Δίκτυα 2 2020

Report

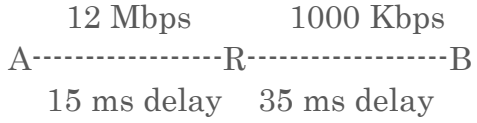
Αναφορά εργαστηρίου

Λιάπης Σάββας 57403

Υπεύθυνος καθηγητής : Τσαουσίδης Βασίλειος

Υπόθεση

Θεωρήστε την παρακάτω τοπολογία που ορίζεται στον κώδικα του αρχείου lab.cc:



queue at R: size 10

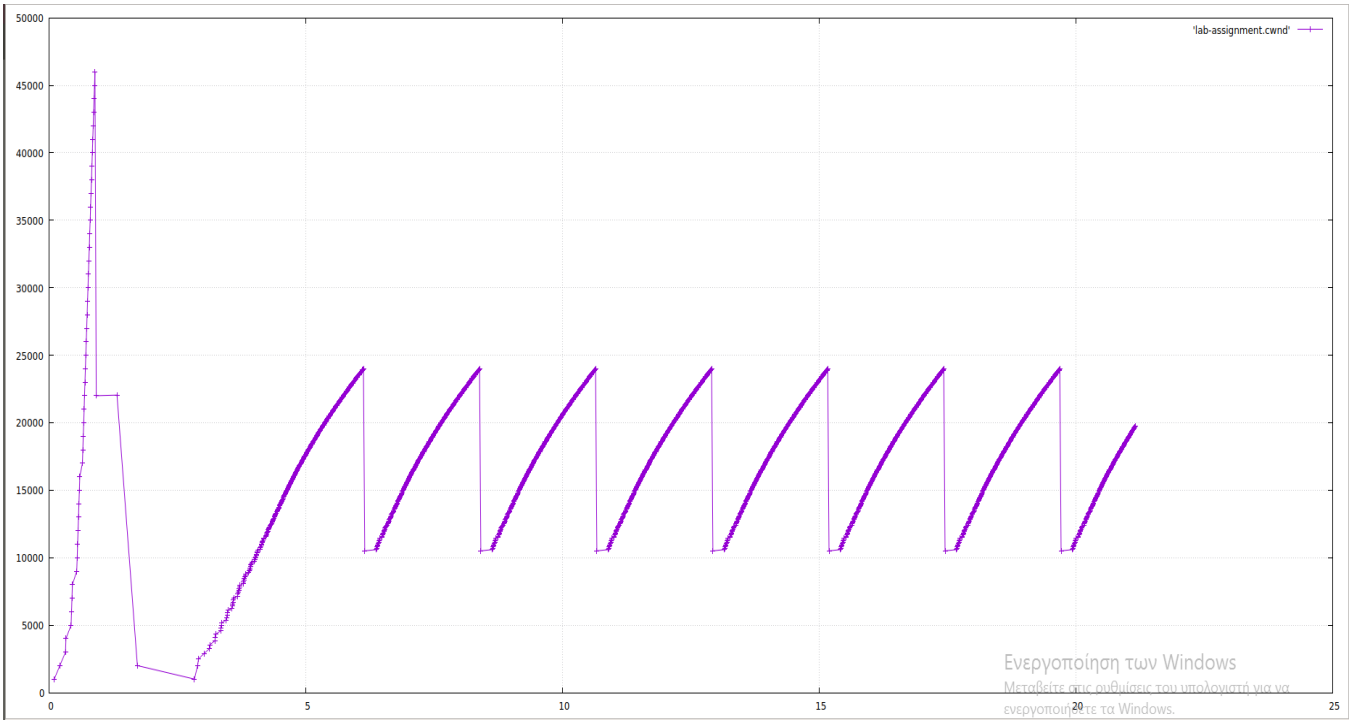
Άσκηση 1

Ζητούμενα

1. Τρέξτε την προσομοίωση με τις default παραμέτρους.
    a. Κάντε plot το διάγραμμα του παραθύρου συμφόρησης (cwnd) ως προς τον χρόνο και σχολιάστε το.
    b. Υπολογίστε το συνολικό throughput της προσομοίωσης.
2. Τρέξτε δύο διαδοχικές προσομοιώσεις, όπου θα ορίσετε το εύρος ζώνης της bottleneck σύνδεσης ίσο με [(y*100)+100] Kbps και (11-y)Mbps, αντίστοιχα.
    a. Κάντε plot το διάγραμμα του παραθύρου συμφόρησης (cwnd) ως προς τον χρόνο.
        - Τι αλλαγές παρατηρείτε ως προς το cwnd;
        - Για ποιο λόγο συμβαίνουν αυτές οι αλλαγές;
    b. Υπολογίστε το συνολικό throughput κάθε προσομοίωσης και συγκρίνετέ το με το αποτέλεσμα του ερωτήματος 1.1.b.
3. Θεωρώντας τις default παραμέτρους του δικτύου, τρέξτε δύο διαδοχικές προσομοιώσεις, όπου θα ορίσετε το πρωτόκολλο TCP ως TCPVegas και TCPWestwood.
    a. Κάντε plot το διάγραμμα του παραθύρου συμφόρησης (cwnd) ως προς τον χρόνο.
    b. Υπολογίστε το συνολικό throughput κάθε προσομοίωσης.

Απαντήσεις

Απάντηση (1α)



Γνωρίζουμε από τον κώδικα lab.cc οτί χρησιμοποιείται το πρωτόκολλο TCP New Reno. Αυτό μας βοηθάει να ερμηνέυσουμε καλύτερα το παραπάνω διάγραμμα.

Κατά την αφετηρία της εκτέλεσης του αλγορίθμου TCP New Reno το παράθυρο συμφόρισης (congestion window) αυξάνεται εκθετικά (slow start) εως ότου παρατηρηθεί απώλεια πακέτων. Στο διάγραμμα βλεπουμε την φάση slow start τέρμα αριστερά, όπου ξεκιναει η αποστολή πακέτων και το congestion window αυξάνεται τιουτοτρόπως.

Η αυξηση των πακέτων εκθετικά δεν είναι δυνατόν να συνεχίζεται για μεγάλο χρονικό διάστημα διότι το μέγεθος του buffer του δρομολογητή (πρωτίστως) και του παραλλήπτη είναι πεπερασμένο οπότε με την εκθετική αυξηση των πακέτων αποστολής σύντομα θα σημειωθεί απώλεια πακέτων καθώς δεν θα χωράνε στον buffer του router. Για αυτό τίθεται ένα ανώτατο όριο (slow start thresshold) κατά το οποίο θα πραγματοποιείται η διαδικασία slow start. Βέβαια, μπορεί, αν αυτό το όριο έχει τεθεί πολύ υψηλά, να σταλεί και πάλι μεγάλος αριθμός πακέτων, τα οποία ο buffer του δρομολογητή δεν είναι ικανός να χωρέσει, με αποτέλεσμα την απώλεια τους. Όταν λοιπόν σημειωθεί αυτή η απώλεια υπάρχουν δύο δυνατές εκφάνσεις:

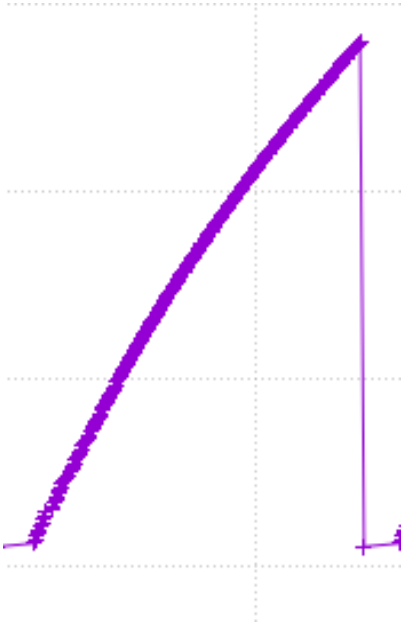
1) Απώλεια μεγάλου αριθμού πακέτων (καταστροφική συμφόρηση) . Σε αυτή την περίπτωση το πρωτόκολλο ενεργεί ακολούθως: το congestion window μειώνεται στην αρχική του τιμή (1πακέτο) και το ssthresh υποδιπλασιάζεται. Η διαδικασια αυτή ονομάζεται γρήγορη επαναμετάδωση.

2) Απώλεια πολύ μικρού αριθμού πακέτων (περιορισμένη συμφόρηση) . Σε αυτή την περίπτωση η επαναμετάδωση του χαμένου πακέτου πραγματοποιείται μετά από τρείς (συνήθως) διπλές επιβεβαιώσεις του τελευταίου πακέτου που παρελλήφθει. Σε αυτή την φάση τόσο το congestion window όσο και το ssthresh υποδιπλασιάζονται. . Η διαδικασία αυτή ονομάζεται γρήγορη ανάκαμψη και αποσκοπεί στο να μην ελαχιστοποιηθεί η ροή πακέτων στο δίκτυο.

Όλα αυτά παρατηρούνται στο διάγραμμα με τον εξής τρόπο. Παρατηρούμε ότι δημιουργείται μία μύτη στο μέγεθος του congestion window. Εκει καταλαβαίνουμε ότι έχει πραγματοποιηθεί συμφόρηση και έχουν χαθεί πακέτα. Το δίκτυο επιλέγει να μειώσει τον αριθμό του cwnd στο μισό της τιμής που έχει λάβει οπότε θεωρεί ότι δεν έχει επέλθει καταστροφική συμφόριση και μπαίνει στην φάση της γρήγορης ανάκαμψης (εδώ μειώνεται και το slow start threshold στο μισό της αρχική του τιμής) . Ωστόσο αφού επέλθει Timeout ο αποστολέας καταλαβαίνει ότι στην πραγματικότητα έχει επέλθει καταστροφική συμφόριση οπότε πραγματοποιεί την διαδικασία της γρήγορης ανάκαμψης . το cwnd λαμβάνει τώρα την αρχική του τιμή και το threshold ξανα υποδιπλασιάζεται. Η γρήγορη επαναμετάδωση ολοκληρώνεται και ξεκινάει εκ νέου ο μηχανισμός αργής εκκίνησης.

Δεν έχουμε εξηγήσει ακόμα τι συμβαίνει όταν το παράθυρο συμφόρισης ξεπεράσει το κατώφλι αργής εκκίνησης. Από το σημείο αυτό και μετά ενεργοποιείται η αθροιστική αύξηση δηλαδή το παράθυρο συμφόρισης αυξάνεται κατά 1 κάθε φορά.

Στο παραπάνω διάγραμμα αφού το congestion window λάβει την ελάχιστη τιμή του ξεκινάει πάλι τον μηχανισμό αργής εκκίνησης όπως έχουμε περιγράψει παραπάνω. Ωστόσο αυτή την φορά βλεπουμε ότι η εκθετική αύξηση διαρκεί για πολύ μικρό χρονικό διάστημα και μετά από αυτό παρατηρείται ότι ενεργοποιείται η αθροιστική αυξηση. Αυτό μας δινει να καταλάβουμε ότι το ssthresh μετά τον υποτετραπλασιασμό του έχει λάβει μία τιμή γυρω στο 5000 (μικρότερη), του διαγράμματος. Από εκει και μετά λοιπόν πραγματοποιείται προσθετική αυξηση εως ότου υπάρξει περιορισμένη συμφόρηση, όπου το congestion window θα υποδιπλασιαστεί.



Αφού το παραθυρό συμφόρησης υποδιπλασιαστεί παρατηρούμε ότι ξεκινάμε πάλι αποστολή με προσθετική αύξηση (και όχι με slow start) καθώς το μέγεθος του congestion window παραμένει μεγαλύτερο (ή ισο) του ssthresh. Αυτή η διαδικασία (που περιγράφεται σε αυτή την παράγραφο) πραγματοποιείται 7 φορές εως ότου ολοκληρωθεί η αποστολή.

Μία σημείωση που πρέπει να ληφθεί υπ όψιν κατά την παρατήρηση του διαγράμματος είναι ότι ένω περιμένουμε ιδανικά κατά την προσθετική αυξηση να εμφανίζεται ένα αυθύγραμο τμήμα σε κλίση εμείς βλεπουμε μία ελαφρια καμπυλότητα. Καθώς γεμίζει ο buffer του router τα πακέτα που φτάνουν περιμένουν περισσότερη ώρα στην ουρά για να σταλούν από τον δρομολογητή στον παραλλήπτη. Αυτό εξηγει και αυτή την κυρτότητα που εμφανίζεται .

Μια τελευταία παρατήρηση έχει να κάνει με τον χρονικό σημείο που φαίνεται να σταματά η αποστολή πακέτων στο διάγραμμα . Εμείς έχουμε ορίσει ότι η αποστολή θα σταματάει στα 20 sec όμως βλέπουμε ότι το congestion window συνεχίζει να αυξάνεται για λίγα second ακόμη. Αυτό υποθέτουμε ότι συμβαίνει επειδή μπορεί να έχει σταματήσει η αποστολή πακέτων στα 20 second αλλα ο παραλλήπτης δεν τα έχει παραλλάβει όλα σε εκείνο το χρονικό σημείο οπότε συνεχίζει να αποστέλει επιβεβαιώσεις με αποτέλεσμα το cwnd να αυξάνεται χωρίς όμως αυτό να σημαίνει ότι στέλνει δεδομένα μετά τα 20 sec.

Απάντηση (1.1.b)

Αποτελέσματα προσομοίωσης για τις προεπιλεγμένες τιμές είναι :

```
savvas@savvas-VirtualBox:~/workspace/bake/source/ns-3.29$ ./waf --run lab
Waf: Entering directory `/home/savvas/workspace/bake/source/ns-3.29/build'
Waf: Leaving directory `/home/savvas/workspace/bake/source/ns-3.29/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (33.053s)
queuesize=10, delayRB=35
A's address: 10.0.0.1
B's address: 10.0.1.2
R's #1 address: 10.0.0.2
R's #2 address: 10.0.1.1
Total Bytes Received from B: 2123000
savvas@savvas-VirtualBox:~/workspace/bake/source/ns-3.29$
```

Το throughput είναι η μετρήσιμη ταχύτητα ροής, η μέγιστη ροή, όταν υπάρχουν πάντα δεδομένα προς αποστολή κατά το διάστημα της μέτρησης. Μετριέται σε Byte ανά δευτερόλεπτο.

$$Throughput = \frac{Σταλθέντα\ δεδομένα\ (Bytes)}{χρόνος\ (seconds)}$$

Βλέποντας τα αποτελέσματα της συγκεκριμένης προσομοίωσης πληροφορούμαστε ότι έχουν σταλεί 2123000 Bytes και γνωρίζουμε ότι ο χρόνος είναι 20 second. Συνεπώς :

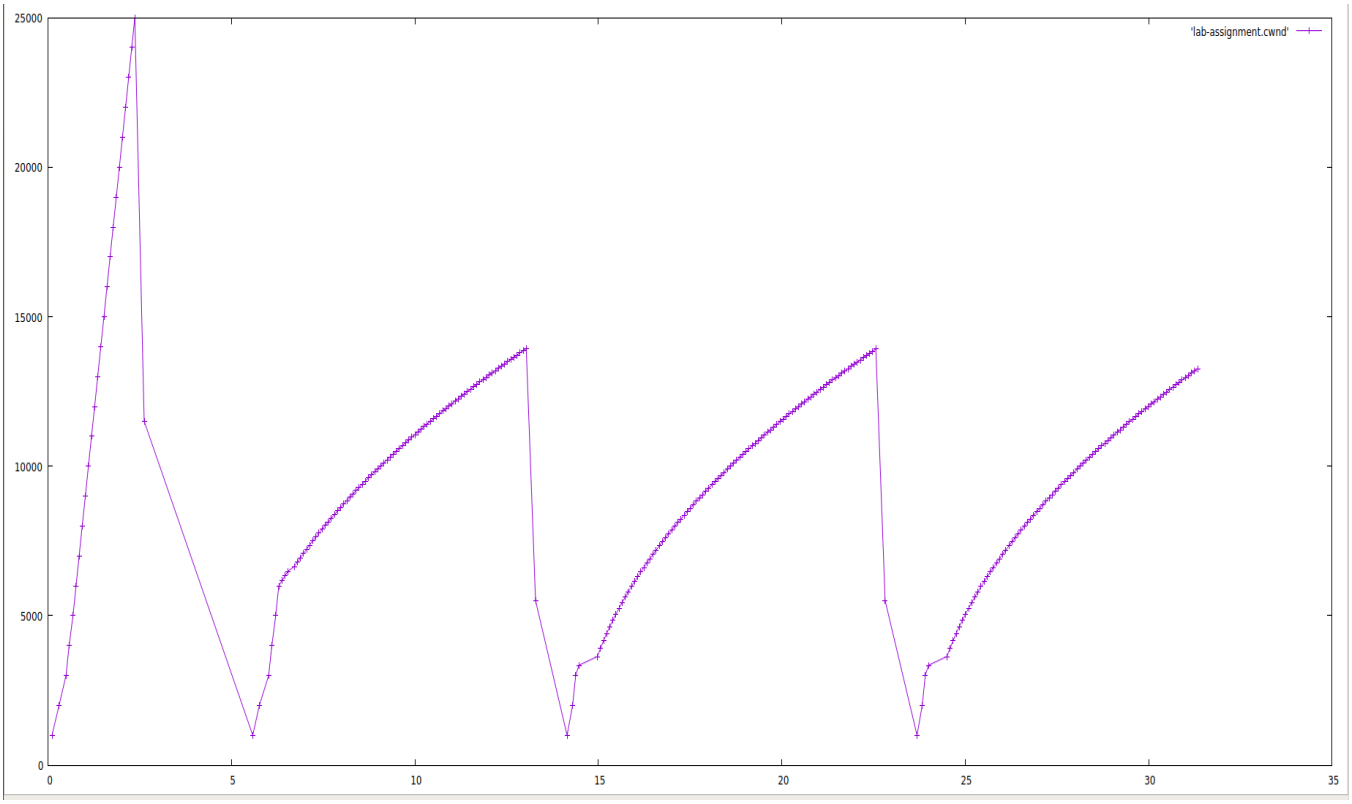
$$Throughput = \frac{2123000}{20} = 106150\ Bps = 106{,}2\ KBps$$

Απάντηση (1.2.α)

Από το ΑΕΜ 57403 προκύπτει ως y το 0, οπότε θα ορίσουμε σαν εύρος ζώνης της bottleneck σύνδεσης ίσο με

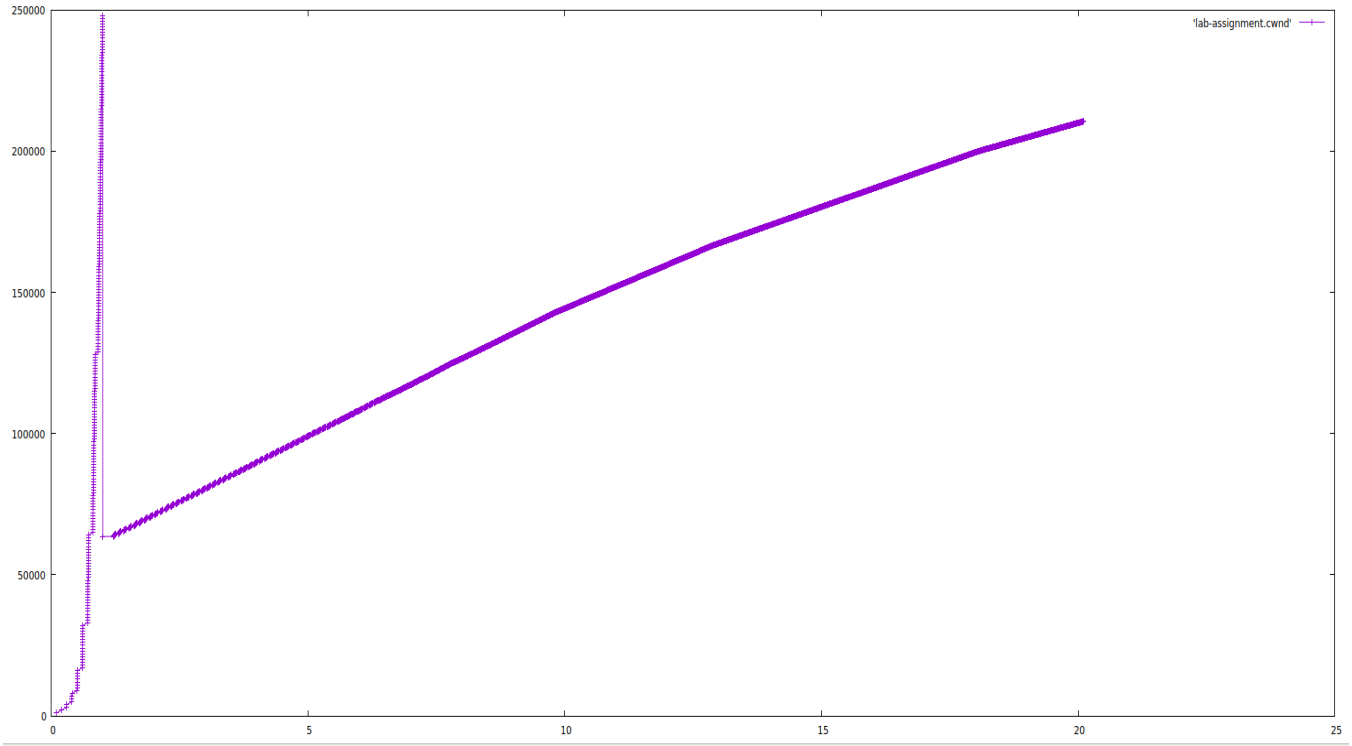
i. 100 kbps
ii. 11 Mbps

100kbps



Σε αυτό το διάγραμμα παρατηρώ ότι η καταστροφική συμφόρηση πραγματοποιείται πολύ πιο γρήγορα από το πρώτο ερώτημα καθώς το bottleneck περιορίζεται στο 1/10 του αρχικού. Επίσης σε αυτή την περίπτωση βλέπουμε ότι πραγματοποιείται καταστροφική συμφόρηση 3 φορές μέχρι την ολοκλήρωση της αποστολής δεδομένων. Μπορούμε να παρατηρήσουμε επίσης την μείωση του threshold που μετά την πρώτη καταστροφική συμφόρηση βρίσκεται πιο ψηλά από ότι μετά την δεύτερη και τρίτη καταστροφική συμφόρηση. Αυτό το καταλαβαίνουμε γιατί σε κάθε αύξηση του cwnd (3 «πτερύγια»), πριν ενεργοποιηθεί ο μηχανισμός αθροιστικής αύξησης , μετά την πρώτη καταστροφική συμφόρηση ο μηχανισμός slow start είναι ενεργοποιημένος περισσότερο από ότι στην δεύτερη και Τρίτη .

11 Mbps



Σε αυτή την περίπτωση έχουμε ορίσει ένα πολύ μεγάλο bottleneck, τόσο μεγάλο που το congestion window μειώνεται μία μονο φορά μόνο ωστόσο δεν είναι ξεκάθαρο αν οφείλεται σε καταστροφική συμφόριση ή όχι, καθώς ούτε φτάνει στην αρχική του τιμή αλλα ούτε μειώνεται στην μισή της τελευταίας του τιμής. Για αυτό που είμαστε σίγουροι είναι ότι μετά την συμφόριση το congestion window είναι πλεόν μεγαλύτερο (ή ίσο) του threshold καθώς από εκει και μετά έχουμε αθροιστική αύξηση.

Γενικότερα παρατηρούμε στα δύο διαγράμματα ότι καθώς αλλάζει το bottleneck bandwidth αλλάζει και η μέγιστη τιμή που μπορεί να φτάσει το congestion window. Αυτό οφείλεται στο ότι με την αύξηση του bottleneck bandwidth στην ουσία αυξάνεται και η ποσότητα δεδομένων που επιτρέπονται στο δύκτιο. Το πρωτόκολλο TCP αυτό που προσπαθεί είναι να ελεγξει πού και αν υπάρχει σημείο στένωσης. Ο αποστολέας λοιπόν για να το καταφέρει αυτό, χρησιμοποιεί το congestion window για να ιχνηλατεί με έναν δυναμικό τρόπο την χωρητικότητα του δικτύου. Για αυτό το παράθυρο συμφόρισης καταλήγει να έχει τιμή ανάλογη (ή κοντινή με αυτή) της χωρητικότητας του δικτύου.

Απάντηση (1.2.b)

Αποτελέσματα προσομοίωσης για botleneck bandwidth 100 Kbps :

```
savvas@savvas-VirtualBox:~/workspace/bake/source/ns-3.29$ ./waf --run lab
Waf: Entering directory `/home/savvas/workspace/bake/source/ns-3.29/build'
[2694/2766] Compiling scratch/lab.cc
[2725/2766] Linking build/scratch/lab
Waf: Leaving directory `/home/savvas/workspace/bake/source/ns-3.29/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (40.142s)
queuesize=10, delayRB=35
A's address: 10.0.0.1
B's address: 10.0.1.2
R's #1 address: 10.0.0.2
R's #2 address: 10.0.1.1
Total Bytes Received from B: 347000
savvas@savvas-VirtualBox:~/workspace/bake/source/ns-3.29$
```

Το throughput που προκύπτει για την προσομοίωση με botleneck bandwidth 100 Kbps είναι

$$Throughput = \frac{347000}{20} = 17350\ Bps = 17.35\ KBps$$

Παρατηρούμε ότι το throughput είναι το 1/10 περίπου του throughput του ερωτήματος (1.1.b)

Αποτελέσματα προσομοίωσης για botleneck bandwidth 11 Mbps :

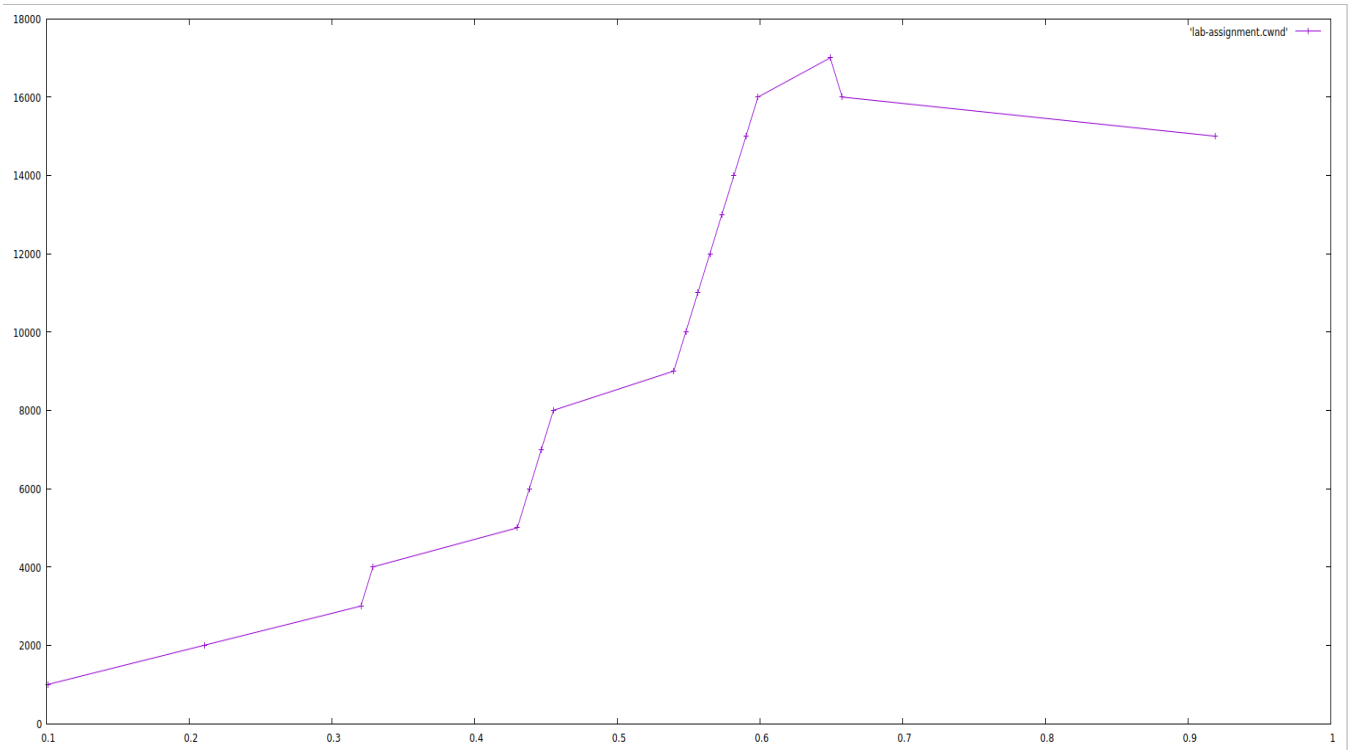
```
savvas@savvas-VirtualBox:~/workspace/bake/source/ns-3.29$ ./waf --run lab
Waf: Entering directory `/home/savvas/workspace/bake/source/ns-3.29/build'
Waf: Leaving directory `/home/savvas/workspace/bake/source/ns-3.29/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (32.831s)
queuesize=10, delayRB=35
A's address: 10.0.0.1
B's address: 10.0.1.2
R's #1 address: 10.0.0.2
R's #2 address: 10.0.1.1
Total Bytes Received from B: 22272000
savvas@savvas-VirtualBox:~/workspace/bake/source/ns-3.29$
```

$$Throughput = \frac{22272000}{20} = 1113600\ Bps = 1.114\ MBps$$

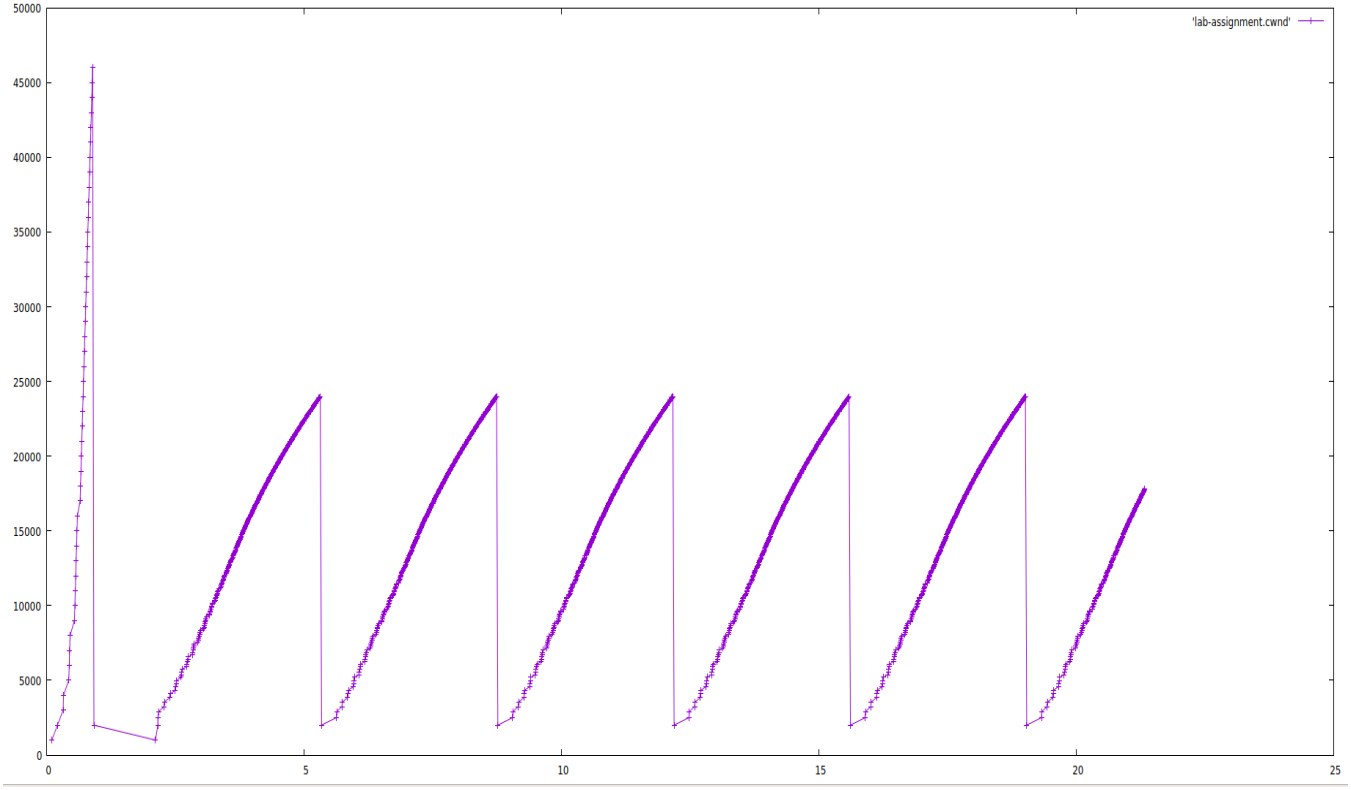
Αν συγκρίνουμε αυτό το throughput με αυτό που προκύπτει στην προσομοίωση με default τιμές βλέπουμε ότι είναι περίπου 10 φορές μεγαλύτερο από ότι το αρχικό. Αυτά τα δύο παραδείγματα μας κάνουν ξεκάθαρο ότι το throughput εξαρτάται σε μεγάλο βαθμό από το bottleneck bandwidth (σχεδόν γραμμική εξάρτηση).

Απάντηση (1.3.α)

Tcp Vegas



TcpWest Wood



Απάντηση (1.3.b)

Αποτελέσματα προσομοίωσης με default παραμέτρους και πρωτόκολλο TCP Vegas :

```
savvas@savvas-VirtualBox:~/workspace/bake/source/ns-3.29$ ./waf --run lab
Waf: Entering directory `/home/savvas/workspace/bake/source/ns-3.29/build'
[2694/2766] Compiling scratch/lab.cc
[2725/2766] Linking build/scratch/lab
Waf: Leaving directory `/home/savvas/workspace/bake/source/ns-3.29/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (38.181s)
queuesize=10, delayRB=35
A's address: 10.0.0.1
B's address: 10.0.1.2
R's #1 address: 10.0.0.2
R's #2 address: 10.0.1.1
Total Bytes Received from B: 2441000
savvas@savvas-VirtualBox:~/workspace/bake/source/ns-3.29$
```

Το throughput που προκύπτει είναι :

$$Throughput = \frac{2441000}{20} = 122050\ Bps = 0.12\ MBps$$

Αποτελέσματα προσομοίωσης με default παραμέτρους και πρωτόκολλο TCP West Wood :

```
savvas@savvas-VirtualBox:~/workspace/bake/source/ns-3.29$ ./waf --run lab
Waf: Entering directory `/home/savvas/workspace/bake/source/ns-3.29/build'
[2694/2766] Compiling scratch/lab.cc
[2725/2766] Linking build/scratch/lab
Waf: Leaving directory `/home/savvas/workspace/bake/source/ns-3.29/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (38.804s)
queuesize=10, delayRB=35
A's address: 10.0.0.1
B's address: 10.0.1.2
R's #1 address: 10.0.0.2
R's #2 address: 10.0.1.1
Total Bytes Received from B: 1804000
savvas@savvas-VirtualBox:~/workspace/bake/source/ns-3.29$
```

Το throughput που προκύπτει είναι :

$$Throughput = \frac{1804000}{20} = 90200\ Bps = 0.90\ KBps$$

Άσκηση 2

Ζητούμενα

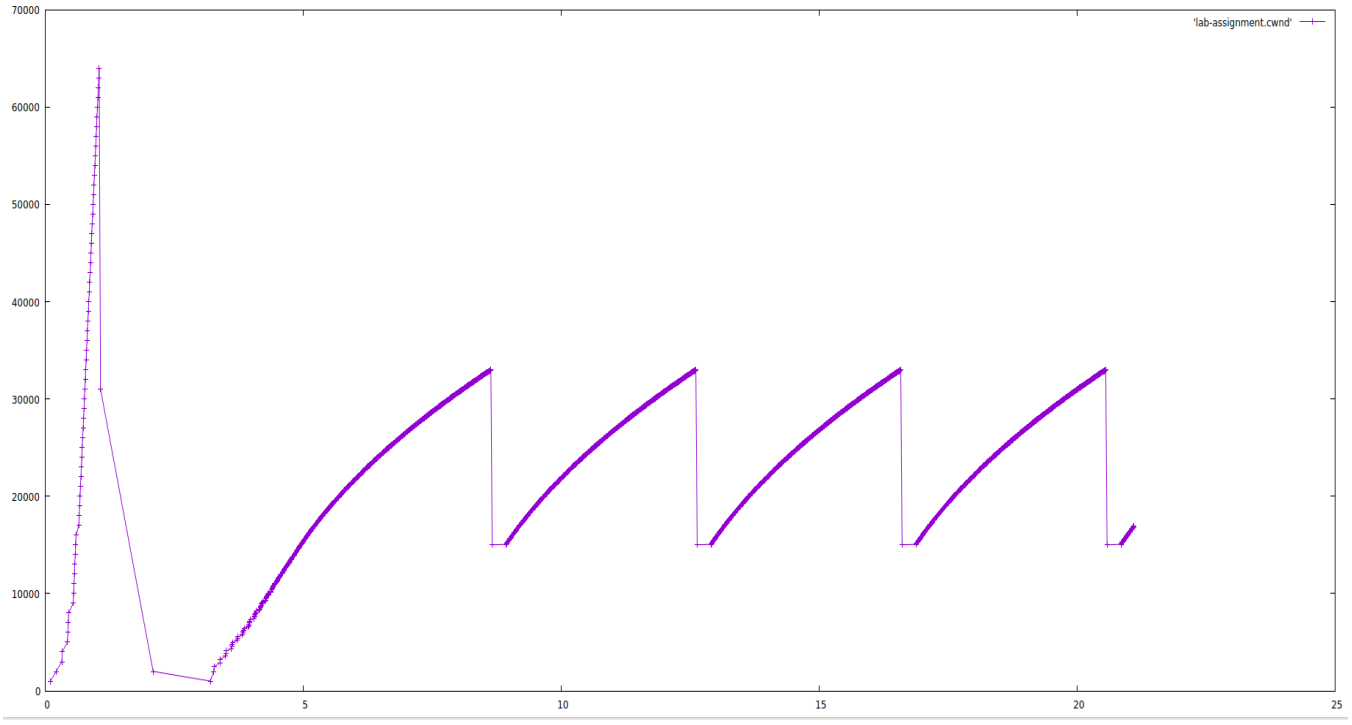
Θεωρήστε και πάλι τις default παραμέτρους της τοπολογίας.

1. Ορίστε το μέγεθος της ουράς ίσο με (18 + z/2). Οι υπόλοιπες παράμετροι είναι ίσες με τις default τιμές τους.
    a. Κάντε plot το διάγραμμα του παραθύρου συμφόρησης (cwnd) ως προς τον χρόνο.
        - Τι αλλαγές παρατηρείτε ως προς το cwnd;
        - Για ποιο λόγο συμβαίνουν αυτές οι αλλαγές;
    b. Υπολογίστε το συνολικό throughput της προσομοίωσης και συγκρίνετέ το με το αποτέλεσμα του ερωτήματος 1.1.b.
2. Ορίστε ως πολιτική διαχείρισης ουράς τη RED1 . Όσον αφορά τις παραμέτρους της RED minimum_threshold, maximum_threshold και queue_weight, χρησιμοποιήστε τις ενδεδειγμένες τιμές από τη θεωρία.
    a. Κάντε plot το διάγραμμα του παραθύρου συμφόρησης (cwnd) ως προς τον χρόνο.
    b. Υπολογίστε το συνολικό throughput της προσομοίωσης και συγκρίνετέ το με το ερώτημα 2.1.b.

Απαντήσεις

Απάντηση (2.1.α)

Από το ΑΕΜ 57403 προκύπτει ότι z=3 οπότε το μέγεθος ουράς θα είναι 19.5 και επειδή πρέπει να είναι ακέραιος 19.



Σε αυτό το διάγραμμα παρατηρούμε, αρχικά, ότι κατά την διάρκεια του πρώτου slow start η μέγεθος του congestion window καταλήγει να είναι μεγαλύτερο από αυτό του cwnd του ερωτήματος (1.1.α). Πάνω σε αυτό παρατηρούμε ότι και τα «πτερύγια» που σχηματίζονται φτάνουν σε μεγαλύτερα τοπικά μέγιστα από ότι αυτά του (1.1.α) και είναι και λιγότερα. Αυτά τα στοιχεία οφείλονται στο γεγονός ότι με την αύξηση της ουράς υπάρχει μεγαλύτερος χώρος ωστέ να αποθηκευτούν περισσότερα πακέτα. Αφού υπάρχει περισσότερος χώρος για πακέτα, αυτό σημαίνει ότι η εμφάνιση συμφόρησης καθυστερεί καθώς ο ρουτερ δεν τα απορριπτει μέχρι να καλυφτούν όλες οι θέσεις της ουράς. Αφού αργεί να εμφανιστεί συμφόρηση το congestion window συνεχίζει να αυξάνεται και φτάνει σε μεγαλύτερες τιμές από ότι αυτό του ερωτήματος (1.1.α).

Εκτός αυτού, με την αύξηση του μεγέθους της ουράς, καθώς γεμίζει η ουρά κάποιο πακέτο που βρίσκεται στις τελευταίες θέσεις θα αργήσει περισσότερο (από ότι στο πρώτο διάγραμμα) να αποσταλλεί στον παραλλήπτη, από τον ρούτερ. Εξαιτίας αυτού του φαινομένου παρατηρούμεότι τα «πτερύγια» εμφανίζουν μεγαλύτερη κυρτότητα από αυτά του πρώτου διαγράμματος

Απάντηση (2.1.b)

Αποτέλεσμα προσομοίωσης με μέγεθος ουράς 19 :

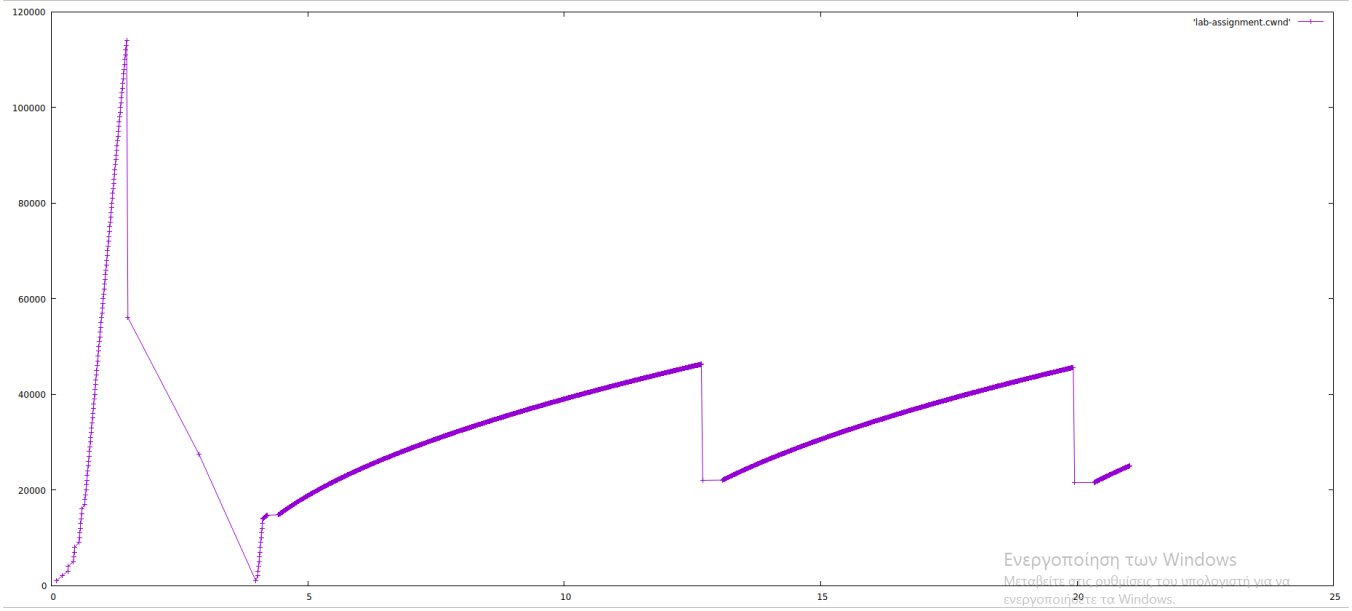
```
savvas@savvas-VirtualBox:~/workspace/bake/source/ns-3.29$ ./waf --run lab
Waf: Entering directory `/home/savvas/workspace/bake/source/ns-3.29/build'
[2694/2766] Compiling scratch/lab.cc
[2725/2766] Linking build/scratch/lab
Waf: Leaving directory `/home/savvas/workspace/bake/source/ns-3.29/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (39.233s)
queuesize=19, delayRB=35
A's address: 10.0.0.1
B's address: 10.0.1.2
R's #1 address: 10.0.0.2
R's #2 address: 10.0.1.1
Total Bytes Received from B: 2177000
savvas@savvas-VirtualBox:~/workspace/bake/source/ns-3.29$
```

Οπότε το throughput που προκύπτει είναι :

$$Throughput = \frac{2177000}{20} = 108850\ Bps = 108.9\ KBps$$

Το συνολικό throughput που προκύπτει από αυτή την προσομοίωση είναι περίπου 2 KBps μεγαλύτερο από αυτό του ερωτήματος 1.1.b . Το συγκεκριμένο throughput είναι ελαφρώς μεγαλύτερο καθώς, όπως έχει προαναφερθεί, με την αύξηση της ουράς εμφανίζεται σπανιότερα συμφόρηση οπότε το δίκτυο δεν χρειάζεται να μειώνει τόσο συχνά το παράθυρο συμφόρησης στο μισό και να ξανά ξεκινάει. Ωστόσο δεν υπάρχει μεγάλη βελτίωση στο throughput γιατί ενώ γίνεται, μεν, σπανιότερα συμφόρηση, τα πακέτα περιμένουν περισσότερη ώρα στην ουρά ρίχνοντας έτσι λίγο το συνολικό throughput.

Απάντηση (2.2.a)



Απάντηση (2.2.b)

Αποτέλεσματα προσομοίωσης με μέγεθος ουράς 19 και πολιτική διαχείρησης ουράς :

```
savvas@savvas-VirtualBox:~/workspace/bake/source/ns-3.29$ ./waf --run lab
Waf: Entering directory `/home/savvas/workspace/bake/source/ns-3.29/build'
[2694/2768] Compiling scratch/lab.cc
[2727/2768] Linking build/scratch/lab
Waf: Leaving directory `/home/savvas/workspace/bake/source/ns-3.29/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (1m15.389s)
queuesize=19, delayRB=35
A's address: 10.0.0.1
B's address: 10.0.1.2
R's #1 address: 10.0.0.2
R's #2 address: 10.0.1.1
Total Bytes Received from B: 2293000
savvas@savvas-VirtualBox:~/workspace/bake/source/ns-3.29$
```

Οπότε το throughput που προκύπτει είναι :

$$Throughput \ = \frac{2293000}{20} = 114650 \ Bps = 114.65 \ KBps$$

Συγκρινοντας αυτό το throughput με το throughput του ερωτήματος 2.1.b παρατηρούμε ότι είναι μεγαλύτερο κατά 6 KBps περίπου. Αυτή η διαφορά δεν είναι τρομακτικά μεγαλή ούτε όμως μπορεί να θεωρηθεί αμελητέα. Από αυτή την διαφορά, όμως, μπορούμε να καταλάβουμε ότι η επιλογή μίας καλής πολιτικής διαχείρησης ουράς μπορεί να βελτιώσει το througput του δικτύου μας.