ΕΡΓΑΣΙΑ
ΔΙΑΧΕΊΡΙΣΗ ΔΙΚΤΥΩΝ

ΟΜΑΔΑ:

Ον/μο: ΚΟΜΙΝΗ ΣΥΜΕΛΑ ΦΩΤΕΙΝΗ
ΑΜ: 1115201400072

Ον/μο: ΜΑΣΤΟΡΑΚΗ ΑΙΚΑΤΕΡΙΝΑ
ΑΜ: 1115201400100

1ο ΠΕΙΡΑΜΑ

Σε κάθε φάση προσθέσαμε τις κατάλληλες εντολές για τα flows του switch και τα rooting tables. Στο τέλος της κάθε φάσης διαγράφουμε τις ρυθμίσεις της προηγούμενης.

1η φάση

```
os.system('ovs-ofctl mod-flows switch in_port=1,actions=output:4')
os.system('ovs-ofctl mod-flows switch in_port=4,actions=output:1')
os.system('ovs-ofctl mod-flows switch in_port=2,actions=drop')
os.system('ovs-ofctl mod-flows switch in_port=3,actions=drop')
os.system('ovs-ofctl del-flows eNodeB1')
os.system('ovs-ofctl del-flows eNodeB2')
os.system('ovs-ofctl del-flows rsu1')

car[0].cmd('ip route add 200.0.10.2 via 200.0.10.50')
client.cmd('ip route add 200.0.10.100 via 200.0.10.150')
```

2η φάση

```
os.system('ovs-ofctl mod-flows switch in_port=1,actions=drop')
os.system('ovs-ofctl mod-flows switch in_port=2,actions=output:4')
os.system('ovs-ofctl mod-flows switch in_port=4,actions=output:2,3')
os.system('ovs-ofctl mod-flows switch in_port=3,actions=output:4')
os.system('ovs-ofctl del-flows eNodeB1')
os.system('ovs-ofctl del-flows eNodeB2')
os.system('ovs-ofctl del-flows rsu1')

car[0].cmd('ip route del 200.0.10.2 via 200.0.10.50')
client.cmd('ip route del 200.0.10.100 via 200.0.10.150')
```

3η φάση

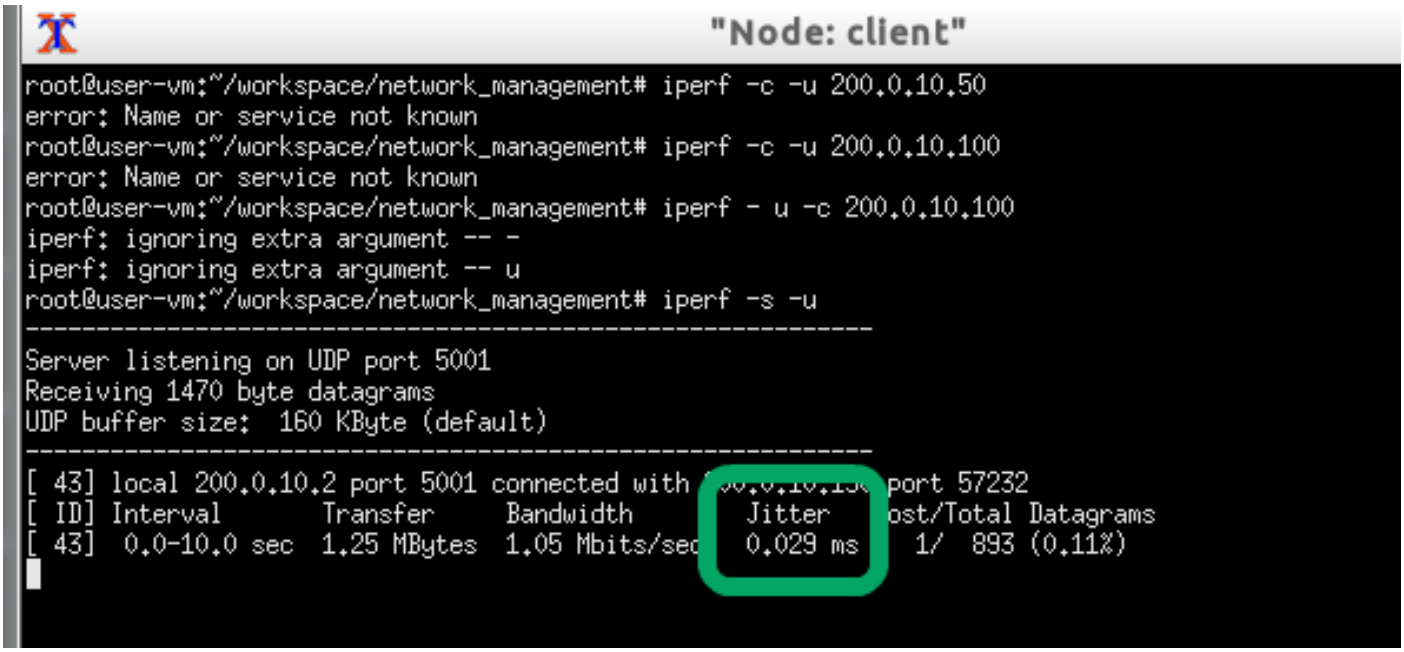
```
os.system('ovs-ofctl mod-flows switch in_port=1,actions=drop')
os.system('ovs-ofctl mod-flows switch in_port=3,actions=drop')
os.system('ovs-ofctl mod-flows switch in_port=2,actions=output:4')
os.system('ovs-ofctl mod-flows switch in_port=4,actions=output:2')
os.system('ovs-ofctl del-flows eNodeB1')
os.system('ovs-ofctl del-flows eNodeB2')
os.system('ovs-ofctl del-flows rsu1')
```

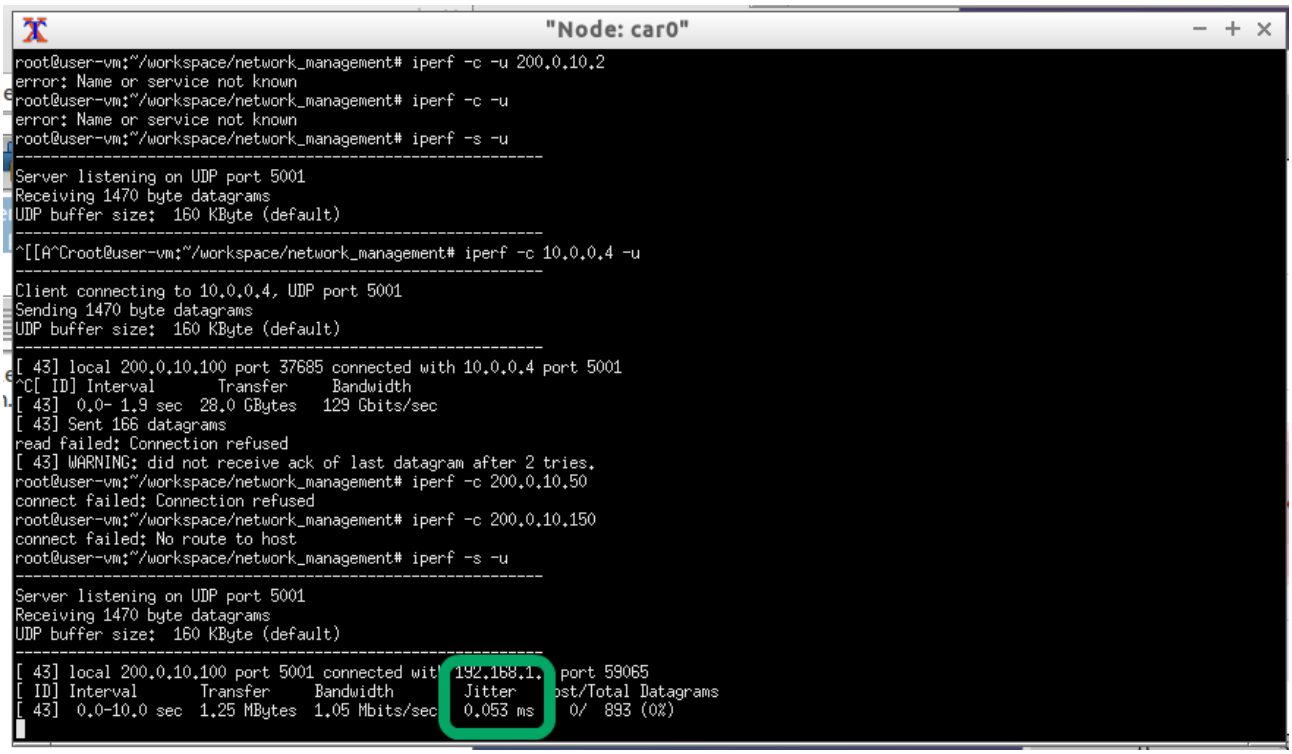
Σε κάθε φάση παίρνουμε μετρήσεις για:

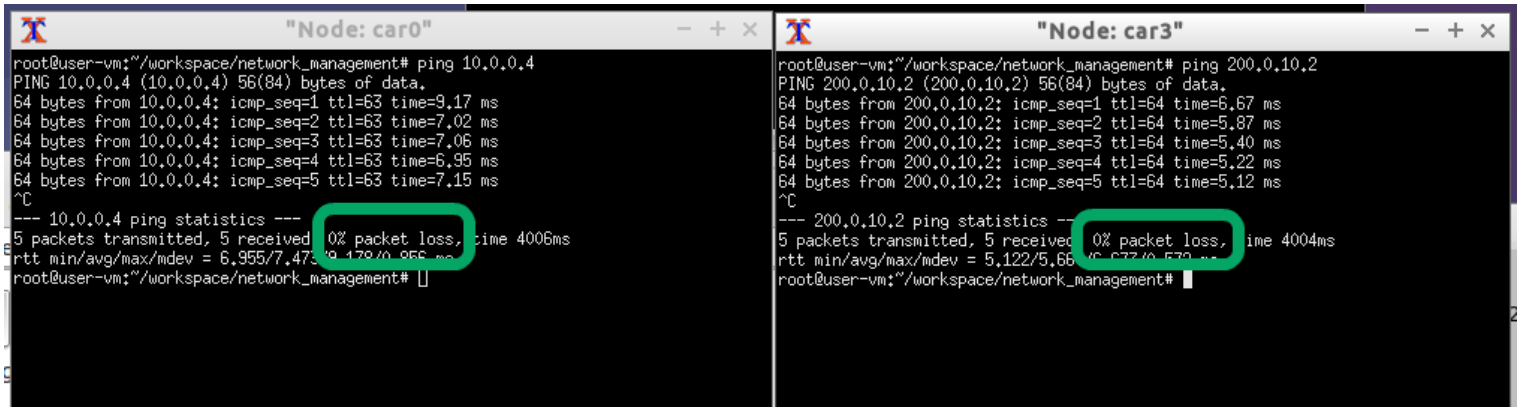
- ✓ Throughput
- ✓ Jitter
- ✓ Packet loss
- ✓ Latency

Οι μετρήσεις των jitter και packet loss, σε κάθε φάση, φαίνεται στα παρακάτω screenshots:

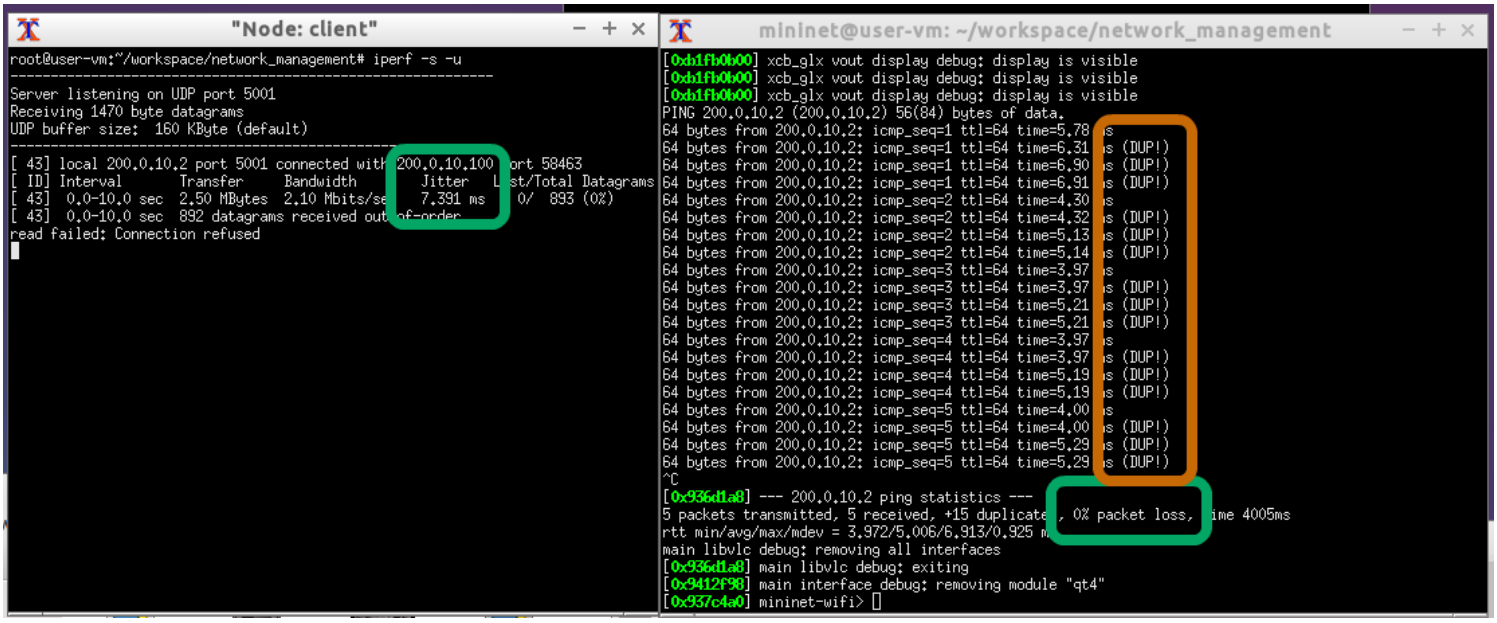
1η φάση




"Node: car0"
root@user-vm:~/workspace/network_management# iperf -c -u 200.0.10.2
error: Name or service not known
root@user-vm:~/workspace/network_management# iperf -c -u
error: Name or service not known
root@user-vm:~/workspace/network_management# iperf -s -u
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 160 KByte (default)
^[[A^Croot@user-vm:~/workspace/network_management# iperf -c 10.0.0.4 -u
Client connecting to 10.0.0.4, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size: 160 KByte (default)
[43] local 200.0.10.100 port 37685 connected with 10.0.0.4 port 5001
^C[ID] Interval Transfer Bandwidth
[43] 0.0- 1.9 sec 28.0 GBytes 129 Gbits/sec
[43] Sent 166 datagrams
read failed: Connection refused
[43] WARNING: did not receive ack of last datagram after 2 tries.
root@user-vm:~/workspace/network_management# iperf -c 200.0.10.50
connect failed: Connection refused
root@user-vm:~/workspace/network_management# iperf -c 200.0.10.150
connect failed: No route to host
root@user-vm:~/workspace/network_management# iperf -s -u
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 160 KByte (default)
[43] local 200.0.10.100 port 5001 connected wit 192.168.1. port 59065
[ID] Interval Transfer Bandwidth Jitter ost/Total Datagrams
[43] 0.0-10.0 sec 1.25 MBytes 1.05 Mbits/sec 0.053 ms 0/ 893 (0%)

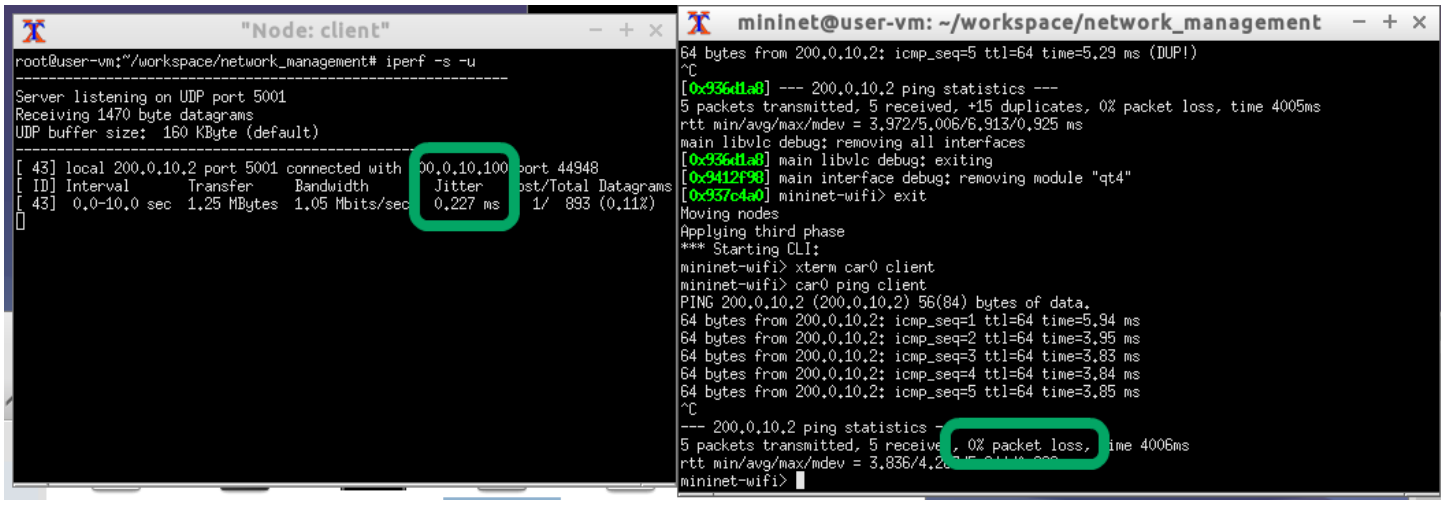

"Node: car0"
root@user-vm:~/workspace/network_management# ping 10.0.0.4
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
64 bytes from 10.0.0.4: icmp_seq=1 ttl=63 time=9.17 ms
64 bytes from 10.0.0.4: icmp_seq=2 ttl=63 time=7.02 ms
64 bytes from 10.0.0.4: icmp_seq=3 ttl=63 time=7.06 ms
64 bytes from 10.0.0.4: icmp_seq=4 ttl=63 time=6.95 ms
64 bytes from 10.0.0.4: icmp_seq=5 ttl=63 time=7.15 ms
^C
--- 10.0.0.4 ping statistics ---
5 packets transmitted, 5 received 0% packet loss, time 4006ms
root@user-vm:~/workspace/network_management#
"Node: car3"
root@user-vm:~/workspace/network_management# ping 200.0.10.2
PING 200.0.10.2 (200.0.10.2) 56(84) bytes of data.
64 bytes from 200.0.10.2: icmp_seq=1 ttl=64 time=6.67 ms
64 bytes from 200.0.10.2: icmp_seq=2 ttl=64 time=5.87 ms
64 bytes from 200.0.10.2: icmp_seq=3 ttl=64 time=5.40 ms
64 bytes from 200.0.10.2: icmp_seq=4 ttl=64 time=5.22 ms
64 bytes from 200.0.10.2: icmp_seq=5 ttl=64 time=5.12 ms
^C
--- 200.0.10.2 ping statistics ---
5 packets transmitted, 5 received 0% packet loss, time 4004ms
rtt min/avg/max/mdev = 5.122/5.66
root@user-vm:~/workspace/network_management#

2η φάση

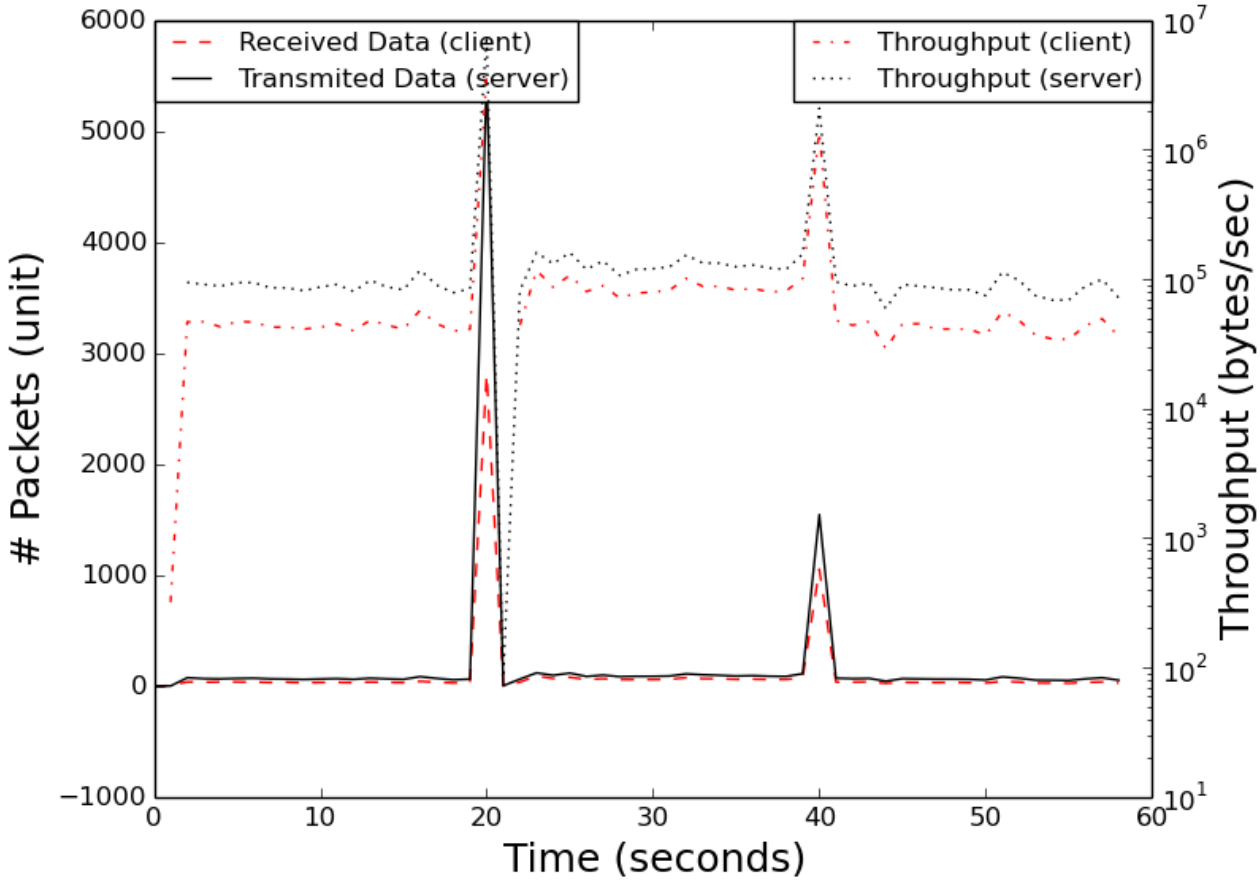


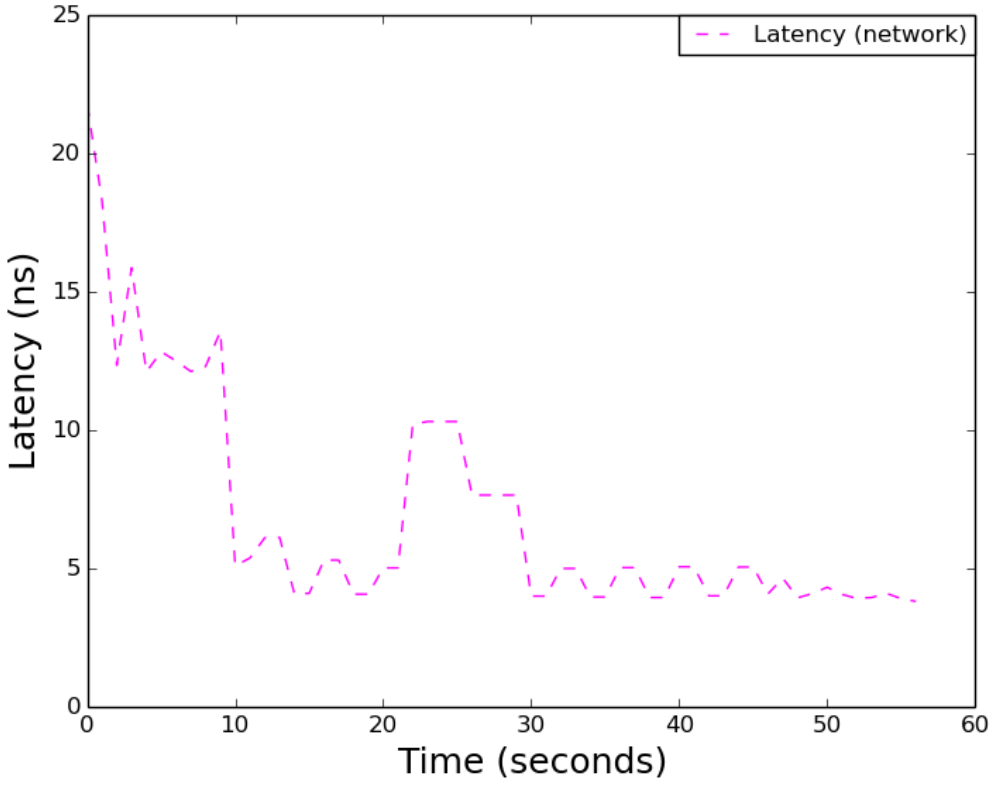
Εδώ παρατηρούμε ότι εμφανίζεται η ένδειξη DUP δίπλα από κάθε πακέτο, γιατί τα πακέτα μεταδίδονται bicasting και ο client τα λαμβάνει διπλά.

3η φάση



Οι μετρήσεις για το throughput και το latency φαίνονται στα παρακάτω διαγράμματα, τα οποία εμφανίζονται μετά από την 3η φάση.





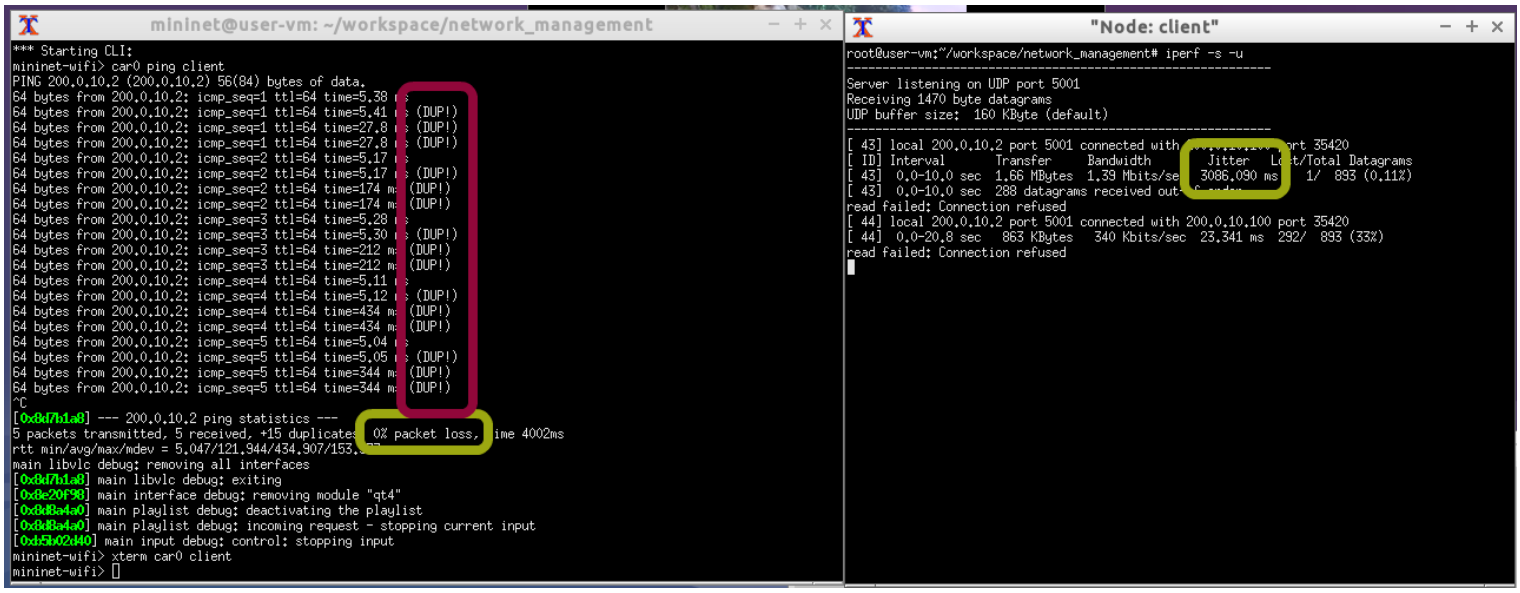
2ο ΠΕΙΡΑΜΑ – bicasting

Έχουμε ρυθμίσει τα flows του switch, με αντίστοιχο τρόπο με το 1ο πείραμα για κάθε φάση.

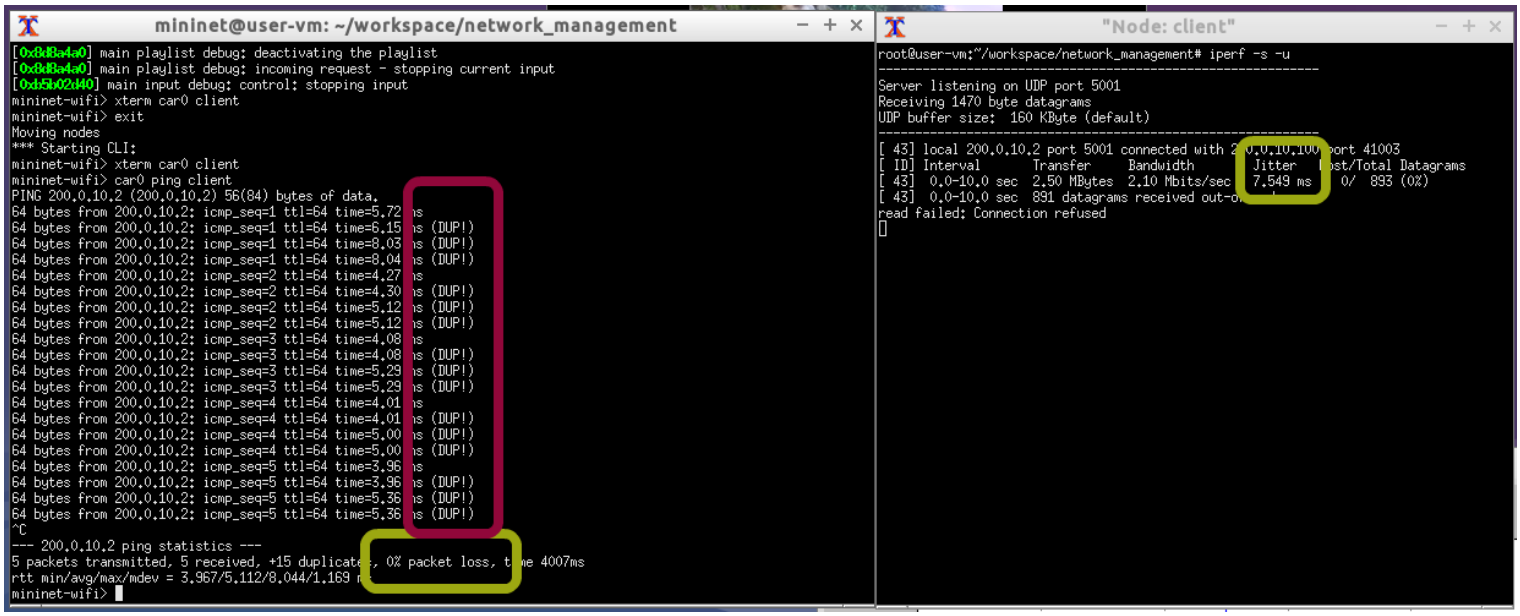
Οι μετρήσεις των jitter και packet loss για κάθε φάση είναι οι παρακάτω:

1η φάση

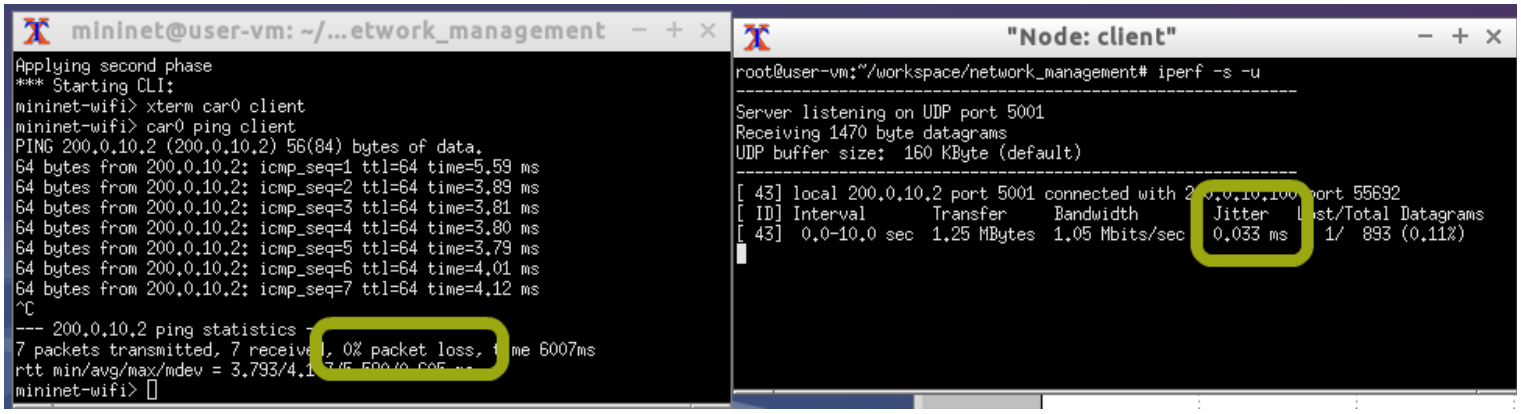
eNodeB1 – rsu1



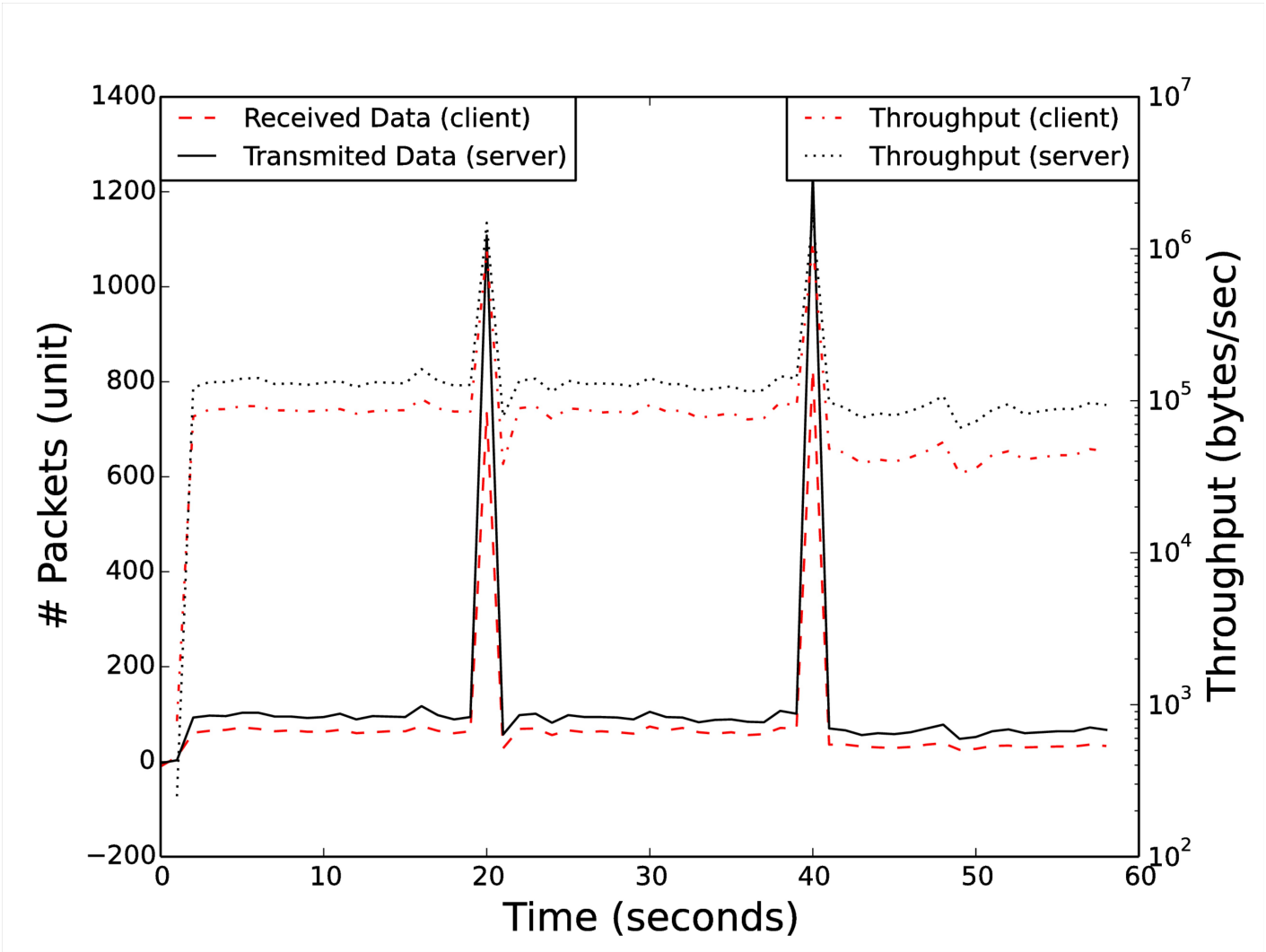
eNodeB2 – rsu1

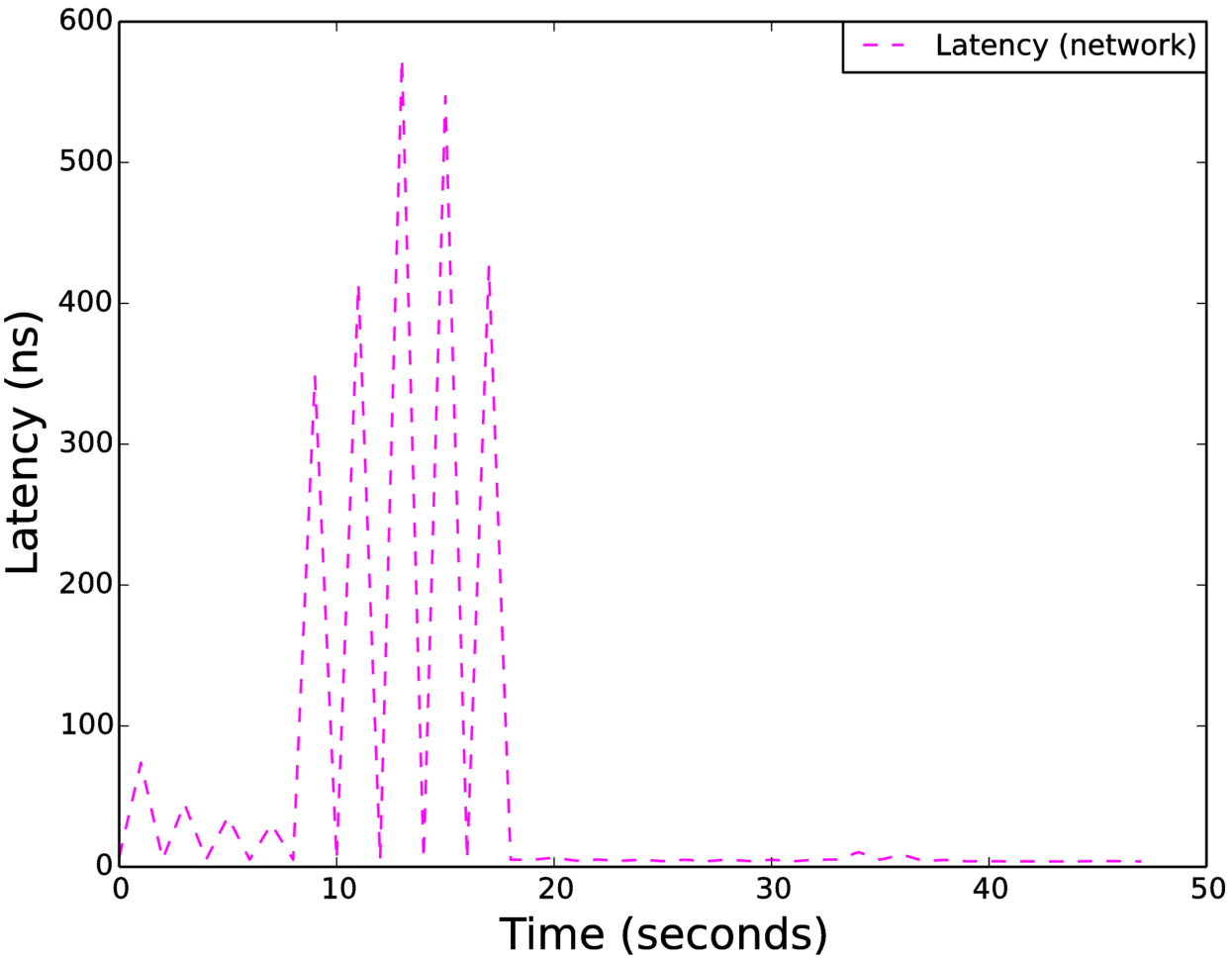


2η φάση



Ομοίως με το 1ο πείραμα, οι μετρήσεις του throughput και του latency εμφανίζονται με γραφική παράσταση μετά τη 2η φάση.





Σύγκριση αποτελεσμάτων

Από τα αποτελέσματα των παραπάνω πειραμάτων παρατηρούμε ότι:

- Το latency είναι μικρότερο όταν το car0 επικοινωνεί κατευθείαν με τον client μέσω μίας μόνο κεραίας, ενώ αυξάνεται όταν επικοινωνεί είτε μέσω τρίτου είτε με bicasting.
- Το throughput, παρατηρούμε ότι, αυξάνεται κατά ένα μεγάλο ποσοστό στις περιπτώσεις που έχουμε bicasting, γεγονός που είναι λογικό καθώς ο συνολικός αριθμός πακέτων που μεταδίδονται είναι ο διπλάσιος.
- Όσον αφορά το packet loss, στην πλειονότητα των περιπτώσεων είναι 0 - καθώς κανένα πακέτο δεν χάνεται – ωστόσο, στα παραπάνω screenshots παρατηρείτε την περίπτωση όπου ένα πακέτο έχει χαθεί λόγω χρήσης UDP πρωτοκόλλου.