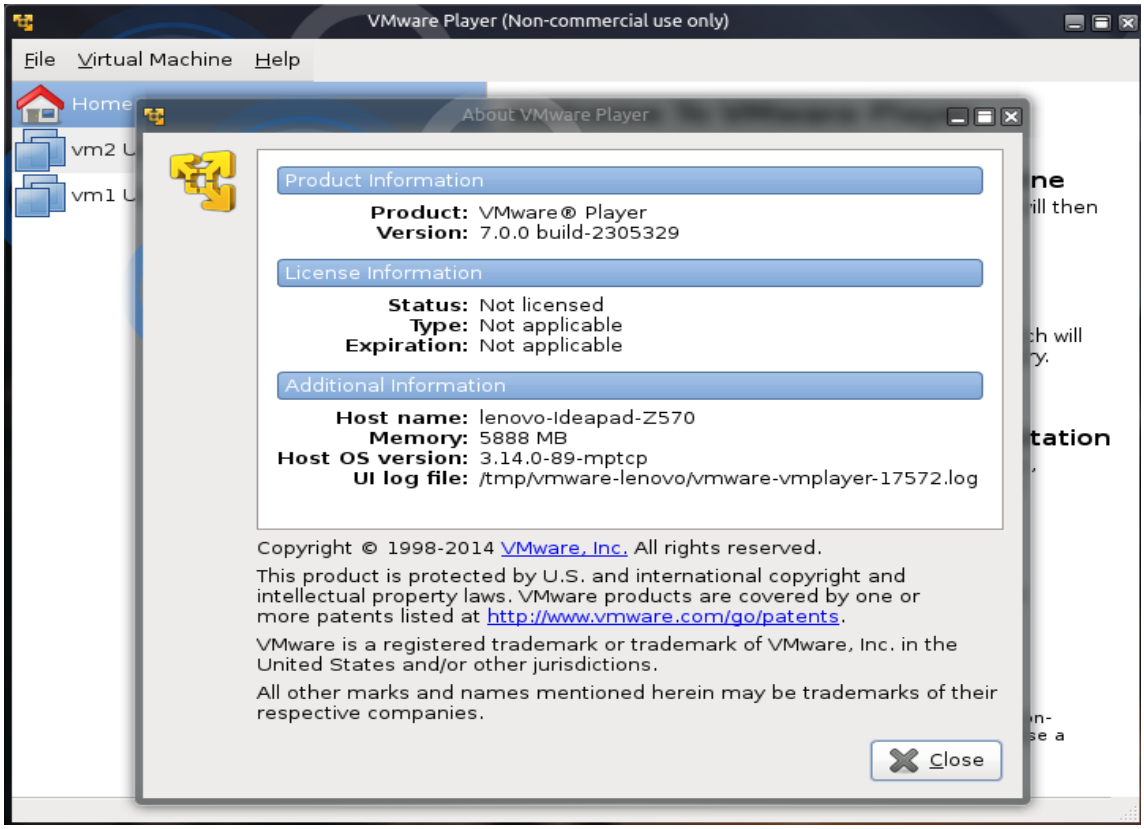


Εικόνα 9: *bmon network monitoring*

Ρίχνοντας ένα interface θα παρατηρήσουμε την αλλαγή στην κίνηση των υπόλοιπων interfaces που είναι ενεργά.

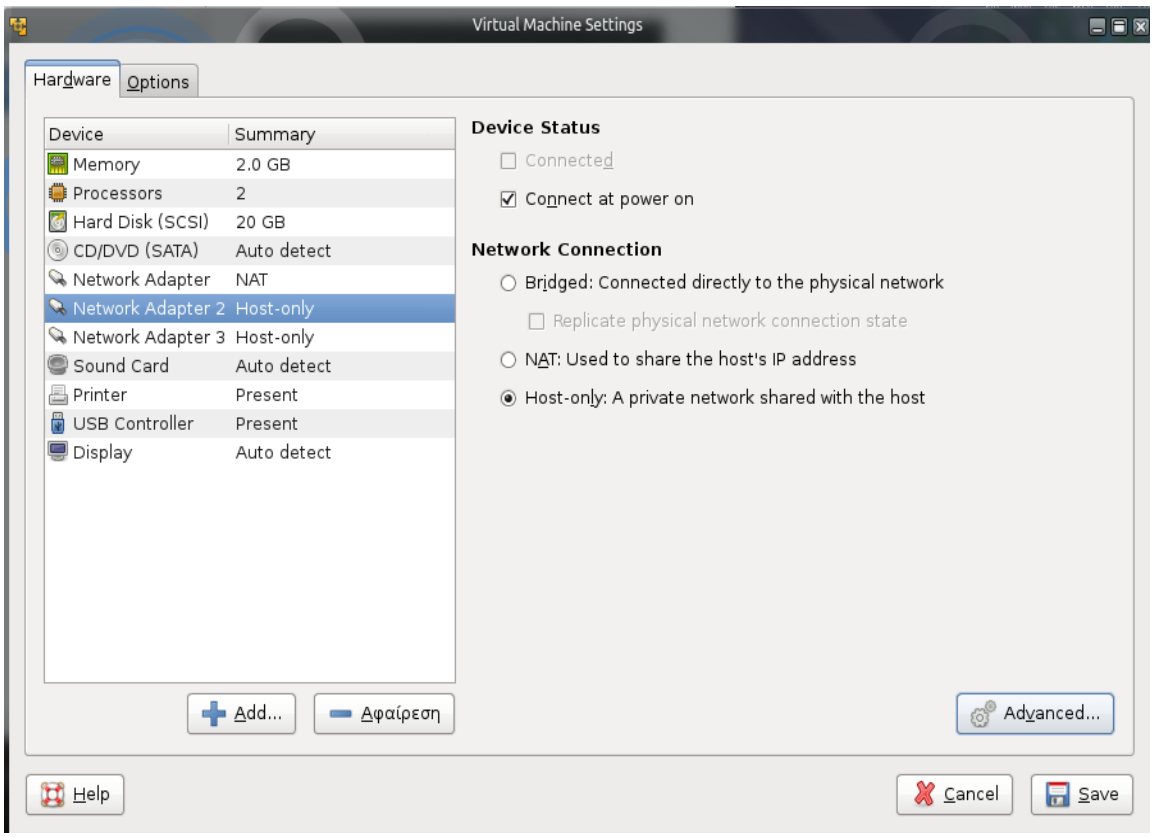
5. Κεφάλαιο - Εφαρμογή του Linux-MPTCP σε virtual περιβάλλον

Σε περίπτωση που δεν υπάρχει η δυνατότητα να εγκαταστήσουμε τα Linux σε ένα πραγματικό σύστημα τότε μπορούμε να χρησιμοποιήσουμε εικονικά μηχανήματα (virtual machines). Για αυτό τον σκοπό χρησιμοποιήθηκε το Vmware Player για να εγκατασταθούν το Linux και το MPTCP, με τον τρόπο που περιγράψαμε παραπάνω (**Κεφάλαιο 4**) .



Εικόνα 10: *Vmware about*

Θα πρέπει να δοθεί έμφαση στο μνήμη που θα χρησιμοποιηθεί, στον χώρο του virtual disc καθώς και στα interfaces που θα χρησιμοποιηθούν. Κάθε πόρος που χρησιμοποιείται επιβαρύνει το πραγματικό σύστημα. Η επιλογή που έγινε για δυο virtual machines ήταν 2GB RAM, 2 processors, 20GB hard disk και 3 ethernet interfaces εκ των οποίων το ένα είναι NAT για να υπάρχει σύνδεση με το Internet και 2 host-only στα οποία θα ορίσουμε εμείς IP.



Εικόνα 11: *Τα settings του virtual machine.*

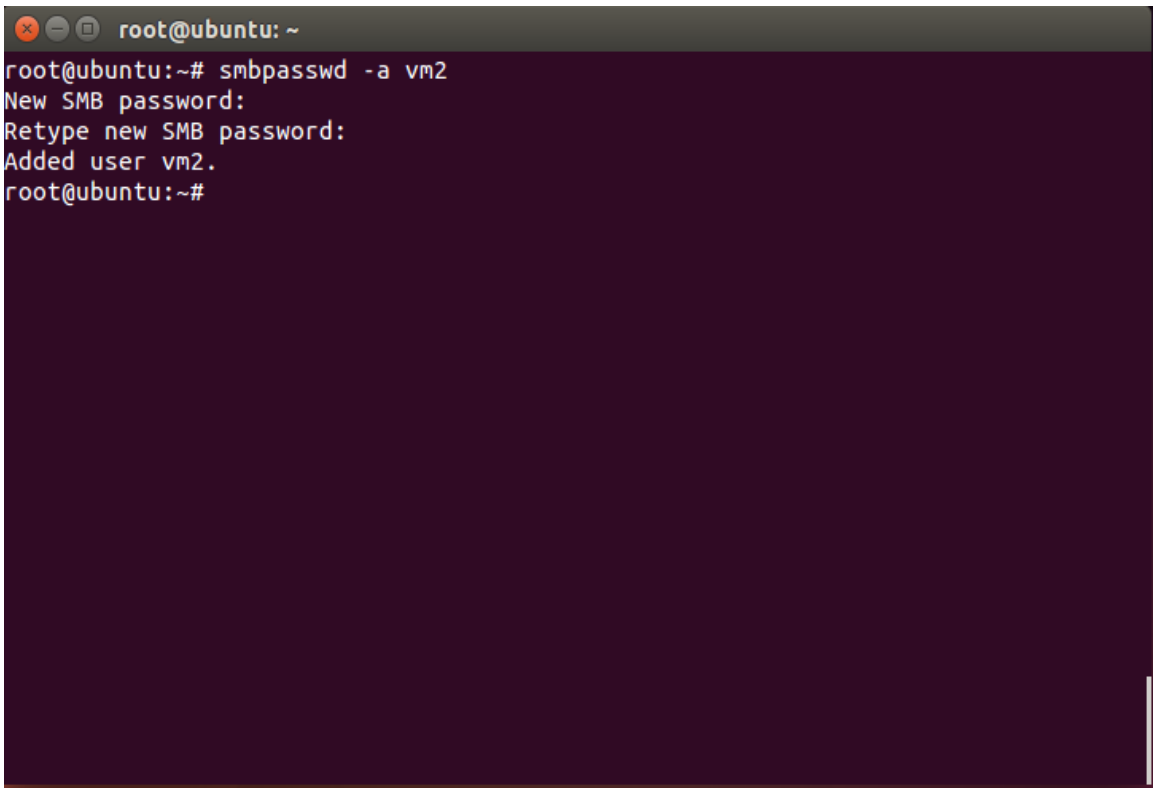
Για την ανταλλαγή αρχείων χρησιμοποιήθηκε το **Samba File Server** (για το οποίο πληροφορίες μπορείτε να βρείτε στο)

https://help.ubuntu.com/12.04/serverguide/samba-fileserver.html

Για να μπορέσει να γίνει μεταφορά αρχείων μέσω του Samba θα πρέπει σε κάθε host να προσθέσουμε έναν network user. Η προσθήκη γίνεται με την εντολή

sudo smbpasswd -a <username>

όπου username το όνομα του host που θέλουμε να ορίσουμε.



Εικόνα 12: *Προσθήκη χρήστη στο Samba*

Σε κάθε host ορίζουμε ένα share folder στον οποίο θα γίνονται όλες οι μεταφορές των αρχείων και φτιάχνουμε και ένα routing table για κάθε virtual machine.

Για δυο virtual machines έγινε ο παρακάτω σχεδιασμός:

VM 1 | | VM 2

|10.0.0.1/24 10.0.0.2/24 |

eth0 o----------o eth0

| |

eth1 o----------o eth1

|10.0.1.1/24 10.0.1.2/24 |

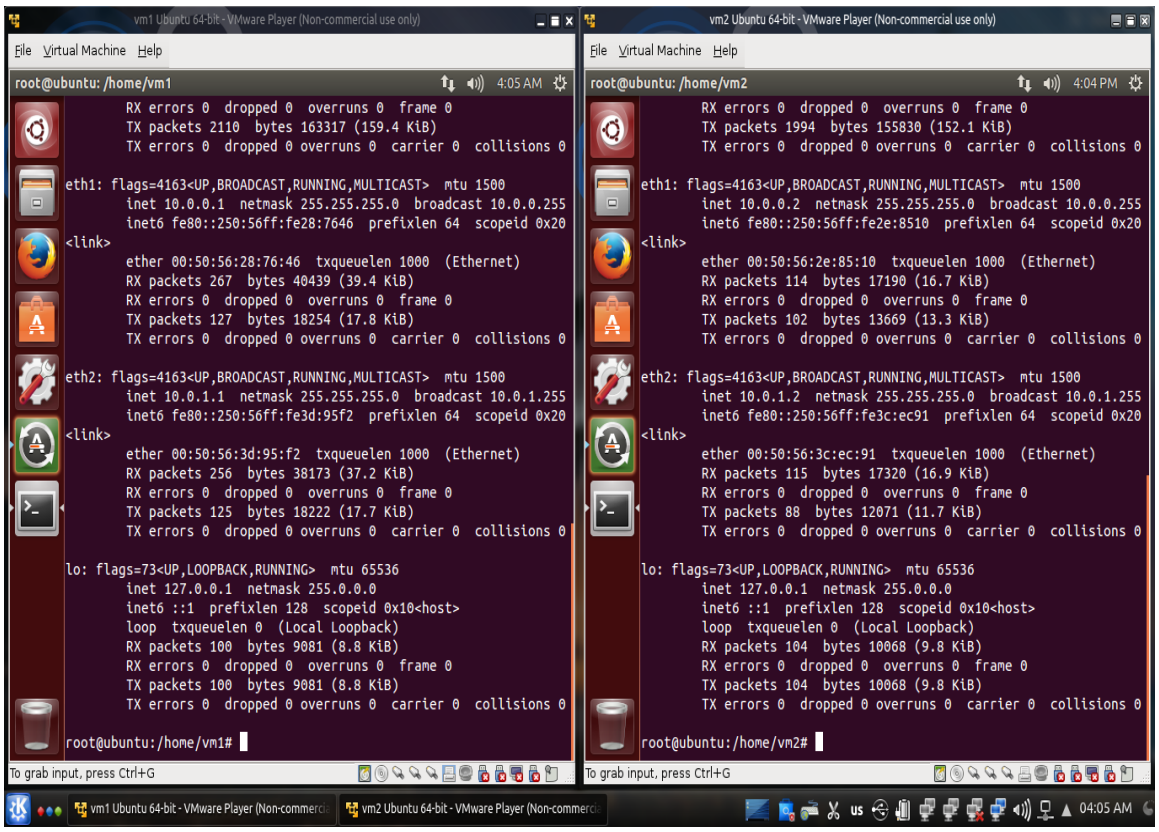
Έτσι προκύπτουν τα παρακάτω routing tables:

Routing Configuration for vm1:

```
#VM 1 routing configuration:

ip rule add from 10.0.0.1 table 1

ip rule add from 10.0.1.1 table 2


#Configure the two different routing tables

ip route add 10.0.0.0/24 dev eth1 scope link table 1

ip route add default via 10.0.0.244 dev eth1 table 1


ip route add 10.0.1.0/24 dev eth2 scope link table 2

ip route add default via 10.0.1.244 dev eth2 table 2


 #Enable interfaces for MPTCP

ip link set dev eth1 multipath on

ip link set dev eth2 multipath on
```

Routing Configuration for vm2:

```
#VM 2 routing configuration:

ip rule add from 10.0.0.2 table 1

ip rule add from 10.0.1.2 table 2


#Configure the two different routing tables

ip route add 10.0.0.0/24 dev eth1 scope link table 1

ip route add default via 10.0.0.244 dev eth1 table 1


ip route add 10.0.1.0/24 dev eth2 scope link table 2

ip route add default via 10.0.1.244 dev eth2 table 2


#Enable interfaces for MPTCP

ip link set dev eth1 multipath on

ip link set dev eth2 multipath on
```



Εικόνα 13: *Τα interfaces των δυο virtual machines*

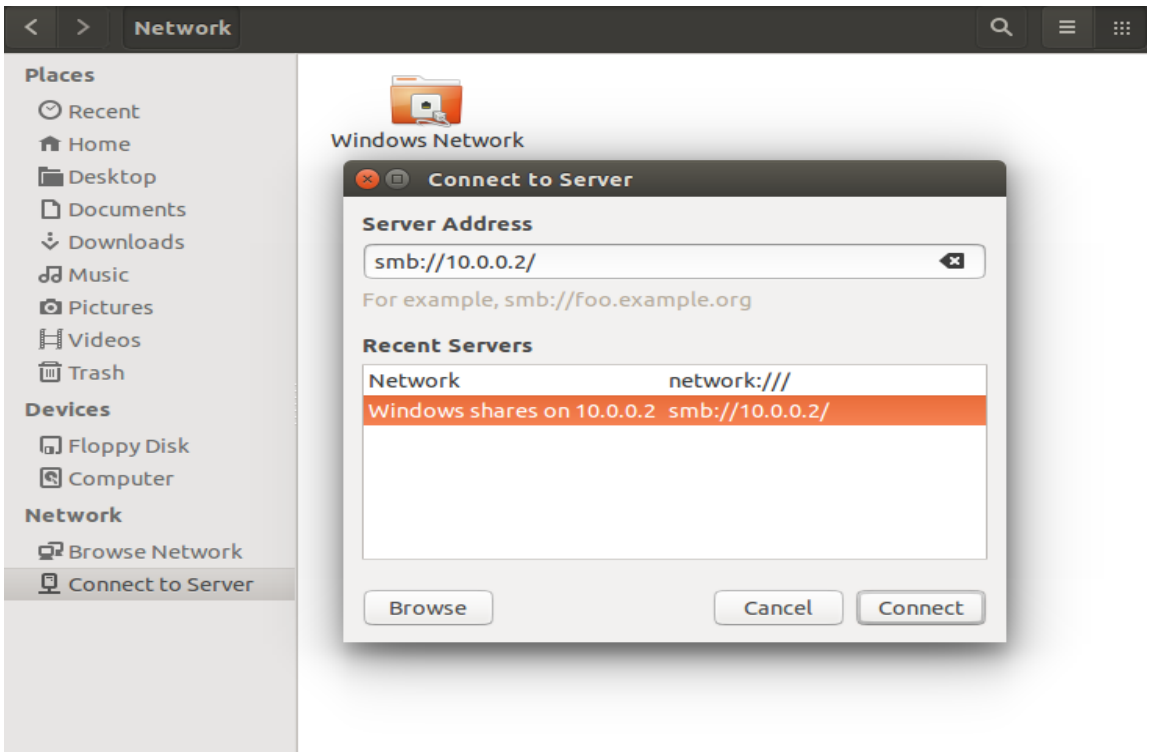
```
root@ubuntu:/home/vm1# sudo ip route
default via 192.168.223.2 dev eth0  proto static
10.0.0.0/24 dev eth1  proto kernel  scope link  src 10.0.0.1  metric 1
10.0.1.0/24 dev eth2  proto kernel  scope link  src 10.0.1.1  metric 1
192.168.223.0/24 dev eth0  proto kernel  scope link  src 192.168.223.130  metric 1
root@ubuntu:/home/vm1# sudo ip rule
0:      from all lookup local
32764:  from 10.0.1.1 lookup 2
32765:  from 10.0.0.1 lookup 1
32766:  from all lookup main
32767:  from all lookup default
root@ubuntu:/home/vm1# sudo ip route show table 1
default via 10.0.0.244 dev eth1
10.0.0.0/24 dev eth1  scope link
root@ubuntu:/home/vm1# sudo ip route show table 2
default via 10.0.1.244 dev eth2
10.0.1.0/24 dev eth2  scope link
root@ubuntu:/home/vm1#
```

Εικόνα 14: *Το routing table του vm1*

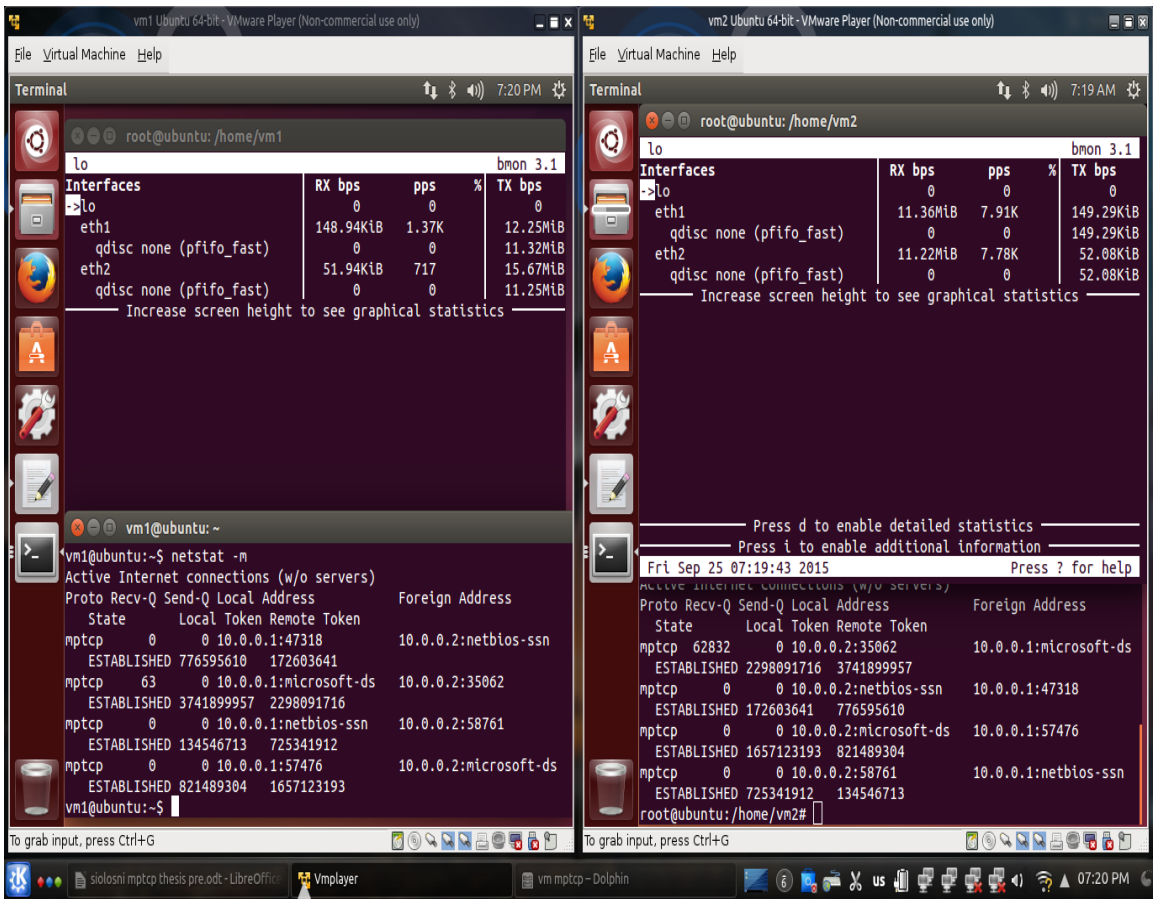
```
root@ubuntu:/home/vm2# sudo ip route
default via 192.168.223.2 dev eth0  proto static
10.0.0.0/24 dev eth1  proto kernel  scope link  src 10.0.0.2  metric 1
10.0.1.0/24 dev eth2  proto kernel  scope link  src 10.0.1.2  metric 1
192.168.223.0/24 dev eth0  proto kernel  scope link  src 192.168.223.131  metric 1
root@ubuntu:/home/vm2# sudo ip rule
0:      from all lookup local
32764:  from 10.0.1.2 lookup 2
32765:  from 10.0.0.2 lookup 1
32766:  from all lookup main
32767:  from all lookup default
root@ubuntu:/home/vm2# sudo ip route show table 1
default via 10.0.0.244 dev eth1
10.0.0.0/24 dev eth1  scope link
root@ubuntu:/home/vm2# sudo ip route show table 2
default via 10.0.1.244 dev eth2
10.0.1.0/24 dev eth2  scope link
root@ubuntu:/home/vm2#
```

Εικόνα 15: *Το routing table του vm2*

Η σύνδεση μέσω Samba γίνεται αφού ανοίξουμε το file folder



Εικόνα 16: *Σύνδεση με τον έτερο host.*



Εικόνα 17: *Ανταλλαγή δεδομένων μεταξύ δυο virtual machines με τη χρήση του mptcp*

Το αποτέλεσμα της ανταλλαγής δεδομένων με τη χρήση του mptcp έχουμε την δυνατότητα να το δούμε με το **bmon**, όπου ενεργοποιώντας ή απενεργοποιώντας interfaces παρακολουθούμε την κίνηση. Επίσης μπορούμε να δούμε και τις mptcp συνδέσεις που έχουν εγκατασταθεί.

6. Κεφάλαιο - Αξιολόγηση αλγορίθμων έλεγχου συμφόρησης

Όπως προαναφέραμε το MPTCP διανέμει το φορτίο των δεδομένων και δημιουργεί ξεχωριστά subflows. Πως θα γινόταν λοιπόν σε αυτή την περίπτωση η εφαρμογή του congestion control; Μια πρώτη/εύκολη απάντηση θα ήταν να τρέχαμε το υπάρχον congestion control του regular TCP σε κάθε subflow ξεχωριστά. Μια τέτοια λύση όμως δεν θα ήταν δίκαιη για τις υπόλοιπες συνδέσεις TCP, οι οποίες μοιράζονται το ίδιο bottleneck με το MPTCP.

Γι αυτό το λόγο υπάρχουν τρεις βασικές προϋποθέσεις για να υπάρξει ένας εφαρμόσιμος multipath congestion control αλγόριθμος :

1. **Βελτίωση της απόδοσης.** Ένα multipath flow θα πρέπει να αποδίδει τουλάχιστον τόσο καλά όσο μια ροή μονής διαδρομής στο καλύτερο από τα μονοπάτια που έχει διάθεσή της.
2. **Να μην προκαλεί προβλήματα.** Ένα multipath flow δεν θα πρέπει να έχει μεγαλύτερη χωρητικότητα από οποιαδήποτε από τις πηγές του, που μοιράζεται μεταξύ των διαφορετικών ροών του από ότι να ήταν ένα single flow και να χρησιμοποιούσε μια από αυτές τις διαδρομές.
3. **Ισορροπημένη συμφόρηση**. Ένα multipath flow θα πρέπει να οδηγεί όσο το δυνατόν περισσότερη κίνηση μακριά από διαδρομές με μεγάλη συμφόρηση.

Ο LIA (Coupled Congestion Control Algorithm (LIA) [RFC6356]) έχει εφαρμογή στο increase phase του congestion avoidance κάνοντας σαφή τον τρόπο με τον οποίο αυξάνεται το window, κατά την λήψη ενός ACK. Οι slow start, fast retransmit και fast recovery αλγόριθμοι όπως και το multiplicative decrease του congestion avoidance παραμένουν οι ίδιοι με το standard TCP [RFC5681]. Συνεπώς το συνολικό throughput ενός multipath flow εξαρτάται από τις τιμές του παράγοντα alpha και των loss rates, MSS και RTT του κάθε μονοπατιού και ο υπολογισμός τους γίνεται σύμφωνα με τους τύπους 1 και 2 της παραγράφου 3 του RFC6356.

- Για κάθε ληφθέν ACK στο subflow 1 , το cwnd_1 αυξάνει κατά :

$$\min\left(\frac{alpha * bytes\_acked * MSS\_i}{cwnd\_total}, \frac{bytes\_acked * MSS\_i}{cwnd\_i}\right) \quad (1)$$

- Το alpha υπολογίζεται από τον παρακάτω τύπο :

$$alpha = cwnd\_total * \frac{MAX\ (cwnd\_i/rtt\_i^2)}{(SUM\ (cwnd\_i/rtt\_i))^2} \quad (2)$$

Τέλος ο LIA καταφέρνει να καλύψει τις 2 πρώτες προϋποθέσεις που αναφέρθηκαν παραπάνω όμως δεν καλύπτει το resource pooling.

Η πρόταση του OLIA έρχεται να καλύψει το resource pooling. Εφαρμόζεται δηλαδή στο increase phase του congestion avoidance με μια επιπρόσθετη διαμόρφωση στον slow start algorithm, οπού ορίζεται το ssthresh να είναι 1 MSS όταν εγκαθίστανται multipath συνδέσεις, με σκοπό να αποφεύγεται η μετάδοση κίνησης μέσα σε ένα μονοπάτι που είναι σε συμφόρηση ενώ υπάρχουν διαθέσιμα εναλλακτικά μονοπάτια. Έτσι το συνολικό throughput ενός multipath flow εξαρτάται από τις τιμές του παράγοντα alpha και των cwnd του κάθε subflow, του RTT του κάθε subflow καθώς και από μετρήσεις για την ανάδειξη των καλύτερων μονοπατιών.

Ο υπολογισμός τους γίνεται συμφώνα με τον τύπο 1 της παραγράφου 3 του [Khalili].

- Για κάθε ACK στο path r, το w_r αυξάνει κατά :

$$\left(\frac{w\_r/rtt\_r^2}{(SUM\_\{p\ in\ all\_paths\}\ (w\_p/rtt\_p))^2} + \frac{alpha\_r}{w\_r}\right) \quad (1)$$

multiplied by MSS_r * bytes_acked.

Το άθροισμα στον παρονομαστή του πρώτου όρου είναι για όλα τα μονοπάτια p στο all_paths. Το w_p και rtt_p δηλώνουν το window size και το round trip time του path p.

Το alpha_r υπολογίζεται :

-Εάν το r βρίσκεται μέσα στα collected_paths, τότε

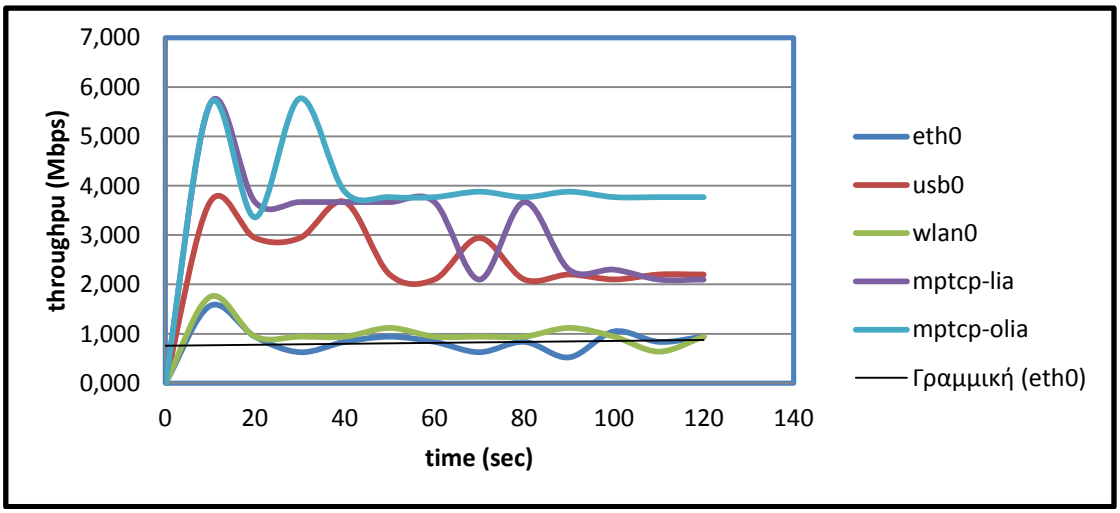
$$alpha\_r = \frac{1/number\_of\_paths}{|collected\_paths|}$$

- Εάν το r βρίσκεται μέσα στα max_w_paths και άν τα collected_paths δέν ειίναι άδεια, τότε

$$alpha\_r = \frac{1/number\_of\_paths}{|max\_w\_paths|}$$

- Αλλιώς, alpha_r=0.

Στη συνέχεια έγιναν μετρήσεις σε πραγματικό περιβάλλων με την βοήθεια των παραπάνω congestion control algorithms. Η τοπολογία που χρησιμοποιήθηκε περιλάμβανε έναν οικιακό router με ethernet και Wi-Fi σύνδεση στον υπολογιστή καθώς και ένα usb 3G stick. Στο σχήμα 14 φαίνονται τα αποτελέσματα των μετρήσεων που έγιναν με έναν multipath server με την βοήθεια του iperf.



Σχήμα 15:*Μετρήσεις throughput*

Στο παραπάνω γράφημα παρατηρήρουμε τη συμπεριφορά των interfaces που λειτουργούν ξεχωριστά το ένα από το άλλο καθώς και την αύξηση του throughput όταν χρησιμοποιείται το 3G stick. Ο LIA παρατηρείται πως έχει μεγαλύτερη αστάθεια καθ' οτι τα μεμονωμένα μονοπάτια λειτουργούν ανταγωνιστικά το ένα προς το άλλο σε αντίθεση με τον OLIA ο οποίος καταφέρνει να κάνει καλύτερο pool resourcing διότι ο τρόπος με τον οποίο διαχειρίζεται τα μονοπάτια έχει άμεση σχέση με τον αριθμό τους και τις μετρήσεις που γίνονται για κάθε ένα από αυτά, όπως έχει επισημανθεί. Στην περίπτωση όπου τα interfaces χρησιμοποιούν το ίδιο bottleneck τότε παρατηρούμε πως το throughput με MPTCP δεν είναι μεγαλύτερο από το μεγαλύτερο throughput του καλύτερου interface.

7. Κεφάλαιο - Συμπεράσματα

Το MPTCP είναι μια σημαντική επέκταση του TCP και μας δίνει την δυνατότητα να ανταλλάσουμε δεδομένα σε υπάρχοντα συστήματα μέσα από διαφορετικά interfaces και διαδρομές. Αυτό ως αποτέλεσμα μας δίνει την δυνατότητα να αυξήσουμε το διαθέσιμο bandwidth στην σύνδεση μας.

Μέσα από την παρούσα διπλωματική είχαμε την δυνατότητα να γνωρίσουμε τις βασικές αρχές και τον τρόπο λειτουργίας του MPTCP, να το εγκαταστήσουμε, να το ρυθμίσουμε και να κάνουμε επιτυχή χρήση του. Ο χρόνος που χρειάστηκε ήταν αρκετός, καθώς δεν ήταν μόνο το MPTCP που έπρεπε να γίνε κατανοητό αλλά και το ίδιο το OS (Linux) που χρησιμοποιήθηκε για να εγκατασταθεί (routing table , topology, congestion control, cpu, interface buffers, MSS, RTT).

Γενικά μπορούμε να πούμε πως το MPTCP δεν έχει ολοκληρωθεί ως πρωτόκολλο και η έρευνα και βελτιστοποίηση του συνεχίζεται σε πολλά επίπεδα. Το θέμα της ασφάλειας είναι ένα από τα πρώτα ζητήματα που συνεχίζονται να αναπτύσσονται. Το retransmission χρειάζεται περαιτέρω βελτιστοποίηση καθώς στηρίζεται ακόμα στον retransmission μηχανισμό του TCP. Επίσης είδαμε πως το MPTCP χρησιμοποιεί όλα τα interfaces που είναι ενεργά για ανταλλαγή δεδομένων κάτι που δεν δίνει απαραίτητα καλύτερη απόδοση. Ένας μηχανισμός επιλογής interface μέσω μετρήσεων του κάθε μονοπατιού (RTT) θα έδινε καλύτερα αποτελέσματα καθώς μονοπάτια με μεγάλες διαφορές RTT δεν δίνουν καλύτερα αποτελέσματα. Επιπλέον θα πρέπει να υπάρξει βελτιστοποίηση της αλληλεπιδράσεις του MPTCP με τα middle-boxes.

Η τελική εντύπωση που μένει είναι πως το MPTCP δεν χρειάζεται πλέον πολύπλοκα επαγγελματικά μηχανήματα για την παράλληλη χρήση πολλαπλών interface και ο στόχος είναι να επεκταθεί πέρα από τα data centers, στο σπίτι αλλά και στην κινητή τηλεφωνία.

Βιβλιογραφία

[BPDU] Experience with Multipath TCP draft-ietf-mptcp-experience-00

O. Bonaventure,C. Paasch,G. Detal,UCLouvain,September 16, 2014

[BPB] MultiPath TCP: From Theory to Practice

Sébastien Barré, Christoph Paasch, and Olivier Bonaventure. ICTEAM, Université catholique de Louvain B-1348 Louvain-la-Neuve, Belgium

[RPBF+] How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP

Costin Raiciu† , Christoph Paasch‡ , Sebastien Barre‡ , Alan Ford,

Michio Honda⋄ , Fabien Duchene‡ , Olivier Bonaventure‡ and Mark Handley⋆

Universitatea Politehnica Bucuresti, ‡ Universite Catholique de Louvain

Keio University, ⋆ University College London

[UvAS12] MultiPath TCP: Hands-On Gerrie Veerman

Universiteit van Amsterdam System & Network Engineering July 9, 2012

[BL] Implementation and Assessment of Modern Host-based Multipath Solutions - Sebastien Barre - ICTEAM Louvain School of Engineering

[PL] Improving Multipath TCP Christoph Paasch - ICTEAM Louvain School of Engineering

[KHALI] Opportunistic Linked-Increases Congestion Control Algorithm for MPTCP Ramin Khalili, Nicolas Gast, T-Labs/TU-Berlin, Miroslav Popovic, Le Boudec, EPFL-LCA2, February 17, 2013

[BPBL11] MultiPath TCP - Guidelines for implementers S. Barre, C. Paasch, O. Bonaventure, UCLouvain, Belgium March 7, 2011

[RFC 2581] IEEE RFC 2581 TCP Congestion Control M. Allman, V. Paxson, ICSI, E. Blanton, Purdue University

[RFC 6182] IEEE RFC 6182 Architectural Guidelines for Multipath TCP Development

[RFC 6356] IEEE RFC 6356 Coupled Congestion Control for Multipath Transport Protocols

[RFC 6824] IEEE RFC 6824 TCP Extensions for Multipath Operation with Multiple Addresses

[RFC 6897] IEEE RFC 6897 Multipath TCP (MPTCP) Application Interface Considerations

[RFC 5681] TCP Congestion Control M. Allman, V. Paxson, ICSI, E. Blanton, Purdue University, September 2009

[RFC 1323] TCP Extensions for High Performance. V. Jacobson, LBL, R.Braden, ISI, D. Borman, Cray Research, May 1992