ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Αξιολόγηση του δικτυακού λειτουργικού συστήματος DANOS σε ταχύτητες άνω των 20 Gbps

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΚΩΣΤΟΠΟΥΛΟΥ ΚΥΡΙΜΗ ΠΑΝΟΥ

Επιβλέπων : Ευστάθιος Συκάς
Καθηγητής Ε.Μ.Π

Αθήνα, Μάρτιος 2023

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Αξιολόγηση του δικτυακού λειτουργικού συστήματος DANOS σε ταχύτητες άνω των 20 Gbps

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΚΩΣΤΟΠΟΥΛΟΥ ΚΥΡΙΜΗ ΠΑΝΟΥ

Επιβλέπων : Ευστάθιος Συκάς
Καθηγητής Ε.Μ.Π

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 14η Μαρτίου 2023

.....	.....	.....
Ευστάθιος Συκάς Καθηγητής Ε.Μ.Π.	Νικόλαος Μήτρου Καθηγητής Ε.Μ.Π.	Ιωάννα Ρουσσάκη Αναπληρώτρια Καθηγήτρια Ε.Μ.Π.

Αθήνα, Μάρτιος 2023

…….

ΠΑΝΟΣ ΚΩΣΤΟΠΟΥΛΟΣ ΚΥΡΙΜΗΣ

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π

ΠΕΡΙΛΗΨΗ

Με την ταχεία ανάπτυξη του διαδικτύου στις μέρες μας, ο πυρήνας Linux αδυνατεί πια να επεξεργαστεί τις απαιτούμενες ταχύτητες από 20 έως και 100Gbps ροής πακέτων. Σύγχρονες τεχνολογίες που αξιοποιούν την πολυπυρηνική αρχιτεκτονική, για την εφαρμογή παραλληλίας, τεχνολογίες αξιοποίησης της κεντρικής μνήμης και των κρυφών μνημών καθώς και τεχνολογίες πολλαπλών ουρών στις κάρτες δικτύων, έδωσαν την δυνατότητα κατασκευής λειτουργικών συστημάτων για δικτυακή χρήση. Τα σύγχρονα υπολογιστικά συστήματα διαθέτουν μεγάλη υπολογιστική ισχύ, που δεν αξιοποιείται βέλτιστα από τον πυρήνα Kernel, και μπορούν να αξιοποιηθούν ως δρομολογητές για την επεξεργασία των πακέτων, μέσω δικτυακών λειτουργικών συστημάτων. Στα πλαίσια της διπλωματικής χρησιμοποιείται ένα τέτοιο λειτουργικό σύστημα, το DANOS[1], στο οποίο γίνεται παραμετροποίηση και δοκιμές δικτυακής κίνησης με σκοπό να μετρηθούν οι δυνατότητες του DANOS στην επεξεργασία τυπικών επιχειρησιακών ροών και στην χρησιμοποίηση διαφόρων πρωτοκόλλων. Το DANOS βασίζεται στο λειτουργικό Linux και αξιοποιεί την τεχνολογία του DPDK[2], μια τεχνολογία που παρακάμπτει τον πυρήνα των Linux για την επεξεργασία των πακέτων και την οδηγεί στον χώρο χρήστη. Ο τρόπος που γίνεται η παράκαμψη καθώς και οι τεχνολογίες που χρησιμοποιούνται για την βελτιστοποίηση της επεξεργασίας των πακέτων στον χώρο χρήστη, αναλύονται στο θεωρητικό μέρος της διπλωματικής. Για την ανάπτυξη ταχυτήτων της τάξεως των 100Gbps χρησιμοποιείται μία ενδιαφέρουσα και πρωτότυπη τεχνική ανακύκλωσης, που περιγράφεται στο μέρος της παραμετροποίησης του DANOS. Τέλος, γίνονται πειράματα παραγωγής κίνησης δικτυακής ροής την οποία το DANOS αναλαμβάνει να επεξεργαστεί, δοκιμάζονται τείχη προστασίας και μετάφραση CGNAT[3] για να αξιολογηθεί το DANOS σε πραγματικές συνθήκες δρομολογητή. Η κατασκευή της τοπολογίας για την κίνηση των πακέτων, στα πλαίσια της διπλωματικής ,αφορά τόσο την χρήση εικονικού περιβάλλοντος και εικονικών μηχανών όσο και την χρησιμοποίηση φυσικών μηχανημάτων server και switch τα οποία λειτουργούν στο κέντρο δικτύων του ΕΜΠ.

Λέξεις Κλειδιά : DANOS, DPDK, πολυπυρηνική αρχιτεκτονική, παράκαμψη πυρήνα Linux, τείχη προστασίας, CGNAT, τοπολογία, εικονικές μηχανές

---

[1] Disaggregated Network Operating System βλ. Linux Foundation, *danos project* [ιστοσελίδα], https://www.danosproject.org/, (τελευταία πρόσβαση 27 Ιανουάριος 2023)

[2] Data Plane Development Kit βλ. Linux Foundation, *dpdk* [ιστοσελίδα], https://www.dpdk.org/, (τελευταία πρόσβαση 27 Ιανουαρίου 2023)

[3] Carrier-grade NAT βλ. σχετικά S. Jiang, D. Guo και B. Carpenter, 'An Incremental Carrier-Grade NAT (CGN) for IPv6 Transition', *rfc editor* [ιστοσελίδα], 6264, Ιούνιος 2011, https://www.rfc-editor.org/rfc/rfc6264, (τελευταία πρόσβαση 28 Ιανουαρίου 2023)

ABSTRACT

Nowadays, with the fast development of the internet, the Linux kernel is incapable of processing the required speeds from 20 up to 100 Gbps of traffic flows. Modern technologies that make good use of multicore architecture, for implementing parallelism, of main and cache memories, but also of multiqueue in network cards, made the creation of networking operating systems possible. Modern processing systems, possess large amounts of processing power, power that the Linux kernel can not take advantage of, and can be transformed into network routers by networking operating systems. In the context of this diploma thesis, a similar operating system is used, DANOS[1], on which certain configurations are implemented and is tested with network flows in order to measure its capabilities at processing typical enterprise flows and usage of various protocols. DANOS is a Linux-based operating system that makes good use of DPDK[2] technology, a kernel bypass technology for processing internet packers that drives the processing into the user space. The way this bypassing is implemented and the technologies that are being used for the optimization of packet processing will be discussed in the theoretical part of this thesis. For the production of up to 100Gbps of network flow, a rather interesting and original recycling technique is used, that is discussed in the configuration part of the thesis. Finally, certain network flow are produced and DANOS is responsible for processing them, firewalls and CGNAT[3] are implemented in order for DANOS to be tested on real router conditions. The construction of the topology for packet flow, in the context of this thesis, comprises the usage of virtual environment, virtual machines and the usage of physical machines, server and switch, which are hosted in the network center of NTUA.

Keywords: DANOS, DPDK, multicore architecture, linux kernel bypass, firewalls, CGNAT, topology, virtual machines

---

[1] Disaggregated Network Operating System βλ. Linux Foundation, *danos project* [ιστοσελίδα], https://www.danosproject.org/, (τελευταία πρόσβαση 27 Ιανουαρίου 2023)

[2] Data Plane Development Kit βλ. Linux Foundation, *dpdk* [ιστοσελίδα], https://www.dpdk.org/, (τελευταία πρόσβαση 27 Ιανουαρίου 2023)

[3] Carrier-grade NAT βλ. σχετικά S. Jiang, D. Guo και B. Carpenter, 'An Incremental Carrier-Grade NAT (CGN) for IPv6 Transition', *rfc editor* [ιστοσελίδα], 6264, Ιούνιος 2011, https://www.rfc-editor.org/rfc/rfc6264, (τελευταία πρόσβαση 28 Ιανουαρίου 2023)

ΕΥΧΑΡΙΣΤΙΕΣ

Αρχικά, θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή κ. Ευστάθιο Συκά, για τα εξαιρετικά ενδιαφέροντα μαθήματα που διδάσκει στο πρόγραμμα σπουδών της σχολής, τα οποία με οδήγησαν να επιλέξω την συγκεκριμένη διπλωματική, για την εμπιστοσύνη που μου επέδειξε με την ανάθεσή της και για τα πολύτιμα σχόλια του κατά την ολοκλήρωσή της. Επιπλέον θα ήθελα να ευχαριστήσω τον κ. Δημήτρη Καλογερά για την συνεργασία μας στο θεωρητικό κομμάτι της εργασίας, για την σημαντική βοήθειά του στην κατανόηση των χρησιμοποιούμενων τεχνολογιών, στην επιμέλεια του κειμένου και στην οργάνωση της παρουσίασης. Τέλος, θα ήθελα να ευχαριστήσω θερμά τον κ. Σπύρο Παπαγεωργίου για την συνεργασία μας στο πρακτικό μέρος της διπλωματικής, την πολύτιμη βοήθειά του στην κατανόηση εργαλείων και τεχνολογιών και στην εύρεση πρωτότυπων λύσεων στα ζητήματα που προέκυψαν.

Περιεχόμενα

Πίνακας Σχημάτων

Πίνακας Πινάκων

Πίνακας Διαγραμμάτων

Κεφάλαιο 1 : Εισαγωγή

1.1 Περιγραφή του προβλήματος

Στις μέρες μας, με την αύξηση των χρηστών του διαδικτύου και των αναγκών τους για μεγάλες ταχύτητες, δημιουργείται η ανάγκη για υπολογιστικά συστήματα με δυνατότητα να επεξεργάζονται ταχύτητες από 20 έως και 100 Gbps. Η μαζική διαθεσιμότητα της πολυπυρηνικής αρχιτεκτονικής ακόμη και σε Η/Υ τελικού χρήστη, δημιουργεί επαρκείς προϋποθέσεις για την ικανοποίηση της παραπάνω ανάγκης. Εν τέλει, η ικανοποίηση της ανάγκης εξαρτάται από τους μηχανισμούς που προσφέρουν τα λειτουργικά συστήματα και κατ' επέκταση οι πυρήνες τους (kernel). Σε ταχύτητες έως 1Gbps ο Linux πυρήνας, μπορεί με διάφορες τεχνολογίες βελτιστοποίησης να ανταποκριθεί στις απαιτήσεις του δικτύου, όμως για μεγαλύτερες ταχύτητες ο ευέλικτος πυρήνας του Linux παραμένει στενωπός και δεν επαρκεί.

Μία σύγχρονη προσέγγιση στο παραπάνω πρόβλημα, είναι η παράκαμψη του πυρήνα του Linux και η μεταφορά της επεξεργασίας των πακέτων στον χώρο χρήστη. Ένα πολύ σημαντικό λογισμικό, το DPDK[4], δημιουργήθηκε με αυτήν τη φιλοσοφία. Αξιοποιώντας υπάρχουσες τεχνολογίες όπως η δυνατότητα παραλληλίας στην επεξεργασία ή πολλαπλές ουρές στην κάρτα δικτύου, το DPDK, μπορεί να αξιοποιήσει το υλικό για την επίτευξη ταχυτήτων άνω των 20 Gbps με ευκολία. Το DPDK έδωσε την δυνατότητα για κατασκευή λειτουργικών συστημάτων βασισμένων σε Linux για δικτυακή χρήση, όπως το DANOS. Το DANOS[5] δημιουργήθηκε ως ένα λειτουργικό σύστημα που χρησιμοποιεί το DPDK και παρέχει δυνατότητες δρομολογητή ενώ παράλληλα είναι βασισμένο σε Linux.

1.2 Στόχος και Περιεχόμενο της Διπλωματικής

Ο στόχος της διπλωματικής είναι να δοκιμαστεί το εξιδεικευμένο λειτουργικό σύστημα DANOS σε κίνηση υψηλών ταχυτήτων και να αξιολογηθούν οι επιδόσεις του. Καθώς παρέχει πολλές δυνατότητες δρομολογητή, θα δοκιμαστούν υπηρεσίες όπως τείχη προστασίας (Firewall)[6] και

---

[4] ο.π. Linux Foundation, *danos project* [ιστοσελίδα]
[5] ο.π. Linux Foundation, *dpdk* [ιστοσελίδα]
[6] K. Scarfone, P. Hoffman, *Guidelines on Firewalls and Firewall Policy*, National Institute of Standards and Technology, Σεπτέμβρης 2009

μετάφραση CGNAT[7] και θα εξεταστεί πώς/αν επηρεάζουν τις επιδόσεις του λειτουργικού στην επεξεργασία δικτυακής κίνησης. Στα πλαίσια των δοκιμών, δημιουργείται αρχικά μία τοπολογία, χρησιμοποιώντας έναν switch και server, η οποία αποτελείται από εικονικές μηχανές (φιλοξενούμενες στον server) και φυσικά μηχανήματα (switch). Καθώς δεν υπάρχει δυνατότητα παραγωγής φυσικής κίνησης πέρα των 20Gbps με το διαθέσιμο υλικό, χρησιμοποιείται μια πρωτότυπη τεχνική ανακύκλωσης που επιτυγχάνει τον πολλαπλασιασμό κίνησης. Στην συνέχεια, γίνεται παραμετροποίηση στα μηχανήματα για την καθοδήγηση της κίνησης στην τοπολογία και την συλλογή πληροφοριών. Επιπλέον, στο πειραματικό μέρος το DANOS δοκιμάζεται σε διαφορετικές ροές πακέτων σε υψηλές ταχύτητες, σε διαφορετικά στάδια, αρχικά χωρίς κάποια επιπλέον παραμετροποίηση, έπειτα σε τείχη προστασίας και τέλος στο CGNAT. Για τις μετρήσεις παρατίθενται αποτελέσματα και αναλύονται οι περιορισμοί και οι επιδόσεις.

1.3 Οργάνωση της Διπλωματικής

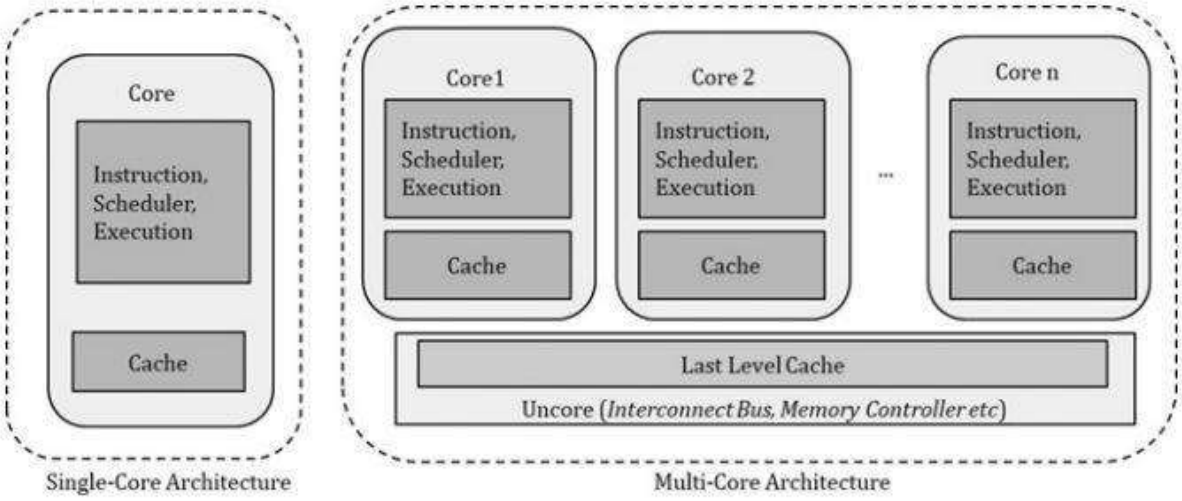
- Κεφάλαιο 2: Παρουσιάζονται βασικές σύγχρονες τεχνολογίες που αποτελούν θεμέλια για την δημιουργία λογισμικών όπως το DPDK και κατ' επέκταση το DANOS.
- Κεφάλαιο 3: Περιγράφεται η λειτουργία του DPDK, οι τεχνολογίες που αξιοποιεί και οι πρωτότυπες ιδέες και υλοποιήσεις που το ξεχωρίζουν.
- Κεφάλαιο 4: Αναλύεται η αρχιτεκτονική του λειτουργικού συστήματος DANOS.
- Κεφάλαιο 5: Περιγράφεται το υλικό που χρησιμοποιήθηκε στα πλαίσια της διπλωματικής, η κατασκευή των εικονικών μηχανών, η φυσική διασύνδεση των μηχανημάτων και η βασική παραμετροποίηση για την λογική λειτουργία της τοπολογίας.
- Κεφάλαιο 6: Περιγράφεται η προετοιμασία των δοκιμών και τα εργαλεία που χρησιμοποιούνται. Αφού γίνουν οι μετρήσεις, παρατίθενται τα αποτελέσματα και τα συμπεράσματα που προκύπτουν από αυτά. Έπειτα δοκιμάζεται το DANOS στις ίδιες δοκιμές, έχοντας παράλληλα υλοποιήσει διαφορετικά τείχη προστασίας στις διεπαφές του.
- Κεφάλαιο 7: Περιγράφεται η παραμετροποίηση του CGNAT στο DANOS, παρατίθενται αποτελέσματα και αναλύονται.
- Κεφάλαιο 8: Γίνεται σύνοψη του περιεχομένου της διπλωματικής και των αποτελεσμάτων και κατατίθενται προτάσεις για περεταίρω μελέτη.

[7] ο.π. S. Jiang, D. Guo και B. Carpenter

Κεφάλαιο 2 : Τεχνολογικό Υπόβαθρο

2.1 Πολυπυρηνικό σύστημα

Για την αύξηση των επιδόσεων του επεξεργαστή υπάρχουν δύο τεχνικές: αύξηση της συχνότητας (κάθετη επέκταση) και αύξηση των πυρήνων (οριζόντια επέκταση). Καθώς η αύξηση της συχνότητας των πυρήνων άρχισε να τείνει προς κορεσμό, πολλές σύγχρονες αρχιτεκτονικές και τεχνολογίες υιοθετούν και χρησιμοποιούν το πολυπυρηνικό μοντέλο. Η στροφή αυτή, προϋποθέτει αλλαγή της φιλοσοφίας του λογισμικού για να μπορεί να αξιοποιήσει τις δυνατότητες του πολυπυρηνικού συστήματος. Το λογισμικό, για να βελτιώσει τις επιδόσεις του, πρέπει να εκμεταλλευτεί την δυνατότητα παραλληλίας, δηλαδή να μπορεί να χωριστεί σε διαφορετικές διεργασίες, κάθε μία ανεξάρτητη, που να μπορεί να επεξεργάζεται από διαφορετικό πυρήνα. Στο παρακάτω σχήμα φαίνεται η δομή ενός πολυπηρνικού συστήματος όπου οι πυρήνες μοιράζονται το τελευταίο επίπεδο cache (LLC). Τα υπόλοιπα επίπεδα cache αφορούν τοπικά τον κάθε πυρήνα (βλ. εικόνα 1).

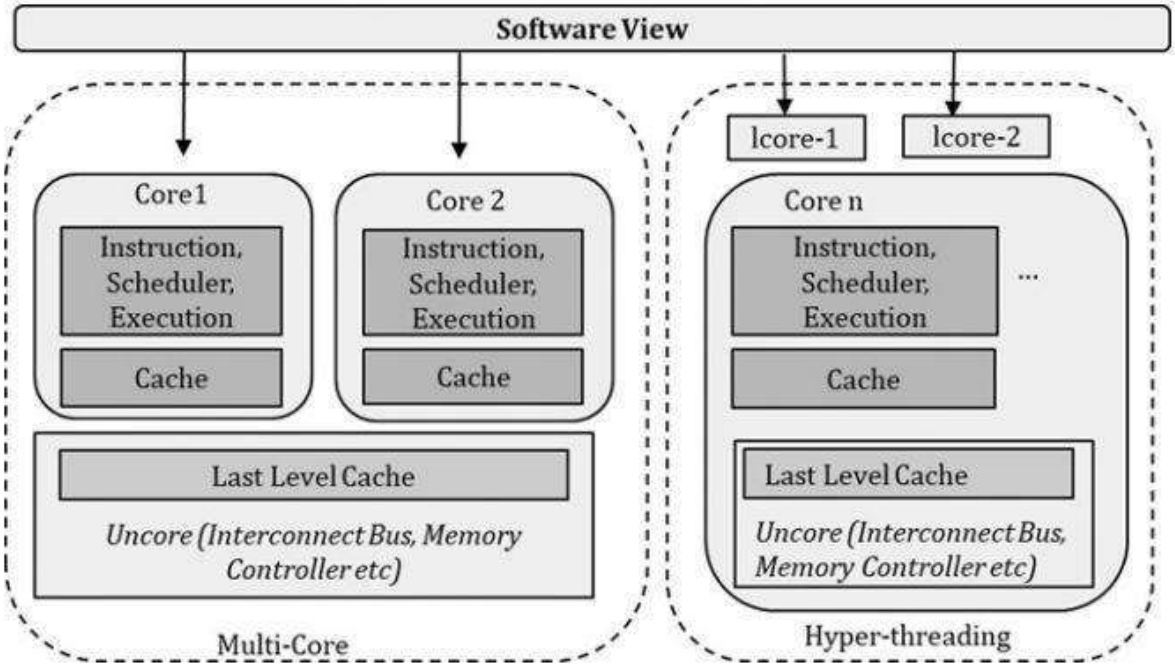


Εικόνα 1: Μονοπύρηνη και Πολυπύρηνη αρχιτεκτονική

2.1.1 Hyperthreading

Για να αξιοποιηθεί ο idle χρόνος ενός πυρήνα, δηλαδή ο χρόνος που ο πυρήνας δεν αξιοποιείται για την επεξεργασία δεδομένων, υπάρχει η τεχνολογία Hyperthreading (HT). Για κάθε φυσικό

πυρήνα το λογισμικό βλέπει δύο λογικούς πυρήνες και με αυτόν τον τρόπο καταφέρνει να χρησιμοποιήσει τον κενό χρόνο για να επεξεργαστεί δεδομένα. Το HT βελτιώνει τις επιδόσεις, όταν ο cpu βρίσκεται σε idle κατάσταση για αρκετό χρονικό διάστημα. Αν η εργασία που εκτελεί έχει υψηλές απαιτήσεις σε επεξεργαστική ισχύ τότε δεν επιφέρει βελτίωση στις επιδόσεις. Στην εικόνα 2 φαίνεται η οπτική του λογισμικού σε ένα πολυπυρηνικό σύστημα με και χωρίς HT.



Εικόνα 2: Πολυπυρηνική αρχιτεκτονική και Hyper-threading

2.1.2 NUMA

Για την αύξηση των επιδόσεων, πέρα από την αύξηση συχνότητας των πυρήνων ή την αύξηση του πλήθους των πυρήνων σε έναν επεξεργαστή, υπάρχει και η λύση των πολλαπλών επεξεργαστών(chips) στην ίδια μητρική πλακέτα. Μια μητρική πλακέτα με πολλαπλούς επεξεργαστές στην σιλικόνη μπορεί να πετύχει επιδόσεις όσο το άθροισμα των επιδόσεων των επεξεργαστών που χρησιμοποιεί. Κάθε socket διαθέτει την δική του μνήμη και τους δικούς του πυρήνες. Το πρόβλημα που δημιουργείται για το λογισμικό είναι ότι σε περιβάλλον με πολλούς επεξεργαστές τίθενται θέματα τοπικής και απομακρυσμένης μνήμης και πυρήνων, καθώς το κάθε socket διαθέτει το δικό του υλικό. Τυχόν πρόσβαση σε απομακρυσμένη μνήμη θα προκαλέσει καθυστέρηση πρόσβασης και μείωση στις επιδόσεις. Με αυτήν την λογική δημιουργήθηκε ο σχεδιασμός NUMA για συστήματα με πολλαπλούς πυρήνες και βασίζεται στις εξής δύο αρχές:

- Τοπικός πυρήνας και τοπική μνήμη: Κάθε πυρήνας χρησιμοποιεί μόνος του την τοπική μνήμη για δεδομένα που χρησιμοποιούνται συχνά, αποφεύγοντας το πολυπυρηνικό σύστημα να ανταγωνίζεται τα δεδομένα, όσο αυτό είναι δυνατόν
- Τοπική συσκευή με τοπικό πυρήνα: Η συσκευή είναι φυσικά συνδεδεμένη με θήρα PCIe, η οποία υπόκειται στο τοπικό socket με πολλούς πυρήνες. Η αρχή είναι να χρησιμοποιεί την τοπική μνήμη των τοπικών πυρήνων για να επεξεργαστεί το πακέτο. Οι σχεδιασμένες δομές δεδομένων και οι buffers πακέτων διατίθενται από την τοπική μνήμη.

2.2 Εικονοποίηση

Οι επεξεργαστικές δυνατότητες των CPU, ξεπέρασαν κατά πολύ τις ανάγκες των λειτουργικών συστημάτων για πόρους. Η ανάπτυξη των πολυπύρηνων αρχιτεκτονικών έδωσε την δυνατότητα ακόμα και σε προσωπικούς υπολογιστές να μπορούν να διαχειρίζονται με ευκολία πολλαπλά λειτουργικά συστήματα. Έτσι, δημιουργήθηκαν οι εικονικές μηχανές, για να μπορέσουν να αξιοποιήσουν τις δυνατότητες για την ταυτόχρονη συνύπαρξη πολλών λειτουργικών συστημάτων, φιλοξενούμενων στο ίδιο υλικό.

2.2.1 Εικονικές μηχανές

Οι εικονικές μηχανές είναι εξαιρετικά χρήσιμες, κάτι που φαίνεται και στην παρούσα διπλωματική, καθώς διατίθενται μηχανήματα με πολύ μεγάλη υπολογιστική δύναμη, ενώ τα λειτουργικά συστήματα δεν χρειάζονται συγκριτικά τόσους πόρους. Έτσι, υπάρχει η δυνατότητα να τρέχουν παράλληλα διαφορετικά λειτουργικά, να δημιουργούνται τοπολογίες, όπως στην συγκεκριμένη διπλωματική, εύκολα και γρήγορα. Είναι εφικτή η δοκιμή νέων λειτουργικών συστημάτων, εφαρμογών χωρίς τον κίνδυνο να επηρεαστούν πληροφορίες άλλων λειτουργικών. Κάθε εικονική μηχανή τρέχει το δικό της λειτουργικό σύστημα και λειτουργίες ξεχωριστά από κάθε άλλη εικονική μηχανή, ακόμα και αν όλες τρέχουν ταυτόχρονα στον ίδιο φυσικό υπολογιστή. Η εικονική μηχανή τρέχει ως διαδικασία σε ένα παράθυρο εφαρμογής, παρόμοια με άλλες εφαρμογές, πάνω στο λειτουργικό σύστημα της φυσικής μηχανής. Οι εικονικές μηχανές αποτελούν ένα αρχείο υπολογιστή, συχνά ονομαζόμενο εικόνα (image) και συμπεριφέρονται σαν ένας κανονικός υπολογιστής. Παρέχουν μεγαλύτερη ασφάλεια, καθώς λειτουργούν απομονωμένα

από το μηχάνημα που τα φιλοξενεί. Το κόστος μειώνεται με την χρήση εικονικών μηχανών, καθώς δεν χρειάζονται ξεχωριστά φυσικά μηχανήματα για την εκτέλεση δοκιμών, εφαρμογών κλπ.[8]

Ένα Virtual Machine είναι ένας υπολογιστικός πόρος που χρησιμοποιεί λογισμικό αντί για φυσικό υπολογιστή για να τρέξει προγράμματα και να αναπτύξει εφαρμογές. Μία ή περισσότερες "φιλοξενούμενες" εικονικές μηχανές τρέχουν πάνω σε μία μηχανή "οικοδεσπότη", χρησιμοποιώντας το δικό τους ξεχωριστό, λειτουργικό σύστημα.[9]

2.2.2 Hypervisor

Με την επέκταση χρήσης των εικονικών μηχανών, υπήρξε η ανάγκη για έναν μηχανισμό διαχείρισής τους. Η επικοινωνία των εικονικών μηχανών με το υλικό, ο διαμοιρασμός των φυσικών πόρων, η ακεραιότητα των δεδομένων κάθε εικονικής μηχανής είναι απαραίτητα για την συνύπαρξη πολλών εικονικών μηχανών. Προς αυτόν τον σκοπό οι ειδικές εφαρμογές τύπου hypervisor αναλαμβάνουν τον συντονισμό και την λειτουργία των εικονικών μηχανών.

Ο hypervisor είναι ένα λογισμικό διαχείρισης που δημιουργεί, τροποποιεί, καταστρέφει και τρέχει εικονικές μηχανές. Επιτρέπει την δημιουργία πολλών εικονικών μηχανών και τον διαμοιρασμό των πόρων του μηχανήματος, όπως μνήμη και υπολογιστή ισχύ. Υπάρχουν δύο τύποι hypervisor, τύπου 1 και τύπου δύο, με την διαφορά ότι ο πρώτος λειτουργεί ως ένα ελαφρύ λειτουργικό σύστημα και τρέχει απευθείας πάνω στο φιλοξενούν μηχάνημα και ο δεύτερος σαν εφαρμογή σε κάποιο λειτουργικό. Ο πιο διαδεδομένος τρόπος είναι ο τύπος 1, καθώς αυτοί οι hypervisor δεν χρειάζονται λειτουργικό σύστημα για να τρέξουν και είναι πολύ ασφαλείς, αφού δεν επηρεάζονται από επιθέσεις σε λειτουργικά συστήματα. Φυσικά αποτελούν και την πιο γρήγορη λύση, καθώς το υλικό επικοινωνεί με τις εικονικές μηχανές χωρίς την διαμεσολάβηση λειτουργικού συστήματος. Οι τύπου 2 χρησιμοποιούνται κυρίως από τελικούς χρήστες ή για την δοκιμή εφαρμογών στις οποίες δεν είναι τόσο σημαντική η ασφάλεια και η ταχύτητα.[10]

---

[8] Microsoft Azure, 'What is a virtual machine(VM)', azure microsoft [ιστοσελίδα], https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-a-virtual-machine/#overview, (τελευταία πρόσβαση 28 Ιανουαρίου 2023)

[9] VMware, 'What is a virtual machine', vmware [ιστοσελίδα], https://www.vmware.com/topics/glossary/content/virtual-machine.html, (τελευταία πρόσβαση 28 Ιανουαρίου 2023)

[10] VMware, 'What is a hypervisor', *VMware* [ιστοσελίδα], https://www.vmware.com/topics/glossary/content/hypervisor.html#:~:text=A%20hypervisor%2C%20also%20known%20as,such%20as%20memory%20and%20processing, (τελευταία πρόσβαση 29 Ιανουαρίου 2023)

2.2.3 KVM

Στα πλαίσια αυτής της διπλωματικής διατίθεται ένας server με μεγάλη επεξεργαστική δύναμη, με στόχο την φιλοξενία μιας εικονικής μηχανής με λειτουργικό danos. Για αυτό το σκοπό είναι βέλτιστο να χρησιμοποιηθεί ένας hypervisor τύπου 1 καθώς δεν υφίσταται μεσάζοντας και μπορεί να συνδέσει τις εικονικές μηχανές απευθείας με το υλικό, γεγονός που τον καθιστά ιδανικό για έλεγχο επιδόσεων σε μεγάλες ταχύτητες. Προς αυτόν τον σκοπό επιλέχθηκε η πλατφόρμα proxmox που ενσωματώνει KVM hypervisor (τύπου 1 hypervisor) και δίνει την δυνατότητα για εύκολη και γρήγορη δημιουργία εικονικών μηχανών. Μάλιστα υποστηρίζει PCI-passthrough που δίνει την δυνατότητα, συγκεκριμένες πόρτες NIC, να δοθούν εξολοκλήρου σε μια εικονική μηχανή. Το παραπάνω είναι πολύ σημαντικό, καθώς το danos, για να δοκιμαστεί σε πολύ μεγάλες ροές πακέτων ,πρέπει να χρησιμοποιεί απευθείας το υλικό.

Ο KVM hypervisor (kernel-based virtual machine) είναι μια λύση πλήρους εικονοποίησης για περιβάλλον Linux. Στην KVM αρχιτεκτονική κάθε εικονική μηχανή υλοποιείται σαν μία συνήθης διαδικασία των Linux.[11] Κάθε μία από αυτές τις εικονικές μηχανές έχει ιδιωτικό, εικονοποιημένο υλικό, το οποίο περιλαμβάνει εικονικό BIOS(βασικό σύστημα εισόδου/εξόδου), επεξεργαστή, μνήμη, αποθηκευτικό χώρο, adapter γραφικών και κάρτα δικτύου. Αυτό επιτρέπει στο KVM να ωφελείται από όλα τα χαρακτηριστικά του Linux πυρήνα. Ο KVM hypervisor εγκαθίσταται εγγενώς σε όλες τις διανομές του Linux, και μετασχηματίζει φυσικούς servers σε hypervisors, έτσι ώστε να μπορούν να φιλοξενούν πολλαπλές, απομονωμένες εικονικές μηχανές. Το KVM μπορεί να χρησιμοποιηθεί μόνο σε επεξεργαστές με επεκτάσεις εικονοποίησης υλικού όπως Intel-VT ή AMD-V.[12]

Ο KVM hypervisor είναι hypervisor τύπου 1 και είναι ο hypervisor που χρησιμοποιεί το proxmox για να φιλοξενεί τις εικονικές μηχανές. Με αυτόν τον τρόπο υπάρχει απευθείας επαφή του λειτουργικού των VM με το hardware, και στα πλαίσια της διπλωματικής, του DANOS με το υλικό του server.

---

[11] IBM, 'Kernel-based Virtual Machine', *ibm* [ιστοσελίδα], https://www.ibm.com/docs/en/license-metric-tool?topic=connections-kernel-based-virtual-machine, (τελευταία πρόσβαση 29 Ιανουαρίου 2023)

[12] Ubuntu, 'KVM hypervisor: a beginners' guide', Ubuntu blog [ιστολόγιο], 8 Σεπτεμβρίου 2021, https://ubuntu.com/blog/kvm-hyphervisor, (τελευταία πρόσβαση 29 Ιανουαρίου 2023)

2.2.4 Proxmox

Το Proxmox χρησιμοποιείται στα πλαίσια της διπλωματικής αυτής, γιατί παρέχει γρήγορη και εύκολη δημιουργία εικονικών μηχανών, καθώς και ασφαλή backup σε περίπτωση σφάλματος. Καθώς ενσωματώνει KVM hypervisor, είναι ιδανικός για δοκιμές μεγάλων ταχυτήτων στις οποίες είναι επιθυμητό να δοκιμαστεί το λειτουργικό DANOS. Ακόμα, παρέχει την δυνατότητα για PCI passthrough, μια τεχνική παραχώρησης μιας NIC αποκλειστικά σε κάποιο VM. Επειδή το danos δοκιμάζεται σε πολύ υψηλές ταχύτητες, το παραπάνω χαρακτηριστικό είναι απαραίτητο για να έχει το danos απευθείας επαφή με το υλικό χωρίς μεσάζοντα.

Το proxmox είναι μια ολοκληρωμένη, ανοιχτού λογισμικού διαχειριστική πλατφόρμα για δημιουργία, μεταφορά, καταστροφή εικονικών μηχανών. Ενσωματώνει KVM hypervisor και Linux Containers(LXC), αποθήκευση ορισμένη από λογισμικό(software defined) και δικτυακή λειτουργικότητα. Με την ενσωματωμένη, βασισμένη στο διαδίκτυο, διεπαφή χρήστη δίνει την δυνατότητα διαχείρισης VMs και containers, υψηλή διαθεσιμότητα για clusters ή ενσωματωμένο εργαλείο ανάκτησης σε περίπτωση σφάλματος.[13]

2.3 Μνήμη

Αναλόγως με την βελτίωση των επεξεργαστών ,η μνήμη εξελίχθηκε με τα χρόνια, παρέχοντας όλο και καλύτερους χρόνους πρόσβασης και μεγαλύτερο αποθηκευτικό χώρο. Μνήμες με μεγάλη χωρητικότητα υστερούν λόγω εσωτερικής τεχνολογίας και απόστασης τους από τον επεξεργαστή, ενώ μικρότερες και ταχύτερες μνήμες τοποθετούνται κοντά στον επεξεργαστή. Συνήθως η ιεραρχία της μνήμης σε ένα υπολογιστικό σύστημα αποτελείται από εξωτερικές συσκευές αποθήκευσης, κεντρική μνήμη και μνήμες cache. Στην περίπτωση δικτυακής κίνησης, η NIC κάνει DMA[14], το οποίο τοποθετεί τα πακέτα απευθείας στην μνήμη ή cache(DDIO[15]).

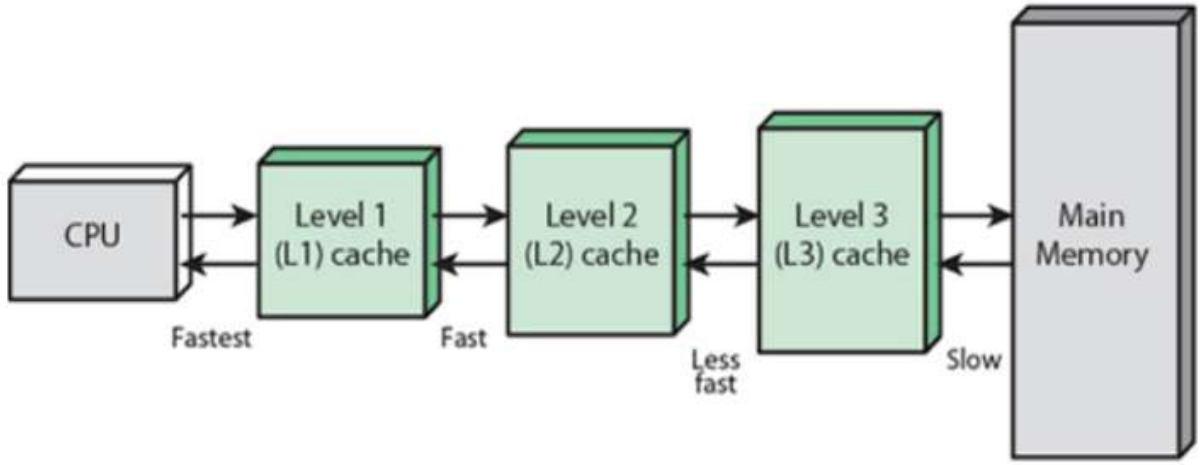
---

[13] Proxmox, 'Proxmox Virtual Environment', *proxmox* [ιστοσελίδα], https://www.proxmox.com/en/proxmox-ve, (τελευταία πρόσβαση 29 Ιανουαρίου 2023)

[14] J. Corbet, A. Rubini and G. Kroah-Hartman, 'Memory Mapping and DMA' in *Linux Device Drivers*, 3rd edition, 21 January 2005.

[15] Intel, 'Data Direct I/O Technology', *intel* [ιστοσελίδα], https://www.intel.com/content/www/us/en/io/data-direct-i-o-technology.html#:~:text=Intel%20DDIO%20makes%20the%20processor,and%20out%20of%20the%20processor, (τελευταία πρόσβαση 29 Ιανουαρίου 2023)

2.3.1 Cache



Εικόνα 3: Κεντρική μνήμη και cache

Ο επεξεργαστής χρειάζεται να περιμένει εκατοντάδες κύκλους, για να καταφέρει να διαβάσει δεδομένα από την μνήμη (συνήθως DRAM). Η αναμονή της διαθεσιμότητας των δεδομένων ονομάζεται καθυστέρηση πρόσβασης. Για να μειωθεί η καθυστέρηση πρόσβασης, τα σύγχρονα υπολογιστικά συστήματα χρησιμοποιούν τις μνήμες cache.

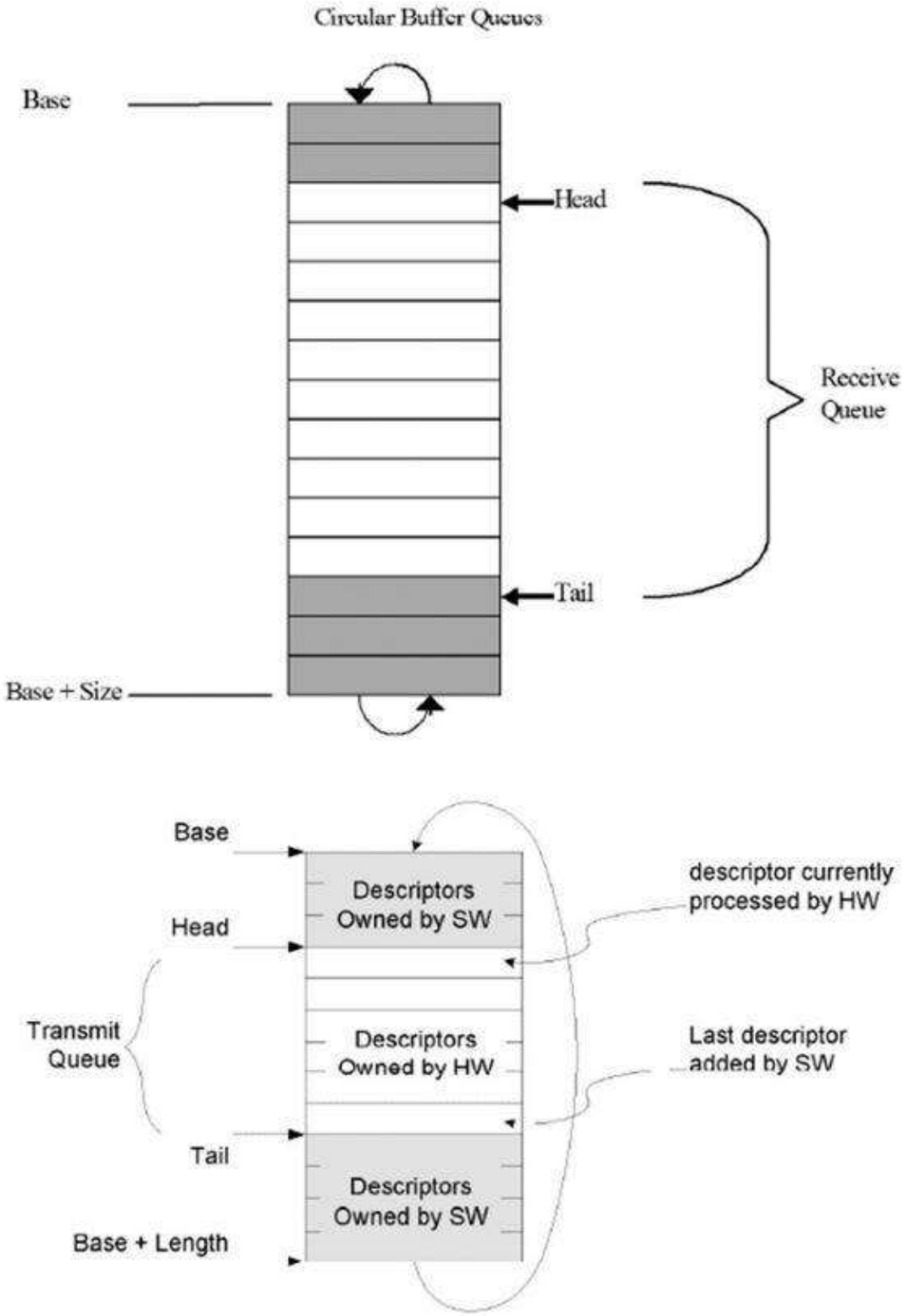
Οι μνήμες cache (συνήθως SRAM), είναι μνήμες μικρής χωρητικότητας με γρήγορη πρόσβαση και βρίσκονται πιο κοντά στον επεξεργαστή. Δομούνται σε επίπεδα, συνήθως L1,L2,L3 με την L1 να βρίσκεται πιο κοντά στον επεξεργαστή και την L3 πιο μακριά από αυτόν. Σε κάθε μνήμη cache τοποθετούνται “blocks” ,κομμάτια μνήμης από το μεγαλύτερο επίπεδο μνήμης, ανάλογα με την συχνότητα χρησιμοποίησής τους. Συχνά χρησιμοποιούμενα δεδομένα κατευθύνονται σε μικρότερα επίπεδα cache και με αυτόν τον τρόπο ο επεξεργαστής έχει την δυνατότητα να τα επεξεργαστεί γρηγορότερα. Σε συνδυασμό με τους πολλούς πυρήνες, τα χαμηλά επίπεδα cache αφορούν κάθε πυρήνα ξεχωριστά (συνήθως L1,L2), με τα μεγαλύτερα επίπεδα (συνήθως L3) να χρησιμοποιούνται από όλους τους πυρήνες (βλ. εικόνα 3).

2.4 NIC

Παράλληλα με την αύξηση των πυρήνων στους επεξεργαστές, με την επέκταση του διαδικτύου, τα υπολογιστικά συστήματα έρχονται αντιμέτωπα με ροές πακέτων 1,10 έως και 100 Gbps. Για την επεξεργασία ροής πακέτων τέτοιων τάξεων μεγέθους, χρειάζεται να αξιοποιηθεί το πολυπυρηνικό σύστημα. Προτού αναδειχθεί η αξιοποίηση των πολλών πυρήνων από την NIC, είναι σημαντικό να γίνει αναφορά στον τρόπο επικοινωνίας της NIC με τον επεξεργαστή.

2.4.1 NIC descriptor

Η NIC αλληλεπιδρά με τον επεξεργαστή μέσω μιας ουράς, βασισμένης σε τοπολογία δαχτυλιδιού (ring), η οποία αποτελείται από πολλούς προσδιοριστές πακέτων, με το δαχτυλίδι να αποτελεί έναν κυκλικό buffer στην μνήμη συστήματος(βλ. εικόνα 4). Συνήθως, κάθε πακέτο έχει τον δικό του buffer Μνήμης, και έναν συσχετισμένο προσδιοριστή buffer πακέτου. Αυτό σημαίνει ότι ο κάθε προσδιοριστής αφορά στην buffer διεύθυνση της μνήμης του πακέτου. Η ουρά έχει βασικούς καταχωρητές ελέγχου, όπως τους δείκτες: Βάση, Μέγεθος, Κεφαλή και Ουρά. Ο επεξεργαστής μπορεί να αναθέσει συνεχόμενη φυσική μνήμη, και η αρχική διεύθυνση ανατίθεται στον καταχωρητή Βάση. Ο καταχωρητής μεγέθους μετρά τον αριθμό όλων των προσδιοριστών. Ο καταχωρητής Κεφαλή συνήθως μπορεί να γραφεί μόνο από την NIC, ενώ μπορεί να διαβαστεί από το λογισμικό, καθώς αναφέρεται στον προσδιοριστή που βρίσκεται σε χρήση εκείνη την χρονική στιγμή. Ο καταχωρητής ουράς ανήκει στον driver λογισμικού.



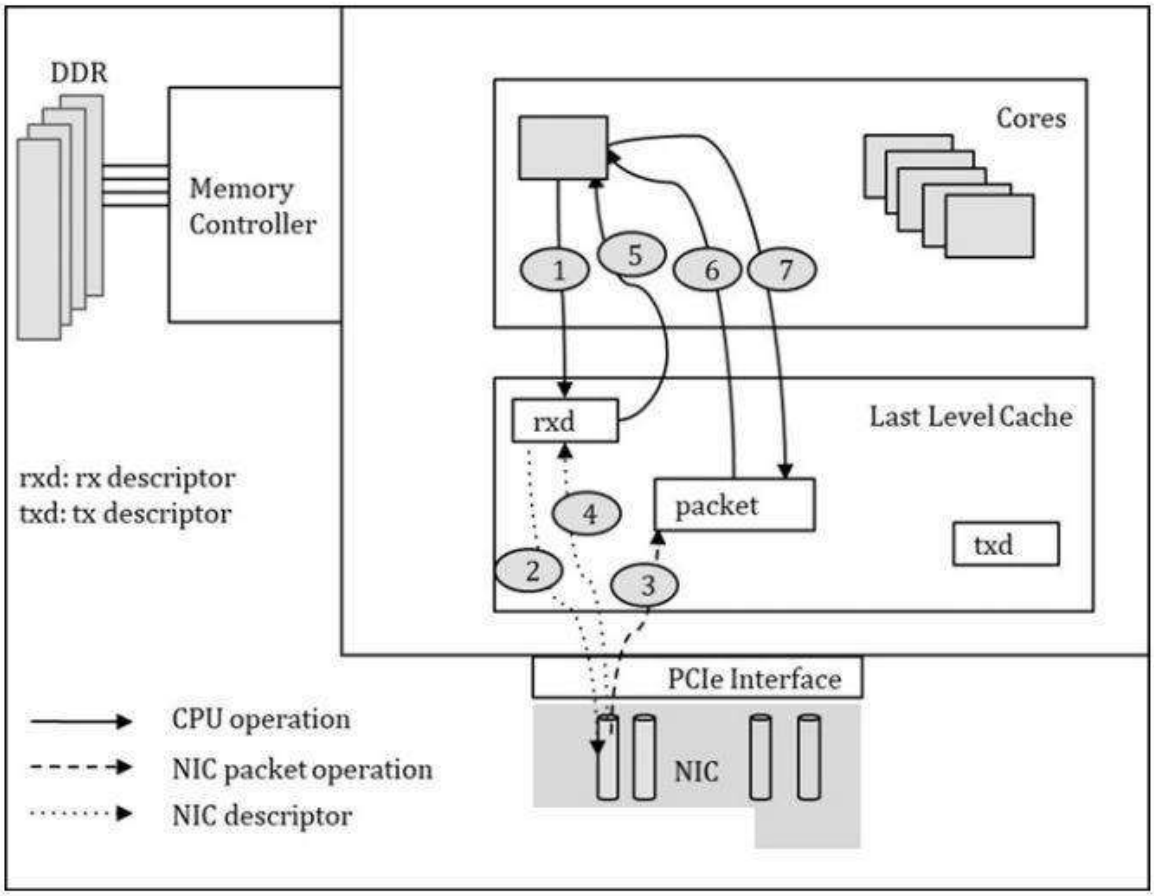
Εικόνα 4: Δακτύλιος Buffer πακέτων στην μνήμη

2.4.2 DDIO

Το DDIO είναι μια τεχνολογία που επιτρέπει στο NIC και στον επεξεργαστή να ανταλλάσσουν δεδομένα απευθείας από και προς την L3 cache. Παραδοσιακά, όταν ο επεξεργαστής χρειαζόταν να επεξεργαστεί ένα πακέτο, έπρεπε αυτό να φορτωθεί από την κύρια μνήμη στην μνήμη cache, κάτι το οποίο καταναλώνει πολλούς πόρους. Σε βελτιστοποιημένα συστήματα μειώνει την άφιξη και αποστολή των Ethernet πακέτων κατά 100 κύκλους. Το DDIO μεταφέρει πακέτα στην LLC μόνο εφόσον υπάρχει διαθέσιμος χώρος, αλλιώς τα τοποθετεί στην κύρια μνήμη.

2.4.3 Διαδικασία Rx/Tx

Στις εικόνες 5 και 6 φαίνεται πώς ένα σύστημα χρησιμοποιεί το DDIO για να μεταφέρει τα πακέτα προς και από το τελευταίο επίπεδο των cache.



Εικόνα 5: Διαδικασία Rx και DDIO

1. Ο επεξεργαστής τοποθετεί την διεύθυνση buffer του πακέτου στον RX προσδιοριστή
2. Η NIC διαβάζει τον RX προσδιοριστή για να πάρει την διεύθυνση μνήμης

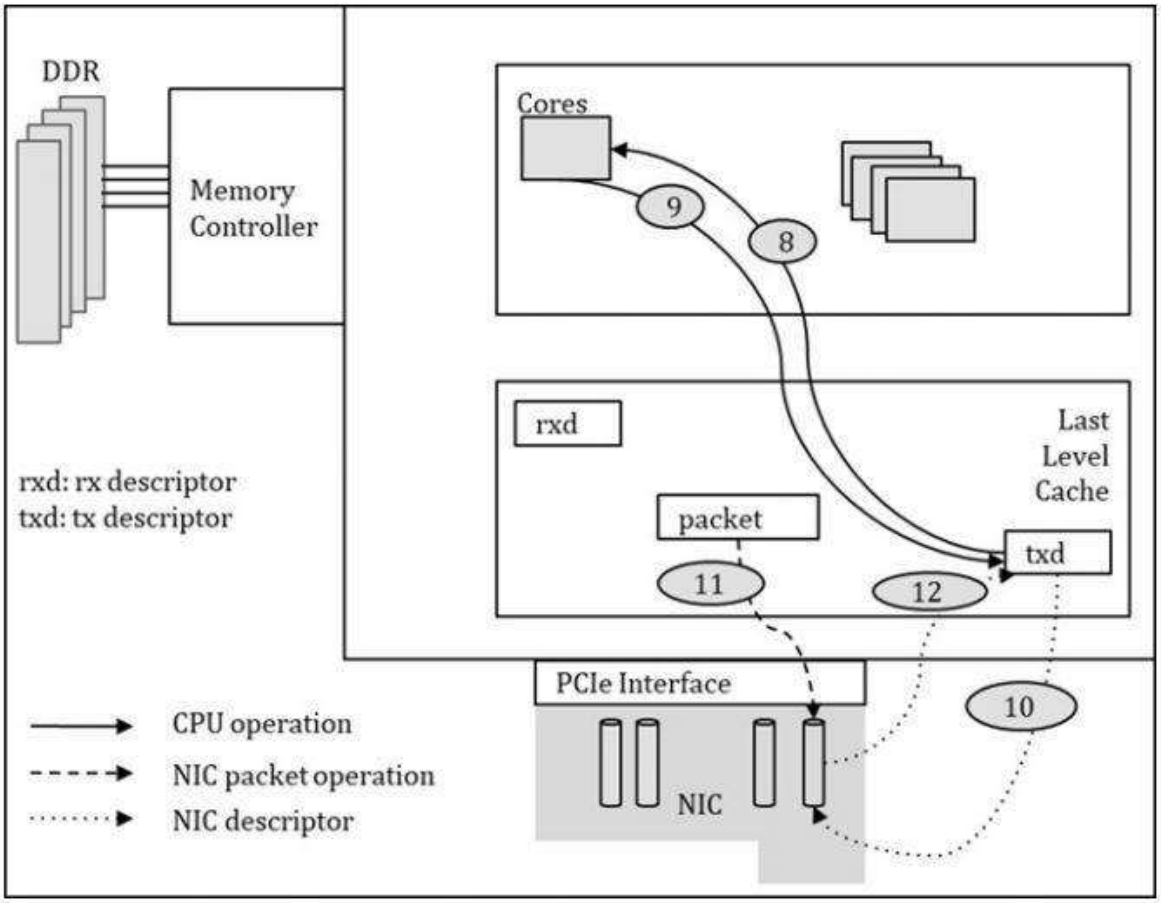
3. Η NIC γράφει το πακέτο στην ορισμένη διεύθυνση μνήμης (DMA)
4. Η NIC ενημερώνει τον RX προσδιοριστή για να επιβεβαιώσει την ολοκλήρωση του RX
5. Ο CPU διαβάζει τον RX προσδιοριστή και ολοκληρώνει το RX του πακέτου

   Τα βήματα 1 και 5 αφορούν πρόσβαση μνήμης, ο επεξεργαστής διαβάζει και γράφει τα δεδομένα από την μνήμη στην LLC.

   Το βήμα 2 είναι διαδικασία PCIe downstream

   Τα βήματα 3 και 4 είναι διαδικασίες PCIe upstream
6. Ο CPU επεξεργάζεται το πακέτο και αποφασίζει το επόμενο βήμα(επεξεργασία).
7. Ο CPU μπορεί να αλλάξει το πακέτο και να το στείλει.

   Τα βήματα 6 και 7 βρίσκονται πέραν της επεξεργασίας της NIC και δεν χρησιμοποιούν PCIe.



Εικόνα 6: Διαδικασία Tx και DDIO

8. Ο CPU διαβάζει τον TX προσδιοριστή για να ελέγξει εάν κάποιο πακέτο έχει αποσταλεί επιτυχώς.

9. Ο CPU γράφει την διεύθυνση του πακέτου στον buffer, στον επόμενο TX προσδιοριστή.
10. Η NIC διαβάζει τον TX προσδιοριστή για να βρει το πακέτο στην μνήμη.
11. Η NIC διαβάζει τα δεδομένα του πακέτου από την διεύθυνση.
12. Η NIC ενημερώνει τον TX προσδιοριστή αφού έχει ολοκληρωθεί η αποστολή του πακέτου (TX).

Τα βήματα 8 και 9 αφορούν πρόσβαση στην μνήμη, και διάβασμα και γράψιμο του CPU στην LLC.

Τα βήματα 10 και 11 είναι PCIe downstream διαδικασίες.

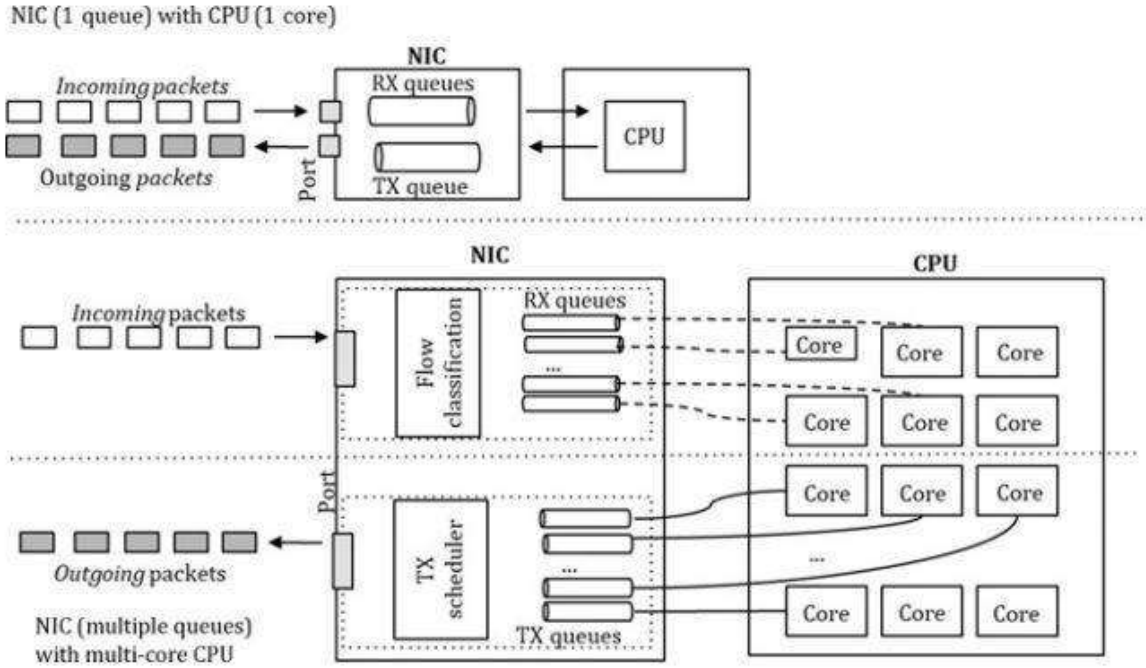
Το βήμα 12 είναι PCIe upstream διαδικασία.

2.5 Παραλληλισμός βασισμένος στην NIC

Στα πλαίσια λειτουργίας μια κάρτας δικτύου (NIC) σε περιβάλλον πολυπυρηνικής τεχνολογίας ενός επεξεργαστή, η εισερχόμενη κίνηση της NIC χρειάζεται να υιοθετήσει την φιλοσοφία της παραλληλίας, δηλαδή να χωριστεί σε επιμέρους ουρές, κάθε μία προς ένα συγκεκριμένο πυρήνα του επεξεργαστή για περαιτέρω επεξεργασία.

2.5.1 Πολλαπλές ουρές

Οι πολλαπλές ουρές είναι ένας μηχανισμός διαμοιρασμού των πακέτων που λαμβάνονται/αποστέλλονται σε πολλές ουρές. Έτσι αναθέτοντας κάθε ουρά σε συγκεκριμένο πυρήνα του επεξεργαστή υπάρχει η δυνατότητα παραλληλισμού της επεξεργασίας των πακέτων, δηλαδή υψηλότερες ταχύτητες επεξεργασίας (βλ. εικόνα 7). Οι πολλαπλές ουρές υποστηρίζονται από την NIC και προς τις δύο κατευθύνσεις (λήψη/αποστολή). Τα εισερχόμενα πακέτα μπορούν να οδηγηθούν σε διαφορετικές ουρές και είναι ένας μηχανισμός εξισορρόπησης φορτίου για μεγάλη κίνηση πακέτων. Σε συνδυασμό με την πολυπυρηνική αρχιτεκτονική, δημιουργούνται βάσεις για μία πολυ-νηματική αρχιτεκτονική επεξεργασίας πακέτων που δίνει την δυνατότητα για υψηλές επιδόσεις.



Εικόνα 7: Πολλαπλές ουρές και πολλαπλοί πυρήνες

Οι πολλαπλές ουρές πρέπει να ενσωματωθούν με ενεργοποίηση από το λογισμικό, και για drivers χώρου χρήστη όπως το DPDK, η διεπαφή Ethernet PMD είναι φτιαγμένη να υποστηρίζει πολλαπλές ουρές, ο αριθμός των οποίων προσδιορίζεται κατά την αρχικοποίηση της NIC(μπορεί να διαφέρει μεταξύ RX/TX). Το σύστημα θα αποφασίσει για τα παρακάτω:

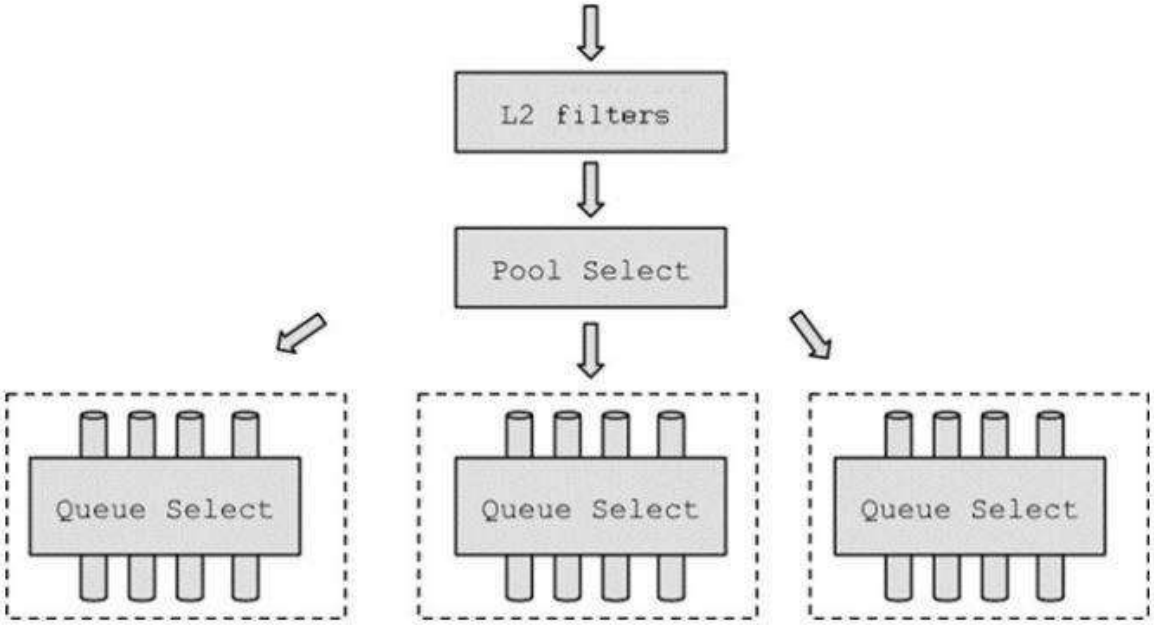
- Πόσες ουρές πρέπει να οριστούν;
- Πόσοι πυρήνες μπορούν να χρησιμοποιηθούν από τον NIC driver;
- Πόσοι πυρήνες είναι διαθέσιμοι για άλλη χρήση από λογισμικό επεξεργασίας πακέτων;
- Ποια είναι η μέθοδος διανομής των πακέτων από τον πυρήνα εισόδου εξόδου σε διαφορετικούς πυρήνες εργάτες(επεξεργασία με εξισορροπημένο φορτίο, ή επεξεργασία φορτίου με την κατεύθυνση που ορίζει η εφαρμογή);

Η NIC, διανέμοντας την κίνηση σε πολλές ουρές ,που εξυπηρετούνται από διαφορετικούς πυρήνες, επιτρέπει την παράλληλη επεξεργασία αυτών. Η επεξεργασία συγκεκριμένης κίνησης που ορίζεται από την εφαρμογή μπορεί να φανεί χρήσιμη, όπως στην περίπτωση του IPSEC. Εκεί η NIC μπορεί να αναθέσει μία ή περισσότερες ουρές μόνο για τα πακέτα IPsec.

2.5.2 Διαμοιρασμός Κίνησης

Για τον διαμοιρασμό της κίνησης υπάρχουν πολλές τεχνολογίες που ακολουθούν την παρακάτω φιλοσοφία. Όταν ένα πακέτο φτάνει λοιπόν στην NIC, ακολουθείται η εξής διαδικασία:

1. Αναγνώριση του πακέτου που φθάνει στην Ethernet πόρτα
2. Φιλτράρισμα του πακέτου σύμφωνα με L2 Ethernet πρωτόκολλο
3. Ανάθεση ουράς (επιλογή δεξαμενής(pool), επιλογή ουράς)
4. Μεταφορά του πακέτου στην FIFO ουρά
5. Μεταφορά του πακέτου στην μνήμη που ανήκει στην συγκεκριμένη ουρά
6. Ενημέρωση της κατάστασης του προσδιοριστή buffer του πακέτου



Εικόνα 8: Διαμοιρασμός σε ουρές από την NIC

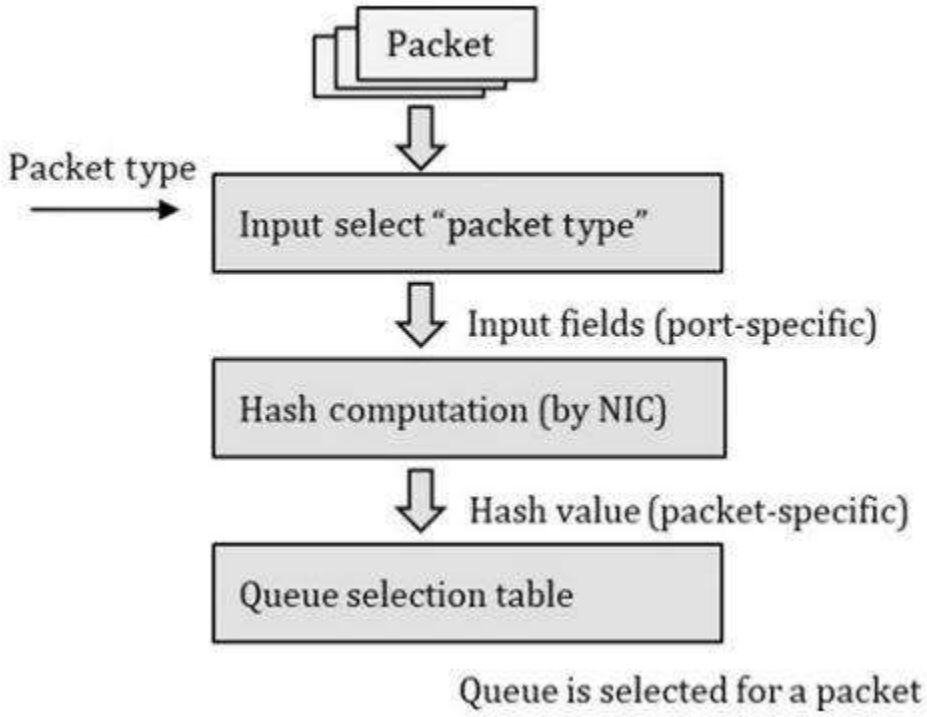
Ο έλεγχος του πακέτου αφορά το αν έχει επιτρεπτό μέγεθος και αν δεν υπάρχει λάθος στο CRC. Έπειτα το φιλτράρισμα αφορά την L2 επικεφαλίδα: διεύθυνση MAC, VLAN, Ethernet τύπος. Ενέργειες για να δεχθεί ή απορριφθεί ένα πακέτο αφορούν την παραμετροποίηση του λογισμικού. Αν αυτό γίνει δεκτό, το επόμενο βήμα είναι η τοποθέτηση του σε ουρά, δηλαδή επιλογή δεξαμενής και ουράς (βλ εικόνα 8).

Κατά τη διανομή των πακέτων σε ουρές, ειδικά σε ένα περιβάλλον εικονοποίησης με πολλές εικονικές μηχανές, η NIC μπορεί να παραμετροποιηθεί, έτσι ώστε να χωριστεί σε PFs/VFs

(φυσικές/εικονικές συναρτήσεις – όρος του SR-IOV), κάθε μία από τις οποίες μπορεί να αντιστοιχιστεί σε συγκεκριμένη εικονική μηχανή. Οι δεξαμενές (pools) μπορούν να αντιστοιχιστούν σε VFs, και κάθε δεξαμενή περιέχει πολλαπλές ουρές. Έτσι σε υπηρεσίες βασισμένες σε cloud όπου υπάρχει υψηλή απαίτηση ταχυτήτων, μπορεί να γίνει αποτελεσματικά η επεξεργασία της κίνησης για κάθε VM.

2.5.3 RSS

Το RSS (receive-side scaling) είναι ένα χαρακτηριστικό υλικού NIC, σχεδιασμένο για ισομοιρασμό της κίνησης, που διανέμει τα πακέτα σε διαφορετικές ουρές σύμφωνα με ένα hash αλγόριθμο των επικεφαλίδων των πακέτων (βλ. εικόνα 9).[16]



Εικόνα 9: Λειτουργία του RSS

Είναι δύσκολο να προβλεφθεί εάν το RSS θα μοιράσει ισομερώς τα πακέτα στις ουρές. Εξαρτάται προφανώς από την εισερχόμενη κίνηση και τον αλγόριθμο hash που χρησιμοποιεί η NIC.

[16] Microsoft, 'Introduction to Receive Side Scaling', *mircrosoft* [ιστοσελίδα], https://learn.microsoft.com/en-us/windows-hardware/drivers/network/introduction-to-receive-side-scaling, (τελευταία πρόσβαση 29 Ιανουαρίου 2023)

Αναλόγως το πρωτόκολλο L4 (UDP,TCP,κα) σε συνδυασμό με IPv4,IPv6, επιλέγονται διαφορετικές παράμετροι και hashing αλγόριθμος.

2.5.4 FDIR

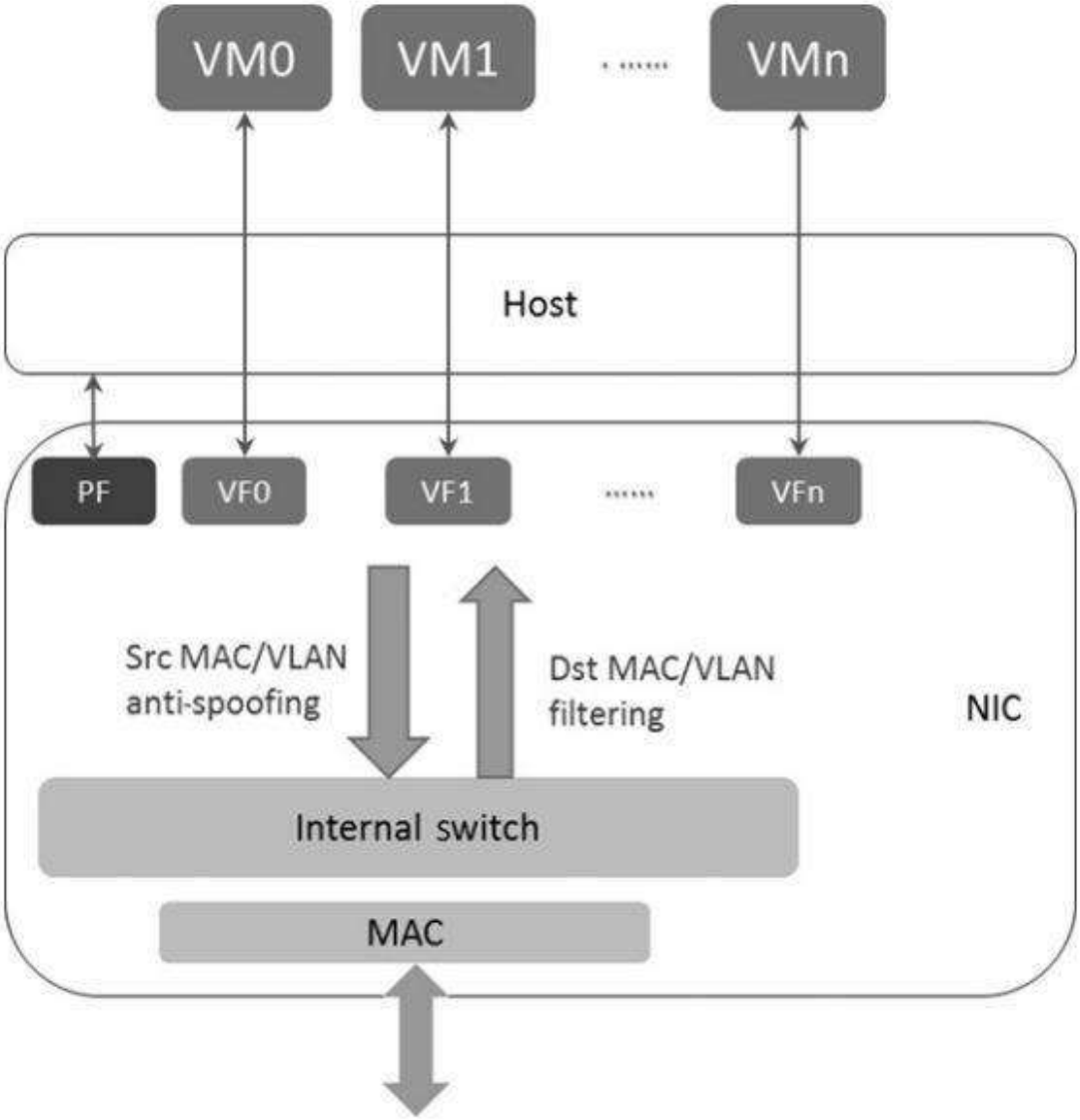
Το FDIR(Flow Director) είναι ένας μηχανισμός που επιτρέπει στην NIC να διανέμει πακέτα σε διαφορετικές ουρές αναλόγως τις επικεφαλίδες του πακέτου. Η διαφορά με το RSS είναι ότι στο FDIR κάθε είδος πακέτου όπως πχ IPsec,UDP κατευθύνεται σε μία ουρά. Με αυτόν τον τρόπο επιτρέπεται στην εφαρμογή να επιλέγει συγκεκριμένες ροές και να τις κατευθύνει σε αντίστοιχους πυρήνες.

Ένα Flow Director table δημιουργείται στην NIC, όπου γίνεται αντιστοίχηση του τύπου πρωτοκόλλου ενός πακέτου εισόδου με την αντίστοιχη καταχώρηση και περαιτέρω ανάθεση σε ορισμένη ουρά.

2.5.5 SR-IOV

Για την περίπτωση του cloud, όταν υπάρχει έναν φιλοξενούν μηχάνημα (host) και πολλές φιλοξενούμενες (guest) εικονικές μηχανές, τα πακέτα οδηγούνται στην μνήμη του host ή στην μνήμη που έχει παραχωρηθεί σε κάθε εικονική μηχανή. Ο έλεγχος που πρέπει να κάνει ο hypervisor για την επιλογή προορισμού του πακέτου δημιουργεί πρόσθετες έμμεσες καθυστερήσεις και δεν επιτρέπει την επίτευξη μεγάλων ταχυτήτων εισόδου/ εξόδου.

Το SR-IOV (single root I/O Virtualization) δίνει λύση στο παραπάνω πρόβλημα χωρίζοντας την NIC σε VFs(εικονικές συναρτήσεις), που ανατίθενται στα VMs χωρίς την μεσολάβηση από το hypervisor. Όταν λαμβάνεται ένα πακέτο, η NIC επιλέγει ποια εικονική μηχανή θα το παραδώσει. Από την οπτική της εικονικής μηχανής, ένα εξερχόμενο πακέτο μπορεί να εξέλθει από την NIC προς ένα εξωτερικό μηχάνημα, ή σε ένα άλλο VM του ίδιου host.



Εικόνα 10: SR-IOV και κατεύθυνση ροών προς τις εικονικές μηχανές

Στην NIC, εάν υπάρχει switch βασισμένος στο SR-IOV, ελέγχεται το πακέτο σύμφωνα με τα πεδία MAC,VLAN για να κατευθύνει το πακέτο αναλόγως στην αντίστοιχη εικονική μηχανή (βλ. εικόνα 10). Σε συστήματα μεγάλης κλίμακας, η απόφαση σε ποιο VM γίνεται η προώθηση της κίνησης έχει μεταφερθεί στο L3, και αφορά και τις διευθύνσεις IP. Με αυτόν τον τρόπο η επιλογή προορισμού ανατίθεται στην NIC και όχι σε κάποιον ελεγκτή εικονοποίησης, συνδέοντας ουσιαστικά τις εικονικές μηχανές με το υλικό. Η απουσία μεσάζοντα δηλαδή στην καθοδήγηση των πακέτων επιτρέπει την επίτευξη υψηλών επιδόσεων εισόδου εξόδου σε συστήματα cloud.

Κεφάλαιο 3 : DPDK

3.1 Παραλαβή πακέτων σε ένα σύστημα Linux

Παραδοσιακά η επεξεργασία των πακέτων από την NIC συσκευή, σε ένα σύστημα εξυπηρετητή γινόταν μέσω του Linux με τα εξής βήματα

- Το πακέτο φτάνει σε ένα NIC
- Το NIC ολοκληρώνει απευθείας πρόσβαση στην μνήμη (DMA) και αντιγράφει το πακέτο σε μια δομή αποθήκευσης γνωστή ως packet buffer
- Το NIC στέλνει εξαίρεση για να ξυπνήσει τον επεξεργαστή
- Ο επεξεργαστής διαβάζει και γράφει τον προσδιοριστή του πακέτου και τον buffer των πακέτων
- Το πακέτο στέλνεται στο Linux kernel protocol stack για περαιτέρω επεξεργασία πρωτοκόλλων, όπως για παράδειγμα απόφαση ελέγχου πρόσβασης σχετικά με το πρωτόκολλο IP
- Αν η εφαρμογή βρίσκεται στον χώρο χρήστη, τα δεδομένα του πακέτου θα αντιγραφούν από το χώρο πυρήνα στον χώρο χρήστη
- Αν η εφαρμογή βρίσκεται στον χώρο πυρήνα, τα δεδομένα θα επεξεργαστούν στον χώρο πυρήνα (μικρό ποσοστό του συνόλου των εφαρμογών)

Σε παλιά συστήματα, κάθε πακέτο που λαμβάνεται δημιουργεί μια διακοπή. Το φορτίο της εξαίρεσης περιλαμβάνει την αλλαγή/διακοπή περιεχομένου(context switch). Μία διακοπή είναι αποδοτική, όταν δεν έρχονται πολλά πακέτα στο σύστημα σε μικρό χρονικό διάστημα. Τα τελευταία χρόνια όμως, ενώ η συχνότητα του επεξεργαστή έχει παραμείνει σχεδόν σταθερή, οι ταχύτητες του διαδικτύου συνεχίζουν να αυξάνονται με μεγάλους ρυθμούς και μάλιστα με άλμα από 10/100 Mbps σε ταχύτητες πια 1,10,25,40 και 100 Gbps.

Έτσι, το λογισμικό έρχεται αντιμέτωπο με την διαχείριση μεγάλου φόρτου πακέτων ,σε ένα διαρκώς αναπτυσσόμενο διαδίκτυο, όπου τεράστιοι αριθμοί πακέτων καταφθάνουν, με αποτέλεσμα το σύστημα να μην έχει την δυνατότητα για μεγάλο αριθμό επεξεργασίας διακοπών. Η μέθοδος NAPI[17] μπορεί, σε μεγάλο βαθμό, να βελτιώσει την αποτελεσματικότητα της επεξεργασίας πακέτων σε ένα σενάριο έκρηξης κίνησης. Μετέπειτα, το πρωτόκολλο Netmap[18]

[17] Linux Foundation, 'napi', *linux foundation wiki* [ιστοσελίδα], https://wiki.linuxfoundation.org/networking/napi, (τελευταία πρόσβαση 31 Ιανουαρίου 2023)
[18] L. Rizzo, 'netmap: a novel framework for fast packet I/O', *21st USENIX Security Symposium*, 2012

χρησιμοποιήθηκε για υψηλών αποδόσεων δίκτυα εισόδου εξόδου. Το Netmap χρησιμοποιεί κοινό αποθηκευτικό χώρο για τα πακέτα για να μειώσει την αντιγραφή από τον χώρο πυρήνα στον χώρο χρήστη. Οι δύο παραπάνω λύσεις, βοήθησαν σε μεγάλο βαθμό το παραδοσιακό σύστημα Linux να αυξήσει τις επιδόσεις του σε επεξεργασία πακέτων. Υπάρχει όμως περιθώριο για περαιτέρω βελτίωση.

Ως ένα λειτουργικό σύστημα διαμοιρασμού χρόνου, το Linux προγραμματίζει εργασίες με τον μηχανισμό των χρονικών διαστημάτων. Ο scheduler του Linux αναλαμβάνει να κατανέμει τα χρονικά διαστήματα στις διάφορες εργασίες. Παλιότερα ο αριθμός των πυρήνων του επεξεργαστή ήταν μικρός, οπότε για την ολοκλήρωση των εργασιών σε λογικό χρονικό διάστημα, ο μηχανισμός αυτός ήταν αρκετά αποτελεσματικός για να μπορούν διαφορετικές εργασίες να χρησιμοποιούν τον πολύτιμο πόρο των κύκλων επεξεργαστή. Αργότερα που τα συστήματα είχαν στην διάθεση τους περισσότερους πυρήνες, οδηγήθηκαν σε νέες, πιο αποδοτικές, λύσεις.

Καθώς ο διαμοιρασμός χρονικών διαστημάτων χρησιμοποίησης των κύκλων του επεξεργαστή δεν είναι πολύ αποδοτικός όσον αφορά την ταχύτητα, προτάθηκε η λύση της εξολοκλήρου αφοσίωσης κάποιων πυρήνων σε συγκεκριμένες εργασίες. Το Netmap μειώνει την αντιγραφή στη μνήμη μεταξύ πυρήνα και χρήστη αλλά δεν την εξαλείφει. Ο επιπλέον φόρτος από την αλλαγή μεταξύ εργασιών και η αλλαγή περιεχομένου στις cache που προκύπτει από την αλλαγή εργασιών, επιβαρύνουν την επίδοση του συστήματος.

Εκ φύσεως, ο φόρτος εργασίας του διαδικτύου είναι ευαίσθητος στην καθυστέρηση, και τα πακέτα μπορεί να ταξιδέψουν μέσα από πολλούς κόμβους. Ως αποτέλεσμα, ο πραγματικός χρόνος είναι μείζων ζήτημα και απαίτηση για ένα δικτυακό σύστημα. Από την εξαίρεση της NIC, μέχρι τις διακοπές λογισμικού και τελικά στην επεξεργασία του πακέτου από την εφαρμογή, χρησιμοποιούνται πολλοί κύκλοι μη αποδοτικά. Τίθεται λοιπόν το ερώτημα πώς θα μετατραπεί το σύστημα Linux σε περιβάλλον επεξεργασίας πακέτων αξιοποιώντας τις δυνατότητες που παρέχει η πολυπυρηνική αρχιτεκτονική.

3.2 Η λύση του DPDK

Το λογισμικό ανοιχτού κώδικα DPDK (data plane development kit) που δημιουργήθηκε από την INTEL το 2010 , είναι μια δικτυακή τεχνολογία παράκαμψης του Kernel και χρησιμοποιείται κατά

κόρον από συστήματα ασφάλειας, δικτύωσης επιχειρήσεων,, τηλεπικοινωνιών και cloud[19]. Αποτελείται από βιβλιοθήκες χώρου χρήστη και drivers που επιταχύνουν την επεξεργασία πακέτων σε όλες τις μεγάλες αρχιτεκτονικές υπολογιστών. Το DPDK, εξαιτίας της μεγάλης αποδοχής και διάδοσης του, αποτελεί πλέον project υπό το Linux Foundation.

Το DPDK παρέχει λύσεις στα παραπάνω προβλήματα ειδικά μέσω του PMD (polling mode driver) και έχει εδραιωθεί ως ένα λογισμικό του Linux για επεξεργασία πακέτων σε μεγάλες ταχύτητες. Βασίζεται σε ένα σύνολο από αρχές βελτιστοποίησης λογισμικού, που υλοποιούν υψηλές επιδόσεις μετακίνησης πακέτων σε πολυπύρηνους επεξεργαστές :

- *Polling mode*: Ανάθεση πυρήνων αποκλειστικά για λήψη και αποστολή πακέτων μέσω του NIC. Με αυτήν την προσέγγιση το σύστημα δεν μοιράζεται πυρήνες με άλλες εργασίες λογισμικού και ο πυρήνας τρέχει ατέρμονα, για να ελέγξει αν έχει φτάσει κάποιο πακέτο ή χρειάζεται να στείλει κάποιο πακέτο, με αποτέλεσμα να μην υπάρχει ανάγκη για διακοπές και την καθυστέρηση που αυτές επιφέρουν.
- *Driver χώρου χρήστη*: Στη μεγάλη πλειονότητα των περιπτώσεων, το πακέτο εν τέλει θα πρέπει να σταλεί στον χώρο του χρήστη. Οι Linux NIC drivers είναι ως επί το πλείστον βασισμένοι στον πυρήνα. Ο driver χώρου χρήστη μπορεί να αποφύγει περιττές αντιγραφές πακέτων από τον χώρο πυρήνα στον χώρο χρήστη και σώζει από το κόστος των κλήσεων συστήματος. Η μορφή της επικεφαλίδας της μνήμης buffer βασισμένη στο DPDK, είναι ευέλικτη ως προς τον ορισμό της, έτσι ώστε να μπορεί να σχεδιαστεί για DMA από την NIC. Αυτή η ευελιξία προσθέτει στο όφελος επιδόσεων.
- *Συγγένεια πυρήνα*: Θέτοντας την συγγένεια (affinity) ενός νήματος με έναν συγκεκριμένο πυρήνα του επεξεργαστή, συγκεκριμένες εργασίες μπορούν να συνδεθούν με πυρήνες. Χωρίς την συγγένεια νημάτων, μπορεί να υπάρχει αλλαγή εργασιών μεταξύ διαφορετικών πυρήνων. Το πρόβλημα που δημιουργεί αυτό είναι ότι η αλλαγή εργασιών προκαλεί αστοχίες της μνήμης cache και write-backs. Ένα ακόμα βήμα είναι να ζητηθεί από έναν πυρήνα να αποκλειστεί από το σύστημα scheduling του Linux, έτσι ώστε ο πυρήνας να χρησιμοποιείται μόνο για την συγκεκριμένη εργασία.
- *Βελτιστοποιημένη διαχείριση μνήμης*: Το ζητούμενο είναι η όσο το δυνατόν ταχύτερη τροφοδότηση με δεδομένα πακέτων στην cache, ώστε να μην καθυστερεί ο επεξεργαστής. Η επεξεργασία δικτυακής ροής υψηλού ρυθμού είναι ένα σενάριο έντασης φόρτου εισόδου/ εξόδου.

[19] AvidThink, 'White Paper: Myth-Busting DPDK in 2020', *dpdk* [ιστοσελίδα], https://www.dpdk.org/about/news/, (τελευταία πρόσβαση 29 Ιανουαρίου 2023)

Τόσο ο επεξεργαστής όσο και το NIC χρειάζονται πρόσβαση στα δεδομένα στην μνήμη (cache ή/και DRAM). Η βέλτιστη πρόσβαση στην μνήμη περιλαμβάνει την δέσμευση και χρήση σελίδων μνήμης HugePages[20] (μεγάλου μεγέθους π.χ 2GB) από συνεχόμενες περιοχές μνήμης, ώστε να μειώνονται οι αστοχίες από την μετάφραση εικονικής σε φυσική μνήμη μέσω TLB(Translation Lookaside Buffer[21])[22]. Επιπλέον, η πρόσβαση στην μνήμη μέσω πολλών καναλιών προς αποφυγή του κόστους πρόσβασης στην μνήμη διαδοχικά και η ασύμμετρη πρόσβαση στην μνήμη μπορούν να μειώσουν την καθυστέρηση πρόσβασης. Η βασική ιδέα είναι να οδηγηθούν τα δεδομένα στην cache όσο το δυνατόν γρηγορότερα για να μην καθυστερεί ο επεξεργαστής.

- *Ρυθμίσεις λογισμικού*: Πρακτικές όπως ευθυγράμμιση γραμμής της cache μιας δομής δεδομένων, αποφυγή πολλοί πυρήνες να μοιράζονται κοινούς πόρους, προανάκτηση δεδομένων κα.
- *Χρησιμοποίηση των τελευταίων τεχνολογιών και συνόλων εντολών*: Για παράδειγμα η τεχνολογία του DDIO, που ενισχύει τις επιδόσεις εισόδου εξόδου, αφού τα πακέτα μπορούν να τοποθετηθούν απευθείας στην μνήμη cache, μειώνοντας την καθυστέρηση πρόσβασης από την κύρια μνήμη.
- *NIC driver tuning*: Όταν ένα πακέτο εισέρχεται στο σύστημα μέσω μιας PCIe διεπαφής, η επίδοση εισόδου/εξόδου επηρεάζεται από την ταχύτητα συναλλαγής μεταξύ της βασισμένης σε PCIe συσκευή, το bus και την μνήμη του συστήματος. Η συγχώνευση πακέτων, δηλαδή η μεταφορά πολλών πακέτων ταυτόχρονα, μπορεί να επιφέρει μεγάλη διαφορά σε ταχύτητες, καθώς αξιοποιεί καλύτερα το bus. Σύγχρονες NICs υποστηρίζουν τεχνικές εξισορρόπησης φορτίου όπως RSS που επιτρέπουν στο NIC να αξιοποιεί το πολυπυρηνικό μοντέλο του επεξεργαστή. Το RSS επιτρέπει στο NIC να μοιράζει την επεξεργασία των πακέτων που λαμβάνει σε πολλαπλούς πυρήνες, και στην περίπτωση του DPDK χρησιμοποιείται με την μέθοδο των ουρών που ανατίθενται σε συγκεκριμένους επεξεργαστές, έχοντας πολλές ουρές ανά port της NIC. Το NIC offloading έχει την δυνατότητα να ελέγχει το άθροισμα της επικεφαλίδας των πακέτων. Ακόμα μπορεί να κάνει TSO, μια τεχνική που επιτρέπει επεξεργασία πακέτων TCP πολύ μεγάλου μεγέθους από την NIC, LRO, μία τεχνική συσσωμάτωσης πακέτων ίδιας ροής για να μειωθεί η επιβάρυνση στον επεξεργαστή και υπάρχει η δυνατότητα επεξεργασίας επικεφαλίδων tunnel.
- *Δικτυακή εικονοποίηση και επιτάχυνση cloud*: Το dpdk παρέχει τον βέλτιστο τρόπο μεταφοράς πακέτων από το φιλοξενούν στα φιλοξενούμενα μηχανήματα (VMs). Το παραπάνω στοιχείο είναι

[20] Debian, 'Hugepages', *debian wiki* [ιστοσελίδα], https://wiki.debian.org/Hugepages, (τελευταία πρόσβαση 1 Φεβρουαρίου 2023)
[21] R. H. ARPACI-DUSSEAU, A. C. ARPACI-DUSSEAU, 'Paging: Faster Translations (TLBs)' στο *OPERATING SYSTEMS: THREE EASY PIECES*, Arpaci-Dusseau Books, Αύγουστος 2018
[22] Βλ. σχετικά 3.4.1

πολύ σημαντικό για τις υποδομές cloud και NFV (network functions virtualization). Το DPDK υποστηρίζει SR-IOV και vSwitch βελτιστοποίηση μέσω PMD.

3.3 Αξιοποίηση του πολυπυρηνικού συστήματος και τεχνολογίες απομόνωσης.

3.3.1 Affinity

Σε ένα πολυπυρηνικό σύστημα ο scheduler του λειτουργικού συστήματος είναι υπεύθυνος για την ανάθεση των διαδικασιών στους πυρήνες. Χωρίς την ύπαρξη μηχανισμού affinity, ένα νήμα λογισμικού μπορεί να κινείται από τον έναν πυρήνα στον άλλον. Αυτή η προσέγγιση είναι χρήσιμη σε περιπτώσεις ίσου διαμοιρασμού φορτίου στους πυρήνες, αλλά δεν είναι ιδανική για επίτευξη μέγιστων επιδόσεων. Η μετακίνηση αυτή επιφέρει καθυστερήσεις και προβλήματα μολυσμένης μνήμης cache. Για αυτόν τον λόγο χρησιμοποιείται affinity, δηλαδή ανάθεση μίας συγκεκριμένης διαδικασίας σε συγκεκριμένο πυρήνα. Ο κάθε πυρήνας διαθέτει την δική του L1/L2 μνήμη cache, όπου διατηρεί τα συχνά χρησιμοποιούμενα δεδομένα. Έτσι αναθέτοντας την εργασία στον συγκεκριμένο πυρήνα, υπάρχει μεγαλύτερη πιθανότητα για cache hit, αποφεύγοντας καθυστερήσεις.

3.3.2 Απομόνωση πυρήνων

Ο μηχανισμός affinity, ενώ αναθέτει μια διαδικασία σε έναν πυρήνα, δεν αποτρέπει τον scheduler των Linux να φορτώσει και άλλες διαδικασίες στον συγκεκριμένο πυρήνα. Η αλλαγή μεταξύ των διαδικασιών επιφέρει καθυστερήσεις, για αυτό υπάρχει η δυνατότητα να οριστούν κάποιοι πυρήνες μη διαθέσιμοι προς τον scheduler των Linux και να χρησιμοποιηθούν μόνο για συγκεκριμένες διαδικασίες.

Με affinity και απομόνωση πυρήνα(core isolation) ,επιτυγχάνεται συγκεκριμένοι πυρήνες να χρησιμοποιούνται αποκλειστικά για συγκεκριμένες διαδικασίες λογισμικού. Είναι φανερό ότι ο παραπάνω συνδυασμός είναι βασική αρχή του DPDK, καθώς επιτρέπει την χρησιμοποίηση 100% των εν λόγω πυρήνων για την επεξεργασία πακέτων. Προφανώς η χρησιμοποίηση 100% των πυρήνων γίνεται δυνατή μόνο με τη χρήση πολυπυρηνικών συστημάτων, καθώς σε μονοπυρηνικά συστήματα υπάρχουν και άλλες απαραίτητες διαδικασίες που χρειάζονται κύκλους επεξεργαστή.

3.3.3 Μοντέλο RTC

Το μοντέλο RTC (run to completion) είναι βασική φιλοσοφία του DPDK και συνδυάζεται με τις τεχνικές απομόνωσης, affinity και δέσμευσης πυρήνα για την γρήγορη εκτέλεση εργασιών. Το μοντέλο αφορά την εκτέλεση εργασιών από έναν συγκεκριμένο πυρήνα του επεξεργαστή για όλη

την διάρκεια της εργασίας, έτσι ώστε να αποφευχθούν αντιγραφές δεδομένων. Το μοντέλο RTC ταιριάζει απόλυτα με την πολυπυρηνική αρχιτεκτονική και φιλοσοφία παραλληλίας. Είναι ένα σειριακό μοντέλο, που έχει ως αποτέλεσμα την καθυστέρηση του πυρήνα σε περιπτώσεις πρόσβασης στην μνήμη.

3.4 NIC και μνήμη

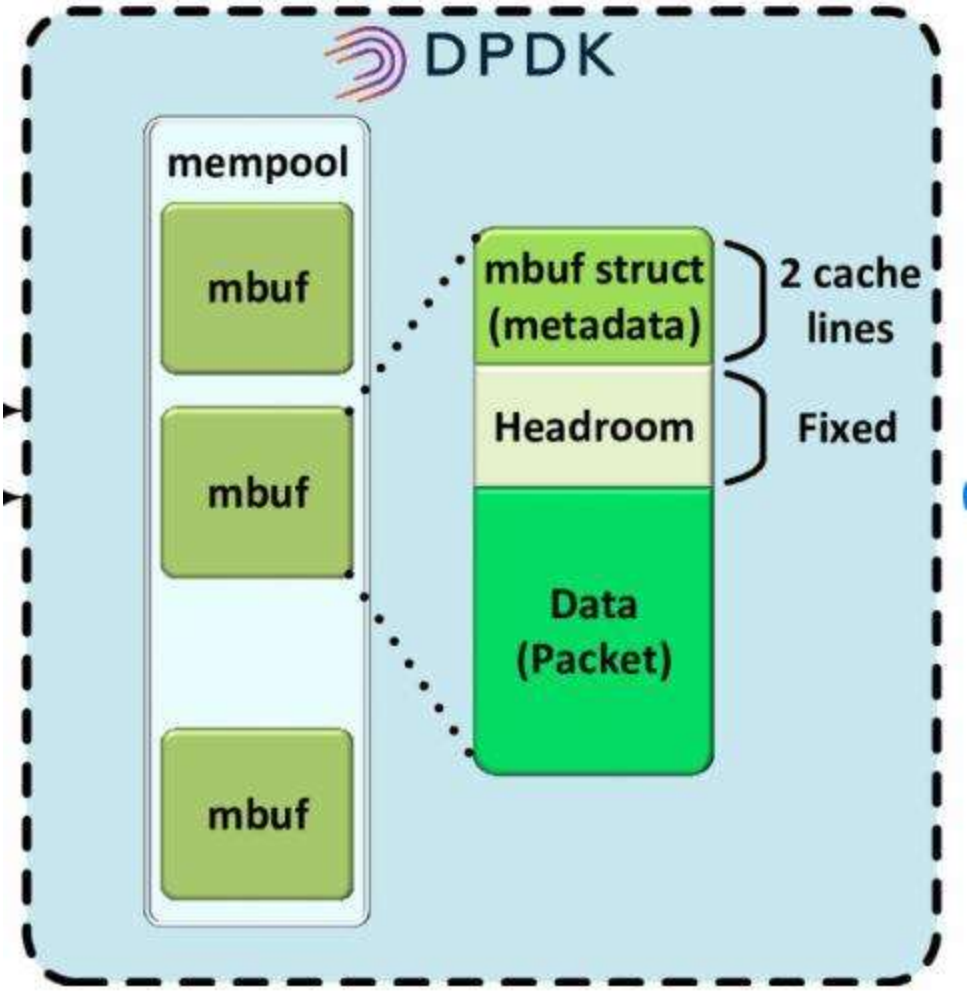
Το DPDK βελτιστοποιεί την χρήση της μνήμης και την επικοινωνία της με την NIC με σκοπό την επεξεργασία ροών πακέτων υψηλής ταχύτητας. Ένα βασικό χαρακτηριστικό του DPDK είναι η χρήση μεγάλων σελίδων Hugepages έτσι ώστε να υπάρχει συνεχώς καταγραφή στην TLB και να μην συμβαίνει page walk που καθυστερεί την πρόσβαση στα δεδομένα. Μέσα στις Hugepages το DPDK δημιουργεί την mempool στην οποία βρίσκονται όλα τα πακέτου τύπου mbuf.

3.4.1 TLB και HUGEPAGE

Σε ένα υπολογιστικό σύστημα, το λογισμικό χρησιμοποιεί την εικονική διεύθυνση μνήμης και όχι την φυσική διεύθυνση μνήμης. Η μνήμη συστήματος είναι δομημένη σε σελίδες με παραδοσιακό μέγεθος τα 4 KB. Αργότερα εμφανίζεται το Hugepage και υποστηρίζεται από το Linux ως 2MB ή 1 GB. Η μετάφραση μνήμης είναι μια πολυεπίπεδη αναζήτηση σελίδας σε πίνακα. Το TLB είναι μέρος του CPU και μπορεί να επιταχύνει την διαδικασία εύρεσης της φυσικής θέσης μνήμης από την εικονική, καθώς περιέχει την αντιστοιχία εικονικής σελίδας με την φυσική σελίδα στην μνήμη. Αν λοιπόν η επιθυμητή σελίδα βρίσκεται στην TLB, η πρόσβαση γίνεται αμέσως και με ελάχιστη καθυστέρηση. Αντίθετα, αν αστοχήσει, η αναζήτηση μπορεί να πάρει πολλούς κύκλους, με μία μέθοδο που ονομάζεται page walk, μέχρι να βρει την πραγματική διεύθυνση. Αν ακόμα δεν βρίσκεται στην cache, θα χρειαστεί εκατοντάδες κύκλους επεξεργαστή για να την φέρει από την μνήμη. Για αυτό, το DPDK αξιοποιεί το hugepage θέτοντας hugepages για μνήμη επεξεργασίας πακέτων, τα οποία λόγω του τεράστιου μεγέθους τους, χρησιμοποιούνται συνεχώς και βρίσκονται πάντα στο TLB. Έτσι, το DPDK, αποφεύγει πολλούς κύκλους πρόσβασης στην μνήμη.

3.4.2 Mempool

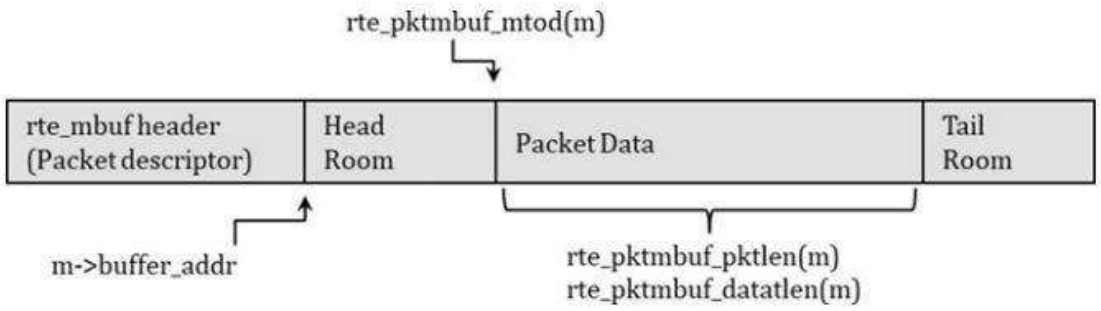
Η mempool αξιοποιεί τις μεγάλες σελίδες για να φιλοξενήσει τα πακέτα που προέρχονται από την NIC και αποθηκεύονται σε αυτήν (βλ. εικόνα 12). Επειδή βρίσκεται στον χώρο χρήστη, είναι άμεσα προσβάσιμη από εφαρμογές, για οποιαδήποτε επεξεργασία πακέτου, και δεν χρειάζεται καθυστέρηση για αλλαγή περιεχομένου από χώρο χρήστη και χώρο πυρήνα. Κάθε nic ουρά έχει και αντίστοιχο Mempool στην μνήμη.



Εικόνα 11: Δομή της mempool και mbuf στην μνήμη

3.4.3 Mbuf

Κάθε πακέτο που λαμβάνεται ή αποστέλλεται μέσω της NIC χρειάζεται ένα μέρος στην μνήμη. Το DPDK αναφέρεται σε αυτό το κομμάτι της μνήμης ως mbuf. Στο DPDK το mbuf είναι σχεδιασμένο ως δομή δεδομένων βασισμένη σε πακέτα (βλ. εικόνα 13). Η επικεφαλίδα του mbuf είναι ο προσδιοριστής πακέτου (ή προσδιοριστής buffer), είναι metadata, που περιγράφουν το πακέτο.



Εικόνα 12: Δομή της mbuf

Τα head και tail μέρη, είναι διαθέσιμα για αλλαγές στο πακέτο και ο χώρος μπορεί να δεσμευτεί για να μην υπάρχει ανάγκη για αντιγραφή στην μνήμη. Οι περισσότερες αλλαγές στα πακέτα συμβαίνουν στην κεφαλή (όπως προσθήκη/αφαίρεση επικεφαλίδων πρωτοκόλλων) ή/και στην ουρά του πακέτου (όπως προσθήκη/έλεγχος της ακεραιότητας του πακέτου). Έτσι η αντιγραφή του πακέτου μπορεί να διατηρηθεί στην ίδια θέση, οι αλλαγές στην κεφαλή και την ουρά μπορούν να γίνουν στους δεσμευμένους χώρος και ο δείκτης να μεταφερθεί αναλόγως. Ο προσδιοριστής πακέτου περιέχει δεδομένα όπως VLAN ετικέτα, RSS hash τιμές, και αριθμό port που ελήφθη το πακέτο.

3.4.4 Τύπος πακέτου

Η NIC μπορεί να αναγνωρίσει πολλούς τύπους πακέτου τους οποίους καταγράφει στον περιγραφέα πακέτων. Στο παράδειγμα της εικόνας 13, φαίνεται η δομή δεδομένων που περιέχει τύπους πακέτων επιπέδου 2,3,4 (L2,L3,L4) και tunnel.

```
struct rte_mbuf {
        ……
        union {
                uint32_t packet_type; /**< L2/L3/L4 and tunnel information. */
                struct {
                        uint32_t l2_type:4; /**< (Outer) L2 type. */
                        uint32_t l3_type:4; /**< (Outer) L3 type. */
                        uint32_t l4_type:4; /**< (Outer) L4 type. */
                        uint32_t tun_type:4; /**< Tunnel type. */
                        uint32_t inner_l2_type:4; /**< Inner L2
                        type. */
                        uint32_t inner_l3_type:4; /**< Inner L3
                        type. */
                        uint32_t inner_l4_type:4; /**< Inner L4
                        type. */
                };
        };
        ……
}
```

Εικόνα 13: Παράδειγμα δομής του περιγραφέα πακέτου

Η NIC, μπορεί να χρησιμοποιήσει τις πληροφορίες αυτές για επεξεργασία χωρίς μεταφορά του πακέτου από την μνήμη.

3.5 Συνεκτικότητα δεδομένων και μηχανισμοί lock

Σε ένα πολυπυρηνικό σύστημα, σε περιοχές της μνήμης και δεδομένα που πρέπει να έχουν πρόσβαση πολλοί πυρήνες, τίθεται το ζήτημα της ακεραιότητας δεδομένων. Προς αποφυγή προβλημάτων συνοχής στα δεδομένα, υπάρχουν μηχανισμοί lock που δεν επιτρέπουν σε πυρήνες να έχουν πρόσβαση σε δεδομένα όταν αυτά μεταβάλλονται ή χρησιμοποιούνται από άλλον πυρήνα.

3.5.1 RWLOCK

Το κλείδωμα RW (ανάγνωση/εγγραφή) είναι μια τεχνική κλειδώματος που εξασφαλίζει την αξιοπιστία των δεδομένων. Όταν η πρόσβαση γίνεται μόνο για ανάγνωση, η κοινή μνήμη δεν θα μεταβληθεί και άρα οι πυρήνες έχουν ασφαλή πρόσβαση. Από την άλλη, η εγγραφή δεδομένων αλλάζει το περιεχόμενο της μνήμης και άρα η πρόσβαση πρέπει να είναι αποκλειστική στον πυρήνα που μεταβάλλει τα δεδομένα. Σε σύγκριση με το spinlock που θα εξεταστεί παρακάτω, ο συγκεκριμένος μηχανισμός κλειδώματος βελτιώνει την ανάγνωση σε ένα πολυπυρηνικό σύστημα. Κατά την διάρκεια εγγραφής δεδομένων οι υπόλοιποι πυρήνες που επιθυμούν να έχουν πρόσβαση στα συγκεκριμένα δεδομένα πρέπει να περιμένουν μέχρι την ολοκλήρωση της εγγραφής. Το ίδιο

ισχύει σε περίπτωση εγγραφής ενώ κάποιος πυρήνας διαβάζει τα δεδομένα. Οι πυρήνες που επιθυμούν να εγγράψουν πρέπει να αναμένουν να ολοκληρωθεί η ανάγνωση από τον αρχικό πυρήνα.

3.5.2 SPINLOCK

Ο Spinlock είναι ένας διαφορετικός μηχανισμός κλειδώματος που παραχωρεί την κλειδαριά στον πυρήνα που επιχειρεί ανάγνωση ή εγγραφή της μνήμης. Έπειτα ο μηχανισμός ελέγχει συνεχώς, εάν έχει απελευθερωθεί η κλειδαριά, μέχρι να είναι διαθέσιμη και να μπορεί να παραχωρηθεί σε άλλον πυρήνα. Πρόσβαση στα δεδομένα έχει μόνο ο πυρήνας με την κλειδαριά, κάτι που περιορίζει τις επιδόσεις του συστήματος, καθώς θα μπορούσαν πολλοί πυρήνες να διαβάσουν τα δεδομένα χωρίς να υπάρξει θέμα ακεραιότητας των δεδομένων.

Το dpdk διαθέτει υλοποιήσεις και για rwlock και spinlock για διαφορετικές χρήσεις.

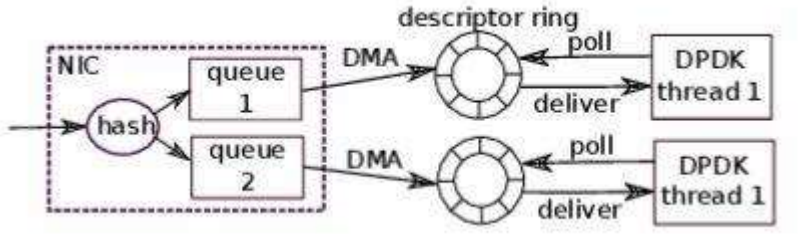
3.5.3 Lock-free

Καθώς η κύρια χρήση του dpdk είναι η γρήγορη επεξεργασία δικτυακής κίνησης στον χώρο χρήστη, η ύπαρξη μηχανισμών κλειδώματος δεν επιτρέπει στους πυρήνες που περιμένουν, να χρησιμοποιηθούν με τον βέλτιστο τρόπο. Ο μηχανισμός κλειδώματος επιτρέπει τον συγχρονισμό δεδομένων, αλλά υποχρεώνει του πυρήνες να περιμένουν έως ότου οι κοινοί πόροι είναι διαθέσιμοι. Για να επιτευχθούν υψηλές επιδόσεις, ιδανικά, πρέπει η κοινή χρήση δεδομένων να είναι η ελάχιστη δυνατή. Συχνά όμως αυτή είναι απαραίτητη, όπως και στην περίπτωση του DPDK, όπου χρησιμοποιείται ένας μηχανισμός χωρίς κλείδωμα (lock-free) για την βελτίωση των επιδόσεων του λογισμικού. Για να επιτευχθεί αυτό, το DPDK εισήγαγε μια βιβλιοθήκη δαχτυλιδιού, γνωστή ως rte_ring, ειδικά σχεδιασμένη για επεξεργασία φόρτου ροής πακέτων.

Συχνά η βιβλιοθήκη rte_ring χρησιμοποιείται σε συνδυασμό με την rte_mbuf (buffer πακέτων). Έτσι, η κοινή χρήση πακέτων από τους πυρήνες γίνεται μέσω του rte_ring, και οι δείκτες buffer είναι αντικείμενα στην rte_ring. Τα νήματα DPDK (πυρήνες) μπορούν να λειτουργούν ως παραγωγοί ή καταναλωτές, με τα πραγματικά πακέτα να παραμένουν στην ίδια μνήμη buffer. Καμία πραγματική αντιγραφή μνήμης για τα πακέτα δεν χρειάζεται, μόνο οι δείκτες αναφοράς στην μνήμη είναι σε κοινή χρήση στο rte_ring.

3.5.4 RTE_RING

Το rte_ring είναι μια απλή δομή δεδομένων που χρησιμοποιείται από το DPDK για κάθε RX/TX ουρά της NIC. Στην περίπτωση της λήψης πακέτων από την NIC οι περιγραφείς των πακέτων, τοποθετούνται στο δαχτυλίδι από τους παραγωγούς. Η τοποθέτηση των περιγραφέων και των δεδομένων των πακέτων γίνεται με DMA/DDIO όπου η NIC τοποθετεί τα πακέτα στην μνήμη/cache χωρίς να χρειάζεται αναμονή ο επεξεργαστής. Τα πακέτα εξέρχονται από το δαχτυλίδι από τον/τους πυρήνες καταναλωτές προς επεξεργασία.

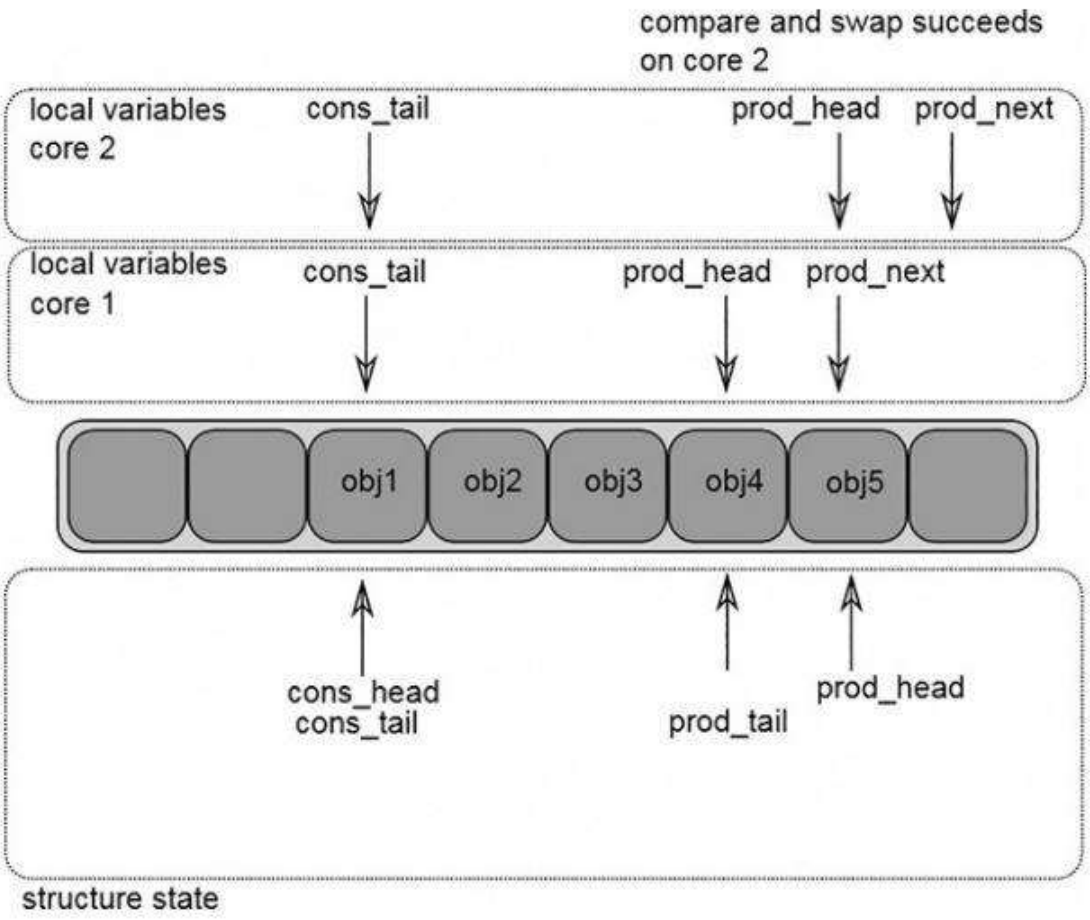


Εικόνα 14: Επικοινωνία της NIC και του polling πυρήνα μέσω RTE ring

Το δαχτυλίδι αποτελείται από τους περιγραφείς(descriptors) που έχουν αναφερθεί στο προηγούμενο κεφάλαιο, οι οποίοι μεταξύ άλλων περιέχουν την θέση των πακέτων(δεδομένων) στην μνήμη mempool υπό την μορφή mbuf. Ο αντίστοιχος για κάθε δαχτυλίδι πυρήνας κάνει Poll, δηλαδή δουλεύει στο 100% συνεχώς, ελέγχοντας για νέα πακέτα στο δαχτυλίδι (βλ. εικόνα 15).

3.5.5 RTE_RING και ευθυγράμμιση cache

Το rte_ring χρησιμοποιεί ευθυγράμμιση cache(cache alignment) μια τεχνική που τοποθετεί δεδομένα που χρησιμοποιούνται από διαφορετικούς πυρήνες σε διαφορετικά block στην μνήμη cache. Με αυτόν τον τρόπο αποφεύγεται η αναμονή πρόσβασης ενός πυρήνα στο block της cache, καθώς δεν υπάρχει άλλος πυρήνας που να επεξεργάζεται τα δεδομένα στο ίδιο block. Εφόσον το rte_ring είναι ένας lock-free μηχανισμός, η παραπάνω τεχνική είναι απαραίτητη για να μην υπάρχουν προβλήματα συγχρονισμού δεδομένων. Για την προσθήκη/αφαίρεση στοιχείων στο δαχτυλίδι χρησιμοποιούνται δείκτες παραγωγού/καταναλωτή για κάθε πυρήνα (βλ. εικόνα 16).



Εικόνα 15: Πρόσβαση στη δομή ring από πολλαπλούς πυρήνες

3.6 PMD

Προγενέστερα του DPDK, οι παραδοσιακοί NIC drivers έτρεχαν στο χώρο πυρήνα, και συνήθως χρησιμοποιούσαν διακοπές για RX/TX πακέτων. Μία από τις βασικές ιδέες του DPDK που το κάνει να ξεχωρίζει, είναι η υλοποίηση poll mode στον χώρο χρήστη, για να οδηγεί αποτελεσματικά την NIC σε υψηλές ταχύτητες. Με αυτήν την μέθοδο αφιερωμένοι πυρήνες μπορούν να επικεντρωθούν στην λήψη/αποστολή πακέτων.

3.6.1 Λειτουργία διακοπών

Για την κατανόηση της αξίας της χρήσης PMD έναντι λειτουργίας διακοπών πρέπει να αναδειχθούν οι καθυστερήσεις που επιφέρει η λειτουργία διακοπών σε ένα σύστημα κατά την άφιξη/αποστολή πακέτων.

Όταν ένα πακέτο εισέρχεται στην NIC, οδηγείται σε μία RX ουρά. Η NIC παράγει σήμα διακοπής, η οποία διακοπή οδηγείται στον επεξεργαστή και έπειτα ένας χειριστής υπηρεσιών διακοπών είναι υπεύθυνος για την λήψη του πακέτου. Η λειτουργία αυτή έχει το κόστος της αλλαγής περιεχομένου (context switch), το οποίο έμμεσο κόστος ισχύει για κάθε υπηρεσία διακοπών. Όταν ο επεξεργαστής δουλεύει σε συχνότητες αρκετά μεγαλύτερες της συχνότητας της συσκευής εισόδου εξόδου, το κόστος αυτό , είναι διαχειρίσιμο. Ο επεξεργαστής, λόγω της ασύγχρονης φύσης των διακοπών, μπορεί να χρησιμοποιείται για άλλες εργασίες, όσο δεν υπάρχουν πολλά πακέτα στο σύστημα.

Σε NIC υψηλών ταχυτήτων όμως, μία έκρηξη ροής πακέτων οδηγεί σε πολλές διακοπές. Η NIC και ο driver λογισμικού έχουν την δυνατότητα να παράγουν διακοπές για ομάδες πακέτων μειώνοντας το έμμεσο κόστος της διακοπής. Σε πολύ υψηλές ταχύτητες ούτε η παραπάνω λύση επαρκεί.

Αντίστοιχα, κατά την αποστολή ενός πακέτου, ο NIC driver ετοιμάζει την μνήμη πακέτων και τον προσδιοριστή NIC στην ουρά TX χρησιμοποιώντας τον buffer προσδιοριστή. Όταν η αποστολή του πακέτου ολοκληρωθεί, η NIC επίσης παράγει διακοπή υλικού, και ο επεξεργαστής θα χρησιμοποιήσει την διακοπή για να ολοκληρώσει τις εναπομείνασες εργασίες μετά το TX του πακέτου, όπως απελευθέρωση μνήμης buffer. Η αποστολή πακέτων έχει παρόμοιο έμμεσο κόστος κατά την αλλαγή περιεχομένου. Είναι φανερό ότι η μέθοδος διακοπών είναι αποτελεσματική μόνο εάν δεν υπάρχει μεγάλη απαίτηση σε RX/TX πακέτων.

3.6.2 Λειτουργία Poll

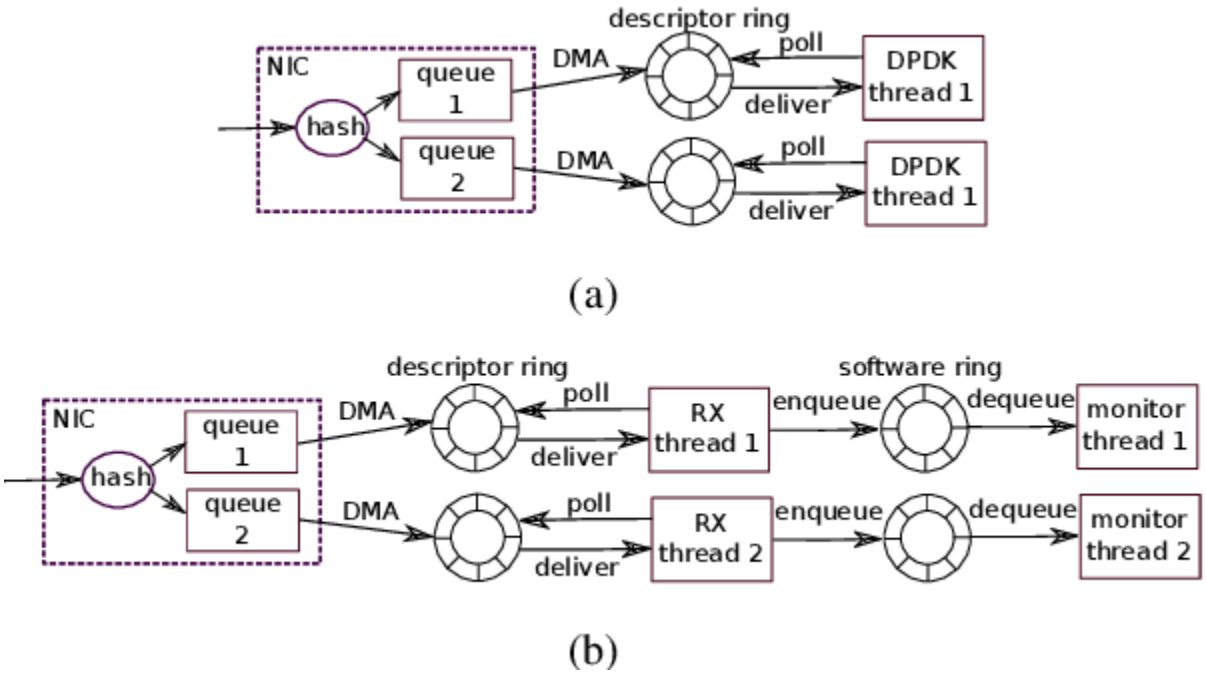
Στο DPDK η λειτουργία Poll αφορά την χρήση αφοσιωμένων πυρήνων για επεξεργασία πακέτων όπου ο πυρήνας τρέχει σε λειτουργία Poll υψηλών απαιτήσεων, χωρίς να χρησιμοποιεί τον μηχανισμό των διακοπών. Η μέθοδος αυτή είναι πιο αποτελεσματική σε σενάριο υψηλών ταχυτήτων. Χωρίς την χρησιμοποίηση διακοπών, γλιτώνει το κόστος αλλαγής περιεχομένου.

Όποιο πακέτο εισέρχεται στην NIC περνά από έλεγχο ακεραιότητας, ανάλυση επικεφαλίδων, φιλτράρισμα πακέτων, προτού τελικά τοποθετηθεί σε μία RX ουρά. Γενικά, κάθε ουρά RX είναι αλληλένδετη με μία ουρά λογισμικού με στοιχεία mbuf, όπως αναφέρθηκε στην ενότητα του RTE δαχτυλιδιού. Ο PMD(poll mode driver) είναι υπεύθυνος για την αρχικοποίηση και οργάνωση των θηρών/ουρών της NIC. Η διαδικασία αυτή, εμπεριέχει τη συμπλήρωση των στοιχείων mbuf στον προσδιοριστή υλικού και την ενημέρωση της NIC να τοποθετήσει το εισερχόμενο πακέτο στην

ορισμένη διεύθυνση μνήμης. Τα metadata του πακέτου ενημερώνονται στον προσδιοριστή πακέτων.

3.6.3 RTC, pipeline και PMD

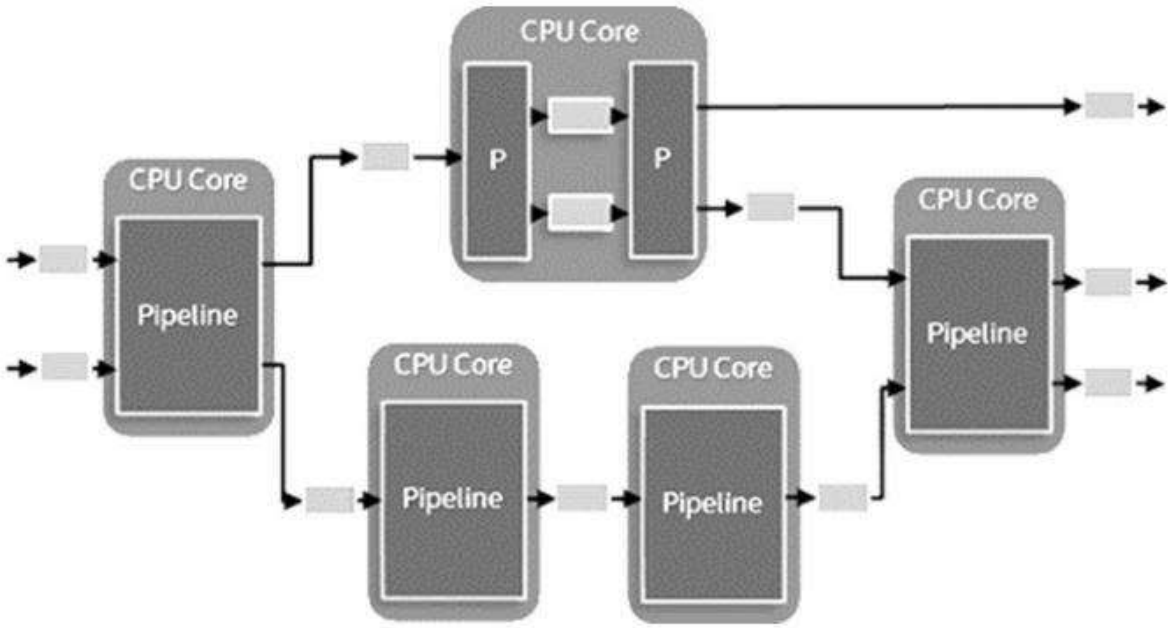
Σύμφωνα με την φιλοσοφία του DPDK, κάθε πακέτο, για όλη την διάρκεια της διαδικασίας RX/TX, το διαχειρίζεται ο ίδιος πυρήνας, γνωστό και ως RTC μοντέλο (βλ. εικόνα 17α). Όταν το πακέτο εισέλθει στο σύστημα, η επεξεργασία του λαμβάνεται πια από το λογισμικό, που μπορεί να εφαρμόσει pipeline μοντέλο για την σύνθετη επεξεργασία των πακέτων.[23]



Εικόνα 16: (α) RTC μοντέλο (β) Pipeline μοντέλο

Στην περίπτωση του pipeline λογισμικού, μεταξύ των σταδίων χρειάζονται ουρές. Αυτές, στο DPDK υλοποιούνται επίσης με την λογική δαχτυλιδιού και ονομάζονται δαχτυλίδια λογισμικού(sw rings) (βλ. εικόνα 17β). Κάθε στάδιο του pipeline το αναλαμβάνει ένας πυρήνας του επεξεργαστή (βλ. εικόνα 18)

[23] Intel, 'Poll Mode Driver', *dpdk docs* [ιστοσελίδα], https://dpdk.readthedocs.io/en/v2.1.0/prog_guide/poll_mode_drv.html , (τελευταία πρόσβαση 29 Ιανουαρίου 2023)

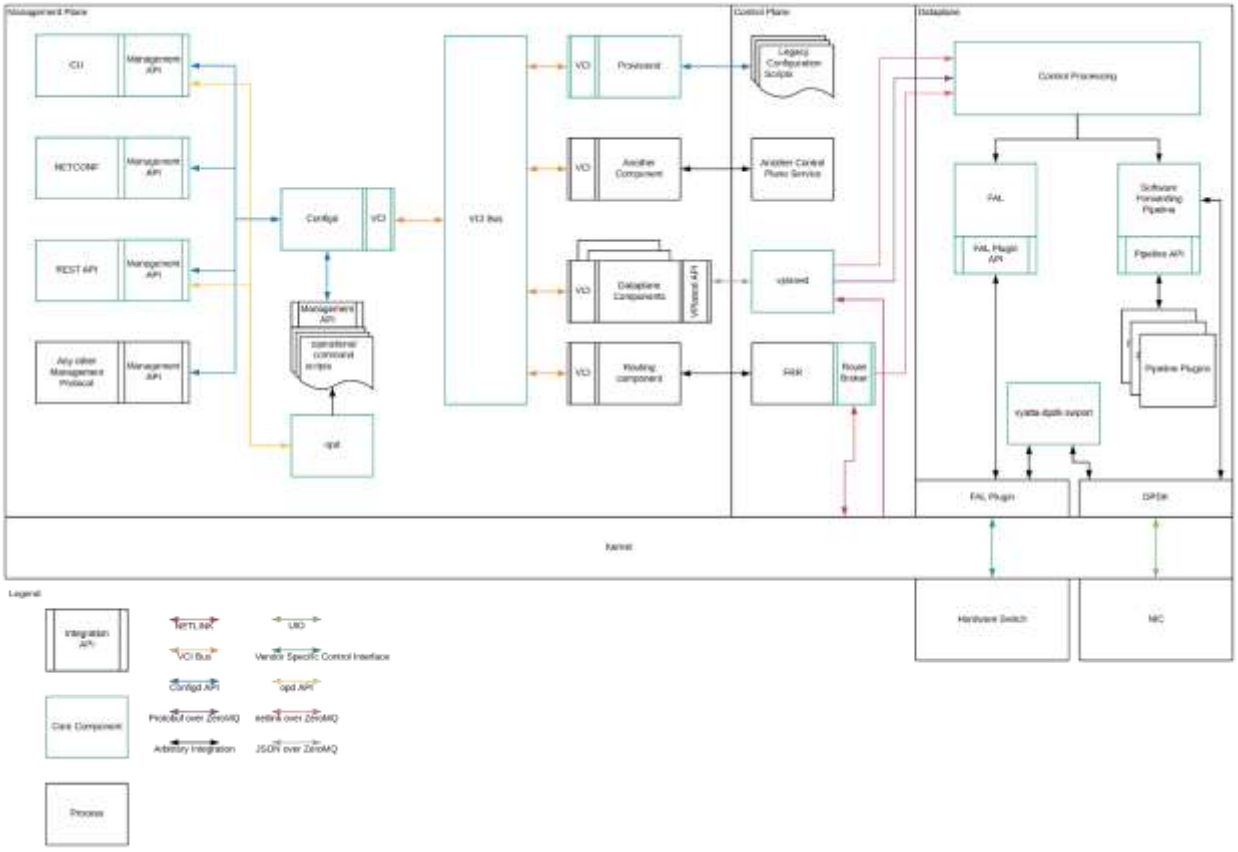


Εικόνα 17: Παράδειγμα σταδίων στο pipeline λογισμικό

Κεφάλαιο 4 : Αρχιτεκτονική Danos

4.1 Εισαγωγή-Βασικά Στοιχεία

Το DANOS αποτελείται από αρκετές υπηρεσίες που βρίσκονται στα επίπεδα διαχείρισης(management), ελέγχου(control) και δεδομένων(data) του δρομολογητή.[24] Στην εικόνα 19 φαίνεται σχηματικά η δομή των βασικών χαρακτηριστικών και επιπέδων του DANOS.



Εικόνα 18: Βασική Αρχιτεκτονική του DANOS

Τα βασικά συστατικά του DANOS αντιπροσωπεύουν τις απαραίτητες δομές και API για το DANOS, την ταυτότητα του που το ξεχωρίζει από άλλα συστήματα. Κάποια από αυτά χρησιμοποιούνται κατά κόρον στην διπλωματική αυτή και χρειάζεται να επισημανθούν.

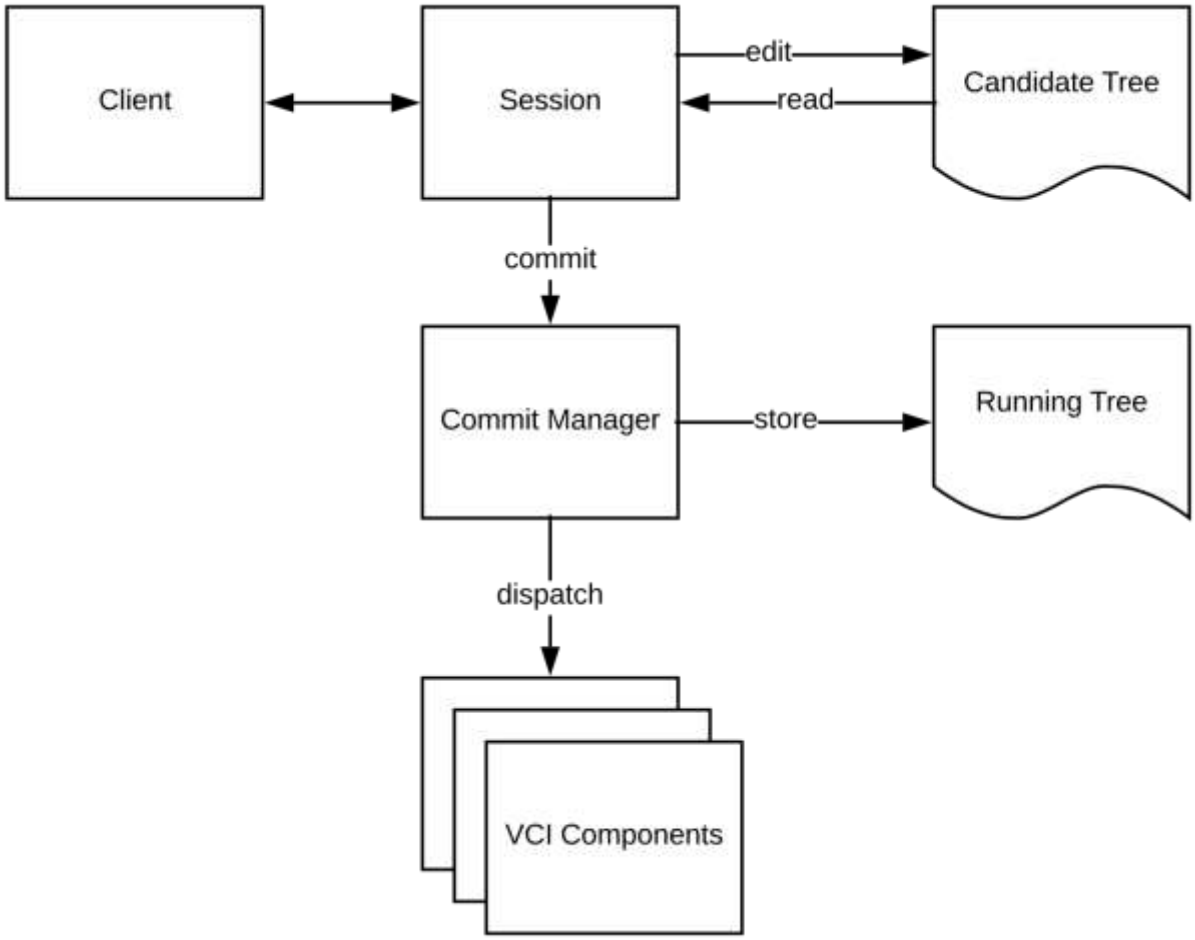
[24] J. Southworth, 'Architectural Overview', *danos project* [ιστοσελίδα], 10 Νοεμβρίου 2020, https://danosproject.atlassian.net/wiki/spaces/DAN/pages/684457996/Architectural+Overview, (τελευταία πρόσβαση 29 Ιανουαρίου 2023)

4.2 Επίπεδο Διαχείρισης

Το επίπεδο διαχείρισης αποτελείται από δομές παραμετροποίησης και λειτουργικότητας. Στα πλαίσια της διπλωματικής δοκιμάστηκαν πολλές παραμετροποιήσεις και ελέγχθηκαν από τους μηχανισμούς του DANOS. Για λανθασμένες παραμετροποιήσεις το DANOS παρέχει ακριβείς πληροφορίες, που βοήθησαν πολύ για την εύκολη και γρήγορη υλοποίηση της παραμετροποίησης.

4.2.1 Διαχείριση παραμετροποίησης

Η πρόσβαση και διαχείριση της παραμετροποίησης του DANOS γίνεται από την υπηρεσία configd. Ο χειρισμός της παραμετροποίησης βασίζεται στην ιδέα του "session". Όταν ένας χρήστης επιθυμεί να χειριστεί την παραμετροποίηση του συστήματος, δημιουργείται ένα "session" στο configd με συγκεκριμένο όνομα. Αυτό το όνομα πρέπει να είναι μοναδικό και διατηρεί συνδέσεις με το configd. Σε αυτό το "session" ο χρήστης λαμβάνει ένα ιδιωτικό "υποψήφιο" δέντρο παραμετροποίησης που μπορεί να χειριστεί όπως αυτός επιθυμεί. Όταν θεωρήσει ο χρήστης ότι το "υποψήφιο" δέντρο αντικατοπτρίζει τις αλλαγές που επιθυμεί να γίνουν, τότε μπορεί να επιβεβαιώσει αυτές τις αλλαγές (commit). Όταν γίνει η επιβεβαίωση αυτή, όλα τα "υποψήφια" δέντρα που υπάρχουν σε διαφορετικά "sessions" βασίζονται πια στο νέο δέντρο που έχει υποβληθεί λαμβάνοντας όλες τις αλλαγές που αυτό επιβάλλει. Αντίθετα η πρόσβαση στην παραμετροποίηση δεν χρειάζεται τη δημιουργία κάποιου "session" και δίνεται πρόσβαση στα δεδομένα από το "τρέχον" δέντρο. Ένα εικονικό (dummy) session υπάρχει για να διατηρεί την σύνδεση στο δέντρο συνεχή με πρόσβαση σε "υποψήφια" δέντρα. Το "τρέχον" δέντρο μπορεί να μεταβληθεί μόνο μέσω της λειτουργίας του commit. Στην εικόνα 20 φαίνεται η παραπάνω δομή.



Εικόνα 19: Λειτουργία παραμετροποίησης στο DANOS

4.2.2 Σημασιολογία του commit

Το μοντέλο παραμετροποίησης στο DANOS είναι ότι αντιμετωπίζεται ως η επιθυμητή κατάσταση συστήματος από τον χρήστη. Έπειτα, τα μέρη του VCI ενεργούν ως πράκτορες για τα διάφορα μέρη της παραμετροποίησης προσπαθώντας να πραγματοποιήσουν τις απαραίτητες αλλαγές, όσο το σύστημα εξελίσσεται με την πάροδο του χρόνου. Για αυτούς τους λόγους λοιπόν, η παραμετροποίηση πρέπει να είναι εσωτερικά συνεπής και όχι να βασίζεται στην τωρινή κατάσταση του συστήματος για να καθορίσει εάν είναι έγκυρη. Η επιβεβαίωση αλλαγών στον DANOS αφορά μια διαδικασία δύο σταδίων:

1. Οι αλλαγές ελέγχονται για την εγκυρότητα τους σύμφωνα με τους περιορισμούς στο data-model και οποιονδήποτε επιπλέον στατικό έλεγχο υλοποιήσει το μέρος του VCI που διαχειρίζεται το συγκεκριμένο κομμάτι της παραμετροποίησης

2. Οι αλλαγές αποστέλλονται σε μέρη που αναλαμβάνουν την συμφιλίωση της νέας επιθυμητής παραμετροποίησης με την κατάσταση του συστήματος. Αν δεν είναι δυνατή η συμφιλίωση όλων των αλλαγών που δίνονται στο σύστημα εκείνη τη στιγμή, τότε οι αλλαγές αναβάλλονται, έως ότου αλλάξουν οι συνθήκες.

4.3 Επίπεδο Ελέγχου

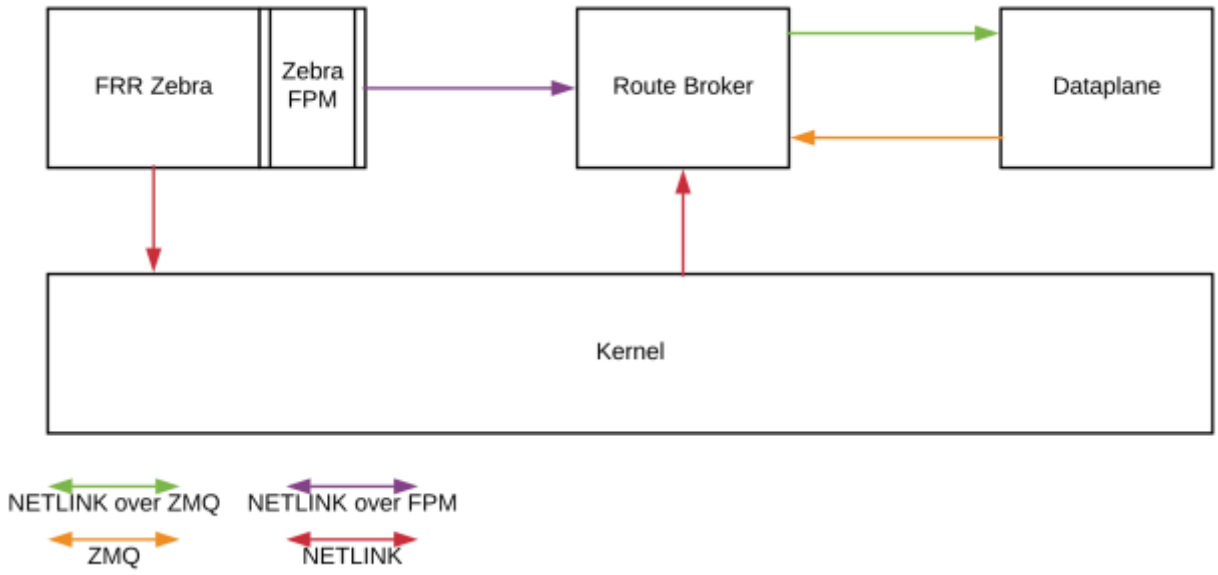
Το επίπεδο Ελέγχου αποτελείται από δύο βασικά στοιχεία της δομής του DANOS, τα route broker και vplaned. Συνιστούν βασικά στοιχεία και για την κατανόηση της λειτουργίας του DANOS, καθώς υλοποιούν επικοινωνία μεταξύ του Kernel και dataplane για βασικές λειτουργίες, όπως δρομολόγηση πακέτων και παραμετροποίηση.

4.3.1 ROUTE-BROKER

Ο Route Broker είναι ένας daemon που εξασφαλίζει την συνοχή μεταξύ των FIB στο επίπεδο ελέγχου, στο kernel και στο dataplane. Έχει τη δυνατότητα να γνωρίζει ποια πληροφορία θα δοθεί σε κάθε ένα από τα παραπάνω μέρη. Χρησιμοποιεί pull μοντέλο για να τραβά πληροφορίες από τα υπόλοιπα μέρη. Στο DANOS ο Route Broker είναι το περίβλημα της Route Broker βιβλιοθήκης που συνδέεται με την FRR Zebra Fib Push Interface[25]. Επειδή το FRR έχει ήδη μια διεπαφή με το dataplane για να προωθεί διαδρομές (την zebra fib push interface), το DANOS χρησιμοποιεί το Route Broker ως έναν μετασχηματιστή μεταξύ αυτής της διεπαφής και της δικής του route broker βιβλιοθήκης. Η FRR Zebra προωθεί ενημερώσεις διαδρομών, μέσω NETLINK[26] μηνυμάτων, προς τον Route Broker. Ταυτόχρονα ο Route Broker επικοινωνεί και με τον Kernel για να ενημερώνεται για kernel διαδρομές ξανά μέσω NETLINK. Για να επιτευχθεί αυτό, ο Router Broker daemon κάνει polling στο NETLINK socket και στο Zebra FPM socket για ενημερώσεις διαδρομών και έπειτα τις περνά μέσα από την Route Broker βιβλιοθήκη προς διανομή για το dataplane. Η εικόνα 21 δείχνει ακριβώς, πως ο Route Broker αλληλεπιδρά με το Zebra και το dataplane.

[25] FRR, 'zebra FIB push interface', *frrouting docs* [ιστοσελίδα], http://docs.frrouting.org/en/latest/zebra.html#zebra-fib-push-interface, (τελευταία πρόσβαση 29 Ιανουαρίου 2023)

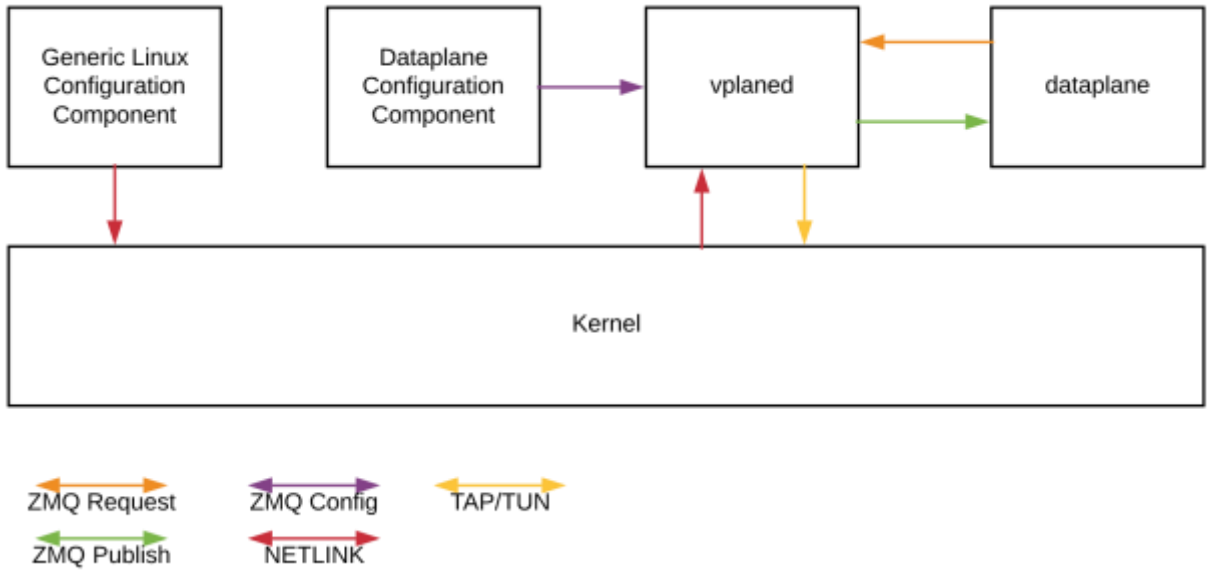
[26] Web archive, 'An Overview of netlink sockets', *web archive* [ιστοσελίδα], 3 October 1999, https://web.archive.org/web/20170430074145/http://qos.ittc.ku.edu/netlink/html/node4.html, (τελευταία πρόσβαση 29 Ιανουαρίου 2023)



Εικόνα 20: Επικοινωνία μεταξύ των επιμέρους πινάκων δρομολόγησης

4.3.2 Vplaned

Το Vplaned είναι ένας daemon που διαχειρίζεται την αλληλεπίδραση με το dataplane, λειτουργεί ως σύνδεσμος μεταξύ διαχειριστικού επιπέδου και υπηρεσιών επιπέδου ελέγχου και επιπέδου δεδομένων (βλ. εικόνα 22). Παρέχει ένα ενδιάμεσο στάδιο επεξεργασίας για εντολές netlink state και εντολές παραμετροποίησης. Στο Vplaned αποθηκεύονται netlink state και εντολές παραμετροποίησης, οι οποίες πληροφορίες μπορούν να ζητηθούν από το dataplane με μορφή dump των cached πληροφοριών, για να μπορέσει το dataplane να επαναφέρει τις προηγούμενες παραμέτρους συστήματος σε ενδεχόμενη επανεκκίνηση. Είναι σημαντικό να σημειωθεί ότι το Vplaned δεν κατανοεί τις πληροφορίες που έχει αποθηκεύσει, λειτουργεί απλώς ως ένα ενδιάμεσο στάδιο για να βοηθήσει στη διαχείριση του dataplane.

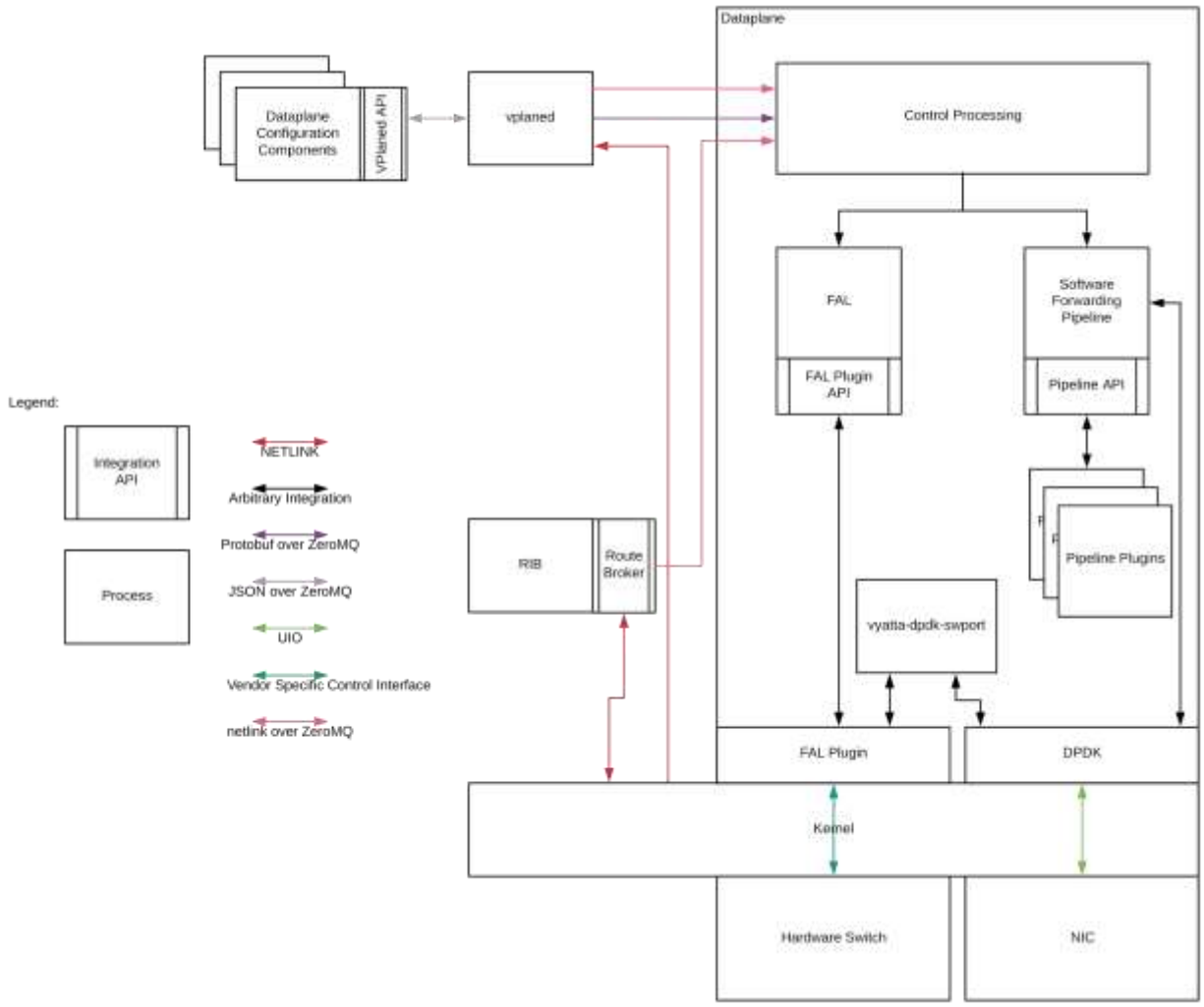


Εικόνα 21: Επικοινωνία του Kernel με το dataplane μέσω του vplaned

Μια εξαιρετικά σημαντική λειτουργία του Vplaned είναι η δημιουργία των shadow devices, συσκευές στον Linux kernel σε αντιστοιχία με τις συσκευές στο dataplane. Μέσω αυτών των συσκευών, το dataplane διαχειρίζεται την αποστολή και λήψη πακέτων από και προς το επίπεδο ελέγχου.

4.4 Επίπεδο Δεδομένων

Το dataplane είναι μια εφαρμογή FIB, αφού λαμβάνει καταστάσεις προωθητικού χαρακτήρα από το υπόλοιπο σύστημα. Ανάλογα με την κατάσταση που λαμβάνει, μπορεί να χρησιμοποιηθεί για την κατασκευή προωθητικού αγωγού για λογισμικό, υλικό ή και τα δύο (software/hardware forwarding pipeline). Το dataplane χρησιμοποιεί το DPDK για να οδηγεί NIC χαρακτήρα υλικό και το FAL για να οδηγεί switching silicon. Τα δύο αυτά μέρη μπορούν να συνδυαστούν για να δημιουργήσουν μια πλατφόρμα βασισμένη σε switch-silicon για ένα ταχύ μονοπάτι προώθησης για χαρακτηριστικά που δεν υποστηρίζονται στη σιλικόνη. Στην εικόνα 23 παρουσιάζεται η δομή των επιμέρους μερών του dataplane και η επικοινωνία τους με τα μέρη ελέγχου και διαχείρισης.



Εικόνα 22: Δομή του επιπέδου δεδομένων στο DANOS και API αφαίρεσης προς το υλικό

Το dataplane αλληλεπιδρά με το υπόλοιπο DANOS μέσω του vplaned και του route broker. Έχει τρία σημαντικά λειτουργικά μέρη: το "main thread" που εκτελεί έλεγχο επεξεργασίας (control processing) από εξωτερικές διεπαφές, τον αγωγό που υλοποιεί προώθηση λογισμικού σε υψηλές ταχύτητες και το FAL που διαχειρίζεται υλικό switching silicon.

4.4.1 Έλεγχος επεξεργασίας

Ο έλεγχος επεξεργασίας του dataplane (control processing, επίσης γνωστό ως "main thread") είναι υπεύθυνο για το χειρισμό της αλληλεπίδρασης του vplaned και του route broker. Χειρίζεται τόσο επεξεργασία μηνυμάτων που έχουν προωθηθεί ως proxy από το vplaned όσο και ενημερώσεις διαδρομών που στέλνονται από το route broker.

4.4.2 Υποδομή προώθησης λογισμικού

Το dataplane είναι υλοποιημένο πάνω από το DPDK, επιτρέποντας γρήγορη, επεκτάσιμη επεξεργασία πακέτων από το λογισμικό. Το DPDK χρησιμοποιείται ως το γενικό επίπεδο αφαίρεσης υλικού στο επίπεδο δεδομένων. Ακόμα, το dataplane βασίζεται σε μεγάλο βαθμό στο RCU (read copy update) επιπέδου χρήστη για να παρέχει επεκτάσιμες lock-free δομές δεδομένων, όταν δεδομένα πρέπει να μοιραστούν μεταξύ των πυρήνων. Εκτός από το "main thread", υπάρχουν νήματα προώθησης που έχουν ανατεθεί στους lcores στους οποίους το dataplane είναι υλοποιημένο να τρέχει. Το lcore αναφέρεται σε μια λογική μονάδα εκτέλεσης στον επεξεργαστή, γνωστό και ως νήμα υλικού (hardware thread). Υπάρχει ακριβώς ένα νήμα προώθησης για κάθε lcore προώθησης. Αν υπάρχουν πολλές ουρές που έχουν ανατεθεί σε έναν lcore, τότε επεξεργάζονται με round-robin μορφή. Αυτό ακολουθεί την φιλοσοφία της αρχιτεκτονικής του DPDK και με την προσπάθεια να αποφευχθεί μεγάλη χρησιμοποίηση του πυρήνα (cpu overhead) που σχετίζεται με την αλλαγή περιεχομένου μεταξύ νημάτων. Για λόγους επιδόσεων, το επίπεδο δεδομένων δεν κάνει lock ούτε στις RX ή TX ανά-ουρά δομές δεδομένων, αλλά επιτρέπει σε πολλαπλές ουρές RX ή TX να εξυπηρετούνται από την ίδια θήρα (port) ταυτόχρονα. Έτσι το επίπεδο δεδομένων αναθέτει ουρές RX, έτσι ώστε να μπορούν να χρησιμοποιηθούν από έναν μόνο lcore(ένας λογικός πυρήνας μπορεί να αποτελείται από έναν φυσικό πυρήνα είτε από δύο λογικούς CPUs, δηλαδή HyperThreads) και αντίστοιχα ουρές TX, έτσι ώστε να μπορούν να χρησιμοποιηθούν από έναν μόνο lcore.

4.4.3 Pipeline

Η υλοποίηση του DANOS σύμφωνα με τις αρχές του dpdk βασίζεται στο RTC μοντέλο. Στο DANOS υπάρχει η δυνατότητα χωρισμού της επεξεργασίας σε ξεχωριστά στάδια, με αυστηρά ορισμένες εισόδους/εξόδους. Ο λόγος είναι, για να είναι δυνατός ο ορισμός αυτών των σταδίων κατά το τρέξιμο ή κατά την μεταγλώττιση. Τα στάδια του pipeline μπορούν να γραφούν μέσω του ειδικού pipeline API.

4.4.4 Επίπεδο αφαίρεσης προώθησης (FAL)

Παρόλο που στα πλαίσια της διπλωματικής δεν βρίσκει εφαρμογή, καθώς ο server διαθέτει NIC και όχι hardware switch και ως επίπεδο αφαίρεσης χρησιμοποιείται το DPDK, είναι ένα πολύ χρήσιμο χαρακτηριστικό για ανάγκες προγραμματισμού σε μηχανήματα που διαθέτουν hardware

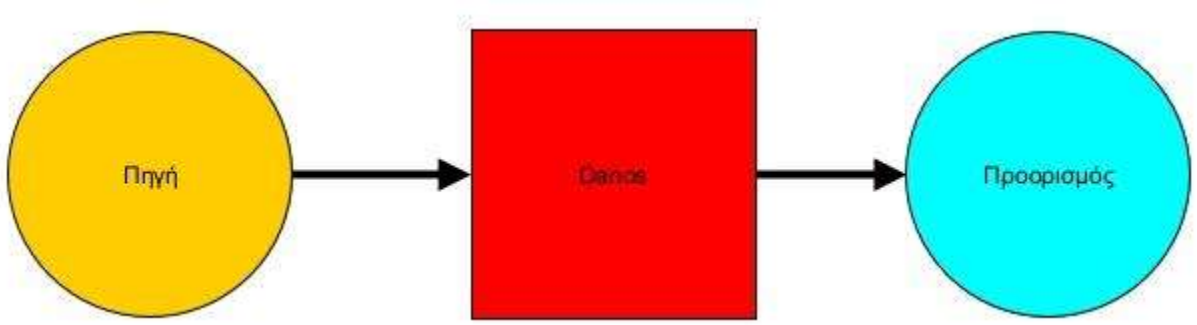
switch. Το FAL είναι το σημείο σύνδεσης με switch συσκευές υλικού. Παρέχει μία σειρά από API για να προγραμματίσει το dataplane τον switch υλικού.

Κεφάλαιο 5 : Τοπολογία και configuration

5.1 Εισαγωγή

Στο παρόν κεφάλαιο παρουσιάζεται η τοπολογία πάνω στην οποία γίνονται δοκιμές για να εξακριβωθεί πώς ανταποκρίνεται το DANOS σε μεγάλες ταχύτητες άνω των 10 Gbps. Χρησιμοποιώντας τα διαθέσιμα υλικά και αξιοποιώντας τις διαθέσιμες τεχνολογίες και παραμέτρους που παρέχουν αυτά, από κίνηση τάξεως μεγέθους του 1Gbps με συγκεκριμένες τεχνικές επιτυγχάνεται εν τέλει η δημιουργία κίνησης της τάξεως των 60Gbps και έτσι δοκιμάζεται το DANOS σε μεγάλες ταχύτητες. Οι παρακάτω τεχνικές αποτελούν μια πρακτική μέθοδο για να γίνονται δοκιμές κίνησης σε πολύ μεγάλες ταχύτητες, χωρίς να υπάρχει δυνατότητα παραγωγής πραγματικής κίνησης από πηγές, χρησιμοποιώντας έτσι ελάχιστους πόρους και υλικό.

5.2 Αρχική ιδέα για την τοπολογία



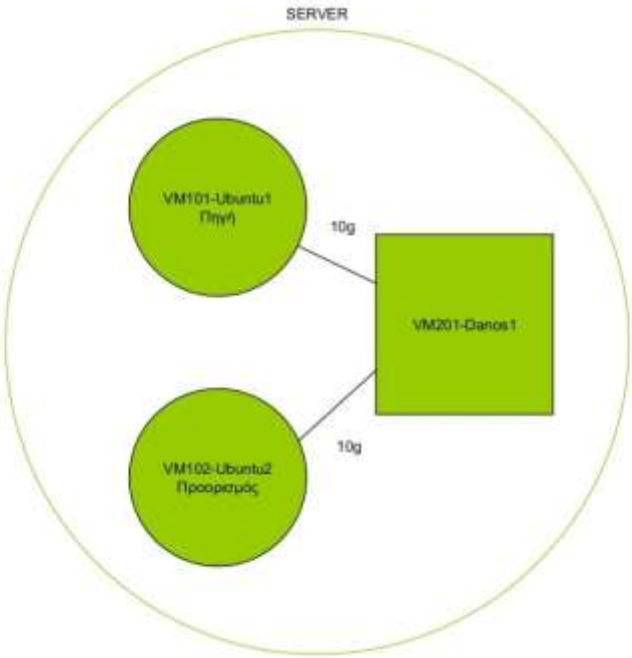
Εικόνα 23: Αρχική ιδέα για την τοπολογία

Με σκοπό να δοκιμαστεί το DANOS σε υψηλές ταχύτητες, πρέπει να κατασκευαστεί μία τοπολογία με μία πηγή και έναν προορισμό, συνδεδεμένα με το DANOS, έτσι ώστε το DANOS να προωθεί την κίνηση από την πηγή στον προορισμό. Καθώς δεν υπήρχε διαθεσιμότητα για τρία διαφορετικά φυσικά μηχανήματα, επιλέχθηκε να φιλοξενηθούν σε έναν server, ως εικονικές μηχανές.

5.3 Εικονική τοπολογία των Vm στον Server και Proxmox

Στον server επιλέχθηκε το proxmox ως λογισμικό που ενσωματώνει hypervisor εικονικών μηχανών, επειδή παρέχει την δυνατότητα δημιουργίας virtual συνδέσεων μεταξύ VMs με την

επιπλέον δυνατότητα να εκχωρεί NIC (κάρτες δικτύου) αποκλειστικά στα VMs (PCIe Passthrough).



Εικόνα 24: Τοπολογία εικονικών μηχανών

Στην συγκεκριμένη περίπτωση το λογισμικό Proxmox χρησιμοποιείται για να δημιουργηθούν και να φιλοξενηθούν: α) τα τρία VMs που αφορούν την τοπολογία και β) ένα επιπλέον VM που χρησιμοποιείται ως grafana server λαμβάνοντας δεδομένα κίνησης μέσω του πρωτοκόλλου snmp με τρόπο που θα εξηγηθεί στην ενότητα των δοκιμών. Αναφορικά με τα τρία προαναφερθέντα VMs, τα δύο από αυτά είναι VM τύπου (O/S) Lubuntu, που υλοποιούν την λειτουργία του source και dump και το τρίτο VM είναι το DANOS.

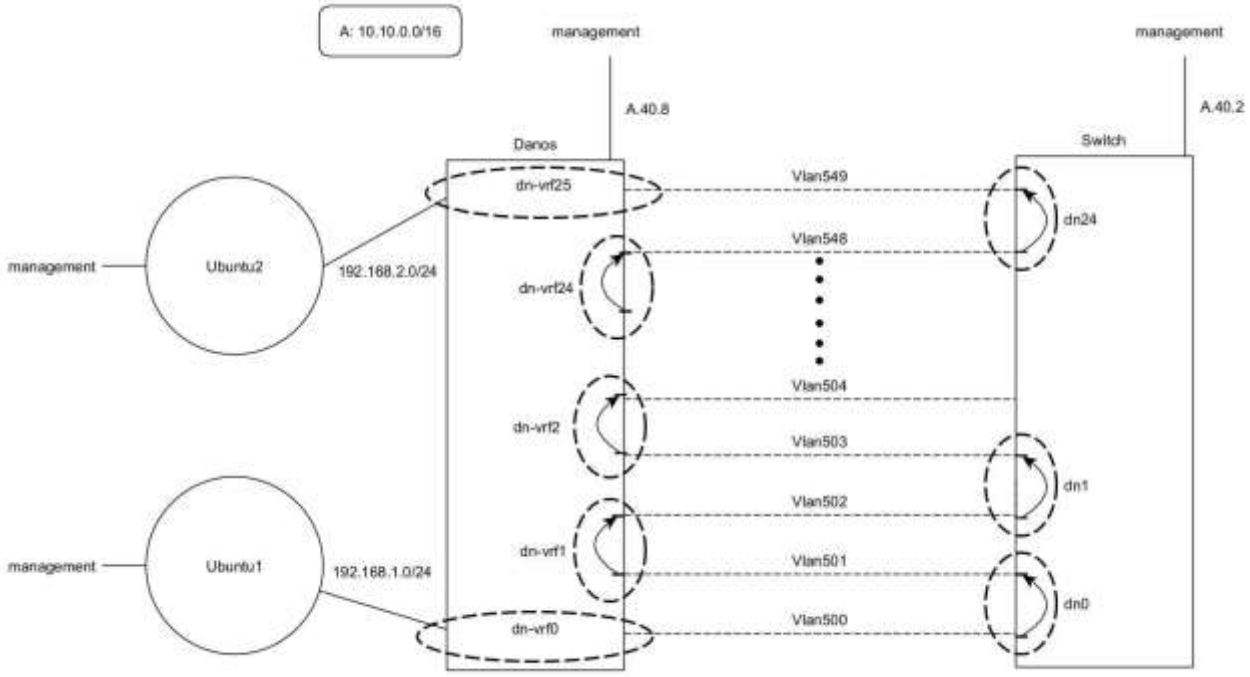
Τα δύο Ubuntu μηχανήματα συνδέονται με το DANOS, το Ubuntu1 χρησιμοποιείται ως η πηγή της ροής πακέτων και το μηχάνημα Ubuntu2 ως ο προορισμός της ροής πακέτων. Το proxmox παρέχει την δυνατότητα δημιουργίας virtual συνδέσεων μεταξύ των μηχανημάτων καθώς και PCIe Passthrough, τα οποία χρησιμοποιούνται και τα δύο για την συνδεσμολογία. Πιο αναλυτικά οι συνδέσεις Ubuntu1-DANOS και Ubuntu2-DANOS αφορούν Linux Bridge.

5.4 Πρόβλημα παραγωγής κίνησης μεγαλύτερης των 20Gbps

Αρχικά, για τις συνδέσεις μεταξύ DANOS και Ubuntu μηχανημάτων είχε επιλεχθεί ο driver e1000 που επέτρεπε έως 1Gbps κίνηση. Είναι φανερό ότι 1Gbps κίνηση δεν αρκεί για την δοκιμή του DANOS, καθώς χρειάζεται να δοκιμαστεί σε ταχύτητες άνω των 20Gbps. Κατά τις δοκιμές που θα αναλυθούν σε επόμενο κεφάλαιο βρέθηκε προβληματικός ο driver e1000 και αντικαταστάθηκε από τον Virtio που παρέχει το proxmox με ταχύτητες έως 10Gbps. Τα 10Gbps αποτελούν σαφή βελτίωση αλλά ακόμα δεν είναι αρκετή για τις δοκιμές που πρέπει να γίνουν στο DANOS.

5.5 Τεχνική ανακύκλωσης για παραγωγή ταχυτήτων άνω των 20Gbps

Μια ενδιαφέρουσα λύση στο πρόβλημα της παραγωγής κίνησης τέτοιας τάξης μεγέθους είναι ο πολλαπλασιασμός της αρχικής κίνησης μέσω ανακυκλώσεων. Για να επιτευχθεί αυτό, εκτός από τις τρεις εικονικές μηχανές, χρειάζεται ένα φυσικό μηχάνημα συνδεδεμένο με το DANOS, έτσι ώστε να μπορούν να ανακυκλώνουν μεταξύ τους την παραγόμενη από την πηγή κίνηση, έως ότου αυτή καταλήξει τελικά στον προορισμό.



Εικόνα 25: Τεχνική Ανακύκλωσης πάνω στην τοπολογία

Η ανακύκλωση αυτή υλοποιείται συνδέοντας το DANOS με το επιπλέον μηχάνημα και ανακατευθύνοντας την κίνηση συνεχώς μεταξύ του DANOS και του δεύτερου μηχανήματος. Προς τον σκοπό αυτό το δεύτερο μηχάνημα που χρησιμοποιήθηκε ήταν ένας switch.

5.6 Πολλαπλασιασμός κίνησης μέσω χρήσης VRFs

Σκοπός είναι να ανακυκλωθεί η παραγόμενη κίνηση από την πηγή (Ubuntu1), έτσι ώστε κάθε στιγμή το DANOS να επεξεργάζεται κίνηση άνω των 20Gbps. Προς αυτόν τον σκοπό χρησιμοποιείται μια τεχνική ανακύκλωσης όπου η ροή ανακυκλώνεται συνεχώς μεταξύ των DANOS και Switch, μέχρι η γραμμή να φτάσει την επιθυμητή ταχύτητα. Η ροή, έπειτα από συγκεκριμένο αριθμό ανακυκλώσεων, καταλήγει τελικά στον αποδέκτη, δηλαδή το δεύτερο μηχάνημα Ubuntu. Τα πακέτα αντίθετης ροής που έρχονται ως απάντηση στην κίνηση ακολουθούν την αντίστροφη διαδρομή.

Είναι φανερό ότι τα πακέτα που ανακυκλώνονται έχουν πάντα τις ίδιες διευθύνσεις πηγής και προορισμού, καθώς δεν υπόκεινται σε μεταφράσεις. Επίσης, σε κάθε δρομολογητή, ο πίνακας διαδρομών για κάθε υποδίκτυο προορισμού έχει μία ενεργή διαδρομή. Φαίνεται λοιπόν ότι το ίδιο πακέτο δεν μπορεί να δρομολογηθεί σε δύο διαφορετικούς προορισμούς από τον ίδιο δρομολογητή και κάνει αδύνατη την ανακύκλωση. Για αυτό το σκοπό χρησιμοποιούνται VRFs, υπο-πίνακες δρομολόγησης, σε συνδυασμό με VLANs για να είναι δυνατή η κατά περίπτωση δρομολόγηση.

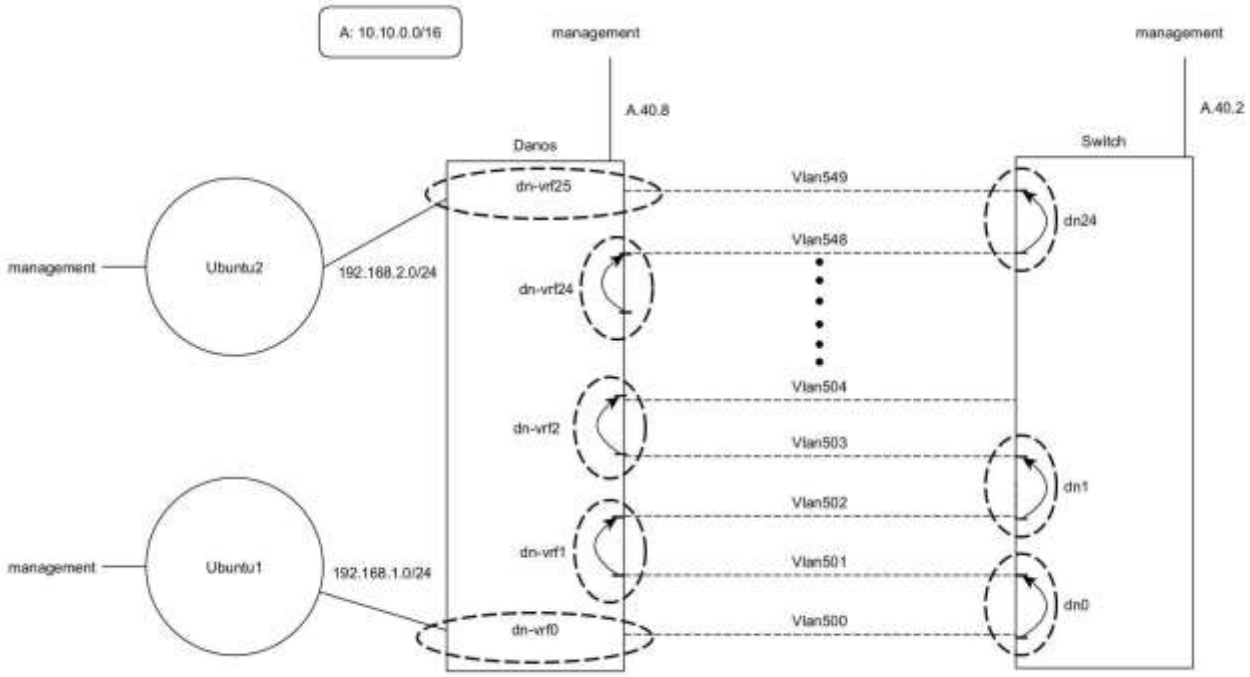
5.6.1 VRF-lite

Τα VRFs[27] αποτελούν μια τεχνολογία που αρχικά χρησιμοποιήθηκε από παρόχους διαδικτύου που χρησιμοποιούν το πρωτόκολλο MPLS για να υπάρχει η δυνατότητα μοιράσματος των ίδιων διευθύνσεων IP σε υποδίκτυα. Αυτό συμβαίνει διότι τα VRF είναι υπό-routing tables και για τις διεπαφές που ανήκουν σε αυτά έχουν ξεχωριστή δρομολόγηση. Στην συγκεκριμένη περίπτωση χρησιμοποιείται VRF-lite η οποία είναι έκδοση του VRF που χρησιμοποιείται εκτός MPLS και παρέχει περιορισμένες αλλά χρήσιμες δυνατότητες.

Στην συγκεκριμένη περίπτωση αξιοποιείται η δυνατότητα των VRFs να παρέχουν δρομολόγηση για την ανακύκλωση της ροής. Η δρομολόγηση γίνεται σε συνδυασμό με VLANs πάνω στην

[27] Cisco, 'Configuring VRF-lite' στο *Catalyst 4500 Series Switch Cisco IOS Software Configuration Guide, 12.2(31)SG*, 4 Μαίου 2007 σ. 2-3

σύνδεση του DANOS και του switch. Τα VLANs δημιουργούν εικονικές διεπαφές στα δύο μηχανήματα και μπορούν να κατανεμηθούν ανάλογα σε VRF. Η κίνηση που λαμβάνεται από κάθε εικονική διεπαφή ανακατευθύνεται στην επόμενη εικονική διεπαφή μέσω του πίνακα δρομολόγησης του VRF. Όπως φαίνεται στο παρακάτω σχήμα για το σκοπό αυτό, ομαδοποιούνται τα VLAN σε ζεύγη ανά VRF. Όποια κίνηση έρχεται, επιστρέφεται από το αντίστοιχο ζεύγος. Εξαίρεση αποτελούν φυσικά το πρώτο και τελευταίο VRF στο DANOS, καθώς εκεί ανήκει το υποδίκτυο της πηγής και προορισμού αντίστοιχα της ροής για να μπορεί να γίνει η αντίστοιχη προώθηση.



Εικόνα 26: Πορεία ροής στην τοπολογία

Η ροή ξεκινά από το Ubuntu1 και καταλήγει στο DANOS σε διεπαφή που ανήκει στο VRF0. Αυτό προωθεί το πακέτο στη διεπαφή που διαθέτει για το VLAN500 που ανήκει και αυτό στο ίδιο VRF, με προορισμό την διεύθυνση του Switch στο VLAN500. Το πακέτο λαμβάνεται στο VRF dn0 του Switch, ο οποίος με την ίδια διαδικασία προωθεί το πακέτο στο VLAN501 που ανήκει στο ίδιο VRF. Αυτή η διαδικασία ακολουθείται, μέχρι το πακέτο να φτάσει στο DANOS στο τελικό VRF, όπου προωθείται προς το υποδίκτυο του Ubuntu2 για να φτάσει τελικά στον προορισμό του. Το πακέτο καθ' όλη τη διάρκεια της πορείας του διατηρεί την ίδια διεύθυνση προορισμού και κατευθύνεται από τα επιμέρους routing tables (VRFs) συνεχώς σε διαφορετικό VLAN. Επειδή η

κίνηση είναι συνεχής, παρατηρείται ότι η αρχική κίνηση των 1Gbps ανακυκλώνεται τόσες φορές όσες είναι τα VLAN πάνω στην γραμμή, με αποτέλεσμα να επιτυγχάνεται θεωρητικά έως 60 Gbps κίνηση πάνω στην γραμμή που συνδέει τα DANOS και Switch. Για περαιτέρω κατανόηση παρατίθενται οι πίνακες δρομολόγησης του DANOS για τα vrf 0,1,2,23,24 και 25.

Vrf0	
10.10.50.0/29	Directly connected
192.168.1.0/24	Directly connected
192.168.2.0/24	Via 10.10.50.2, dp0bond0.500

Vrf1	
10.10.51.0/29	Directly connected
10.10.52.0/29	Directly connected
192.168.1.0/24	Via 10.10.51.2, dp0bond0.501
192.168.2.0/24	Via 10.10.52.2, dp0bond0.502

Vrf2	
10.10.53.0/29	Directly connected
10.10.54.0/29	Directly connected
192.168.1.0/24	Via 10.10.53.2, dp0bond0.503
192.168.2.0/24	Via 10.10.50.2, dp0bond0.504

Vrf23	
10.10.95.0/29	Directly connected
10.10.96.0/29	Directly connected
192.168.1.0/24	Via 10.10.95.2, dp0bond0.545
192.168.2.0/24	Via 10.10.96.2, dp0bond0.546

Vrf24	
10.10.97.0/29	Directly connected
10.10.98.0/29	Directly connected
192.168.1.0/24	Via 10.10.97.2, dp0bond0.547
192.168.2.0/24	Via 10.10.98.2, dp0bond0.548

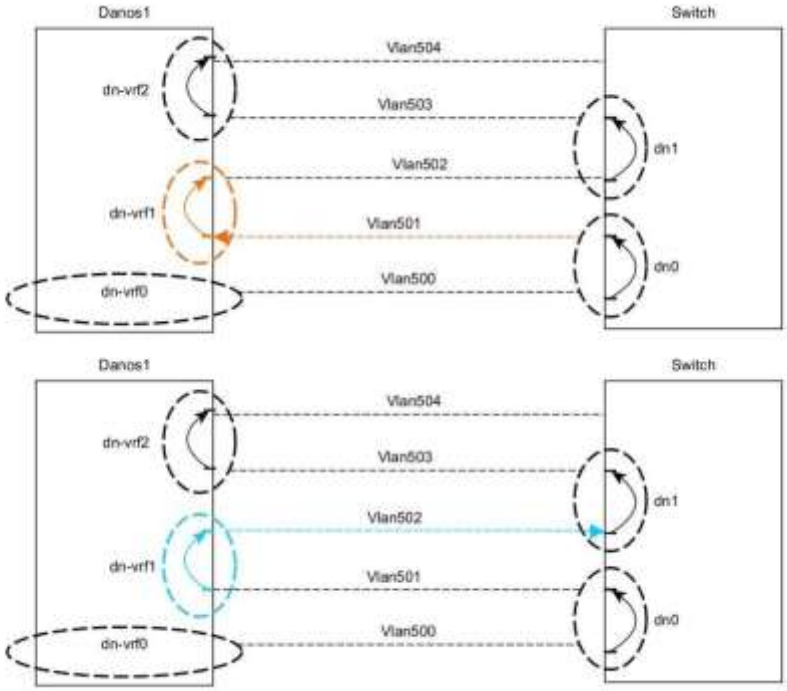
Vrf25	
10.10.99.0/29	Directly connected
192.168.1.0/24	Via 10.10.99.2, dp0bond0.549
192.168.2.0/24	Directly connected

Πίνακας 1: Καταγραφές διαδρομών σε διάφορους πίνακες δρομολόγησης VRF στο DANOS

Σε κάθε vrf υπάρχει μία καταγραφή για το υποδίκτυο 192.168.1.0/24 και μία για το υποδίκτυο 192.168.2.0/24. Οι καταγραφές για το 192.168.1.0/24 αφορούν τις απαντήσεις των πακέτων που στέλνονται και οι καταγραφές για το 192.168.2.0/24 αφορούν τα πακέτα που στέλνονται από την πηγή προς τον προορισμό.

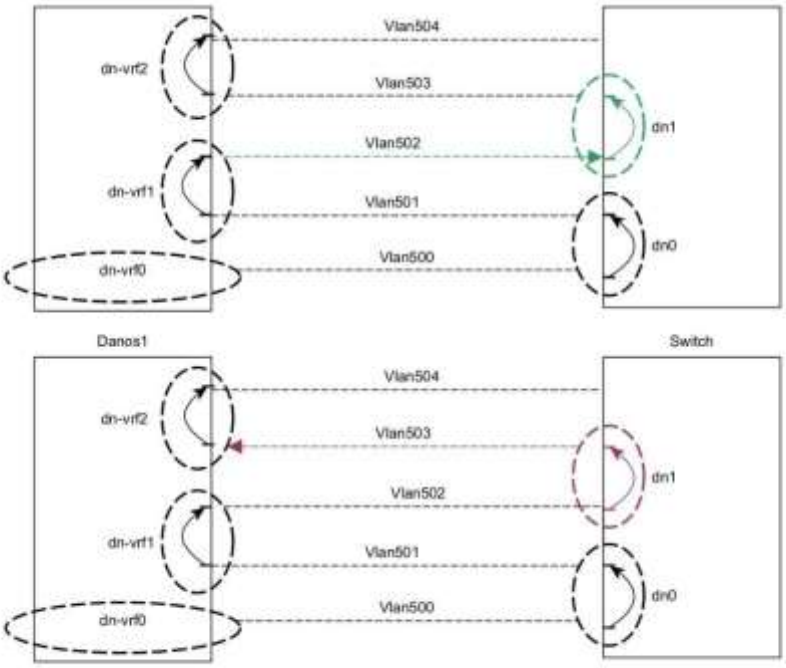
Όσον αφορά την κατεύθυνση από το Ubuntu1 στο Ubuntu2, η καταγραφή στον πίνακα δρομολόγησης του εκάστοτε vrf κατευθύνει την κίνηση πάντα προς τον Switch. Στο vrf1 για παράδειγμα, η κίνηση λαμβάνεται από την διεπαφή dp0bond0.501 (Vlan 501) και κατευθύνεται σύμφωνα με την αντίστοιχη καταγραφή του πίνακα, προς την διεπαφή dp0bond0.502(Vlan 502), δηλαδή πίσω στον Switch. Ο switch έχει αντίστοιχη παραμετροποίηση και καταγραφές για πακέτα

με προορισμό το 192.168.2.0/24 για να τα προωθήσει σε vlan διεπαφές μεγαλύτερου αύξοντα αριθμού. Η παραπάνω ανακύκλωση φαίνεται στο παρακάτω σχήμα.



Εικόνα 27: Ανακύκλωση κίνησης στο DANOS

Με αυτόν τον τρόπο, η κίνηση ξεκινώντας από χαμηλά VLANs, καταλήγει να κάνει ανακυκλώσεις και να φτάσει τελικά στον προορισμό της. Αντίστοιχα, στον switch φαίνεται στο παρακάτω σχήμα.



Εικόνα 28: Ανακύκλωση κίνησης στο switch

5.7 Υλικό για την σύνδεση μεταξύ DANOS και switch

Όπως αναφέρθηκε παραπάνω, το DANOS φιλοξενείται από τον server. Όταν γίνεται αναφορά για φυσική σύνδεση, αυτή αφορά την σύνδεση μεταξύ server και switch. Έχοντας δημιουργήσει την ιδέα για τον πολλαπλασιασμό κίνησης, εμφανίζεται το πρόβλημα της φυσικής σύνδεσης. Μεταξύ των server και switch μεταφέρεται κίνηση μεγαλύτερη από 20 Gbps, οπότε πρέπει οι θύρες που θα επιλεγούν και οι συνδέσεις να υποστηρίζουν τέτοιες ταχύτητες. Στον server διατίθενται 6 θήρες των 10 Gbps οι οποίες μπορούν να αντιστοιχιστούν με 6 θήρες στον switch για να δημιουργηθούν 6 συνδέσεις.

Από την πλευρά του server οι 6 θήρες αυτές παραχωρούνται στο DANOS μέσω της λειτουργίας PCIe passthrough, που δίνει απευθείας πρόσβαση στις συγκεκριμένες NIC, με σκοπό την βελτίωση των επιδόσεων εισόδου/εξόδου πακέτων. Η βελτίωση έγκειται στο γεγονός ότι δεν

υπάρχει πια μεσολαβητής ανάμεσα στο DANOS και τις NIC, μειώνοντας τις περιττές καθυστερήσεις.

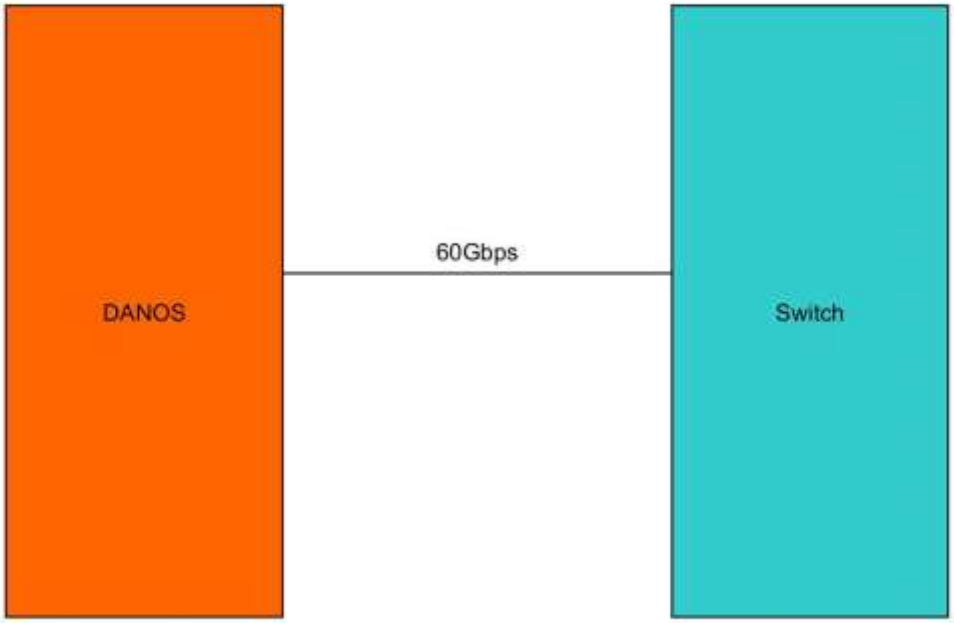
Εικόνα 29: Απόκτηση θηρών NIC από το DANOS μέσω pci-passtrough

5.8 Ένωση των 6 full-duplex γραμμών σε μία 60Gbps γραμμή

Η παραπάνω υλοποίηση με 6 ξεχωριστές συνδέσεις μεταξύ των δύο μηχανημάτων δημιουργεί το εξής πρόβλημα: τα μηχανήματα για να αποφύγουν τις ανακυκλώσεις και πλημμύρες πακέτων, ενεργοποιούν μόνο μία από τις έξι συνδέσεις, ενώ αφήνουν τις υπόλοιπες πέντε ως εφεδρικές. Αυτό γίνεται μέσω του πρωτοκόλλου STP που κατασκευάζει τοπολογία δέντρου (δηλαδή απουσία κύκλων) σε δίκτυα υπολογιστών. Χωρίς περαιτέρω ενέργειες, μεταξύ των δύο μηχανημάτων θα υπήρχε μόνο μία ενεργή σύνδεση 10Gbps, που φυσικά δεν αρκεί για τις επιθυμητές ταχύτητες.

Λύση στο παραπάνω πρόβλημα δίνει η τεχνική Link-aggregation[28], μια τεχνική που ενώνει δύο και παραπάνω γραμμές σε μια νοητή γραμμή, μοιράζοντας την εισερχόμενη κίνηση σε όλες τις επιμέρους διεπαφές.

[28] Cisco, 'IEEE 802.3ad Link Bundling', *cisco* [ιστοσελίδα], 4 Δεκεμβρίου 2006, IEEE 802.3ad Link Bundling, https://www.cisco.com/c/en/us/td/docs/ios/12_2sb/feature/guide/sbcelacp.html, (τελευταία πρόσβαση 29 Ιανουαρίου 2023).



Εικόνα 30: Ένωση διεπαφών για τη δημιουργία 60 Gbps full-duplex γραμμής

Στο DANOS το Link-aggregation υπάρχει υπό την μορφή bonding interface ενώ στον switch υπό την μορφή etherchannel και πιο συγκεκριμένα channel-group. Για την ένωση των διεπαφών χρησιμοποιείται το πρωτόκολλο LACP[29] με έναν εκ των switch και DANOS σε ενεργή κατάσταση(active mode) και τον άλλον σε παθητική κατάσταση(passive mode). Το ίδιο θα μπορούσε να επιτευχθεί και με τα δύο άκρα σε ενεργή κατάσταση.

5.9 Περιορισμοί αριθμού VRF και εύρους ζώνης γραμμής

Ο πρώτος περιορισμός κίνησης αφορά την ζεύξη μεταξύ των DANOS και switch. Η 60Gbps full duplex γραμμή δίνει την δυνατότητα να επεξεργάζεται ανά πάσα στιγμή το DANOS έως 120Gbps (60 ανά κατεύθυνση).

Ο δεύτερος περιορισμός αφορά τον switch που δεν επιτρέπει την δημιουργία περισσότερων των 25 VRF, οπότε το ανώτερο θεωρητικό όριο που μπορεί να επιτευχθεί είναι οι 25 ανακυκλώσεις ανά κατεύθυνση. Στην συγκεκριμένη υλοποίηση επιλέχθηκε ο μέγιστος αριθμός, δηλαδή 25 φορές, για να επιτευχθεί αντίστοιχα και ο μέγιστος πολλαπλασιασμός. Οι 25 ανακυκλώσεις θα

[29] Για περεταίρω πληροφορίες σχετικά με τον τρόπο λειτουργίας του LACP βλ. Huawei, 'What Is Link Aggregation Control Protocol', Huawei encyclopedia, https://info.support.huawei.com/info-finder/encyclopedia/en/LACP.html, (τελευταία πρόσβαση 29 Ιανουαρίου 2023).

αποτελούσαν σημαντικό περιορισμό σε συνδυασμό με την 1Gbps σύνδεση, καθώς δεν θα μπορούσε η πολλαπλασιασμένη κίνηση να φτάσει πέρα από τα 25 Gbps ανά κατεύθυνση, πολύ λιγότερο από τις δυνατότητες της γραμμής (60Gbps). Η γραμμή του 1Gbps στις δοκιμές παρουσίασε προβλήματα και επιλέχθηκε εν τέλει ο Virtio(paravirtualised) driver που παρέχει δυνατότητα παραγωγής κίνησης έως 10Gbps. Η ταχύτητα αυτή, μέσω των ανακυκλώσεων, είναι φυσικά ικανή, ακόμα και με τον περιορισμό των 25 ανακυκλώσεων, να παράξει θεωρητικώς έως 250Gbps ανά κατεύθυνση. Ο αριθμός αυτός περιορίζεται από τα 60Gbps της γραμμής. Έτσι, θα φανεί στις δοκιμές, ότι η ταχύτητα που στέλνει η πηγή, περιορίζεται ιδανικά σε ταχύτητα, που πολλαπλασιασμένη επί τις 25 ανακυκλώσεις, δίνει το επιθυμητό 60Gbps, δηλαδή στα 2.4Gbps. Ο περιορισμός των VRF οδηγεί και στον περιορισμό των Vlans, καθώς αντιστοιχίζονται 2 Vlan ανά VRF και άρα ορίζονται συνολικά 50 Vlans. Να σημειωθεί ότι, λόγω της δομής της ανακύκλωσης, καθώς το DANOS έχει άλλα δύο υποδίκτυα να συμπεριλάβει, στα VRF διαθέτει 26 Vrf.

5.10 Χρησιμοποιούμενο Υλικό

Server: Dell EMC PowerEdge R740

Cpu: intel xeon gold 6226R Processor

Διαθέτει 2 sockets κάθε ένα από 16 cores και υποστηρίζει hyperthreading, παρέχοντας συνολικά 64 CPUs.

Switch: **Cisco C9500-24Y4C**

Switch της σειράς Catalyst 9500 25G high-performance. Χρησιμοποιεί το λογισμικό Cisco IOS XE Fuji έκδοση 16.9.5. Διαθέτει 24 1/10/25G Gigabit Ethernet και 4 40/100 Uplink.

5.11 Σύνοψη

Η τοπολογία αποτελείται από δύο μηχανήματα Ubuntu, ένα πηγής και ένα προορισμού, που συνδέονται με το μηχάνημα του DANOS. Το DANOS στέλνει την παραγόμενη κίνηση στο switch και το switch την επιστρέφει. Η διαδικασία αυτή γίνεται 25 φορές και έπειτα το DANOS προωθεί την κίνηση στον προορισμό. Για την λειτουργία αυτή χρησιμοποιούνται 50Vlans και στα δύο μηχανήματα, 25VRFs στο Switch και 26 στο DANOS. Η σύνδεση μεταξύ DANOS και Switch αποτελείται από έξι full-duplex 10Gbps συνδέσεις που έχουν ενωθεί και δημιουργούν μια 60Gbps full-duplex διεπαφή.

Κεφάλαιο 6 : Δοκιμή ροής

6.1 Εργαλεία

6.1.1 Iperf3

Το iperf3 είναι ένα εργαλείο για μέτρηση του μέγιστου δυνατού εύρους ζώνης που μπορεί να επιτευχθεί σε δίκτυο IP. Υποστηρίζει αλλαγές σε διάφορες μεταβλητές που αφορούν τον χρονισμό τους, buffers, συνδέσεις κα. Υποστηρίζει TCP και UDP πακέτα που χρησιμοποιούνται στις μετρήσεις. Το iperf λειτουργεί δημιουργώντας έναν client και έναν server με τον client να προωθεί κίνηση προς τον server. Το default port για έναρξη επικοινωνίας των δύο μηχανημάτων που φιλοξενούν τους server και client είναι το 5201, στο οποίο ακούει ο server και συνδέεται ο client. Υπάρχει πλήθος παραμέτρων που καθορίζουν το πρωτόκολλο επικοινωνίας, το μέγεθος των πακέτων, την ταχύτητα αποστολής τους ανά σύνδεση κα. Μερικές από αυτές τις παραμέτρους θα χρησιμοποιηθούν στο πλαίσιο των δοκιμών. Όσον αφορά το iperf, το μηχάνημα Ubuntu1 λαμβάνει τον ρόλο του client(source) και το Ubuntu2 του server(destination).

6.1.2 Telegraf

Το telegraf είναι ένας server-based agent που συλλέγει και στέλνει μετρικές και συμβάντα από και προς βάσεις, συστήματα και αισθητήρες IoT. Μπορεί να λάβει δεδομένα μέσω διαφόρων πηγών, συμπεριλαμβανομένου του snmp που χρησιμοποιείται στα πλαίσια της διπλωματικής, για να ληφθούν δεδομένα διαδικτυακής κίνησης από απομακρυσμένα μηχανήματα και να προωθηθούν στη βάση δεδομένων influxdb.

6.1.3 Influxdb

Το InfluxDB αποτελεί μια βάση δεδομένων που χρησιμοποιεί Telegraf plugins για να συλλέγει δεδομένα. Μπορεί να επεξεργάζεται δεδομένα, να απαντά σε queries και να τα εμφανίζει σε UI dashboards μέσω εργαλείων όπως το grafana

6.1.4 Grafana

Το grafana[30] είναι ένα λογισμικό ανοιχτού κώδικα που παρέχει εικονοποίηση και αναλυτικά στοιχεία πάνω σε δεδομένα. Επιτρέπει την δημιουργία queries προς τις βάσεις δεδομένων που

[30] Για βασικά χαρακτηριστικά του grafana βλ. Grafana Labs, 'Gradana OSS', *grafana* [ιστοσελίδα], https://grafana.com/docs/grafana/latest/introduction/, (τελευταία πρόσβαση 29 Ιανουαρίου 2023)

συνδέεται, τα οποία μπορεί και παρουσιάζει διαγραμματικά σε μορφή χρονοσειρών. Στα πλαίσια της διπλωματικής, χρησιμοποιείται σε συνδυασμό με τα InfluxDB και telegraf για την απεικόνιση των δεδομένων δικτυακής κίνησης από τα μηχανήματα.

6.1.5 Snmp

Το snmp[31] είναι ένα πρωτόκολλο επιπέδου εφαρμογής που επιτρέπει στις συσκευές να ανταλλάσσουν διαχειριστικές πληροφορίες. Με αυτό το πρωτόκολλο οι διαχειριστές δικτύου μπορούν να επιβλέπουν την απόδοση του δικτύου, να βρίσκουν και να επιλύουν δικτυακά προβλήματα και να αναπτύσσουν επέκταση του δικτύου. Το snmp μεταφέρει πληροφορίες για τα συστήματα στα οποία είναι ενεργοποιημένο, χρησιμοποιώντας την βάση δεδομένων MIB-II[32] η οποία συλλέγει συγκεκριμένες πληροφορίες συστήματος, και κυρίως όσον αφορά τα δικτυακό μέρος αυτού, τις οποίες ομαδοποιεί και αποθηκεύει. Χρησιμοποιώντας snmp queries μπορεί ο διαχειριστής να ζητήσει πληροφορίες από το εκάστοτε μηχάνημα, παραθέτοντας το OID της πληροφορίας που ζητάει. Το OID χαρακτηρίζει μοναδικά πίνακες ή καταχωρήσεις στην βάση. Το snmp χρησιμοποιείται στα πλαίσια της διπλωματικής αυτής για να συλλεχθούν πληροφορίες από τα DANOS και switch όσον αφορά τα bandwidth εισόδου και εξόδου στις διεπαφές τους, όπως θα αναδειχθεί παρακάτω.

6.1.6 Top

Το πρόγραμμα top παρέχει μια δυναμική οπτική πραγματικού χρόνου ενός τρέχοντος συστήματος. Μπορεί να απεικονίζει περίληψη πληροφοριών συστήματος καθώς και διαδικασίες και νήματα που διαχειρίζονται από τον πυρήνα του Linux.[33] Το εργαλείο αυτό χρησιμοποιείται για να καταγραφεί η χρησιμοποίηση των πυρήνων CPU από την διαδικασία dataplane, η οποία αφορά την επεξεργασία των πακέτων κίνησης στον χώρο χρήστη. Μέσω του top φαίνεται αναλυτικά πόση από αυτή τη χρησιμοποίηση κατανέμεται σε κάθε πυρήνα, καθώς και πόσο ποσοστό του χρόνου του καταναλώνει ο πυρήνας για λειτουργίες kernel/user space κα.

---

[31] Για περαιτέρω ανάγνωση για το SNMP βλ. σχετικά J. Case, M. Fedor και M. L. Schoffstall, 'A Simple Network Management Protocol (SNMP)', *rfc editor* [ιστοσελίδα], 1157, Μάιος 1990, https://www.rfc-editor.org/rfc/rfc1157, (τελευταία πρόσβαση 29 Ιανουαρίου 2023)

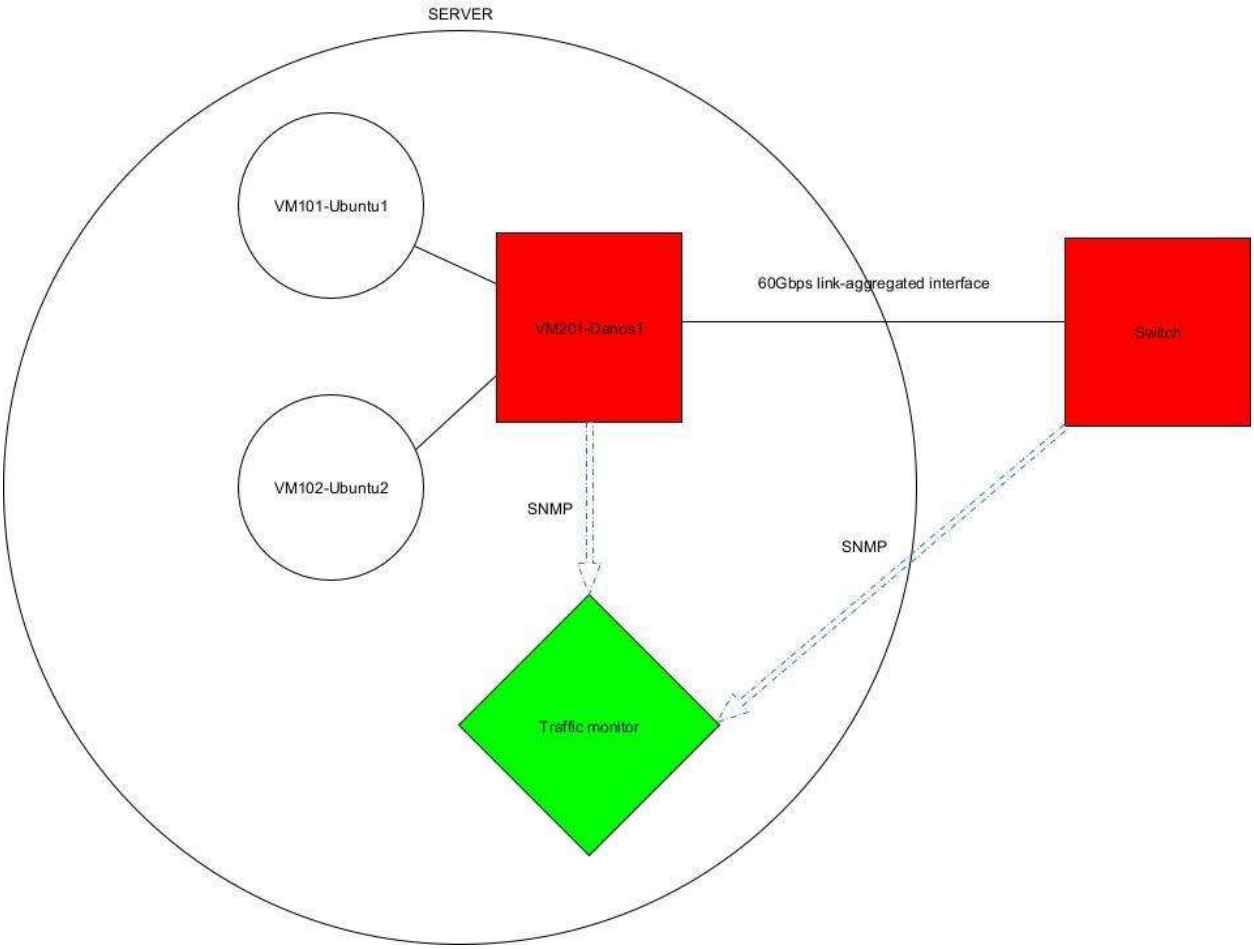
[32] Για αναλυτικότερες πληροφορίες για την βάση MIB βλ. K. McCloghrie, M. Rose, 'Management Information Base for Network Management of TCP/IP-based internets: MIB-II', *rfc editor* [ιστοσελίδα], 1213, Μάρτιος 1991, https://www.rfc-editor.org/rfc/rfc1213.html, (τελευταία πρόσβαση 29 Ιανουαρίου 2023)

[33] M. Kerrisk, 'top(1) – Linux manual page', *man* [ιστοσελίδα], https://man7.org/linux/man-pages/man1/top.1.html, (τελευταία πρόσβαση 29 Ιανουαρίου 2023).

6.2 Προετοιμασία των Δοκιμών

6.2.1 Μηχάνημα Συλλογής Μετρήσεων

Στον proxmox server που φιλοξενεί τα VM, έχει δημιουργηθεί ένα μηχάνημα Ubuntu το οποίο έχει αναλάβει την συλλογή δεδομένων από το DANOS και Switch. Σε αυτό το μηχάνημα τρέχουν οι υπηρεσίες telegraf, influxdb και grafana και με την χρήση του snmp το telegraf καταφέρνει και συλλέγει τις προαναφερθείσες πληροφορίες. Οι μετρήσεις αυτές παρουσιάζονται σε UI μέσω browser στο grafana και αναπαρίστανται γραφικές κίνησης. Όπου κρίνεται αναγκαίο και συντελεί στην κατανόηση των αποτελεσμάτων και στην επίδειξη γραφικά των μετρικών, παρατίθενται γραφήματα που έχουν δημιουργηθεί στο grafana.



Εικόνα 31: Εικονική μηχανή τηλεμετρίας

6.2.2 Παράμετροι iperf

Οι βασικές παράμετροι που χρησιμοποιούνται στο πρώτο μέρος των δοκιμών είναι ο αριθμός των συνδέσεων μεταξύ client και server και το εύρος ζώνης ανά σύνδεση. Εξετάζεται πώς η τυχαία

επιλογή των Ports από το iperf επηρεάζει το τελικό bandwidth, καθώς αλληλεπιδρά και με το LACP. Για αυτό το μέρος των δοκιμών χρησιμοποιούνται τα μέγιστα φορτία UDP και TCP που αντιστοιχούν σε 1472 και 1460 Bytes για να μην γίνεται fragmentation και ταυτόχρονα να υπάρχει όσο το δυνατόν καλύτερη αναλογία φορτίου και επικεφαλίδων. Κάθε test τρέχει για πέντε λεπτά, που είναι υπεραρκετός χρόνος, καθώς από τα πρώτα δευτερόλεπτα υπάρχει σταθεροποίηση. Επιλέχθηκε τόσος μεγάλος χρόνος για να φανεί σε βάθος χρόνου, αν καταφέρνει το DANOS να διατηρεί τις συγκεκριμένες ταχύτητες. Έπειτα, για έναν συνδυασμό παραμέτρων που επιτυγχάνεται ένα υψηλό εύρος ζώνης, δοκιμάζονται διαφορετικά μεγέθη πακέτων τόσο για UDP και TCP, για να φανεί αν και κατά πόσο επηρεάζουν το εύρος ζώνης. Όλα τα παραπάνω tests έχουν υλοποιηθεί σε scripts φλοιού και παρατίθενται στο παράρτημα.

Σημείωση: Οι δοκιμές με αριθμό συνδέσεων άνω των 128, που αποτελεί το όριο του iperf3, αποτελούν αριθμό που προκύπτει από παράλληλες εκτελέσεις iperf. Δηλαδή για την επίτευξη 200 και 500 συνδέσεων, έγιναν 2 και 5 διαφορετικές iperf client-server συνδέσεις αντίστοιχα, με διαφορετικό listen-port για τον server. Πιο συγκεκριμένα στα port 5201-5205.

6.3 Test εύρους ζώνης συναρτήσει των συνδέσεων και του εύρους ζώνης ανά σύνδεση

6.3.1 Προέλευση Μετρικών

Οι τρεις παράμετροι TCP/UDP, αριθμός συνδέσεων και εύρος ζώνης ανά σύνδεση, αποτελούν τις εισόδους της πειραματικής διαδικασίας. Οι στήλες Iperf3 Sender Bandwidth, iperf3 Reciever Bandwidth και Error Rate (μόνο για UDP) ελήφθησαν από τα αποτελέσματα του iperf. Καθώς παρατηρήθηκε μικρή διαφορά μεταξύ των αποτελεσμάτων αυτών και την πραγματική κίνηση που μετρά το DANOS ως είσοδο και έξοδο, παρατίθενται και αυτές οι μετρήσεις. Οι στήλες DANOS Input Traffic, DANOS Output Traffic, DANOS Bandwidth και Switch Bandwidth ελήφθησαν από τις γραφικές παραστάσεις του grafana, όταν σταθεροποιούνται οι γραφικές παραστάσεις. Οι παράμετροι αυτές προέρχονται από την βάση MIB σε κάθε μηχάνημα, από την οποία συλλέγει πληροφορίες το telegraf μέσω του πρωτοκόλλου SNMP.

6.3.2 Επεξήγηση Μετρικών

Οι μετρικές iperf3 sender/receiver bandwidth αφορούν το μέσο εύρος ζώνης που μέτρησε, ως εύρος ζώνης αποστολής και λήψης αντίστοιχα, το iperf από τα δύο μηχανήματα Ubuntu. Το Error Rate αφορά την διαφορά της ταχύτητας λήψης και αποστολής μεταξύ των δύο μηχανημάτων και

αφορά μόνο το UDP. Οι μετρικές DANOS In/Out Traffic αφορούν την κίνηση μεταξύ Ubuntu1-DANOS και DANOS-Ubuntu2 αντίστοιχα και αφορούν τι αντίστοιχες μετρήσεις του iperf και στα πλαίσια της διπλωματικής, θεωρούνται πιο αξιόπιστες, καθώς προκύπτουν μέσω του πρωτοκόλλου snmp. Οι μετρικές DANOS Bandwidth και Switch Bandwidth αφορούν το εύρος ζώνης μεταξύ DANOS και Switch, όπως αυτό μετριέται από το καθένα. Επειδή ουσιαστικά αφορούν ακριβώς την ίδια κίνηση, αναμένεται να ταυτίζονται πλήρως.

6.3.3 Θεωρητικές Τιμές

Για το UDP, το εύρος ζώνης που μετριέται στο DANOS και Switch πάνω στην link-aggregated διεπαφή αναμένεται να βρίσκεται ανάμεσα στο εύρος ζώνης αποστολής και στο εύρος ζώνης λήψης των Ubuntu μηχανημάτων πολλαπλασιασμένο κατά 50 και πιο συγκεκριμένα κατά 25 ανά κατεύθυνση. Ο πολλαπλασιαστικός παράγοντας οφείλεται στην ανακύκλωση της ροής 25 φορές ανά κατεύθυνση. Το τελικό εύρος ζώνης όμως βρίσκεται ανάμεσα σε αυτές τις δύο τιμές, διότι στο UDP οι routers απορρίπτουν πακέτα που δεν φτάνουν ποτέ στον προορισμό τους.

Για το TCP το εύρος ζώνης αναμένεται να είναι ακριβώς 50 φορές αυτό που στέλνει ο δέκτης και είναι ίδιο και με αυτό που λαμβάνει ο παραλήπτης ως αποτέλεσμα της δομής και υπηρεσιών ελέγχου του TCP. Αυτό συμβαίνει, διότι όταν σταθεροποιηθεί το TCP, υπάρχουν ελάχιστα σφάλματα, τα οποία αν προκύψουν, υπάρχει αναμετάδοση, οπότε υπάρχουν και ελάχιστες απορρίψεις στους routers. Αυτό έχει ως αποτέλεσμα ο πολλαπλασιαστικός παράγοντας να αφορά πρακτικά ακριβώς τον ρυθμό αποστολής και λήψης των μηχανημάτων Ubuntu.

6.3.4 Αποτελέσματα Μετρήσεων

TCP/UDP	**Number of Connections**	**Bandwidth per Connection**	**Packet Size** **Bytes**	**Iperf3 Sender bandwidth** **Mbps**	**iperf3 Reciever bandwidth** **Mbps**	**Error Rate (only for UDP)**	**DANOS Input Traffic** **Mbps**	**DANOS Output Traffic** **Mbps**	**DANOS bandwidth** **In/Out (Gbps)**	**Switch bandwidth** **In/Out (Gbps)**
UDP	1	300	1472	300	300	0%	287	287	7.2	7.23

UDP	1	400	1472	400	368	8%	383	353	9.17	9.19
UDP	1	500	1472	500	307	39%	479	295	9.16	9.2
UDP	2	300	1472	600	596	0.60%	575	575	14.4	14.5
UDP	2	400	1472	800	717	10.00%	766	697	18.3	18.4
UDP	2	500	1472	1000	594	41%	958	575	18.24	18.3
UDP	5	300	1472	1500	627	58%	1440	619	22.3	22.3
UDP	5	400	1472	2000	463	77%	1920	443	22.6	22.6
UDP	5	500	1472	2500	272	89%	2390	258	20.9	20.8
UDP	10	100	1472	1000	918	8.20%	958	881	22.9	22.8
UDP	10	150	1472	1500	440	71%	1440	424	18.3	18.4
UDP	10	200	1472	2000	1.19	40%	1920	1140	37.3 36.6	36.8 37.4
UDP	10	250	1472	2500	351	86%	2390	331	22.4	22.5
UDP	10	300	1472	3000	434	86%	2870	416	28.6	28.6
UDP	20	50	1472	1000	711	29.00%	958	683	20.1	20.1
UDP	20	75	1472	1500	1500	0%	1440	1440	36	36.1
UDP	20	100	1472	2000	770	62%	1920	738	29.5	29.6
UDP	20	125	1472	2500	888	64%	2400	849	35.7	35.7
UDP	20	150	1472	3000	637	79%	2870	606	33.3 33.6	33.8 33.4
UDP	50	20	1472	1000	100	0%	958	958	24	24

UDP	50	30	1472	15000	1480	1.10%	1440	1420	35.8	35.9
UDP	50	40	1472	2000	1260	37%	1920	1210	37.8	37.8
UDP	50	50	1472	2500	1160	54%	2390	1110	40.5	40.5
UDP	100	10	1472	1000	1000	0%	958	958	24	24
UDP	100	15	1472	1500	1500	0%	1440	1440	36	36
UDP	100	20	1472	2000	1840	8%	1920	1770	46.0 46.1	46.2 46.1
UDP	100	25	1472	2500	1280	49%	2390	1230	42.5 42.6	42.7 42.6
UDP	200	5	1472	1000	1000	0%	958	958	24	24.1
UDP	200	7	1472	1400	1400	0%	1340	1340	33.6	33.7
UDP	200	10	1472	2000	1548	22%	1920	1430	42	42
UDP	200	12	1472	2400	1382	42%	2300	1320	43.6	43.7
UDP	500	2	1472	1000	1000	0%	958	958	24	24.1
UDP	500	3	1472	1500	1500	0%	1440	1440	36	36.1
UDP	500	4	1472	2000	2000	0%	1920	1920	48	48.1
UDP	500	5	1472	2500	1551	38%	2390	1490	46.9	47
TCP	1	300	1460	300			292	292	7.33	7.36
TCP	1	400	1460	375			365	365	9.16	9.19
TCP	1	500	1460	375			365	365	9.16	9.19
TCP	2	300	1460	600			584	584	14.7	14.7

TCP	2	400	1460	731			723	723	18.2	17.9
TCP	2	500	1460	374			365	365	9.16	9.19
TCP	5	300	1460	788			777	777	19.6	19.7
TCP	5	400	1460	531			520	520	13	13
TCP	5	500	1460	745			725	725	18.3	18.3
TCP	10	100	1460	844			826	826	20.8	21
TCP	10	150	1460	1300			1300	1300	31.2	31.2
TCP	10	200	1460	742			752	752	19	18.8
TCP	10	250	1460	1280			1240	1240	31.2	31.3
TCP	10	300	1460	1060			970	970	24.4	24.8
TCP	20	50	1460	920			896	896	22.4	22.6
TCP	20	75	1460	1450			1420	1420	35.7	35.8
TCP	20	100	1460	1210			1180	1180	29.8	29.8
TCP	20	125	1460	945			924	924	23.3	23.6
TCP	20	150	1460	1111			1080	1080	27.3	27.4
TCP	50	20	1460	1000			978	978	24.7	24.6
TCP	50	30	1460	1420			1390	1390	34.9	35
TCP	50	40	1460	1840			1800	1800	45.3	45.4
TCP	50	50	1460	1920			1870	1870	47.2	47.3
TCP	100	10	1460	1000			978	978	24.8	24.9

TCP	100	15	1460	1500			1460	1460	37.1	37.2
TCP	100	20	1460	1890			1850	1840	46.6	46.7
TCP	100	25	1460	1880			1840	1830	46.4	46.5
TCP	200	5	1460	1000			984	974	24.8	24.9
TCP	200	7	1460	1400			1380	1360	34.8	34.9
TCP	200	10	1460	1928			1900	1880	47.8	47.8
TCP	200	12	1460	1892			1860	1840	46.9	46.9
TCP	500	2	1460	1000			991	971	25.3	25.3
TCP	500	3	1460	1500			1480	1460	37.4	37.6
TCP	500	4	1460	1966			1960	1910	49.1	49.3
TCP	500	5	1460	2112			2140	2060	53.4	53.6

Πίνακας 2: Αποτελέσματα μετρήσεων για δοκιμές με iperf

6.3.5 Παρατηρήσεις και Συμπεράσματα

Συνδέσεις

Για αρχή εξετάζεται η ειδική περίπτωση αριθμού συνδέσεων μικρότερου του 6, δηλαδή τον αριθμό των διεπαφών που έχουν "ενωθεί" για την κατασκευή της 60Gbps γραμμής.

Όπως δείχθηκε παραπάνω, το LACP κατευθύνει κάθε udp/tcp connection σε συγκεκριμένη από τις 6 διεπαφές κάνοντας κατακερματισμό στις διευθύνσεις και θύρες του πακέτου. Το iperf παράλληλα επιλέγει τυχαίες θύρες για τις παράλληλες συνδέσεις του. Όταν ο αριθμός των συνδέσεων γίνει πολύ μεγάλος, οι συνδέσεις μοιράζονται ίσα μεταξύ των 6 διεπαφών από το LACP.

Μία σύνδεση

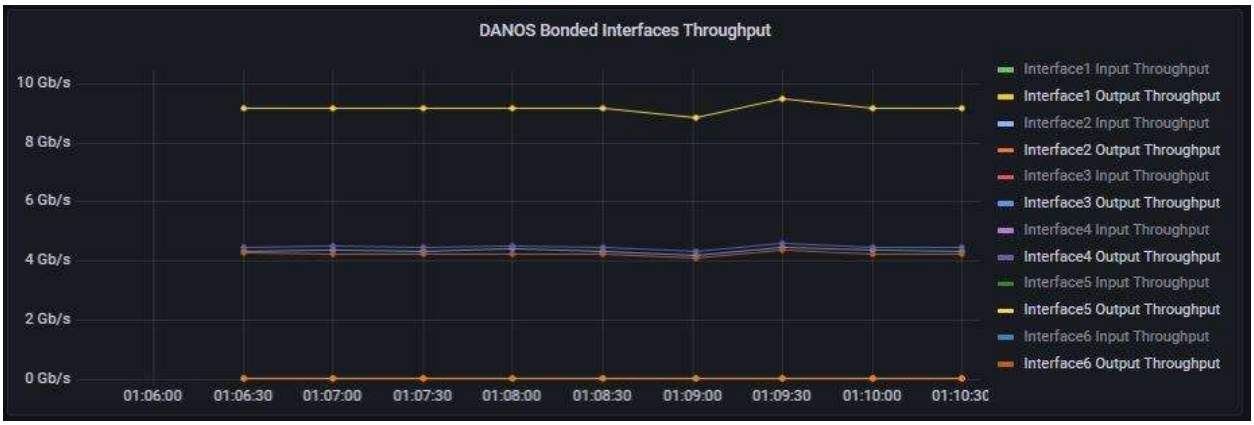
Στην περίπτωση της μίας σύνδεσης η ροή πάντα προωθείται στην ίδια διεπαφή από κάθε ένα από τα δύο μηχανήματα(το κάθε ένα μπορεί να την προωθεί σε διαφορετική αλλά δεν επηρεάζει λόγω του full-duplex). Έτσι ουσιαστικά δεν αξιοποιείται η δυνατότητα για 60Gbps αλλά μόνο 10Gbps, αφού η ροή από μία σύνδεση προωθείται σε συγκεκριμένη γραμμή που καθορίζεται από τον αλγόριθμο κατακερματισμού. Για αυτόν τον λόγο, παρατηρείται ότι περί τα 400Mbps ταχύτητας αποστολής είναι το ανώτατο ορίου εύρους ζώνης που μπορεί να επιτευχθεί.

Δύο συνδέσεις

Για πάνω από μία σύνδεση, αρχίζει και φαίνεται η εξάρτηση της ταχύτητας από το πού τοποθετούνται οι ροές από το LACP. Αν τύχει οι δύο συνδέσεις να τοποθετηθούν σε διαφορετικές διεπαφές, τότε μπορεί να επιτευχθεί αναλόγως έως 20Gbps, καθώς αξιοποιούνται πια δύο διεπαφές (ανά κατεύθυνση). Αν όμως επαναληφθεί πολλές φορές η ίδια δοκιμή, μπορεί να τύχει να συμπέσουν οι δύο ροές και να γίνουν hash στην ίδια διεπαφή. Αυτό θα έχει ως αποτέλεσμα ξανά τον περιορισμό στα 10Gbps για ευνόητους λόγους. Είναι φανερή λοιπόν η άμεση εξάρτηση του εύρους ζώνης που μπορεί να αξιοποιηθεί, από τον αριθμό των συνδέσεων. Επειδή σε μικρό αριθμό συνδέσεων, όταν γίνεται προσπάθεια επίτευξης μεγάλων ταχυτήτων, ορίζεται μεγάλη ταχύτητα ανά σύνδεση, αν τύχουν αρκετές ροές μαζί, δημιουργείται bottleneck.

Πέντε συνδέσεις

Σε αυτήν την περίπτωση προσεγγίζεται το όριο των 6 συνδέσεων με τουλάχιστον μία σύνδεση να μένει ανενεργή. Όπως αναφέρθηκε παραπάνω αναλόγως, η ταχύτητα μπορεί να φτάσει έως τα 50Gbps αν και εφόσον τοποθετηθούν όλες οι συνδέσεις σε διαφορετικές διεπαφές. Στην συγκεκριμένη δοκιμή για UDP παρατηρείται ότι επιτυγχάνεται μέγιστη ταχύτητα 22.6Gbps. Αυτό οφείλεται στην διανομή των συνδέσεων από το LACP σε ίδιες διεπαφές, με αποτέλεσμα να γεμίζουν κάποιες, ενώ κάποιες άλλες δεν χρησιμοποιούνται πλήρως βλ. εικ.

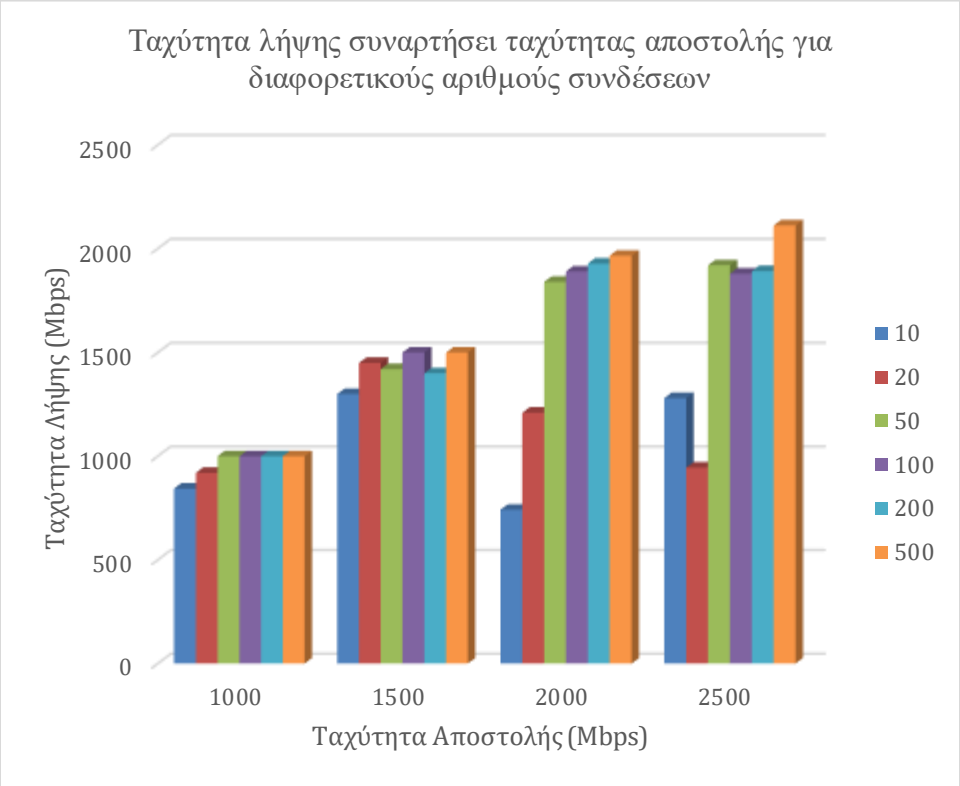


Διάγραμμα 1:Διαμοιρασμός κίνησης στις έξι διεπαφές (UDP-5συνδέσεις)

Όπως φαίνεται στο διάγραμμα, μία διεπαφή δουλεύει πλήρως κοντά στα 10Gbps , ενώ τρεις διεπαφές δουλεύουν κατά το ήμισυ και υπάρχουν και δύο ανενεργές.

Αφού χρησιμοποιούνται 4 από τις 6 διεπαφές με 5 αρχικές συνδέσεις, συνεπάγεται ότι 2 συνδέσεις τύχανε στην ίδια διεπαφή. Αυτό οδήγησε στο να γεμίσει η μια διεπαφή και λόγω σφαλμάτων περιορίστηκε κάθε ταχύτητα σύνδεσης περί τα 4.5 Gbps για να μπορούν δύο ροές να κατευθύνονται σε μία διεπαφή.

Συνδέσεις με αριθμό μεγαλύτερο των 6 συνδέσεων



Διάγραμμα 2:Ταχύτητα λήψης συναρτήσει ταχύτητας αποστολής για διαφορετικούς αριθμούς συνδέσεων

Η περίπτωση αυτή αφορά 10,20,50,100,200 και 500 συνδέσεις. Στο διάγραμμα 2 φαίνεται, ότι με μικρό αριθμό συνδέσεων (10-20) δεν μπορούν να επιτευχθούν μεγάλες ταχύτητες και επέρχεται κορεσμός κοντά στα 1.2Gbps. Όσο αυξάνεται ο αριθμός των συνδέσεων, παρατηρούνται μεγαλύτερες ταχύτητες λήψης ανάλογες της αποστολής. Καθώς προσεγγίζεται, το όριο της γραμμής (κοντά στα 2-2.5 Gbps αν διαιρεθούν τα 60 Gbps με τον συντελεστή πολλαπλασιασμού), οι συνδέσεις σε αριθμό μεγαλύτερο του 50 επιφέρουν ικανοποιητικά αποτελέσματα. Με 500 συνδέσεις επιτυγχάνεται επίδοση κοντά στα 54Gbps, το οποίο αποτελεί εξαιρετικό αποτέλεσμα. Οι πολλές συνδέσεις μοιράζονται πιο ομοιόμορφα και τυχόν ανωμαλίες στον διαμοιρασμό αυτών, επιφέρουν μικρή επιβάρυνση, καθώς ανά σύνδεση αναλογεί μικρότερη ταχύτητα.

Γενικό συμπέρασμα

Ο αριθμός των συνδέσεων είναι άμεσα συνυφασμένος με το τελικό εύρος ζώνης και αποτελεί έναν παράγοντα τυχαιότητας για κάθε εκτέλεση σύνδεσης. Η τυχαιότητα θα φανεί σε επόμενες δοκιμές που θα αναδειχθεί το εξής: για ίδιες παραμέτρους, για κάθε επανάληψη εκτέλεσης, εμφανίζεται

διαφορετικό σφάλμα (UDP) ή διαφορετική ταχύτητα αποστολής(TCP). Αν υπήρχε δυνατότητα θεωρητικώς, να χρησιμοποιηθεί μία 60Gbps γραμμή θα ήταν δυνατό με λιγότερες συνδέσεις να επιτευχθούν μεγαλύτερες ταχύτητες και να υπάρχει μεγαλύτερη συνέπεια μεταξύ των επαναλήψεων. Με το διαθέσιμο hardware, για να προσεγγιστεί το όριο των γραμμών, πρέπει να χρησιμοποιηθεί μεγάλος αριθμός συνδέσεων. Για συγκεκριμένες εφαρμογές, αν υπήρχε a-priori γνώση για την κατανομή των ports καθώς και να ήταν δυνατή στο DANOS και Switch η αλλαγή του hashing αλγορίθμου, θα μπορούσε να επιλεχθεί ο κατάλληλος, έτσι ώστε, με αρκετά μικρό αριθμό συνδέσεων, να μοιράζονται ισομερώς και να επιτυγχάνονται σχεδόν πάντα υψηλές ταχύτητες.

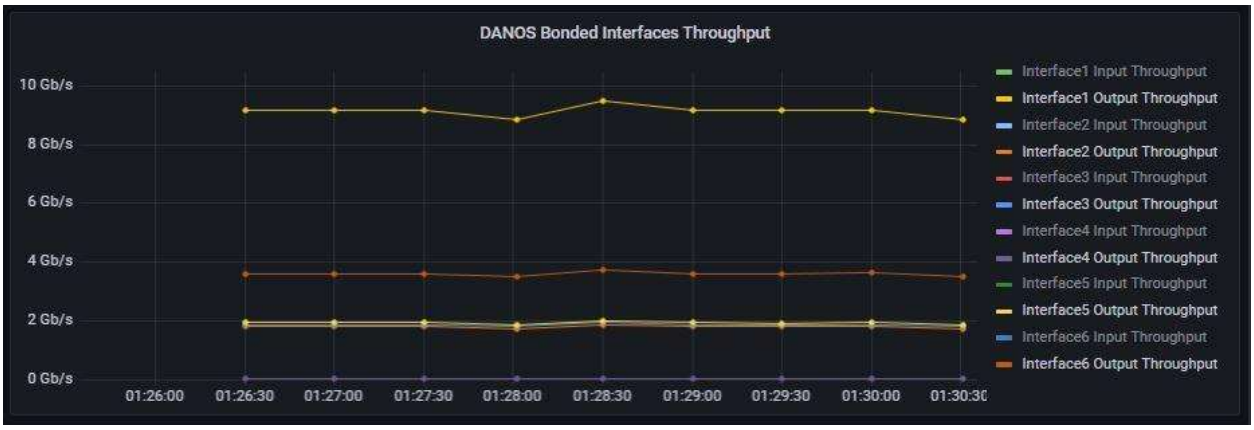
Εύρος ζώνης ανά σύνδεση

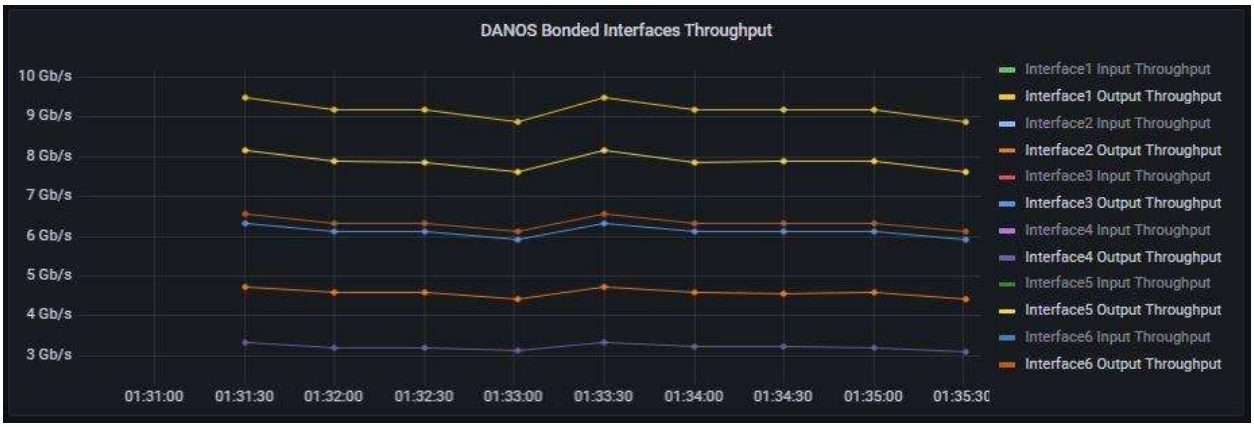
Το εύρος ζώνης ανά σύνδεση είναι συνυφασμένο με τον αριθμό συνδέσεων. Αν θεωρηθεί η ταχύτητα αποστολής ως σταθερή, το εύρος ζώνης ανά σύνδεση, ισούται με την ταχύτητα διαιρεμένη με τον αριθμό των συνδέσεων. Ο παράγοντας που πραγματικά επηρεάζει τις ταχύτητες και το LACP είναι ο αριθμός των συνδέσεων και όχι το εύρος ζώνης.

Παρατήρηση ανωμαλιών

Σε κάποιες περιπτώσεις παρατηρούνται, για ίδιο αριθμό συνδέσεων, μεγαλύτερα σφάλματα σε μικρότερες ταχύτητες. Αυτό συμβαίνει, διότι η επιλογή των port σε κάθε περίπτωση είναι ανεξάρτητη από την προηγούμενη. Σε περιπτώσεις που οι θήρες δεν μοιράζονται αρκετά ισομερώς στις διεπαφές, οι διεπαφές που αναλαμβάνουν άνισα μέρη της ροής, αποτελούν bottleneck και οδηγούνται σε πολλαπλά σφάλματα.

Ένα χαρακτηριστικό αποτέλεσμα των μετρήσεων είναι για UDP 10 συνδέσεων με 150 Mbps και 200 Mbps ταχύτητα ανά σύνδεση. Ενώ για τα 200Mbps αναμένεται να εμφανιστεί μεγαλύτερο σφάλμα, καθώς υπάρχει μεγαλύτερος κίνδυνος να φτάσει κάποια γραμμή στο όριο της , το σφάλμα είναι 40% ενώ για 150Mbps 70%. Αυτό συμβαίνει, επειδή οι συνδέσεις δεν μοιράστηκαν σωστά από το LACP στην δεύτερη περίπτωση. Αναλόγως στην περίπτωση TCP θα παρατηρούνταν μείωση της ταχύτητας προς αποφυγή σφαλμάτων. Το UDP δεν διαθέτει μηχανισμούς ελέγχου, οπότε η ανωμαλία εκφράζεται σε μεγαλύτερο σφάλμα.

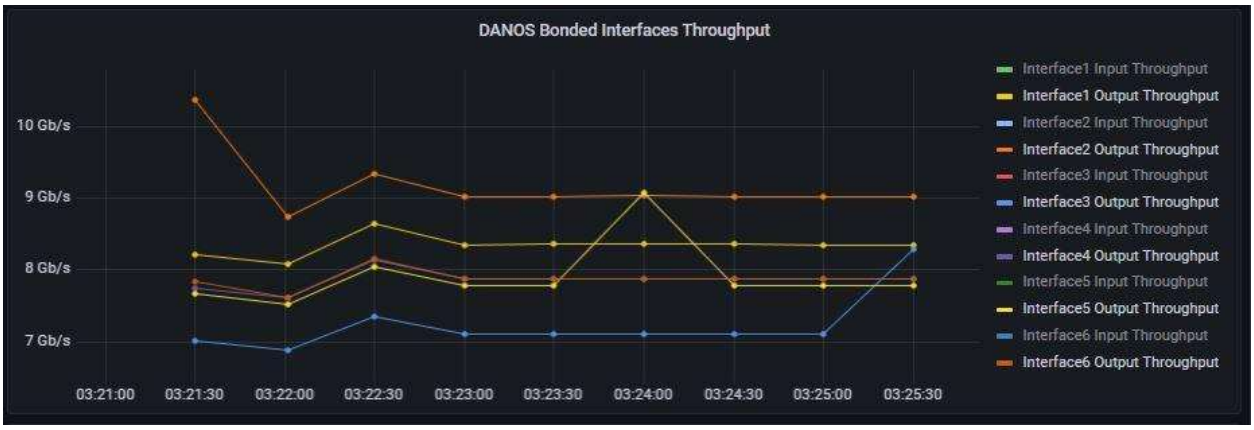




Διάγραμμα 3: Διαμοιρασμός κίνησης στις έξι διεπαφές για 6 συνδέσεις (α) 150Mbps/σύνδεση (β) 200 Mbps/σύνδεση

Στα παραπάνω σχήματα παρατηρείται ότι ο διαμοιρασμός δεν είναι ιδανικός σε καμία από τις δύο περιπτώσεις και εμφανίζονται αρκετά σφάλματα. Στην περίπτωση όμως της ταχύτητας 150Mbps/σύνδεση εμφανίζεται μεγαλύτερη ανομοιομορφία, με μία γραμμή να φορτώνεται μεγάλο μέρος της κίνησης και έτσι το σύστημα οδηγείται σε πολύ περισσότερα σφάλματα.

Αν εξεταστεί αντίστοιχα το καλύτερο αποτέλεσμα του πρώτου μέρους των δοκιμών για UDP που επιτεύχθηκε ταχύτητα 48Gbps για 500 συνδέσεις με 4Mbps ανά σύνδεση, παρατηρείται ότι η κίνηση μοιράζεται αρκετά πιο ομοιόμορφα, αν και ακόμα όχι ιδανικά.



Διάγραμμα 4: Διαμοιρασμός κίνησης στις έξι διεπαφές (UDP 500 συνδέσεις 4Mbps/σύνδεση)

6.4 Δοκιμές με παράμετρο το μέγεθος πακέτου για UDP/TCP

6.4.1 Μετρήσεις

Πρωτόκολλο	**Μέγεθος Πακέτου (Bytes)**	**Ταχύτητα Αποστολής (Mbps)**	**Ταχύτητα Λήψης (Mbps)**	**Σφάλμα**
UDP	128	1268	211.5	83%
UDP	256	2000	414.3	79%
UDP	512	2000	847	58%
UDP	768	2000	1591	21%
UDP	1024	2000	1899	5%
UDP	1152	2000	2000	0%
UDP	1472	2000	1980	1%
TCP	536	977	974	
TCP	600	1091	1089	
TCP	800	1448	1445	
TCP	1000	1854	1848	
TCP	1200	1961	1959	
TCP	1460	1927	1920	

Πίνακας 3: Μετρήσεις εύρους ζώνης συναρτήσει του μεγέθους πακέτου

6.4.2 Συμπεράσματα

Διάγραμμα 5:Εύρος ζώνης λήψης συναρτήσει του μεγέθους πακέτου

Είναι φανερό ότι η επίτευξη μεγάλων ταχυτήτων λήψης είναι ανάλογη με το μέγεθος πακέτου. Και για UDP και για TCP, όσο μεγαλώνει το μέγεθος πακέτου τόσο το DANOS επεξεργάζεται λιγότερα πακέτα και άρα επιτυγχάνει μεγαλύτερες ταχύτητες.

Αυτό συμβαίνει, διότι ένα μεγάλο μέρος των καθυστερήσεων κατά την επεξεργασία των πακέτων αφορά την επικεφαλίδα. Για οικονομία θα γίνει αναφορά στην περίπτωση του TCP. Όταν ένα πακέτο έχει μέγεθος 600 bytes, τα 40 από αυτά είναι οι επικεφαλίδες IP και TCP. Υπάρχει δηλαδή ωφέλιμο φορτίο 560/600 bytes. Αντίστοιχα, σε πακέτο 1460 bytes, το ωφέλιμο φορτίο είναι 1420/1460. Επειδή λοιπόν οι καθυστερήσεις αφορούν κυρίως την επικεφαλίδα και αυτή έχει σταθερό μέγεθος, ανεξάρτητο του μεγέθους του πακέτου (για πακέτα χωρίς fragmentation), η καθυστέρηση επεξεργασίας μπορεί να θεωρηθεί ανάλογη του αριθμού των πακέτων που εισέρχονται στο σύστημα.

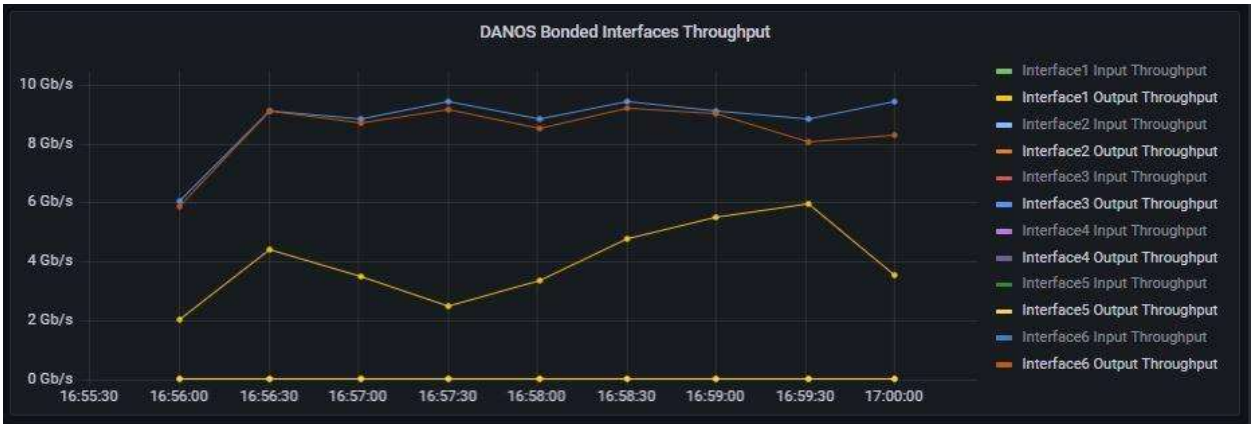
Για να επιτευχθεί ταχύτητα επεξεργασίας 2Gbps με μέγεθος πακέτου 600 bytes χρειάζονται περίπου 417 χιλιάδες πακέτα (2Gbps/(600*8)). Αντίστοιχα για μέγεθος πακέτου 1460 bytes χρειάζονται 171 χιλιάδες πακέτα. Είναι φανερό λοιπόν ότι ένα σύστημα έρχεται αντιμέτωπο με πολύ μεγαλύτερο φόρτο επεξεργασίας, όταν έχει να αντιμετωπίσει μικρά πακέτα, στην περίπτωση που πρέπει να επιτευχθεί ένας στόχος ταχύτητας. Για αυτό το λόγο λοιπόν και στις μετρήσεις παρατηρήθηκε η παραπάνω αναλογία.

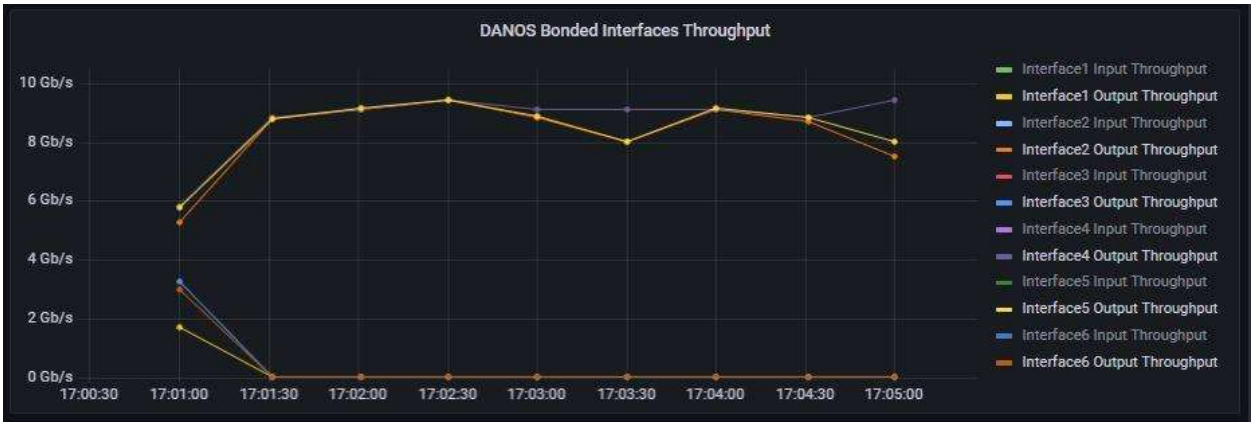
6.5 Ανάδειξη Επιρροής της επιλογής port στο τελικό bandwidth

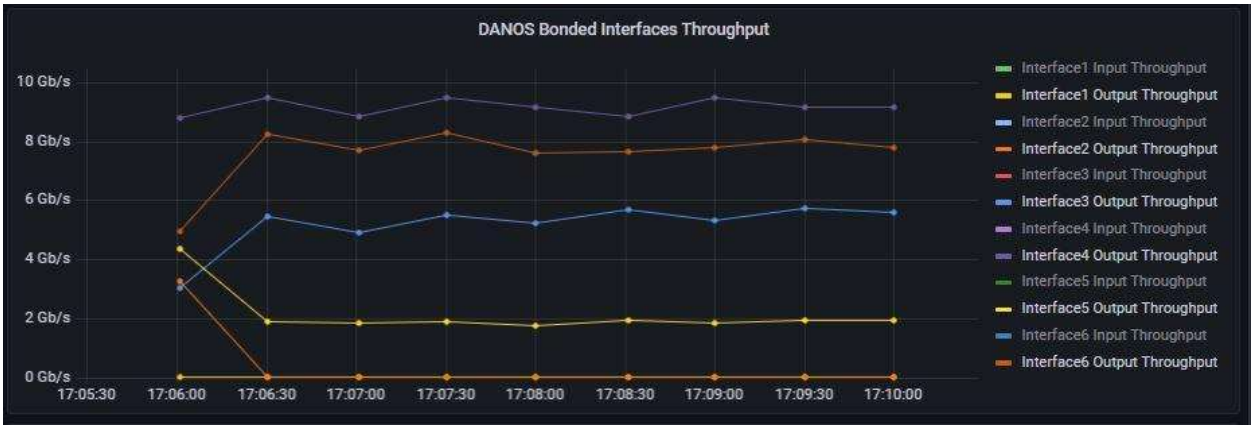
Στις συγκεκριμένες δοκιμές φαίνεται η επιρροή των port που επιλέγονται από το iperf, καθώς αυτά μοιράζονται από το LACP. Επιλέχθηκε μικρός αριθμός συνδέσεων και πιο συγκεκριμένα 10 συνδέσεις, έτσι ώστε να είναι πιο ευδιάκριτες οι διαφορές σε κάθε επανάληψη. Για κάθε δοκιμή οι παράμετροι έχουν τις ίδιες τιμές και πιο συγκεκριμένα: 10 συνδέσεις, 200Mbps/σύνδεση και MSS 1460 Bytes. Παρακάτω φαίνεται ο πίνακας των τιμών για κάθε επανάληψη και έπειτα τα 5 διαγράμματα που αναπαρίσταται η κατανομή του φορτίου σε κάθε μία από τις 6 διεπαφές.

Επανάληψη	Ταχύτητα Αποστολής/Λήψης (TCP)
1	901Mbps
2	1.09Mbps
3	987Mbps
4	1.52Gbps
5	1.02Gbps

Πίνακας 4: Αποτελέσματα εξέτασης τυχαιότητας λόγω LACP


DANOS Bonded Interfaces Throughput
10 Gb/s
8 Gb/s
6 Gb/s
4 Gb/s
2 Gb/s
0 Gb/s
16:55:30
16:56:00
16:56:30
16:57:00
16:57:30
16:58:00
16:58:30
16:59:00
16:59:30
17:00:00
Interface1 Input Throughput
Interface1 Output Throughput
Interface2 Input Throughput
Interface2 Output Throughput
Interface3 Input Throughput
Interface3 Output Throughput
Interface4 Input Throughput
Interface4 Output Throughput
Interface5 Input Throughput
Interface5 Output Throughput
Interface6 Input Throughput
Interface6 Output Throughput


DANOS Bonded Interfaces Throughput
10 Gb/s
8 Gb/s
6 Gb/s
4 Gb/s
2 Gb/s
0 Gb/s
17:00:30
17:01:00
17:01:30
17:02:00
17:02:30
17:03:00
17:03:30
17:04:00
17:04:30
17:05:00
Interface1 Input Throughput
Interface1 Output Throughput
Interface2 Input Throughput
Interface2 Output Throughput
Interface3 Input Throughput
Interface3 Output Throughput
Interface4 Input Throughput
Interface4 Output Throughput
Interface5 Input Throughput
Interface5 Output Throughput
Interface6 Input Throughput
Interface6 Output Throughput


DANOS Bonded Interfaces Throughput
10 Gb/s
8 Gb/s
6 Gb/s
4 Gb/s
2 Gb/s
0 Gb/s
17:05:30
17:06:00
17:06:30
17:07:00
17:07:30
17:08:00
17:08:30
17:09:00
17:09:30
17:10:00
Interface1 Input Throughput
Interface1 Output Throughput
Interface2 Input Throughput
Interface2 Output Throughput
Interface3 Input Throughput
Interface3 Output Throughput
Interface4 Input Throughput
Interface4 Output Throughput
Interface5 Input Throughput
Interface5 Output Throughput
Interface6 Input Throughput
Interface6 Output Throughput

Διάγραμμα 6: Διαμοιρασμός κίνησης στις έξι διεπαφές για κάθε μία από τις επαναλήψεις του Πίνακα 4.

Το θεωρητικό όριο αποστολής είναι τα 2Gbps, καθώς υφίστανται 10 συνδέσεις των 200Mbps. Παρατηρείται ότι για πέντε μόνο εκτελέσεις επιτεύχθηκε ελάχιστη ταχύτητα αποστολής 901Mbps και μέγιστη ταχύτητα αποστολής 1.52Gbps. Παρόλο που δεν επιτεύχθηκε ο στόχος των 2Gbps, μπορεί να επισημανθεί ότι υπάρχει αρκετά μεγάλος παράγοντας τύχης όσον αφορά την κατανομή των ports από το LACP.

Μπορεί μάλιστα να παρατηρηθεί ότι στις τρεις περιπτώσεις χρησιμοποιούνται μόνο τρεις από τις έξι διεπαφές, σε μία περίπτωση τέσσερις μόνο και σε μία περίπτωση πέντε από τις έξι. Παρόλο που δεν αποτελεί καθοριστικό κριτήριο ταχύτητας, αφού με τέσσερις ενεργές διεπαφές δεν επιτεύχθηκε μεγαλύτερη ταχύτητα ,αποτελεί σημαντικό παράγοντα καθώς με τις πέντε ενεργές διεπαφές επιτυγχάνεται η μέγιστη ταχύτητα. Αυτό συμβαίνει διότι πρέπει να υπάρχει συνδυασμός χρήσης πολλών διεπαφών και ίσης κατανομής κίνησης σε αυτές. Στην περίπτωση 4 που επιτυγχάνεται η καλύτερη ταχύτητα παρατηρείται ότι εμφανίζονται πολλές ενεργές διεπαφές και

ταυτόχρονα αρκετές από αυτές βρίσκονται σε υψηλές ταχύτητες. Το φαινόμενο αυτό υποχωρεί όσο αυξάνονται οι συνδέσεις.

Θα ήταν δυνατόν να επιτευχθούν, καλύτερες ταχύτητες αν το iperf δεν περιόριζε όλες τις συνδέσεις στο ίδιο εύρος ζώνης αλλά μόνο αυτές στις οποίες παρουσιάζονται σφάλματα λόγω κορεσμού της γραμμής.

6.6 Δοκιμές σε συνθήκες πραγματικού δρομολογητή

Στις παραπάνω δοκιμές επιτεύχθηκαν πολύ υψηλές ταχύτητες, όμως αυτές αποτελούν ιδανικές τιμές καθώς ο δρομολογητής DANOS δεν έκανε περαιτέρω επεξεργασία και έλεγχο στα πακέτα. Σε πραγματικές συνθήκες, μεταξύ άλλων, οι δρομολογητές διαθέτουν firewalls για να περιορίζουν την κίνηση που διέρχεται από αυτούς . Ένα πακέτο, όταν διέρχεται από ένα τείχος προστασίας, καθυστερεί, καθώς πρέπει να εξεταστεί με βάση τους κανόνες του τείχους προστασίας, μέχρι να ταιριάζει με κάποιο κανόνα.

6.6.1 Προετοιμασία δοκιμών

Όσον αφορά το τείχος προστασίας, γίνονται τρεις δοκιμές με τρία διαφορετικά firewalls με 100, 1000 και 10000 κανόνες αντίστοιχα. Οι κανόνες αυτοί δημιουργήθηκαν από script με σκοπό κάθε πακέτο να περνά όλους τους κανόνες, πριν ταιριάξει με τον τελευταίο κανόνα κάθε φορά. Οι 99, 999 και 9999 πρώτοι κανόνες δηλαδή, δημιουργήθηκαν έτσι ώστε κανένα πακέτο να μην ταιριάξει με αυτούς και να οδηγηθούν στον τελευταίο κανόνα. Επιλέχθηκε αυτή η υλοποίηση, διότι εξετάζεται πόσο επηρεάζει ο έλεγχος κανόνων την ταχύτητα επεξεργασίας των πακέτων. Να σημειωθεί ότι το DANOS παρέχει τείχος προστασίας έως 10000 κανόνες, για αυτό και είναι το μέγιστο που εξετάζεται. Για κάθε κανόνα υπάρχει δυνατότητα μέσω του protocol-group να οριστούν πολλαπλά πρωτόκολλα για τον ίδιο κανόνα. Στην συγκεκριμένη περίπτωση χρησιμοποιείται για κάθε κανόνα και udp και tcp.

Όσον αφορά την τοποθέτηση των τειχών προστασίας, αυτή πρέπει να γίνει στο σύνολο των διεπαφών που αφορούν την κίνηση του πακέτου. Αν εφαρμοστεί το τείχος προστασίας μόνο σε μία εικονική διεπαφή, θα γίνει ταίριασμα κανόνων μόνο για το 1/50 της κίνησης. Ο σκοπός του ελέγχου όμως είναι να εξεταστεί αν το DANOS σε μεγάλες ταχύτητες, που αφορούν το σύνολο των διεπαφών του, καταφέρνει να διατηρήσει τις επιδόσεις του ακόμα και με μεγάλα τείχη προστασίας. Άρα λοιπόν πρέπει οι παράμετροι να εφαρμοστούν στο σύνολο της κίνησης και άρα σε κάθε VRF και vif αντίστοιχα.

6.6.2 Αποτελέσματα Μετρήσεων για τείχη προστασίας 100,100,10000 κανόνων

Μετρήσεις για τείχος προστασίας 100 κανόνων

TCP/UDP	Number of Connections	Bandwidth per Connection	Packet Size Bytes	Iperf3 Sender bandwidth Mbps	iperf3 Reciever bandwidth Mbps	Error Rate (only for UDP)	DANOS Input Traffic Mbps	DANOS Output Traffic Mbps	DANOS bandwidth In/Out (Gbps)	Switch bandwidth In/Out (Gbps)
UDP	1	300	1472	300	300	0%	287	287	7.2	7.2
UDP	1	400	1472	400	367	8%	383	352	9.17	9.2
UDP	1	500	1472	500	297	41%	479	285	9.16	9.2
UDP	2	300	1472	600	597	0.54%	575	575	14.4	14.5
UDP	2	400	1472	800	718	10.00%	766	698	18.3	18.4
UDP	2	500	1472	1000	568	43%	958	547	18.3	18.3
UDP	5	300	1472	1500	573	62%	1440	548	21.8	21.8
UDP	5	400	1472	2000	294	85%	1920	282	18.3-19.9	19.9-18.4
UDP	5	500	1472	2500	123	95%	2390	117	15	15
UDP	10	100	1472	1000	1000	0.00%	958	958	24	24
UDP	10	150	1472	1500	844	44%	1440	809	25.5-26.1	26.2-25.6
UDP	10	200	1472	2000	561	72%	1920	538	23.9-24.8	24.9-24
UDP	10	250	1472	2500	433	83%	2480	399	25.2-25.8	25.8-25.3
UDP	10	300	1472	3000	148	95%	2870	142	17.7	18.1
UDP	20	50	1472	1000	914	8.60%	958	880	22.8	22.8

UDP	20	75	1472	1500	760	49%	1440	726	25.6	25.7
UDP	20	100	1472	2000	1150	43%	1920	1100	36.2	36.3
UDP	20	125	1472	2500	751	70%	2390	718	33.6-34.2	34.3-33.7
UDP	20	150	1472	3000	610	80%	2780	562	33.2-33.5	33.6-33.3
UDP	50	20	1472	1000	1000	0%	958	958	24	24
UDP	50	30	1472	1500	1480	1.10%	1440	1420	35.8	35.9
UDP	50	40	1472	2000	1190	41%	1920	1140	36.9-37	36.9-36.8
UDP	50	50	1472	2500	1010	60%	2400	963	37.9-38.2	38.4-38
UDP	100	10	1472	1000	1000	0%	991	991	24	24
UDP	100	15	1472	1500	1500	0%	1440	1440	36	36.1
UDP	100	20	1472	2000	1000	50%	1920	960	33.9	34.1
UDP	100	25	1472	2500	1020	59%	2400	980	38.4-38.6	38.7-38.5
UDP	200	5	1472	1000	1000	0%	958	958	24	24
UDP	200	7	1472	1400	1400	0%	1340	1340	33.6	33.7
UDP	200	10	1472	2000	1570	22%	1920	1500	42.4	42.5
UDP	200	12	1472	2400	1415	41%	2300	1350	43.9	44
UDP	500	2	1472	1000	1000	0%	958	958	24	24
UDP	500	3	1472	1500	1500	0%	1440	1440	36	36
UDP	500	4	1472	2000	1823	9%	1970	1850	45.7	45.8
UDP	500	5	1472	2500	1586	37%	2390	1520	47.5	47.6
TCP	1	300	1460	300			283	283	7.58	7.61
TCP	1	400	1460	375			365	365	9.16	9.16

TCP	1	500	1460	375			365	365	9.16	9.16
TCP	2	300	1460	600			584	584	14.7	14.7
TCP	2	400	1460	374			365	365	9.16	9.16
TCP	2	500	1460	375			365	365	9.16	9.16
TCP	5	300	1460	697			671	671	16.9	16.7
TCP	5	400	1460	958			983	983	24.7	24.5
TCP	5	500	1460	737			727	727	18.3	18.4
TCP	10	100	1460	743			724	724	18.3	18.3
TCP	10	150	1460	1250			1220	1220	30.6	30.9
TCP	10	200	1460	1020			977	977	25.3	25.3
TCP	10	250	1460	1450			1410	1410	35.4	35.6
TCP	10	300	1460	782			790	790	20.1	20.5
TCP	20	50	1460	922			897	897	22.6	22.5
TCP	20	75	1460	813			792	792	20	20.1
TCP	20	100	1460	1490			1430	1430	36	36.1
TCP	20	125	1460	1270			1260	1260	31.5	31.4
TCP	20	150	1460	1220			1200	1200	30.3	30.3
TCP	50	20	1460	1000			974	974	24.7	25
TCP	50	30	1460	1470			1440	1440	36.3	36.5
TCP	50	40	1460	1720			1680	1680	42.3	42.4
TCP	50	50	1460	1980			1930	1930	48.9	48.9
TCP	100	10	1460	1000			978	978	24.8	24.9
TCP	100	15	1460	1500			1470	1470	37.1	37.2
TCP	100	20	1460	1600			1560	1560	39.4	39.5
TCP	100	25	1460	2060			2002	2002	50.9	51

TCP	200	5	1460	1000			985	985	24.8	25
TCP	200	7	1460	1400			1390	1390	34.8	34.8
TCP	200	10	1460	1852			1800	1800	46.1	46.3
TCP	200	12	1460	2060			2004	2004	51.3	51.3
TCP	500	2	1460	1000			1002	1002	25.9	25.7
TCP	500	3	1460	1500			1500	1500	37.6	37.8
TCP	500	4	1460	1929			1960	1960	48.8	49.1
TCP	500	5	1460	1923			1950	1950	48.7	48.9

Πίνακας 5: Αποτελέσματα για τείχος προστασίας 100 κανόνων

Μετρήσεις για τείχος προστασίας 1000 κανόνων

TCP/ UDP	**Number of Connections**	**Bandwidth per Connection**	**Packet Size** Bytes	**Iperf3 Sender bandwidth** Mbps	**iperf3 Reciever bandwidth** Mbps	**Error Rate (only for UDP)**	**DANOS Input Traffic** Mbps	**DANOS Output Traffic** Mbps	**DANOS bandwidth** In/Out (Gbps)	**Switch bandwidth** In/Out (Gbps)
UDP	1	300	1472	300	300	0%	287	287	7.2	7.2
UDP	1	400	1472	400	368	8%	383	353	9.17	9.2
UDP	1	500	1472	500	303	39%	479	291	9.17	9.2
UDP	2	300	1472	600	597	0.54%	575	575	14.4	14.5
UDP	2	400	1472	800	718	10.00%	766	697	18.3	18.4
UDP	2	500	1472	1000	594	41%	958	571	18.3	18.3
UDP	5	300	1472	1500	397	74%	1440	380	18.7	18.7

UDP	5	400	1472	2000	401	80%	1920	384	21.7	21.8
UDP	5	500	1472	2500	340	86%	2390	327	24	24.1
UDP	10	100	1472	1000	583	42.00%	958	558	18.2	18.2
UDP	10	150	1472	1500	1070	28%	1440	1030	30.5	30.6
UDP	10	200	1472	2000	706	65%	1920	680	28.1-28.5	28.7-28.4
UDP	10	250	1472	2500	1180	53%	2390	1130	41.2	41.3
UDP	10	300	1472	3000	386	87%	2870	370	27.8	28
UDP	20	50	1472	1000	1000	0.00%	958	958	24	24
UDP	20	75	1472	1500	1040	31%	1440	996	30.1	30
UDP	20	100	1472	2000	513	74%	1920	489	24.9	25.1
UDP	20	125	1472	2500	884	65%	2390	846	36.5	36.4
UDP	20	150	1472	3000	276	91%	2870	260	24	24.1
UDP	50	20	1472	1000	1000	0%	958	958	24	24
UDP	50	30	1472	1500	1500	25.00%	1440	1440	36	36.1
UDP	50	40	1472	2000	1500	25%	1920	1430	41.5	41.5
UDP	50	50	1472	2500	1170	53%	2390	1120	41	41.1
UDP	100	10	1472	1000	1000	0%	958	958	24	24.1
UDP	100	15	1472	1500	1500	0%	1440	1440	36	36.1
UDP	100	20	1472	2000	937	53%	1920	897	33	33.2
UDP	100	25	1472	2500	1450	42%	2390	1390	45.4	45.6
UDP	200	5	1472	1000	1000	0%	958	958	24	24.1
UDP	200	7	1472	1400	1400	0%	1340	1340	33.6	33.7
UDP	200	10	1472	2000	1921	4%	1920	1840	47.1	47.2
UDP	200	12	1472	2400	1064	55%	2300	1020	38.5	38.7

UDP	500	2	1472	1000	1000	0%	958	958	24	24.1
UDP	500	3	1472	1500	1500	0%	1440	1440	36	36.1
UDP	500	4	1472	2000	1986	1%	1920	1920	47.9	47.9
UDP	500	5	1472	2500	1467	41%	2390	1400	46	46.1
TCP	1	300	1460	300			292	292	7.33	7.36
TCP	1	400	1460	375			365	365	9.17	9.19
TCP	1	500	1460	375			365	365	9.16	9.2
TCP	2	300	1460	600			584	584	14.7	14.7
TCP	2	400	1460	374			365	365	9.16	9.19
TCP	2	500	1460	374			365	365	9.16	9.19
TCP	5	300	1460	1050			1020	1020	25.7	25.7
TCP	5	400	1460	997			998	998	25	25.1
TCP	5	500	1460	957			922	922	23.2	23.2
TCP	10	100	1460	720			708	708	17.8	17.6
TCP	10	150	1460	1200			1160	1160	29.3	29.4
TCP	10	200	1460	845			833	833	21	20.9
TCP	10	250	1460	1720			1670	1670	42.1	42.2
TCP	10	300	1460	824			818	818	20.6	20.2
TCP	20	50	1460	986			963	963	24.2	24.2
TCP	20	75	1460	1140			1110	1110	27.9	28
TCP	20	100	1460	1130			1110	1110	27.6	27.9
TCP	20	125	1460	1190			1180	1180	29.7	29.3
TCP	20	150	1460	938			919	919	23.2	23.1
TCP	50	20	1460	1000			979	979	24.7	24.8
TCP	50	30	1460	1250			1210	1210	30.7	30.8

TCP	50	40	1460	1820			1770	1770	44.9	44.9
TCP	50	50	1460	1880			1840	1840	46.3	46.5
TCP	100	10	1460	1000			981	981	24.8	24.9
TCP	100	15	1460	1470			1430	1430	36.2	36.4
TCP	100	20	1460	1860			1830	1830	45.9	46.1
TCP	100	25	1460	1960			1920	1920	48.5	48.6
TCP	200	5	1460	1000			975	975	24.8	25
TCP	200	7	1460	1400			1360	1360	34.7	34.8
TCP	200	10	1460	1932			1930	1930	48.5	48.7
TCP	200	12	1460	1908			1890	1860	47.5	47.7
TCP	500	2	1460	1000			993	993	25.1	25.4
TCP	500	3	1460	1500			1490	1490	37.5	37.6
TCP	500	4	1460	1813			1810	1810	45.2	45.4
TCP	500	5	1460	1767			1790	1730	44.7	44.9

Πίνακας 6: Αποτελέσματα μετρήσεων για τείχος προστασίας 1000 κανόνων

Μετρήσεις για τείχος προστασίας 10000 κανόνων

TCP/UDP	**Number of Connections**	**Bandwidth per Connection**	**Packet Size** Bytes	**Iperf3 Sender bandwidth** Mbps	**iperf3 Reciever bandwidth** Mbps	**Error Rate (only for UDP)**	**DANOS Input Traffic** Mbps	**DANOS Output Traffic** Mbps	**DANOS bandwidth** In/Out (Gbps)	**Switch bandwidth** In/Out (Gbps)
UDP	1	300	1472	300	300	0%	287	287	7.2	7.22
UDP	1	400	1472	400	366	9%	383	351	9.17	9.2
UDP	1	500	1472	500	287	43%	479	279	9.17	9.2

UDP	2	300	1472	600	596	0.70%	575	575	14.4	14.5
UDP	2	400	1472	800	564	29.00%	766	544	16.1	16.1
UDP	2	500	1472	1000	135	87%	958	130	9.17-10	10-9.2
UDP	5	300	1472	1500	285	81%	1440	263	16.3	16.3
UDP	5	400	1472	2000	190	90%	1920	181	15.5	15.6
UDP	5	500	1472	2500	350	86%	2390	336	22.5	22.5
UDP	10	100	1472	1000	731	27.00%	958	701	19.5	19.6
UDP	10	150	1472	1500	407	73%	1440	389	18.9	18.9
UDP	10	200	1472	2000	587	71%	1920	562	25.1	25.3
UDP	10	250	1472	2500	654	74%	2390	626	29.3-30.7	30.8-29.4
UDP	10	300	1472	3000	250	92%	2870	240	20.6-21.3	21.4-20.7
UDP	20	50	1472	1000	1000	0.00%	958	958	24	24.1
UDP	20	75	1472	1500	903	40%	1440	865	27.7-28.0	28.0-27.8
UDP	20	100	1472	2000	967	52%	1920	940	29.7	29.8
UDP	20	125	1472	2500	728	71%	2390	600	29.9	30.1
UDP	20	150	1472	3000	335	89%	2870	320	26.8	26.9
UDP	50	20	1472	1000	1000	0%	957	957	24	24.1
UDP	50	30	1472	1500	1110	26.00%	1440	1070	28.5	28.6
UDP	50	40	1472	2000	999	50%	1920	954	28.8	29
UDP	50	50	1472	2500	926	63%	2390	891	29.3	29.4
UDP	100	10	1472	1000	1000	0%	958	958	24	24.1
UDP	100	15	1472	1500	1080	28%	1440	1030	27.8	27.9
UDP	100	20	1472	2000	1030	48%	1850	990	28.2	28.3

UDP	100	25	1472	2500	951	62%	2390	913	28.4	28.4
UDP	200	5	1472	1000	1000	0%	958	958	24	24.1
UDP	200	7	1472	1400	1074	23%	1340	1030	27.2	27.3
UDP	200	10	1472	2000	970	52%	1920	923	27.1	27.1
UDP	200	12	1472	2400	934	61%	2300	885	26.9	27.2
UDP	500	2	1472	1000	1000	0%	958	958	24	24
UDP	500	3	1472	1500	1077	28%	1440	1030	27.3	27.4
UDP	500	4	1472	2000	999	50%	1920	955	27.6	27.6
UDP	500	5	1472	2500	922	63%	2390	886	27.5	27.6
TCP	1	300	1460	300			292	292	7.33	7.36
TCP	1	400	1460	375			365	365	9.16	9.19
TCP	1	500	1460	375			365	365	9.16	9.19
TCP	2	300	1460	375			365	365	9.16	9.19
TCP	2	400	1460	674			654	654	16.4	16.6
TCP	2	500	1460	372			362	362	9.1	9.13
TCP	5	300	1460	374			364	364	9.16	9.19
TCP	5	400	1460	734			711	711	17.9	18.1
TCP	5	500	1460	743			725	725	18.3	18.3
TCP	10	100	1460	836			816	816	20.5	20.7
TCP	10	150	1460	726			718	718	18.1	18
TCP	10	200	1460	610			632	632	15.5	15
TCP	10	250	1460	778			749	749	18.9	18.9
TCP	10	300	1460	1050			1030	1030	26	26.2
TCP	20	50	1460	961			937	937	23.6	23.6
TCP	20	75	1460	995			976	976	24.6	24.9

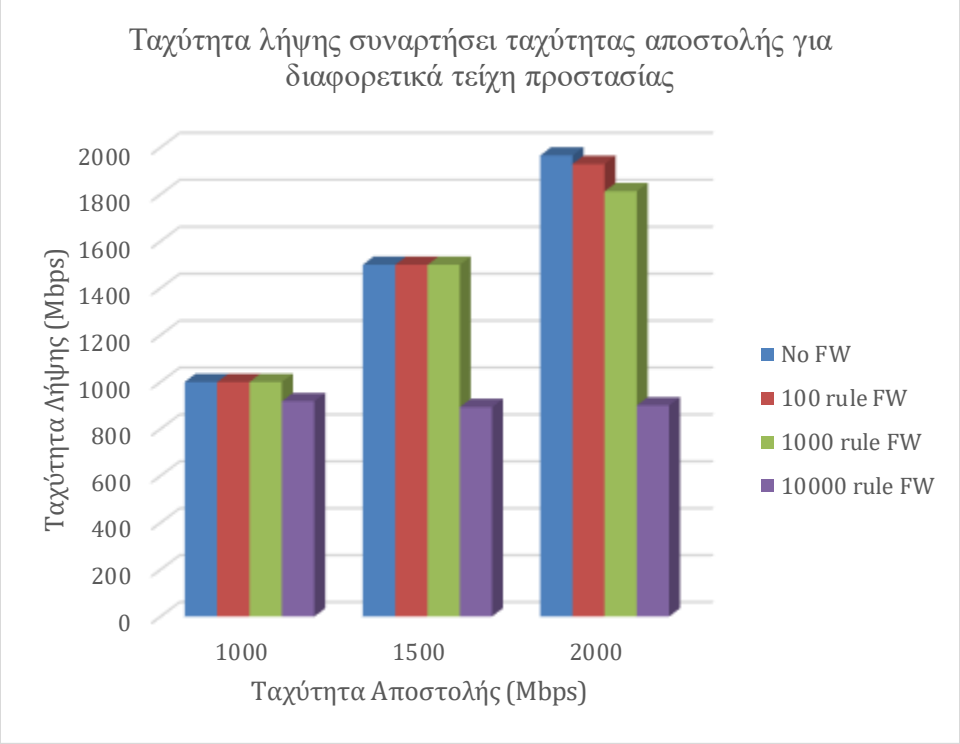
TCP	20	100	1460	840			816	816	20.5	20.7
TCP	20	125	1460	946			932	932	23.5	23.6
TCP	20	150	1460	371			348	348	8.84	8.76
TCP	50	20	1460	999			974	974	24.6	24.7
TCP	50	30	1460	1080			1050	1050	26.5	26.7
TCP	50	40	1460	1070			1040	1040	26.2	26.3
TCP	50	50	1460	1090			1060	1060	26.7	26.9
TCP	100	10	1460	1000			1000	1000	25.8	25.8
TCP	100	15	1460	1030			1010	1010	25.4	25.6
TCP	100	20	1460	1010			981	981	24.7	24.8
TCP	100	25	1460	1030			995	995	25.2	25.3
TCP	200	5	1460	970			944	944	23.9	24
TCP	200	7	1460	970			933	933	23.7	23.9
TCP	200	10	1460	972			946	946	23.9	23.8
TCP	200	12	1460	976			947	947	23.9	24
TCP	500	2	1460	918			897	897	22.7	22.7
TCP	500	3	1460	893			874	865	22	22
TCP	500	4	1460	900			879	869	22.1	22.2
TCP	500	5	1460	891			870	861	21.9	22

Πίνακας 7: Αποτελέσματα μετρήσεων για τείχος προστασίας 10000 κανόνων

6.6.3 Συμπεράσματα Μετρήσεων

Όπως αναφέρθηκε και παραπάνω, κάθε επανάληψη δοκιμής με ίδιες παραμέτρους, μπορεί να φέρει διαφορετικά αποτελέσματα ,καθώς εξαρτάται από την διανομή των port από το LACP στις διεπαφές. Για αυτό, παρατηρούνται ,με ίδιες παραμέτρους ,πολλές φορές καλύτερες επιδόσεις σε τείχη προστασίας με περισσότερους κανόνες ή παρατηρούνται χειρότερες επιδόσεις σε τείχη προστασίας με λιγότερους κανόνες.

Για λίγες συνδέσεις δεν γίνεται να επιτευχθούν υψηλές ταχύτητες, καθώς δεν αξιοποιούνται πλήρως οι διεπαφές (κάτω από 6 συνδέσεις). Σε ενδιάμεσες συνδέσεις επιτυγχάνονται καλύτερες επιδόσεις αλλά υπάρχει και πάλι περιορισμός, λόγω της ανομοιομορφίας διανομής των θηρών.



Διάγραμμα 7: Ταχύτητα λήψης συναρτήσει ταχύτητας αποστολής για διαφορετικά τείχη προστασίας

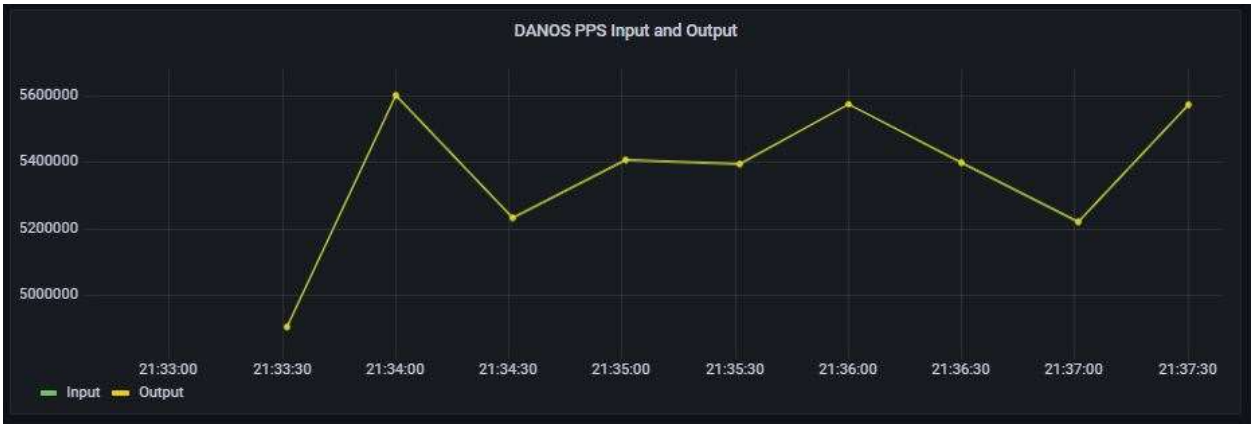
Τείχος προστασίας 100 κανόνων: Παρατηρούνται επιδόσεις αντίστοιχες αυτών χωρίς τείχος προστασίας, έως 51.3 Gbps ανά κατεύθυνση.

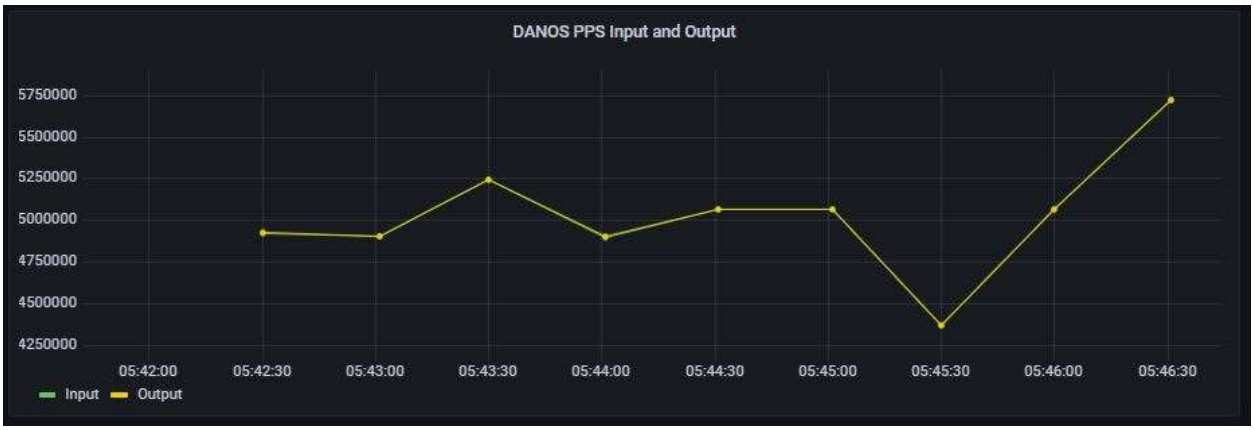
Τείχος προστασίας 1000 κανόνων: Παρατηρούνται επίσης επιδόσεις αντίστοιχες αυτών χωρίς τείχος προστασίας, που φτάνουν τα 48.5 Gbps ανά κατεύθυνση. Η επίδοση αυτή θεωρείται στα πλαίσια του σφάλματος και βλέποντας την πλήρη εικόνα των αποτελεσμάτων, είναι φανερό ότι το συγκεκριμένο τείχος προστασίας δεν επηρεάζει τις επιδόσεις ή τις επηρεάζει σε αμελητέο βαθμό.

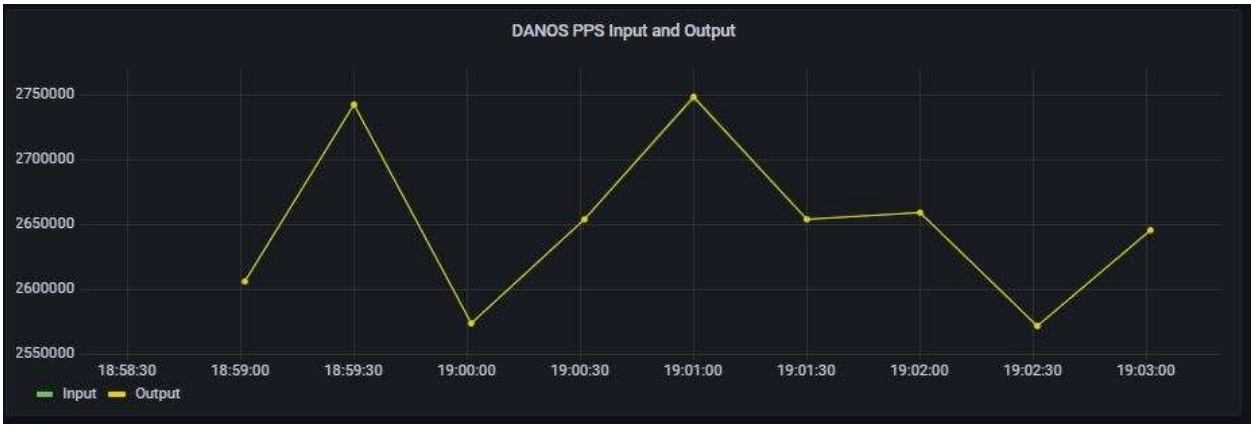
Τείχος προστασίας 10000 κανόνων: Σε δοκιμές με στόχο περί τα 1Gbps αποστολής, δηλαδή περί τα 25Gbps επεξεργασίας από το DANOS ανά κατεύθυνση, δεν παρουσιάζεται μείωση. Σε δοκιμές όπου ο στόχος είναι άνω του 1Gbps, το DANOS φτάνει σε ένα όριο που δεν μπορεί να ξεπεράσει. Παρατηρείται από τις δοκιμές ότι στην στήλη DANOS Output traffic ,που είναι η στήλη που αναφέρεται στην κίνηση που θα φτάσει στο μηχάνημα Ubuntu2, επιτυγχάνονται μέγιστες τιμές

περί τα 1 , 1.1 Gbps. Αυτό αντιστοιχεί σε μέγιστες ταχύτητες 26-27 Gbps ανά κατεύθυνση επεξεργασίας από το DANOS. Συμπεραίνεται άρα, ότι λόγω του τείχους προστασίας των 10000 κανόνων, το DANOS δεν μπορεί να επεξεργαστεί πακέτα με ταχύτητες άνω των 26-27 Gbps ανά κατεύθυνση . Αυτό αποτελεί μια πτώση επιδόσεων της τάξης του 50% από τις επιδόσεις κοντά στα 50-53 Gbps δίχως τείχος προστασίας.

Παρατίθενται τα διαγράμματα PPS για τις καλύτερες επιδόσεις, για κάθε τείχος προστασίας:







Διάγραμμα 8: PPS (πακέτα ανά δευτερόλεπτο) για τις καλύτερες επιδόσεις για κάθε τείχος προστασίας

Παρατηρείται και από τα διαγράμματα για τα PPS που επεξεργάζεται το DANOS στην bonded γραμμή, ότι για 10000 κανόνες υπάρχει περιορισμός στα 2650000 πακέτα ανά δευτερόλεπτο. Αντίθετα για 100 και 1000 κανόνες επιτυγχάνεται επεξεργασία σε 5400000 και 5050000 πακέτα ανά δευτερόλεπτο.

Κεφάλαιο 7 : CGNAT

7.1 Εισαγωγή

Στις πρώτες μέρες του internet, κάθε υπολογιστής συνδεδεμένος στο διαδίκτυο είχε την δική του δημόσια IP διεύθυνσης. Η διεύθυνση αυτή, στην έκδοση 4 ορίζεται από τέσσερα οκτέτα, τέσσερις ομάδες οκτώ bits, το πρωτόκολλο IPv4. Με αυτόν τον αριθμό από bits υπήρχε η δυνατότητα διευθυνσιοδότησης περισσότερων από 4 δις διευθύνσεων (για την ακρίβεια 4294967296), οπότε για την εποχή εκείνη ήταν υπεραρκετές.

Από τα τέλη της δεκαετίας του 80 όμως, έγινε εμφανές ότι η δραματική αύξηση τις υιοθέτησης διευθύνσεων από υπολογιστές εν τέλει θα ξεπερνούσε σε αριθμό αυτόν των διαθέσιμων διευθύνσεων του IPv4. Ως λογική συνέχεια λοιπόν, δημιουργήθηκε το πρωτόκολλο IPv6 που διαθέτει διευθύνσεις 128 bits. Παρόλα αυτά, το IPv6 δεν υποστηρίζει συμβατότητα προς τα πίσω και άρα δημιουργήθηκε ανάγκη για έναν τρόπο ώστε να μπορέσουν να χρησιμοποιηθούν η υπάρχουσα δομή και πρωτόκολλα. Έτσι δημιουργήθηκε το Carrier Grade NAT ως λύση στο παραπάνω πρόβλημα, κυρίως για τους παρόχους υπηρεσιών διαδικτύου.[34]

Η παραδοσιακή μετάφραση NAT (standard nat) υπήρχε ως πρωτόκολλο για μετάφραση πολλών ιδιωτικών διευθύνσεων σε μια και μοναδική δημόσια διεύθυνση. Αυτό λειτουργούσε σε εταιρικά δίκτυα αλλά με την επέκταση του σε οικιακά δίκτυα και δίκτυα κινητής τηλεφωνίας η εξάντληση των διευθύνσεων ip δημιούργησε την ανάγκη για μια νέα επέκταση, το CGNAT.

Το CGNAT προσθέτει ένα παραπάνω επίπεδο μετάφρασης. Έτσι οι πάροχοι μπορούν να επεξεργάζονται την κίνηση στο εσωτερικό ιδιωτικό τους δίκτυο, διατηρώντας παράλληλα την δημόσια διεύθυνσή τους.

Σε συνδυασμό με την μετάφραση διευθύνσεων, υπάρχει η επέκταση του NAT Port Address Translation (PAT) η οποία μεταφράζει και τις θύρες, παρέχοντας την δυνατότητα εξυπηρέτησης

[34] A10 networks, 'What is Carrier-grate NAT (CGN/CGNAT)?', *a10 networks* [ιστοσελίδα], https://www.a10networks.com/glossary/what-is-carrier-grade-nat-cgn-cgnat/, (τελευταία πρόσβαση 30 Ιανουαρίου 2023)

πολλαπλών συσκευών με μία διεύθυνση IP. Σε συνδυασμό με την NAT μετάφραση παρέχει μεγαλύτερο εύρος διαθέσιμων διευθύνσεων.[35]

Στην συγκεκριμένη περίπτωση το DANOS παρέχει δυνατότητα για CGNAT που μεταφράζει διευθύνσεις IP σε συνδυασμό με TCP/UDP ports.

7.2 Εργαλεία

7.2.1 Tcpdump

Το tcpdump[36] είναι ένα εργαλείο της γραμμής εντολών του Linux που εκτυπώνει το περιεχόμενο των πακέτων διαδικτυακής κίνησης, σε κάποια διεπαφή, που ταιριάζει με την ορισμένη λογική έκφραση από τον χρήστη. Μπορεί να αποθηκεύει την καταγεγραμμένη κίνηση σε αρχείο και να διαβάζει σε κίνηση από αρχείο. Χρησιμοποιείται για τον έλεγχο της δικτυακής κίνησης και λύση τυχόν προβλημάτων του δικτύου.

Στα πλαίσια της διπλωματικής χρησιμοποιείται στο DANOS για τον έλεγχο της μετάφρασης διευθύνσεων από το CGNAT και εντοπισμό και διόρθωση τυχόν προβλημάτων.

7.2.2 Wireshark

Το wireshark[37] είναι ένα λογισμικό ανοιχτού κώδικα που καταγράφει και αναλύει πρωτόκολλα διαδικτύου. Υποστηρίζει εκατοντάδες πρωτόκολλα, υποστηρίζεται από πολλές πλατφόρμες (Windows,Linux,macOS,Solaris,FreeBSD κα). Παρέχει GUI σε αντίθεση με το tcpdump και έχει την δυνατότητα επίσης να κάνει ανάλυση της κίνησης και αποκωδικοποίηση δεδομένων, εφόσον υπάρχουν τα αντίστοιχα κλειδιά κρυπτογράφησης. Μπορεί να επεξεργάζεται αρχεία από διάφορες πηγές, όπως tcpdump,pcap ng, Microsoft network monitor κα.

Στα πλαίσια της διπλωματικής χρησιμοποιείται στην πηγή κίνησης ,Ubuntu1, για τον έλεγχο της παραγόμενης κίνησης από το Ostinato.

---

[35] Cisco, 'Port Address Translation', *cisco* [ιστοσελίδα], https://www.cisco.com/assets/sol/sb/RV320_Emulators/RV320_Emulator_v1-1-0-09/help/Setup13.html, (τελευταία πρόσβαση 30 Ιανουαρίου 2023)

[36] Linux Foundation, 'tcpdump man page', *tcpdump man page* [ιστοσελίδα], https://linux.die.net/man/8/tcpdump, (τελευταία πρόσβαση 21 Φεβρουαρίου 2023)

[37] Wireshark Foundation, 'The world's most popular network protocol analyzer', *Wireshark* [ιστοσελίδα], https://www.wireshark.org/, (τελευταία πρόσβαση 21 Φεβρουαρίου 2023)

7.2.3 Ostinato

Το Ostinato[38] είναι ένα λογισμικό ανοιχτού κώδικα και χρησιμοποιείται για να παράγει κίνηση διαδικτύου. Υποστηρίζει πλήθος πρωτοκόλλων, φιλικό GUI και Python API για αυτοματοποιημένους ελέγχους του διαδικτύου. Δημιουργήθηκε έχοντας πρότυπο το wireshark, υλοποιώντας όμως την αντίστροφη λειτουργία δηλαδή αυτή της παραγωγής κίνησης.

Στα πλαίσια της διπλωματικής χρησιμοποιείται για την παραγωγή δικτυακής κίνησης και συγκεκριμένα με διαφορετικές διευθύνσεις και πόρτες, οι οποίες χρησιμοποιούνται για τον έλεγχο της μετάφρασης CGNAT από το DANOS.

7.3 CGNAT configuration στο DANOS

Όπως έχει υπογραμμιστεί και παραπάνω σε δοκιμές, η κίνηση από το Ubuntu1 (source) προς το DANOS αφορά κίνηση της τάξεως των 1-2 Gbps που λόγω των ανακυκλώσεων φτάνει τα 40-55 Gbps. Σε ένα κλασσικό configuration για CGNAT θα αρκούσε να οριστεί ένα υποδίκτυο για το οποίο θα γίνει η μετάφραση και μια οικογένεια διευθύνσεων και θυρών για την μετάφραση. Το πρώτο υποδίκτυο θα αφορούσε τις ιδιωτικές διευθύνσεις και το δεύτερο δημόσιες, μικρότερες σε πλήθος.

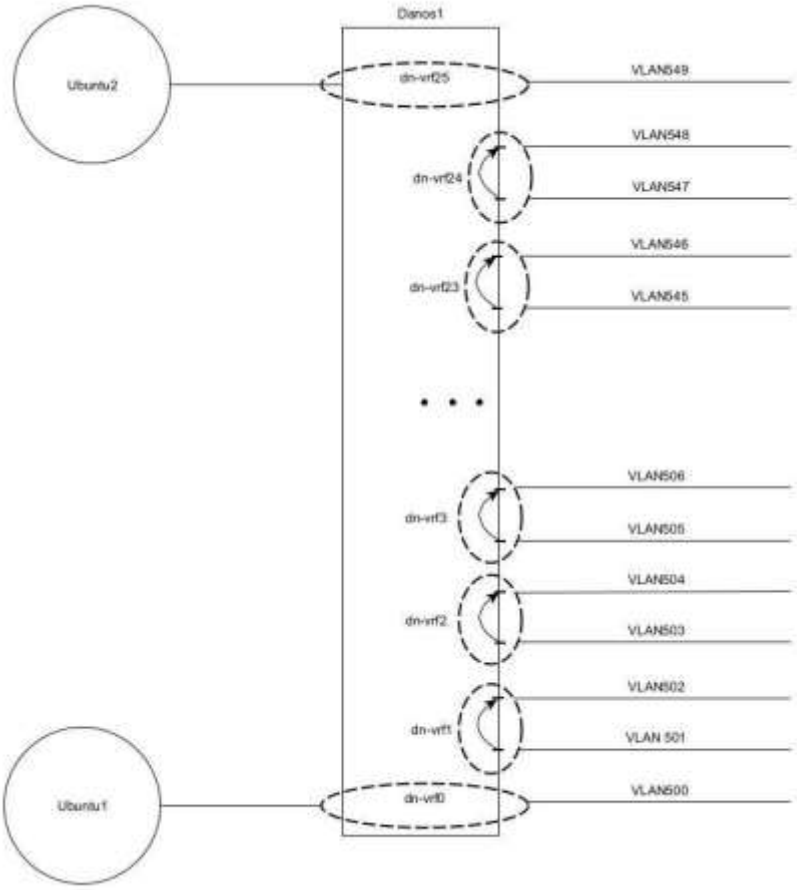
Επειδή δεν διατίθενται δημόσιες διευθύνσεις και δεν επηρεάζει τις συγκεκριμένες δοκιμές, καθώς δεν υπάρχει ανάγκη επαφής με το διαδίκτυο, και οι δύο πλευρές της μετάφρασης αφορούν ιδιωτικές διευθύνσεις. Πιο συγκεκριμένα χρησιμοποιούνται τα δίκτυα 192.168.1.0/24 και 172.30.1.0/24

Λόγω των ανακυκλώσεων λοιπόν, δεν αρκεί να εφαρμοστεί μια μετάφραση, απλώς στην κίνηση που προέρχεται από το Ubuntu, καθώς αυτό αποτελεί μικρό μέρος της κίνησης. Αντίστοιχο πρόβλημα να σημειωθεί αναφέρθηκε και στα τείχη προστασίας. Στην συγκεκριμένη περίπτωση λοιπόν η λύση στο πρόβλημα δίνεται από την τοποθέτηση CGNAT σε κάθε εικονική διεπαφή του DANOS που αντιστοιχεί σε κάθε VLAN ανάμεσα στο DANOS και Switch. Έτσι γίνεται μετάφραση σε κάθε κίνηση που διέρχεται από ένα ζεύγος διεπαφών και υλοποιείται μετάφραση για το σύνολο της κίνησης. Να σημειωθεί ότι στο DANOS η μετάφραση γίνεται κατά την

[38] P. Srivats, 'About', *Ostinato traffic generator for Network Engineers* [ιστοσελίδα], https://ostinato.org/about, (τελευταία πρόσβαση 21 Φεβρουαρίου 2023)

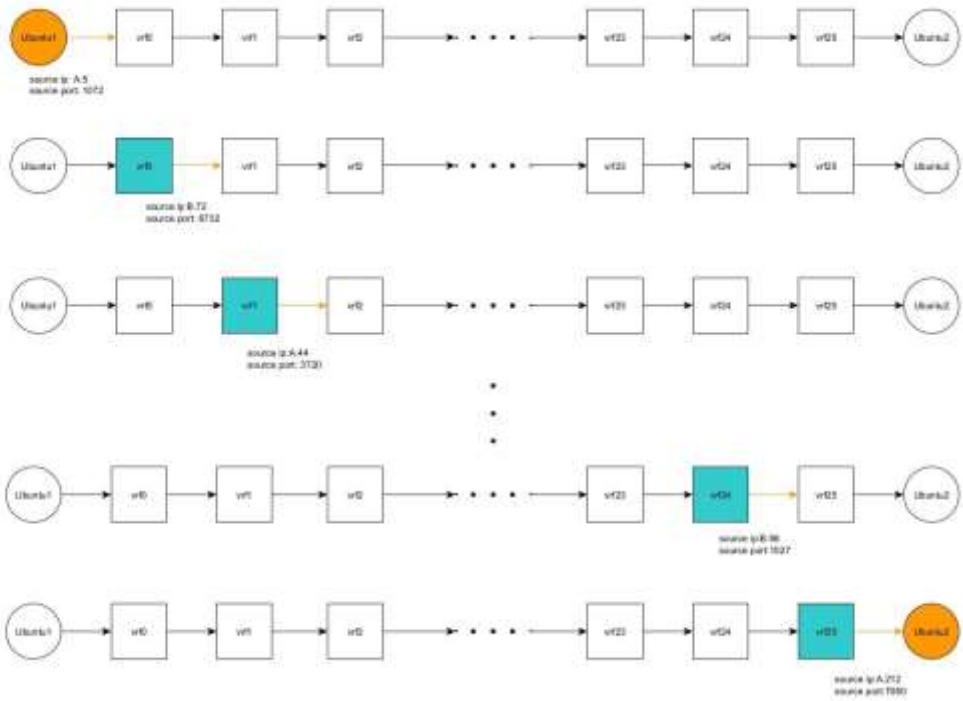
αποστολή πακέτου (outbound), όπου γίνεται ταίριασμα της διεύθυνσης πηγής με την πολιτική του CGNAT.

Λόγω των συνολικά 26 μεταφράσεων από διαφορετικά CGNAT που γίνονται, κατά την διάρκεια της πορείας του πακέτου προς τον προορισμό ,δεν μπορεί να χρησιμοποιηθεί μετάφραση σε μικρότερο υποδίκτυο. Εκτός από ότι μπορεί να ξεμείνει κάποιο CGNAT από διαθέσιμες διευθύνσεις, τίθεται και το θέμα των επιδόσεων. Με τη χρήση πολλών CGNAT, στόχος είναι η επίτευξη αλλαγής διευθύνσεων σε πολύ μεγάλη κίνηση, ώστε να φανεί πώς ανταποκρίνεται το μηχάνημα. Για τους παραπάνω λόγους, επιλέγονται ίδιου μεγέθους υποδίκτυα κατά την μετάφραση(τα 192.168.1.0/24 και 172.30.1.0/24) και μάλιστα τα ίδια για όλα τα CGNAT, εναλλάξ, για εξοικονόμηση χρήσης υποδικτύων. Πιο συγκεκριμένα κάθε φορά που γίνεται μετάφραση σε μια διεπαφή από το υποδίκτυο A στο υποδίκτυο B, στην επόμενη ανακύκλωση γίνεται μετάφραση από το υποδίκτυο B στο υποδίκτυο A. Κατά την αντίθετη διαδρομή για το ενδεχόμενο απάντησης του Ubuntu2 προς το Ubuntu1 γίνεται αντίστροφη μετάφραση, μέσω των ήδη υπάρχοντων CGNAT. Για να γίνει πιο σαφές, αξίζει να παρατηρηθεί στην τοπολογία το μέρος του DANOS.



Εικόνα 32: Οπτική τοπολογίας από την πλευρά του DANOS

Κάθε ανακύκλωση, δηλαδή κάθε είσοδος και έξοδος του πακέτου σε vrf, μπορεί να ερμηνευθεί ως το πέρασμα του πακέτου από ένα δρομολογητή. Σε κάθε έξοδο του δρομολογητή, επειδή υφίσταται outbound CGNAT, γίνεται μετάφραση από το υποδίκτυο A στο B ή από το υποδίκτυο B στο A. Με την παραπάνω σκέψη, οι μεταφράσεις στις οποίες υπόκειται το πακέτο κατά την διάρκεια της πορείας του φαίνονται αναλυτικά στο παρακάτω σχήμα.



Εικόνα 33: Ξεδίπλωμα της ανακύκλωσης σε επιμέρους vrf, πορεία και μετάφραση του πακέτου από την πηγή στον προορισμό

Κάθε τετράγωνο αφορά μία ανακύκλωση και ουσιαστικά απεικονίζει την οντότητα του vrf. Προσοχή όμως, η μετάφραση του CGNAT δεν αφορά τα vrf, καθώς ταυτίζεται με διεπαφές, δηλαδή κάθε πολιτική μετάφρασης εφαρμόζεται σε συγκεκριμένη διεπαφή, και μάλιστα στην διεπαφή εξόδου του πακέτου. Τα μπλε γραμμοσκιασμένα τετράγωνα αφορούν το μέρος που γίνεται η μετάφραση της διεύθυνσης πηγής και της πόρτας πηγής. Στα πορτοκαλί γραμμοσκιασμένα μέρη αναδεικνύεται η διεύθυνση που έχει το πακέτο, όταν τα διασχίζει.

Να σημειωθεί ότι το παραπάνω αφορά ένα παράδειγμα μετάφρασης και συγκεκριμένα οι αλλαγές πορτών επιλέγονται ως τυχαίες. Στην παραμετροποίηση της διπλωματικής επιλέχθηκε η διαδοχική ανάθεση θηρών.

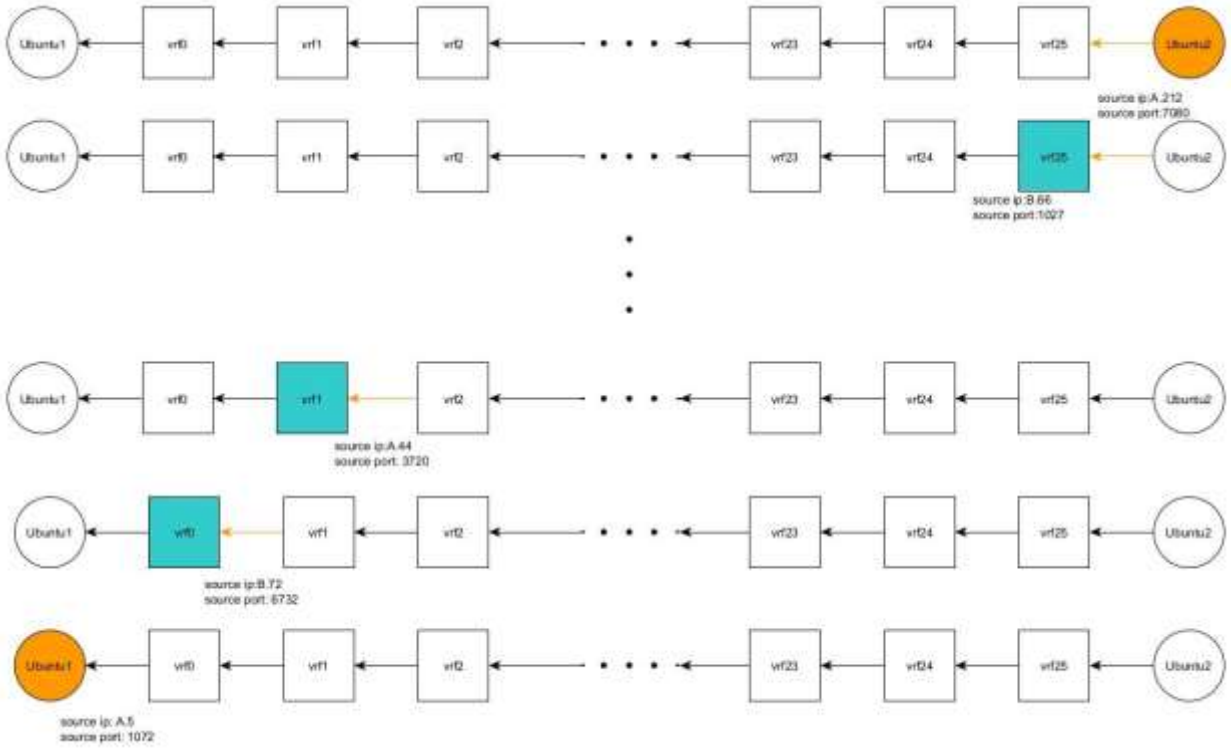
Όπως φαίνεται στο σχήμα, σε κάθε έξοδο από διεπαφή του DANOS γίνεται μια μετάφραση διεύθυνσης και πόρτας σύμφωνα με την πολιτική του CGNAT. Πιο συγκεκριμένα, από υποδίκτυο A σε B και B σε A και κάθε φορά αλλαγή πόρτας σε μία άλλη τυχαία. Παρακάτω φαίνονται οι πίνακες που κρατάει το κάθε CGNAT για να μπορεί να επαναφέρει τις διευθύνσεις και τις πόρτες κατά τις ενδεχόμενες απαντήσεις του προορισμού. Επειδή κάθε φορά που γίνεται μετάφραση το

CGNAT αποθηκεύει τις αντιστοιχίσεις (βλ. πίνακα 8),κατά την διάρκεια της απάντησης από τον προορισμό, συντελείται επαναφορά όλων των ενδιάμεσων διευθύνσεων με τη σωστή σειρά.

			Subscriber		Public			Destination				
ID	Proto	State	Address	Port	Address	Port	Intf	Address	Port	Timeout	PktOut	PktIn
1303279.0	udp	OP	192.168.1.199	1024	172.30.1.137	1024	dp0bond0.516	-	7830	30	182	0
865548.0	udp	CL	192.168.1.199	1024	172.30.1.122	1024	dp0bond0.504	-	7830	0	432	0
1304149.0	udp	OP	192.168.1.66	1036	172.30.1.235	1026	dp0bond0.508	-	7830	30	182	0
1307989.0	udp	OP	192.168.1.20	1942	172.30.1.223	1043	dp0bond0.500	-	7830	30	181	0
870717.0	udp	CL	192.168.1.20	1942	172.30.1.223	1033	dp0bond0.500	-	7830	0	431	0
865641.0	udp	CL	192.168.1.194	1024	172.30.1.148	1024	dp0bond0.504	-	7830	0	432	0
1306279.0	udp	OP	172.30.1.29	1040	192.168.1.215	1040	dp0bond0.502	-	7830	30	182	0
1303447.0	udp	OP	172.30.1.142	1024	192.168.1.98	1024	dp0bond0.518	-	7830	30	182	0
865764.0	udp	CL	172.30.1.142	1024	192.168.1.57	1024	dp0bond0.506	-	7830	0	432	0
1307508.0	udp	OP	172.30.1.169	1038	192.168.1.87	1040	dp0bond0.510	-	7830	30	182	0
1302128.0	udp	OP	192.168.1.26	1148	172.30.1.229	1035	dp0bond0.500	-	7830	30	182	0
865469.0	udp	CL	192.168.1.26	1148	172.30.1.229	1025	dp0bond0.500	-	7830	0	432	0
1304589.0	udp	OP	192.168.1.8	1530	172.30.1.211	1039	dp0bond0.500	-	7830	30	182	0
868137.0	udp	CL	192.168.1.8	1530	172.30.1.211	1029	dp0bond0.500	-	7830	0	431	0
1311328.0	udp	OP	192.168.1.219	1031	172.30.1.132	1032	dp0bond0.516	-	7830	30	182	0
870042.0	udp	CL	192.168.1.219	1031	172.30.1.138	1031	dp0bond0.504	-	7830	0	431	0
1306594.0	udp	OP	172.30.1.142	1040	192.168.1.57	1040	dp0bond0.506	-	7830	30	182	0

Πίνακας 8: Αποθηκευμένες αντιστοιχίες διευθύνσεων και θηρών από το CGNAT στο DANOS

Κατά την επιστροφή κοιτά τις διευθύνσεις και θύρες πηγής κατά την είσοδο του πακέτου και κάνει αντίστροφη μετάφραση χρησιμοποιώντας τις αποθηκευμένες αντιστοιχίες.



Εικόνα 34: Αντίστροφη μετάφραση CGNAT κατά την επιστροφή των απαντήσεων

7.3.1 Παράμετροι CGNAT-NAT pool

- Διάθεση διευθύνσεων: round-robin

Είναι η μόνη διαθέσιμη επιλογή

- Ομαδοποίηση διευθύνσεων: ζεύγη

  Είναι η μόνη διαθέσιμη επιλογή, κάθε εσωτερική διεύθυνση αντιστοιχίζεται με μία μοναδική εξωτερική διεύθυνση.

- Εύρος διευθύνσεων:

  Για το NAT pool 1 ορίστηκε 192.168.1.2-192.168.1.254

  Για το NAT pool 2 ορίστηκε 172.30.1.2-172.30.1.254

- Διάθεση θηρών: διαδοχική, τυχαία

  Επιλέχθηκε διαδοχική για την συγκεκριμένη παραμετροποίηση, αν και η τυχαία επιλογή θα επέφερε ίδια αποτελέσματα

- Εύρος θηρών:

  Και για τα δύο NAT pool ορίστηκε 1024-65535. Το εύρος είναι υπεραρκετό για τον αριθμό συνδέσεων που γίνονται.

- Μέγεθος block:

  Επιλέχθηκε και για τα δύο pools 128

- Μέγιστα blocks για κάθε χρήστη

  Επιλέχθηκε και για τα δύο pools 8

Για το μέγεθος block και μέγιστα block ανά χρήστη μπορούν να δοθούν πολλές παράμετροι, αρκεί οι θύρες που χρειάζεται ο κάθε χρήστης (κάθε διαφορετική IP πηγής) να είναι λιγότερες από το γινόμενο μέγεθος block επί μέγιστα blocks ανά χρήστη.

Στο DANOS οι διευθύνσεις προς μετάφραση ορίζονται μέσω των address-group. Για τις διευθύνσεις και θύρες που αφορούν την μετάφραση στο DANOS ορίζονται στο CGNAT pool. Για την σύνδεση μεταξύ των δύο υπάρχει η λειτουργία των POLICY που αντιστοιχούν τις διευθύνσεις προς μετάφραση με το CGNAT. Έπειτα κάθε POLICY μπορεί να εφαρμοστεί μόνο σε μία διεπαφή για αυτό και όπως αναφέρθηκε παραπάνω, δημιουργήθηκαν 26 POLICIES.

7.3.2 Παράμετροι Ostinato

Για να δοκιμαστεί η αντοχή του DANOS ενώ υλοποιεί και CGNAT για το σύνολο της κίνησης που διέρχεται από αυτό, υπάρχουν δύο παράγοντες που πρέπει να οριστούν.

- Ταχύτητα ροής από το μηχάνημα πηγής

Δοκιμάζονται αντίστοιχες ταχύτητες με παραπάνω δοκιμές χωρίς CGNAT για να μπορεί να γίνει σύγκριση και να είναι ορατή η επιβάρυνση του συστήματος από το CGNAT

- Πλήθος διαφορετικών ροών (συνδυασμός διεύθυνσης IP, και θύρας πηγής) Ένας καλός αριθμός για διαφορετικές ροές είναι οι 1000 ροές, με 100 διαφορετικούς χρήστες και 10 θύρες για τον κάθε χρήστη. Ο δρομολογητής DANOS δηλαδή δοκιμάζεται στην εξυπηρέτηση 100 πελατών με 10 συνδέσεις ταυτόχρονα.

Με αυτό το σκεπτικό λοιπόν, στο Ostinato δημιουργήθηκε ροή με 10000 πακέτα udp, τα οποία αποτελούν συνδυασμό 100 διαφορετικών χρηστών (διεύθυνση πηγής) και 1000 διαφορετικών θηρών 10 για κάθε χρήστη. Η συνολική ροή αποτελούμενη από 1000 επιμέρους ροές επαναλαμβάνεται συνεχώς με συγκεκριμένη ταχύτητα προσομοιάζοντας μια διαρκή εξυπηρέτηση 100 χρηστών από το DANOS. Η ταχύτητα της ροής είναι ο παράγοντας που θα εξεταστεί.

7.3.3 Εμφάνιση παραμέτρων CGNAT στο DANOS

Για να γίνει καλύτερη η κατανόηση θα παρατεθούν εικόνες από το CGNAT. Η πρώτη αφορά την μετάφραση στην διεπαφή του DANOS στο VLAN500, την πρώτη διεπαφή δηλαδή που εξέρχονται τα πακέτα και γίνεται η πρώτη μετάφραση CGNAT. Η δεύτερη αφορά την διεπαφή του DANOS στο VLAN502, την δεύτερη διεπαφή που εξέρχονται τα πακέτα.

Subscriber		Public	
Address	Port	Address	Port
192.168.1.2	1024	172.30.1.69	1424
192.168.1.2	1124	172.30.1.69	1425
192.168.1.2	1224	172.30.1.69	1426
192.168.1.2	1324	172.30.1.69	1427
192.168.1.2	1424	172.30.1.69	1428
192.168.1.2	1524	172.30.1.69	1429
192.168.1.2	1624	172.30.1.69	1430
192.168.1.2	1724	172.30.1.69	1431
192.168.1.2	1824	172.30.1.69	1432
192.168.1.2	1924	172.30.1.69	1433
192.168.1.3	1025	172.30.1.179	1024
192.168.1.3	1125	172.30.1.179	1025
192.168.1.3	1225	172.30.1.179	1026
192.168.1.3	1325	172.30.1.179	1027
192.168.1.3	1425	172.30.1.179	1028
192.168.1.3	1525	172.30.1.179	1029
192.168.1.3	1625	172.30.1.179	1030
192.168.1.3	1725	172.30.1.179	1031
192.168.1.3	1825	172.30.1.179	1032
192.168.1.3	1925	172.30.1.179	1033
192.168.1.4	1026	172.30.1.180	1024
192.168.1.4	1126	172.30.1.180	1025
192.168.1.4	1226	172.30.1.180	1026
192.168.1.4	1326	172.30.1.180	1027
192.168.1.4	1426	172.30.1.180	1028
192.168.1.4	1526	172.30.1.180	1029
192.168.1.4	1626	172.30.1.180	1030
192.168.1.4	1726	172.30.1.180	1031
192.168.1.4	1826	172.30.1.180	1032
192.168.1.4	1926	172.30.1.180	1033
192.168.1.5	1027	172.30.1.181	1024
192.168.1.5	1127	172.30.1.181	1025
192.168.1.5	1227	172.30.1.181	1026
192.168.1.5	1327	172.30.1.181	1027
192.168.1.5	1427	172.30.1.181	1028
192.168.1.5	1527	172.30.1.181	1029
192.168.1.5	1627	172.30.1.181	1030
192.168.1.5	1727	172.30.1.181	1031
192.168.1.5	1827	172.30.1.181	1032
192.168.1.5	1927	172.30.1.181	1033

Subscriber		Public	
Address	Port	Address	Port
172.30.1.2	1024	192.168.1.118	1024
172.30.1.2	1025	192.168.1.118	1025
172.30.1.2	1026	192.168.1.118	1026
172.30.1.2	1027	192.168.1.118	1027
172.30.1.2	1028	192.168.1.118	1028
172.30.1.2	1029	192.168.1.118	1029
172.30.1.2	1030	192.168.1.118	1030
172.30.1.2	1031	192.168.1.118	1031
172.30.1.2	1032	192.168.1.118	1032
172.30.1.2	1033	192.168.1.118	1033
172.30.1.3	1024	192.168.1.65	1024
172.30.1.3	1025	192.168.1.65	1025
172.30.1.3	1026	192.168.1.65	1026
172.30.1.3	1027	192.168.1.65	1027
172.30.1.3	1028	192.168.1.65	1028
172.30.1.3	1029	192.168.1.65	1029
172.30.1.3	1030	192.168.1.65	1030
172.30.1.3	1031	192.168.1.65	1031
172.30.1.3	1032	192.168.1.65	1032
172.30.1.3	1033	192.168.1.65	1033
172.30.1.4	1024	192.168.1.112	1024
172.30.1.4	1025	192.168.1.112	1025
172.30.1.4	1026	192.168.1.112	1026
172.30.1.4	1027	192.168.1.112	1027
172.30.1.4	1028	192.168.1.112	1028
172.30.1.4	1029	192.168.1.112	1029
172.30.1.4	1030	192.168.1.112	1030
172.30.1.4	1031	192.168.1.112	1031
172.30.1.4	1032	192.168.1.112	1032
172.30.1.4	1033	192.168.1.112	1033
172.30.1.5	1024	192.168.1.75	1024
172.30.1.5	1025	192.168.1.75	1025
172.30.1.5	1026	192.168.1.75	1026
172.30.1.5	1027	192.168.1.75	1027
172.30.1.5	1028	192.168.1.75	1028
172.30.1.5	1029	192.168.1.75	1029
172.30.1.5	1030	192.168.1.75	1030
172.30.1.5	1031	192.168.1.75	1031
172.30.1.5	1032	192.168.1.75	1032
172.30.1.5	1033	192.168.1.75	1033

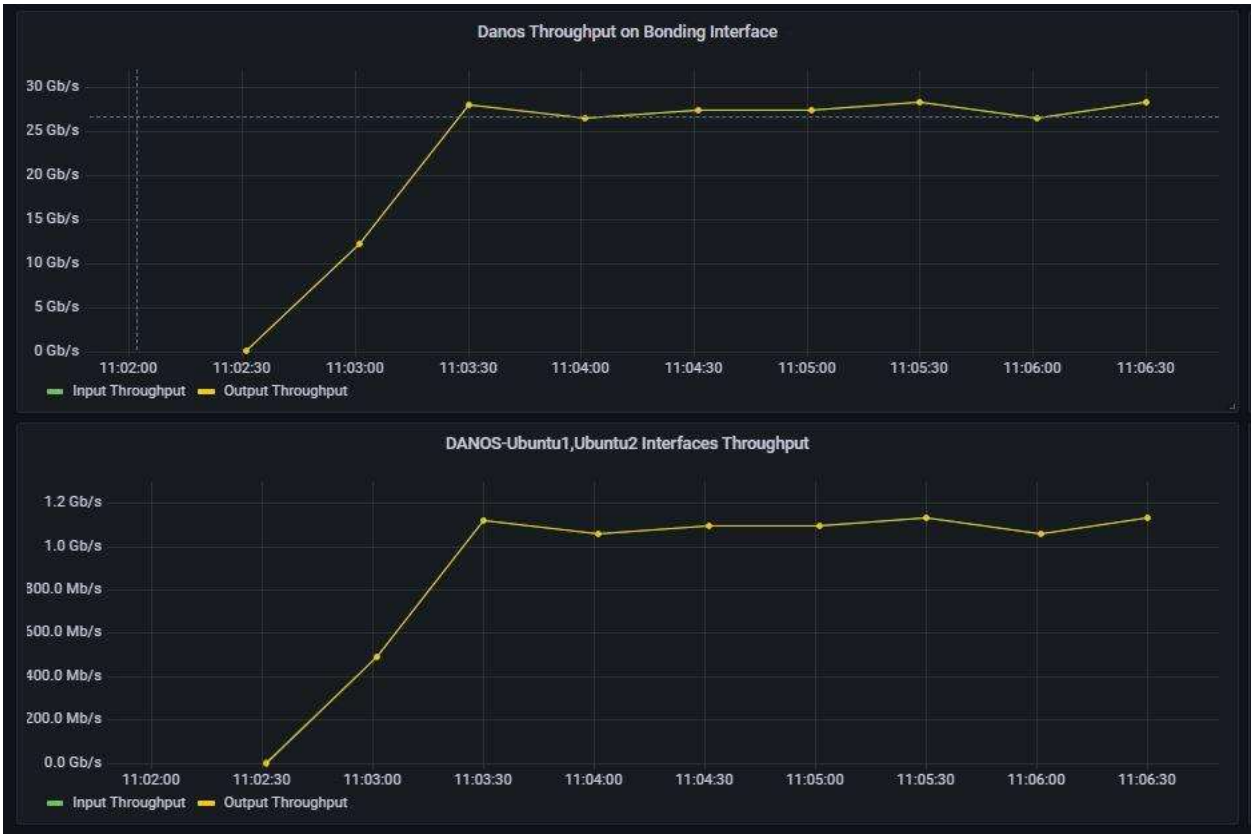
Εικόνα 35: Αποθηκευμένες αντιστοιχίες μετάφρασης CGNAT (α) VLAN500 (β) VLAN502

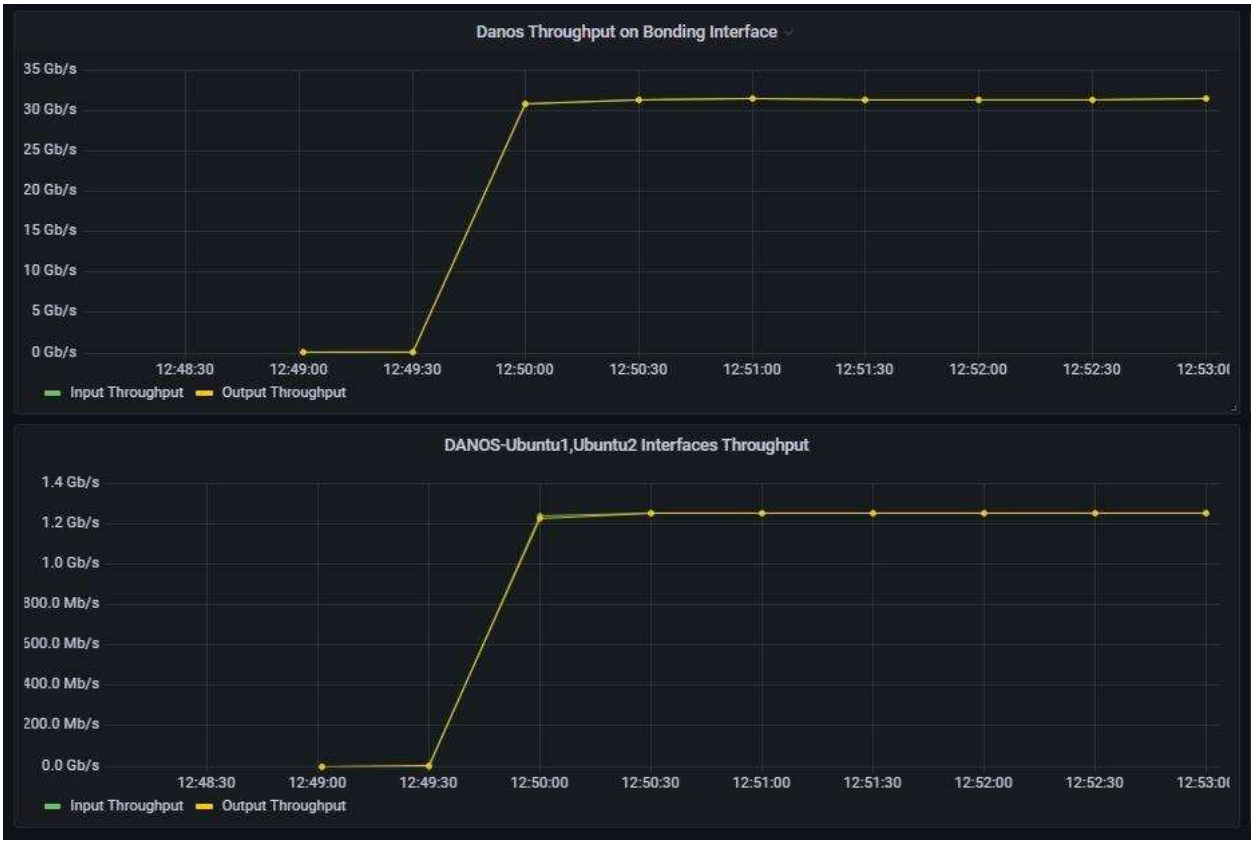
Όπως τονίστηκε παραπάνω, η πρώτη μετάφραση αφορά το δίκτυο 192.168.1.0/24 σε 172.30.1.0/24 ενώ η δεύτερη αντίστροφα από το 172.30.1.0/24 στο 192.168.1.0/24. Και στις δύο μεταφράσεις συντελείται και μετάφραση θήρας. Παρόλο που στην δεύτερη μετάφραση φαίνεται ότι έχει διατηρηθεί η πόρτα, έχει αλλάξει, απλώς λόγω της σειριακής ανάθεσης πορτών εν τέλει παίρνει την ίδια τιμή.

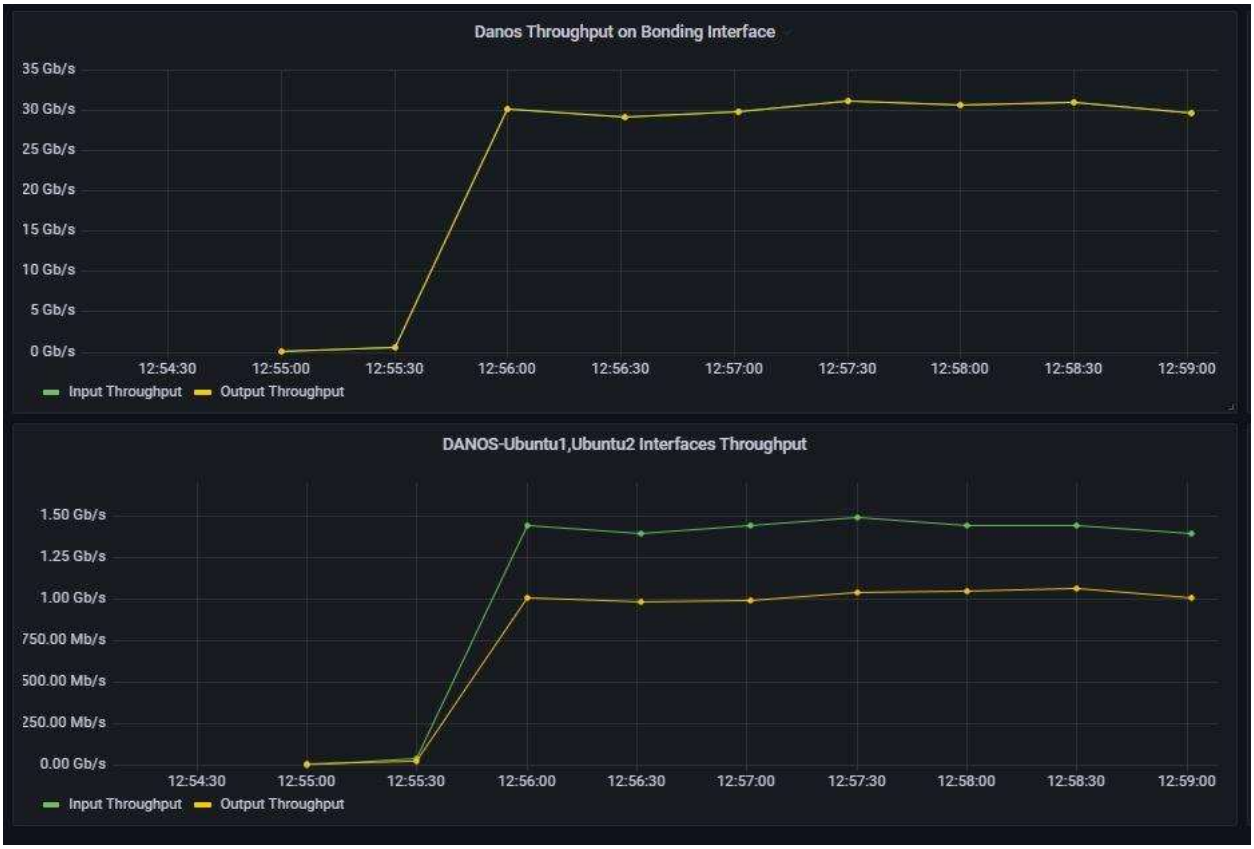
7.4 Αποτελέσματα Μετρήσεων

Ο παράγοντας των μετρήσεων, όπως αναφέρθηκε παραπάνω, είναι η ταχύτητα αποστολής και λήψης, καθώς μέσω των επαναλήψεων καθορίζουν την ποσότητα της κίνησης που κατάφερε το DANOS να επεξεργαστεί και να μεταφράσει μέσω CGNAT. Ακολουθούν τα αποτελέσματα για ταχύτητες αποστολής 1,1.2,1.4,1.6,1.8 και 2Gbps


Danos Throughput on Bonding Interface
25 Gb/s
20 Gb/s
15 Gb/s
10 Gb/s
5 Gb/s
0 Gb/s
11:09:00
11:09:30
11:10:00
11:10:30
11:11:00
11:11:30
11:12:00
11:12:30
11:13:00
11:13:30
Input Throughput
Output Throughput
DANOS-Ubuntu1,Ubuntu2 Interfaces Throughput
1.0 Gb/s
800.0 Mb/s
600.0 Mb/s
400.0 Mb/s
200.0 Mb/s
0.0 Gb/s
11:09:00
11:09:30
11:10:00
11:10:30
11:11:00
11:11:30
11:12:00
11:12:30
11:13:00
11:13:30
Input Throughput
Output Throughput


Danos Throughput on Bonding Interface
30 Gb/s
25 Gb/s
20 Gb/s
15 Gb/s
10 Gb/s
5 Gb/s
0 Gb/s
11:02:00
11:02:30
11:03:00
11:03:30
11:04:00
11:04:30
11:05:00
11:05:30
11:06:00
11:06:30
Input Throughput
Output Throughput
DANOS-Ubuntu1,Ubuntu2 Interfaces Throughput
1.2 Gb/s
1.0 Gb/s
800.0 Mb/s
600.0 Mb/s
400.0 Mb/s
200.0 Mb/s
0.0 Gb/s
11:02:00
11:02:30
11:03:00
11:03:30
11:04:00
11:04:30
11:05:00
11:05:30
11:06:00
11:06:30
Input Throughput
Output Throughput


Danos Throughput on Bonding Interface
35 Gb/s
30 Gb/s
25 Gb/s
20 Gb/s
15 Gb/s
10 Gb/s
5 Gb/s
0 Gb/s
12:48:30
12:49:00
12:49:30
12:50:00
12:50:30
12:51:00
12:51:30
12:52:00
12:52:30
Input Throughput
Output Throughput
DANOS-Ubuntu1,Ubuntu2 Interfaces Throughput
1.4 Gb/s
1.2 Gb/s
1.0 Gb/s
800.0 Mb/s
600.0 Mb/s
400.0 Mb/s
200.0 Mb/s
0.0 Gb/s
12:48:30
12:49:00
12:49:30
12:50:00
12:50:30
12:51:00
12:51:30
12:52:00
12:52:30
Input Throughput
Output Throughput

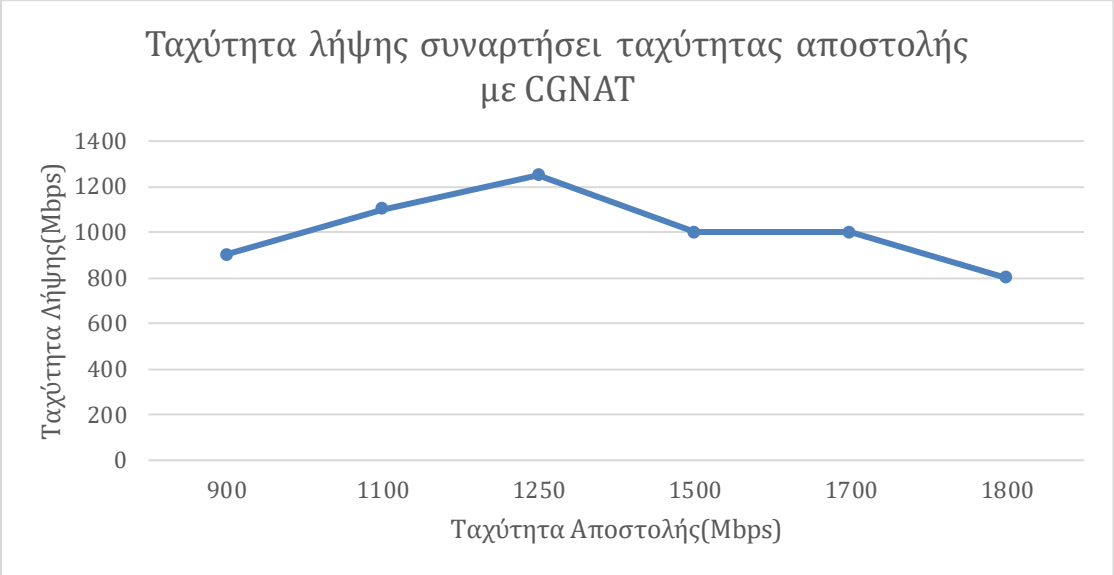

Danos Throughput on Bonding Interface
35 Gb/s
30 Gb/s
25 Gb/s
20 Gb/s
15 Gb/s
10 Gb/s
5 Gb/s
0 Gb/s
12:54:30
12:55:00
12:55:30
12:56:00
12:56:30
12:57:00
12:57:30
12:58:00
12:58:30
12:59:00
Input Throughput
Output Throughput
DANOS-Ubuntu1,Ubuntu2 Interfaces Throughput
1.50 Gb/s
1.25 Gb/s
1.00 Gb/s
750.00 Mb/s
500.00 Mb/s
250.00 Mb/s
0.00 Gb/s
12:54:30
12:55:00
12:55:30
12:56:00
12:56:30
12:57:00
12:57:30
12:58:00
12:58:30
12:59:00
Input Throughput
Output Throughput


Danos Throughput on Bonding Interface
35 Gb/s
30 Gb/s
25 Gb/s
20 Gb/s
15 Gb/s
10 Gb/s
5 Gb/s
0 Gb/s
13:00:00
13:00:30
13:01:00
13:01:30
13:02:00
13:02:30
13:03:00
13:03:30
13:04:00
13:04:30
Input Throughput
Output Throughput
DANOS-Ubuntu1,Ubuntu2 Interfaces Throughput
1.75 Gb/s
1.50 Gb/s
1.25 Gb/s
1.00 Gb/s
750.00 Mb/s
500.00 Mb/s
250.00 Mb/s
0.00 Gb/s
13:00:00
13:00:30
13:01:00
13:01:30
13:02:00
13:02:30
13:03:00
13:03:30
13:04:00
13:04:30
Input Throughput
Output Throughput


Danos Throughput on Bonding Interface
35 Gb/s
30 Gb/s
25 Gb/s
20 Gb/s
15 Gb/s
10 Gb/s
5 Gb/s
0 Gb/s
13:11:00
13:11:30
13:12:00
13:12:30
13:13:00
13:13:30
13:14:00
13:14:30
13:15:00
13:15:30
Input Throughput
Output Throughput
DANOS-Ubuntu1,Ubuntu2 Interfaces Throughput
2.0 Gb/s
1.5 Gb/s
1.0 Gb/s
500.0 Mb/s
0.0 Gb/s
13:11:00
13:11:30
13:12:00
13:12:30
13:13:00
13:13:30
13:14:00
13:14:30
13:15:00
13:15:30
Input Throughput
Output Throughput

Διάγραμμα 9: Ταχύτητες λήψης, αποστολής και επεξεργασίας από το DANOS

7.5 Συμπεράσματα Μετρήσεων



Διάγραμμα 10: Ταχύτητα λήψης συναρτήσει ταχύτητας αποστολής με CGNAT

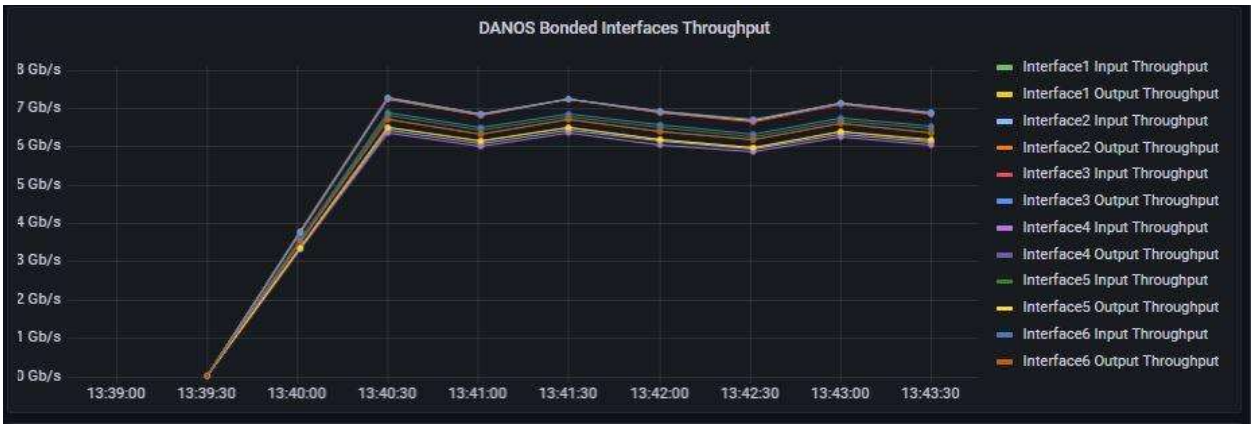
Για ταχύτητες 1, 1.2 και 1.4 Gbps αποστολής που αντιστοιχούν μέσω των ανακυκλώσεων σε 25,28,31 Gbps, το DANOS ανταποκρίνεται εξαιρετικά, καταφέρνοντας να μεταφράσει όλη την κίνηση. Σε μεγαλύτερες ταχύτητες, παρατηρείται ότι προσεγγίζεται το όριο επεξεργασίας του DANOS, το οποίο φαίνεται από το χάσμα μεταξύ της ταχύτητας λήψης από το Ubuntu1 και αποστολής προς το Ubuntu2.

Όσο μεγαλώνει η ταχύτητα αποστολής, παρατηρείται ότι μεγαλώνει το χάσμα. Το όριο του DANOS με CGNAT φαίνεται ότι περιορίζεται κοντά στα 30-33 Gbps ανά κατεύθυνση. Όπως και στο μέρος δοκιμών με τείχος προστασίας, παρατηρείται ότι το DANOS επηρεάζεται από την μετάφραση πακέτων και τις αναζητήσεις που αυτή επιβάλλει στο σύστημα. Για 1000 διαφορετικούς συνδυασμούς διευθύνσεων IP και θηρών το DANOS μπορεί να επεξεργαστεί έως 30-33 Gbps.

Η ταχύτητα αυτή είναι αρκετά μικρότερη από τις ταχύτητες που παρατηρήθηκαν στις δοκιμές του DANOS χωρίς CGNAT και χωρίς firewall στο πρώτο μέρος των δοκιμών, όπου είχε επιτευχθεί ταχύτητα κοντά στα 50Gbps.

7.6 CGNAT και LACP στο DANOS

Κατά την διάρκεια των δοκιμών με το CGNAT, παρατηρήθηκαν στο διάγραμμα των επιμέρους έξι διεπαφών, ομοιομορφία και ισομοιρασμός της κίνησης μεταξύ τους. Το παρακάτω διάγραμμα αφορά την δοκιμή για αποστολή 2Gbps αλλά αντιπροσωπεύει όλες τις δοκιμές, καθώς σε όλες τις ταχύτητες παρατηρήθηκε ομοιομορφία στην κίνηση στις διεπαφές.



Διάγραμμα 11: Κατανομή ροών πακέτων από το LACP στις διεπαφές όταν χρησιμοποιείται CGNAT

Λόγω των 1000 διαφορετικών συνδέσεων, το LACP μοιράζει τις συνδέσεις σχεδόν ομοιόμορφα μεταξύ των 6 διεπαφών. Σε συνδυασμό με το γεγονός ότι δεν παρατηρείται κάποια διεπαφή να φτάνει τα 10Gbps, οποιοσδήποτε περιορισμός στην ταχύτητα οφείλεται στην αδυναμία του DANOS να επεξεργαστεί την κίνηση. Η παρατήρηση αυτή είναι πολύ σημαντική, καθώς αφαιρεί μία παράμετρο λάθους από τις μετρήσεις και καθιστά μοναδικό παράγοντα την επεξεργαστική ισχύ του DANOS.

Κεφάλαιο 8 : Σύνοψη

8.1 Συμπεράσματα

Με τις δοκιμές που έγιναν, το DANOS ανταποκρίθηκε ως επί το πλείστον ικανοποιητικά, επιτυγχάνοντας υψηλές επιδόσεις. Πιο συγκεκριμένα, επιτεύχθηκαν έως 54 Gbps ταχύτητες επεξεργασίας ανά κατεύθυνση στο όριο των 60 Gbps που αφορά το υλικό. Το DANOS ανά στιγμή κατάφερε να επεξεργαστεί ταχύτητες τάξεως των 110 Gbps, αν συμψηφιστούν και οι δύο κατευθύνσεις και οι ροές πηγής και προορισμού. Σε δοκιμές με επιπλέον παραμετροποίηση, δηλαδή τείχη προστασίας και CGNAT ,παρατηρήθηκε μείωση των επιδόσεων κατά περίπου 50%. Πιο συγκεκριμένα, για το μέγιστο δυνατό τείχος προστασίας (10000 κανόνες) και για την παραμετροποίηση CGNAT, οι ταχύτητες επεξεργασίας περιορίστηκαν στα 25 Gbps ανά κατεύθυνση. Για μικρότερα τείχη προστασίας, οι ταχύτητες ήταν ικανοποιητικές, αντίστοιχες με τις βέλτιστες.

8.2 Μελλοντικές Προεκτάσεις

8.2.1 Πειράματα

Στα πλαίσια της διπλωματικής, εξετάστηκαν χαρακτηριστικά του DANOS, όπως η χρήση τειχών προστασίας και CGNAT μετάφραση. Ως επέκταση των παραπάνω, αρχικά προτείνεται η ένταξη στο σύστημα μεγάλου πίνακα δρομολόγησης. Οι πραγματικοί δρομολογητές που διαχειρίζονται πολλούς χρήστες διαθέτουν πολύ μεγάλους πίνακες δρομολόγησης. Για κάθε πακέτο που καταφθάνει, εξετάζεται η διεύθυνση προορισμού. Προτείνεται να εξεταστεί η επίπτωση της αναζήτησης καταγραφής στον πίνακα δρομολόγησης, κατά την άφιξη των πακέτων στο σύστημα. Για περαιτέρω δοκιμές, μπορούν να παραμετροποιηθούν L2,L3 tunnel πρωτόκολλα διαθέσιμα στο DANOS καθώς και PPPoe.

8.2.2 Υλικό

Από την πλευρά του υλικού, η επεξεργαστική δύναμη του server ήταν υπέρ αρκετή, υποστηρίζοντας εύκολα τις εικονικές μηχανές που φιλοξένησε. Μια προτεινόμενη βελτίωση, όσον αφορά το υλικό, είναι η χρήση όχι 6 διεπαφών, αλλά μίας διεπαφής 40 ή 100 Gbps. Η χρήση πολλαπλών διεπαφών σε ζεύξη, χρησιμοποιώντας το πρωτόκολλο LACP, δημιούργησε προβλήματα τυχαιότητας κατά την διάρκεια των μετρήσεων, καθώς οι πόρτες διανέμονταν

διαφορετικά με κάθε επανάληψη των μετρήσεων. Επομένως, για την απαλοιφή ενός παράγοντα που μπορεί να επηρεάσει τις μετρήσεις, προτείνεται η επιλογή μιας ζεύξης. Προφανώς, τέτοιου είδους διεπαφή πρέπει να υποστηρίζεται και από τα δύο άκρα της φυσικής σύνδεσης server και switch.

8.2.3 Λογισμικό

Όσον αφορά το λογισμικό, προτείνεται προσθήκη χαρακτηριστικών στο λειτουργικό DANOS, τόσο για την επίτευξη καλύτερων επιδόσεων όσο και για τη διευκόλυνση του χρήστη στην παραμετροποίηση και χαλιναγώγηση των πόρων. Αρχικά, η δημιουργία εργαλείου ethtool που υποστηρίζει τους drivers που χρησιμοποιεί το DANOS για τις διεπαφές του στον χώρο χρήστη, θα βοηθούσε στην κατανόηση της λειτουργίας της NIC. Στις διαθέσιμες έκδοσεις δεν υπάρχει δυνατότητα να φανούν οι χρησιμοποιούμενες Rx,Tx ουρές και οι αντιστοιχίες τους με τους πυρήνες του επεξεργαστή. Εκτός από την εμφάνιση των ουρών, που αποτελεί πολύ χρήσιμο εργαλείο κατανόησης του συστήματος από τον προγραμματιστή, προτείνεται η επέκταση αυτού, σε δυνατότητα ορισμού από τον χρήστη του DANOS συγκεκριμένου αριθμού Rx,Tx ουρών και των αντίστοιχων πυρήνων που αναλαμβάνουν την επεξεργασία αυτών. Κατά τις δοκιμές παρατηρήθηκε χρησιμοποίηση μόνο 2-4 πυρήνων του επεξεργαστή, αυτόματα από το σύστημα. Η εύκολη πρόσβαση στις ουρές και τους πυρήνες από τον χρήστη, θα αποτελούσε εξαιρετικά χρήσιμο εργαλείο, καθώς θα υπήρχε η δυνατότητα δοκιμών με παραμέτρους τον αριθμό ουρών και πυρήνων. Το DPDK παρέχει δυνατότητα ορισμού ουρών και αντιστοιχία με πυρήνες από τον χρήστη, χαρακτηριστικό που υπάρχει σε πολλά από τα παραδείγματα που παρέχει. Προτείνεται λοιπόν, η ενσωμάτωση του παραπάνω χαρακτηριστικού στο DANOS, που άλλωστε χρησιμοποιεί το DPDK για την επεξεργασία των πακέτων.

Παράρτημα Α: Κώδικας

Αρχική παραμετροποίηση DANOS

- Διευθυνσιοδότηση διεπαφών

set interfaces dataplane dp0p8s18 address 10.10.40.8/24

set interfaces dataplane dp0p8s19 address 192.168.1.1/24

set interfaces dataplane dp0p8s20 address 192.168.2.1/24

- Υπηρεσία ssh για την σύνδεση εξ αποστάσεως στο μηχάνημα

set service ssh

- Υπηρεσία snmp για να μπορεί να λαμβάνει πληροφορίες ο Ubuntu server από το DANOS

set service snmp community public

- Υπηρεσία dns για επικοινωνία με το διαδίκτυο

set service dns forwarding listen-on dp0p8s18

set service dns forwarding name-server 8.8.8.8

- Διευθυνσιοδότηση διεπαφών στα VLANs

set protocols static route 0.0.0.0/0 next-hop 10.10.40.1 interface dp0p8s18

set interfaces bonding dp0bond0

set interfaces bonding dp0bond0 lacp-options activity active

set interfaces bonding dp0bond0 vif 500 address 10.10.50.1/29

set interfaces bonding dp0bond0 vif 501 address 10.10.51.1/29

set interfaces bonding dp0bond0 vif 502 address 10.10.52.1/29

set interfaces bonding dp0bond0 vif 503 address 10.10.53.1/29

set interfaces bonding dp0bond0 vif 504 address 10.10.54.1/29

set interfaces bonding dp0bond0 vif 505 address 10.10.55.1/29

set interfaces bonding dp0bond0 vif 506 address 10.10.56.1/29

set interfaces bonding dp0bond0 vif 507 address 10.10.57.1/29

set interfaces bonding dp0bond0 vif 508 address 10.10.58.1/29

set interfaces bonding dp0bond0 vif 509 address 10.10.59.1/29

set interfaces bonding dp0bond0 vif 510 address 10.10.60.1/29

```
set interfaces bonding dp0bond0 vif 511 address 10.10.61.1/29
set interfaces bonding dp0bond0 vif 512 address 10.10.62.1/29
set interfaces bonding dp0bond0 vif 513 address 10.10.63.1/29
set interfaces bonding dp0bond0 vif 514 address 10.10.64.1/29
set interfaces bonding dp0bond0 vif 515 address 10.10.65.1/29
set interfaces bonding dp0bond0 vif 516 address 10.10.66.1/29
set interfaces bonding dp0bond0 vif 517 address 10.10.67.1/29
set interfaces bonding dp0bond0 vif 518 address 10.10.68.1/29
set interfaces bonding dp0bond0 vif 519 address 10.10.69.1/29
set interfaces bonding dp0bond0 vif 520 address 10.10.70.1/29
set interfaces bonding dp0bond0 vif 521 address 10.10.71.1/29
set interfaces bonding dp0bond0 vif 522 address 10.10.72.1/29
set interfaces bonding dp0bond0 vif 523 address 10.10.73.1/29
set interfaces bonding dp0bond0 vif 524 address 10.10.74.1/29
set interfaces bonding dp0bond0 vif 525 address 10.10.75.1/29
set interfaces bonding dp0bond0 vif 526 address 10.10.76.1/29
set interfaces bonding dp0bond0 vif 527 address 10.10.77.1/29
set interfaces bonding dp0bond0 vif 528 address 10.10.78.1/29
set interfaces bonding dp0bond0 vif 529 address 10.10.79.1/29
set interfaces bonding dp0bond0 vif 530 address 10.10.80.1/29
set interfaces bonding dp0bond0 vif 531 address 10.10.81.1/29
set interfaces bonding dp0bond0 vif 532 address 10.10.82.1/29
set interfaces bonding dp0bond0 vif 533 address 10.10.83.1/29
set interfaces bonding dp0bond0 vif 534 address 10.10.84.1/29
set interfaces bonding dp0bond0 vif 535 address 10.10.85.1/29
set interfaces bonding dp0bond0 vif 536 address 10.10.86.1/29
set interfaces bonding dp0bond0 vif 537 address 10.10.87.1/29
set interfaces bonding dp0bond0 vif 538 address 10.10.88.1/29
set interfaces bonding dp0bond0 vif 539 address 10.10.89.1/29
set interfaces bonding dp0bond0 vif 540 address 10.10.90.1/29
```

```
set interfaces bonding dp0bond0 vif 541 address 10.10.91.1/29
set interfaces bonding dp0bond0 vif 542 address 10.10.92.1/29
set interfaces bonding dp0bond0 vif 543 address 10.10.93.1/29
set interfaces bonding dp0bond0 vif 544 address 10.10.94.1/29
set interfaces bonding dp0bond0 vif 545 address 10.10.95.1/29
set interfaces bonding dp0bond0 vif 546 address 10.10.96.1/29
set interfaces bonding dp0bond0 vif 547 address 10.10.97.1/29
set interfaces bonding dp0bond0 vif 548 address 10.10.98.1/29
set interfaces bonding dp0bond0 vif 549 address 10.10.99.1/29
```

- Δημιουργία της ζεύξης των διεπαφών (link-aggregation)

```
set interfaces dataplane dp0p1s0 bond-group dp0bond0
set interfaces dataplane dp0p1s0 description Twe1/0/15
set interfaces dataplane dp0p2s0 bond-group dp0bond0
set interfaces dataplane dp0p2s0 description Twe1/0/13
set interfaces dataplane dp0p3s0 bond-group dp0bond0
set interfaces dataplane dp0p3s0 description Twe1/0/14
set interfaces dataplane dp0p4s0 bond-group dp0bond0
set interfaces dataplane dp0p4s0 description Twe1/0/12
set interfaces dataplane dp0p5s0 bond-group dp0bond0
set interfaces dataplane dp0p5s0 description Twe1/0/11
set interfaces dataplane dp0p6s0 bond-group dp0bond0
set interfaces dataplane dp0p6s0 description Twe1/0/16
```

- Δημιουργία VRFs, ανάθεση διεπαφών στα VRF και ορισμός στατικών διαδρομών για τα πακέτα

```
set routing routing-instance vrf0 instance-type vrf
set routing routing-instance vrf0 interface dp0bond0.500
set routing routing-instance vrf0 interface dp0p8s19
set routing routing-instance vrf0 protocols static route 192.168.2.0/24 next-hop 10.10.50.2 interface dp0bond0.500
set routing routing-instance vrf1 instance-type vrf
```

```
set routing routing-instance vrf1 interface dp0bond0.501
set routing routing-instance vrf1 interface dp0bond0.502
set routing routing-instance vrf1 protocols static route 192.168.1.0/24 next-hop 10.10.51.2 interface dp0bond0.501
set routing routing-instance vrf1 protocols static route 192.168.2.0/24 next-hop 10.10.52.2 interface dp0bond0.502
set routing routing-instance vrf2 instance-type vrf
set routing routing-instance vrf2 interface dp0bond0.503
set routing routing-instance vrf2 interface dp0bond0.504
set routing routing-instance vrf2 protocols static route 192.168.1.0/24 next-hop 10.10.53.2 interface dp0bond0.503
set routing routing-instance vrf2 protocols static route 192.168.2.0/24 next-hop 10.10.54.2 interface dp0bond0.504
set routing routing-instance vrf3 instance-type vrf
set routing routing-instance vrf3 interface dp0bond0.505
set routing routing-instance vrf3 interface dp0bond0.506
set routing routing-instance vrf3 protocols static route 192.168.1.0/24 next-hop 10.10.55.2 interface dp0bond0.505
set routing routing-instance vrf3 protocols static route 192.168.2.0/24 next-hop 10.10.56.2 interface dp0bond0.506
set routing routing-instance vrf4 instance-type vrf
set routing routing-instance vrf4 interface dp0bond0.507
set routing routing-instance vrf4 interface dp0bond0.508
set routing routing-instance vrf4 protocols static route 192.168.1.0/24 next-hop 10.10.57.2 interface dp0bond0.507
set routing routing-instance vrf4 protocols static route 192.168.2.0/24 next-hop 10.10.58.2 interface dp0bond0.508
set routing routing-instance vrf5 instance-type vrf
set routing routing-instance vrf5 interface dp0bond0.509
set routing routing-instance vrf5 interface dp0bond0.510
set routing routing-instance vrf5 protocols static route 192.168.1.0/24 next-hop 10.10.59.2 interface dp0bond0.509
```

set routing routing-instance vrf5 protocols static route 192.168.2.0/24 next-hop 10.10.60.2 interface dp0bond0.510

set routing routing-instance vrf6 instance-type vrf

set routing routing-instance vrf6 interface dp0bond0.511

set routing routing-instance vrf6 interface dp0bond0.512

set routing routing-instance vrf6 protocols static route 192.168.1.0/24 next-hop 10.10.61.2 interface dp0bond0.511

set routing routing-instance vrf6 protocols static route 192.168.2.0/24 next-hop 10.10.62.2 interface dp0bond0.512

set routing routing-instance vrf7 instance-type vrf

set routing routing-instance vrf7 interface dp0bond0.513

set routing routing-instance vrf7 interface dp0bond0.514

set routing routing-instance vrf7 protocols static route 192.168.1.0/24 next-hop 10.10.63.2 interface dp0bond0.513

set routing routing-instance vrf7 protocols static route 192.168.2.0/24 next-hop 10.10.64.2 interface dp0bond0.514

set routing routing-instance vrf8 instance-type vrf

set routing routing-instance vrf8 interface dp0bond0.515

set routing routing-instance vrf8 interface dp0bond0.516

set routing routing-instance vrf8 protocols static route 192.168.1.0/24 next-hop 10.10.65.2 interface dp0bond0.515

set routing routing-instance vrf8 protocols static route 192.168.2.0/24 next-hop 10.10.66.2 interface dp0bond0.516

set routing routing-instance vrf9 instance-type vrf

set routing routing-instance vrf9 interface dp0bond0.517

set routing routing-instance vrf9 interface dp0bond0.518

set routing routing-instance vrf9 protocols static route 192.168.1.0/24 next-hop 10.10.67.2 interface dp0bond0.517

set routing routing-instance vrf9 protocols static route 192.168.2.0/24 next-hop 10.10.68.2 interface dp0bond0.518

set routing routing-instance vrf10 instance-type vrf

set routing routing-instance vrf10 interface dp0bond0.519

set routing routing-instance vrf10 interface dp0bond0.520

```
set routing routing-instance vrf10 protocols static route 192.168.1.0/24 next-hop 10.10.69.2 interface dp0bond0.519
set routing routing-instance vrf10 protocols static route 192.168.2.0/24 next-hop 10.10.70.2 interface dp0bond0.520
set routing routing-instance vrf11 instance-type vrf
set routing routing-instance vrf11 interface dp0bond0.521
set routing routing-instance vrf11 interface dp0bond0.522
set routing routing-instance vrf11 protocols static route 192.168.1.0/24 next-hop 10.10.71.2 interface dp0bond0.521
set routing routing-instance vrf11 protocols static route 192.168.2.0/24 next-hop 10.10.72.2 interface dp0bond0.522
set routing routing-instance vrf12 instance-type vrf
set routing routing-instance vrf12 interface dp0bond0.523
set routing routing-instance vrf12 interface dp0bond0.524
set routing routing-instance vrf12 protocols static route 192.168.1.0/24 next-hop 10.10.73.2 interface dp0bond0.523
set routing routing-instance vrf12 protocols static route 192.168.2.0/24 next-hop 10.10.74.2 interface dp0bond0.524
set routing routing-instance vrf13 instance-type vrf
set routing routing-instance vrf13 interface dp0bond0.525
set routing routing-instance vrf13 interface dp0bond0.526
set routing routing-instance vrf13 protocols static route 192.168.1.0/24 next-hop 10.10.75.2 interface dp0bond0.525
set routing routing-instance vrf13 protocols static route 192.168.2.0/24 next-hop 10.10.76.2 interface dp0bond0.526
set routing routing-instance vrf14 instance-type vrf
set routing routing-instance vrf14 interface dp0bond0.527
set routing routing-instance vrf14 interface dp0bond0.528
set routing routing-instance vrf14 protocols static route 192.168.1.0/24 next-hop 10.10.77.2 interface dp0bond0.527
set routing routing-instance vrf14 protocols static route 192.168.2.0/24 next-hop 10.10.78.2 interface dp0bond0.528
set routing routing-instance vrf15 instance-type vrf
```

```
set routing routing-instance vrf15 interface dp0bond0.529
set routing routing-instance vrf15 interface dp0bond0.530
set routing routing-instance vrf15 protocols static route 192.168.1.0/24 next-hop 10.10.79.2 interface dp0bond0.529
set routing routing-instance vrf15 protocols static route 192.168.2.0/24 next-hop 10.10.80.2 interface dp0bond0.530
set routing routing-instance vrf16 instance-type vrf
set routing routing-instance vrf16 interface dp0bond0.531
set routing routing-instance vrf16 interface dp0bond0.532
set routing routing-instance vrf16 protocols static route 192.168.1.0/24 next-hop 10.10.81.2 interface dp0bond0.531
set routing routing-instance vrf16 protocols static route 192.168.2.0/24 next-hop 10.10.82.2 interface dp0bond0.532
set routing routing-instance vrf17 instance-type vrf
set routing routing-instance vrf17 interface dp0bond0.533
set routing routing-instance vrf17 interface dp0bond0.534
set routing routing-instance vrf17 protocols static route 192.168.1.0/24 next-hop 10.10.83.2 interface dp0bond0.533
set routing routing-instance vrf17 protocols static route 192.168.2.0/24 next-hop 10.10.84.2 interface dp0bond0.534
set routing routing-instance vrf18 instance-type vrf
set routing routing-instance vrf18 interface dp0bond0.535
set routing routing-instance vrf18 interface dp0bond0.536
set routing routing-instance vrf18 protocols static route 192.168.1.0/24 next-hop 10.10.85.2 interface dp0bond0.535
set routing routing-instance vrf18 protocols static route 192.168.2.0/24 next-hop 10.10.86.2 interface dp0bond0.536
set routing routing-instance vrf19 instance-type vrf
set routing routing-instance vrf19 interface dp0bond0.537
set routing routing-instance vrf19 interface dp0bond0.538
set routing routing-instance vrf19 protocols static route 192.168.1.0/24 next-hop 10.10.87.2 interface dp0bond0.537
```

```
set routing routing-instance vrf19 protocols static route 192.168.2.0/24 next-hop 10.10.88.2 interface dp0bond0.538
set routing routing-instance vrf20 instance-type vrf
set routing routing-instance vrf20 interface dp0bond0.539
set routing routing-instance vrf20 interface dp0bond0.540
set routing routing-instance vrf20 protocols static route 192.168.1.0/24 next-hop 10.10.89.2 interface dp0bond0.539
set routing routing-instance vrf20 protocols static route 192.168.2.0/24 next-hop 10.10.90.2 interface dp0bond0.540
set routing routing-instance vrf21 instance-type vrf
set routing routing-instance vrf21 interface dp0bond0.541
set routing routing-instance vrf21 interface dp0bond0.542
set routing routing-instance vrf21 protocols static route 192.168.1.0/24 next-hop 10.10.91.2 interface dp0bond0.541
set routing routing-instance vrf21 protocols static route 192.168.2.0/24 next-hop 10.10.92.2 interface dp0bond0.542
set routing routing-instance vrf22 instance-type vrf
set routing routing-instance vrf22 interface dp0bond0.543
set routing routing-instance vrf22 interface dp0bond0.544
set routing routing-instance vrf22 protocols static route 192.168.1.0/24 next-hop 10.10.93.2 interface dp0bond0.543
set routing routing-instance vrf22 protocols static route 192.168.2.0/24 next-hop 10.10.94.2 interface dp0bond0.544
set routing routing-instance vrf23 instance-type vrf
set routing routing-instance vrf23 interface dp0bond0.545
set routing routing-instance vrf23 interface dp0bond0.546
set routing routing-instance vrf23 protocols static route 192.168.1.0/24 next-hop 10.10.95.2 interface dp0bond0.545
set routing routing-instance vrf23 protocols static route 192.168.2.0/24 next-hop 10.10.96.2 interface dp0bond0.546
set routing routing-instance vrf24 instance-type vrf
set routing routing-instance vrf24 interface dp0bond0.547
set routing routing-instance vrf24 interface dp0bond0.548
```

set routing routing-instance vrf24 protocols static route 192.168.1.0/24 next-hop 10.10.97.2 interface dp0bond0.547

set routing routing-instance vrf24 protocols static route 192.168.2.0/24 next-hop 10.10.98.2 interface dp0bond0.548

set routing routing-instance vrf25 instance-type vrf

set routing routing-instance vrf25 interface dp0bond0.549

set routing routing-instance vrf25 interface dp0p8s20

set routing routing-instance vrf25 protocols static route 192.168.1.0/24 next-hop 10.10.99.2 interface dp0bond0.549

Αρχική παραμετροποίηση switch

- Ορισμός VRFs
- Ορισμός VLANs και διευθυνσιοδότηση
- Δημιουργία στατικών διαδρομών στα VRFs
- Δημιουργία port-range
- Παραμετροποίηση υπηρεσίας snmp για λήψη δεδομένων από το Ubuntu server
- Ορισμός channel-group (link aggregation), ένωση των έξι διεπαφών και ανάθεση τους στο channel-group και επίτρεψη των VLANs

vrf definition dn0

address-family ipv4

exit

vrf definition dn1

address-family ipv4

exit

vrf definition dn2

address-family ipv4

exit

vrf definition dn3

address-family ipv4

exit

vrf definition dn4

address-family ipv4

exit

```
vrf definition dn5
address-family ipv4
exit
vrf definition dn6
address-family ipv4
exit
vrf definition dn7
address-family ipv4
exit
vrf definition dn8
address-family ipv4
exit
vrf definition dn9
address-family ipv4
exit
vrf definition dn10
address-family ipv4
exit
vrf definition dn11
address-family ipv4
exit
vrf definition dn12
address-family ipv4
exit
vrf definition dn13
address-family ipv4
exit
vrf definition dn14
address-family ipv4
exit
```

```
vrf definition dn15
address-family ipv4
exit
vrf definition dn16
address-family ipv4
exit
vrf definition dn17
address-family ipv4
exit
vrf definition dn18
address-family ipv4
exit
vrf definition dn19
address-family ipv4
exit
vrf definition dn20
address-family ipv4
exit
vrf definition dn21
address-family ipv4
exit
vrf definition dn22
address-family ipv4
exit
vrf definition dn23
address-family ipv4
exit
vrf definition dn24
address-family ipv4
exit
```

```
interface Vlan 500
vrf forwarding dn0
ip address 10.10.50.2 255.255.255.252
interface Vlan 501
vrf forwarding dn0
ip address 10.10.50.2 255.255.255.252
interface Vlan 502
vrf forwarding dn0
ip address 10.10.50.2 255.255.255.252
interface Vlan 503
vrf forwarding dn0
ip address 10.10.50.2 255.255.255.252
interface Vlan 504
vrf forwarding dn0
ip address 10.10.50.2 255.255.255.252
interface Vlan 505
vrf forwarding dn0
ip address 10.10.50.2 255.255.255.252
interface Vlan 506
vrf forwarding dn0
ip address 10.10.50.2 255.255.255.252
interface Vlan 507
vrf forwarding dn0
ip address 10.10.50.2 255.255.255.252
interface Vlan 508
vrf forwarding dn0
ip address 10.10.50.2 255.255.255.252
interface Vlan 509
vrf forwarding dn0
```

```
ip address 10.10.50.2 255.255.255.252
interface Vlan 510
vrf forwarding dn0
ip address 10.10.50.2 255.255.255.252
interface Vlan 511
vrf forwarding dn0
ip address 10.10.50.2 255.255.255.252
interface Vlan 512
vrf forwarding dn0
ip address 10.10.50.2 255.255.255.252
interface Vlan 513
vrf forwarding dn0
ip address 10.10.50.2 255.255.255.252
interface Vlan 514
vrf forwarding dn0
ip address 10.10.50.2 255.255.255.252
interface Vlan 515
vrf forwarding dn0
ip address 10.10.50.2 255.255.255.252
interface Vlan 516
vrf forwarding dn0
ip address 10.10.50.2 255.255.255.252
interface Vlan 517
vrf forwarding dn0
ip address 10.10.50.2 255.255.255.252
interface Vlan 518
vrf forwarding dn0
ip address 10.10.50.2 255.255.255.252
interface Vlan 519
vrf forwarding dn0
```

```
ip address 10.10.50.2 255.255.255.252
interface Vlan 520
vrf forwarding dn0
ip address 10.10.50.2 255.255.255.252
interface Vlan 521
vrf forwarding dn0
ip address 10.10.50.2 255.255.255.252
interface Vlan 522
vrf forwarding dn0
ip address 10.10.50.2 255.255.255.252
interface Vlan 523
vrf forwarding dn0
ip address 10.10.50.2 255.255.255.252
interface Vlan 524
vrf forwarding dn0
ip address 10.10.50.2 255.255.255.252
interface Vlan 525
vrf forwarding dn0
ip address 10.10.50.2 255.255.255.252
interface Vlan 526
vrf forwarding dn0
ip address 10.10.50.2 255.255.255.252
interface Vlan 527
vrf forwarding dn0
ip address 10.10.50.2 255.255.255.252
interface Vlan 528
vrf forwarding dn0
ip address 10.10.50.2 255.255.255.252
interface Vlan 529
vrf forwarding dn0
```

```
ip address 10.10.50.2 255.255.255.252
interface Vlan 530
vrf forwarding dn0
ip address 10.10.50.2 255.255.255.252
interface Vlan 531
vrf forwarding dn0
ip address 10.10.50.2 255.255.255.252
interface Vlan 532
vrf forwarding dn0
ip address 10.10.50.2 255.255.255.252
interface Vlan 533
vrf forwarding dn0
ip address 10.10.50.2 255.255.255.252
interface Vlan 534
vrf forwarding dn0
ip address 10.10.50.2 255.255.255.252
interface Vlan 535
vrf forwarding dn0
ip address 10.10.50.2 255.255.255.252
interface Vlan 536
vrf forwarding dn0
ip address 10.10.50.2 255.255.255.252
interface Vlan 537
vrf forwarding dn0
ip address 10.10.50.2 255.255.255.252
interface Vlan 538
vrf forwarding dn0
ip address 10.10.50.2 255.255.255.252
interface Vlan 539
vrf forwarding dn0
```

```
ip address 10.10.50.2 255.255.255.252
interface Vlan 540
vrf forwarding dn0
ip address 10.10.50.2 255.255.255.252
interface Vlan 541
vrf forwarding dn0
ip address 10.10.50.2 255.255.255.252
interface Vlan 542
vrf forwarding dn0
ip address 10.10.50.2 255.255.255.252
interface Vlan 543
vrf forwarding dn0
ip address 10.10.50.2 255.255.255.252
interface Vlan 544
vrf forwarding dn0
ip address 10.10.50.2 255.255.255.252
interface Vlan 545
vrf forwarding dn0
ip address 10.10.50.2 255.255.255.252
interface Vlan 546
vrf forwarding dn0
ip address 10.10.50.2 255.255.255.252
interface Vlan 547
vrf forwarding dn0
ip address 10.10.50.2 255.255.255.252
interface Vlan 548
vrf forwarding dn0
ip address 10.10.50.2 255.255.255.252
interface Vlan 549
vrf forwarding dn0
```

```
ip address 10.10.50.2 255.255.255.252

ip route 0.0.0.0 0.0.0.0 10.10.40.1
ip route vrf dn0 192.168.1.0 255.255.255.0 Vlan500 10.10.50.1
ip route vrf dn0 192.168.2.0 255.255.255.0 Vlan501 10.10.51.1
ip route vrf dn1 192.168.1.0 255.255.255.0 Vlan502 10.10.52.1
ip route vrf dn1 192.168.2.0 255.255.255.0 Vlan503 10.10.53.1
ip route vrf dn2 192.168.1.0 255.255.255.0 Vlan504 10.10.54.1
ip route vrf dn2 192.168.2.0 255.255.255.0 Vlan505 10.10.55.1
ip route vrf dn3 192.168.1.0 255.255.255.0 Vlan506 10.10.56.1
ip route vrf dn3 192.168.2.0 255.255.255.0 Vlan507 10.10.57.1
ip route vrf dn4 192.168.1.0 255.255.255.0 Vlan508 10.10.58.1
ip route vrf dn4 192.168.2.0 255.255.255.0 Vlan509 10.10.59.1
ip route vrf dn5 192.168.1.0 255.255.255.0 Vlan510 10.10.60.1
ip route vrf dn5 192.168.2.0 255.255.255.0 Vlan511 10.10.61.1
ip route vrf dn6 192.168.1.0 255.255.255.0 Vlan512 10.10.62.1
ip route vrf dn6 192.168.2.0 255.255.255.0 Vlan513 10.10.63.1
ip route vrf dn7 192.168.1.0 255.255.255.0 Vlan514 10.10.64.1
ip route vrf dn7 192.168.2.0 255.255.255.0 Vlan515 10.10.65.1
ip route vrf dn8 192.168.1.0 255.255.255.0 Vlan516 10.10.66.1
ip route vrf dn8 192.168.2.0 255.255.255.0 Vlan517 10.10.67.1
ip route vrf dn9 192.168.1.0 255.255.255.0 Vlan518 10.10.68.1
ip route vrf dn9 192.168.2.0 255.255.255.0 Vlan519 10.10.69.1
ip route vrf dn10 192.168.1.0 255.255.255.0 Vlan520 10.10.70.1
ip route vrf dn10 192.168.2.0 255.255.255.0 Vlan521 10.10.71.1
ip route vrf dn11 192.168.1.0 255.255.255.0 Vlan522 10.10.72.1
ip route vrf dn11 192.168.2.0 255.255.255.0 Vlan523 10.10.73.1
ip route vrf dn12 192.168.1.0 255.255.255.0 Vlan524 10.10.74.1
ip route vrf dn12 192.168.2.0 255.255.255.0 Vlan525 10.10.75.1
ip route vrf dn13 192.168.1.0 255.255.255.0 Vlan526 10.10.76.1
```

```
ip route vrf dn13 192.168.2.0 255.255.255.0 Vlan527 10.10.77.1
ip route vrf dn14 192.168.1.0 255.255.255.0 Vlan528 10.10.78.1
ip route vrf dn14 192.168.2.0 255.255.255.0 Vlan529 10.10.79.1
ip route vrf dn15 192.168.1.0 255.255.255.0 Vlan530 10.10.80.1
ip route vrf dn15 192.168.2.0 255.255.255.0 Vlan531 10.10.81.1
ip route vrf dn16 192.168.1.0 255.255.255.0 Vlan532 10.10.82.1
ip route vrf dn16 192.168.2.0 255.255.255.0 Vlan533 10.10.83.1
ip route vrf dn17 192.168.1.0 255.255.255.0 Vlan534 10.10.84.1
ip route vrf dn17 192.168.2.0 255.255.255.0 Vlan535 10.10.85.1
ip route vrf dn18 192.168.1.0 255.255.255.0 Vlan536 10.10.86.1
ip route vrf dn18 192.168.2.0 255.255.255.0 Vlan537 10.10.87.1
ip route vrf dn19 192.168.1.0 255.255.255.0 Vlan538 10.10.88.1
ip route vrf dn19 192.168.2.0 255.255.255.0 Vlan539 10.10.89.1
ip route vrf dn20 192.168.1.0 255.255.255.0 Vlan540 10.10.90.1
ip route vrf dn20 192.168.2.0 255.255.255.0 Vlan541 10.10.91.1
ip route vrf dn21 192.168.1.0 255.255.255.0 Vlan542 10.10.92.1
ip route vrf dn21 192.168.2.0 255.255.255.0 Vlan543 10.10.93.1
ip route vrf dn22 192.168.1.0 255.255.255.0 Vlan544 10.10.94.1
ip route vrf dn22 192.168.2.0 255.255.255.0 Vlan545 10.10.95.1
ip route vrf dn23 192.168.1.0 255.255.255.0 Vlan546 10.10.96.1
ip route vrf dn23 192.168.2.0 255.255.255.0 Vlan547 10.10.97.1
ip route vrf dn24 192.168.1.0 255.255.255.0 Vlan548 10.10.98.1
ip route vrf dn24 192.168.2.0 255.255.255.0 Vlan549 10.10.99.1
snmp-server community public RO
interface Port-channel1
description portchannel 1
switchport trunk allowed vlan 500-549
switchport mode trunk
interface TwentyFiveGigE 1/0/11
description dp0p5s0
```

```
interface TwentyFiveGigE 1/0/12
description dp0p4s0
interface TwentyFiveGigE 1/0/13
description dp0p2s0
interface TwentyFiveGigE 1/0/14
description dp0p3s0
interface TwentyFiveGigE 1/0/15
description dp0p1s0
interface TwentyFiveGigE 1/0/16
description dp0p6s0
interface range TwentyFiveGigE 1/0/11-16
switchport mode trunk
switchport trunk allowed vlan add 500-539
channel-group 1 mode active
```

Παραμετροποίηση firewall στο DANOS

- vcli scipt για δημιουργία των τειχών προστασίας με 100 1000 και 10000 κανόνες αντίστοιχα

```
#!/bin/vcli -f
configure
trap "{ end_configure; }" EXIT HUP
set security firewall name firewall_100
set security firewall name firewall_1000
set security firewall name firewall_10000

for ((i=1;i<99;i++))
do
    A=$(( $RANDOM % 256 )).$(( $RANDOM % 256 )).$(( $RANDOM % 256 )).$(( $RANDOM % 255 + 1 ))
```

```
    B=$(( $RANDOM % 256 )).$(( $RANDOM % 256 )).$(( $RANDOM % 256 )).$(( $RANDOM % 255 + 1 ))

    set security firewall name firewall_100 rule $i action accept
    set security firewall name firewall_100 rule $i destination address $A
    set security firewall name firewall_100 rule $i protocol-group udp-tcp
    set security firewall name firewall_100 rule $i destination port $(($RANDOM+1))
    set security firewall name firewall_100 rule $i source address $B
    set security firewall name firewall_100 rule $i source port $(($RANDOM+1))
done

set security firewall name firewall_100 rule 99 action accept

for ((i=1;i<999;i++))
do
    A=$(( $RANDOM % 256 )).$(( $RANDOM % 256 )).$(( $RANDOM % 256 )).$(( $RANDOM % 255 + 1 ))
    B=$(( $RANDOM % 256 )).$(( $RANDOM % 256 )).$(( $RANDOM % 256 )).$(( $RANDOM % 255 + 1 ))

    set security firewall name firewall_1000 rule $i action accept
    set security firewall name firewall_1000 rule $i destination address $A
    set security firewall name firewall_1000 rule $i protocol-group udp-tcp
    set security firewall name firewall_1000 rule $i destination port $(($RANDOM+1))
    set security firewall name firewall_1000 rule $i source address $B
    set security firewall name firewall_1000 rule $i source port $(($RANDOM+1))

done

set security firewall name firewall_1000 rule 999 action accept
```

```
for ((i=1;i<9999;i++))
do
    A=$(( $RANDOM % 256 )).$(( $RANDOM % 256 )).$(( $RANDOM % 256 )).$(( $RANDOM % 255 + 1 ))
    B=$(( $RANDOM % 256 )).$(( $RANDOM % 256 )).$(( $RANDOM % 256 )).$(( $RANDOM % 255 + 1 ))

    set security firewall name firewall_10000 rule $i action accept
    set security firewall name firewall_10000 rule $i destination address $A
    set security firewall name firewall_10000 rule $i protocol-group udp-tcp
    set security firewall name firewall_10000 rule $i destination port $(($RANDOM+1))
    set security firewall name firewall_10000 rule $i source address $B
    set security firewall name firewall_10000 rule $i source port $(($RANDOM+1))

done

set security firewall name firewall_10000 rule 9999 action accept


if ! validate; then
    exit 1
fi
if ! commit; then
    exit 1
fi
```

- **vcli script για ανάθεση των τειχών προστασίας σε διεπαφές**

Για την επιλογή τείχους προστασίας που θα τοποθετηθεί στις διεπαφές αλλάζει η παράμετρος firewall_10000 στο επιθυμητό τείχος προστασίας

```
#!/bin/vcli -f
configure
```

```
trap "{ end_configure;  }" EXIT HUP

for ((i=500;i<550;i++))
do
	set interfaces bonding dp0bond0 vif $i firewall in firewall_10000
	set interfaces bonding dp0bond0 vif $i firewall out firewall_10000
done

if ! validate; then
	exit 1
fi
if ! commit;  then
	exit 1
fi
```

Παραμετροποίηση CGNAT στο DANOS

- Δημιουργία των ομάδων διευθύνσεων και θυρών για το CGNAT
- Δημιουργία των ομάδων μετάφρασης για διευθύνσεις και θύρες για το CGNAT

```
set resources group address-group AG_1
set resources group address-group AG_1 address-range 192.168.1.2 to 192.168.1.254
set resources group address-group AG_2
set resources group address-group AG_2 address-range 172.30.1.2 to 172.30.1.254
set resource group protocol-group udp-tcp protocol tcp
set resource group protocol-group udp-tcp protocol udp
set service nat pool NAT_POOL1 entry range1 ip-address range start 172.30.1.2
set service nat pool NAT_POOL1 entry range1 ip-address range end 172.30.1.254
set service nat pool NAT_POOL1 type CGNAT
set service nat pool NAT_POOL1 address-allocation round-robin
set service nat pool NAT_POOL1 address-pooling paired
set service nat pool NAT_POOL1 port dynamic-block-allocation block-size 128
set service nat pool NAT_POOL1 port dynamic-block-allocation max-blocks-per-subscriber 8
```

```
set service nat pool NAT_POOL1 port allocation sequential
set service nat pool NAT_POOL1 port range start 1024
set service nat pool NAT_POOL1 port range end 65535
set service nat pool NAT_POOL2 entry range1 ip-address range start 192.168.1.2
set service nat pool NAT_POOL2 entry range1 ip-address range end 192.168.1.254
set service nat pool NAT_POOL2 type CGNAT
set service nat pool NAT_POOL2 address-allocation round-robin
set service nat pool NAT_POOL2 address-pooling paired
set service nat pool NAT_POOL2 port dynamic-block-allocation block-size 128
set service nat pool NAT_POOL2 port dynamic-block-allocation max-blocks-per-subscriber 8
set service nat pool NAT_POOL2 port allocation sequential
set service nat pool NAT_POOL2 port range start 1024
set service nat pool NAT_POOL2 port range end 65535
```

- script για την δημιουργία πολιτικών αντιστοίχισης ομάδων διευθύνσεων με τις αντίστοιχες ομάδες μετάφρασης του CGNAT

```
#!/bin/vcli -f
configure
trap "{ end_configure; }" EXIT HUP

for ((i=1;i<27;i++))
do
    if [ $((i%2)) == 1 ]; then

        set service nat cgnat policy POLICY$i match source address-group AG_1
        set service nat cgnat policy POLICY$i priority $i
        set service nat cgnat policy POLICY$i translation pool NAT_POOL1
    else
        set service nat cgnat policy POLICY$i match source address-group AG_2
        set service nat cgnat policy POLICY$i priority $i
        set service nat cgnat policy POLICY$i translation pool NAT_POOL2
```

```
        fi
done

if ! validate; then
        exit 1
fi
if ! commit;  then
        exit 1
fi
```

- script για την ανάθεση πολιτικών στις διεπαφές

```
#!/bin/vcli -f
configure
trap "{ end_configure;  }" EXIT HUP

for ((i=500;i<550;i+=2))
do
        set service nat cgnat  interface dp0bond0.$i  policy POLICY$((($i-500)/2)+1))
done

set service nat cgnat  interface dp0p8s20 policy POLICY26

if ! validate; then
        exit 1
fi
if ! commit;  then
        exit 1
fi
```

- Ορισμός στατικών διαδρομών για το 172.30.1.0/24

```
set routing  routing-instance  vrf1 protocols static route  172.30.1.0/24 next-hop  10.10.51.2 interface
dp0bond0.501
```

set routing routing-instance vrf2 protocols static route 172.30.1.0/24 next-hop 10.10.53.2 interface dp0bond0.503

set routing routing-instance vrf3 protocols static route 172.30.1.0/24 next-hop 10.10.55.2 interface dp0bond0.505

set routing routing-instance vrf4 protocols static route 172.30.1.0/24 next-hop 10.10.57.2 interface dp0bond0.507

set routing routing-instance vrf5 protocols static route 172.30.1.0/24 next-hop 10.10.59.2 interface dp0bond0.509

set routing routing-instance vrf6 protocols static route 172.30.1.0/24 next-hop 10.10.61.2 interface dp0bond0.511

set routing routing-instance vrf7 protocols static route 172.30.1.0/24 next-hop 10.10.63.2 interface dp0bond0.513

set routing routing-instance vrf8 protocols static route 172.30.1.0/24 next-hop 10.10.65.2 interface dp0bond0.515

set routing routing-instance vrf9 protocols static route 172.30.1.0/24 next-hop 10.10.67.2 interface dp0bond0.517

set routing routing-instance vrf10 protocols static route 172.30.1.0/24 next-hop 10.10.69.2 interface dp0bond0.519

set routing routing-instance vrf11 protocols static route 172.30.1.0/24 next-hop 10.10.71.2 interface dp0bond0.521

set routing routing-instance vrf12 protocols static route 172.30.1.0/24 next-hop 10.10.73.2 interface dp0bond0.523

set routing routing-instance vrf13 protocols static route 172.30.1.0/24 next-hop 10.10.75.2 interface dp0bond0.525

set routing routing-instance vrf14 protocols static route 172.30.1.0/24 next-hop 10.10.77.2 interface dp0bond0.527

set routing routing-instance vrf15 protocols static route 172.30.1.0/24 next-hop 10.10.79.2 interface dp0bond0.529

set routing routing-instance vrf16 protocols static route 172.30.1.0/24 next-hop 10.10.81.2 interface dp0bond0.531

set routing routing-instance vrf17 protocols static route 172.30.1.0/24 next-hop 10.10.83.2 interface dp0bond0.533

set routing routing-instance vrf18 protocols static route 172.30.1.0/24 next-hop 10.10.85.2 interface dp0bond0.535

set routing routing-instance vrf19 protocols static route 172.30.1.0/24 next-hop 10.10.87.2 interface dp0bond0.537

set routing routing-instance vrf20 protocols static route 172.30.1.0/24 next-hop 10.10.89.2 interface dp0bond0.539

set routing routing-instance vrf21 protocols static route 172.30.1.0/24 next-hop 10.10.91.2 interface dp0bond0.541

set routing routing-instance vrf22 protocols static route 172.30.1.0/24 next-hop 10.10.93.2 interface dp0bond0.543

set routing routing-instance vrf23 protocols static route 172.30.1.0/24 next-hop 10.10.95.2 interface dp0bond0.545

set routing routing-instance vrf24 protocols static route 172.30.1.0/24 next-hop 10.10.97.2 interface dp0bond0.547

set routing routing-instance vrf25 protocols static route 172.30.1.0/24 next-hop 10.10.99.2 interface dp0bond0.549

Παραμετροποίηση CGNAT στο switch

- Ορισμός στατικών διαδρομών για το 172.30.1.0/24

ip route vrf dn0 172.30.1.0 255.255.255.0 Vlan500 10.10.50.1

ip route vrf dn1 172.30.1.0 255.255.255.0 Vlan502 10.10.52.1

ip route vrf dn2 172.30.1.0 255.255.255.0 Vlan504 10.10.54.1

ip route vrf dn3 172.30.1.0 255.255.255.0 Vlan506 10.10.56.1

ip route vrf dn4 172.30.1.0 255.255.255.0 Vlan508 10.10.58.1

ip route vrf dn5 172.30.1.0 255.255.255.0 Vlan510 10.10.60.1

ip route vrf dn6 172.30.1.0 255.255.255.0 Vlan512 10.10.62.1

ip route vrf dn7 172.30.1.0 255.255.255.0 Vlan514 10.10.64.1

ip route vrf dn8 172.30.1.0 255.255.255.0 Vlan516 10.10.66.1

ip route vrf dn9 172.30.1.0 255.255.255.0 Vlan518 10.10.68.1

ip route vrf dn10 172.30.1.0 255.255.255.0 Vlan520 10.10.70.1

ip route vrf dn11 172.30.1.0 255.255.255.0 Vlan522 10.10.72.1

ip route vrf dn12 172.30.1.0 255.255.255.0 Vlan524 10.10.74.1

ip route vrf dn13 172.30.1.0 255.255.255.0 Vlan526 10.10.76.1

ip route vrf dn14 172.30.1.0 255.255.255.0 Vlan528 10.10.78.1

ip route vrf dn15 172.30.1.0 255.255.255.0 Vlan530 10.10.80.1

ip route vrf dn16 172.30.1.0 255.255.255.0 Vlan532 10.10.82.1

```
ip route vrf dn17 172.30.1.0 255.255.255.0 Vlan534 10.10.84.1
ip route vrf dn18 172.30.1.0 255.255.255.0 Vlan536 10.10.86.1
ip route vrf dn19 172.30.1.0 255.255.255.0 Vlan538 10.10.88.1
ip route vrf dn20 172.30.1.0 255.255.255.0 Vlan540 10.10.90.1
ip route vrf dn21 172.30.1.0 255.255.255.0 Vlan542 10.10.92.1
ip route vrf dn22 172.30.1.0 255.255.255.0 Vlan544 10.10.94.1
ip route vrf dn23 172.30.1.0 255.255.255.0 Vlan546 10.10.96.1
ip route vrf dn24 172.30.1.0 255.255.255.0 Vlan548 10.10.98.1
```

Scripts για iperf3

- script στο Ubuntu1 για τις αρχικές δοκιμές του iperf και τις δοκιμές με τείχη προστασίας

```
time=500;
connections=(1 1 1 2 2 2 5 5 5 10 10 10 10 10 20 20 20 20 20 50 50 50 50 100 100 100 100)
b_per_c=(300M 400M 500M 300M 400M 500M 300M 400M 500M 100M 150M 200M 250M 300M 50M 75M 100M 125M 150M 20M 30M 40M 50M 10M 15M 20M 25M 5M 7M 10M 12M 2M 3M 4M 5M)
echo Iperf Bandwidth Tests > test.txt
echo UDP tests >> test.txt
for (( c=0; c<27;c++))
do
        date "+%H:%M:%S   %d/%m/%y" >> test.txt
        iperf3 -u -t 300 -P ${connections[c]} -b ${b_per_c[c]} -l 1472B -c 192.168.2.2 | tail -n 4 | head -n 3 >> test.txt
        date "+%H:%M:%S   %d/%m/%y" >> test.txt
done
echo Testing with 2 Iperf >> test.txt
for ((i=27;i<31;i++))
do
        date "+%H:%M:%S   %d/%m/%y" >> test.txt
        iperf3 -u -t 300 -P 100 -b ${b_per_c[c]} -l 1472B -c 192.168.2.2 | tail -n 4 | head -n 3 >> test.txt
        P1=$!
```

```
	iperf3 -u -t 300 -P 100 -b ${b_per_c[c]} -l 1472B -c 192.168.2.2 -p 5202 | tail -n 4 | head
-n 3 >> test.txt
	P2=$!
	wait $P1 $P2
	date "+%H:%M:%S	%d/%m/%y" >> test.txt
done
echo End of 2 Iperf test >> test.txt
echo Testing with 5 Iperf >> test.txt
for ((i=31;i<35;i++))
do
	date "+%H:%M:%S	%d/%m/%y" >> test.txt
	iperf3 -u -t 300 -P 100 -b ${b_per_c[c]} -l 1472B -c 192.168.2.2 | tail -n 4 | head -n 3 >>
test.txt
	P1=$!
	iperf3 -u -t 300 -P 100 -b ${b_per_c[c]} -l 1472B -c 192.168.2.2 -p 5202 | tail -n 4 | head
-n 3 >> test.txt
	P2=$!
	iperf3 -u -t 300 -P 100 -b ${b_per_c[c]} -l 1472B -c 192.168.2.2 -p 5203 | tail -n 4 | head
-n 3 >> test.txt
	P3=$!
	iperf3 -u -t 300 -P 100 -b ${b_per_c[c]} -l 1472B -c 192.168.2.2 -p 5204 | tail -n 4 | head
-n 3 >> test.txt
	P4=$!
	iperf3 -u -t 300 -P 100 -b ${b_per_c[c]} -l 1472B -c 192.168.2.2 -p 5205 | tail -n 4 | head
-n 3 >> test.txt
	P5=$!
	wait $P1 $P2 $P3 $P4 $P5
	date "+%H:%M:%S	%d/%m/%y" >> test.txt
done
echo End of 5 Iperf test >> test.txt
echo TCP tests >> test.txt
for (( c=0; c<27;c++))
```

```
do
        date "+%H:%M:%S   %d/%m/%y" >> test.txt
        iperf3 -t 300 -P ${connections[c]}  -b ${b_per_c[c]} -l 1460B -c 192.168.2.2 | tail -n 4 | head -n 3 >> test.txt
        date "+%H:%M:%S   %d/%m/%y" >> test.txt
done
echo Testing with 2 Iperf >> test.txt
for ((i=27;i<31;i++))
do
        date "+%H:%M:%S   %d/%m/%y" >> test.txt
        iperf3 -t 300 -P 100 -b ${b_per_c[c]} -l 1460B -c 192.168.2.2 | tail -n 4 | head -n 3 >> test.txt
        P1=$!
        iperf3 -t 300 -P 100 -b ${b_per_c[c]} -l 1460B -c 192.168.2.2 -p 5202 | tail -n 4 | head -n 3 >> test.txt
        P2=$!
        wait $P1 $P2
        date "+%H:%M:%S   %d/%m/%y" >> test.txt
done
echo End of 2 Iperf test >> test.txt
echo Testing with 5 Iperf >> test.txt
for ((i=31;i<35;i++))
do
        date "+%H:%M:%S   %d/%m/%y" >> test.txt
        iperf3 -t 300 -P 100 -b ${b_per_c[c]} -l 1460B -c 192.168.2.2 | tail -n 4 | head -n 3 >> test.txt
        P1=$!
        iperf3 -t 300 -P 100 -b ${b_per_c[c]} -l 1460B -c 192.168.2.2 -p 5202 | tail -n 4 | head -n 3 >> test.txt
        P2=$!
        iperf3 -t 300 -P 100 -b ${b_per_c[c]} -l 1460B -c 192.168.2.2 -p 5203 | tail -n 4 | head -n 3 >> test.txt
```

```
        P3=$!
        iperf3 -t 300 -P 100 -b ${b_per_c[c]} -l 1460B -c 192.168.2.2 -p 5204 | tail -n 4 | head -n 3 >> test.txt
        P4=$!
        iperf3 -t 300 -P 100 -b ${b_per_c[c]} -l 1460B -c 192.168.2.2 -p 5205 | tail -n 4 | head -n 3 >> test.txt
        P5=$!
        wait $P1 $P2 $P3 $P4 $P5
        date "+%H:%M:%S   %d/%m/%y" >> test.txt
done
echo End of 5 Iperf test >> test.txt
```

- script στο Ubuntu2 για τις όλες τις δοκιμές

```
iperf3 -s &
iperf3 -s -p 5202 &
iperf3 -s -p 5203 &
iperf3 -s -p 5204 &
iperf3 -s -p 5205 &
```

- script στο Ubuntu1 για τις δοκιμές με μέγεθος πακέτου

```
udp_p=(128B, 256B, 512B, 768B, 1024B, 1152B, 1472B)
tcp_p=(536B, 600B, 800B, 1000B, 1200B, 1460B)
echo Testing for Max Throughput > packet_size.txt
echo UDP >> packet_size.txt
for ((i=0;i<7;i++))
do
        date "+%H:%M:%S   %d/%m/%y" >> packet_size.txt
        iperf3 -u -t 120 -P 100 -b 4M -l ${udp_p[i]} -c 192.168.2.2 | tail -n 4 | head -n 3 >> packet_size.txt &
        P1=$!
        iperf3 -u -t 120 -P 100 -b 4M -l ${udp_p[i]} -c 192.168.2.2 -p 5202 | tail -n 4 | head -n 3 >> packet_size.txt &
        P2=$!
```

```
        iperf3 -u -t 120 -P 100 -b 4M -l ${udp_p[i]} -c 192.168.2.2 -p 5203 | tail -n 4 | head -n 3
>> packet_size.txt &
        P3=$!
        iperf3 -u -t 120 -P 100 -b 4M -l ${udp_p[i]} -c 192.168.2.2 -p 5204 | tail -n 4 | head -n 3
>> packet_size.txt &
        P4=$!
        iperf3 -u -t 120 -P 100 -b 4M -l ${udp_p[i]} -c 192.168.2.2 -p 5205 | tail -n 4 | head -n 3
>> packet_size.txt &
        P5=$!
        wait $P1 $P2 $P3 $P4 $P5
        date "+%H:%M:%S   %d/%m/%y >> packet_size.txt
done
echo TCP >> packet_size.txt
for ((i=0;i<6;i++))
do
        date "+%H:%M:%S   %d/%m/%y" >> packet_size.txt
        iperf3 -t 120 -P 100 -b 4M -l ${tcp_p[i]} -c 192.168.2.2 | tail -n 4 | head -n 3 >>
packet_size.txt &
        P1=$!
        iperf3 -t 120 -P 100 -b 4M -l ${tcp_p[i]} -c 192.168.2.2 -p 5202 | tail -n 4 | head -n 3 >>
packet_size.txt &
        P2=$!
        iperf3 -t 120 -P 100 -b 4M -l ${tcp_p[i]} -c 192.168.2.2 -p 5203 | tail -n 4 | head -n 3 >>
packet_size.txt &
        P3=$!
        iperf3 -t 120 -P 100 -b 4M -l ${tcp_p[i]} -c 192.168.2.2 -p 5204 | tail -n 4 | head -n 3 >>
packet_size.txt &
        P4=$!
        iperf3 -t 120 -P 100 -b 4M -l ${tcp_p[i]} -c 192.168.2.2 -p 5205 | tail -n 4 | head -n 3 >>
packet_size.txt &
        P5=$!
        wait $P1 $P2 $P3 $P4 $P5
        date "+%H:%M:%S   %d/%m/%y >> packet_size.txt
```

```
done
```

- script στο Ubuntu1 για την τυχαιότητα του διαμοιρασμού των θυρών

```
echo Randomness > random.txt
for ((i=0;i<5;i++))
do
	date "+%H:%M:%S  %d/%m/%y >> random.txt
	iperf3 -t 300 -P 10 -b 200M -M 1460B -c 192.168.2.2 -p 5205 | tail -n 4 | head -n 3 >> random.txt
done
```

Παραμετροποίηση telegraf στον Ubuntu server

- Αλλαγές στο αρχείο /etc/telegraf/telegraf.cong για συλλογή δεδομένων μέσω snmp

```
[[outputs.influxdb]]
[[inputs.snmp]]
agents = ["udp://10.10.40.2:161"]
version = 2
interval = "30s"
community = "public"
retries = 1
[[inputs.snmp.table]]
name = "switch-interface"
oid = "IF-MIB::ifXTable"
[[inputs.snmp.table.field]]
name = "ifDescr"
oid = "IF-MIB::ifDescr"
is_tag = true
[[inputs.snmp]]
agents= ["udp://10.10.40.8:161"]
version = 2
interval = "30s"
```

```
community  = “public”
retries = 1
[[inputs.snmp.table]]
name  = “danos-interface”
oid = “IF-MIB::ifXTable”
[[inputs.snmp.table.field]]
name  = “ifDescr”
oid = “IF-MIB::ifDescr”
is_tag = true
```

Βιβλιογραφία

Kurose, J. F., Ross, K. W., *Δίκτυα Υπολογιστών*, 7η Έκδοση, Μ. Γκιούρδας, Αθήνα, 2018

Silberschatz, A., Galvin, P. B. και Gagne, G., *Λειτουργικά Συστήματα*, 9η Έκδοση, Μ. Γκιούρδας, Αθήνα, 2018

Stevens, W. R., *Unix Network Programming*, Τόμος 1, 2η Έκδοση, Pearson education, 2002

Tanenbaum, A. S., Wetherall, D. J., *Δίκτυα Υπολογιστών*, 5η Έκδοση, Κλειδάριθμος, Αθήνα, 2011

Zhu, H.(επιμέλεια), *Data plane development kit: a software optimization guide to the user space-base network applications*, 1η έκδοση, CRC Press, 2021