National Technical University of Athens
School of Electrical and Computer Engineering
Division of Communication, Electronic and Information Engineering
Network Management and Optimal Design Laboratory

DDoS Detection using Trust-Aware Federated Learning for Heterogeneous Collaborators

Diploma Thesis

of

VASILIS PETRAKOPOULOS

Supervisor: Symeon Papavassiliou
Professor, NTUA

Athens, July 2022

National Technical University of Athens
School of Electrical and Computer Engineering
Division of Communication, Electronic and Information Engineering
Network Management and Optimal Design Laboratory

DDoS Detection using Trust-Aware Federated Learning for Heterogeneous Collaborators

Diploma Thesis

of

VASILIS PETRAKOPOULOS

Supervisor: Symeon Papavassiliou
Professor, NTUA

Approved by the examination committee on examination date.

(Signature)	*(Signature)*	*(Signature)*
....	....	....
Symeon Papavassiliou	Efstathios Sykas	Giorgos Stamou
Professor, NTUA	Professor, NTUA	Professor, NTUA

Athens, July 2022

National Technical University of Athens
School of Electrical and Computer Engineering
Division of Communication, Electronic and Information Engineering
Network Management and Optimal Design Laboratory

(Signature)

....
Vasilis Petrakopoulos

Electrical & Computer
Engineer

Περίληψη

Οι Κατανεμημένες Επιθέσεις Άρνησης Παροχής Υπηρεσιών (DDoS) αποτελούν μείζον πρόβλημα στη σύγχρονη εποχή τού διαδικτύου, καθώς στοχεύουν στην παράλυση υπηρεσιών και στην εξάντληση των πόρων που αυτές χρησιμοποιούν, καθιστώντας τις απροσπέλαστες από τους νόμιμους χρήστες τους. Η καταπολέμηση των επιθέσεων αυτών είναι μια πρόκληση, την οποία για να ξεπεράσουν οι οργανισμοί ασφαλείας συχνά χρειάζεται να συνεργαστούν και πιθανώς να μοιραστούν τα δεδομένα που διαθέτουν, για την εκπαίδευση μοντέλων με μεγαλύτερη ακρίβεια ανίχνευσης. Εντούτοις, η συνεργασία αυτή δεν μπορεί εύκολα να καταστεί εφικτή κάτω από τις παραδοσιακές συνθήκες εκπαίδευσης με Κατανεμημένη Μηχανική Μάθηση (Distributed Machine Learning), όπου τα δεδομένα εκπαίδευσης εκτίθενται σε έναν κεντρικό διακομιστή, λόγω του τεράστιου όγκου των δεδομένων αυτών, του εμπορικού ανταγωνισμού και των αυστηρών πρωτοκόλλων ιδιωτικότητας και ασφάλειας. Προκειμένου να αντιμετωπιστούν οι παραπάνω δυσκολίες, σε αυτήν τη διπλωματική προτείνουμε, σε πρώτο επίπεδο, ένα περιβάλλον συνεργατικής εκπαίδευσης που βασίζεται στη σύγχρονη μέθοδο τής Ομοσπονδιακής Μάθησης (Federated Learning).

Κατά την εκπαίδευση με Federated Learning, τα μόνα δεδομένα που ανταλλάσσονται είναι τα τοπικά μοντέλα που εκπαιδεύονται σε κάθε συνεργαζόμενη οντότητα και αποστέλλονται σε έναν κεντρικό διακομιστή για να συμψηφιστούν σε ένα νέο γενικό μοντέλο. Τα δεδομένα εκπαίδευσης παραμένουν προστατευμένα στις τοπικές συσκευές καθόλη τη διάρκεια της εκπαίδευσης. Ωστόσο, με τη διαδικασία αυτή προκύπτουν δύο ζητήματα. Αρχικά, απαιτείται σε παραδοσιακές αρχιτεκτονικές Federated Learning οι συνεργαζόμενες οντότητες να καταλήξουν ομόφωνα σε κοινή αρχιτεκτονική μοντέλου, πράγμα που μπορεί να μην είναι εύκολο εξαιτίας αφενός της ετερογένειας των δεδομένων τους και αφετέρου των εκάστοτε διαθέσιμων πόρων κάθε οντότητας. Το δεύτερο είναι το ενδεχόμενο κάποια από τις συνεγαζόμενες οντότητες να διαθέτει ελαττωματικά δεδομένα, με αποτέλεσμα να στέλνει διαφορετικά (αποκλίνοντα) μοντέλα σε σχέση με τις υπόλοιπες και να επηρεάζεται αρνητικά η όλη διαδικασία.

Στόχος, λοιπόν, της παρούσας διπλωματικής, σε δεύτερο επίπεδο, είναι η αντιμετώπιση των δύο προαναφερθέντων ζητημάτων. Για το λόγο αυτό αναπτύσσεται μια πειραματική διάταξη Federated Learning κατάλληλη για ετερογενή μοντέλα, δηλαδή μοντέλα ίδιων νευρωνικών δικτύων, αλλά με διαφορετικούς αριθμούς νευρώνων, και εμπλουτισμένη με έναν παράγοντα εμπιστοσύνης (trust) με σκοπό τη μείωση της επιρροής των συνεργατών που στέλνουν αποκλίνοντα μοντέλα. Μας ενδιαφέρει να δούμε αν μπορούμε πράγματι να χρησιμοποιήσουμε το Federated Learning αντιμετωπίζοντας τα ζητήματα αυτά.

Λέξεις Κλειδιά

Κατανεμημένες Επιθέσεις Άρνησης Παροχής Υπηρεσιών, Ανίχνευση Δικτυακών Επιθέσεων, Ομοσπονδιακή Μάθηση, Ετερογενή Μοντέλα, Ομοσπονδιακή Μάθηση με Παράγοντα Εμπιστοσύνης

Abstract

Distributed Denial of Service (DDoS) attacks constitute a major problem in the modern era of the internet, as they aim to paralyze services and exhaust the resources they use, making them inaccessible to their legitimate users. Mitigation of these attacks is a challenge that security agencies often need to work together to overcome and possibly share their data to train models with greater detection accuracy. Nevertheless, this collaboration can not easily be achieved under traditional Distributed Machine Learning training conditions, where training data is stored on a central server, due to the vast amount of such data, commercial competition and strict privacy and security protocols. In order to address the above difficulties, in this thesis we propose, at first level, a collaborative learning environment based on the modern method of Federated Learning.

During training via Federated Learning, the only data exchanged is the local models that are trained in each collaborating entity and sent to a central server to be aggregated into a new global model. Training data remains protected on local devices throughout the procedure. However, this process raises two issues. Firstly, traditional Federated Learning architectures require the collaborating entities to unanimously arrive at a common model architecture, which may not be easy due to the heterogeneity of their data on the one hand and the resources available to each entity on the other. The second one is the possibility that some of the collaborating entities have defective data, as a result of which they send different (divergent) models in relation to the others and the whole process is negatively affected.

So, the aim of the present thesis, at a second level, is to address the two aforementioned issues. For this purpose we develop an experimental Federated Learning set-up suitable for heterogeneous models, i.e. models of the same neural network, but with different numbers of neurons, and enriched with a trust factor in order to diminish the influence of collaborators sending divergent models. We are interested to see if we can actually use Federated Learning while addressing these issues.

Keywords

DDoS Attacks, Network Attacks Detection, Federated Learning, Heterogeneous Models, Trust-Aware Federated Learning

Ευχαριστίες

Θα ήθελα καταρχάς να ευχαριστήσω τον καθηγητή κ. Συμεών Παπαβασιλείου για την επίβλεψη αυτής της διπλωματικής εργασίας. Ευχαριστώ τον ομότιμο καθηγητή κ. Βασίλειο Μάγκλαρη που μου πρότεινε την εργασία αυτή και μου έδωσε την ευκαιρία να την εκπονήσω στο εργαστήριο NET-MODE (Network Management & Optimal Design Laboratory) τής σχολής ΗΜΜΥ του ΕΜΠ. Επίσης, ευχαριστώ ιδιαίτερα τον υποψήφιο διδάκτωρα Μαρίνο Δημολιάνη για την καθοδήγησή του και την εξαιρετική συνεργασία που είχαμε καθόλη τη διάρκεια της διπλωματικής αυτής. Τέλος, θα ήθελα να ευχαριστήσω γονείς, συγγενείς και φίλους για την ηθική συμπαράσταση και βοήθεια που μου προσέφεραν όλα αυτά τα χρόνια και συνέβαλαν στη δημιουργία ευχάριστων αναμνήσεων που θα τα χαρακτηρίζουν.

Βασίλης Πετρακόπουλος

Αθήνα, Ιούλιος 2022

Contents

List of Figures

List of Tables

Κεφάλαιο 1

Εκτεταμένη Ελληνική Περίληψη

1.1 Εισαγωγή

Όπως είναι γνωστό, το Διαδίκτυο χωρίζεται σε δικτυακές περιοχές που αποκαλούμε Αυτόνομα Συστήματα [7]. Κάποιες φορές, ένα τέτοιο σύστημα, που συχνά μπορεί να αποτελεί έναν οργανισμό, γίνεται θύμα δικτυακών επιθέσεων. Πιο συγκεκριμένα, οι Κατανεμημένες Επιθέσεις Άρνησης Παροχής Υπηρεσιών (DDoS) αποτελούν μείζον πρόβλημα, καθώς στοχεύουν στην παράλυση υπηρεσιών και εξάντληση πόρων, καθιστώντας τις μη προσβάσιμες από τους νόμιμους χρήστες, κάτι που μπορεί να έχει οικονομικές επιπτώσεις στις επηρεαζόμενες οντότητες. Η ανίχνευση και αντιμετώπιση τέτοιων επιθέσεων αποτελεί δύσκολο έργο για τους διαχειριστές δικτύων και τους σχεδιαστές συστημάτων ασφαλείας.

Ένας σύγχρονος τρόπος ανίχνευσης επιθέσεων είναι επιστρατεύοντας τη Μηχανική Μάθηση (Machine Learning) για τον διαχωρισμό τής δικτυακής κίνησης σε καλόβουλη και κακόβουλη. Εξαιτίας της διαφορετικότητας της δικτυακής κίνησης από περιοχή σε περιοχή, μια πετυχημένη ταξινόμηση θα απαιτούσε τη συγκέντρωση όσο το δυνατόν περισσότερων δεδομένων σε έναν κεντρικό διακομιστή. Κάτι τέτοιο αφενός είναι ασύμφορο λόγω του τεράστιου όγκου τέτοιων δεδομένων και αφετέρου η δικτυακή κίνηση αναμφισβήτητα αποτελεί προσωπικό δεδομένο, με τα αυστηρά παγκόσμια πρωτόκολλα προστασίας των δεδομένων και ιδιωτικότητας να καθιστούν τους οργανσιμούς απρόθυμους να τα μοιραστούν. Η λύση στο πρόβλημα αυτό είναι μια τεχνική που ονομάζεται Ομοσπονδιακή Μάθηση (Federated Learning) που επιτρέπει την εκπαίδευση μοντέλων με δεδομένα τα οποία μένουν αποκλειστικά στις οντότητες που συμμετέχουν.

Κατά την εκπαίδευση με Federated Learning, τα μόνα δεδομένα που ανταλλάσσονται είναι τα τοπικά μοντέλα που εκπαιδεύονται σε κάθε συνεργαζόμενη οντότητα και αποστέλλονται σε έναν κεντρικό διακομιστή για να συμψηφιστούν σε ένα νέο γενικό μοντέλο. Ωστόσο, με τη διαδικασία αυτή προκύπτουν δύο νέα ζητήματα. Το πρώτο είναι η απαίτηση οι συνεργαζόμενες οντότητες να καταλήξουν ομόφωνα σε κοινή αρχιτεκτονική μοντέλου, πράγμα που μπορεί να μην είναι εύκολο εξαιτίας αφενός της ετερογένειας των δεδομένων τους και αφετέρου των εκάστοτε διαθέσιμων πόρων κάθε οντότητας. Το δεύτερο είναι το ενδεχόμενο κάποια από τις συνεγαζόμενες οντότητες να στέλνει διαφορετικά (αποκλίνοντα) μοντέλα σε σχέση με τις υπόλοιπες, πιθανώς εξαιτίας των δεδομένων που διαθέτει, με αποτέλεσμα να επηρεάζεται αρνητικά η όλη διαδικασία. Η επίλυση αυτών των ζητημάτων θα διευκολύνει την αποτελεσματική χρήση τού Federated Learning, καθιστώντας το έναν από τους πλεόν πιο υποσχόμενους τρόπους αντιμετώπισης (DDoS) επιθέσεων σε κλίμακα παγκοσμίου επιπέδου στο Διαδίκτυο.

Επομένως, η συνεισφορά μας στο πλαίσιο αυτής της διπλωματικής είναι η υλοποίηση ενός περιβάλλοντος Federated Learning με τα παρακάτω χαρακτηριστικά:

- θα επιτρέπει στις συνεργαζόμενες οντότητες να χρησιμοποιούν ετερογενή μεταξύ τους μοντέλα, δηλαδή μοντέλα που δε θα έχουν κατ' ανάγκη ίδιους αριθμούς εισόδων και νευρώνων στο κρυμμένο επίπεδο· φυσικά, τα μοντέλα πρέπει να είναι ίδιου τύπου νευρωνικού δικτύου και με ίδιο αριθμό επιπέδων.

- θα είναι εμπλουτισμένο με έναν παράγοντα εμπιστοσύνης (trust) που θα μειώνει κατάλληλα κατά τον συμψηφισμό την επιρροή των συνεργατών των οποίων τα μοντέλα αποκλίνουν, ώστε να προκύψει τελικά το καλύτερο δυνατό μοντέλο με βάση τα χαρακτηριστικά των δεδομένων της πλειοψηφίας των συνεργατών.

Ο τελικός στόχος είναι μια διαδικασία επικερδής για κάθε συνεργαζόμενη οντότητα σε σύγκριση με το να δουλεύει μόνη της.

1.2 Θεωρητικό Υπόβαθρο

Μια Επίθεση Άρνησης Παροχής Υπηρεσιών (DoS) είναι ένα είδος κυβερνοεπίθεσης στην οποία ένας κακόβουλος χρήστης προσπαθεί να καταστήσει μια συσκευή μη διαθέσιμη προς χρήση. Οι επιθέσεις αυτές λειτουργούν πλημμυρίζοντας το στοχευμένο μηχάνημα με αιτήματα έως ότου κορεστεί και είναι πλέον αδύνατη η επεξεργασία της φυσιολογικής κίνησης [8]. Μια Κατανεμημένη Επίθεση Άρνησης Παροχής Υπηρεσιών (DDoS) είναι ένας τύπος (DoS) επίθεσης που προέρχεται από πολλές διαφορετικές πηγές, οι οποίες είναι διάφορες συσκευές τις οποίες ο επιτιθέμενος έχει μολύνει με κακόβουλο λογισμικό, ώστε να μπορεί να τις ελέγχει. Σύμφωνα με στατιστικές [9] [10] [11], η εμφάνιση τέτοιων επιθέσεων αυξάνεται και η ανάγκη για έγκαιρη ανίχνευσή τους γίνεται επιτακτική.

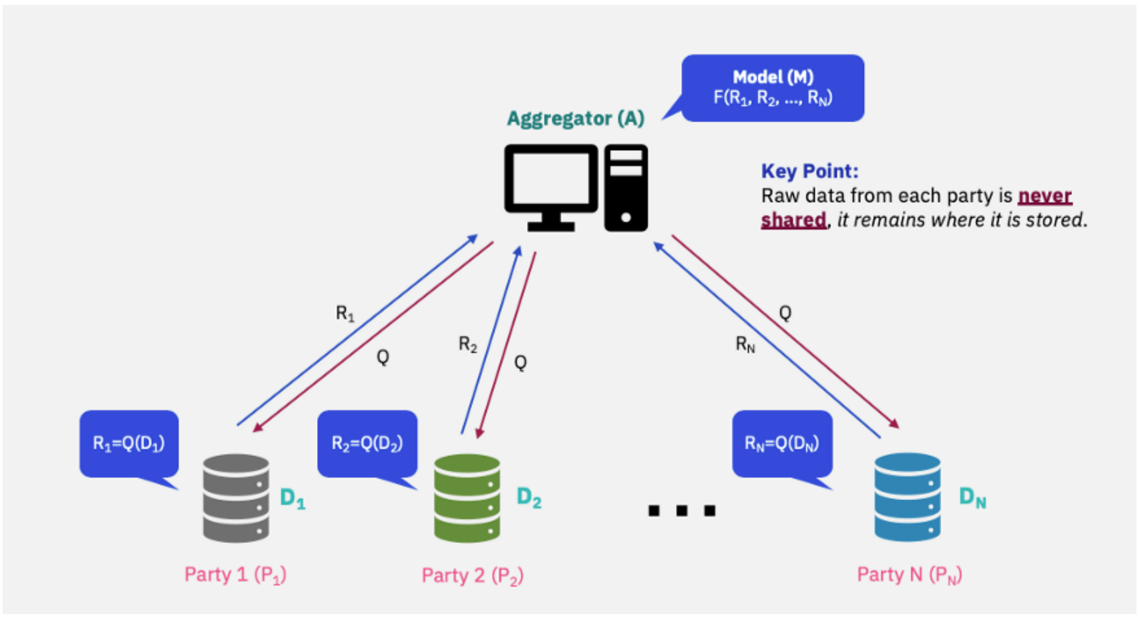

Figure 1.1. *Περιβάλλον Federated Learning. Πηγή: [1]*

Το Federated Learning προσφέρει νέες προοπτικές στον σκοπό αυτό. Επινοήθηκε από επιστήμονες της Google το 2017 [12]. Τυπικά, τα βήματα σε ένα απλό περιβάλλον Federated Learning είναι τα ακόλουθα:

- Βήμα 1: Ο κεντρικός διακομιστής αρχικοποιεί ένα γενικό μοντέλο.
- Βήμα 2: Ο κεντρικός διακομιστής στέλνει το γενικό μοντέλο σε κάθε συσκευή που συμμετέχει.
- Βήμα 3: Κάθε συσκευή εκπαιδεύει τοπικά το μοντέλο που έλαβε χρησιμοποιώντας μόνο τα δικά της δεδομένα.

- Βήμα 4: Μετά την εκπαίδευση, κάθε συσκευή στέλνει το τοπικό της μοντέλο πίσω στον κεντρικό διακομιστή.
- Βήμα 5: Αφού ο κεντρικός διακομιστής συγκεντρώσει όλα τα τοπικά μοντέλα, τα συμψηφίζει χρησιμοποιώτας κάποιον αλγόριθμο συμψηφισμού (παραδοσιακά τον Federated Averaging) και προκύπτει ένα νέο, ενημερωμένο γενικό μοντέλο.
- Βήμα 6: Ο κεντρικός διακομιστής στέλενι ξανά στις επιμέρους συσκευές το συμψηφισμένο γενικό μοντέλο, ώστε να εκπαιδευτούν τοπικά και η διαδικασία επαναλαμβάνεται μέχρι κάποιον προκαθορισμένο αριθμό εποχών ή μέχρι την ικανοποίηση κάποιου κριτηρίου.

Ένα από τα σημαντικότερα σημεία στην παραπάνω διαδικασία είναι ο αλγόριθμος συμψηφισμού. Στον Federated Averaging [13] τα βάρη τού νέου μοντέλου υπολογίζονται ως ο σταθμισμένος μέσος όρος των αντίστοιχων βαρών των τοπικών μοντέλων των συσκευών που συμμετέχουν. Η στάθμιση γίνεται με βάση τον αριθμό των δειγμάτων που περιέχουν τα σύνολα δεδομένων των επιμέρους συσκευών.

1.3 Σχετικές Εργασίες

Θα αναφέρουμε εν συντομία μερικές εργασίες σχετικές με Federated Learning και ανίχνευση DDoS επιθέσεων. Οι Zhao et al. [14] συνδύασαν το Federated Learning με το transfer learning έχοντας την ιδέα ότι κάποιες οντότητες που έχουν ειδικευτεί να αναγνωρίζουν διαφορετικές κακόβουλες ενέργειες μπορούν να συνεργαστούν για να ανιχνεύουν γενικότερα ανωμαλίες στη δικτυακή κίνηση. Οι Tian et al. [15], πρότειναν ένα «ελαφρύ» residual network για ανίχνευση επιθέσεων και κατηγοριοποίηση κίνησης. Οι συγγραφείς τού [4] υιοθέτησαν μια καινοτόμα προσέγγιση συνδυάζοντας το Federated Learning με το multi-task learning και πέτυχαν με μια μόνο εκπαίδευση να κάνουν ανίχνευση δικτυακών ανωμαλιών, αναγνώριση VPN/Tor κίνησης και κατηγοριοποίηση κίνησης, κερδίζοντας συνολικά σε κόστος χρόνου εκπαίδευσης.

Οι Diao et al. [5], πρότειναν ένα νέο σύστημα, το HeteroFL, για να αντιμετωπίσουν την ετερογένεια των συνεργαζόμενων οντοτήτων που ενδεχομένως να έχουν διαφορετική υπολογιστική ικανότητα. Η κεντρική ιδέα είναι ότι οι συνεργάτες είναι χωρισμένοι σε ομάδες ανάλογα με τις δυνατότητές τους και τα μοντέλα αυτών με την υψηλότερη υπολογιστική ικανότητα περιέχουν όλα τα βάρη, ενώ κάθε επόμενη ομάδα περιέχει ένα υποσύνολο των βαρών. Κατά τον συμψηφισμό, για κάθε βάρος η τελική τιμή του υπολογίζεται ως ο μέσος όρος μόνο όσων μοντέλων το περιέχουν.

Οι Gholami et al. [6] διατύπωσαν μια γενίκευση του Federated Averaging, την οποία ονόμασαν Trusted Federated Averaging και στην οποία έχουμε βασιστεί κι εμείς. Κατά τον συμψηφισμό, κάθε βάρος κάθε μοντέλου πολλαπλασιάζεται με έναν παράγοντα που αποτελεί τη «βαθμολογία εμπιστοσύνης» κάθε συνεργάτη. Η βαθμολογία αυτή υπολογίζεται με βάση το πόσο διαφέρει το μοντέλο ενός συνεργάτη από τα μοντέλα όλων των υπολοίπων. Για τη σύγκριση των μοντέλων χρησιμοποιείται η Ευκλείδεια απόσταση των βαρών τους.

1.4 Μεθοδολογία

Θα παρουσιάσουμε τη μεθοδολογία για τα ετερογενή μοντέλα με ένα παράδειγμα. Στην εικόνα 1.2 έχουμε δύο MLP μοντέλα 2 επιπέδων, τα οποία έχουν από ένα κοινό κι ένα διαφορετικό χαρακτηριστικό εισόδου, καθώς και διαφορετικό αριθμό νευρώνων στο κρυμμένο επίπεδο. Το συμψηφισμένο μοντέλο των μοντέλων 1 και 2 φαίνεται στην εικόνα 1.3. Το σύνολο των χαρακτηριστικών εισόδου είναι η ένωση των συνόλων των επιμέρους μοντέλων. Ο αριθμός των νευρώνων σε κάθε επίπεδο είναι τουλάχιστον τόσος όσος ο μεγαλύτερος αριθμός νευρώνων στο αντίστοιχο επίπεδο των συνεργατών.

Κάθε βάρος **w** τού συμψηφισμένου μοντέλου υπολογίζεται ως εξής:

$$\mathbf{w} = \begin{cases} \textit{μέσος όρος των σχετικών βαρών των μοντέλων που περιέχουν το } \mathbf{w} \\ 0 \textit{ αν κανένα μοντέλο δεν περιέχει το } \mathbf{w} \end{cases} \tag{1.1}$$

Λέμε ότι δοθέντος ενός βάρους **w**, ένα μοντέλο περιέχει ένα αντίστοιχο βάρος **w'** αν αυτό αφορά το ίδιο χαρακτηριστικό εισόδου (αν είναι στο πρώτο επίπεδο) και τον ίδιο νευρώνα. Φυσικά, στην εικόνα 1.3 ο 6ος νευρώνας που έχει όλα τα βάρη μηδενικά μπορεί να παραληφθεί.

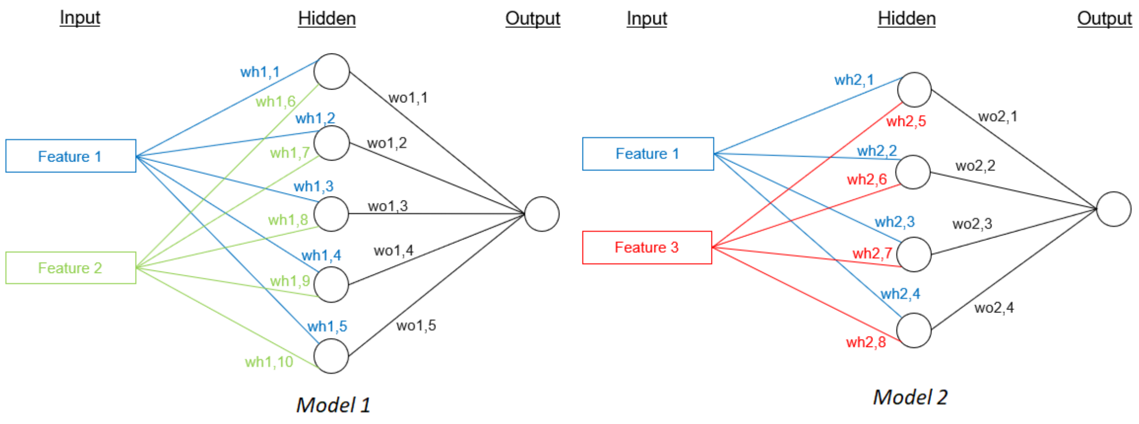

Figure 1.2. *MLP μοντέλα δύο συνεργατών προς συμψηφισμό.*

Η κεντρική ιδέα τής διαδικασίας είναι ότι οι συνεργάτες συνεισφέρουν μόνο στα βάρη που περιέχουν κι έτσι το γενικό μοντέλο μαθαίνει από κάθε συνεργάτη ό,τι αυτός μπορεί να του μάθει.

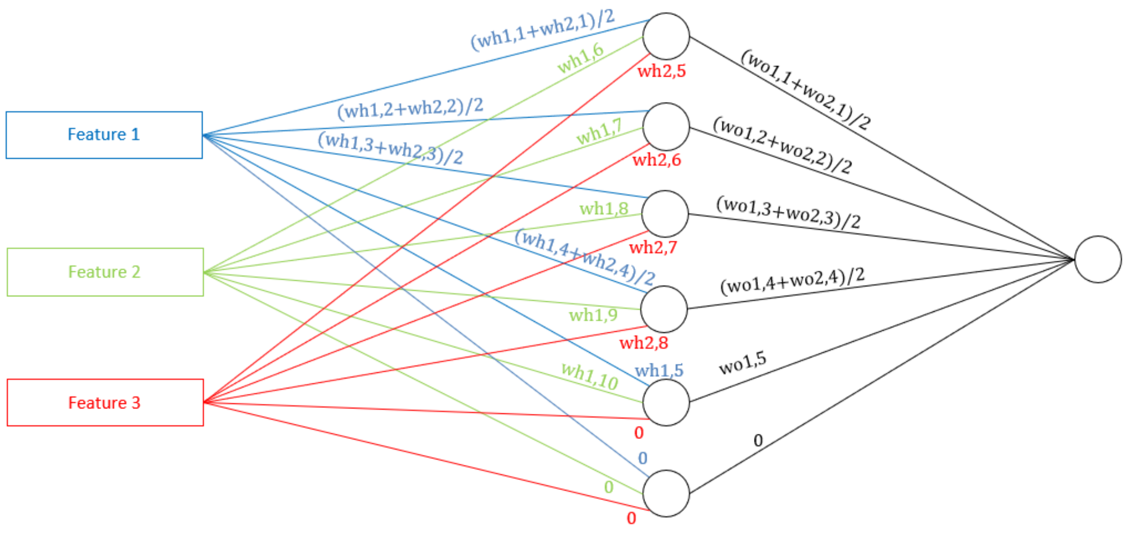

Figure 1.3. *Συμψηφισμένο μοντέλο των μοντέλων 1 και 2.*

Θα παρουσιάσουμε τώρα τη μεθοδολογία για τον παράγοντα εμπιστοσύνης. Χρειαζόμαστε μια τιμή εμπιστοσύνης για τον κάθε συνεργάτη που θα πολλαπλασιάζει κατάλληλα τα βάρη του για να μειώσει την επιρροή του κατά τη διαδικασία συμψηφισμού, αν το μοντέλο του αποκλίνει. Σε κάθε γύρο k, πριν τον συμψηφισμό, ο κεντρικός διακομιστής συγκρίνει όλα τα N τοπικά μοντέλα μεταξύ τους, όπως δείχνει η εξίσωση 1.2. Αν δύο μοντέλα είναι ετερογενή, συγκρίνονται μόνο βάρη που περιέχουν και τα δύο.

$$sum_i^{(k)} = \sum_{j \in N} \frac{||w_j^{(k)} - w_i^{(k)}||_2^2}{|N|} \tag{1.2}$$

Χρησιμοποιώντας την εξίσωση 1.3, αναθέτει τιμή 1 σε όσους συνεργάτες θεωρούνται «έμπιστοι» και 0 σε όσους αποκλίνουν.

$$I_i^{(k)} = \begin{cases} 1 & \text{αν } sum_i^{(k)} \leq th_i \times median(\{sum_j^{(k)}\}_{j \in N}) \\ 0 & \text{αλλιώς} \end{cases} \tag{1.3}$$

Από τις εξισώσεις 1.4, 1.5 και 1.6 υπολογίζεται τελικά η τιμή εμπιστοσύνης κάθε συνεργάτη.

$$r_i^{(k)} = \rho_1 r_i^{(k-1)} + I_i^{(k)} \tag{1.4}$$

$$s_i^{(k)} = \rho_2 s_i^{(k-1)} + 1 - I_i^{(k)} \tag{1.5}$$

$$t_i^{(k)} = \frac{r_i^{(k)} + 1}{r_i^{(k)} + s_i^{(k)} + 2} \tag{1.6}$$

Εξαιτίας της ετερογένειας των μοντέλων και προκειμένου να αντιμετωπιστεί η περίπτωση κάποιος να έχει ένα βάρος με «προβληματική» τιμή που ελάχιστοι ή και κανένας άλλος δεν περιέχει, προσθέτουμε έναν παράγοντα *rareFactor* που μειώνει την επιρροή των σπάνιων βαρών και έχει τιμές που φαίνονται στην παρακάτω εξίσωση.

$$rareFactor(w) = \begin{cases} 4 & \text{αν μόνο ένας συνεργάτης έχει το w} \\ 3 & \text{αν δύο συνεργάτες έχουν το w} \\ 2 & \text{αν τρεις συνεργάτες έχουν το w} \\ 1 & \text{αν τουλάχιστον τέσσερις συνεργάτες έχουν το w} \end{cases} \tag{1.7}$$

Ο τελικός τύπος για τον συμψηφισμό φαίνεται στην εξίσωση 1.8, όπου D_i είναι το μέγεθος της ιδιωτικής βάσης δεδομένων τού συνεργάτη i.

$$w^{(k)} = \sum_{i \in N} \frac{D_i t_i^{(k)}}{\sum_{i \in N} D_i t_i^{(k)}} \frac{w_i^{(k)}}{rareFactor} \tag{1.8}$$

1.5 Πειράματα και Υλοποίηση

Από το σύνολο δεδομένων για καλόβουλη κίνηση που χρησιμοποιήσαμε, επιλέξαμε 21 Αυτόνομα Συστήματα ως προορισμούς πακέτων, φτιάχνοντας έτσι 21 συνεργάτες για το Federated Learning. Σε κάθε έναν από αυτούς μοιράσαμε μια από τις 7 επιθέσεις που περιέχουν τα δεδομένα κακόβουλης κίνσης που χρησιμοποιήσαμε. Σημειώνεται εδώ, ότι η επίθεση 5 μοιάζει περισσότερο με καλόβουλη κίνηση, δεν είναι πετυχημένη και οδηγεί τον συνεργάτη που υποτίθεται ότι την αντιμετώπισε να τείνει να στέλνει ελαφρώς αποκλίνοντα μοντέλα· εδώ είναι που χρειαζόμαστε τον παράγοντα εμπιστοσύνης. Ύστερα, για κάθε συνεργάτη ξεχωριστά εκτελέσαμε στο ιδιωτικό του σύνολο δεδομένων μια διαδικασία feature selection χρησιμοποιώντας random forest από όπου προέκυψαν και τα ετερογενή μοντέλα. Όλα τα μοντέλα είναι MLP 2 επιπέδων με n εισόδους και $2n + 1$ νευρώνες στο κρυφό επίπεδο. Ως κριτήριο επίδοσης χρησιμοποιούμε την ακρίβεια (accuracy) με περισσότερο βάρος (55% έναντι 45%) στο ποσοστό αληθώς αρνητικών δειγμάτων (true negative rate), δηλαδή στην ακρίβεια στην καλόβουλη κίνηση.

Στο πρώτο πείραμα συγκρίνουμε τις επιδόσεις των συνεργατών όταν είναι ο καθένας μόνος του σε σχέση με το Federated Learning στο οποίο συνεργάζονται. Τα αποτελέσματα όταν συμμετέχουν 7 συνεργάτες φαίνονται στα παρακάτω διαγράμματα.

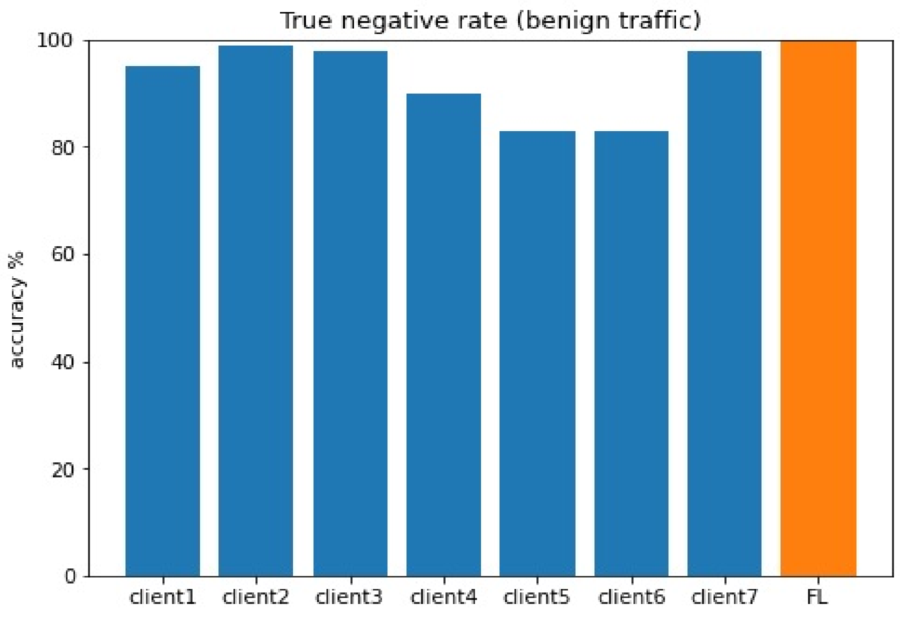

Figure 1.4. *Επίδοση 7 συνεργατών και FL μοντέλου στην καλόβουλη κίνηση.*

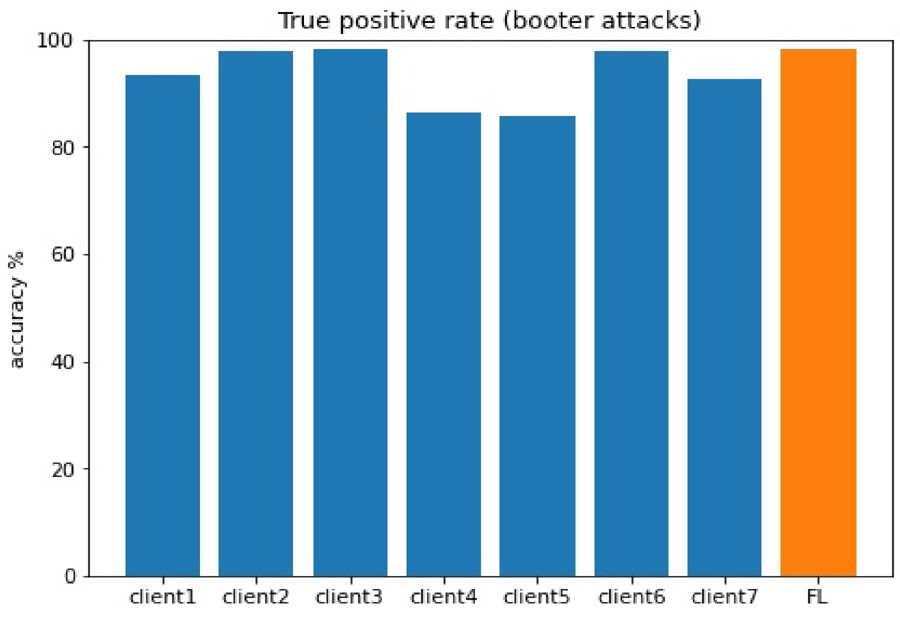

Figure 1.5. *Μέση επίδοση 7 συνεργατών και FL μοντέλου στην κακόβουλη κίνηση, εξαιρώντας την 5η επίθεση.*

Το Federated Learning (FL) μοντέλο ξεπερνά σε επίδοση όλους τους μεμονωμένους συνεργάτες στην καλόβουλη κίνηση και είναι εξίσου καλό στην κακόβουλη με τους καλύτερους από τους συνεργάτες. Παρόμοια αποτελέσματα πήραμε και όταν εκτελέσαμε το πείραμα για 14 και 21 συνεργάτες. Συμπεραίνουμε ότι υπάρχει όφελος για κάποιον συνεργάτη να συμμετέχει στη διαδικασία και να υιοθετήσει το FL μοντέλο.

Δοκιμάσαμε επίσης να εκτελέσουμε 20 φορές το πείραμα αυτό, αλλά κάθε φορά με τυχαίο (διαφορετικό) μοίρασμα των επιθέσεων στους συνεργάτες. Ύστερα πήραμε για τον κάθε συνεργάτη και για το FL μοντέλο τον μέσο όρο της επίδοσης από αυτές τις 20 εκτελέσεις. Τα αποτελέσματα φαίνονται στα παρακάτω διαγράμματα.

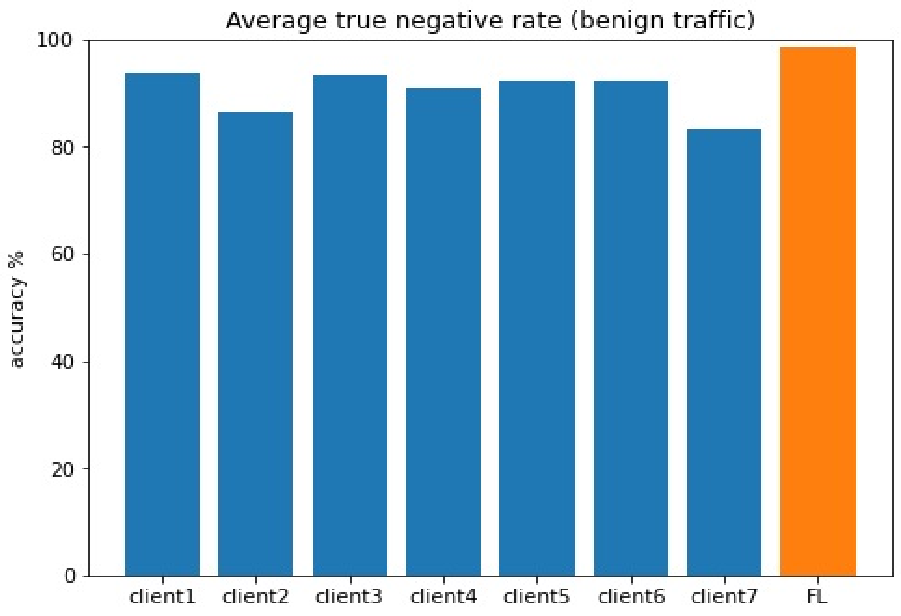

Figure 1.6. *Μέση επίδοση 7 συνεργατών και FL μοντέλου στην καλόβουλη κίνηση σε 20 εκτελέσεις με τυχαία διανομή επιθέσεων.*

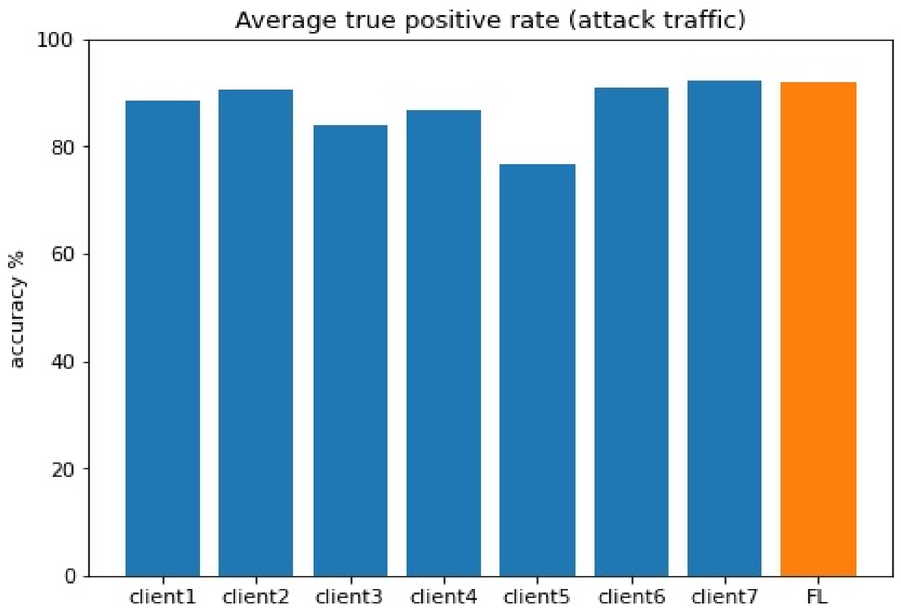

Figure 1.7. *Μέση επίδοση 7 συνεργατών και FL μοντέλου στην κακόβουλη κίνηση (μέσος όρος επιθέσεων εξαιρώντας την 5η) σε 20 εκτελέσεις με τυχαία διανομή επιθέσεων.*

Στο δεύτερο πείραμα εξετάζουμε τη χρησιμότητα του παράγοντα εμπιστοσύνης που έχουμε προσθέσει. Τα αποτελέσματα φαίνονται στο παρακάτω διάγραμμα, όπου το κέρδος είναι αναμφισβήτητο.

Στο τελευταίο πείραμα δείχνουμε τα οφέλη από τη διαδικασία του feature selection. Δεν το εκτελέσαμε απλά και μόνο προκειμένου να προκύψουν τα ετερογενή μοντέλα, αλλά αντίθετα προκείται για μια διαδικασία επιθυμητή και χρήσιμη για οποιαδήποτε οντότητα εκπαιδεύει τοπικά κάποιο μοντέλο. Το κέρδος για έναν συνεργάτη που από τα 16 αρχικά χαρακτηριστικά εισόδου, καταλήγει σε 8 μετά το feature selection φαίνεται στον πιο κάτω πίνακα. Σημειώνεται ότι δεν υπάρχει παρατηρήσιμη

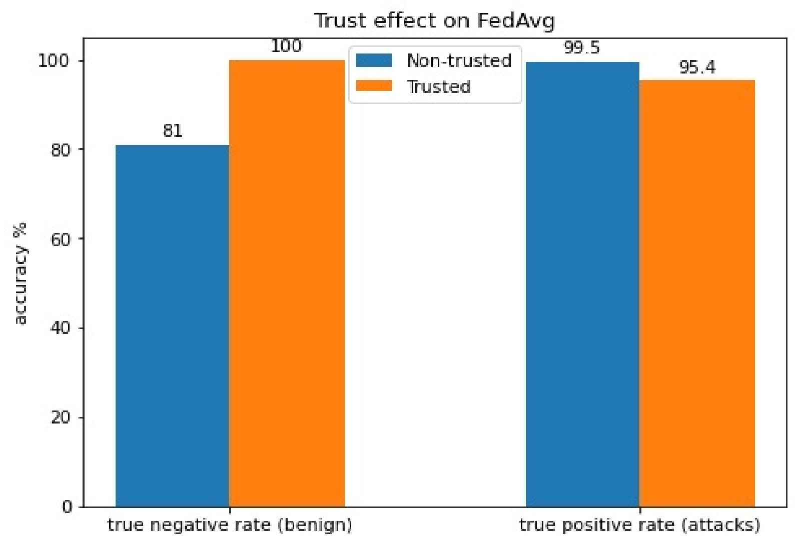

Figure 1.8. *FL χωρίς και με παράγοντα εμπιστοσύνης για 7 συνεργάτες.*

διαφορά στην επίδοση.

Υπολογιστικοί Πόροι	Χωρίς Feature Selection	Με Feature Selection	Κέρδος
Χρόνος Εκπαίδευσης	10 min	8.5 min	15%
Μνήμη RAM	510 MB	460 MB	10%
Χώρος Αποθήκευσης στο Δίσκο	3.6 MB	2.2 MB	40%

Table 1.1. *Όφελος της διαδικασίας feature selection.*

Περισσότερα αποτελέσματα και αντίστοιχη ανάλυση υπάρχουν στο κεφάλαιο Experiments and Implementation.

1.6 Συμπεράσματα

Σε αυτήν τη διπλωματική επεκτείναμε τον κλασικό αλγόριθμο του Federated Learning, προκειμένου να προσαρμόσουμε τη διαδικασία στο σύγχρονο ρεαλιστικό σενάριο της συνεργατικής ανίχνευσης DDoS επιθέσεων από οργανισμούς και γενικότερα περιοχές τού Διαδικτύου σε παγκόσμια κλίμακα. Τα αποτελέσματα μας δείχνουν μια πολλά υποσχόμενη διαδικασία που μπορεί να υλοποιηθεί και να εξυπηρετήσει τον σκοπό της στην πραγματικότητα. Επίσης, προτείνουμε τις παρακάτω κατευθύνσεις προς τις οποίες η δουλειά μας θα μπορούσε να επεκταθεί στο μέλλον:

- Διαφορετική προσέγγιση του παράγοντα εμπιστοσύνης, ώστε το σύστημα να μπορεί να διατηρήσει την επίδοσή του απέναντι σε οποιοδήποτε αποκλίνον μοντέλο, ακόμη και σε κακόβουλους συνεργάτες.

- Επέκταση της διαδικασίας με στόχο να λειτουργεί και χωρίς κάποιον κεντρικό διακομιστή. Τα μοντέλα θα ανταλλάσσονται απευθείας μεταξύ των συνεργατών και η γνώση που έχει αποκτηθεί θα διατηρείται κάθε φορά.

- Μελέτη μηχανισμών ταξινόμησης που μπορούν να αναγνωρίσουν ταυτόχρονα πολλαπλές επιθέσεις ή να εκτελέσουν κι άλλη χρήσιμη ανάλυση δικτυακής κίνησης μέσω της τεχνικής τού multi-task learning.

Chapter 2

Introduction

2.1 Motivation

As it is widely known, the Internet is divided in areas, which are smaller or bigger networks. It is not a secret that sometimes such an area, which usually is a distinct organization, may constitute a target for network attacks. In particular, Distributed Denial of Service (DDoS) attacks are a major problem, aiming to paralyze services and exhaust used resources making them inaccessible to legitimate users. With botnets recruitment by attackers side and with utilization of intelligent techniques, these attacks appear with continually increasing frequency and complexity. Marginalization of services may lead to economic consequences of the affected entities. Detection and mitigation of attacks constitutes a challenging task for network administrators and security system designers.

Attack detection essentially comes to correctly distinguishing benign and attack network traffic. As it is known, network traffic is made up of packets, which, on their turn, are nothing more than series of numbers. A classification task of this kind nowadays, seems to be a typical Machine Learning problem. Machine Learning has attracted interest of cyber security specialists in great depth, and rightly so. Of course, every Machine Learning task requires data to train models, which in this case is network traffic. However, due to diversity in both benign and attack traffic among different areas of the Internet, a successful classification would require data from as many parts or distinct organizations as possible.

Gathering of such data in order to feed a Machine Learning model is nothing but straightforward. The vast amount of network traffic data is not even the most significant difficulty. Network traffic undoubtedly constitutes personal data. Global strict protocols for data protection and security as well as unwillingness of organizations to share their network traffic data, make the whole process seem to be almost impossible. The solution to such a case, where data is preferable to stay to their owners, is a Machine Learning technique named Federated Learning.

In Federated Learning, an algorithm is trained across multiple decentralized edge devices or servers holding local data samples, without exchanging them. Each decentralized entity can be seen, in our case, as an organization or an independent network or area and from now on will be called collaborator or client of the Federated Learning process. So, each collaborator has a local Machine Learning model, which trains on local data. Then, collaborators exchange their trained models in order to calculate a new aggregated model. Collaborators take the new model and the process is repeated. Usually, a central entity orchestrates the Federated Learning and is responsible for gathering, aggregating and sending back models to collaborators.

With the use of Federated Learning, two new problems arise. Firstly, multi-domain collaborators need to agree on common model architectures. Such an agreement might not be easy to reach, since each collaborator will choose the input features based on local data and available resources. Secondly, there is the case of a collaborator sending back models different from others (i.e. with different values of weights), probably unintentionally due to different or faulty data, thus negatively affecting the aggregated model and, consequently, the whole process. Solving these problems will facilitate efficient use of Federated Learning making it one of, if not the, most promising way of detecting DDoS attacks in global Internet scale.

2.2 Thesis Contribution

In contrast with most of the existing works using Federated Learning for attack detection, which focus on designing a neural network for best possible traffic classification, in the context of this thesis, we develop a Federated Learning mechanism for DDoS attack detection with the following two novel characteristics that extend the existing traditional architecture:

- it will enable collaborators to use heterogeneous models, that is models which do not necessarily have the same number of input and hidden layer neurons; of course, models need to be of the same neural network type and have the same number of layers.
- it will be enriched with a trust factor to appropriately diminish influence during aggregation of a collaborator whose model diverges from others. The result after aggregation should be the best possible model based on the characteristics of data of the majority of collaborators.

The ultimate goal is a process beneficial for every collaborator in comparison to working alone.

2.3 Thesis Outline

Chapter 3: Theoretical Background, provides background knowledge to set the stage for the subsequent chapters. We provide an overview on the Internet, DDoS attacks and Federated Learning.

Chapter 4: Related Work, analyzes previous work on detecting attacks with Federated Learning as well as using Federated Learning with heterogeneous clients or trust.

Chapter 5: Methodology, presents the developed algorithm and the methodologies used.

Chapter 6: Experiments and Implementation, describes used datasets, contains the experiments carried out with technical details of the implementation and analyzes the results which came of.

Chapter 7: Conclusions, formulates our conclusions, summarizes our findings and provides an outlook into future work.

Chapter 3

Theoretical Background

3.1 Internet and Autonomous Systems Fundamentals

Nowadays, everyone can sense, more or less, what the Internet is. Formally, the Internet is the global system of interconnected computer networks that uses the Internet protocol suite (TCP/IP) [16] to communicate between networks and devices. It is a network of networks that consists of private, public, academic, business, and government networks of local to global scope, linked by a broad array of electronic, wireless, and optical networking technologies [17].

Going a little deeper, the Internet is divided in areas called Autonomous Systems. An Autonomous System (AS) is a collection of connected Internet Protocol routing prefixes -more simply, a collection of IPs- under the control of one or more network operators on behalf of a single administrative entity or domain, that presents a common and clearly defined routing policy to the Internet [7]. Each AS is assigned an Autonomous System number. So, an Autonomous System can be seen as an independent area of the Internet that receives and sends its characteristic network traffic. In fact, often, an organization all by itself constitutes an Autonomous System, as is the case of our institute, National Technical University of Athens, it being AS3323.

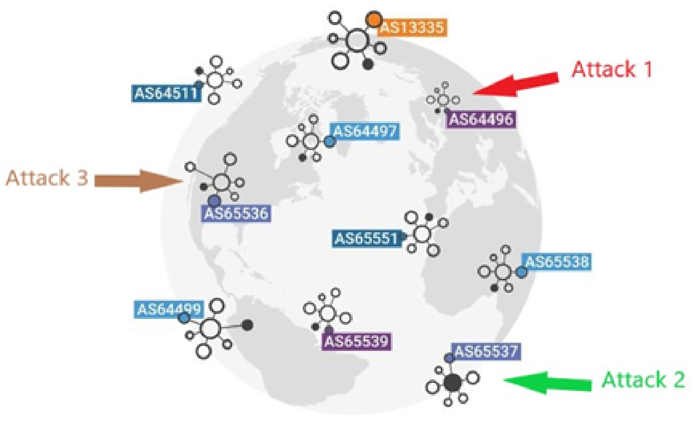

Figure 3.1. *Attacks on Different ASs of the Internet. Source: [2]*

As mentioned earlier, an Autonomous System may, at some point, be the target of a network attack. When that happens, benign receiving network traffic of the particular AS is mixed with malicious traffic of the specific attack. Of the various attacks that exist, in this thesis we are interested in DDoS attacks.

3.2 DDoS Attacks

A Denial of Service (DoS) attack is a type of cyber attack in which a malicious actor aims to render a computer or other device unavailable to its intended users by interrupting the device's normal functioning. DoS attacks typically function by overwhelming or flooding a targeted machine with requests until normal traffic is unable to be processed, resulting in denial-of-service to addition users. A DoS attack is characterized by using a single computer to launch the attack [8].

A Distributed Denial of Service (DDoS) attack is a type of DoS attack that comes from many distributed sources. These attacks achieve effectiveness by utilizing multiple compromised computer systems as sources of attack traffic. Exploited machines can include computers, routers, mobile phones and other networked resources such as IoT devices [18], which have been infected with malware, allowing them to be controlled remotely by an attacker. These individual devices are referred to as bots (or zombies), and a group of bots is called a botnet. Once a botnet has been established, the attacker is able to direct an attack by sending remote instructions to each bot. When a victim's server or network is targeted by the botnet, each bot sends requests to the target's IP address, potentially causing the server or network to become overwhelmed, resulting in a denial-of-service to normal traffic. Because each bot is a legitimate internet device, separating the attack traffic from normal traffic can be difficult [19]. It is in fact very common for attackers to hide the real IP addresses of bots and use random instead (address spoofing), so that it is impossible to locate attack sources.

These attacks often exploit operating system vulnerabilities and also some protocol features, as in the case of SYN flood attacks in three-way handshake of the TCP protocol [20]. Some other times, attacking devices pretend to be the victim of the attack and make requests to other legitimate services, which in turn overwhelm the victim with their responses, as in the case of DNS amplification attacks [21]. In [22] there is an extensive analysis and categorization of DDoS attacks based on a variety of features.

According to statistics [9] [10] [11] the emergence of DDoS attacks increases both in frequency and in volume and complexity. The need to design mechanisms for early detection and mitigation of them in global scale becomes urgent. Federated Learning provides new prospects for designing such systems.

3.3 Federated Learning

Algorithms based on Deep Learning methods need a vast amount of data to achieve desired performance. Usually, required datasets are of large scale and distributed in many sources, a fact that makes access to them from a central server inefficient and costly. So, in the age of Big Data [23], there is the challenge of implementing a distributed Machine Learning system. Furthermore, public awareness of the privacy of their personal data, has led legislators to introduce new regulations, such as the GDPR [24], on privacy protection. Recently (2017), Google scientists have proposed a new form of training that seems to respond to this problem, called Federated Learning [12]. In their study [25] they apply Federated Learning techniques for optimization of Google Gboard experience, related to word completion while typing. Each device locally trains a Machine Learning model and sends its updates to a central server, which is responsible for aggregating all local updates, creating an enhanced global model and sending it back to each device.

Federated Learning is a new Machine Learning architecture, where many clients -or collaborators- (e.g. mobile devices or entire organizations in our case) collaborate to train a model, often orchestrated by a central server, keeping training data decentralized and protected. The goal is to ensure privacy of the data, as well as to minimize the communication cost of transfer of a large amount of data as in traditional Machine Learning methods. This particular term was first used in 2016 by McMahan et al. [13]. Kairouz et al. [26], gave the following definition for Federated Learning:

*"**Federated Learning** is a machine learning setting where multiple entities (clients) collaborate in solving a machine learning problem, under the coordination of a central server or service provider. Each client's raw data is stored locally and not exchanged or transferred; instead, focused updates intended for immediate aggregation are used to achieve the learning objective."*

3.3.1 Federated Learning Cycle

Typically, the steps of training in a simple Federated Learning setting are the following:

- Step 1: Central server initializes a general global model.
- Step 2: Central server sends global model to every participating device.
- Step 3: Each device trains locally the received model using only its own data.
- Step 4: After training, each device sends the local model that came up back to the central server.
- Step 5: When central server has gathered all local models, their aggregation process begins (traditionally using Federated Averaging algorithm) and a new, updated model comes up.
- Step 6: Central server sends anew the aggregated model to every device in order to be trained locally.

After the first round of Federated Learning training, steps 2 to 6 are repeated, until a predetermined number of rounds (epochs) is completed, or a performance criterion is reached (e.g. accuracy).

As mentioned, above steps describe the cycle of a simplified Federated Learning scenario. In a real application, depending on its nature, it is possible for the number of clients to become big enough (millions of devices), to the point that communication and scalability issues arise. For this reason, as described in the study of Kairouz et al. [26], at step 2, only a subset of participating clients is selected for training per round. This selection can be made either randomly or based on some criteria, such as computational capability and speed of each device, or its reputation in relation to the contribution of its local model to the global one, in each round. Also, in a real network, it is possible that significant delays in local training come up, due to errors or a temporary increased load on a client. These clients are classified by the central server as strugglers and after the expiration of a time frame, aggregation is executed, without their local models.

3.3.2 Federated Learning Without Central Server

Recently, the possibility of setting up a Federated Learning environment, without a central entity of any kind, has been explored and referred to as decentralized (or serverless) Federated Learning. In the decentralized Federated Learning setting, the nodes are able to coordinate themselves to obtain the global model. This setup prevents single point failures as the model updates are exchanged only between interconnected nodes without the orchestration of the central server. Nevertheless, the specific network topology may affect the performances of the learning process [27]. In [28] and [29] two state-of-the-art methods for decentralized Federated Learning are described.

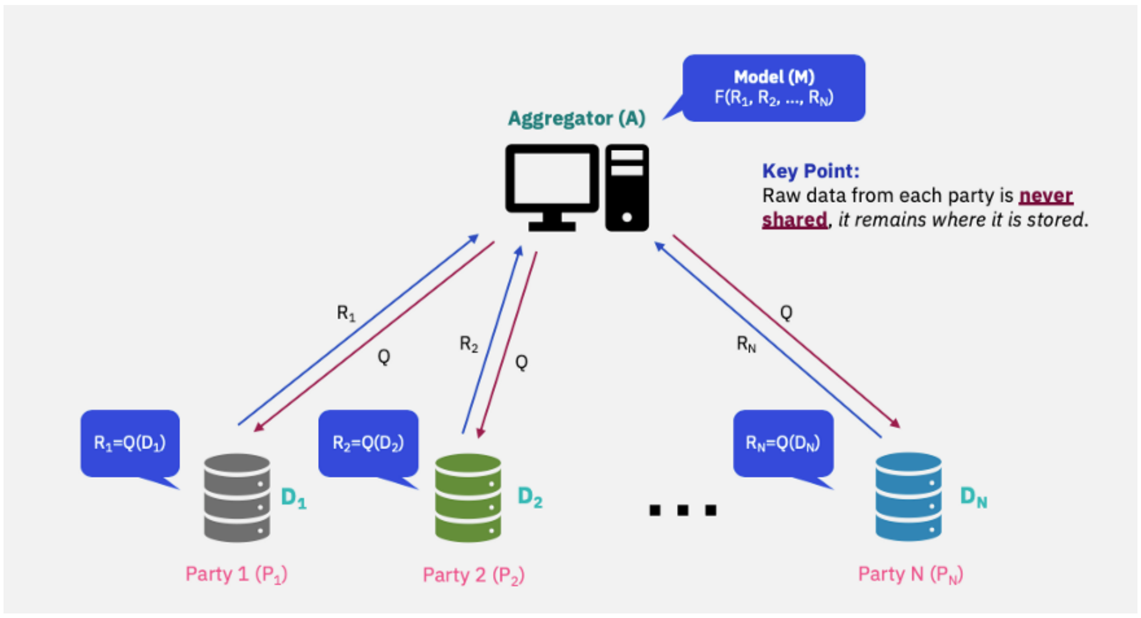

Figure 3.2. *Federated Learning Setting. Source: [1]*

3.3.3 Federated Averaging

One of the most important points in the process of Federated Learning is the way in which local models are aggregated into a new global model. The algorithm traditionally used for this purpose is named Federated Averaging. Basically, this is about a very simple process, in which global model is obtained by calculating the average of the respective weights of all local models. Hence the name given to the algorithm. In general, selection of the appropriate algorithm for calculation of the global model, plays a decisive role in every aspect of Federated Learning, affecting, either positively or negatively, the rate of convergence, communication costs and, ultimately, the accuracy of the final model. McMahan et al. [13], provide the following description of the algorithm in their work as shown in Algorithm 2.1.

From the algorithm described, we notice that the training procedure starts with the central server initializing the global model w_0, where w_t is the array of weights, i.e. the parameters that represent the model, in training round t. Then, the global model is sent to clients. Subsequently, a subset S_t of clients, who will participate in each training round, is selected. This selection concerns cases where the number of clients is large. When their number is manageable, this step can be skipped and all clients can participate in each round. Once clients have completed training of their local models, they send them back to the central server to be aggregated. The new global model w_{t+1} is calculated as the weighted average of the parameters of the respective local models. As weight factor for each local model, the ratio of samples n_k of client k to the total number of samples n of all clients is used.

3.3.4 Federated Learning vs Distributed Machine Learning

In this thesis we are interested in the advantages and disadvantages of Federated Learning in comparison to the more traditional Distributed Machine Learning techniques. So, compared to the conventional Distributed Machine Learning, when training a model in a Federated Learning environment, sending data to a central server is avoided, as a result of which the network is not burdened with a large data exchange process. Thus, training is not limited and is not delayed due to possible bottlenecks of the network. The only data exchanged in each training round are

ALGORITHM 3.1: ***FederatedAveraging***. *The K clients are indexed by k, B is the local minibatch size, E is the number of local epochs and η is the learning rate.*

```
procedure SERVEREXECUTES:
    initialize $w_0$
    for each round t = 1,2,... do
        $m \leftarrow max(C \times K, 1)$
        $S_t \leftarrow (random\ set\ of\ m\ clients)$
        for each client $k \in S_t$ in parallel do
            $w_{t+1}^k \leftarrow ClientUpdate(k, w_t)$
        end for
        $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$
    end for
end procedure

procedure CLIENTUPDATE($k, w$): // run on client k
    $B \leftarrow (split\ P_k\ into\ batches\ of\ B)$
    for each local epoch i from 1 to E do
        for batch $b \in B$ do
            $w \leftarrow w - \eta \nabla \ell(w; b)$
        end for
    end for
    return $w$ to server
end procedure
```

the parameters of local models, in order to be aggregated on the central server. The philosophy of Federated Learning encourages more and more organizations to participate in collaborative procedures to enhance their models, without worrying about potential exposure of their data, as it remains stored on local devices throughout the training. Therefore, due to greater willingness of organizations concerned to contribute, training is conducted on a wider variety of data. In fact, because of the cooperative nature of Federated Learning, more efficient use of network computing resources is observed, since many devices contribute to the training with their computing power at the same time.

On the other hand, there are some limiting factors that affect the performance of Federated Learning. Generally, in Distributed Machine Learning, central server, in order to speed up the process, divides the dataset into Independent and Identically Distributed (IID) parts of the same length each, so that they are trained in a cluster. Contrariwise, in Federated Learning data is already separated on local devices. Therefore, most of the times, the learning environment is characterized by Non-IID data, which can also vary significantly in volume. For example, suppose there are 10 organizations and each one of them has encountered a specific DDoS attack. Because of the inherently different characteristics of each attack traffic, when these organizations collaborate under a Federated Learning setting, the potentially divergent weights of local models may make the convergence difficult. According to a series of studies [30], [31] that have focused on this specific property of Federated Learning, it has been concluded that, as the heterogeneity of data between devices participating in training increases, it is more likely that the accuracy of the final global model deteriorates. This is to be expected, because the arising local models have adjusted their weights to separate different types of data on each device, resulting in significantly different parameters and, therefore, Federated Averaging algorithm converges slowly. However, except in extreme cases of heterogeneity, in most applications, the reduction of accuracy does not make the use of Federated Learning prohibitive. In fact, the benefits of greater involvement in training process, and therefore the greater variety and volume of data, are likely to outweigh this limitation. Another key feature of a real Federated Learning application -and one that we are interested about in this thesis- is the heterogeneity observed in the hardware. As it is reasonable, in a network, each device, or in our case each collaborator's available resources, have different computing capabilities and workload each time, but they still have to agree on a common model architecture.

3.3.5 Other Aggregation Algorithms

In an effort to minimize the negative impact of the issues analyzed above, Li et al. [32], developed an improved model aggregation algorithm compared to traditional Federated Averaging, FedProx. FedProx is a modification of Federated Averaging, in two aspects. Firstly, a parameter $\gamma \in [0, 1]$ is introduced, which expresses tolerance for less accurate local models. This parameter is defined differently for each client in each training round and its purpose is to reduce the loss of information due to strugglers. An increase in the value of the parameter implies an increased tolerance, so a less accurate model, considering the client for which it was set. Essentially, the goal is to combat the heterogeneity in hardware that characterizes Federated Learning architecture. Therefore, instead of eliminating the models of strugglers from aggregation, a higher tolerance γ is chosen for them, in order to calculate a local model, albeit of a lower accuracy than they would normally calculate, within the time limit set. As a result, the information we lack during the aggregation is reduced. The second point at which FedProx differs, is the introduction of a penalty constant $\mu \in [0, 1]$, so that during training local models, updates cause minor changes in the weights, to prevent them from getting too far from the global model. This way, the global model converges slowly, but when the data is highly heterogeneous, it achieves greater final accuracy, as insertion of noise is avoided.

In turn, Wang et al. [33], proposed a new algorithm which they claim achieves higher accuracy after aggregation from both Federated Averaging and FedProx. This algorithm is called FedMa (Federated Matching) and its implementation differs significantly from that of Federated Averaging, showing increased computational complexity. However, as can be seen from experiments performed in this study, the results are indeed impressive, since when comparing the three algorithms on the same applications, there is an improvement of 2-4% in accuracy compared to the already improved FedProx. In short, the central idea of FedMa, is to include during aggregation of local models, the weights that contribute the most to the performance of the model. With this approach, there is a significant reduction in the negative factor of data heterogeneity. Nevertheless, although FedMa is a promising technique, we ought to take into consideration its increasing computational complexity, as well as its generally difficult implementation, compared to the two previous algorithms analyzed.

Except of these popular alternatives, many other aggregating methods have come up recently, with even more expected to appear in following years. Each one is targeted in a very specific point of improvement. FedSGD [13], FedDist [34], FedPer [35], to name but a few.

3.3.6 Security and Privacy Issues

A key issue that Federated Learning has to address, is the protection of data privacy. As already explained, this architecture is a big step towards this direction, as client data remains stored on local devices throughout training, without the risk of exposure and interception. Recent research, however, has shown that Federated Learning alone cannot guarantee absolute data privacy. Wang et al. [36], in order to highlight its possible weaknesses, focused on the fact that clients believe that the central server responsible for aggregating local models is reliable. So, in their research they considered a scenario where the aggregating server does properly perform the procedure for which it is responsible, but at the same time, is curious to obtain information about the nature of client's data (honest-but-curious server). They developed a system, mGAN-AI, which runs on the aggregating server and by studying the local model of a client and its contribution to the performance of the global model, draws conclusions about the structure and nature of client's data. In fact, in some cases there is even the possibility of approximate reconstruction of some samples of the dataset (reverse engineering). This attack is not perceived, as it does not cause any time delay or reduction in performance.

It is therefore understood that with no unshakable evidence of reliability of the central server, there is the risk, even with low probability, of exposing a client's data. Also, when sending local models, the possibility of interference of a malicious entity, who monitors the traffic and steals the messages intended for the aggregating server, should be taken into account (gradient leakage). To eliminate this possibility, a number of techniques have been proposed, such as Differential Privacy, that can be combined with Federated Learning, so that an honest-but-curious server is completely ignorant of clients' data. Differential Privacy is a special form of encryption and is applied, in our case, to achieve the goal known in the literature as secure aggregation. The idea is based on introducing noise to the data, using special functions (e.g. Laplace noise), so that knowing the output of a calculation, it is not possible to determine the input, which in this case is the parameters of the local models [37]. Geyer et al. [38], studied the contribution of Differential Privacy to Federated Learning and concluded that with a small trade-off against accuracy (1%), potential threat of a malicious aggregating server is minimized.

Another threat that Federated Learning has to deal is the so-called data poisoning and model poisoning attacks. These attacks are about intentionally changing the data (e.g. changing the label of certain samples to belong to the wrong class) and modifying the parameters of a local model to reduce its accuracy. The modifications are made very carefully and to a limited extent, so that the anomalies that occur in relation to the genuine local models are not perceived. Such actions are usually performed by malicious clients who participate in the collaborative scheme, with the sole purpose of degrading the quality of training and the accuracy of the final global model. Bhagoji et al. [39], studied various strategies for orchestrating such attacks and concluded that even a malicious presence in a much larger set of clients, could result in a respectable reduction of the accuracy of final model. In addition, they concluded that model poisoning attacks are more effective and more difficult to detect. To combat them, systems have been proposed, such as Auror [40], which detects altered local models and excludes them from aggregating. In fact, its creators managed in their experiments, using Auror, to limit accuracy reduction to 3%, even when 30% of clients are malicious. On the other hand, Chen et al. [41], in order to eliminate the presence of malicious clients, developed a protocol for executing Federated Learning in a Trusted Execution Environment. Each client is checked during aggregation for whether it implements the proposed security protocol, through a special signature. For training of local model, each client is obliged by the protocol to allocate an isolated memory area on his device, where the necessary calculations are performed, without intervention of other programs. If this condition is met, the message to be sent is sealed with the signature indicating the integrity and authenticity of the calculations.

The issue of poisoning attacks is indeed very close to one of the problems which interest us in this thesis and that is to diminish the influence of clients who send divergent local models for aggregation. We are more interested in the case that something like this happens unintentionally, probably due to faulty or simply different data, since we do not expect an entire Autonomous System or a well-known organization to be malicious. Gholami et al. [6], introduced a generalization of Federated Averaging, named Trusted Federated Averaging, to tackle this problem. We will extensively refer to their work later on.

Chapter 4

Related Work

Almost all works using Federated Learning (FL) for DDoS attack detection focus on finding a -usually complex- neural network that will give the best possible traffic classification in terms of a metric like accuracy. Our approach is different. We use simple MLP models and our goal is to extend the conventional Federated Learning architecture in two ways. On the one hand, we want to facilitate federated training of heterogeneous clients, i.e. clients that do not necessarily share the exact same model architecture, although they have to use the same type of neural network and number of layers. On the other hand, we want to add a trust factor in the aggregating process to diminish the influence of clients whose model weights diverge, i.e. have different values than the majority of clients. To the best of our knowledge, this is the first work that combines together at least two of the following topics: attack detection using FL, heterogeneous FL, trust-aware FL aggregation algorithm.

4.1 Federated Learning and DDoS Attack Detection

We will very briefly mention some of the works related to Federated Learning and DDoS attack detection. Dimolianis [3], in his doctoral dissertation, among else, leverages the Federated Learning paradigm for collaborative and privacy-preserving DDoS detection. To mitigate attacks within collaborating Autonomous Systems (ASs), he proposes efficient, scalable and programmable firewalls that can be instantiated on-demand upon request of the AS hosting the victim. His schema consists of a detection and mitigation application mounted in all collaborating domains, as shown in 4.1. The former detects malicious packet signatures, i.e. combinations of packet field values, using Multilayer Perceptrons (MLPs); these are cooperatively trained without exposing private data. The latter filters malicious packets using XDP-enabled firewalls deployed in the victim AS; mitigation can also be activated on-demand within collaborating transit ASs. His approach was evaluated both in terms of packet classification accuracy and packet processing performance using both real and synthetic network traces.

Zhao et al. [14], combined FL with transfer learning. The idea is that some entity may have specialized, for instance, in mitigation of DDoS attacks and some other in malware. So, after the stage of FL, one can use transfer learning to improve performance in detecting network anomalies in general. In [15], Tian et al. proposed a lightweight residual network (ResNet), which except for detection, is also suitable for DDoS classification. As they claim, their network is as efficient as an LSTM (Long Short-Term Memory), but lighter and faster. It is interesting to compare their 9-layer light network with our only 2-layer MLPs that we have used. Li et al. [42], created the federated learning empowered mitigation architecture (FLEAM) to advocate joint defense, incurring a higher hacking expense and making attacker to give up. FLEAM combines FL and fog computing to reduce mitigation time and improve detection accuracy, enabling defenders to jointly combat botnets. Each edge node trains locally its data using GRU models and then

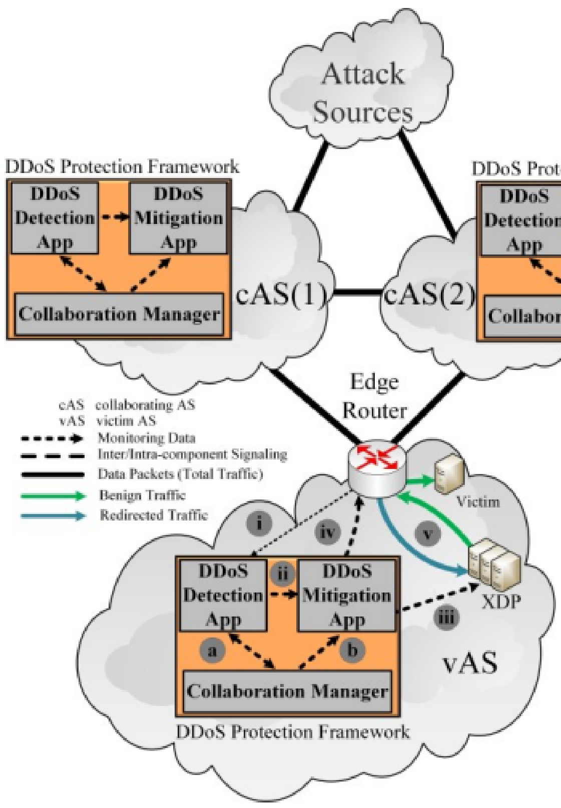

Figure 4.1. *Collaborative DDoS Detection & Mitigation Architecture. Source: [3]*

uploads its parameters to cloud. In [4], writers adopted an innovating approach combining FL with multi-task learning. This way, with only one training procedure, they manage to do network anomaly detection, VPN/Tor traffic recognition and traffic classification, having total profit in terms of training time cost, as shown in 4.2.

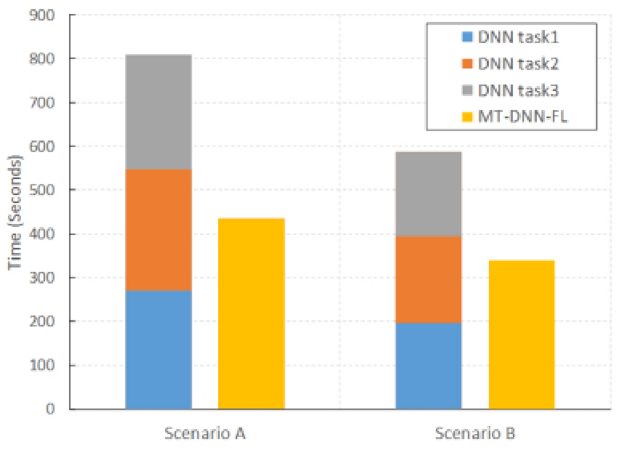

Figure 4.2. *Training time cost of MT-DNN-FL (proposed network) and DNN (ordinary networks). Source: [4]*

In [43], writers propose an architecture with programmable data plane switches. They show that Binarized Neural Networks (BNNs) can be implemented as switch functions at the network edge, classifying incoming packets at the line speed of the switches. To train BNNs in a scalable manner, they adopt a Federated Learning approach that keeps the communication overheads of training small even for scenarios involving many edge network domains. They next develop a prototype using the P4 language and perform evaluations, with results demonstrating that a multi-fold improvement in latency and communication overheads can be achieved compared to state-of-the-art learning architectures. Finally, in [44] there is a comprehensive presentation of various works related to Federated Learning and Intrusion Detection Systems.

4.2 Heterogeneous Federated Learning

Federated Learning for heterogeneous clients is a very recent idea. In fact, at the beginning of our research, the only relevant known work is the one on which we have based our approach and we are going to analyze. Diao et al. [5], proposed a new Federated Learning framework named HeteroFL to address heterogeneous clients equipped with different computation and communication capabilities. Their solution can enable the training of heterogeneous local models with varying computation complexities and still produce a single global inference model. As they claim, for the first time, their computation and communication efficient method challenges the underlying assumption of existing work that local models have to share the exact same architecture as the global model. The main idea is that clients are divided in groups of different computational complexity levels, depending on their capabilities. Models of group with the highest computational complexity level contain all weights. Then, models of each group contain a subset of the weights of the group with immediately higher complexity, as seen in 4.3.

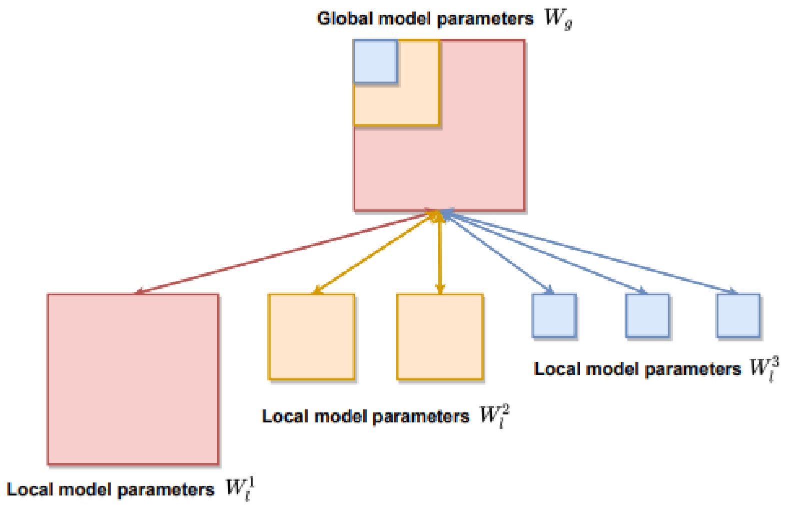
Global model parameters W_g
Local model parameters W_l^1
Local model parameters W_l^2
Local model parameters W_l^3

Figure 4.3. *Global model parameters W_g are distributed to $m = 6$ local clients with $p = 3$ computational complexity levels. Source: [5]*

What happens during aggregation is that for each weight parameter, the final value is calculated as the average of only the models of clients which contain this weight. This way, each client only contributes to weights its model contain according to its computational capability. In order to test their methods, writers ran various experiments and concluded that "weak learners" (i.e. clients with lower computational capability) can benefit when collaborating with "strong learners" (i.e. clients with higher computational capability) without reducing the latter's performance.

4.3 Federated Learning and Trust

Apart from what we mentioned earlier in Security and Privacy Issues, there are currently few works relevant to Trust in Federated Learning. Cao et al. [45], focused on the fact that there is no root trust between central server and clients. So, they proposed that the central server collects a clean bootstrap dataset, in which trains its model, and when it receives updates from clients, checks if any of them diverges from its model. Based on how much each local model differs from the server's, it assigns appropriate trust scores, which show how much will each client be taken into account during aggregation. In [46], reinforcement learning techniques are used, so that the central server trains a data value estimator of the clients' gradients to determine the contribution of each client. In [47], writers propose the building of the Accuracy Approximation Model, which estimates a simulated test accuracy using inputs of sampled data size and extracts the clients' data quality and data size to measure client contribution.

We adopt a different approach based on the work of Gholami et al. [6]. As we have stated before, in the aforementioned paper, writers formulated a generalization of Federated Averaging, which they called Trusted Federated Averaging (Trusted FedAvg). During aggregation, every weight of each model is multiplied by a factor, which is the trust score assigned to the corresponding client. This trust score is updated on each FL round and its value is a function of both its value in previous rounds and of how much the model differs compared to the models of all other clients. For comparison between models, the Euclidean distance of their weights is used. Writers also provide a variation of their algorithm for decentralized (i.e. without a central entity) Federated Learning. They claim that Trusted FedAvg makes FL effective against attacks as long as less than 50% of clients are malicious, as shown in 4.4. We will delve into the algorithm in next chapter, when we analyze our methodologies.

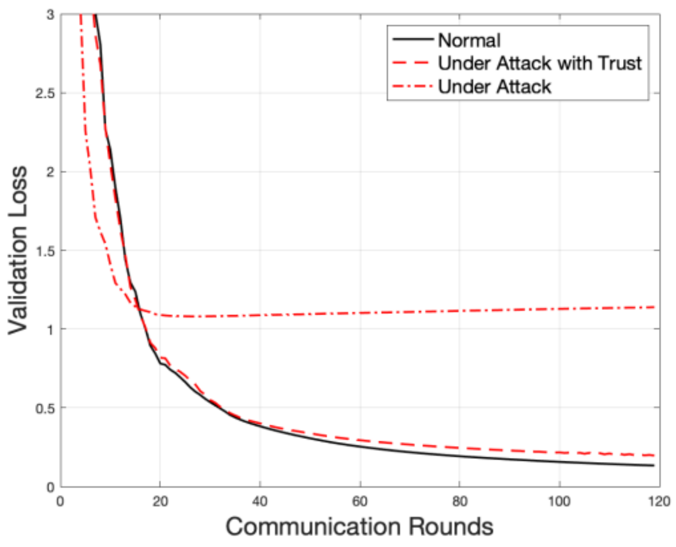

Figure 4.4. *Effect of trust on resilience against attacks. Source: [6]*

Chapter 5

Methodology

5.1 Heterogeneous Federated Learning

We will present the way in which models that do not share the exact same architecture, are aggregated in our work. To do so, we will start by demonstrating a simple example below. Suppose the following two Multilayer Perceptron (MLP) models shown in 5.1. Models need to be of the same neural network, but may have different input features and number of neurons in hidden layers. In this example, both models are 2-layer MLPs, share a common input feature (**Feature 1**), each one has an input feature that the other has not (**Feature 2** and **Feature 3** respectively) and have a different number of neurons in the hidden layer (5 and 4 neurons respectively).

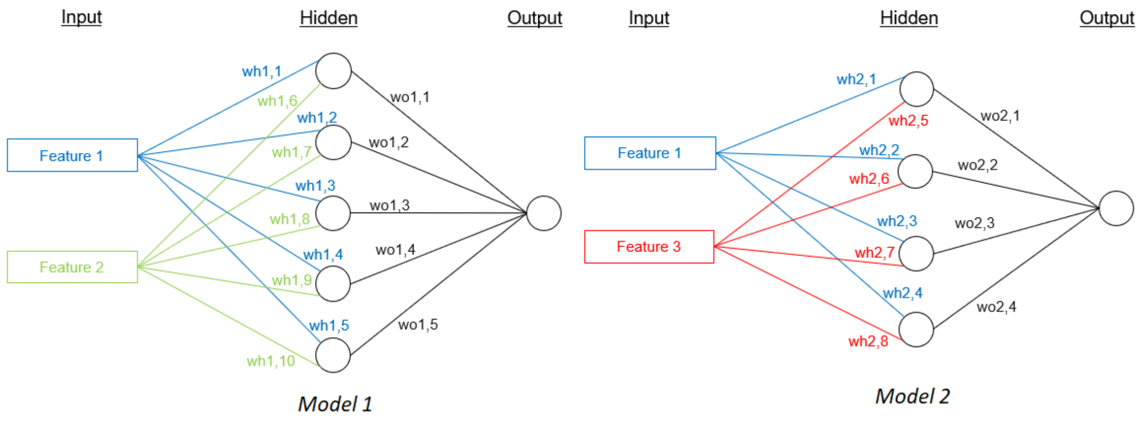

Figure 5.1. *MLP models of two clients to be aggregated.*

First of all, we explain the notation of the weights shown in 5.1. Each weight is named **wxi,j**, where:

- x is either **h** for a hidden layer weight or **o** for an output layer weight.
- i is either **1** for model 1 or **2** for model 2.
- j is the number of weight of x layer.

So, for instance, **wh1,3** is the 3rd weight of hidden layer of model 1, while **wo2,4** is the 4th weight of output layer of model 2. After that, given a weight **w**, we say that a model contains a corresponding weight **w'** to **w**, if **w'** refers to the same feature, same layer and same number of weight in layer as **w**. That said, for instance, given **wh1,3**, model 2 contains the corresponding weight **wh2,3**, since both refer to **Feature 1** and are the 3rd weight of the hidden layer of each model. On the contrary, given **wh1,7**, model 2 does not contain a corresponding weight, since it has no **Feature 2** as input whatsoever. Likewise, given **wo1,5**, model 2 again does not contain a corresponding weight, since it has only 4 output layer weights.

The aggregated model of models 1 and 2 is shown in 5.2. As shown, the set of input features of the aggregated model is the union of the sets of individual models. The number of neurons in each layer is at least as large as the largest number among individual models. Each weight **w** of the aggregated model is calculated as follows:

$$\mathbf{w} = \begin{cases} \textit{average of relevant weights of models which contain } \mathbf{w} \\ 0 \textit{ if no model contains } \mathbf{w} \end{cases} \tag{5.1}$$

Note that, of course, the 6th neuron in hidden layer with all weights set to zero can be omitted and is only useful if another model with 6 hidden layer neurons is expected to take part in the aggregation.

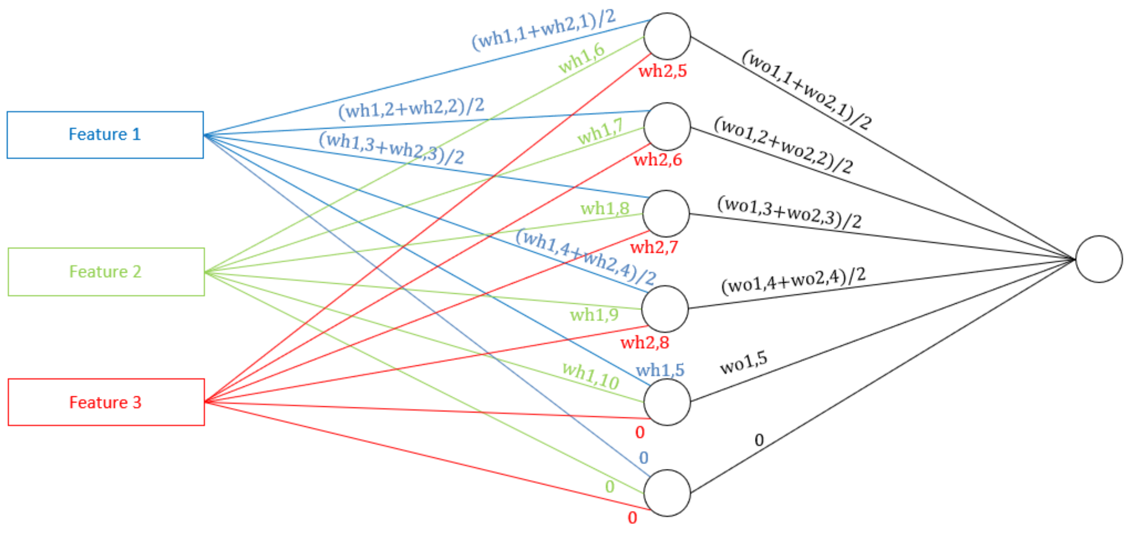

Figure 5.2. *Aggregated model of models 1 and 2.*

The key idea of this process is, on the one hand, that collaborators contribute only to weights which their model contain and, on the other hand, that global model learns from each collaborator exactly what this collaborator is able to teach it.

5.2 Trust-Aware Federated Learning

As we have already mentioned, our goal using trust is to diminish the influence of clients with models whose weights diverge compared to the majority of clients during aggregation process. To do so, we need some trust values to appropriately multiply weights of each client, so that divergent models are assigned smaller such values. In order to calculate trust values, weights of models are directly compared among clients. We present our Trusted FedAvg algorithm in Algorithm 4.1, based on the work of Gholami et al. [6], which we have previously mentioned.

As usual, an FL round begins with clients updating their local models traditionally using gradient descent. Then, the central server has to compute trust values for clients. For this purpose, a *sum* is calculated for each client; the sum of squared 2-norms of the differences of weights with other clients' models, shown in equation 5.2, where N is the set of clients, exponent (k) is the round and w_i is the array of weights of client i. This way, we get a list of such sums. For each client, its *sum* is compared with the median of the list of sums multiplied by a threshold arbitrarily set with its value depending on how much forgiving we choose to be towards a specific divergent client. We have set this threshold to 1.5 in general. If the compared *sum* of a client is less than the median multiplied by the threshold, we consider this client as trustful and set I to 1, while if it is greater, we consider that the client diverges from the majority and set I to 0, as shown in equation 5.3.

$$sum_i^{(k)} = \sum_{j \in N} \frac{||w_j^{(k)} - w_i^{(k)}||_2^2}{|N|} \tag{5.2}$$

$$I_i^{(k)} = \begin{cases} 1 & \text{if } sum_i^{(k)} \leq th_i \times median(\{sum_j^{(k)}\}_{j \in N}) \\ 0 & \text{otherwise} \end{cases} \tag{5.3}$$

ALGORITHM 5.1: ***Trusted FedAvg***. *N is the set of client, D_i is the dataset size of client i, $w_i^{(k)}$ is the array of weights of client i in round k, $t_i^{(k)}$ is the trust value of client i in round k.*

```
Initialize $w_i^{(0)}$ and $t_i^{(0)}$ for each client i
for each round k = 1,2,... do
    Each client i computes $w_i^{(k)}$ from $w_i^{(k-1)}$ using gradient descent and sends it to the server
    Server updates trust values for each client i: $t_i^{(k)} \leftarrow ComputeTrust(i, w_i^{(k)}, t_i^{(k-1)})$
    Server aggregates local updates: $w^{(k)} \leftarrow \sum_{i \in N} \frac{D_i t_i^{(k)}}{\sum_{i \in N} D_i t_i^{(k)}} \frac{w_i^{(k)}}{rareFactor}$
    Server transmits $w^{(k)}$ to all clients
end for

procedure COMPUTETRUST($i, w_i^{(k)}, t_i^{(k-1)}$):
    // Server computes $I_i^{(k)}$ from 5.3:
    for each client $i \in N$ do
        $sum_i \leftarrow 0$
        for each client $j \in N$ do
            $sum_i \leftarrow sum_i + norm^2(w_j^{(k)} - w_i^{(k)})$
        end for
    end for
    if $sum_i \leq th_i \times median(\{sum_i\})$ then
        $I_i^{(k)} \leftarrow 1$
    else
        $I_i^{(k)} \leftarrow 0$
    end if
    // Server computes $t_i^{(k)}$ from 5.6:
    $r_i^{(k)} \leftarrow \rho_1 r_i^{(k-1)} + I_i^{(k)}$
    $s_i^{(k)} \leftarrow \rho_2 s_i^{(k-1)} + 1 - I_i^{(k)}$
    $t_i^{(k)} \leftarrow \frac{r_i^{(k)}+1}{r_i^{(k)}+s_i^{(k)}+2}$
end procedure
```

After that, two parameters, r and s are calculated for each client as stated in [6]. These parameters are a function of I, i.e. whether a client found to be divergent or not, and of their values in the previous round. Of course, we want values of older rounds to be all the less important, so we use the forgetting factors ρ_1 and ρ_2, as shown in equations 5.4 and 5.5. As far as they are concerned, it must be true that $0 < \rho_1 < \rho_2 < 1$; we have set $\rho_1 = 0.2$ and $\rho_2 = 0.8$ troughout our experiments. Using r and s parameters, the central server directly calculates the trust value t_i of each client i from equation 5.6.

$$r_i^{(k)} = \rho_1 r_i^{(k-1)} + I_i^{(k)} \tag{5.4}$$

$$s_i^{(k)} = \rho_2 s_i^{(k-1)} + 1 - I_i^{(k)} \tag{5.5}$$

$$t_i^{(k)} = \frac{r_i^{(k)} + 1}{r_i^{(k)} + s_i^{(k)} + 2} \tag{5.6}$$

Subsequently, before we are ready for the aggregation, we need to take into account the heterogeneity of models. We have already talked about aggregation process of heterogeneous models. The same thing applies here during the calculation of *sum* of each client. So, obviously, between two clients only the norm of relevant weights is taken, that is weights that both clients contain in their models. However, during this procedure another problem arises. Some clients may contain weights that too few, or even no other client contains. If there is a harmful value in such a weight, that is a value that negatively affects performance, the client responsible for this probably will not get an appropriately small trust value due to lack of other clients with which the harmful weight would be compared. Another possibility if one of few clients containing a specific feature has a harmful value, is the increase of *sum* of these few other clients, tending to make them seem divergent.

Accepting that there is no perfect solution, especially for the last case, as a partial solution, we introduce a *rareFactor*, which is an array with the same dimensions as the global model's weight array and its value at each position indicates how many clients contain the respective weight. So, a weight that too few clients contain, is assigned a greater *rareFactor* value, which diminishes the influence of the weight during aggregation. We have set the *rareFactor* for each weight w of the gloabl model as shown in the following equation. We note here that this introduction of *rareFactor* may seem arbitrary, but the fact that a feature is not chosen by most of the clients after feature selection, probably indicates that it is not important in classifying traffic correctly anyway.

$$rareFactor(w) = \begin{cases} 4 & \text{if only one client contains w} \\ 3 & \text{if two clients contain w} \\ 2 & \text{if three clients contain w} \\ 1 & \text{if at least four clients contain w} \end{cases} \tag{5.7}$$

Finally, the aggregation is as shown in equation 5.8, where D_i is the private dataset size of client i. Note, of course, that each client only contributes to weights it contains. So, each term of the following sum is added to the appropriate part of array $w^{(k)}$ of global model's weights .

$$w^{(k)} = \sum_{i \in N} \frac{D_i t_i^{(k)}}{\sum_{i \in N} D_i t_i^{(k)}} \frac{w_i^{(k)}}{rareFactor} \tag{5.8}$$

5.2.1 Limitations

We have to mention, although it is easy to conclude, that our trust-aware method does not constitute panacea to cover every possible harmful value of a weight a client may have. The method is based on using simple factors to appropriately multiply weights. The range of the values of these factors is fixed and limited. There is always the possibility of a harmful weight of big enough value that factors cannot handle. This is also to say that our method cannot mitigate attacks by a potentially malicious client; though for such an attack to be effective against our work, it might be difficult for the perpetrator to go unnoticed either.

Chapter 6

Experiments and Implementation

6.1 Used Data

Our task in general is to train models to correctly classify network traffic as benign and attack. We do so using supervised learning, thus fixed datasets are needed. We used benign traffic from the WIDE backbone [48], an operational testbed network for WIDE Project [49], which carries out research activities through the use of actual network. More information about the data repository exists in [50] and [51]. More specifically, we have used DNS responses from a 10G transit link between WIDE and DIX-IE (an experimental Internet Exchange), henceforth WIDE-G [51], and an 1G transit link between WIDE and an upstream provider, henceforth WIDE-F [51].

For attack traffic, we have used 7 different DNS Booter attacks from [52]. Booters [53] are DDoS services on the Internet that anyone can hire and are still used today by malicious users. A thorough analysis on WIDE-G, WIDE-F and the seven Booter datasets also exists in the work of Dimolianis et al. [54].

As is clear, our experiments concern the DNS protocol (Domain Name System [55]), responsible for helping Internet users and network devices discover websites using human-readable hostnames, instead of numeric IP addresses. The DNS protocol is one of the most popular ones used by attackers because they can easily exploit vulnerabilities in its servers to turn initially small queries into much larger payloads, which are used to bring down the victim's servers. DNS amplification is a type of reflection attack which manipulates publicly-accessible domain name systems, making them flood a target with large quantities of UDP [56] packets. Reflection is achieved by eliciting a response from a DNS resolvers to a spoofed IP address. Using various amplification techniques, perpetrators can "inflate" the size of these UDP packets, making the attack so potent as to bring down even the most robust Internet infrastructure. During a DNS amplification attack, the perpetrator sends out a DNS query with a forged IP address (the victim's) to an open DNS resolver, prompting it to reply back to that address with a DNS response. With numerous fake queries being sent out, and with several DNS resolvers replying back simultaneously, the victim's network can easily be overwhelmed by the sheer number of DNS responses.

6.2 Methodology of Experiments

So, our datasets contain DNS packets, or more precisely, some features of DNS packets' headers. One of these features is the destination AS (Autonomous System) of a packet. To simulate an FL setting, we need distinct clients. We consider every distinct destination AS number that appears in our datasets as a potential client for our experiments. We say "potential", only because our datasets contain hundreds of different AS numbers, while we carried out experiments for 7, 14 and 21 clients. This choice we made is actually realistic, because in a real case scenario what a client -which may well be an AS- would want, would be to classify its receiving traffic -i.e. packets destined to its AS- as benign or malicious.

As mentioned before, our attack traffic dataset contains 7 different attacks. Every time we suppose that each client has been the target of one of these attacks, so that all attacks are distributed to clients. This way, each client has a private dataset consisting of its receiving benign traffic and the attack that it supposedly encountered, which could actually have encountered in a real case scenario.

Now it is a good point to mention that the 7 attacks have different characteristics among each other. For example, 5 of them use type A DNS queries, while the other 2 use type ANY (or type *) [57]. One of these attacks, the 5th one, mostly resembles benign traffic rather than attack one and was found to be ineffective due to not using enough bandwidth. As a result, the client which supposedly encountered this attack, will probably not learn to classify traffic as well as others and, consequently, its model's weights might slightly diverge from other models in a negative way. This is where the trust factor comes into play to diminish the influence of that specific client during aggregation.

In order to have heterogeneous models for our clients, we have each client run a feature selection procedure, using random forest, on its private dataset. After that, clients end up with inherently different feature sets, both in terms of number of features and of what these features are. Of course, feature selection procedure is not something we abusively ran to facilitate our simulation, but rather a very common and desired procedure in Machine Learning that would accelerate local training of clients and would reduce computational resources, especially in our case, where we deal with vast amount of network traffic data. As is obvious, since clients have different data from each other, heterogeneous models is a problem that would almost certainly emerge in a real case scenario.

Then, each client builds its own local MLP (MultiLayer Perceptron) model according to its private datasets. All MLP models are of 2 layers, with n inputs (where n in the number of chosen features for each client), $2n + 1$ neurons in the hidden layer (according to the study of Siaterlis et al. [58]) followed be *ReLu* activation function and 1 output followed by *Sigmoid* activation function. An example of such a model with 2 features as inputs is shown in 6.1.

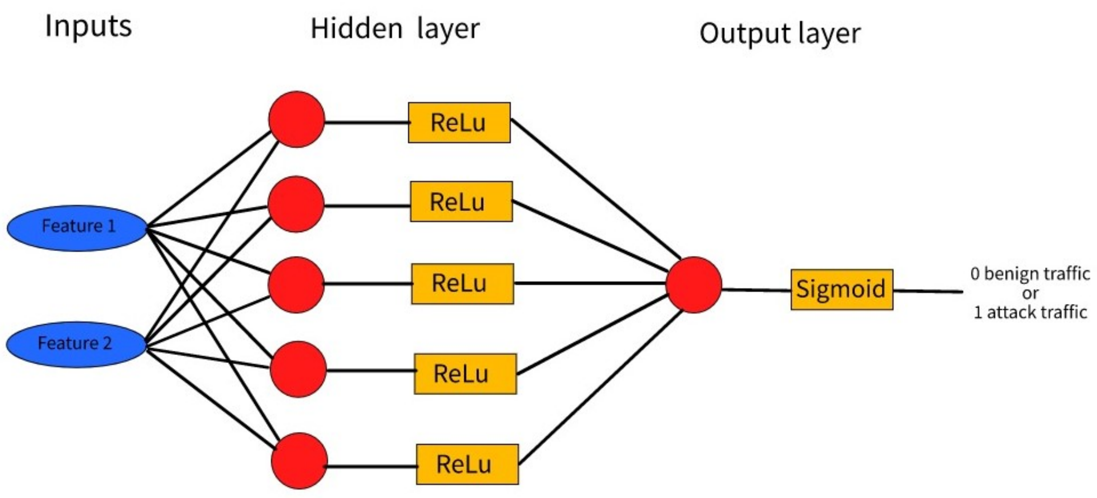

Figure 6.1. *MLP model with 2 input features.*

Our goal, in general, is to compare the performance of clients when they are on their own versus when participating in a heterogeneous Federated Learning setting. More specifically, we conducted the following experiments for the following goals:

- we distributed the 7 attacks to 7, 14 and 21 clients and, firstly, ran classic Machine Learning for each client and then ran our trusted heterogeneous FL for all clients; the goal is to show how collaborative FL process benefits clients compared to each working alone. We ran this experiment 10 times and took the average accuracy of each client and FL model. We also repeated the same experiment 20 times for 7 clients with random distribution of Booter attacks each time; the goal here is to compare the average of each client with the average of FL in these 20 runs.
- we ran heterogeneous Federated Learning again, this time without the trust factor, to demonstrate its positive effect.
- we ran Machine Learning for a client both with and without feature selection process to show how it benefits local training procedure, and thus total FL procedure, in terms of time and computational resources, without compromising performance.

6.3 Technical Details of Implementation

The experimental setup we developed for Federated Learning consists of 21 virtual clients (not all of them used in every experiment) and a central aggregating server, all running on the same machine. The central server is also responsible for testing global model after every epoch. The criterion used to choose the best model among epochs is the sum of accuracy in both attack and benign traffic (sum of true positive and true negative rates), but with greater weight given to accuracy in benign traffic (about 55% compared to 45% for attack traffic). This is because it is more harmful to classify a benign packet as malicious, since it will lead to not answering to a legitimate user, rather than classifying a malicious packet as benign, which will probably not be enough to bring down the service. Calculating the aforementioned accuracy for benign traffic is straightforward using a corresponding general test set. However, for attack traffic we have a test set for each Booter attack and thus we need to take the average of accuracies in these attack traffic test sets, excluding the 5th attack, which mostly resembles benign traffic and, as expected, confuses results.

The aggregation algorithm is previously described in chapter Methodology. After the preprocessing of original datasets, data is split and sent to each client. It is noted here that in a real FL application the data is already stored on local devices. For FL training the following parameters were used:

- batch size set to 128.
- test batch size set to 1000.
- learning rate set to 0.01 in the first epoch and decreased to 0.001 later on, but properly adjusted for some clients if needed.
- number of epochs set to 7 or 10 depending on the experiment.

Code is written in Python (version 3.6). PySyft framework (version 0.2.9) is used to simulate virtual clients and to automate sending of models. Code can be found in thesis' github repository `https://github.com/vpet98/federatedDDoS`.

6.4 Preprocessing

For preprocessing of the original datasets we did the following steps briefly mentioned:

- Deleted columns that do not exist in all datasets.
- Dropped samples with missing values.
- Added target column; 0 for benign and 1 for attack traffic.
- Transformed categorical features to numerical.
- Made a general benign traffic test set.
- Split benign traffic data to 21 clients using destination AS.
- Distributed attack traffic to clients, so that each Booter attack is given to 3 clients.
- Made a test set for each Booter attack.
- Performed feature selection technique using random forest for each client separately. The 16 features that existed before this step were reduced to between 7 and 11 depending on the client.
- Building of appropriate test sets.
- Normalization of datasets.

More thorough analysis on proprocessing can be found in the corresponding notebooks in thesis' github repository `https://github.com/vpet98/federatedDDoS`.

6.5 Results

We present in diagrams and analyze the results of all the experiments we conducted.

6.5.1 Clients On Their Own vs Federated Learning

In this experiment we distributed the 7 attacks to 7, 14 and 21 clients and, firstly, ran classic Machine Learning for each client and then ran our trusted heterogeneous FL for all clients; the goal is to show how collaborative FL process benefits clients compared to each working alone. We ran this experiment 10 times and took the average accuracy of each client and FL model.

For each case of number of clients that participate, i.e. 7, 14 or 21 clients, we have two diagrams. In the first one, we compare the performance of clients each on its own against the performance of the FL model in benign traffic (true negative rate). In the second one, we do the same for attack traffic (true positive rate) taking the average performance in all distinct Booter attacks, excluding the 5th attack, which mostly resembles benign traffic, as we have mentioned earlier. The corresponding diagrams are shown and analyzed below.

6.5.1.1 7 Clients

In 6.2 and 6.3 we see the diagrams when 7 clients participate in the procedure. In benign traffic, FL model outperforms all clients, closely followed by some. In attack traffic, FL model performs about as well as the best of clients. Of course, what matters is looking at both diagrams together to notice that there is no client that performs as well as the FL model in both benign and attack traffic. Using the same criterion as choosing the best model among epochs, normalized as shown in equation 6.1, where *TNR* stands for true negative rate and *TPR* for true positive rate, among all clients, client 2 has the minimum gain of 0.7% adopting the FL model, while client 5 has the maximum gain of 15%.

$$criterion = \frac{1.2 \times TNR + TPR}{2.2} \tag{6.1}$$

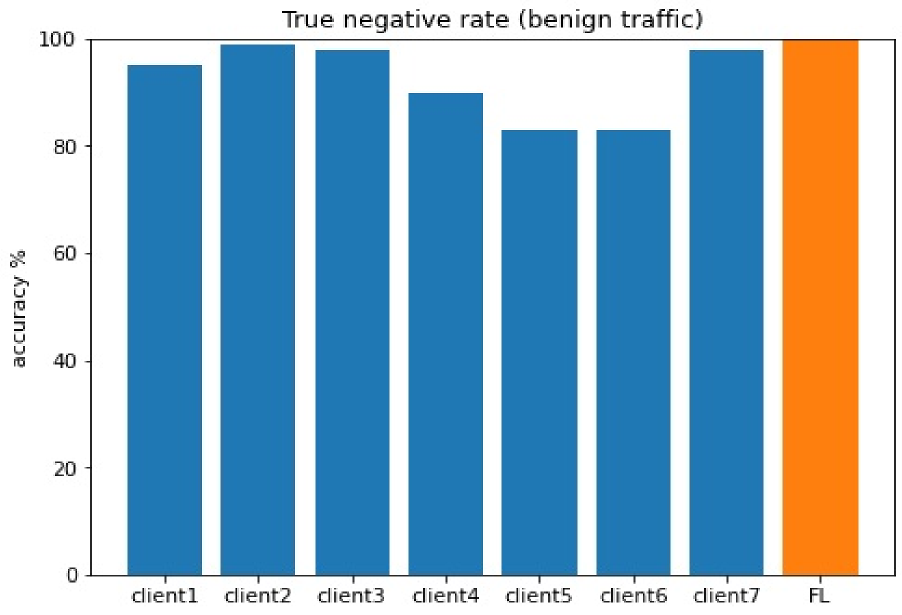

Figure 6.2. *Performance of 7 clients and FL model in benign traffic.*

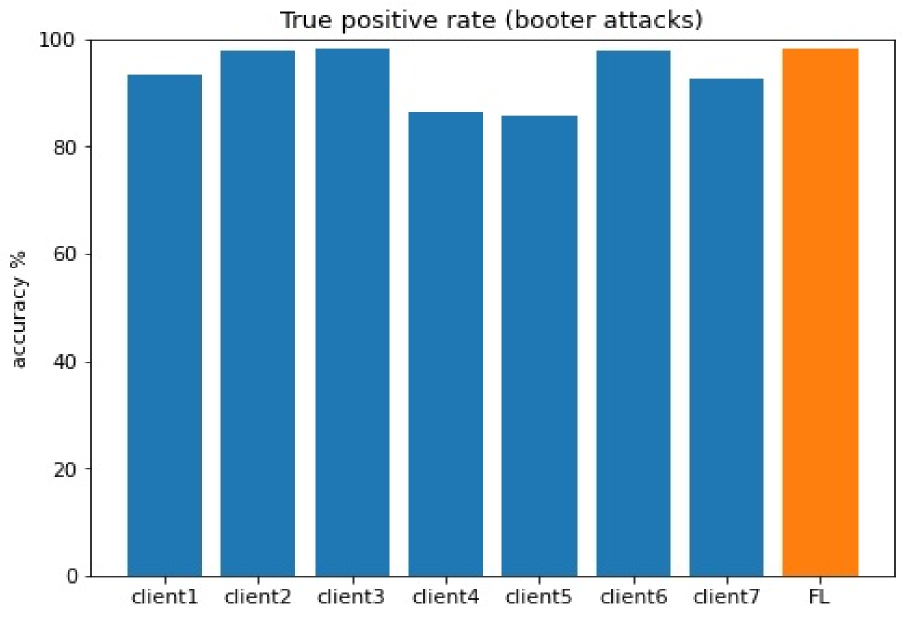

Figure 6.3. *Average performance of 7 clients and FL model in attack traffic, excluding 5th attack.*

It could be argued that the best performing client, client 2, slightly differs from the FL model, meaning it has no actual benefit from the whole procedure. This is expected to happen for some clients in such procedure and it means that their private data are more representative of the union of data of all clients compared to others. More specifically, in our case it could mean that the attack which client 2 supposedly encountered, or also its receiving benign traffic, has more common characteristics with all other attacks that other clients encountered, or also with the benign traffic of our initial datasets. However, no client can know in advance how good are its data compared to other clients and the point is that in the worst case scenario for a client, the process will just not be of much benefit. Even in this case, the benefit is undeniable for most of the clients and it will probably be for all of them if we take into account that network traffic data is constantly updated and the Federated Learning process is to be repeated every so often among collaborating clients.

6.5.1.2 14 Clients

In 6.4 and 6.5 we see the diagrams when 14 clients participate in the procedure. Again, there are some clients close, or maybe even slightly better than the FL model, but there is still clear motivation for some entity to be a member of the whole procedure. Using equation 6.1 as before, among all clients, clients 2 and 9 have the minimum gain (actually loss) of -0.23% adopting the FL model, while client 12 has the maximum gain of 21.4%.

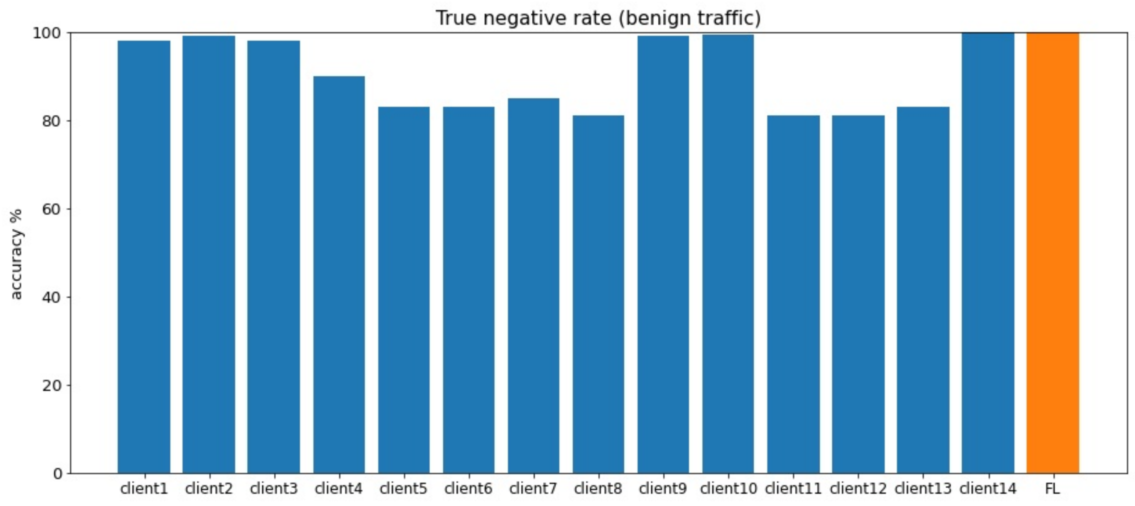

Figure 6.4. *Performance of 14 clients and FL model in benign traffic.*

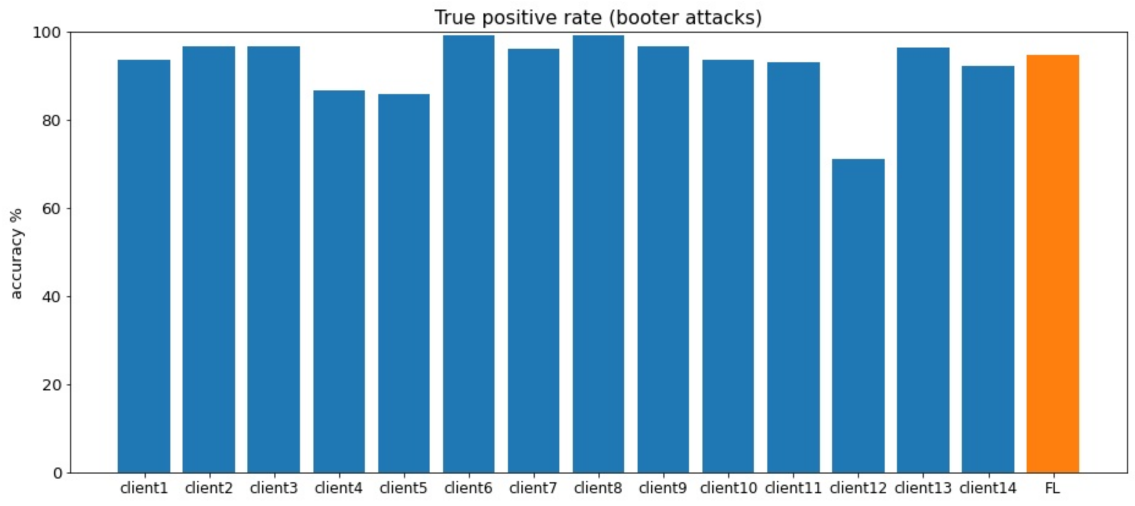

Figure 6.5. *Average performance of 14 clients and FL model in attack traffic, excluding 5th attack.*

6.5.1.3 21 Clients

In 6.6 and 6.7 we see the diagrams when 21 clients participate in the procedure. We make the same observations as before. Using equation 6.1 as before, among all clients, client 9 has the minimum gain (actually loss) of -0.82% adopting the FL model, while client 12 has the maximum gain of 14.4%.

Furthermore, we can comment the scalability of our work. Going from 7 clients to 14 and then to 21, we notice similar patterns in the diagrams, something that promises efficiency in a real case scenario with probably more than a hundred collaborators.

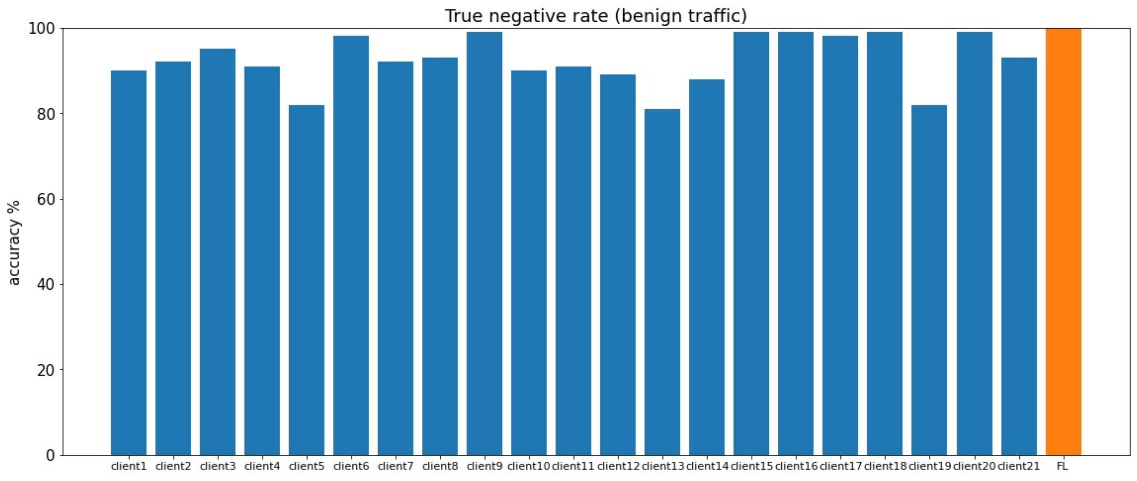

Figure 6.6. *Performance of 21 clients and FL model in benign traffic.*

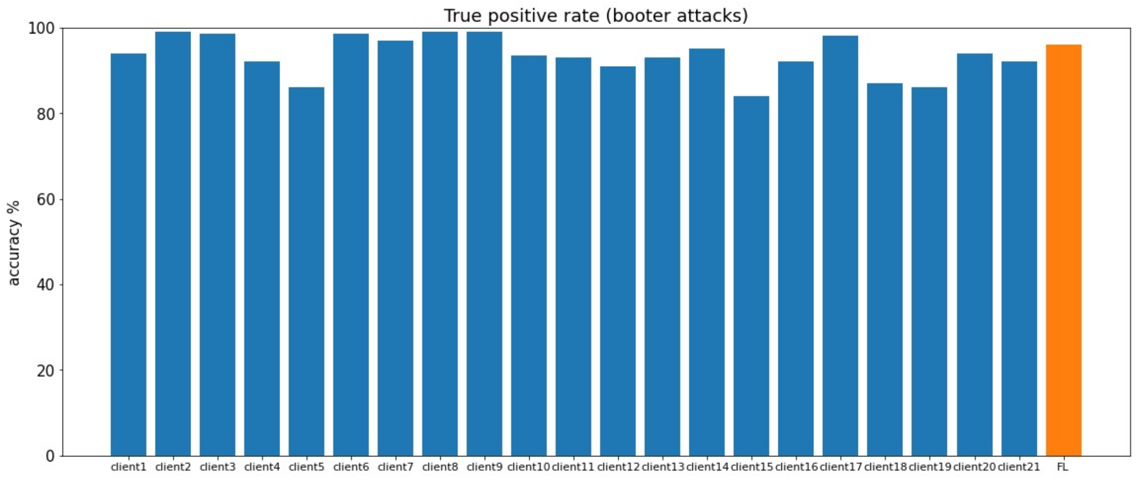

Figure 6.7. *Average performance of 21 clients and FL model in attack traffic, excluding 5th attack.*

6.5.1.4 Random Distribution of Attacks

As we mentioned earlier, we also ran trusted heterogeneous FL 20 times for 7 clients with random distribution of Booter attacks each time; the goal is to compare the average of each client with the average of FL in these 20 runs.

In 6.8 and 6.9 we see the results for benign and attack traffic respectively. The 5th Booter attack is again excluded when calculating performance in attack traffic. Firstly, this experiment shows that as all attacks have been shared equally by all clients and still there is divergence in performance among them, it can be inferred that characteristics of private benign traffic also differ from client to client.

In benign traffic, FL model performs clearly better in average than any other average of client, while it equals the top performing clients in attack traffic. Using equation 6.1 as before, among all clients, client 6 has the minimum gain of 3.8% adopting the FL model, while client 5 has the maximum gain of 10.2%. This experiment highlights even more the benefit of a client joining the procedure, since it demonstrates that FL model performs better in average case data, something which a client needs to suppose it owns before it takes part.

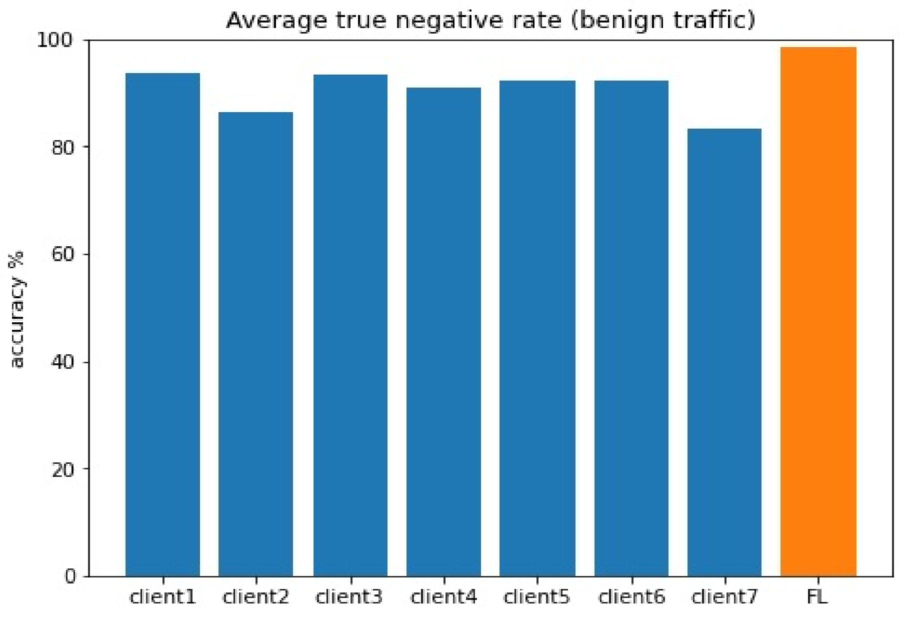

Figure 6.8. *Average performance of 7 clients and FL model in benign traffic in 20 runs with random distribution of attacks.*

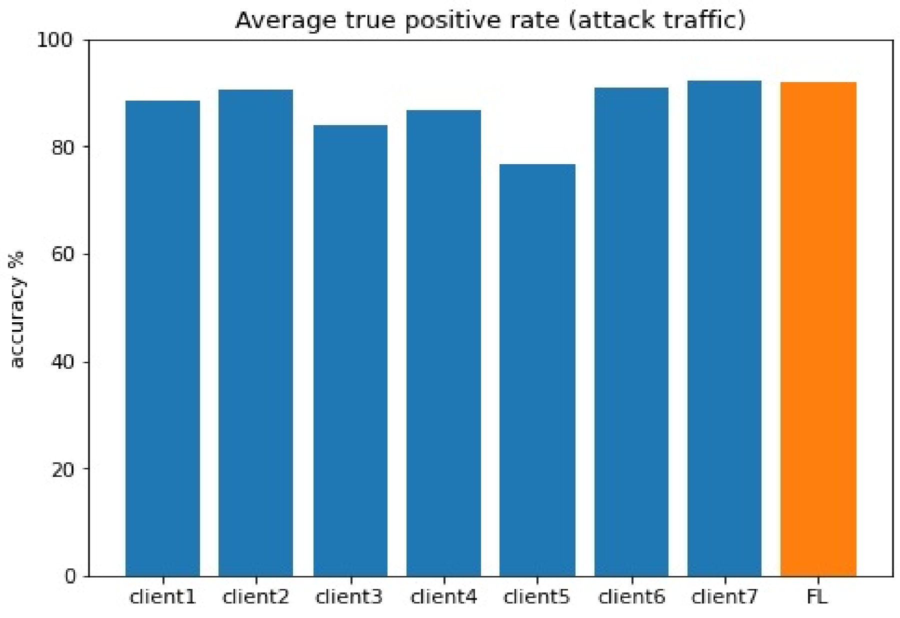

Figure 6.9. *Average performance of 7 clients and FL model in attack traffic (average of attacks excluding the 5th one) in 20 runs with random distribution of attacks.*

6.5.2 Trust Effect

In this experiment we ran heterogeneous Federated Learning again, this time without the trust factor, to demonstrate its positive effect. The 5th Booter attack mostly resembles benign traffic and thus clients that encountered it have not learnt well enough to classify traffic correctly. As a result, their models' weights tend to diverge from others. So, these are our "mistrustful" clients. To emphasize the results of this experiment, we have used appropriate coefficients for these clients during aggregation to increase their influence. It is important to note here that these coefficients do not make the experiment unrealistic, because it simply equals to these clients having bigger private datasets, as can be inferred from equation 5.8 we analyzed in chapter Methodology.

For each case of number of clients that participate, i.e. 7, 14 or 21 clients, we have one diagram the first two columns of which compare FL with and without trust in attack traffic and the other two columns do the same for benign traffic.

6.5.2.1 7 Clients

In 6.10 we see the diagram when 7 clients participate in the procedure. The weights of client that supposedly encountered the 5th attack are multiplied by 2 during aggregation, like if it had a dataset double the size of others or like if there were two clients that encountered only this specific attack.

As we mentioned earlier, the criterion we use to choose the best performance among epochs is the sum of accuracy in benign and attack traffic. So, it only matters to look at both columns, i.e. attack and benign, at the same time. Doing so in the corresponding diagram, the benefit of trust factor is unquestionable. More to the significant divergence in the sum of accuracies, we have also stated earlier that correctly classifying benign traffic is more important than attack traffic. More precisely, using equation 6.1 as before, we calculate that trust factor is responsible for 8.5% increase in performance.

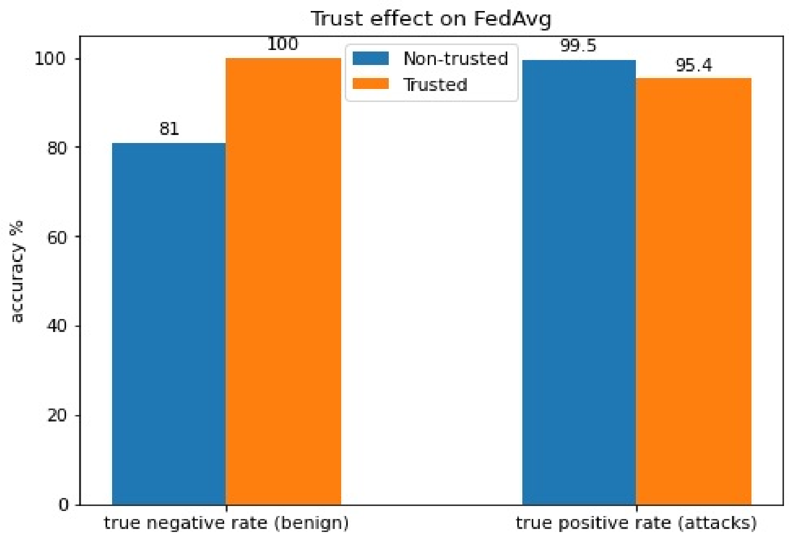

Figure 6.10. *FL without and with trust factor for 7 clients.*

6.5.2.2 14 clients

In 6.11 we see the diagram when 14 clients participate in the procedure. The weights of one of the two clients that supposedly encountered the 5th attack are multiplied by 4 during aggregation. We make the same observations as before. More precisely, using equation 6.1 as before, we calculate that trust factor is responsible for 7.6% increase in performance.

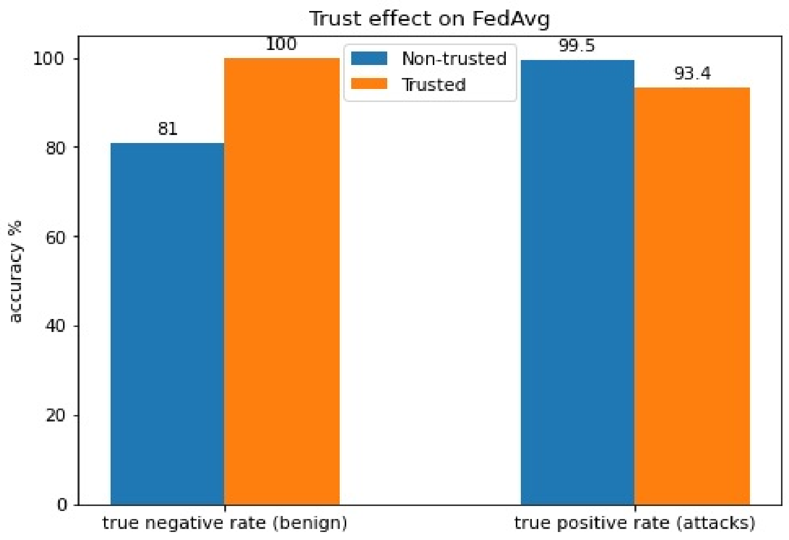

Figure 6.11. *FL without and with trust factor for 14 clients.*

6.5.2.3 21 clients

In 6.12 we see the diagram when 21 clients participate in the procedure. The weights of all the three clients that supposedly encountered the 5th attack are multiplied by 2 during aggregation. Using equation 6.1 as before, we calculate that trust factor is responsible for 8.5% increase in performance. In addition to making the same comments as before, we conclude the scalability of our work concerning trust-aware Federated Learning.

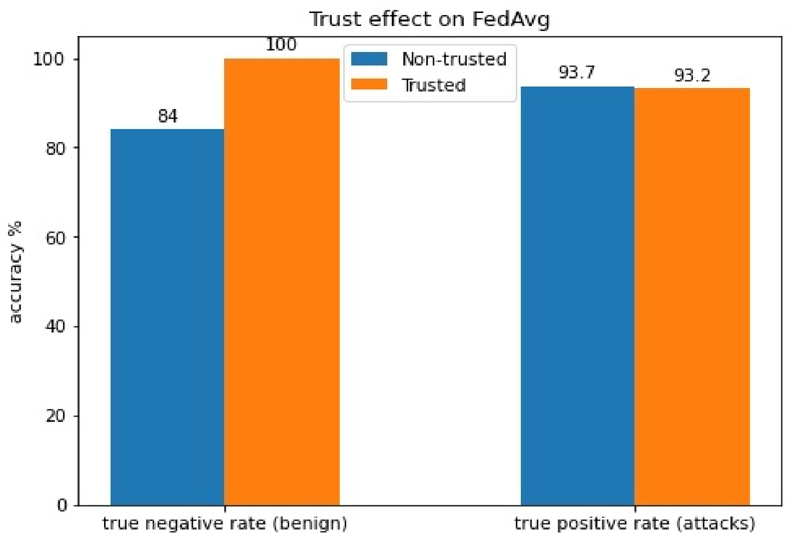

Figure 6.12. *FL without and with trust factor for 21 clients.*

6.5.3 Feature Selection Benefit

As we mentioned earlier, in this experiment we ran Machine Learning for a client both with and without feature selection process to show how it benefits local training procedure, and thus total

FL procedure, in terms of time and computational resources, without compromising performance.

In 10 epochs of training for a client with 8 features after feature selection process compared to 16 features initially, the results are shown in table 6.1. The client gained:

- 15% less training time.
- 10% less RAM memory usage.
- 40% less required disk space for private dataset storage.

Resource	**Without Feature Selection**	**With Feature Selection**	**Gain**
Training Time	10 min	8.5 min	15%
RAM	510 MB	460 MB	10%
Disk Storage Space	3.6 MB	2.2 MB	40%

Table 6.1. *Benefit of feature selection procedure.*

In both cases, i.e. with and without feature selection, CPU usage was at 100% and there is no observable difference in performance. The experiment shows indisputable benefit of feature selection procedure even in our small scale simulation, in terms of data, let alone in Big Data scale, as network traffic data of an organization are of. The benefit of feature selection might be more locally targeted for each client, but it is obvious that by accelerating local training time of clients, the whole FL process is also accelerated. Finally, the experiment also proves that we did not just employed this technique to facilitate our simulations, but we rather managed to combine a process desired by every client with Federated Learning.

Chapter 7

Conclusions

7.1 Summary

In this diploma thesis, we used Federated Learning (FL) for collaborative detection of Distributed Denial of Service (DDoS) attacks. As a novelty, we extended the basic scheme of FL to enable training with heterogeneous models, which do not share the exact same network architecture. In addition, based on existing work, we enriched the classic Federated Averaging aggregation algorithm with a trust factor and combined it together with our heterogeneous FL.

At first, we delved into required theoretical background knowledge, mainly about Federated Learning. Then, we briefly presented existing related works on DDoS detection using FL. Based on some of these works, we developed our trust-aware heterogeneous FL algorithm. Our method promises to relieve the procedure from two important obstacles. The first one being the need of multi-domain collaborators to agree all together on some common model architecture, which will probably not represent the private data or computational resources needs of some of them. The second one, is the possibility of one of the collaborators sending back to the central entity models that negatively affect performance of the global model; this may always happen due to "bad" or "faulty" data of some collaborator and is not easy either to detect or mitigate.

We solved the first problem by aggregating only matching weights among heterogeneous collaborators' models. For the second problem, we added a factor to appropriately multiply weights during aggregation, in order to diminish the influence of individually collaborators sending "divergent" models compared to the majority. We then effectively combined our two solutions to form a single aggregation method.

For the experiments we conducted we used real network traffic data; benign traffic from the WIDE project and as attack traffic seven different Booters utilizing DNS protocol. We built 21 virtual clients and split data to them, so that each represents an Autonomous System (AS), which has supposedly encountered one of the attacks. After that, we ran a feature selection procedure employing random forest for each of our clients separately on its private dataset, from which heterogeneous models emerged.

We segregated our experiments in three categories: comparing performance of clients on their own vs collaborative FL, examine trust effect when there are clients who send divergent models and showing the unquestionable -and already known- benefit of feature selection procedure, which can now be combined with Federated Learning due to our heterogeneous method.

From the results, we conclude that our method can be effective and scalable in a real case scenario as it benefits clients in total. There always may exist some clients that will not gain much in a specific FL procedure, however, some other clients may increase their performance even up to 20%, as we saw earlier in our experiments. Finally, we deduce that trust plays a significant role in cases where some clients send divergent models, as it can increase the performance in up to 8.5%.

7.2 Future Work

Through the end of this thesis, we propose the following directions to which the system we developed could to be improved and extended in the future:

- Different approach of trust factor in order to make the mechanism capable to maintain its performance against any divergent model sent, even by malicious clients.
- Extend the process to work without the need of a central entity, that is decentralized Federated Learning. In that case, model is sent directly from client to client keeping what is has already learnt each time. Authenticity of local models and reliability of such a learning environment could ideally be ensured by employing blockchain [59] technology, another state-of-the-art topic in Federated Learning with recent works like [60] and [61].
- Consider classification mechanisms that can jointly recognize various attack vectors or can perform other useful network traffic analysis via multi-task learning techniques as in [4].

Bibliography

[1] *Introduction to IBM Federated Learning: A Collaborative Approach to Train ML Models on Private Data*. https://towardsdatascience.com/introduction-to-ibm-federated-learning-a-collaborative-approach-to-train-ml-models-on-private-data-2b4221c3839. Access date: 2022-05-05.

[2] *What is an autonomous system? | What are ASNs?* https://www.cloudflare.com/learning/network-layer/what-is-an-autonomous-system/. Access date: 2022-05-06.

[3] M. Dimolianis. *Intelligent Services for Detection and Mitigation of Distributed Denial-of-Service Attacks in Programmable Network Environments*. Doctoral Dissertation, NTUA, 2022. http://artemis.cslab.ece.ntua.gr:8080/jspui/handle/123456789/18312.

[4] Y. Zhao, J. Chen, D. Wu, J. Teng and S. Yu. *Multi-Task Network Anomaly Detection using Federated Learning*. *In Proceedings of the Tenth International Symposium on Information and Communication Technology*, 2019. https://dl.acm.org/doi/abs/10.1145/3368926.3369705.

[5] E. Diao, J. Ding and V. Tarokh. *HeteroFL: Computation and Communication Efficient Federated Learning for Heterogeneous Clients*. *In Proceedings of the International Conference on Learning Representations (ICLR)*, 2021. https://arxiv.org/abs/2010.01264.

[6] A. Gholami, N. Torkzaban and J. S. Baras. *On the Importance of Trust in Next-Generation Networked CPS Systems: An AI Perspective*. *arXiv*, 2021. https://arxiv.org/abs/2104.07853.

[7] *Autonomous system (Internet)*. https://en.wikipedia.org/wiki/Autonomous_system_(Internet). Access date: 2022-05-06.

[8] *What is a denial-of-service attack?* https://www.cloudflare.com/learning/ddos/glossary/denial-of-service/. Access date: 2022-05-03.

[9] *DDoS Attack Trends for Q1 2022*. https://radar.cloudflare.com/notebooks/ddos-2022-q1. Access date: 2022-05-03.

[10] *DDoS attacks in Q1 2022*. https://securelist.com/ddos-attacks-in-q1-2022/106358/. Access date: 2022-05-03.

[11] *GLOBAL DDOS SUMMARY - APRIL 2022*. https://horizon.netscout.com/?atlas=summary. Access date: 2022-05-03.

[12] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh and D. Bacon. *Federated Learning: Strategies for Improving Communication Efficiency*. *arXiv*, 2017. https://arxiv.org/abs/1610.05492.

[13] H. B. McMahan, E. Moore, D. Ramage, S. Hampson and B. Agüera y Arcas. *Communication-Efficient Learning of Deep Networks from Decentralized Data*. *In Proceedings of the 20 th International Conference on Artificial Intelligence and Statistics*, 2016. https://arxiv.org/abs/1602.05629.

[14] Y. Zhao, J. Chen, Q. Guo, J. Teng and D. Wu. *Network Anomaly Detection Using Federated Learning and Transfer Learning. In Proceedings of the International Conference on Security and Privacy in Digital Economy (SPDE)*, 2020. https://link.springer.com/chapter/10.1007/978-981-15-9129-7_16.

[15] Q. Tian, C. Guang, C. Wenchao and W. Si. *A Lightweight Residual Networks Framework for DDoS Attack Classification Based on Federated Learning. In Proceedings of the IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2021. https://ieeexplore.ieee.org/document/9484622.

[16] *Internet protocol suite*. https://en.wikipedia.org/wiki/Internet_protocol_suite. Access date: 2022-05-06.

[17] *Internet*. https://en.wikipedia.org/wiki/Internet. Access date: 2022-05-06.

[18] *What is the Internet of Things (IoT)?* https://www.ibm.com/blogs/internet-of-things/what-is-the-iot/. Access date: 2022-05-03.

[19] *What is a DDoS attack?* https://www.cloudflare.com/en-gb/learning/ddos/what-is-a-ddos-attack/. Access date: 2022-05-03.

[20] *What is a SYN flood attack?* https://www.cloudflare.com/en-gb/learning/ddos/syn-flood-ddos-attack/. Access date: 2022-05-03.

[21] *What is a DNS amplification attack?* https://www.cloudflare.com/en-gb/learning/ddos/dns-amplification-ddos-attack/. Access date: 2022-05-03.

[22] J. Mirkovic and P. Reiher. *A taxonomy of DDoS attack and DDoS defense mechanisms. ACM SIGCOMM Computer Communication Review*, 34(2):39–53, 2004. https://doi.org/10.1145/997150.997156.

[23] *Big data*. https://en.wikipedia.org/wiki/Big_data. Access date: 2022-05-04.

[24] *What is GDPR, the EU's new data protection law?* https://gdpr.eu/what-is-gdpr/. Access date: 2022-05-04.

[25] T. Yang, G. Andrew, H. Eichner, H. Sun, W. Li, N. Kong, D. Ramage and F. Beaufays. *Applied Federated Learning: Improving Google Keyboard Query Suggestions. arXiv*, 2018. https://arxiv.org/abs/1812.02903.

[26] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis et al. *Advances and Open Problems in Federated Learning. arXiv*, 2019. https://arxiv.org/abs/1912.04977.

[27] *Decentralized federated learning*. https://en.wikipedia.org/wiki/Federated_learning#Decentralized_federated_learning. Access date: 2022-05-04.

[28] C. Che, X. Li, C. Chen, X. He and Z. Zheng. *A Decentralized Federated Learning Framework via Committee Mechanism with Convergence Guarantee. arXiv*, 2021. https://arxiv.org/abs/2108.00365.

[29] W. Liu, L. Chen and W. Zhang. *Decentralized Federated Learning: Balancing Communication and Computing Costs. arXiv*, 2022. https://arxiv.org/abs/2107.12048.

[30] K. Hsieh, A. Phanishayee, O. Mutlu and P. B. Gibbons. *The Non-IID Data Quagmire of Decentralized Machine Learning. In Proceedings of the International Conference on Machine Learning (ICML)*, 2020. https://arxiv.org/abs/1910.00189.

[31] X. Li, K. Huang, W. Yang, S. Wang and Z. Zhang. *On the Convergence of FedAvg on Non-IID Data*. *In Proceedings of the International Conference on Learning Representations*, 2020. https://arxiv.org/abs/1907.02189.

[32] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar and V. Smith. *Federated Optimization in Heterogeneous Networks*. *In Proceedings of the 3rd MLSys Conference*, 2019. https://arxiv.org/abs/1812.06127.

[33] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos and Y. Khazaeni. *Federated Learning with Matched Averaging*. *In Proceedings of the 38th Annual IEEE International Conference on Computer Communications (INFOCOM)*, 2019. https://arxiv.org/abs/2002.06440.

[34] S. Ek, F. Portet, P. Lalanda and G. Vega. *A Federated Learning Aggregation Algorithm for Pervasive Computing: Evaluation and Comparison*. *In Proceedings of the 9th IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2021. https://arxiv.org/abs/2110.10223.

[35] M. G. Arivazhagan, V. Aggarwal, A. K. Singh and S. Choudhary. *Federated Learning with Personalization Layers*. *arXiv*, 2019. https://arxiv.org/abs/1912.00818.

[36] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang and H. Qi. *Beyond Inferring Class Representatives: User-Level Privacy Leakage From Federated Learning*. *In Proceedings of the 38th Annual IEEE International Conference on Computer Communications (INFOCOM)*, 2018. https://arxiv.org/abs/1812.00535.

[37] R. Shokri and V. Shmatikov. *Privacy-preserving deep learning*. *In Proceedings of the 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2015. https://ieeexplore.ieee.org/document/7447103.

[38] R. C. Geyer, T. Klein and M. Nabi. *Differentially Private Federated Learning: A Client Level Perspective*. *In Proceedings of the NIPS 2017 Workshop: Machine Learning on the Phone and other Consumer Devices*, 2017. https://arxiv.org/abs/1712.07557.

[39] A. N. Bhagoji, S. Chakraborty, P. Mittal and S. Calo. *Analyzing Federated Learning through an Adversarial Lens*. *In Proceedings of the 36th International Conference on Machine Learning*, 2018. https://arxiv.org/abs/1811.12470.

[40] S. Shen, S. Tople and P. Saxena. *Auror: defending against poisoning attacks in collaborative deep learning systems*. *In Proceedings of the 32nd Annual Conference on Computer Security Applications*, 2016. https://dl.acm.org/doi/10.1145/2991079.2991125.

[41] Y. Chen, F. Luo, T. Li, T. Xiang, Z. Liu and J. Li. *A training-integrity privacy-preserving federated learning scheme with trusted execution environment*. *Information Sciences*, 522:69–79, 2020. https://doi.org/10.1016/j.ins.2020.02.037.

[42] J. Li, L. Lyu, X. Liu, X. Zhang and X. Lyu. *FLEAM: A Federated Learning Empowered Architecture to Mitigate DDoS in Industrial IoT*. *IEEE Transactions on Industrial Informatics*, 18(6):4059–4068, 2021. https://ieeexplore.ieee.org/document/9454328.

[43] Q. Qin, K. Poularakis, K. K. Leung and L. Tassiulas. *Line-Speed and Scalable Intrusion Detection at the Network Edge via Federated Learning*. *In Proceedings of the IFIP Networking Conference*, 2020. https://ieeexplore.ieee.org/abstract/document/9142704.

[44] S. Agrawal, S. Sarkar, O. Aouedi, G. Yenduri, K. Piamrat, S. Bhattacharya, P. K. R. Maddikunta and T. R. Gadekallu. *Federated Learning for Intrusion Detection System: Concepts, Challenges and Future Directions*. *Journal of Network and Computer Applications (JNCA)*, 2021. https://arxiv.org/abs/2106.09527.

[45] X. Cao, M. Fang, J. Liu and N. Z. Gong. *FLTrust: Byzantine-robust Federated Learning via Trust Bootstrapping*. *In Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2021. https://arxiv.org/abs/2012.13995.

[46] J. Zhao, X. Zhu, J. Wang and J. Xiao. *Efficient Client Contribution Evaluation for Horizontal Federated Learning*. *In Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2021. https://arxiv.org/abs/2102.13314.

[47] S. K. Shyn, D. Kim and K. Kim. *FedCCEA : A Practical Approach of Client Contribution Evaluation for Federated Learning*. *arXiv*, 2021. https://arxiv.org/abs/2106.02310.

[48] *WIDE Backbone*. http://two.wide.ad.jp/. Access date: 2022-05-24.

[49] *WIDE Project*. https://www.wide.ad.jp/. Access date: 2022-05-24.

[50] *MAWI Working Group Traffic Archive*. http://mawi.wide.ad.jp/mawi/. Access date: 2022-05-24.

[51] K. Cho, K. Mitsuya and A. Kato. *Traffic Data Repository at the WIDE Project*. *In Proceedings of the USENIX Annual Technical Conference*, 2000. https://www.usenix.org/conference/2000-usenix-annual-technical-conference/traffic-data-repository-wide-project.

[52] J. J. Santanna, R. van Rijswijk-Deij, R. Hofstede, A. Sperotto, M. Wierbosch, L. Z. Granville and A. Pras. *Booters — An analysis of DDoS-as-a-service attacks*. *In Proceedings of the International Symposium on Integrated Network Management*, 2015. https://ieeexplore.ieee.org/document/7140298.

[53] *Booters, Stressers And DDoSers*. https://www.wallarm.com/what/booters-stressers-and-ddosers. Access date: 2022-05-24.

[54] M. Dimolianis, A. Pavlidis and V. Maglaris. *Signature-Based Traffic Classification and Mitigation for DDoS Attacks Using Programmable Network Data Planes*. *IEEE Access*, 9:113061–113076, 2021. https://ieeexplore.ieee.org/abstract/document/9511420.

[55] *Domain Name System*. https://en.wikipedia.org/wiki/Domain_Name_System. Access date: 2022-05-24.

[56] *User Datagram Protocol*. https://en.wikipedia.org/wiki/User_Datagram_Protocol. Access date: 2022-05-24.

[57] *List of DNS record types*. https://en.wikipedia.org/wiki/List_of_DNS_record_types. Access date: 2022-06-02.

[58] C. Siaterlis and B. Maglaris. *Detecting DDoS attacks using a multilayer Perceptron classifier*. *In Proceedings of the 9th IFIP/IEEE International Symposium on Integrated Network Management*, 2004. https://www.researchgate.net/publication/235891315_Detecting_DDoS_attacks_using_a_multilayer_Perceptron_classifier.

[59] *Blockchai*. https://en.wikipedia.org/wiki/Blockchain. Access date: 2022-06-10.

[60] Z. Wang and Q. Hu. *Blockchain-based Federated Learning: A Comprehensive Survey*. 2021. https://arxiv.org/abs/2110.02182.

[61] C. Ma, J. Li, M. Ding, L. Shi, T. Wang, Z. Han and H. V. Poor. *When Federated Learning Meets Blockchain: A New Distributed Learning Paradigm*. 2020. https://arxiv.org/abs/2009.09338.