Ονοματεπώνυμο: Πυλιώτης Αθανάσιος	**Όνομα PC:** DESKTOP-5DLG3IF
Ομάδα: 1	**Ημερομηνία:** 15/03/23

Εργαστηριακή Άσκηση 3
Τοπικά δίκτυα και μεταγωγείς LAN

Απαντήστε στα ερωτήματα στον χώρο που σας δίνεται παρακάτω και στην πίσω σελίδα εάν δεν επαρκεί. Το φυλλάδιο αυτό θα παραδοθεί στον επιβλέποντα.

1

1.1 **ifconfig em0 192.168.1.1 netmask 255.255.255.0** → PC1

ifconfig em0 192.168.1.2 netmask 255.255.255.0 → PC2

1.2 ifconfig em0 up

ifconfig em1 up

1.3 PC1: **ping 192.168.1.2** , PC2: **ping 192.168.1.1**

Παρατηρούμε πως δεν συμβαίνει τίποτα, σαν να μην υπάρχει μηχάνημα με αντίστοιχη IPv4 διεύθυνση. Είναι λογικό να συμβαίνει αυτό καθώς τα 2 VMs βρίσκονται σε διαφορετικά υποδίκτυα LAN (LAN1 και LAN2 συγκεκριμένα) και δεν είναι δυνατό να το δει, παρότι βρίσκονται στο ίδιο υποδίκτυο με βάση IPv4.

1.4 tcpdump -vvv -i em0 icmp

tcpdump -vvv -i em1 icmp

Παρατηρούμε τη προσπάθεια των 2 VMs ΄καθώς στέλνουν ARP Request πακέτα στο LAN τους προσπαθώντας να βρουν την IPv4 διεύθυνση που pingάρουν. Προφανώς δεν τα καταφέρνουν.

1.5 ifconfig bridge0 create → ifconfig bridge0 addm em0 addm em1 up

1.6 Ναι υπάρχει επικοινωνία!

1.7 Το PC2 από το PC1 δεν απέχει καθόλου, βρίσκεται στο εσωτερικό δίκτυο του υπολογιστή μου (TTL = 64, το οποίο είναι η μέγιστη τιμή. Είναι 0 hops μακριά)

1.8 Παρατηρούμε πως έχουν αποθηκευτεί και είναι ίδες στα πρώτα 3 bytes. Λειτουργεί όπως θα περιμέναμε τον arp πίνακα να λειτουργεί.

1.9 B1: tcpdump -vvv -e -i em0 icmp

B1: tcmpdump -vvv -e -i em1 icmp

Μπορούμε να δούμε και τα echo request και τα echo reply και στις δύο διεπαφές στα τοπικά δίκτυα.

1.10 Όχι δεν φαίνεται να κάνει καμία αλλαγή στις IPv4 ή MAC. Παρατηρούμε συγκεκριμένα πως οι δύο καταγραφές είναι ίδιες με εξαίρεση τον χρόνο.

1.11 Το μόνο πεδίο που αλλάζει είναι ο χρόνος αποστολής, άρα κανένα πεδίο του πακέτου.

1.12 Όχι δεν υπάρχει, είναι σαν να μην υπήρξε. Αυτό πιθανότατα συμβαίνει επειδή η bridge0 στέλνει απευθείας τα πακέτα που λαμβάνει από την em0 στην em1 με αποτέλεσμα να μην γίνεται αντιληπτή η ύπαρξη της. Επίσης, είναι λογικό να μην γίνεται εφόσον είναι απλά μεσάζοντας που προωθεί τα πακέτα εκεί που έχουμε ορίσει και δεν προκαλεί άλλες αλλαγές, δεν είναι δρομολογητής που μειώνει TTL και στέλνει μηνύματα σφάλματος.

1.13 PC1: ping 192.168.1.2 → B1: tcpdump -vvv -en -i em1 icmp

1.14 PC2: ifconfig em0 192.168.2.1 netmask 255.255.255.0

Όχι το ping αποτυγχάνει. Συνεχίζει ωστόσο η γέφυρα να προωθεί τα ICMP Πακέτα παρότι έχει αλλάξει το

υποδίκτυο, που μας δείχνει πως η γέφυρα προωθεί στο υποδίκτυο ανεξαρτήτως IPv4.

1.15 όχι, καθώς δεν υπάρχει διεπαφή με την IPv4 διεύθυνση που ζητάει το PC1. Προωθούνται τα πακέτα αλλά δεν αναγνωρίζεται καθόλου από το PC2 πως του κάνουν ping. Έχουν διαφορετική dst IPv4 και για αυτό το λόγο απλά απορρίπτονται και δεν συνεχίζουν να στέλνονται ICMP πακέτα.

1.16 PC3: ifconfig em0 192.168.1.3 netmask 255.255.255.0 → ping 192.168.1.1 → Nope, δεν μπορούμε αφού είναι σε διαφορετικά LAN.

1.17 B1: ifconfig em2 up → B1: ifconfig bridge0 addm em2

1.18 PC3: ping 192.168.1.1 → Yep, τώρα λαμβάνουμε απάντηση. Πέτυχε η σύνδεση με τη γέφυρα!

1.19 B1: tcpdump -vvv -i em1 icmp → PC3: ping 192.168.1.1 ή PC1: ping 192.168.1.3

Όχι δεν λαμβάνονται πακέτα στο LAN2. Πιθανότατα αυτό να συμβαίνει επειδή δεν ζητείται η IPv4 στην οποία προωθεί. Φαίνεται να γνωρίζει που ακριβώς να προωθήσει το πακέτο για να το λάβει το κατάλληλο VM, ανάλογα με το αν είναι το PC1 ή το PC3.

1.20 tcpdump -vvv -e -i em1

Λαμβάνουμε το πακέτο ARP Request καθώς γίνεται broadcasted σε όλα τα δίκτυα που μπορεί να βρει ώστε να βρει ποιος έχει IPv4 192.168.1.3 και ποια είναι η MAC του. Το κατέγραψε λόγω του broadcast.

1.21 Διεπαφές Μέλη → B1: ifconfig bridge0

1.22 Πίνακας προώθησης → B1: ifconfig bridge0 addr

1.23 Στα μηχανήματα em0 → PC1, em1 → PC2, em2 → PC3

1.24 Διαγραφή εγγραφών του πίνακα προώθησης → B1: ifconfig bridge0 flush

1.25 B1: ifconfig bridge0 deletem em2

1.26 B1: ifconfig bridge0 destroy

1.27 Αφαίρεση IPv4 από συσκευές → PC1,PC2,PC3: ifconfig em0 delete

2

2.1 PCx: ifconfig em0 192.168.1.x netmask 255.255.255.0, όπου x = {1,2,3,4} για να ορίσουμε IPv4 διευθύνσεις στα δίκτυα μας

2.2 B1: ifconfig em0 up → ifconfig bridge1 create

ifconfig bridge1 addm em0 addm em1 up (έχουμε ορίσει κατάλληλα στο VM LAN1 και LNK1 στις em0 και em1 αντίστοιχα)

2.3 B2: ifconfig em0 up → ifconfig bridge2 create

ifconfig bridge2 addm em0 addm em1 up (έχουμε ορίσει κατάλληλα στο VM LNK1 και LNK2 στις em0 και em1 αντίστοιχα)

2.4 B3: ifconfig em0 up → ifconfig bridge3 create

ifconfig bridge3 addm em0 addm em1 up (έχουμε ορίσει κατάλληλα στο VM LNK2 και LAN2 στις em0 και em1 αντίστοιχα)

2.5 PC1: 08:00:27:01:e3:c8

PC2: 08:00:27:1b:6f:7a

PC3: 08:00:27:8a:b5:e1

PC4: 08:00:27:f6:7c:ce

arp -ad → σε όλα

2.6 B1(2,3): ifconfig bridge1(2,3) flush

2.7 tcpdump -vvv -en icmp -i em0

2.8 B1: 08:00:27:1b:6f:7a (PC2) vlan1 em1 1190 flash=0<>

08:00:27:01:e3:c8 (PC1) vlan1 em0 1190 flasg=0<>

B2: 08:00:27:1b:6f:7a (PC2) vlan1 em1 1195 flash=0<>

08:00:27:01:e3:c8 (PC1) vlan1 em0 1195 flasg=0<>

B3: 08:00:27:01:e3:c8 (PC1) vlan1 em0 1163 flasg=0<>

2.9 Αρχικά το PC1 στέλνει ARP Request με broadcast, δηλαδή μεταφέρεται σε όλα τα LAN. Για αυτό το λόγω σε όλες τις γέφυρες (το πακέτο προωθείται με flooding στο LNK2) υπάρχει η MAC του PC1, καθώς την αποθηκεύουν μετά το ARP Request. Αντίστοιχα τα B2,B3 βλέπουν από τα LNK1,LNK2 το ARP Request. Παράλληλα, έρχεται απάντηση από το PC2 που έχει την IP που ζητείται. Απαντάει και στέλνει ARP Reply. Λόγω αυτού, τα B1 και B2 βλέπουν το Reply στο LNK1 και για αυτό βλέπουμε πως τα B1,B2 έχουν καταγράψει και τη MAC του PC2. Επειδή έχουν καταγράψει τη MAC του PC1, ξέρουν που να προωθήσουν το πακέτο κατάλληλα. Και στη συνέχεια στέλνονται και τα ICMP request/reply όπως περιμέναμε μεταξύ PC1/PC2.

2.10 Οι B1,B2 πίνακες παραμένουν ίδιοι ενώ στο B3 βρέθηκε μονάχα παραπάνω η MAC από το PC2 em0. Τα μηνύματα προωθούνται σωστά όπως αναφέρθηκε παραπάνω, ενώ το B3 απλά βλέπει το ARP Request και το προωθεί στο PC4.

2.11 Ναι την περιέχει καθώς το ARP Request γίνεται broadcasted και στέλνεται σε όλα τα PC και τα bridges. Έχουμε φτιάξει εσωτερικό δίκτυο με γέφυρες που ενώνει όλα LAN με αποτέλεσμα κάθε ARP Request να στέλνεται σε όλα τα PC και τα B, και να αποθηκεύονται στους πίνακες προώθησης. Την έχει τη διεύθυνση επειδή ο PC2 στέλνει ARP Request και ψάχνει την MAC του PC4. Ο PC4 παίρνει το πακέτο και το προωθεί στη συνέχεια πίσω στον PC2 μέσω του B3/B2. Στέλνει το πακέτο προς το LNK2 συγκεκριμένα που γνωρίζει πως είναι ο PC2. Ο B2 στη συνέχεια στέλνει προς τον LNK1 που γνωρίζει πως είναι ο PC2 και ο B1 απλά καταγράφει τη MAC PC4 em1 χωρίς να τη προωθεί μετά, καθώς ξέρει πως ο PC2 είναι LNK1/em1.

2.12 Καθώς το PC3 κάνει ARP Request για τη MAC του PC2, η MAC PC3 προσθέτεται στους πίνακες τους στη κατάλληλη διεπαφή επειδή το πακέτο στέλνεται σε όλα τα LAN. Επίσης στη συνέχεια ανανεώνεται ο χρόνος για τη MAC του PC2 αφού εκείνο απαντά και το ARP Reply θα περάσει και από το LNK1, και από το LNK2.

2.13 PC1: ping 192.168.1.2

PC4: ping 192.168.1.2 → λειτουργούν κανονικά.

2.14 Παρατηρούμε πως ο time είναι μικρότερος κατά 6ms περίπου, που δείχνει τη διαφορά στην αλλαγή του εσωτερικού δικτύου. Υπάρχει αρχικά βέβαια ένας χρόνος που δεν στέλνονται πακέτα. (από 8-10 πήγε 3-4ms). Το ping ακόμα επιτυγχάνει φυσικά, αφού βρίσκονται στο ίδιο LAN και λαμβάνει κανονικά τα ICMP Request που στέλνει ο PC4.

2.15 Το ping αποτυγχάνει αφού πριν την αλλαγή όλες οι γέφυρες προωθούσαν τα πακέτα για PC2 στο LNK1 και τώρα δεν θα το βρίσκουν. Στη συνέχεια ωστόσο, όταν αυτό γίνει αντιληπτό λόγω της έλλειψης απάντησης, θα ανανεωθούν οι ARP πίνακες.

2.16 Το ping του PC1 στο PC2 ήταν επιτυχές καθώς τώρα όλες οι γέφυρες έμαθαν τη νέα διεύθυνση MAC του PC2 μετά το ping που έστειλε στο PC3 στο LNK2, άρα όλες ξέρουν πως ο PC2 είναι στο LAN2 πλέον και έτσι τον βρίσκουν.

2.17 Αν δεν κάναμε ping ή κάποια εντολή που να έστελνε ARP Request, θα έπρεπε να περιμένουμε μέχρι να λήξουν οι διευθύνσεις στον πίνακα ARP των γεφυρών και μετά θα τον έψαχναν κανονικά τον PC2, ο χρόνος θα ήταν περίπου 2000 sec. Άλλη περίπτωση θα ήταν να λήξει ο ARP πίνακας στο PC1 για να απαιτηθεί να σταλθεί ARP Request και να βρεθεί το PC2.

3

3.1 Αρχικά έχουμε ορίσει τις διεπαγές στις γέφυρες ως εξής: em0 → LAN1 , em1 → LNK1, em2 → LNK2 , Έτσι λοιπόν ορίζουμε ως εξής:

B1: ifconfig bridge1 create

ifconfig bridge1 addm em0 addm em1 up

3.2 Αρχικά έχουμε ορίσει τις διεπαγές στις γέφυρες ως εξής: em0 → LAN2 , em1 → LNK1, em2 → LNK2 , Έτσι λοιπόν ορίζουμε ως εξής:

B2: ifconfig bridge2 create

ifconfig bridge2 addm em0 addm em1 up

3.3 PC1: 08:00:27:01:e3:c8

PC2: 08:00:27:1b:6f:7a

PC3: 08:00:27:8a:b5:e1

PCx: arp -ad

3.4 tcpdump -vvv -en -i em0

PC3: Ping -c 4 192.168.1.2

Ναι, εμφανίζεται ένα ARP Request. Αυτό συμβαίνει επειδή με τις γέφυρες τα πάντα προωθούνται άμεσα και αυτές είναι διαφανείς, σαν να μην υπάρχουν δηλαδή, οπότε το ARP Request προωθείται παντού.

3.5 PC3: ping 192.168.1.1

3.6 B1: ifconfig bridge1 addm em2 up

B2: ifconfig bridge2 addm em2 up

3.7 B1: ifconfig bridge1 addr

B2: ifconfig bridge2 addr

Υπάρχουν εγγραφές για όλα τα μηχανήματα.

Παρατηρούμε πως έχουν πολλές καταγραφές από MAC διευθύνσεις. Συγκεκριμένα παρατηρούμε τις 2 πάνω που είναι ίδιες και στις 2 γέφυρες και έχουν και τον ίδιο αριθμό. Πιθανότατα αυτές να είναι οι MAC διευθύνσεις μέσω τον οποίων στέλνεται ξανά και ξανά το πακέτο του ping. Παράλληλα το tcpdump πήγαινε σαν τρελό και το ping καθυστερούσε απίστευτα πολύ. Οι διεπαφές στις οποίες αυτά εμφανίζονται είανι em1, em2 στο bridge1 και em1, em1 στο bridge2.

3.8 Στο B1 εμφανίζονται η MAC του PC1 στην διεπαφή em0 (LAN1) και MAC του PC3 στην διεπαφή em1 (LNK1)

Στο B2 εμφανίζονται η MAC του PC1 στην διεπαφή em1 (LNK1) και MAC του PC3 στην διεπαφή em0 (LAN2)

3.9 tcpdump -vvv -en -i em0 και στα 2

3.10 arp -d -a

Όχι το ping δεν ήταν επιτυχές επειδή δεν καταφέρνει να λάβει απάντηση σε αυτόν τον κατακλυσμό μηνυμάτων.

3.11 Ναι έγινε κατακλυσμός.

Η MAC του PC1 εμφανίζεται στην διεπαφή em1 (LNK1) και του PC3 στην em1 (LNK1) (και άλλες στην em2 (LNK2)), επειδή έγινε κατακλυσμός του μηνύματος λόγο του κλειστού βρόγχου και η τελευταία φορά που το έλαβε το ARP Request ήταν από τη διεπαφή em1 (ή em2) . Συνεπώς η γέφυρα εκπαιδεύτηκε πως εκεί θα πρέπει να προωθεί τα πακέτα του PC3. Ανάλογα με το πότε το σταματήσουμε, εκείνη τη κατάσταση διατηρεί μετά από λίγες δοκιμές.

3.12 Η ερώτηση που γίνεται είναι Request who-has 192.168.1.1 tell 192.168.1.3 και η απάντηση Reply 192.168.1.1 is-at 08:00:27:01:e3:c8 (MAC του PC1)

3.13 Παρατηρούμε και εδώ τον κατακλυσμό και όλα τα ARP Request έρχονται από τη PC3 (τη MAC PC3). Συνεπώς οι γέφυρες δεν επηρεάζουν καθόλου το μήνυμα και δεν μας απασχολεί που στέλνεται από αυτές εν τέλει, γιατί αυτές είναι όντως διάφανες και απλά προωθούν.

3.14 Λόγω του κατακλυσμού. Μεταξύ των 2 γεφυρών υπάρχουν 2 links, οπότε, αφού το ARP Request γίνεται broadcasted, θα σταλεί κάθε φορά από κάθε διεπαφή. Συνεπώς θα στέλνεται επ'αόριστον, επειδή δεν θα λάβει ποτέ απάντηση. Παράλληλα στέλνεται άπειρες φορές και στα LAN1, LAN2.

3.15 Λόγω του ατέρμονου βρόγχου. Το PC3 αλλάζει θέσεις μία στο LNK1 και μία στο LNK2 όπως είδαμε παραπάνω κι έτσι δεν ξέρει πως βρίσκεται στο LAN2 και το ARP Reply δεν φτάνει ποτέ στον προορισμό του.

4

4.1 B1: ifconfig bridge1 destroy → ifconfig bridge1 create

B2: ifconfig bridge2 destroy → ifconfig bridge2 create

ifconfig emx down

4.2 ifconfig emx up

B1: ifconfig lagg0 create

4.3 B1: ifconfig lagg0 up laggport em1 → ifconfig lagg0 up laggport em2

4.4 B2: ifconfig emx up → ifconfig lagg0 create → ifconfig lagg0 up laggport em1 → ifocnfig lagg0 up laggport em2

4.5 B1: ifconfig bridge1 addm em0 addm lagg0 up

4.6 B2: ifconfig bridge2 addm em0 addm lagg0 up

4.7 εμφανίζεται και με τα δύο ένα ARP Request όπως περιμέναμε στο LAN1 όταν κάνουμε tcpdump. Το πακέτο φαίνεται πως ξεκίνησε από το σωστό PC και προωθείται πρώτα από το B2 στο B1 και στη συνέχεια από το B1 στο LAN1.

4.8 tcpdump -vvv -i em0 -en

4.9 ναι, το ping ήταν επιτυχές και λάβαμε κανονικά όσα πακέτα αναμέναμε, το ARP Request και το ARP Reply, ένα από το καθένα, με τις σωστές MAC Addresses (το request για τη MAC του PC1 και το Reply για τη MAC του PC3).

4.10 B1: tcpdump -vvv -en -i em1

B2: tcpdump -vvv -en -i em2

PC2: ping 192.168.1.1

Τα πακέτα ICMP εμφανίζονται στη ζεύξη LNK1 μόνο. Αυτό συμβαίνει επειδή ως default εφαρμόζεται το πρωτόκολλο του failover, το οποίο κλείνει τη μία δίοδο, την LNK2, εφόσον δεν υπάρχει πρόβλημα σύγκρουσης και για να μην υπάρξει βρόγχος στα πακέτα όπως προηγουμένως. Επίσης, διατηρείται σε περίπτωση που κλείσει η LNK1 ως backup και η LNK1 θεωρείται η primary διεπαφή.

4.11 Αφότου αποσυνδέσαμε τα καλώδια, η ζεύξη με την οποία μεταφέρονται τα ICMP πακέτα είναι η LNK2. Όπως αναφέρθηκε παραπάνω, εφόσον πλέον δεν υπάρχει η LNK1, το failover protocol στέλνει τα πακέτα στην LNK2 απευθείας, ώστε να μην υπάρχει πρόβλημα μεταφοράς. Χρειάστηκε μονάχα λίγος χρόνος πριν να γίνει επιτυχώς η αλλαγή.

4.12 Ναι άλλαξε, εφόσον έχουμε primary και backup διεπαφή, επαναφέρεται στη primary διεπαφή την LNK1 όπως παραπάνω.

5

5.1 B1: ifconfig bridge1 destroy, ifconfig lagg0 destroy, ifconfig em0 down, ifconfig em1 down, ifconfig em2 down

B2: ifconfig bridge2 destroy, ifconfig lagg0 destroy, ifconfig em0 down, ifconfig em1 down, ifconfig em2 down

5.2 B1: ifconfig bridge1 create, ifconfig bridge1 addm em0 addm em1 addm em2 up,

ifconfig em0 up, ifconfig em1 up, ifconfig em2 up

5.3 B1: ifconfig bridge2 create, ifconfig bridge2 addm em0 addm em1 addm em2 up,

ifconfig em0 up, ifconfig em1 up, ifconfig em2 up

5.4 B1: ifconfig bridge1 stp em0 stp em1 stp em2

5.5 B2: ifconfig bridge2 stp em0 stp em1 stp em2

5.6 B1: ifconfig bridge1: Bridge1 id → 08:00:27:2e:32:a0 priority 32768

B2: ifconfig bridge2: Bridge2 id → 08:00:27:48:d4:b6 priority 32768

5.7 Το βλέπουμε από το root id το οποίο δείχνει το id της bridge1, άρα αυτή είναι η ρίζα.

5.8 B1: em0: role designated status forwarding

em1: role designated status forwarding

em2: role designated status forwarding

5.9 Η ριζική θύρα είναι η em1 που αντιστοιχεί στην LNK1 που λέει role root state forwarding

5.10 Η θύρα em2 είναι role alternate state discarding, που δείχνει πως απορρίπτει ό,τι πακέτο λάβει. Έχει τον ρόλο backup λογικά. (LNK2)

5.11 em0: role designated state forwarding (LAN2)

5.12 B1: tcpdump -vvv -en -i em0 , BPDU στέλνονται κάθε 2 second όπως δείχνει το hello-time.

5.13 Χρησιμοποιεί ενθυλάκωση 802.3, το γράφει στην επικεφαλίδα.

5.14 Source MAC: 08:00:27:f4:1a:0d

Destination MAC 01:80:c2:00:00:00

5.15 Ανήκει στην διεπαφή em0 (προς το LAN1) της γέφυρας ρίζας B1

5.16 Η MAC προορισμού των BPDU είναι multicast (πρώτο bit 1, LSB των MSB)

5.17 root ID: 8000.08:00:27:2e:32:a0

Bridge ID: 8000.08:00:27:2e:32:a0.8001

Root pathcost: 0

5.18 Το bridge ID εδώ είναι 8000.08:00:27:2e:32:a0.8002, άρα η προτεραιότητα δείχνεται από το 8000 = 32768 αφού έχουν την ίδια προτεραιότητα όλα.

5.19 Το δεύτερο μέρος είναι το bridge1 id, ή η MAC της γέφυρας και το τρίτο μέρος είναι αναγνωριστικό

για την θύρα από την οποία παράγεται το BPDU, ίσως και μια τελική προτεραιότητα που δίνεται αυτόματα από τη γέφυρα.

5.20 Όχι δεν παρατηρούμε κίνηση με BPDU από το B2 στα LN1, LNK2.

5.21 Στην διεπαφή em0 που για εμένα αντιστοιχεί στο LAN2 στο B2

5.22 root ID: 8000.08:00:27:2e:32:a0

Bridge ID: 8000.08:00:27:48:d4:b6.8001

Root pathcost: 20000

5.23 PC1: ping 192.168.1.2

Ναι είναι επιτυχές.

5.24 Παρατηρούμε πως αν κόψουμε το καλώδιο em1 στην B1 που αντιστοιχεί στο LNK1, η σύνδεση διακόπτεται για περίπου 9 second. Είναι λογική η τιμή αυτή αφού για την αποκατάσταση του δέντρου και την αλλαγή στην τοπολογία του η rstp παίρνει περίπου 3*hello-time second, δηλαδή 6 seconds

5.25 Υπάρχει μικρότερη και σχεδόν ανεπαίσθητη διακοπή, κατά τη διάρκεια της οποίας αποτυγχάνει μονάχα ένα εκ των pings.

6

6.1 B1: ifconfig em3 up

ifconfig bridge1 addm em3 up

ifconfig bridge1 stp em3

6.2 B2: ifconfig em3 up

ifconfig bridge2 addm em3 up

ifconfig bridge2 stp em3

6.3 B3: ifconfig bridge3 create

ifconfig em0 up, ifconfig em1 up, ifconfig em2 up

ifconfig bridge3 addm em0 addm em1 addm em2 up

ifconfig bridge3 stp em0 stp em1 stp em2

6.4 B1,B2,B3: arp -d -a

Ναι είναι επιτυχές το ping από PC1 σε PC2, PC3

6.5 Αφού η γέφυρα με το μικρότερο priority γίνεται ρίζα, τότε απλά θα ορίσουμε ένα μικρότερο priority (και θα βάλουμε ένα πολλαπλάσιο του 16), δηλαδή B1: ifconfig bridge1 priority 28960 (που αντιστοιχεί στο 28672)

6.6 Το pathcost είναι 20000=20Tbit/bandwidth και στις 3 ζεύξεις. Τα interfaces έχουν ταχύτητα 1Gbps (1000baseT), το οποίο στο rstp είναι 20Tbit/bandwidth = 20000

6.7

B1: tcpdump -vvv -e -i em1 → pathcost στο bridge3 of BPDU from LNK3 = 0 (γιατί διαφημίζει 0 η ρίζα)

B2: Tcpdump -vvv -e -i em2 → pathcost στο bridge3 of BPDU from LNK4 = 20000 προσθέτει το κόστος για B2 → B1

6.8 Ριζική θύρα στο bridge3 είναι η em1 που αντιστοιχεί στην LNK3, επειδή μέσα από αυτή πηγαίνει απευθείας στη ρίζα χωρίς κόστος, ενώ μέσω της LNK4 θα έπρεπε να κάνει διαδρομή B3 → B2 → B1

6.9 Στο B3 η em2 (LNK4) είναι role alternate state discarding

Στο B2 η em3 (LNK4) είναι role designated state forwarding

6.10 B3: στα πλαίσια BPDU που παράγει στο em0 (LAN3) το root-pathcost = 20000 = 20Tbit/bandwidth

6.11 ping 192.168.1.3

6.12 ifconfig bridge3 ifpathcost em1 40004 . Η διαδρομή προς τη ρίζα έχει κόστος 20000 μέσω LNK3 αλλά μέσω LNK4 έχει 2 φορές αυτό, οπότε θέλουμε τιμή μεγαλύτερην από 40000.

6.13 πέρασαν περίπου 5-6 seconds (3*hello-time = 6 seconds) μέχρι να αποκατασταθεί η επικοινωνία καθώς χάθηκαν μόνο 2 pings

6.14 B3: em1 (LNK3) role alternate state discarding

B2: em3 (LNK4) role designated state forwarding

6.15 Όχι δεν υπήρξε διαφορά στις τιμές των παραμέτρων της BPDU που λαμβάνει η bridge3

6.16 Ναι, σε όσα παράγει μεταβλήθηκε το root-pathcost από 20000 σε 40000

6.17 Πέρασαν περίπου 10 second μετά την αποσύνδεση του καλωδίου της LNK4 από τη bridge2

6.18 Πέρασαν περίπου 7 second μετά την επανασύνδεση του καλωδίου της LNK4 από τη bridge2

6.19 em0 (LAN3): role designated state forwarding

em3 (LAN3 – 2): role backup state discarding

6.20 ifconfig bridge3 ifpathcost em1 20000. Το κόστος για τη ρίζα μέσω της em1 (LNK3) είναι 20000 ενώ μέσω της em2 (LNK4) είναι 40000, οπότε η LNK3 είναι η root ξανά.

7

7.1 PC1: ifconfig vlan5 create → ifconfig vlan5 vlan 5 vlandev em0 → ifconfig vlan5 inet 192.168.5.1 netmask 255.255.255.0

ifconfig vlan6 create → ifconfig vlan6 vlan 6 vlandev em0 → ifconfig vlan6 inet 192.168.6.1 netmask 255.255.255.0

7.2 B1: ifconfig vlan5 create → ifconfig vlan5 vlan 5 vlandev em0

B1: ifconfig vlan6 create → ifconfig vlan6 vlan 6 vlandev em0

7.3 B1: ifconfig vlan61 create → ifconfig vlan61 vlan 6 vlandev em1

B1: ifconfig vlan51 create → ifconfig vlan51 vlan 5 vlandev em3

7.4 PC2: ifconfig vlan6 create → ifconfig vlan6 vlan 6 vlandev em0 → ifconfig vlan6 inet 192.168.6.2 netmask 255.255.255.0

7.5 B2: ifconfig em2 down → ifconfig vlan6 create → ifconfig vlan6 vlan 6 vlandev em0

(LAN2)

ifconfig vlan61 create → ifconfig vlan6 vlan 6 vlandev em1 (LNK1)

7.6 PC3: ifconfig vlan5 create → ifconfig vlan5 vlan 5 vlandev em0 → ifconfig vlan5 inet 192.168.5.3 netmask 255.255.255.0

7.7 B3: ifconfig vlan5 create → ifconfig vlan5 vlan 5 vlandev em0 (LAN3)

Ifconfig vlan51 create → ifconfig vlan51 vlan 5 vlandev em1 (LNK3)

7.8 PC1: ping 192.168.6.2 , ping 192.168.5.3, ναι μπορούμε να κάνουμε ping.

7.9 B1: ifconfig bridge1 -stp em0

7.10 PC1: tcpdump -vvv -X -i em0

7.11 PC2: arp -d -a → ping -c 1 192.168.1.1

Το ethertype έχει τιμή ARP (0x0806) για τα ARP πακέτα και IPv4 (0x0800) για τα IPv4 πακέτα.

7.12 Τα πλαίσια με το καινούριο ping περιέχουν ethertype VLAN 802.1Q (0x8100)

7.13 Και τα δύο περιέχουν το παραπάνω ethertype (0x8100) αλλά στη συνέχεια παρατηρούμε και το ethertype του IPv4 (0x0800) ή ARP (0x0806) Στα bytes 17-18 από την αρχή.

7.14 Στο Vlan tag 802.1Q που έχει διεύθυνση προορισμού και πηγής

7.15 PC1: tcpdump -vvv -XX -e -i vlan5

7.16 PC3: arp -d -a → ping -c 1 192.168.5.1

Το ethertype έχει τιμή ARP (0x0806) για τα ARP πακέτα και IPv4 (0x0800) για τα IPv4 πακέτα.

Όχι δεν υπάρχει πεδίο σχετικό με το Vlan.

7.17 B1: ifconfig bridge1 stp em0

PC1: tcpdump -vvv -XX -e -i em0

7.18 Είναι πλαίσια 802.3 τα πλαίσιο BPDU και δεν έχουν ethertype, αλλά στη θέση του έχουν το πεδίο length του πλαισίου

7.19 Αν δεν είχαμε αφαιρέσει τη διεπαφή em0 από το stp τότε θα είχαμε

PC1: tcpdump -vvv -XX -e -i em0 ‘ether[12:2]>1500’