ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

M162 – PRIVACY TECHNOLOGIES

Fall 2019-2020
Project #1

Crowds Protocol
Simulation

FEBRUARY 29, 2020

COMPUTER SCIENCE: DATA, INFORMATION AND KNOWLEDGE MANAGEMENT
ΓΙΑΝΝΑΚΗΣ ΣΤΑΥΡΟΣ - CS2180006

Περιεχόμενα

1. Εισαγωγή

Η παρούσα εργασία υλοποιήθηκε στα πλαίσια του μαθήματος “Τεχνικές Ιδιωτικότητας“ του μεταπτυχιακού προγράμματος του τμήματος Πληροφορικής και Τηλεπικοινωνιών του Εθνικού και Καποδιστριακού Πανεπιστημίου Αθηνών.

Αφορά την υλοποίηση προσομοίωσης του πρωτόκολλου ανώνυμης επικοινωνίας “Crowds” και την ανάλυση των αποτελεσμάτων της προσομοίωσης αυτής μέσω του εργαλείου F-BLEAU.

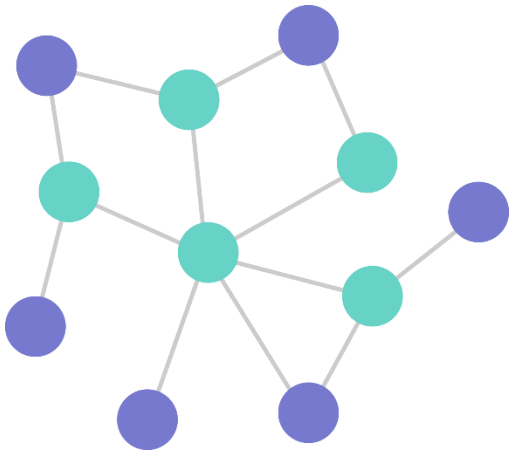
Ακολουθεί μια μικρή, εισαγωγική περιγραφή του πρωτοκόλλου Crowds, έπειτα μια ανάλυση της υλοποίησης και τέλος συγκρίσεις και σχολιασμός των αποτελεσμάτων.

2. Πρωτόκολλο Crowds

2.1. Περιγραφή

Το πρωτόκολλο Crowds αφορά την ανώνυμη επικοινωνία μεταξύ χρηστών σε ένα δίκτυο και προτείνεται κυρίως για anonymous web browsing. Η βασική ιδέα του πρωτοκόλλου είναι πως η ταυτότητα του αρχικού αποστολέα ενός μηνύματος χάνεται μέσα στο δίκτυο, διότι ο αρχικός αποστολέας προωθεί το μήνυμα σε έναν γειτονικό του κόμβο η διαδικασία επαναλαμβάνεται αρκετές φορές, μέχρις ότου παραδοθεί το μήνυμα στον Web Server. Με αυτόν τον τρόπο, οποιοσδήποτε αντίπαλος προσπαθεί να βρει τον αρχικό αποστολέα του πακέτου, δυσκολεύεται λόγω των πολλών routing και του γεγονότος πως το πακέτο που λαμβάνει έρχεται από κάποιον που δεν ξέρουμε εάν είναι ο αρχικός αποστολέας. Θεωρούμε πως ο αντίπαλος έχει “διαφθείρει” διάφορους κόμβους του δικτύου (που ονομάζονται “corrupted”) και χειρίζεται και τον τελικό Web Server που παραλαμβάνει το μήνυμα.

Ειδικότερα, με την εκκίνηση του πρωτοκόλλου ο κάθε χρήστης έχει μια πιθανότητα φ να κάνει προώθηση το μήνυμα σε κάποιον γείτονά του (ή και στον εαυτό του). Η πιθανότητα αυτή ονομάζεται Forwarding Probability. Σε περίπτωση που δεν γίνει forward, το μήνυμα παραδίδεται στον web server από τον τελευταίο χρήστη.



2.2. Περίπτωση εντοπισμού

Σε περίπτωση που κάποιος κακόβουλος χρήστης λάβει το μήνυμα, λέμε πως «εντοπίζει» τον αποστολέα. Δεν γνωρίζει ωστόσο εάν ο αποστολέας αυτός είναι ο αρχικός αποστολέας του πακέτου. Εάν συμβεί αυτό, το μονοπάτι που έχει ακολουθηθεί μέχρι τον κακόβουλο χρήστη «καταστρέφεται» και πρέπει να ξαναγίνει αποστολή με μια από τις δύο παρακάτω στρατηγικές διόρθωσης:

- Initiator: Η αποστολή ξαναγίνεται από την αρχή
- Last-Honest: Η αποστολή συνεχίζεται από τον τελευταίο τίμιο χρήστη.

3. Περιγραφή υλοποίησης

3.1. Γενικές πληροφορίες

Η υλοποίηση της εργασίας simulation του πρωτοκόλλου Crowds έγινε σε γλώσσα προγραμματισμού Python 3.8.1 64bit. Χρειάστηκε η χρήση της τελευταίας έκδοσης Python, διότι το F-BLEAU γίνεται εγκατάσταση μόνο σε αυτή την έκδοση μέσω pip. Η υλοποίηση έγινε σε περιβάλλον Windows 10 64bit αλλά η εκτέλεση της δοκιμάστηκε και σε λειτουργικό σύστημα Linux Ubuntu 18.04.

Η εκτέλεση γίνεται με την εντολή που δίνεται στην εκφώνηση, συγκεκριμένα:

Python simulate.py <graph-file> <corrupted-file> <users-file> <brokenpaths> <strategy>

Όπου, σε περίπτωση broken paths = 0 δεν χρειάζεται να συμπληρωθεί τύπος στρατηγικής επιδιόρθωσης, μιας και το πρωτόκολλο συνεχίζει κανονικά την εκτέλεσή του.

Τα αρχεία εισόδου του γράφου, των corrupted χρηστών και των αποστολέων των μηνυμάτων μπορούν να είναι είτε τύπου .txt είτε .csv. Το development έγινε με την χρήση .txt, αλλά δοκιμάστηκε και επιτυχώς η εκτέλεση με .csv αρχεία.

3.2. Επεξήγηση

Για την απλοποίηση του κώδικα, έχουν δημιουργηθεί κάποιες βοηθητικές συναρτήσεις, οι οποίες είναι οι παρακάτω:

- GetLines: Μέσω της συνάρτησης len, ελέγχει τον αριθμό γραμμών του αρχείου χρηστών που αποστέλλουν μηνύματα. Ο αριθμός αυτός είναι ο αριθμός των επαναλήψεων που θα εκτελέσει το πρωτόκολλο.
- AnalyzeGraph: Διαβάζει το αρχείο που περιέχει τη δομή του γράφου και επιστρέφει μια list of lists, όπου κάθε εμφωλευμένη λίστα είναι η λίστα γειτόνων του χρήστη στη θέση της λίστας. Για παράδειγμα, στη εάν ο γράφος μου είναι ο:
  0 1 0
  1 0 1
  0 1 0
  Τότε η λίστα μου είναι η [[1], [0,2], [1]].
- GetUsers: Διαβάζει το αρχείο των αποστολέων και επιστρέφει λίστα με τους αποστολείς.

- Forward: Διαδικασία προώθησης μηνύματος. Παίρνει ως είσοδο τον current sender και τη λίστα των γειτόνων του, προσθέτει στη λίστα αυτή τον εαυτό του (μιας και μπορεί να προωθήσει το μήνυμα στον εαυτό του) και μέσω τυχαίας επιλογής ομοιόμορφης κατανομής (random.choice), επιλέγει τυχαία έναν γείτονα και τον επιστρέφει.

Κατά την έναρξη του προγράμματος γίνεται ένα τυπικό validation. Ελέγχεται εάν δόθηκε ο σωστός αριθμός ορισμάτων, και σε περίπτωση που έχουμε fixing strategy (δηλαδή broken paths διάφορο του μηδενός) εάν είναι ένα από τα 2 διαθέσιμα strategies που υλοποιούνται (last honest / initiator). Έπειτα, διαβάζονται τα αρχεία εισόδου και αναλύονται με τις 3 πρώτες βοηθητικές συναρτήσεις.

Δημιουργείται μια λίστα detectedNodesTotal η οποία κρατάει το σύνολο των detected nodes για κάθε αποστολέα. Στη λίστα αυτή γίνεται append η λίστα των detected nodes ενός αποστολέα, οπότε είναι μια list of lists, που είναι ευθυγραμμισμένη με την λίστα των αποστολέων. Για κάθε λοιπόν αποστολέα, ξεκινάμε μια κεντρική επανάληψη.

Αρχικοποιούνται κάποιες μεταβλητές και έπειτα ξεκινάει η διαδικασία αποστολής μηνυμάτων. Όσο δεν υπάρχει εντολή να γίνει το μήνυμα deliver στον web server, πρέπει να κάνουμε forwards. Η πρώτη αποστολή είναι πάντα forward. Θέτουμε σαν προηγούμενο χρήστη τον χρήστη ο οποίος στέλνει το μήνυμα και σαν τρέχων χρήστη τον χρήστη που μας επιστρέφει η συνάρτηση Forward που περιεγράφηκε ανωτέρω. Σε περίπτωση που ο χρήστης που μας επιστρέφει η Forward ανήκει στην λίστα των Corrupted χρηστών, έχουμε detection. Στην περίπτωση detection:

- Εάν έχουμε broken paths = 0:
    - Συνεχίζουμε κανονικά την εκτέλεση του πρωτοκόλλου δίχως διακοπή. Εξασφαλίζουμε πως η επόμενη αποστολή θα είναι προώθηση και όχι deliver στον web server.
- Εάν έχουμε broken paths != 0:
    - Γίνεται έλεγχος του αριθμού των detections που έχουν γίνει. Εάν είναι ίσος με τον αριθμό των broken paths, τότε η εκτέλεση για τον συγκεκριμένο χρήστη σταματάει χωρίς να γίνει deliver το μήνυμα.
    - Σε περίπτωση που δεν ισούται με τον αριθμό των broken paths, αυξάνουμε μια μεταβλητή άθροισης και ελέγχουμε τον τύπο της στρατηγικής διόρθωσης. Εάν είναι Initiator, κάνουμε current user τον αρχικό χρήστη. Αλλιώς, εάν είναι last-honest, κάνουμε current user τον previous user, που γνωρίζουμε πως ήταν τίμιος. Σε όλες τις περιπτώσεις αυτές, ο detected user παραμένει στη λίστα.
    - Εξασφαλίζουμε πως η επόμενη αποστολή θα είναι προώθηση και όχι deliver στον web server.

Μετά την πρώτη αποστολή που γνωρίζουμε πως θα είναι forward (όπως και οι αποστολές μετά από κάθε detection), γίνεται έλεγχος συνθήκης σχετικά με το εάν το μήνυμα θα γίνει deliver στον web server ή θα προωθηθεί σε επόμενο χρήστη. Αυτό γίνεται μέσω επιλογής τυχαίου αριθμού μεταξύ 0 και 1 και σύγκρισής του με το Φ (forwarding probability) που δίνεται ως input. Εάν ο τυχαίος αριθμός είναι μικρότερος από το Φ, τότε γίνεται προώθηση σε επόμενο κόμβο. Αλλιώς, γίνεται deliver.

Σε περίπτωση forwarding, ακολουθούνται τα βήματα που περιεγράφηκαν παραπάνω. Σε περίπτωση deliver, σημαίνει πως η εκτέλεσή μας έχει φτάσει στο τέλος της. Άρα:

- Αλλάζουμε την μεταβλητή delivered σε True για να σταματήσει η εσωτερική επανάληψη.
- Αποθηκεύουμε τους detected χρήστες στην λίστα detectedNodesTotal.
- Εκτυπώνουμε στην standard έξοδο ποιοι χρήστες έγιναν detect, με τον ακόλουθο τρόπο:
    - "Initial Sender: {X} – Nodes detected: [{Y_1, Y_2, … Y_n}]
    - Παράδειγμα: Initial Sender: 1 -- Nodes detected: [49, 63, 57]

Με την ολοκλήρωση της εκτέλεσης προχωράμε στον επόμενο αποστολέα μέχρι η λίστα να φτάσει στο τέλος της και να ολοκληρωθεί η διαδικασία της προσομοίωσης του Crowds. Όταν ένα μήνυμα γίνεται deliver, ο web server επειδή είναι στον έλεγχο του αντιπάλου, κάνει detect τον τελευταίο αποστολέα, πράγμα που μας εξασφαλίζει πως σε κάθε εκτέλεση, θα έχουμε τουλάχιστον έναν detected user.

4. Ανάλυση αποτελεσμάτων

4.1. F-BLEAU

Για την ανάλυση των αποτελεσμάτων της προσομοίωσης χρησιμοποιείται το F-BLEAU, το οποίο είναι εργαλείο που λειτουργεί σαν black-box και επιστρέφει διάφορες μετρικές που σχετίζονται με τον βαθμό διαρροής (leakage), ανάλογα τα δεδομένα που του δίνουμε. Συγκεκριμένα, βασίζεται στον υπολογισμό του σφάλματος ενός machine learning μοντέλου. Όπως θα δούμε, δέχεται στις παραμέτρους του διαφορετικούς τύπους classification για τον υπολογισμό της εξόδου.

Χρησιμοποιήθηκε η Python έκδοση του F-BLEAU κι αυτό για να μην χρειάζεται να «τεμαχίζουμε» το πρόγραμμα σε δύο αρχεία, οπού το πρώτο θα παράγει τα αποτελέσματα του simulation και το δεύτερο θα τα καταναλώνει, χάριν συντομίας. Τα δεδομένα που βγαίνουν από το πρώτο μέρος της προσομοίωσης, υπόκεινται σε λίγη επεξεργασία και μετά δίνονται στο F-BLEAU για τον υπολογισμό διαφόρων μετρικών που θα περιεγραφούν παρακάτω.

4.2. Προετοιμασία δεδομένων

Τα δεδομένα με τα οποία τροφοδοτούμε το F-BLEAU είναι:

- Η λίστα των αποστολέων (Secret)
- Η λίστα λιστών των detected χρηστών (Observations)

Όπως είπαμε, επειδή πίσω από το F-BLEAU κρύβεται ένα μοντέλο μηχανικής μάθησης, αυτό πρέπει να εκπαιδευτεί καταλλήλως για να μπορεί να προβλέψει σωστά. Οπότε χρειάζεται να «σπάσουμε» τα δεδομένα μας σε 2 κομμάτια. Ένα Train set και ένα Test set. Γνωρίζουμε πως πρέπει:

- Το σύνολο των Secrets του Train set να υπάρχει και στο Test set,
- Κάθε λίστα detection που ανήκει στο σύνολο, δηλαδή στη μεγάλη λίστα, πρέπει να έχει ίδιο μήκος. Πράγμα που είναι πολύ δύσκολο να γίνει στην εκτέλεση του πρωτοκόλλου διότι, επειδή είναι εντελώς τυχαίο, κάποια

εκτέλεση μπορεί να έχει 1 detected node, ενώ κάποια μπορεί να έχει παραπάνω. Οπότε, για να κάνουμε όλες τις λίστες ίδιου μεγέθους και να μπορεί να τροφοδοτηθεί στο F-BLEAU, προσθέτουμε όπου χρειάζεται την τιμή -1, η οποία σημαίνει no detection.

Πρέπει λοιπόν, αφού συμπληρωθούν με -1 όσες λίστες χρειάζεται, χωρίζουμε τα δεδομένα μας σε train και test datasets με ratio 75% - 25%. Υπάρχει κίνδυνος, για μικρό αριθμό δεδομένων, να μην υπάρχουν κάποια secret του test στο train. Οπότε, τα πειράματα που ακολουθούν έγιναν με μεγάλο αριθμό δεδομένων (όπως ζητήθηκε και στην εκφώνηση) και αποφεύχθηκε αυτό το ενδεχόμενο. Έπειτα, μέσω των python bindings καλείται το F-BLEAU για την λήψη των αποτελεσμάτων.

4.3. Σχετικά με τις μετρήσεις

Οι μετρικές που θα παρακολουθήσουμε είναι:

- Random Guessing Error: **$1-V_b[\pi]$** - Η πιθανότητα του αντιπάλου να κάνει **ΛΑΘΟΣ** πρόβλεψη, όταν εκείνος λαμβάνει υπ' όψη μόνο την αρχική γνώση και όχι τα αποτελέσματα

- Minimum estimate: $1-V_b[\pi,C]$ - Η ελάχιστη πιθανότητα να αποτύχει ο αντίπαλος, αφού δει την έξοδο.

Στις μετρήσεις που ακολουθούν, θα χρησιμοποιηθεί γράφος 100 χρηστών ο οποίος δημιουργήθηκε με adjacency probability = 0.5. Τα υπόλοιπα στοιχεία του συστήματος, όπως Φ, Broken Paths και στρατηγικές διόρθωσης, corrupted users και αριθμός αποστολών μηνυμάτων θα αλλάζουν για να παρατηρηθούν αλλαγές στη συμπεριφορά του F-BLEAU με διαφορετικό τύπο δεδομένων.

Επειδή μιλάμε για πιθανοτικά συστήματα και για μηχανική μάθηση, είναι φυσιολογικό μερικές τιμές να έχουν «σκαμπανεβάσματα». Συγκεκριμένα, μπορεί κάποια τιμή να αναμένεται να είναι μικρότερη μιας άλλης, αλλά αυτό να μην συμβεί στην συγκεκριμένη εκτέλεση της διαδικασίας. Δίνεται περισσότερο βάρος στην «τάση» των διαγραμμάτων, καθώς μερικές μεμονωμένες τιμές μπορεί να είναι απλά τυχαίες.

4.4. Μετρήσεις

Για τις μετρήσεις, έγινε επιλογή του estimator "NN-Bounds". Αυτή η επιλογή βασίζεται στο ότι οι υπόλοιπες επιλογές (KNN-ln, KKN-Log10, NN) επέστρεφαν μεγαλύτερες τιμές minimum estimate, και όπως προτείνεται, επιλέχθηκε εκείνη που επιστρέφει το μικρότερο. Η επιλογή "Frequentist" δεν δοκιμάστηκε διότι επιστρέφει σφάλμα κατά την εκτέλεσή της, για το οποίο δεν βρέθηκε κάποια λύση μετά από έρευνα.

4.4.1. Random Guessing Error

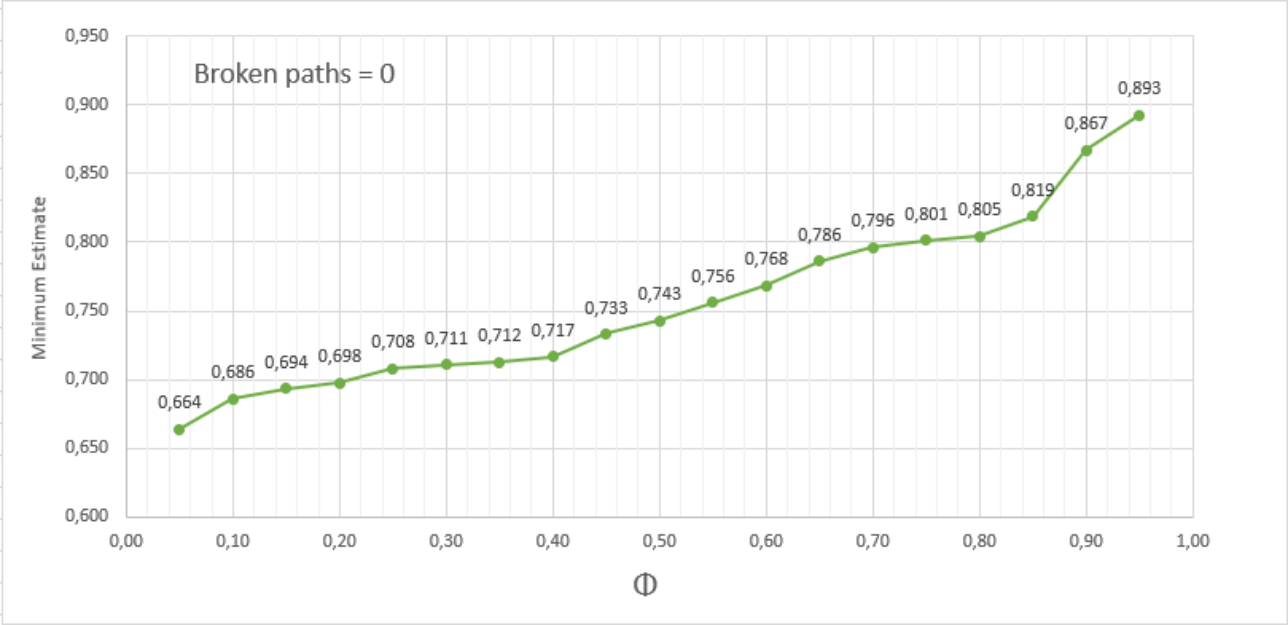
Η πρώτη παρατήρηση, κατά την εκτέλεση των διαφόρων συνδυασμών συστημάτων με όλων τον ειδών διαφορετικές παραμέτρους, όπως επίσης και ακραίες περιπτώσεις, παρατηρούμε πως δεν αλλάζει καθόλου το Random Guessing Error. Αυτό είναι φυσιολογικό, μιας και όπως αναφέραμε ανωτέρω, εξαρτάται **μόνο** από την αρχική γνώση, χωρίς να λάβει υπ' όψη τα αποτελέσματα. Άρα, το Random Guessing Error είναι (με uniform π) για γράφο 100 χρηστών όπου οι 20 είναι corrupted (δηλαδή με 80 τίμιους χρήστες):

$$1 - V_b[\pi] = 0.9828$$

Οι μετρήσεις, τα γραφήματα και ο σχολιασμός που ακολουθούν θα εστιάσουν στο minimum estimate.

4.4.2. Minimum Estimate – Forwarding Probability (Φ)

Οι συγκρίσεις ξεκινούν με την μεταβολή του Φ και την διατήρηση όλων των υπολοίπων παραμέτρων του συστήματος σταθερών, με την μεταβλητή Broken Paths να είναι μηδέν. Συγκεκριμένα, οι υπόλοιπες σταθερές είναι: Corrupted: 20, Μηνύματα: 10.000.

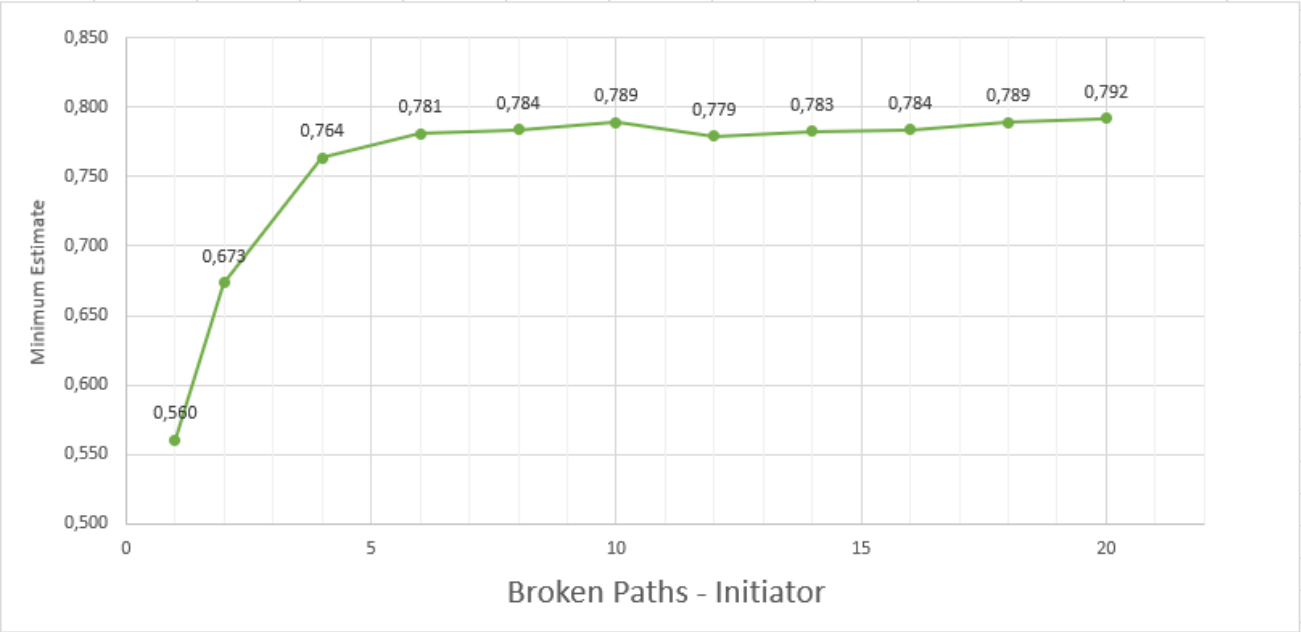


Επίσης, γίνεται μια δοκιμή του συστήματος για Φ αλλά με Broken Paths = 2 και στρατηγική Initiator, για να μελετήσουμε και αυτή τη συμπεριφορά και πόσο βάρος έχει συγκριτικά με την μεταβολή του Φ.

Όσον αφορά και τα δύο διαγράμματα, παρατηρούμε πως η αύξηση του Φ σημαίνει πως η πιθανότητα ο αντίπαλος να κάνει λάθος αυξάνεται, πράγμα που επιβεβαιώνεται και από την θεωρία ποσοτικής ροής, αλλά και από την κοινή λογική. Όταν ένα σύστημα κάνει Forward ένα μήνυμα περισσότερες φορές από ένα άλλο, έχει μεγαλύτερες πιθανότητες το μήνυμα αυτό να μην μπορέσει να γίνει trace πίσω στον αρχικό αποστολέα, λόγω των περισσότερων χρηστών στους οποίους έχει φτάσει και έχει κάνει bounce. Λόγω του περιορισμού των detections σε 2 όταν εφαρμόζουμε την παράμετρο Broken Paths, γίνονται λιγότερα detections, πράγμα που βοηθάει τον αντίπαλο να βρει πιο εύκολα τον αποστολέα, σε αντίθεση με το broken paths = 0. Παρατηρούμε πως ενώ στις μικρές τιμές του Φ (0,1 – 0,3) οι τιμές παραμένουν σχετικά χαμηλές και στα δύο διαγράμματα λόγω των πολλών delivers (χαμηλό forwarding probability). Ωστόσο, όταν έχουμε πολλά forwards (0,7 – 0,9) βλέπουμε τρομακτική διαφορά μεταξύ των δύο διαγραμμάτων, λόγω των πολλών περισσότερων detections στο σύστημα όπου δεν υπάρχει έλεγχος μονοπατιών.

4.4.3. Minimum Estimate – Broken Paths (Initiator)

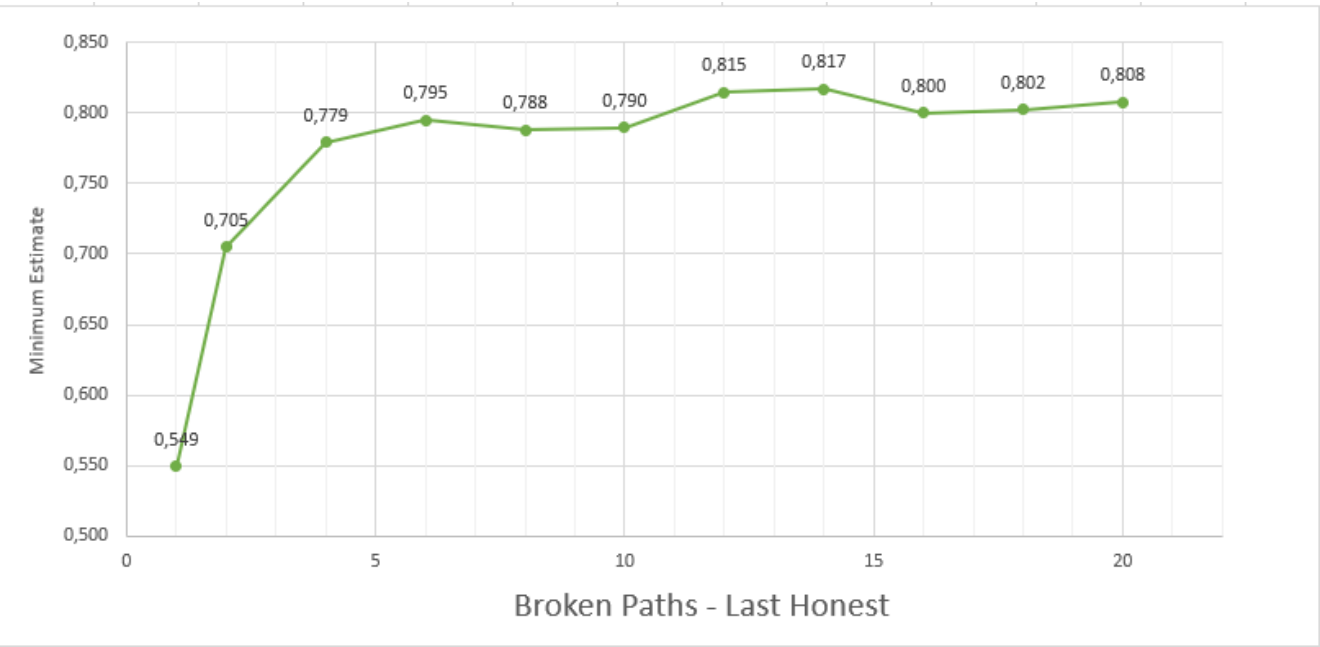
Στην συγκεκριμένη μέτρηση, θα έχουμε μεταβλητό μέγεθος Broken paths, με fixing strategy Initiator. Φ = 0,7, Corrupted = 20, Μηνύματα 10.000.



Όπως αναφέρθηκε και πριν, ο μέγιστος αριθμός των detections είναι αρκετά σημαντικός, κατά κύριο λόγο όταν μιλάμε για λίγα μονοπάτια. Στο σύστημα που μελετάται με 20% διεφθαρμένους χρήστες, όταν σταματάει η προώθηση μετά από έναν και μόλις εντοπισμό, ο αντίπαλος έχει συγκεκριμένο pool από υποψήφιους αποστολείς στο οποίο μπορεί να ψάξει τον πραγματικό sender. Άρα, η πιθανότητα να κάνει λάθος είναι αρκετά μικρότερη από την πιθανότητα όταν αυτός ο αριθμός αυξάνεται. Παρατηρείται ωστόσο πως υπάρχει ένα άνω φράγμα για το πόσο μπορεί να επηρεάσει ο αριθμός των κατεστραμμένων μονοπατιών τον αντίπαλο και πως μετά από ένα σημείο, επειδή οι λίστα των detections παραμένει σχετικά σταθερή, δηλαδή οι corrupted έχουν μαζέψει λογικά το μεγαλύτερο πλήθος χρηστών με τους οποίους είναι συνδεδεμένοι. Οπότε, τα τρομακτικά μεγάλα νούμερα στην παράμετρο broken paths δεν οδηγούν σε πολύ καλύτερα αποτελέσματα όπως παρατηρείται και από το ανωτέρω διάγραμμα. Η πιθανότητα αποτυχίας του αντιπάλου κινείται σε σταθερά πλαίσια μετά το broken paths = 8.

4.4.4. Minimum Estimate – Broken Paths (Last Honest)

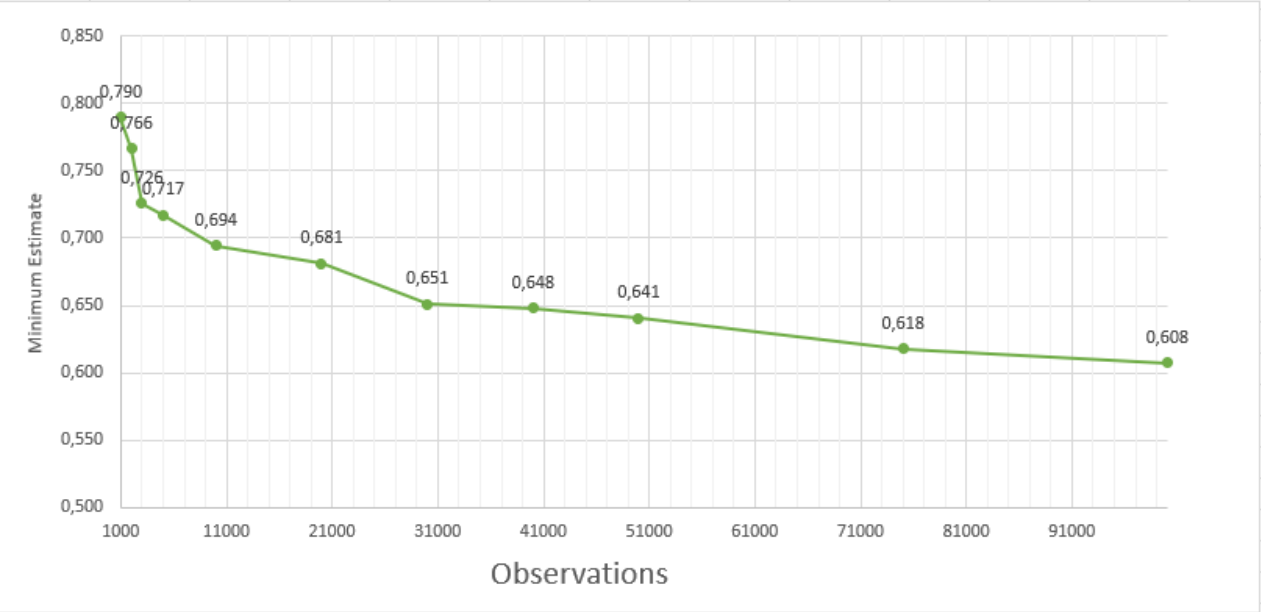
Παρακάτω θα έχουμε και πάλι μεταβλητό μέγεθος Broken paths, με fixing strategy Last honest. Φ = 0,7, Corrupted = 20, Μηνύματα 10.000.



Ενώ στην προηγούμενη υποενότητα ασχοληθήκαμε κυρίως με την επίπτωση του broken paths, θα σχολιαστεί η επίδραση της επιλογής του fixing strategy. Παρατηρούμε αρχικά πως το διάγραμμα είναι αρκετά παρόμοιο σχηματικά με το διάγραμμα όταν χρησιμοποιήθηκε Initiator fixing strategy. Ωστόσο παρατηρείται πως συγκριτικά με το initiator, το last honest παρέχει «καλύτερα» αποτελέσματα μιας και είναι πιθανότερο να κάνει λάθος ο αντίπαλος. Αυτό είναι επίσης αρκετά λογικό, εάν σκεφτούμε πως στο initiator το μήνυμα ξαναστέλνεται από την αρχή, άρα ξεκινάει ξανά από τον αποστολέα που θέλουμε να «κρύψουμε». Με το last honest δεν υπάρχει αυτό το αρνητικό μιας και το μήνυμα επιστρέφεται στον τελευταίο τίμιο χρήστη και συνεχίζει το «ταξίδι» του στο δίκτυο.

4.4.5. Minimum Estimate – # Μηνυμάτων

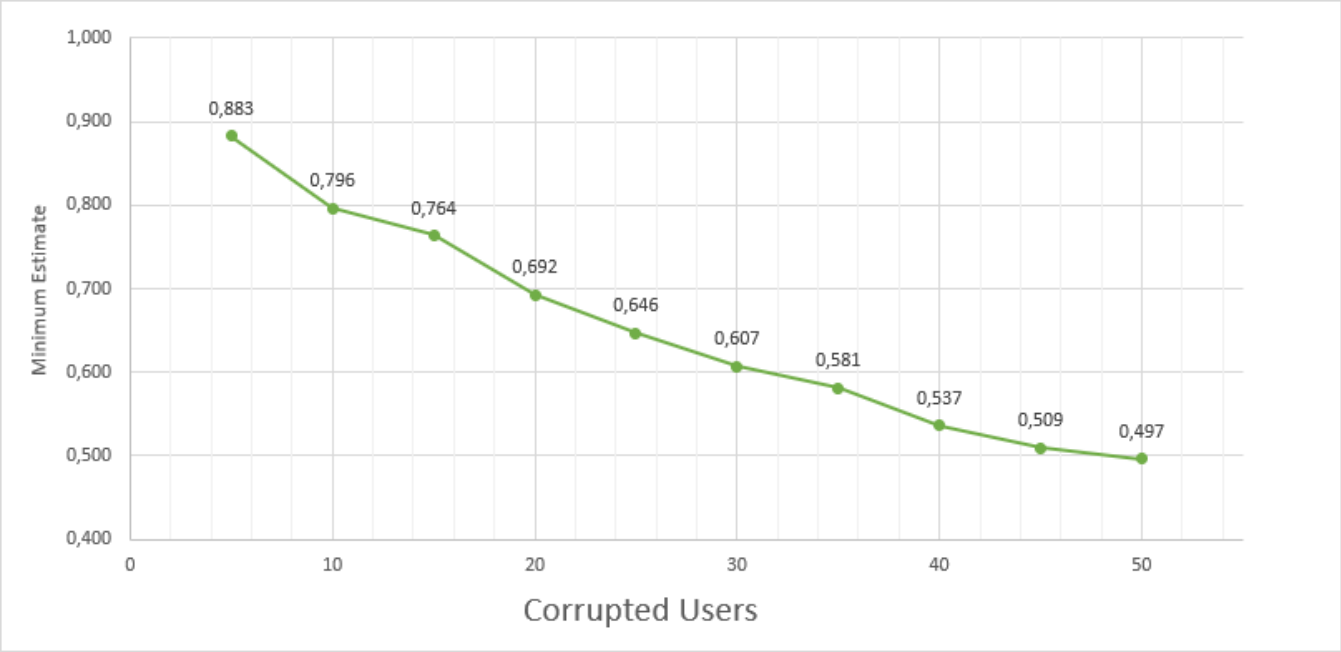
Στο διάγραμμα που ακολουθεί, βλέπουμε την εναλλαγή του minimum estimate συναρτήσει του αριθμού των observations, δηλαδή των εκτελέσεων του πρωτοκόλλου. Φ = 0,7 ,Broken Paths =2 με Initiator, 20 διεφθαρμένους χρήστες.



Είναι εμφανές πως με την αύξηση των μηνυμάτων που αποστέλλονται, τόσο πιο εύκολο είναι για τον αντίπαλο να εντοπίσει σωστά τον αποστολέα, πράγμα που εξηγείται από το ότι ο αντίπαλος έχει περισσότερες «ευκαιρίες» να αντλήσει πληροφορίες για τον αποστολέα και επίσης επιβεβαιώνεται και από την θεωρία ποσοτικής ροής.

4.4.6. Minimum Estimate – Corrupted Users

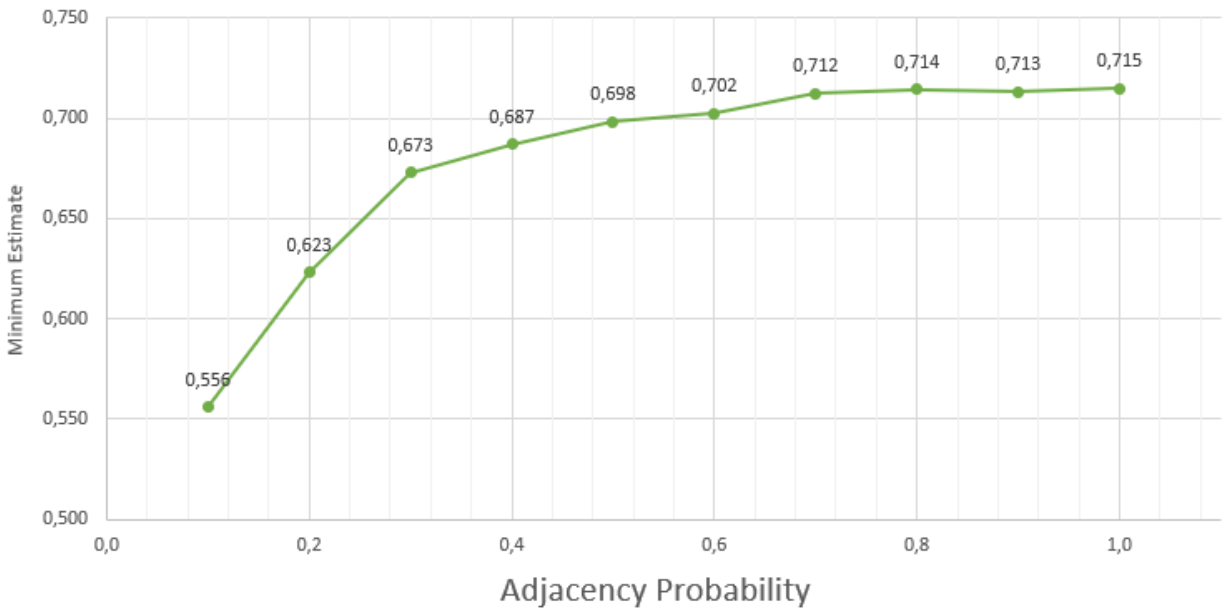
Συνεχίζουμε με την μεταβολή των διεφθαρμένων χρηστών. Οι υπόλοιπες σταθερές είναι: Broken Paths = 2 με initiator, Μηνύματα: 10.000, Φ = 0,7.



Όσο αυξάνεται ο αριθμός των διεφθαρμένων χρηστών, τόσο μεγαλύτερη είναι η πιθανότητα επιτυχίας του αντιπάλου. Ο αντίπαλος έχει περισσότερους «φίλιους» κόμβους μέσα στο δίκτυο με αποτέλεσμα να υπάρχουν detections περισσότερων unique κόμβων. Παρατηρείται μάλιστα πως όταν ο αριθμός των corrupted χρηστών είναι 50, δηλαδή οι μισοί κόμβοι του γράφου, η πιθανότητα επιτυχίας και αποτυχίας είναι 0.5, πράγμα που εμπειρικά φαντάζει πολύ σωστό.

4.4.7. Minimum Estimate – Adjacency Probability

Τέλος, δίνουμε σαν input γράφο 100 χρηστών ο οποίος έχει δημιουργηθεί με διαφορετικό adjacency probability. Οι υπόλοιπες παράμετροι είναι Φ=0,7, Broken Paths = 2 με initiator, 20 corrupted χρήστες.



Έχοντας κατά νου πως όταν έχουμε adjacency probability ίση με την μονάδα, ο γράφος είναι πυκνός και υπάρχουν ακμές από κάθε κόμβο προς κάθε κόμβο. Η μικρότερη τιμή του διαγράμματος (0,1) έχει και την καλύτερη πιθανότητα επιτυχίας του αντιπάλου. Αυτό συμβαίνει διότι ένας corrupted κόμβος είναι συνδεδεμένος με λιγότερους τίμιους κόμβους. Εάν γίνει κάποιο detection, οι πιθανότητες είναι περισσότερες αυτός ο αποστολέας να είναι ο initial sender, σε αντίθεση με όταν ο corrupted χρήστης συνδέεται με μεγάλο αριθμό χρηστών.