University of Ioannina

Προηγμένες Δικτυακές Τεχνολογίες
Using sockets
Διδάσκουσα: Σπυριδούλα Μαργαρίτη

Αιμίλιος Ψαθάς
Βασίλειος Νάστος
Δημήτριος Τζούβας
Μελίνα Παπαδημητρίου

MSc in Informatics and Networks
University of Ioannina, department of informatics and Telecommunications

January 28, 2023

Περίληψη

Το παρόν άρθρο παρουσιάζει μια ανασκόπηση δύο ερευνητικών εργασιών που αποσκοπούν στη βελτίωση της απόδοσης των δικτυακών εφαρμογών, επιτρέποντάς τους να έχουν μεγαλύτερο έλεγχο της στοίβας δικτύου. Η πρώτη εργασία, "Socker: Network-Application Co-programming with Socket Tracing" (Guo et al., 2021), προτείνει ένα σύστημα που ονομάζεται Socker και χρησιμοποιεί μια τεχνική που ονομάζεται "socket tracing" για να επιτρέπει στην εφαρμογή να έχει άμεση πρόσβαση και να χειρίζεται τη στοίβα δικτύου. Αυτό επιτρέπει στην εφαρμογή να έχει μεγαλύτερο έλεγχο του δικτύου και ενδεχομένως να βελτιώσει την απόδοσή του. Η δεύτερη εργασία, "Socket Intents: Leveraging Application Awareness for Multi-Access Connectivity" (Schmidt et al., 2013), προτείνει ένα σύστημα που ονομάζεται Socket Intents και επιτρέπει στην εφαρμογή να εκφράζει τις απαιτήσεις της, όπως το εύρος ζώνης και η ασφάλεια, στο λειτουργικό σύστημα. Το λειτουργικό σύστημα μπορεί στη συνέχεια να χρησιμοποιήσει αυτές τις προθέσεις για να επιλέξει την καταλληλότερη διεπαφή δικτύου για την εφαρμογή. Και οι δύο εργασίες παρουσιάζουν έναν νέο τρόπο αλληλεπίδρασης μεταξύ της εφαρμογής και της στοίβας δικτύου, δηλαδή μέσω μιας νέας διεπαφής που επιτρέπει στην εφαρμογή να εκφράζει τις απαιτήσεις και τις προτιμήσεις της, οι οποίες στη συνέχεια χρησιμοποιούνται από το λειτουργικό σύστημα για τη βελτίωση της απόδοσης της εφαρμογής.

Η ανασκόπηση που πραγματοποιήται από την ομάδα μας, εξετάζει τα αποτελέσματα και τα συμπεράσματα αυτών των εργασιών, καθώς και τη συμβολή τους στον τομέα. Επιπλέον, αναλύει τα δύο συστήματα και τις πιθανές περιπτώσεις χρήσης και τους περιορισμούς τους, καθώς τις προτάσεις και τον τρόπο υλοποιήσης των δύο άρθρων. Συνολικά, αυτή η ανασκόπηση επικεντρώνεται στην ανάλυση των δυο άρθρων καθώς και την αξιολόγηση τους παραθέτωντας κύρια σημεία των άρθρων(τρόπος λειτουργίας, αποτελέσματα) καθώς και παρατηρήσεις στον τρόπο με τον οποίο τα δύο άρθρα αναλύουν τα θέματα τους.

Λέξεις κλειδιά — Sockets, Socker, Socket Intents

Περιεχόμενα

1 Εισαγωγή

Ο πιο σημαντικός ο συνδετικός κρίκος ανάμεσα στις εφαρμογές και στα δίκτυα είναι τα sockets. Οι αναδυόμενες εφαρμογές και τα δίκτυα προβάλλουν περισσότερες απαιτήσεις για αμφίδρομη ευαισθητοποίηση μεταξύ εφαρμογών και δικτύων οπότε και η βελτιστοποίηση των sockets ως προς το κομμάτι της ταχύτητας και τις ασφάλειας είναι απαραίτητη. Για την βελτιστοποίηση τους έχουν προταθεί δύο λύσεις από τα άρθρα τα οποία εξετάσαμε. Τα σύστηματα Socket Intents και Socker. Οι δύο λύσεις προτείνουν η βελτιστοποίηση να γίνει σε προγραμματιστικό επίπεδο με σκοπό την αύξηση της απόδοσης των δικτυακών εφαρμογών. Στη συνεχεία θα αναφερθούμε σε αυτές τις 2 υλοποιήσεις, επισημαίνοντας τα σημαντικά βήματα τις δυσκολίες που αντιμετώπισαν και τα τελικά αποτελέσματα.

2 Socker

2.1 Βασική Ιδέα

Η προσέγγιση των Socker βασίζεται στην λογική ενσωμάτωσης δικτύου-εφαρμογών σε επίπεδο υποδοχής Socker με βάση το eBPF που είναι μια επαναστατική τεχνολογία με προέλευση στον πυρήνα του Linux που μπορεί να εκτελεί προγράμματα sandboxed σε έναν πυρήνα λειτουργικού συστήματος[3]. Συσχετίζοντας υποδοχές με λειτουργίες ελέγχου δικτύου, οι προγραμματιστές μπορούν να πραγματοποιήσουν ευέλικτο έλεγχο δρομολόγησης με βάση τη λογική εφαρμογής, καθώς και δυναμική προσαρμογή λογικής εφαρμογής με βάση τις καταστάσεις του δικτύου[1].

2.2 Το ερέθισμα για την ιδέα

Οι σημερινές κατανεμημένες εφαρμογές για να είναι πιο ανταγωνιστικές στην αγορά εφαρμογών, απαιτούν υψηλή απόδοση με διάφορες μετρήσεις απόδοσης. Αυτό απαιτεί να γίνουν όλο και πιο προσαρμοσμένες ώστε να βελτιστοποιούνται μόνες τους από τη χρήση του υποκείμενου λειτουργικού συστήματος[1]. Από την άλλη, γίνονται και τα τρέχοντα δίκτυα όλο και πιο «ευφυή» δίνοντας τη δυνατότητα στους χειριστές δικτύων να καθορίζουν πολύπλοκες πολιτικές για τη λειτουργία των δικτύων με

ευέλικτο και δυναμικό τρόπο. Σε συνδυασμό με την αυξημένη δυνατότητα προγραμματισμού στο δίκτυο, φαίνεται επίσης δυνατό να ενσωματωθούν συγκεκριμένες απαιτήσεις εφαρμογής στη λογική ελέγχου δικτύου για την παροχή διαφοροποιημένων υπηρεσιών στις εφαρμογές[1]. Ως εκ τούτου, η σύζευξη των εφαρμογών και του ελέγχου του δικτύου έχει γίνει δημοφιλής τα τελευταία χρόνια.

2.3 Τι προτείνουν

Την δημιουργία του Socker ως μία καινοτόμα εφαρμογή δικτύου με βάση το eBPF. Το Socker βασίζεται στον από κοινού εντοπισμό AF, που είναι η λογική της λειτουργίας εφαρμογής, και NF, που είναι η λογική της λειτουργίας ελέγχου δικτύου[4], κάτω από το ίδιο περιβάλλον κεντρικού υπολογιστή. Με αυτόν τον τρόπο, οι εφαρμογές και ο έλεγχος του δικτύου μπορεί να ανταλλάσσει απευθείας πληροφορίες χωρίς απώλεια πληροφοριών, οπότε η συνολική απόδοση θα βελτιωθεί σημαντικά. Επίσης, αν βρίσκονται στο ίδιο πλαίσιο , στους τερματικούς κεντρικούς υπολογιστές ,η ανησυχία για την ασφάλεια ,ότι δηλαδή οι εφαρμογές θα μπορούσαν να διαρρεύσουν σημαντικές πληροφορίες προς τα έξω δεν θα υπάρχει πια[1]. Το NF θα μπορούσε να είναι το ενδιάμεσο σημείο για το αντίστοιχο AF για να επικοινωνεί τις πολιτικές δικτύου με το εκτός δίκτυο, διατηρώντας παράλληλα τις λεπτομέρειες εφαρμογής του AF μέσα. Αυτή η πρόταση απεικονίζει ένα ενιαίο πλαίσιο τόσο για την AF όσο και για NF μέσα στο κατανεμημένο σύστημα. Στο μέλλον όταν μια υπηρεσία γίνεται δημοφιλής, το δίκτυο θα βοηθά τους χρήστες εφαρμογών να βελτιωθούν να κατανοούν και να διαχειρίζονται ολόκληρο το σύστημα[1].

2.4 Οι προκλήσεις

Πολύπλοκη εξάρτηση (Complex dependency): Οι χωριστοί κώδικες AF και NF μετατρέπονται σε ένα στενά συνδεδεμένο περιβάλλον αλλά οι καθιστούν δύσκολο τον εντοπισμό όλων των σημείων όπου η AF και η NF έχουν αλληλεπιδράσεις. Μπορεί να δημιουργηθεί AF με πολλαπλές ροές σε διαφορετικούς χρόνους κάτω από διαφορετικές καταστάσεις, και το NF μπορεί να έχει πολλούς κανόνες που μπορεί να ενεργοποιηθούν σε διαφορετικές εκδηλώσεις. Η πολυπλοκότητα στη στατική ανάλυση των κανόνων NF είναι υπεύθυνη για τις υψηλές ροές AF[1].

Γρήγορος συγχρονισμός (Quick synchronization): Το συχνό πρόβλημα μεταξύ AF και NF είναι ότι επειδή εκτελούνται ανεξάρτητα στο δικό τους περιβάλλον και κατά περιόδους αντλούν πληροφορίες το ένα από το άλλο, όταν υπάρχουν

απότομες αλλαγές που συμβαίνουν στη μέση δύο έλξεων, δεν μπορούν να αντιδράσουν για να συγχρονιστούν[1]. Για την επίλυση σύνθετων εξάρτησεων των AF και NF, η δυναμική παρακολούθηση ροής χρησιμοποιείται με σκοπό τον εντοπισμό των υποκείμενων υποδοχών και τη δυναμική σύνδεση των AF και NF.

2.5 Υλοποίηση

Η λύση που προτάθηκε έχει σκοπό τον γρήγορο συγχρονισμό των AF και NF με την ενσωμάτωση τους στο ίδιο πλαίσιο ώστε να μπορούν να ανταποκριθούν σε όλες τις αλλαγές κατάστασης άμεσα. Το μόνο που χρειάζεται είναι να προσθέσουν μερικές κοινόχρηστες μεταβλητές μεταξύ τους.

Αυτή η λύση επικεντρώθηκε σε δύο τομείς:

Την Υλοποίηση στο χώρο του χρήστη(User space implementation): Όπου με διαμοιρασμό των κοινών μεταβλητών είναι πολύ εύκολο στην υλοποίηση. Ωστόσο, ο AF δεν έχει ορατότητα στο χώρο του πυρήνα, οπότε δεν γνωρίζει τις λεπτομέρειες του υποκείμενου δικτύου, δηλαδή το χαρακτηριστικό υποδοχής και την κατάσταση, μιας ροής που δημιούργησε[1].

Την Υλοποίηση στο χώρο του πυρήνα (Kernel space implementation): Όπου το eBPF προσφέρει ένα είδος δομής δεδομένων που ονομάζεται eBPF maps, οι οποίες είναι κοινόχρηστες μεταβλητές μεταξύ του χώρου του πυρήνα και του χώρου χρήστη[1].

Η υλοποίηση του Socker είναι ο συνδυασμός των δύο.

2.6 Σχεδιαστικές παρατηρήσεις

Οι συγγραφείς θεώρησαν σημαντικό η σύζευξη μεταξύ AF και NF από την άποψη του πρωτοκόλλου να είναι ελεύθερη, επίσης η ροή από ένα socket πρέπει να παρακολουθείται από τον πυρήνα κατά τον χρόνο εκτέλεσης για να είναι σε ετοιμότητα η εφεδρική θύρα.

Το Socker απαρτίζεται από έναν κοινόχρηστο δίαυλο δεδομένων

δικτύου-εφαρμογής(NAB), μια κατάλληλη βιβλιοθήκη και ένα σύνολο προγραμμάτων eBPF.

Socker Data Plane: Το επίπεδο δεδομένων Socker είναι υπεύθυνο για την εφαρμογή των επιθυμητών ενεργειών που επιστρέφονται από τον NF στα εξερχόμενα πακέτα, αυτό γίνεται με την προσάρτηση ενός προγράμματος eBPF στην κίνηση εξόδου ελέγχου (TC) [1].

Χάρτες Socker eBPF: Ο κύριος στόχος των χαρτών eBPF στο Socker είναι η αποθήκευση της αντιστοίχισης από τη ροή που δημιουργείται από τις υποδοχές χώρου χρήστη στα αποτελέσματα εκτέλεσης του NF[1].

Προγράμματα Socker eBPF: Το Socker περιέχει ένα σύνολο προγραμμάτων eBPF που συνδέονται με κλήσεις συστήματος για να παρακολουθούν τις υποδοχές και τις TCP/UDP συνδέσεις που δημιουργούνται από τις εφαρμογές, επιπλέον, επισυνάπτουν ένα πρόγραμμα eBPF στην έξοδο TC για να εκτελεί τον έλεγχο των πακέτων[1].

2.7 Ροή Εργασιών

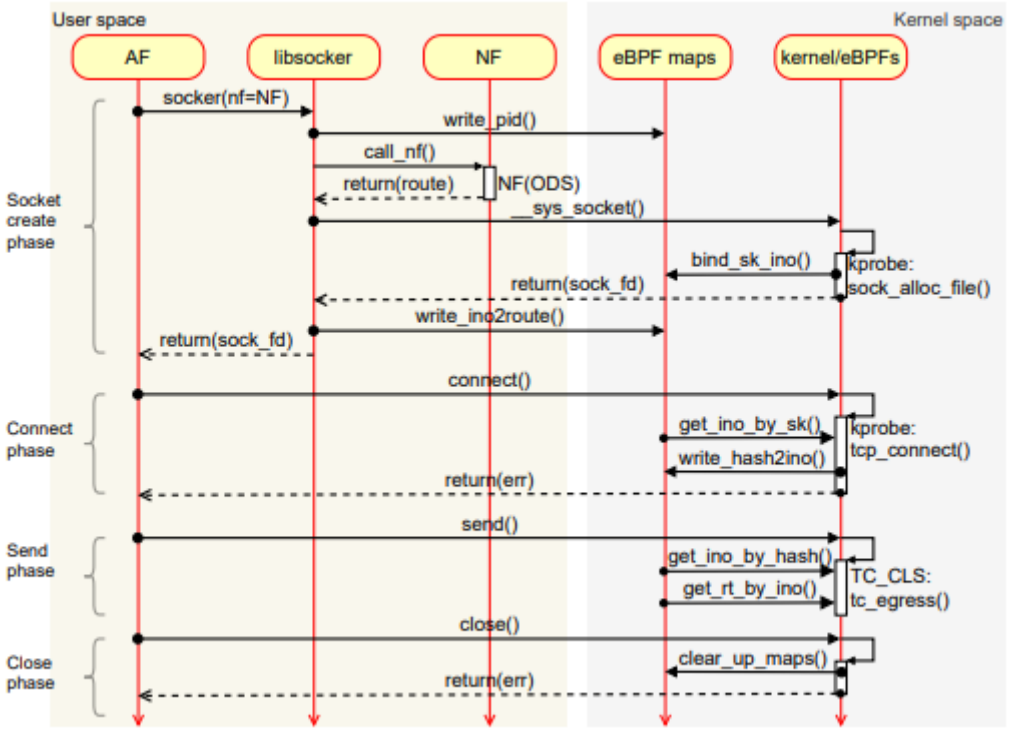
Η διαδικασία ροής εργασιών της σύνδεσης TCP στο Socker ακολουθεί τα εξής βήματα(Εικόνα 1):

1) Δημιουργία Socket: Γίνεται η κατάλληλη δημιουργία socket με χρήση συναρτήσεων και χρήση των eBPF ενώ τα κοινόχρηστα αντικείμενα αποθηκεύονται σε ένα αυτόνομο δίαυλο δεδομένων σε επίπεδο προγράμματος εξοπλισμένο με pub/sub API.Το Pub/Sub API παρέχει μια ενιαία διεπαφή για δημοσίευση και εγγραφή σε συμβάντα πλατφόρμας,[5] συμπεριλαμβανομένων συμβάντων παρακολούθησης συμβάντων σε πραγματικό χρόνο και συμβάντων λήψης δεδομένων αλλαγής.

2) Σύνδεση: Με χρήση κατάλληλων συναρτήσεων κατά την εκπομπή σήματος γίνεται παρακολούθηση από τον πυρήνα και κατάλληλη χρήση των eBPF maps.

3)Αποστολή: Όταν δημιουργηθεί η σύνδεση TCP γίνεται η αποστολή των πακέτων με υπολογισμούς από τις συναρτήσεις για τι διαδρομή που θα πάρει το πακέτο, επαναδιατυπώνοντας το πακέτο πριν σταλεί στη διεπαφή δικτύου.

4)Κλείσιμο Socket: Οταν κλείνει η υποδοχή, ο Socker καθαρίζει όλες τις σχετικές καταχωρήσεις στους χάρτες eBPF.

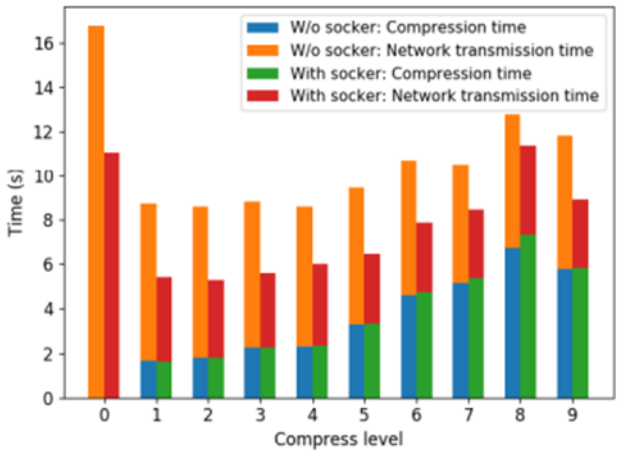


Σχήμα 1: Ροή εργασιών συστήματος Socker

2.8 Αποτελέσματα

Παρατηρούμε ότι υπό τα διαφορετικά επίπεδα συντήρησης το Socker πάντα ξεπερνά το πρόγραμμα Socket του προτύπου και καταφέρνει μείωση του χρόνο μεταφόρτωσης κατά μέσο όρο στο 28.5%[1].

Η επιτυχία βασίζεται στο ότι το Socker επιτρέπει στην εφαρμογή να επιλέγει τις βέλτιστες διαδρομές για κάθε περίπτωση. Επίσης η εφαρμογή μπορεί να προσαρμόζει το επίπεδο συμπίεσης ανάλογα με το εύρος ζώνης της διαδρομής[1].



Σχήμα 2: Ο χρόνος συμπίεσης και μετάδοσης στο δίκτυο υπό διαφορετικά επίπεδα συμπίεσης.

3 Socket Intents

3.1 Βασική Ιδέα

Η προσέγγιση του συστήματος Socket Intents βασίζεται στην ιδέα ότι στα παραδοσιακά δίκτυα, η απόφαση για το ποιο δίκτυο θα χρησιμοποιηθεί βασίζεται στην υποκείμενη δικτυακή υποδομή. Ωστόσο, στα δίκτυα πολλαπλής πρόσβασης, υπάρχουν πολλαπλές διεπαφές δικτύου διαθέσιμες στη συσκευή, όπως WiFi, κινητή τηλεφωνία και Ethernet. Οι συγγραφείς υποστηρίζουν ότι οι απαιτήσεις της εφαρμογής θα πρέπει να λαμβάνονται υπόψη όταν αποφασίζεται ποιο δίκτυο θα χρησιμοποιηθεί και όχι μόνο η υποκείμενη υποδομή. Οι συγγραφείς πρότειναν μια διεπαφή που ονομάζεται "Socket Intents" και επιτρέπει στην εφαρμογή να εκφράσει τις απαιτήσεις της στο λειτουργικό σύστημα, το οποίο μπορεί στη συνέχεια να χρησιμοποιήσει αυτές τις προθέσεις για να επιλέξει την καταλληλότερη διεπαφή δικτύου για την εφαρμογή.

3.2 Το ερέθισμα για την ιδέα

Στο σημερινό Διαδίκτυο όλες οι τελικές συσκευές έχουν πολλαπλές διεπαφές.

Αυτό επιτρέπει στους χρήστες να εναλλάσσουν την πρόσβαση απρόσκοπτα μεταξύ διαφορετικών δίκτυών ή ακόμη και να τα χρησιμοποιούν ταυτόχρονα· για καλύτερη χρήση των πόρων που έχουν στη διάθεσή τους και για την καλύτερη ικανοποίηση των αναγκών τους.[9] Η συνδεσιμότητα πολλαπλής πρόσβασης μας δίνει τη δυνατότητα να θέσουμε την ερώτηση ποια διεπαφή είναι κατάλληλη για την εκφόρτωση δεδομένων κινητής τηλεφωνίας? Το κίνητρο για την επιλογή είναι η απόδοση που παρέχει η κάθε τεχνολογία δικτύου, π.χ. ως προς το εύρος ζώνης, καθυστέρηση, διαθεσιμότητα, συμφόρηση, κόστος ανά byte και το κόστος ενέργειας[6].

3.3 Τι προτείνουν

Όλες οι λύσεις που είχαν προταθεί ως τότε είτε βασίζονταν σε πολιτικές στατικής διαμόρφωσης είτε ήταν αντιδραστικές και όχι προληπτικές όσον αφορά στις ανάγκες της εφαρμογής. Οι συγγραφείς προτείνουν μια προληπτική, ενημερωμένη εφαρμογή το Socket Intent. Ο στόχων των Socket Intents είναι να δώσουν την ελευθερία στις εφαρμογές να επιλέξουν μόνες τους την μορφή επικοινωνίας που θέλουν να έχουν, ώστε να έχουν το βέλτιστο αποτέλεσμα ως προς τα οφέλη της επιλογής τους αλλά όχι με κακόβουλο τρόπο ως προς τις άλλες εφαρμογές[6].

3.4 Από τι αποτελείτε το Socket Intents

Το σύστημά αποτελείται από

i) μια επαυξημένη βιβλιοθήκη socket για την επικοινωνία των αναγκών της εφαρμογής,

ii) ένα σύνολο πολιτικών για την επιλογή ή το συνδυασμό κατάλληλων διεπαφών δικτύου,

iii) μηχανισμούς για το συνδυασμό ή την επιλογή διεπαφών

Η βασική συνεισφορά είναι η εισαγωγή της έννοιας της υποδοχής Intents μαζί με μια πρωτότυπη υλοποίηση και μια πρώτη αξιολόγηση. Οι Socket Intents παρέχουν:

- Ένα γενικό σχήμα που επιτρέπει στις εφαρμογές να εκφράσουν τις δικές τους γνώσεις και ανάγκες για την επικοινωνία τους με μια σύνδεση.

- Μια διεπαφή για την εκπλήρωση των αναγκών των εφαρμογών σε επίπεδο βέλτιστης προσπάθειας, χωρίς να απαιτείται QoS[6].

Ποιότητα υπηρεσίας (QoS)[10] είναι η χρήση μηχανισμών ή τεχνολογιών που λειτουργούν σε ένα δίκτυο για τον έλεγχο της κυκλοφορίας και τη διασφάλιση της απόδοσης κρίσιμων εφαρμογών με περιορισμένη χωρητικότητα δικτύου. Επιτρέπει στους οργανισμούς να προσαρμόσουν τη συνολική κίνηση του δικτύου τους δίνοντας προτεραιότητα σε συγκεκριμένες εφαρμογές υψηλής απόδοσης[7].

3.5 Οι προκλήσεις

Η κύρια δυσκολία είναι ο σχεδιασμός του συστήματος για το ποιες διεπαφές θα χρησιμοποιηθούν για ποια επικοινωνία και με τι πολιτική. Προσπάθησαν να αποφύγουν το συγκεκριμένο πρόβλημα μην κάνοντας επικέντρωση σε κάποια συγκεκριμένη πολιτική, αλλά ορίζοντας ένα γενικό πλαίσιο στο οποίο πολλές πολιτικές μπορούν να εφαρμοστούν.

Με χρήση την Socket Intents δίνουν την δυνατότητα στην τρέχων πολιτική να αποφασίζει τι πρέπει να βελτιστοποιηθεί όταν προηγουμένως δεν μπορούσαν να βγάλουν πόρισμα στο τι θα πρέπει να βελτιστοποιήσει[6].

3.6 Υλοποίηση

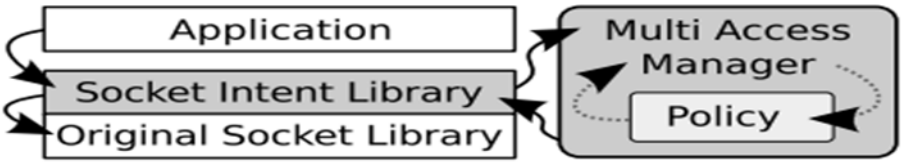
Στην βιβλιοθήκη Augmented Socket(Εικόνα 3) οι τροποποιήσεις στο Socket API είναι τρεις: εισάγουμε στις επιλογές Socket Intent, ενεργοποιούμε το MAM να επιλέξει την πηγή IP και τη διεύθυνση IP προορισμού[6].

Η υλοποίηση του Socket Intents αποτελείται από τρία στοιχεία:

- Multi Access Manager (MAM)

- τις πολιτικές
- Τη βιβλιοθήκη Socket Intent(Εικόνα 3).

Το API Socket έχει τροποποιηθεί ώστε να περιλαμβάνει επιλογές Socket Intent, οι οποίες επιτρέπουν την επιλογή των διευθύνσεων IP πηγής και προορισμού μέσω της χρήσης ενός νέου επιπέδου επιλογών socket που ονομάζεται SOL_INTENTS. Αυτή η προσέγγιση απαιτεί λιγότερες αλλαγές στο Socket API σε σύγκριση με άλλες μεθόδους. Η διεύθυνση IP πηγής επιλέγεται από μια πολιτική, με τη δυνατότητα της εφαρμογής να παρακάμψει την επιλογή μέσω της χρήσης της κλήσης bind. Η διεύθυνση IP προορισμού επιλέγεται μέσω της χρήσης μιας βιβλιοθήκης resolver που βασίζεται στη libevent σε συνδυασμό με τον MAM(Multi Access Manager). Ο επιλύτης εκτελεί ασύγχρονα την επίλυση ονομάτων μέσω πολιτικής σε όλες τις διασυνδέσεις δικτύου που καθορίζονται στο αρχείο διαμόρφωσης MAM και η συσχέτιση μεταξύ των κλήσεων υποδοχής και επίλυσης είναι απαραίτητη για την επιλογή διεύθυνσης προορισμού.



Σχήμα 3: Στοιχεία του Framework

3.7 Σχεδιαστικές παρατηρήσεις

Ο στόχος του Socket Intents είναι να ενεργοποιήσει τις εφαρμογές για να εκφράσουν τις επικοινωνιακές τους προτιμήσεις π.χ. χαμηλή καθυστέρηση ή υψηλή απόδοση και είναι προαιρετική πληροφορία. Οι προθέσεις είναι καθαρά συμβουλευτικές. Δεν προορίζονται να καθορίζουν σκληρές απαιτήσεις ή να συνεπάγονται εγγυήσεις QoS , η οποία λειτουργεί σημειώνοντας πακέτα για τον εντοπισμό τύπων υπηρεσιών και στη συνέχεια διαμορφώνοντας τους δρομολογητές ώστε να δημιουργούν ξεχωριστές εικονικές ουρές για κάθε εφαρμογή, με βάση την προτεραιότητά τους

με αποτέλεσμα, το εύρος ζώνης να δεσμεύεται για κρίσιμες εφαρμογές ή ιστότοπους στους οποίους έχει εκχωρηθεί πρόσβαση προτεραιότητας.

Τα Socket Intents είναι εμπνευσμένα από το DiffServ όπου μπορεί, για παράδειγμα, να χρησιμοποιηθεί για την παροχή χαμηλής καθυστέρησης σε κρίσιμη κυκλοφορία δικτύου, όπως η φωνή ή τα μέσα ροής , ενώ παρέχει την καλύτερη δυνατή υπηρεσία σε μη κρίσιμες υπηρεσίες, όπως η κυκλοφορία ιστού ή οι μεταφορές αρχείων, Καθώς και IntServ με την έννοια ότι καθορίζουν κατηγορίες κυκλοφορίας ανά βάση σύνδεσης, μπορεί για παράδειγμα να χρησιμοποιηθεί για να επιτρέψει στο βίντεο και στον ήχο να φτάσουν στον δέκτη χωρίς διακοπή. Με βάση αυτή τη φιλοσοφία, προτείνουν ένα σύνολο επιλογών Socket Intent(Πίνακας 1)[6][8][9].

Intent	**Type**	**Value**
Category	Enum Query	bulk transfer, control traffic, stream
File size	Int	Number of bytes transferred by the application
Duration	Int	Time between first and last packet in seconds
Bitrate	Int	Size divided by duration in bytes per second
Burstiness	Enum	Random bursts regular bursts no bursts or bulk(congestion window limited)
Timeliness	Enum	Stream (low delay, low jitter) interactive (low delay), transfer (completes eventually) or background traffic (only loose time constraint)
Resilience	Enum	Sensitive to connection loss, undesirable (loss can be handled) or resilient (loss is tolerable)

Πίνακας 1: Προτεινόμενα Socket Intents

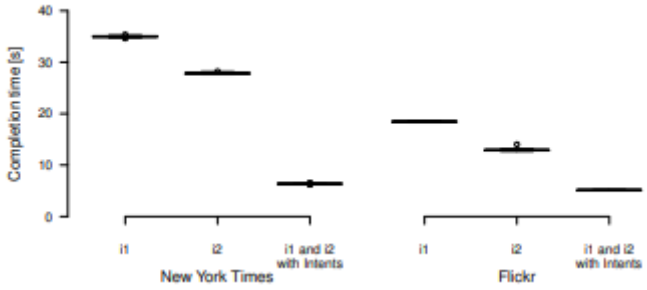
3.8 Περιπτώσεις χρήσης Socket Intents

Για την χρήση του Socket Intents σε client setup δημιούργησαν δύο σενάρια:

Σενάριο 1: Μαζική μεταφορά έναντι ερωτήματος

Στο πρώτο σενάριο εξετάστηκε ένα πρόβλημα απόδοσης που αντιμετωπίζεται συχνά στο σπίτι: περιήγηση σε έναν ιστότοπο κατά τη λήψη ενός μεγάλου αρχείου. Το πρώτο είναι κρίσιμο για τον χρόνο απόκρισης, ενώ το δεύτερο αυξάνει το εύρος ζώνης. Η πολιτική που χρησιμοποιούν εδώ είναι ότι οι "μαζικές μεταφορές" αποστέλλονται στην υψηλότερου εύρους ζώνη και τα «ερωτήματα» πάνω από τη διεπαφή χαμηλού εύρους.

Η ενεργοποίηση των πολιτικών Socket Intent βελτιώνει την απόδοση λήψης σελίδας κατά περισσότερο από 60% χωρίς κυρώσεις για τη μαζική λήψη. Ο λόγος είναι ότι η λήψη της σελίδας μπορεί πλέον να προγραμματιστεί στη διεπαφή χαμηλού εύρους και δεν ανταγωνίζεται πλέον τη μαζική λήψη[6]. Τα αποτελέσματα του πειράματος παρουσιάζονται στην εικόνα 4.



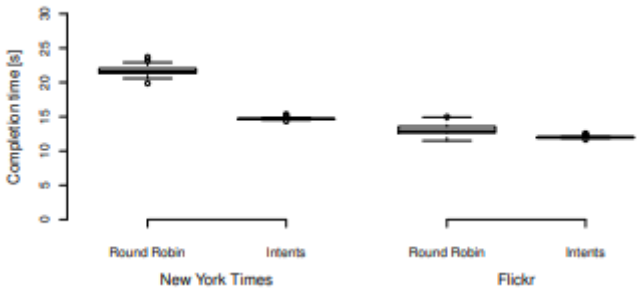
Σχήμα 4: Θηκόγραμμα των χρόνων φόρτωσης των websites(Με χρήση 2 network interfaces και intents).

Σενάριο 2: Αποφόρτωση κατά μέγεθος αρχείου

Στο δεύτερο σενάριο, οι συγγραφείς χρησιμοποιούν την πρόθεση "μέγεθος αρχείου" για ενεργοποίηση της πολιτικής επιλογής των «καλύτερων» διεπαφών δεδομένου ότι μπορεί να έχουν διαφορετικά χαρακτηριστικά. Στη διάταξη αξιολόγησης τους, μικρές μεταφορές θα πρέπει να επωφεληθούν από το χαμηλό RTT της διεπαφής χαμηλού εύρους ενώ οι μεγάλες μεταφορές ωφελούνται από το διεπαφή με το μεγαλύτερο εύρος ζώνης[6].

Ο ένας ιστότοπος επωφελείται από την εφαρμογή διαδικασίας με βελτίωση της απόδοσης κατά 20% και 18% έναντι της διεπαφής μόνο χαμηλού εύρους και της διεπαφής υψηλότερου εύρους . Ενώ ο άλλος ιστότοπος δεν έχει χειρότερη απόδοση[6].

Συμπερασματικά, αυτό το πείραμα τονίζει ότι ακόμη και μια απλή πολιτική εφαρμογής που δεν γνωρίζει μπορεί να βελτιώσει την απόδοση για συσκευές πολλαπλής πρόσβασης και οι πολιτικές με επίγνωση εφαρμογών μπορούν αποδίδουν ακόμη καλύτερη απόδοση από τις εφαρμογές που δεν γνωρίζουν[6]. Τα απολέσματα παρουσιάζονται στην εικόνα 5.



Σχήμα 5: Θηκόγραμμα των χρόνων φόρτωσης των websites(Με χρήση Round Robin και Intents).

4 Μελλοντικές επεκτάσεις

Το πρώτο άρθρο, "Socker: Network-Application Co-programming with Socket Tracing" (Guo et al., 2021), προτείνει ότι η μελλοντική εργασία θα μπορούσε να περιλαμβάνει την αξιολόγηση της απόδοσης άλλων τύπων εφαρμογών που χρησιμοποιούν το σύστημα Socker και τη διερεύνηση των δυνατοτήτων ενσωμάτωσης του Socker με άλλες τεχνολογίες δικτύωσης. Επιπλέον, προτείνουν ότι σχεδιάζουν να δοκιμάσουν το Socker με πραγματικές εφαρμογές και περιβάλλοντα για να δουν πώς αποδίδει σε αυτά τα σενάρια.

Το δεύτερο άρθρο, "Socket Intents: (Schmidt et al., 2013), προτείνει ότι οι μελλοντικές εργασίες θα μπορούσαν να περιλαμβάνουν την αξιολόγηση της απόδοσης άλλων τύπων εφαρμογών που χρησιμοποιούν το σύστημα Socket Intents, καθώς και την επέκταση του συνόλου των προθέσεων που μπορούν να εκφραστούν από την εφαρμογή και τη διερεύνηση του τρόπου με τον οποίο το σύστημα μπορεί να ενσωματωθεί σε άλλες τεχνολογίες δικτύωσης.

Όσον αφορά τη σαφήνεια των μελλοντικών εργασιών που προτείνονται σε αυτά τα άρθρα, συμπερένουμε ότι και τα δύο άρθρα παρέχουν σαφείς και συγκεκριμένες

προτάσεις για μελλοντική έρευνα. Και οι δύο περιγράφουν με σαφήνεια τους τομείς που πιστεύουν ότι θα επωφεληθούν από περαιτέρω έρευνα και παρέχουν μια σαφή πορεία για τη μελλοντική έρευνα.

5 Συμπεράσματα επί του συνόλου

Συνοψίζοντας, το συμπέρασμα του πρώτου άρθρου, είναι ότι η προτεινόμενη προσέγγιση του Socker μπορεί να επιτύχει αποτελεσματικά τον από κοινού έλεγχο των εφαρμογών και του δικτύου μέσω της χρήσης των socket intents και ενός κοινού διαύλου δεδομένων (NAB) μεταξύ των λειτουργιών της εφαρμογής και του δικτύου. Οι συγγραφείς παρουσίασαν ένα πρωτότυπο του Socker και τα αποτελέσματα έδειξαν ότι το συγκεκριμένο σύστημα μπορεί να επιτύχει αποτελεσματικά τον από κοινού έλεγχο των εφαρμογών και του δικτύου και να βελτιώσει την απόδοση του δικτύου.

Το συμπέρασμα του δεύτερου άρθρου, είναι ότι η προτεινόμενη προσέγγιση των Socket Intents είναι ένας πολλά υποσχόμενος τρόπος αξιοποίησης της επίγνωσης των εφαρμογών για τη βελτίωση του χειρισμού της συνδεσιμότητας πολλαπλής πρόσβασης. Η χρήση μιας τροποποιημένης έκδοσης του socket API μπορεί να χρησιμοποιηθεί για τη λήψη καλύτερων αποφάσεων σχετικά με τον τρόπο χειρισμού της κίνησης, με αποτέλεσμα την καλύτερη χρήση των πόρων του δικτύου και τη βελτίωση της απόδοσης της εφαρμογής. Οι συγγραφείς πρότειναν μια υλοποίηση των socket intents χρησιμοποιώντας τον πυρήνα του Linux και τα αποτελέσματα έδειξαν ότι μπορεί να επιτύχει αποτελεσματικά τον από κοινού έλεγχο των εφαρμογών και του δικτύου και να βελτιώσει την απόδοση του δικτύου.

Αναφορές

[1] Dong Guo, Shuhe Wang, Y. Richard Yang, "Socker: Network-application Co-programming with Socket Tracing", pp.14-19, 2021

[2] `https://en.wikipedia.org/wiki/Quality_of_service`

[3] `https://ebpf.io/`

[4] https://www.ericsson.com/nfv=aw.ds

[5] https://developer.salesforce.com/docs/platform/pub-sub-api/overview

[6] Philipp S. Schmidt, Theresa Enghardt, Ramin Khalili, Anja Feldmann, "Socket Intents: Leveraging Application Awareness for Multi-Access Connectivity", pp.295-300, 2013

[7] https://www.fortinet.com/resources/cyberglossary/qos-quality-of-service

[8] https://en.wikipedia.org/wiki/Differentiated_services

[9] https://en.wikipedia.org/wiki/Integrated_services

[10] Aurrecoechea, C., Campbell, A.T. and Hauw, L., 1998. A survey of QoS architectures. Multimedia systems, 6(3), pp.138-151.