universitätbonn

Mobile Communication

Summer Semester 2016

Solution for
Practical Assignment 02

Done by:

Che-Hao, Kang, 2840977

Ahmad, Hemid, 9027822

Exercise 1: Collection of toplogy information and measurement data

Using a programming language of your choosing, implement a program that fetches topology information and measurement data from the URLs given above, processes the contained data and stores all information in a suitable file format for further analysis and processing.
Specifically, your program should . . .

1. Access the URLs in a suitable interval (e.g. repeat each 30 seconds) and download the contained data for further processing.
2. Parse and extract the contained information according to the file formats described above.
3. Transform the extracted information to a data format of your choosing. The data format must contain all relevant parts of the topology information and measurement data along with the temporal information, i.e. the timestamps of the topology status data and the single measurements.
4. Store the transformed data on the computer's hard disk so that it can easily be used for further analysis.

Explain the structure of your data format and give a brief description of how to compile and run your program.

In PA2, there are three types of data we need to manage:

1. **json files** containing systemTime, neighbors, links, topology and routes
2. **HTTP throughput measurement** containing the time for downloading a file and the size of it
3. **End-to-end delay and loss measurement** containing packet loss rate and round-trip time

The following are how we manipulate them:

About **json files**, we create six lists for each router and append the content of every json file of one router to its own list. After this, we put these six lists into another list for future use. At this moment, we simply collect all json files and don't do any advanced data processing because we want to keep the data as pure as it was so that we can use any columns of a json file when we need them. For saving network topology information, we put the content of all json files of one router to a json file (e.g., topologyJson_1_ALL.json)

For **Exercise 2-1**, we employ the **"topology"** of a json file to generate tables for drawing graphs. We traverse all timestamps to collect topology data for each link from time to time. Because there are six routers and these routers may have their own topology data of a link, we tackle this situation by calculating the averages of linkQuality, neighborLinkQuality and tcEdgeCost. The below table is what we generate:

From	To	LinkQuality	NeighborLinkQuality	tcEdgeCost	Timestamp
10.0.0.2	10.0.0.1	1	1	1024	1466541400
10.0.0.3	10.0.0.6	0.909	0.607	1855.333333	1466541400
10.0.0.1	10.0.0.2	1	1	1024	1466541400
10.0.0.4	10.0.0.3	1	0.746	1374.333333	1466541400
10.0.0.5	10.0.0.3	0.776	1	1318	1466541400
10.0.0.6	10.0.0.5	1	1	1024	1466541400
10.0.0.4	10.0.0.5	1	0.8415	1216.666667	1466541400
10.0.0.3	10.0.0.5	1	0.776	1318	1466541400
10.0.0.5	10.0.0.4	0.8055	1	1271.333333	1466541400
10.0.0.5	10.0.0.6	1	1	1024	1466541400
10.0.0.3	10.0.0.2	1	1	1024	1466541400
10.0.0.6	10.0.0.3	0.6345	0.858	1905	1466541400
10.0.0.2	10.0.0.3	1	1	1024	1466541400
10.0.0.3	10.0.0.4	0.723666667	1	1413.666667	1466541400

You can easily observe there is a link from 10.0.0.2 to 10.0.0.1 and what the linkQuality, neighborLinkQuality and tcEdgeCost are. One thing worth noticing is that even if we collect json files of six routers at the same time, the timestamps are lightly different. Due to this, if differences of timestamps of each router are within 10 seconds, our programming sees those json files as collected at the same time. That is, our programming computes averages with those json files.

Also, we use this information to find **the best route from 10.0.0.1 to 10.0.0.6** by employing a recursion `def findBestRouteRecur(topology, nowNode, cost, routeList)`. The below table shows the best route is **10.0.0.1→10.0.0.2→10.0.0.3→10.0.0.6** and **the overall cost** is **4695**.

From	To	Cost
10.0.0.1	10.0.0.2	4695
10.0.0.2	10.0.0.3	4695
10.0.0.3	10.0.0.6	4695

About **HTTP throughput measurement**, we use a dictionary to store data. For example, if at timestamp 1466515456, router 1 downloaded a 4194304-byte file from router 6 and it took 25.27 seconds, the dictionary will be like {1466515456L: [4194304.0, 25.27]}. The key is the timestamp, and the value is a list containing transferred bytes and used time.

For **Exercise 2-2**, we use the above information to generate tables which has **timestamp**, **throughput** and **bytes**:

timestamp	throughput	bytes
1466541416	1448	86677
1466541476	1088	129914
1466541536	847	151431
1466541596	691	164260
1466541656	616	182881
1466541716	662	236254
1466541776	768	320035
1466541836	876	416848
1466541896	794	425333
1466541956	719	428026
1466542016	655	429271

The **throughput** is computed by the transferred bytes divided by the used time till a certain timestamp; the **bytes** is the downloaded bytes till a certain timestamp.

Because the earliest timestamp of **HTTP throughput measurement** is 1463735418 which is far ahead of json files' timestamps (from 1466541400 to 1466548982), we truncate the data and draw the plot from around 1466541400.

About **End-to-end delay and loss measurement**, we include a dictionary to store data. For example, if at timestamp 1466531931, router 1 pinged router 6 and got

```
50 packets transmitted, 45 packets received, 10% packet loss
round-trip min/avg/max = 5.424/6.933/14.456 ms
```

, the dictionary will be like {1466531931L: ['50 45 10%', '5.424/6.933/14.456']}. The key is the timestamp, and the value is a list containing packet-related data and round-trip time.

For **exercise 2-3**, we transform the above information into the below table:

timestamp	packetloss	min	avg	max
1466541445	12	5.424	6.626	11.034
1466541496	4	4.958	6.514	10.068
1466541546	8	4.657	6.3	11.016
1466541596	14	4.938	6.426	15.809
1466541646	4	4.893	6.166	10.937
1466541696	12	5.067	6.481	12.283
1466541746	18	5.454	16.2	305.687
1466541796	16	4.699	7.27	12.137
1466541846	14	4.902	6.27	11.022
1466541896	16	4.598	7.399	13.498

In one timestamp, there are packet loss percentage and round-trip min/avg/max.

Similar to **HTTP throughput measurement**, the earliest timestamp of **End-to-end delay and loss measurement** is 1463735419 which is too far away from json files' timestamps (from 1466541400 to 1466548982). We also remove those data and draw plots from 1466541400.

How to **compile** and **run** **(the detailed comments of our programming is in practical_2.py)**

1. If you only want to manipulate data and don't need downloading, you could comment out the codes between **# @@@+++++ turn on when downloading** and **# @@@----- turn on when downloading**
2. For compiling and running, just simply open a terminal and type "**python practical_2.py**"

Exercise 2: Performance analysis

Using the program implemented in Exercise 1, collect the topology information and measurement data of at least a 2 hour timeframe. Then, using the collected data, perform an in-depth analysis of the network performance. This analysis should cover the following aspects:

1. Find a way to detect and visualize changes to the network topology over time. This should not only include changes to the used routes, but also cover all significant changes to the quality of the individual links. Give reasons for your solution and describe your findings.

Answer (1):

In order to depict the network topology over time, we have collected data related to the link quality from both variable ("linkQuality" and "neighborLinkQuality") from JSON files. Then the link cost between each two nodes with a link between them, could be calculated from **equation (1)** or just simply from the variable "tcEdgeCost"

$$\text{LinkCost} = \frac{1}{\text{linkQuality} \times \text{neighborLinkQuality}} \qquad (1)$$

We made a code to collect the required data fields using python programming language, source code can be found in "practical_2.py" file under scripts folder. Now, the data required is collected, the next step for us to draw the network topology. "Task_2-2-1.R" draws the links between the different nodes. We follow a certain criteria to highlight the significant changes of the network topology based on the following rules:

- If a new link appears in the topology records in JSON file
- If an existing link disappears from the topology records in JSON file
- If the link quality for a certain link reduces or increases by 0.4 value (It is considered from our side as a significant increase or decrease)

The following figure shows the major 18 changes of the network topology over the period of 2 hours (left-side) plus the best route from node 10.0.0.1 to reach node 10.0.0.1 based on the current state of the topology .

In the below figure, the following can be seen,

- a timestamp to show the time of an event (refined by a number in a red color to make it easy for us during documentation phase)
- Network nodes with the assigned IP address as it is in the given assignment.
- The strength of the link quality (LQ) between two nodes with coloring with green (to show high LQ), orange (to show medium LQ), black (to show low LQ).
- A link was removed, compared to the previous timestamp figure (the link shows no result in the topology records in JSON file, due to either "linkQuality" and "neighborLinkQuality" comes to zero value.

Give reasons for your solution and describe your findings

The Test Result:

The plotted result in the below figure shows that some links are not stable during the test period. Interestingly, that affects the best route between certain sources and destinations (one case is depicted to show the best route from 10.0.0.1 to 10.0.0.6). The major changes have been noticed are as follows:

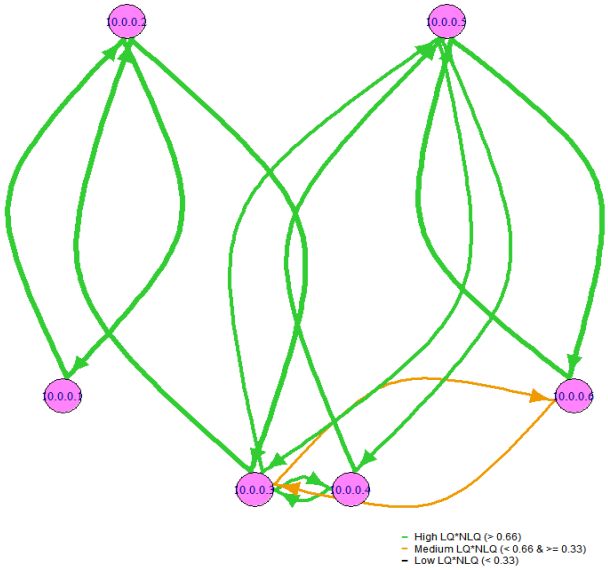
- Instability of a link: the bidirectional link between node 10.0.0.3 and 10.0.0.6 shows several transitions, it starts with medium LQ (at timestamp 1), then rises to high LQ in the next timestamp, then goes back to medium LQ. Moreover, at timestamp 14, the direction from 10.0.0.3 to 10.0.0.6 decreases to low LQ, while the opposite direction shows medium LQ. The following timestamps 15-18 both directions shows low LQ. Finally, the bidirectional link goes back to medium LQ at timestamp 19. Similar instability can be seen between nodes 10.0.0.3 and 10.0.05 and also between 10.0.0.4 and 10.0.0.5.
- Link failure: the bidirectional link between node 10.0.0.3 and 10.0.0.4 has several failure events, to mention some, at timestamps 4, 5, 12, 13, 14, 15, 16. As well, the bidirectional link between node 10.0.0.4 and 10.0.0.5 has suddenly disappeared at timestamp 15.
- New link is added or is recovered after a failure: some of the links fail then recover after a certain time like bidirectional link between node 10.0.0.3 and 10.0.0.4 fails at timestamps 4 – 5 and 12 – 16, then recovers at timestamps 6 and 17 respectively. Also, the bidirectional link between node 10.0.0.4 and 10.0.0.5 has recovered at timestamp 16.

The Result Discussion:

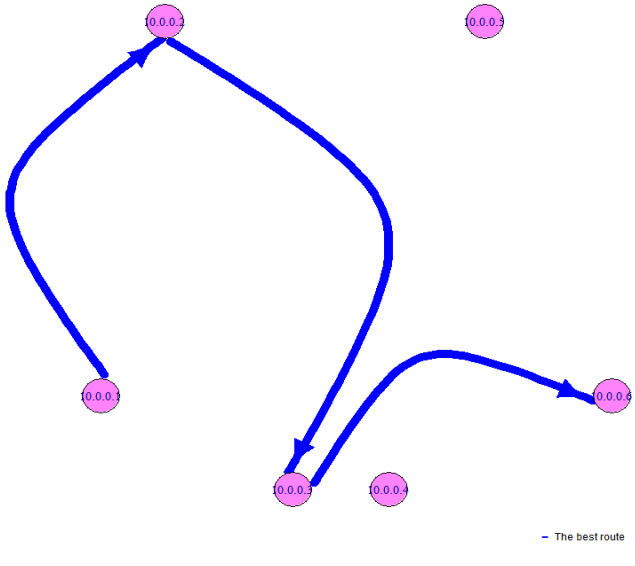
Generally, from the result we can understand the instability of wireless links since some link were not reliable for the whole test period and they goes from high LQ*NLQ to medium or even to lower values. As well as, the link could be broken due to the unreachability of the wireless signals between two nodes. Once values of LQ*NLQ change significantly, routers will start to search for another best route which is to minimize the cost. As can be seen in the diagram, the best route was **10.0.0.1→10.0.0.2→10.0.0.3→10.0.0.6** and later became **10.0.0.1 →10.0.0.2→10.0.0.3→10.0.0.5→10.0.0.6**.

For wireless signal strength, it can be influenced by attenuation, shadowing, reflection, scattering, diffraction and refraction. All these factors result in latency, packet loss, data rate fluctuation, etc. When latency and packet loss get higher, we could recognize that the link between two certain nodes is unstable at that moment. This also leads LQ or NLQ to change correspondingly. That is what we can observe from the diagram.

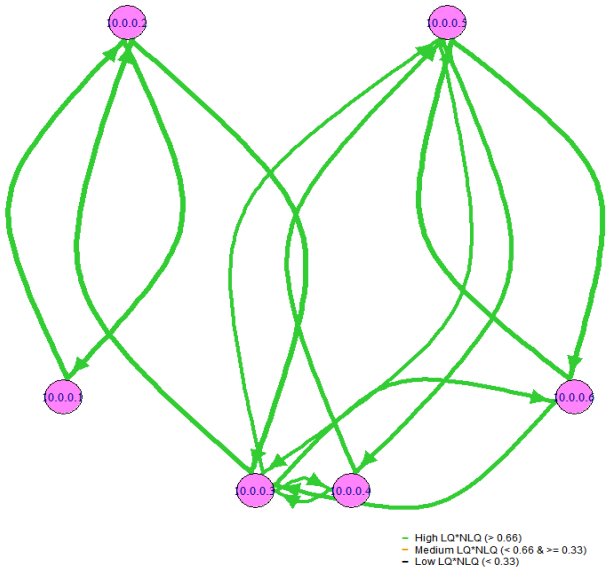
Timestamp: 1466541400 (1)



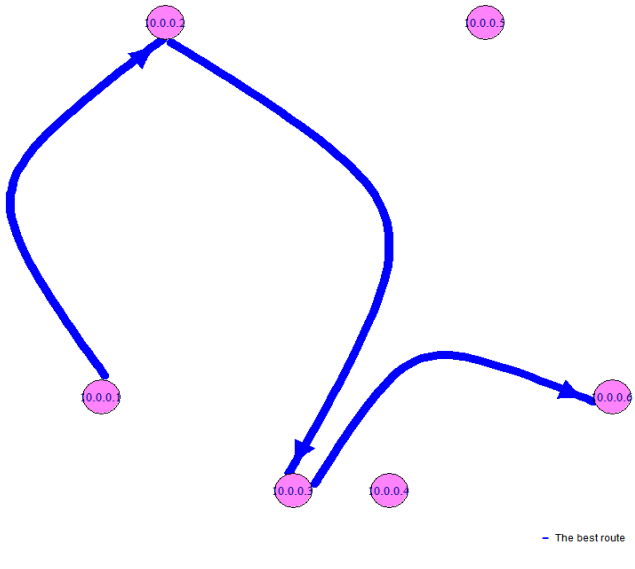
Timestamp: 1466541400 (1)
Cost: 3903.3333



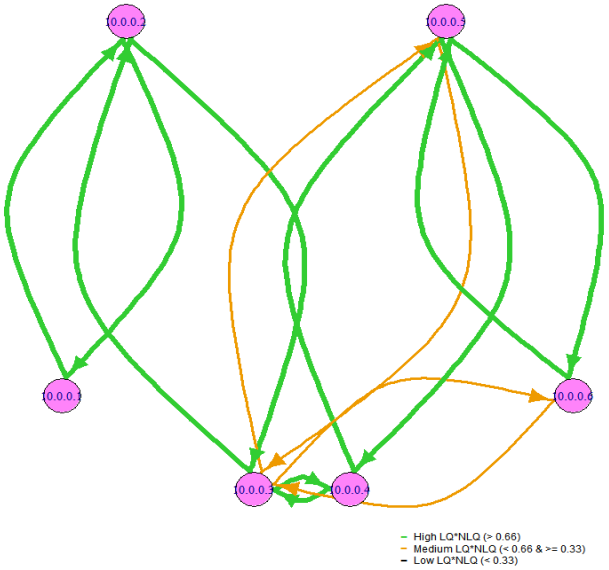
Timestamp: 1466541431 (2)



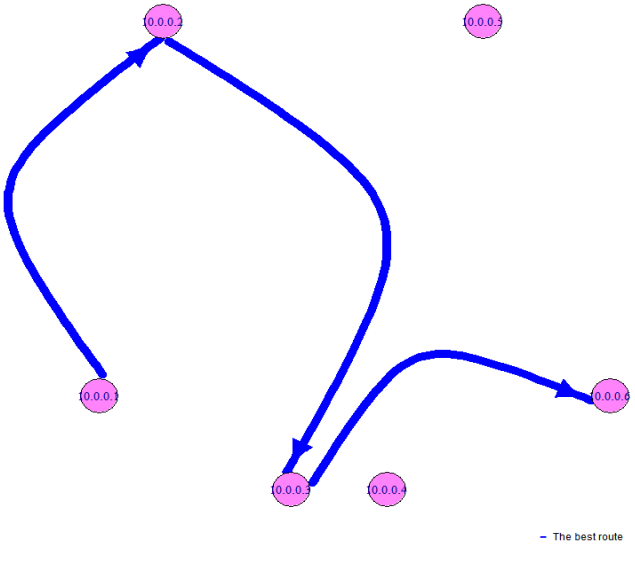
Timestamp: 1466541431 (2)
Cost: 3130.3333



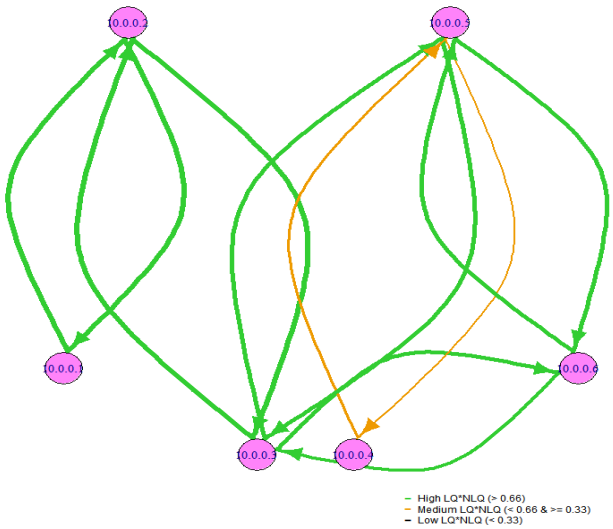
Timestamp: 1466541741 (3)



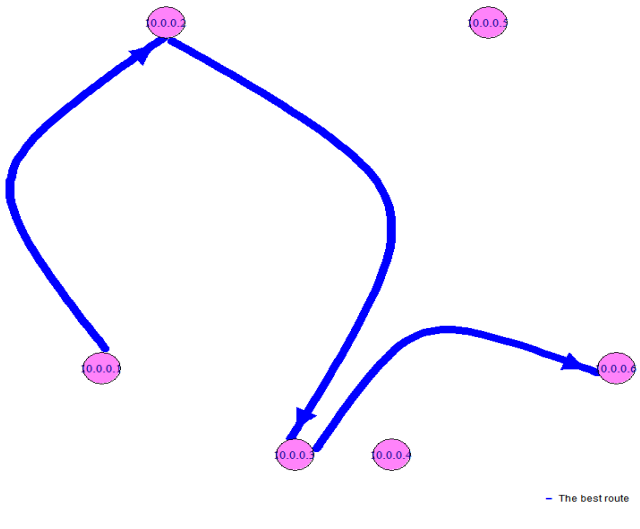
Timestamp: 1466541741 (3)
Cost: 3865.16666



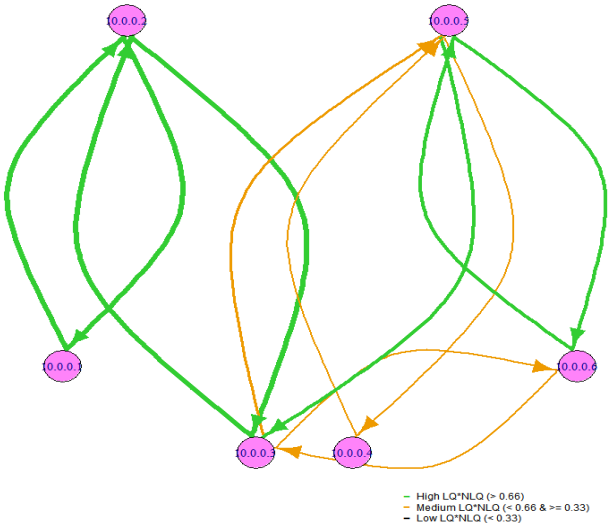
Timestamp: 1466541927 (4)



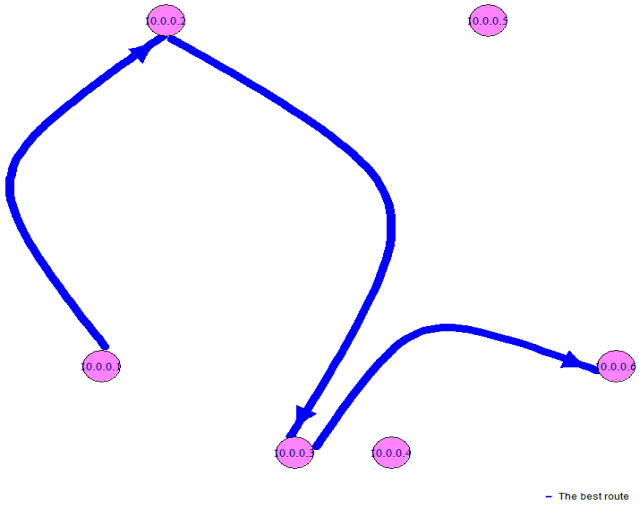
Timestamp: 1466541927 (4)
Cost: 3253.66666



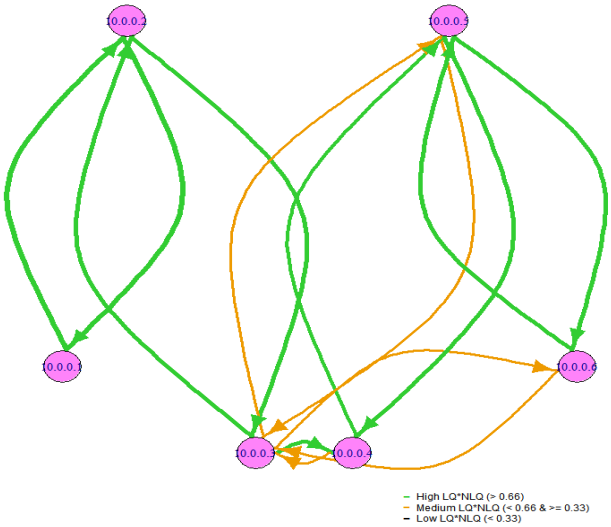
Timestamp: 1466542144 (5)



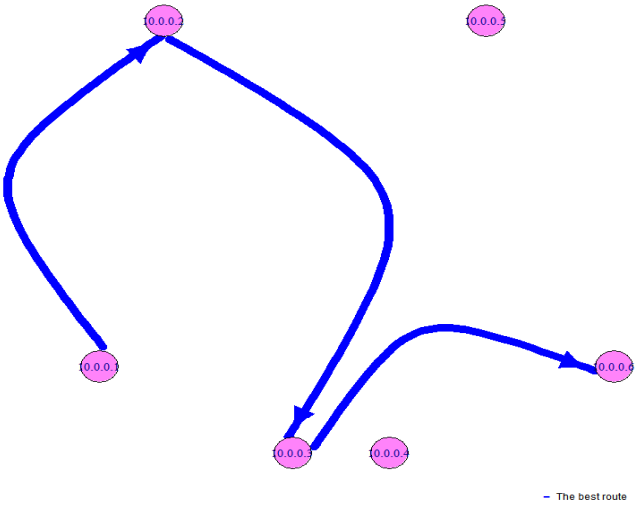
Timestamp: 1466542144 (5)
Cost: 4695

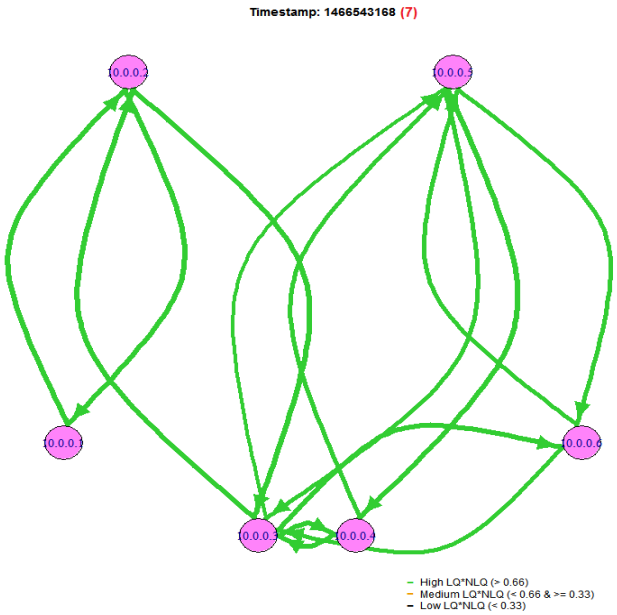


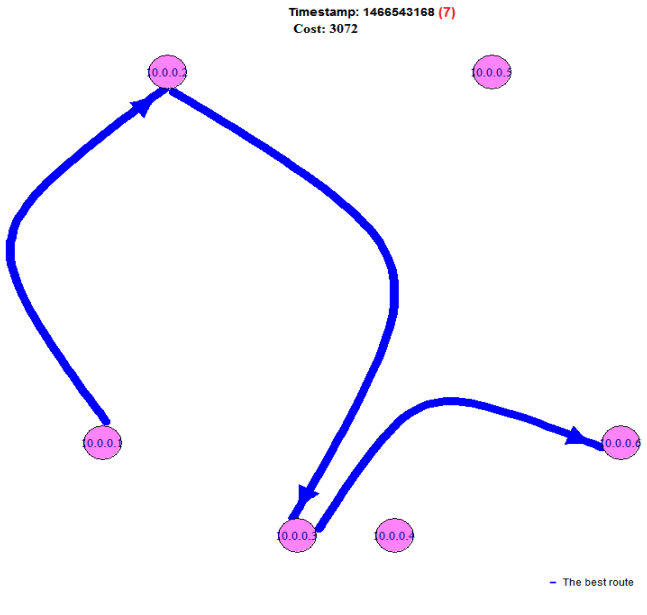
Timestamp: 1466542238 (6)

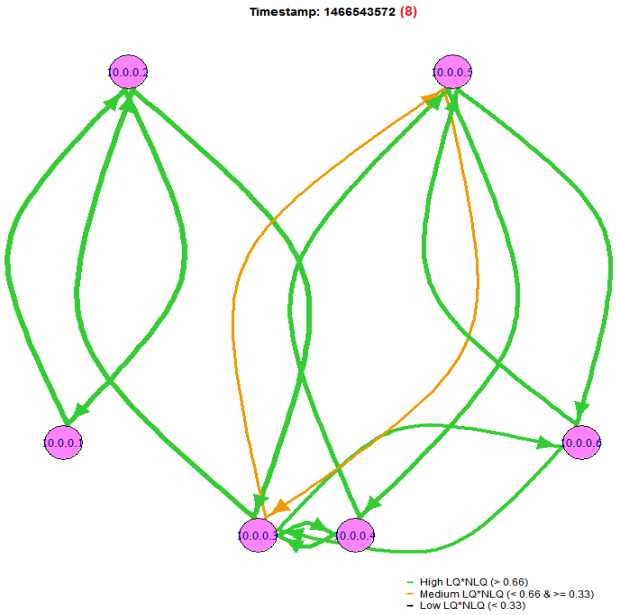


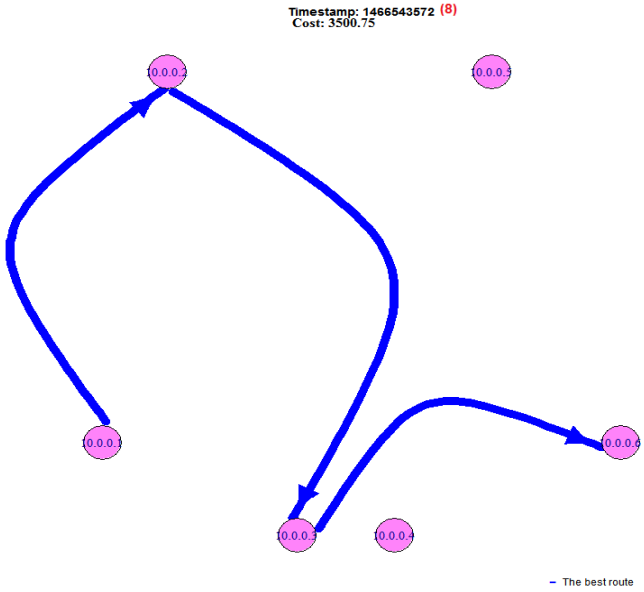
Timestamp: 1466542238 (6)
Cost: 3818.3333

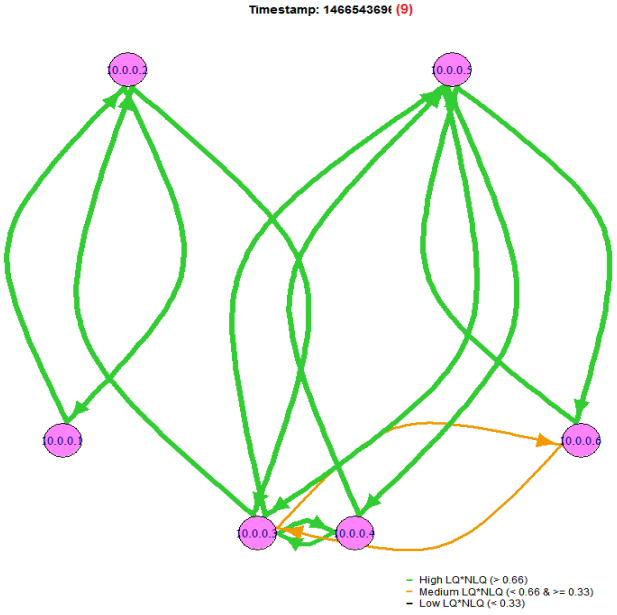


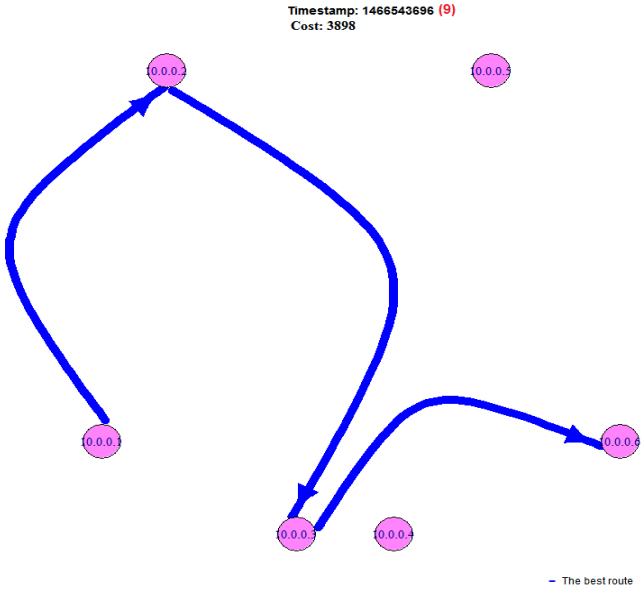

Timestamp: 1466543168 (7)
10.0.0.2
10.0.0.3
10.0.0.1
10.0.0.6
10.0.0.5
10.0.0.4
High LQ*NLQ (> 0.66)
Medium LQ*NLQ (< 0.66 & >= 0.33)
Low LQ*NLQ (< 0.33)

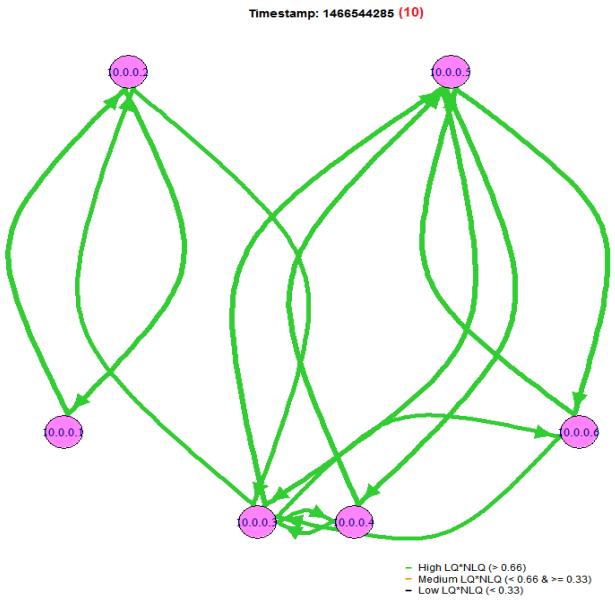

Timestamp: 1466543168 (7)
Cost: 3072
10.0.0.2
10.0.0.3
10.0.0.1
10.0.0.6
10.0.0.5
10.0.0.4
The best route

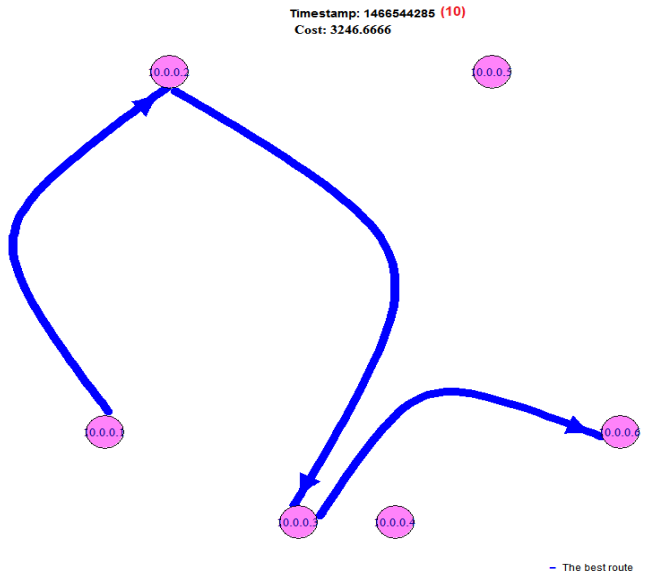

Timestamp: 1466543572 (8)
10.0.0.2
10.0.0.3
10.0.0.1
10.0.0.6
10.0.0.5
10.0.0.4
High LQ*NLQ (> 0.66)
Medium LQ*NLQ (< 0.66 & >= 0.33)
Low LQ*NLQ (< 0.33)

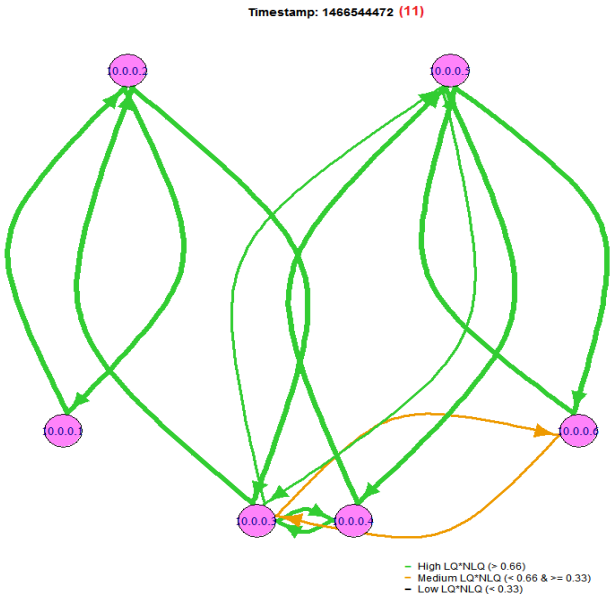

Timestamp: 1466543572 (8)
Cost: 3500.75
10.0.0.2
10.0.0.3
10.0.0.1
10.0.0.6
10.0.0.5
10.0.0.4
The best route

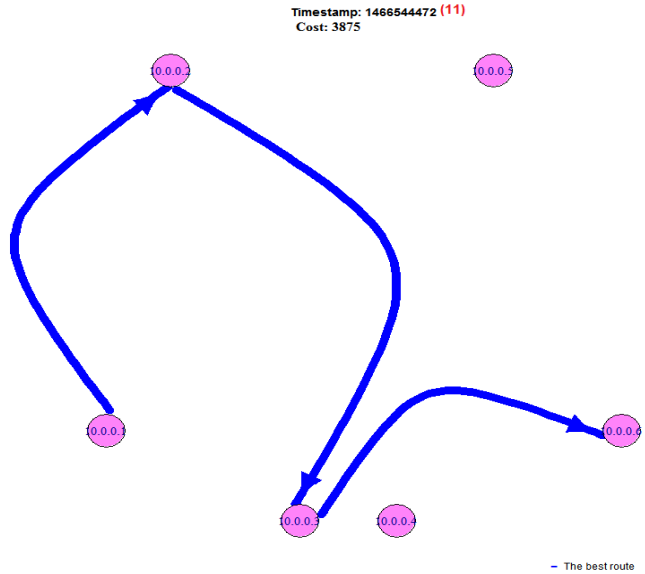

Timestamp: 1466543696 (9)
10.0.0.2
10.0.0.3
10.0.0.1
10.0.0.6
10.0.0.5
10.0.0.4
High LQ*NLQ (> 0.66)
Medium LQ*NLQ (< 0.66 & >= 0.33)
Low LQ*NLQ (< 0.33)

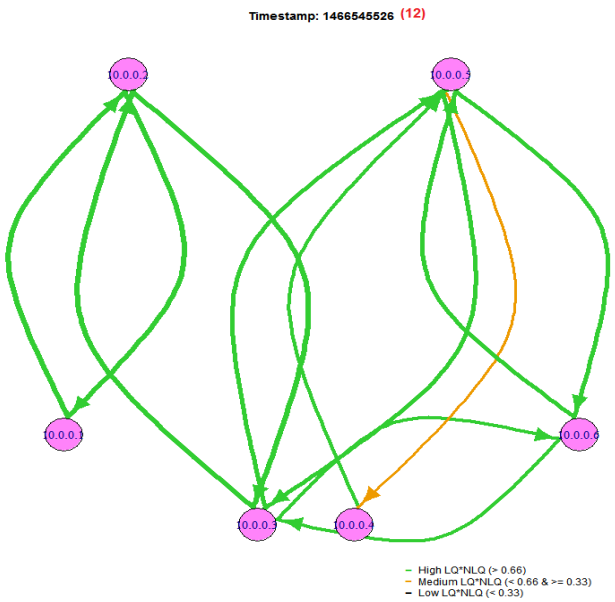

Timestamp: 1466543696 (9)
Cost: 3898
10.0.0.2
10.0.0.3
10.0.0.1
10.0.0.6
10.0.0.5
10.0.0.4
The best route

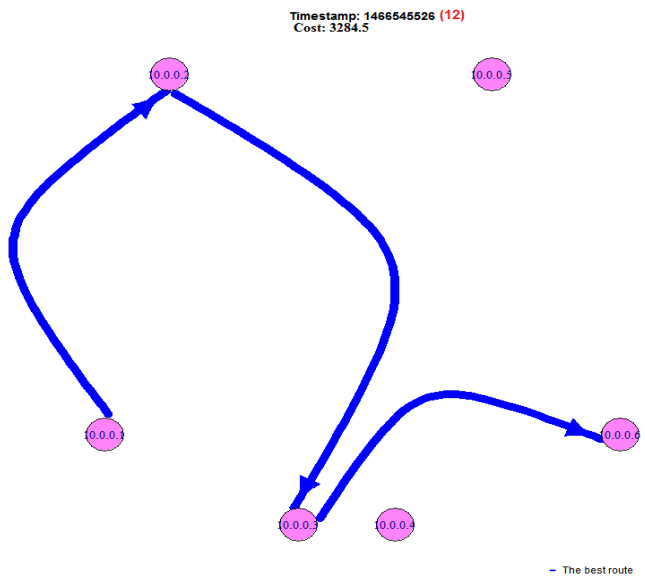

Timestamp: 1466544285 (10)
High LQ*NLQ (> 0.66)
Medium LQ*NLQ (< 0.66 & >= 0.33)
Low LQ*NLQ (< 0.33)


Timestamp: 1466544285 (10)
Cost: 3246.6666
The best route

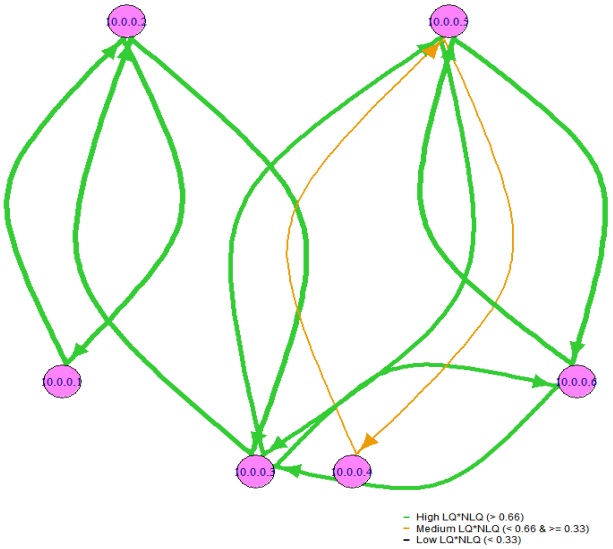

Timestamp: 1466544472 (11)
High LQ*NLQ (> 0.66)
Medium LQ*NLQ (< 0.66 & >= 0.33)
Low LQ*NLQ (< 0.33)


Timestamp: 1466544472 (11)
Cost: 3875
The best route

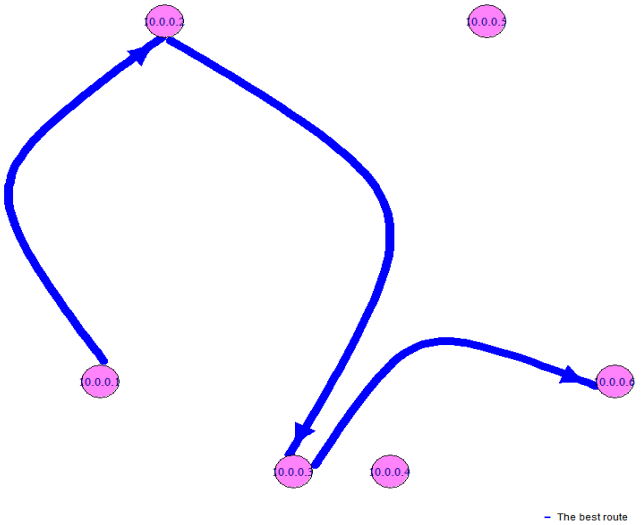

Timestamp: 1466545526 (12)
High LQ*NLQ (> 0.66)
Medium LQ*NLQ (< 0.66 & >= 0.33)
Low LQ*NLQ (< 0.33)


Timestamp: 1466545526 (12)
Cost: 3284.5
The best route

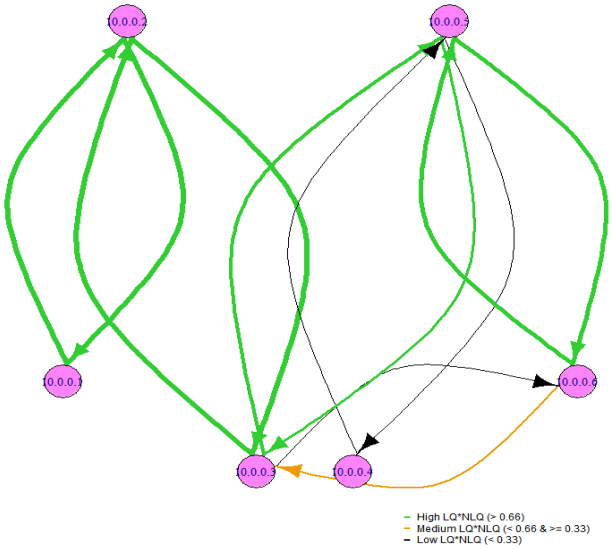
Timestamp: 1466545557 (13)



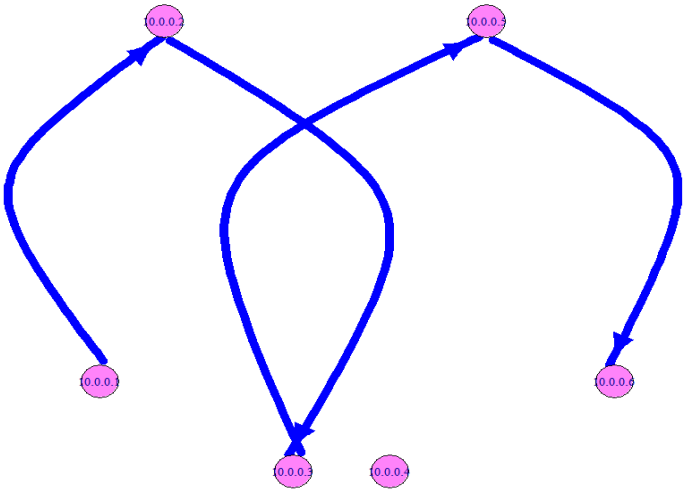
Timestamp: 1466545557 (13)
Cost: 3171.8333



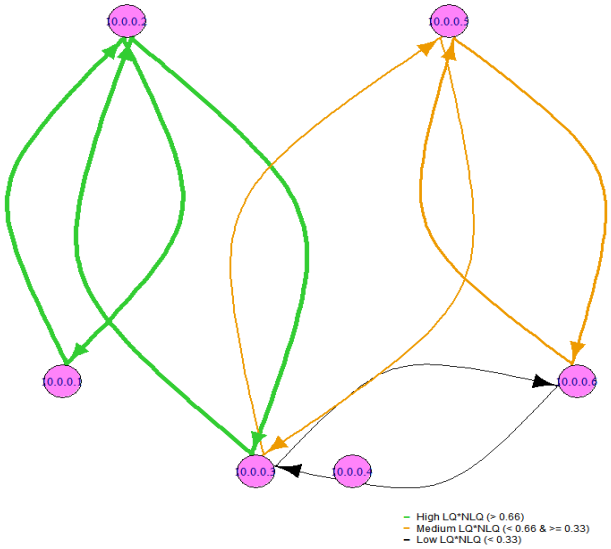
Timestamp: 1466545712 (14)



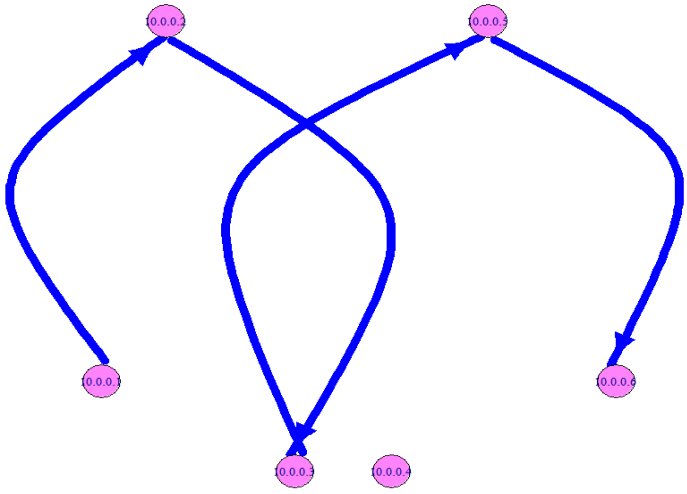
Timestamp: 1466545712 (14)
Cost: 4614.5

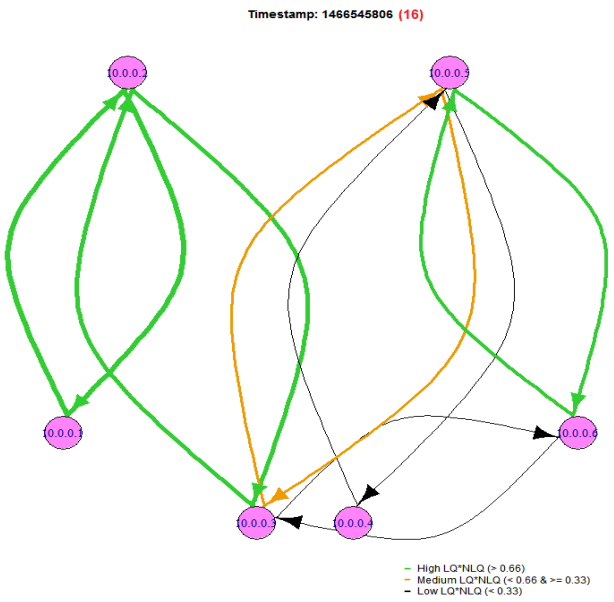


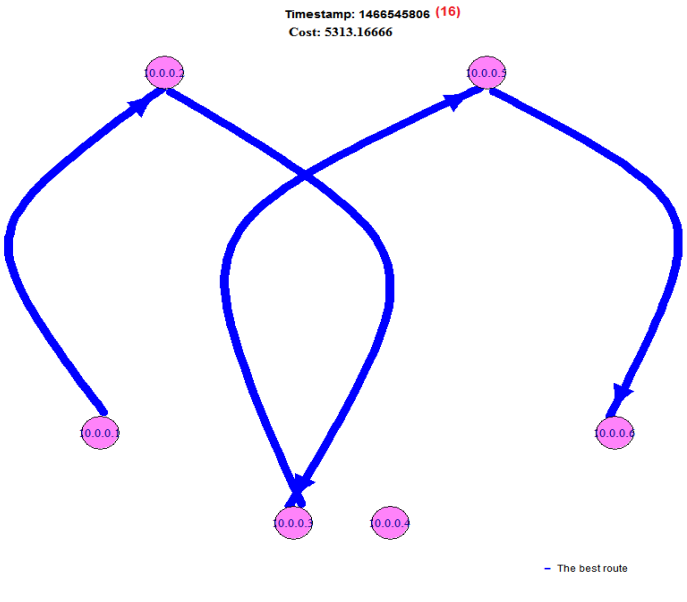
Timestamp: 1466545744 (15)

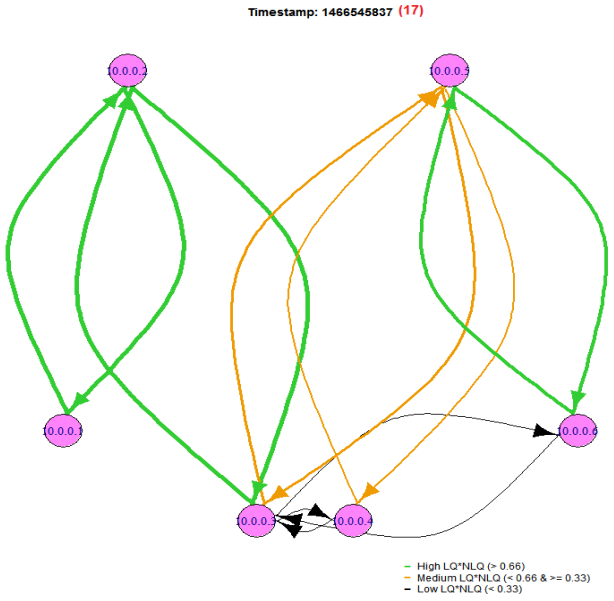


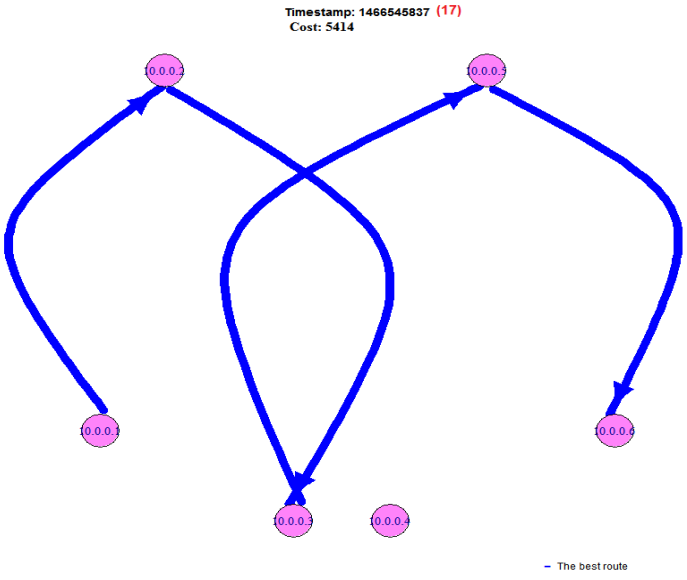
Timestamp: 1466545744 (15)
Cost: 6036.16666

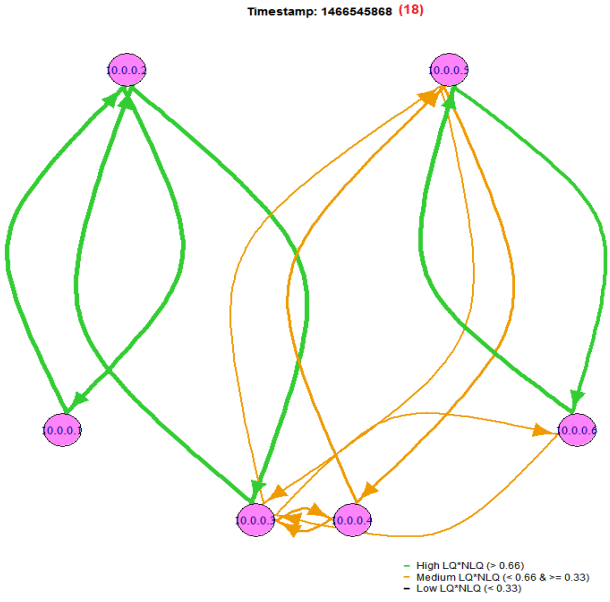


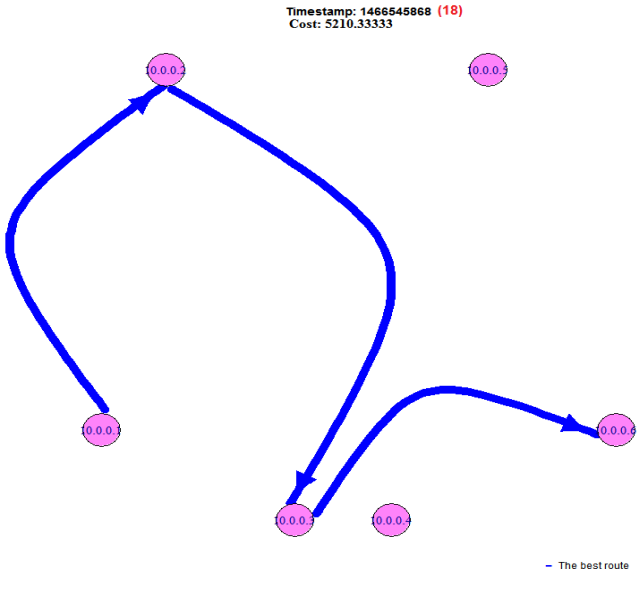

Timestamp: 1466545806 (16)
10.0.0.2
10.0.0.5
10.0.0.1
10.0.0.6
10.0.0.3
10.0.0.4
High LQ*NLQ (> 0.66)
Medium LQ*NLQ (< 0.66 & >= 0.33)
Low LQ*NLQ (< 0.33)


Timestamp: 1466545806 (16)
Cost: 5313.16666
10.0.0.2
10.0.0.5
10.0.0.1
10.0.0.6
10.0.0.3
10.0.0.4
The best route


Timestamp: 1466545837 (17)
10.0.0.2
10.0.0.5
10.0.0.1
10.0.0.6
10.0.0.3
10.0.0.4
High LQ*NLQ (> 0.66)
Medium LQ*NLQ (< 0.66 & >= 0.33)
Low LQ*NLQ (< 0.33)


Timestamp: 1466545837 (17)
Cost: 5414
10.0.0.2
10.0.0.5
10.0.0.1
10.0.0.6
10.0.0.3
10.0.0.4
The best route


Timestamp: 1466545868 (18)
10.0.0.2
10.0.0.5
10.0.0.1
10.0.0.6
10.0.0.3
10.0.0.4
High LQ*NLQ (> 0.66)
Medium LQ*NLQ (< 0.66 & >= 0.33)
Low LQ*NLQ (< 0.33)


Timestamp: 1466545868 (18)
Cost: 5210.33333
10.0.0.2
10.0.0.5
10.0.0.1
10.0.0.6
10.0.0.3
10.0.0.4
The best route

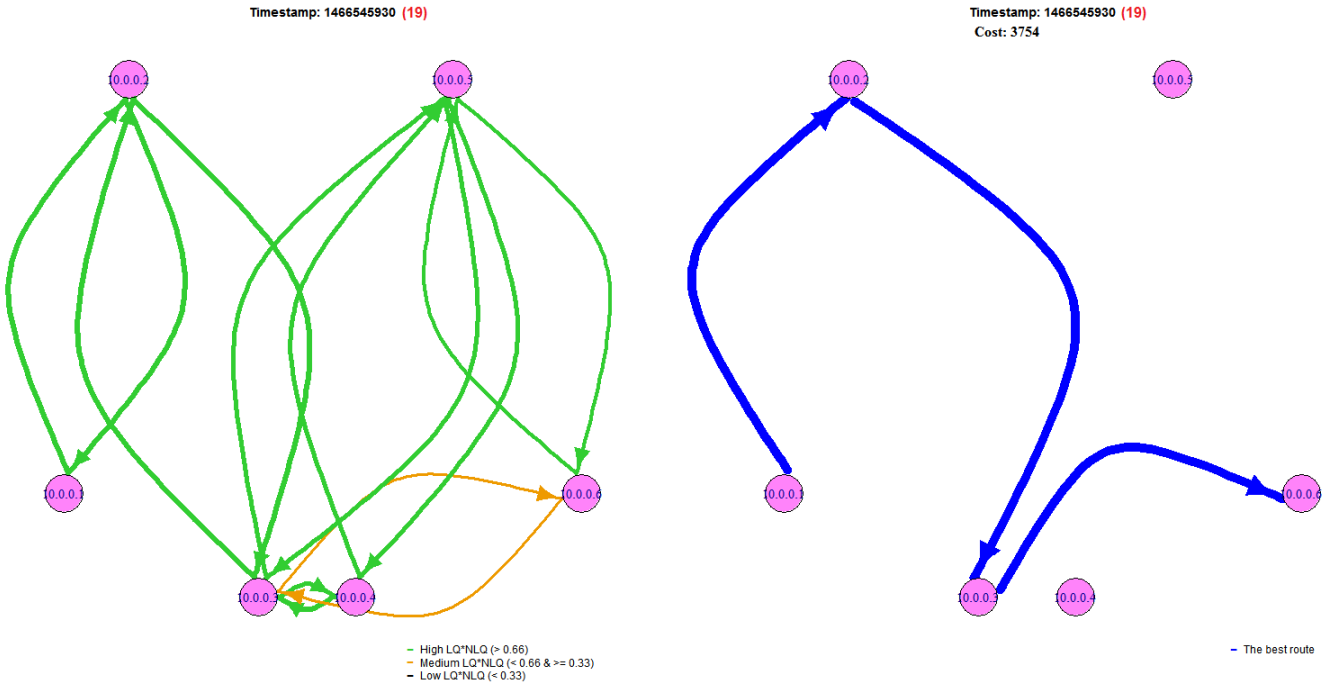


Figure 1: Left-Side – The current state of the network topology with the quality of each link
Right-Side – the best route from node 10.0.0.1 to 10.0.06

2. Evaluate the HTTP measurement data. Provide a plot of the data throughput and the total number of transferred bytes over time. Can you observe any patterns in these plots? Describe your findings and try to correlate them to the results of the topology analysis.

Answer (2):

We collect the data from measurement_HTTP_ALL.txt under DATA folder and then we generate the data fields required to plot both the data throughput and the total number of transferred bytes over time in httpMeasurementTable.txt file. Throughput was calculated by dividing the number of sent bytes by the transferred time. Total number of sent byte was accumulated during the following timestamps.

Figure 2 describes both throughput and total number of bytes sent over time. Throughput shows two peaks one reaches 5.586 KB/sec and the other goes up to 9.268 KB/sec. After the first peak the total number of bytes was sharply increased then rises gradually, the same pattern has been also noticed after the second peak.

At the beginning of the measurement, throughput fluctuates with lower values between 1.5 KB/sec and 0.5 KB/sec. Then after the first peak it decreases steadily and the same happens after the second peak.

For the **first peak**, we can see the throughput reach 5.586 KB/sec. We took this timestamp to compare with the topology diagram in exercise 2-1 and found that around timestamp 1466542136, the cost of the best route was changing significantly from 3253 to 4695; for the **second peak**, the throughput is 9.268 KB/sec at timestamp 1466545857. Around this timestamp, the best route was changing from **10.0.0.1→10.0.0.2→10.0.0.3→→10.0.0.5→10.0.0.6** to **10.0.0.1→10.0.0.2→10.0.0.3→10.0.0.6**.

As for the analysis of the first peak, it seems that the throughput is positively-correlated to the cost. However, this doesn't follow our intuition. Normally, when the throughput is higher, the cost is getting lower. We think this is the special case because the best route doesn't change (still **10.0.0.1→10.0.0.2→10.0.0.3→10.0.0.6**) either. We could assume there are some algorithms implemented in routers which influence byte-transferring. The algorithm may be like: Once routers notice that the cost of the best route is changing sharply, they will tune some parameters to let transferred bytes come back with ease. We can observe this phenomenon by looking into the two peaks: In the first peak, when the cost became higher (1400 difference), the throughput also got higher; the timestamps of the second peak are between 1466545619 and 1466545919, and the cost fluctuated within 3171 and 6036.

Based on these two peaks, we could say some methods are executed inside routers. Otherwise, it is hard to imagine why the diagram is like this.

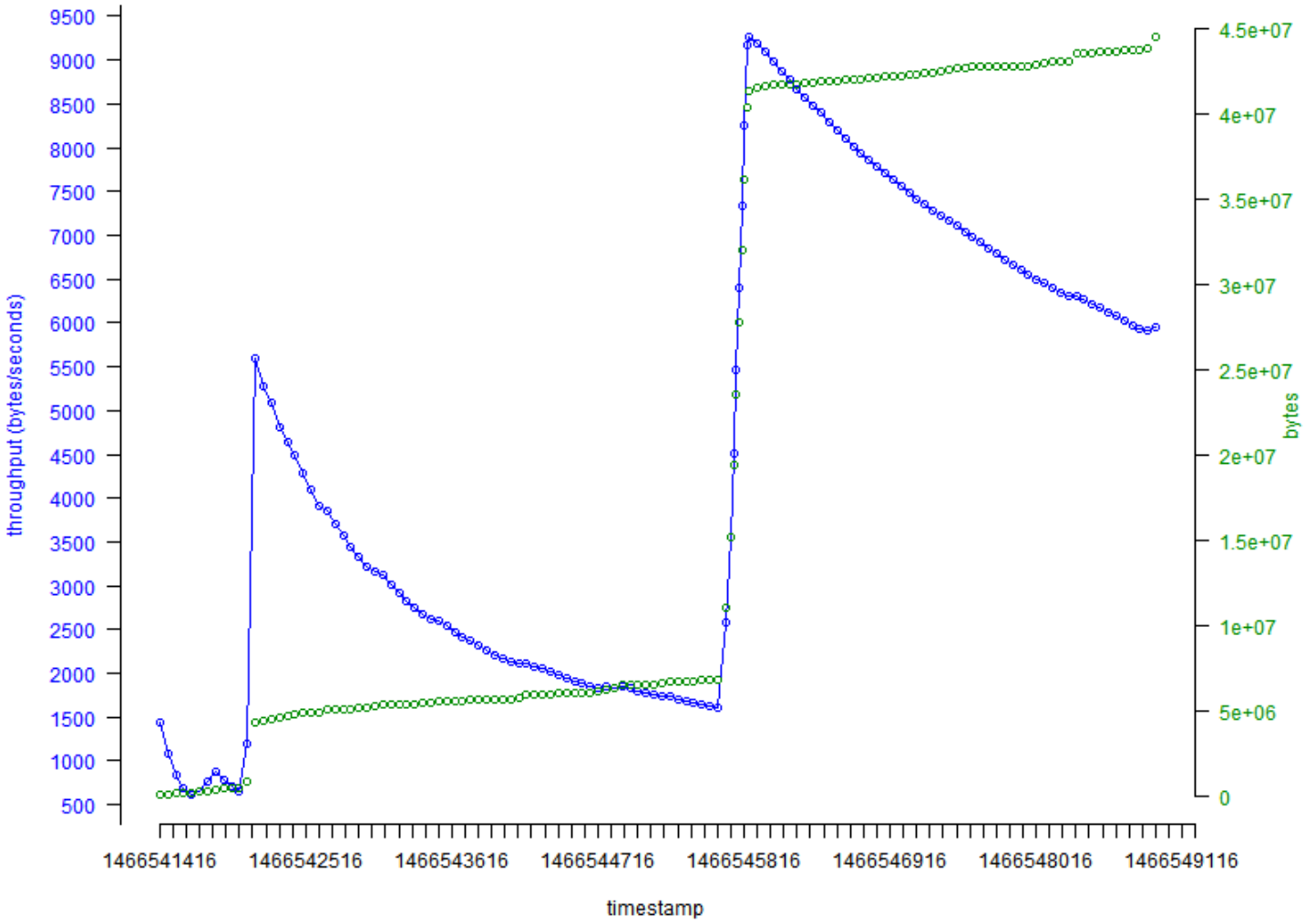


Figure 2: Network throughput and total number of bytes sent over time

3. Analyze the ping measurement data and provide plots of the end-to-end packet loss and RTT over time. Additionally, do a statistical evaluation of the individual measurement values. Based on your findings, state whether or not such a network is suitable for the following application areas: Multiplayer computer games, Voice-over-IP telephony, E-Mail

Answer (3):

In order to plot the end-to-end packet loss and Round-trip Time (RTT) over time, we employ the data from measurement_PING_ALL.txt. There we got the variables for packet loss and Min., Average, and Max. of Round-trip Time. Based on the information we generate the data fields in pingMeasurementTable.txt, source code used to do this data generation can be found in practical_2.py file under scripts folder.

We draw if there is a relation between RTT and Packet loss, we have plotted the following 3 plots:

- Figure 3: Shows packet loss % with average RTT
- Figure 4: Shows packet loss % with min. RTT
- Figure 5: Shows packet loss % with max. RTT

According to our best knowledge, we can say this wireless network is only workable in the application area such E-mail. But other application areas such Multiplayer computer games and Voice-over-IP telephony, this will not be suitable for, as listed in the following table. The reasons are because the average percentage of packet loss is around 11.14% and also the long delay and latency are noticed by dividing RRT by 2. The latter 2 applications are sensitive to both latency and packet loss, which could give bad voice quality of Voice-over-IP telephony and the same can be observed for multiplayer computer games.

For Voice over IP traffic, one commentator reckoned that "missing one or two packets every now and then will not affect the quality of the conversation. Losses between 5% and 10% of the total packet stream will affect the quality significantly." [1]

Application areas	workable
Multiplayer computer games	×
Voice-over-IP telephony	×
E-Mail	✓

Table: the suitability of the wireless network for the application areas

Statistical Measurements have been done to study the important factors of both Packet loss and RTT values (Min, Max, and Avg.). The following table describes the result of these statistical measurements. The most important thing to show here is the mean for packet loss which supports our thinking for which application areas this wireless network is suitable for (the mean of packet loss is 10%).

Statistical Measurement	Packetloss %	RTT (min) ms	RTT (avg) ms	RTT (max) ms
Mean	11.14666667	5.129493333	9.37278	35.88957
Median	10	4.957	7.571	15.4395
Minimum	2	4.295	6.166	10.068
Maximum	28	20.099	67.353	1004.179

Table: Descriptive Statistics for packet loss and different RRT values

We have study also if there is a correlation between the different packet loss and different RRT values, but according to the following table. The Correlation coefficients between the Packet loss and the other RTT values are almost zero which means that they don't have any correlations. Similarly, the same pattern of the weak correlation can be noted between the different RTT values themselves.

	Packet Loss	RTT (min)	RTT (avg)	RTT (max)
Packet Loss	1			
RTT (min)	-0.09124361	1		
RTT (avg)	0.027986527	0.584676171	1	
RTT (max)	0.091630817	0.071085958	0.406961399	1

Table: Correlation coefficients for packet loss and different RRT values

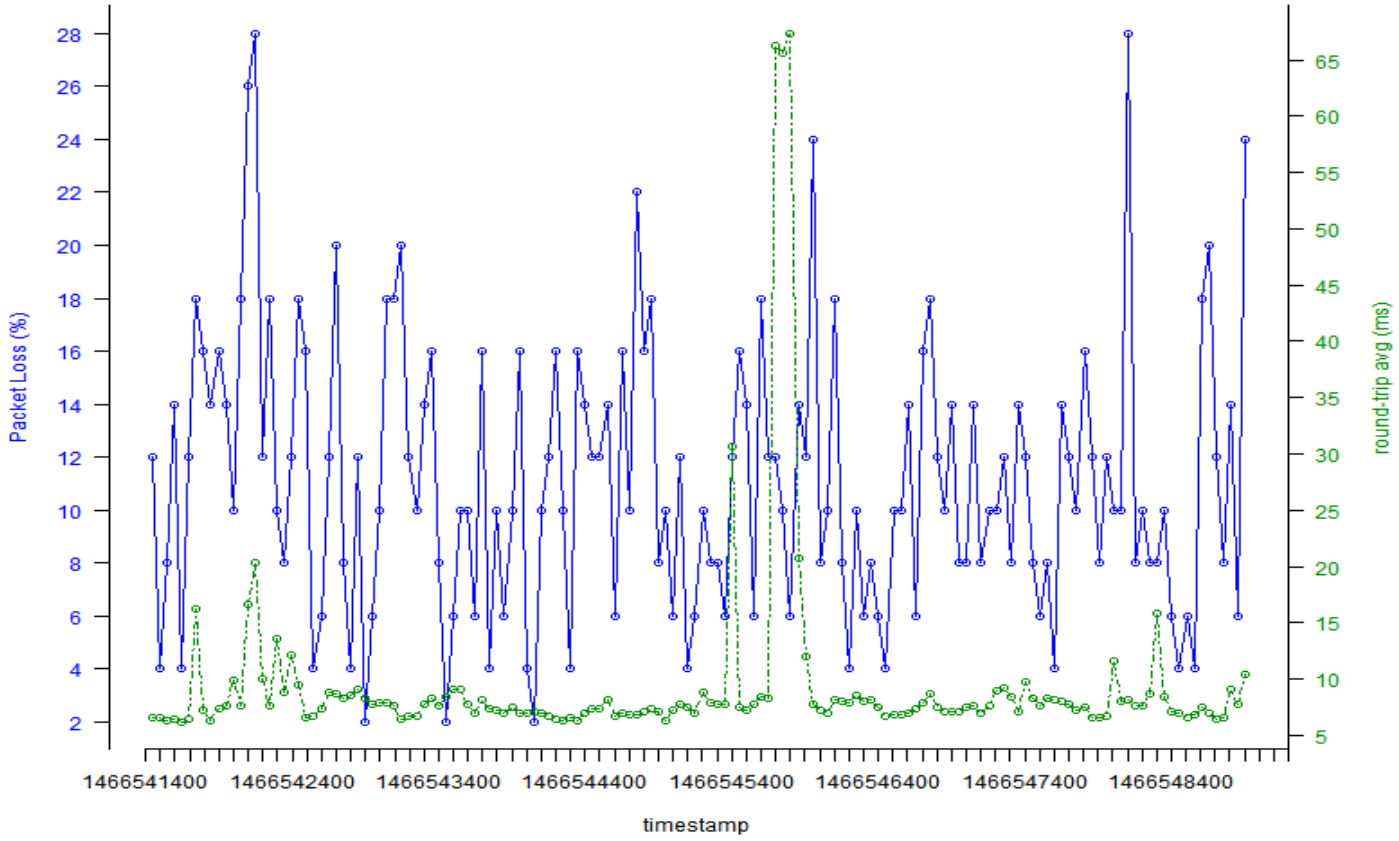


Figure 3: Percentage of Packet Loss and Average of Round-trip Time

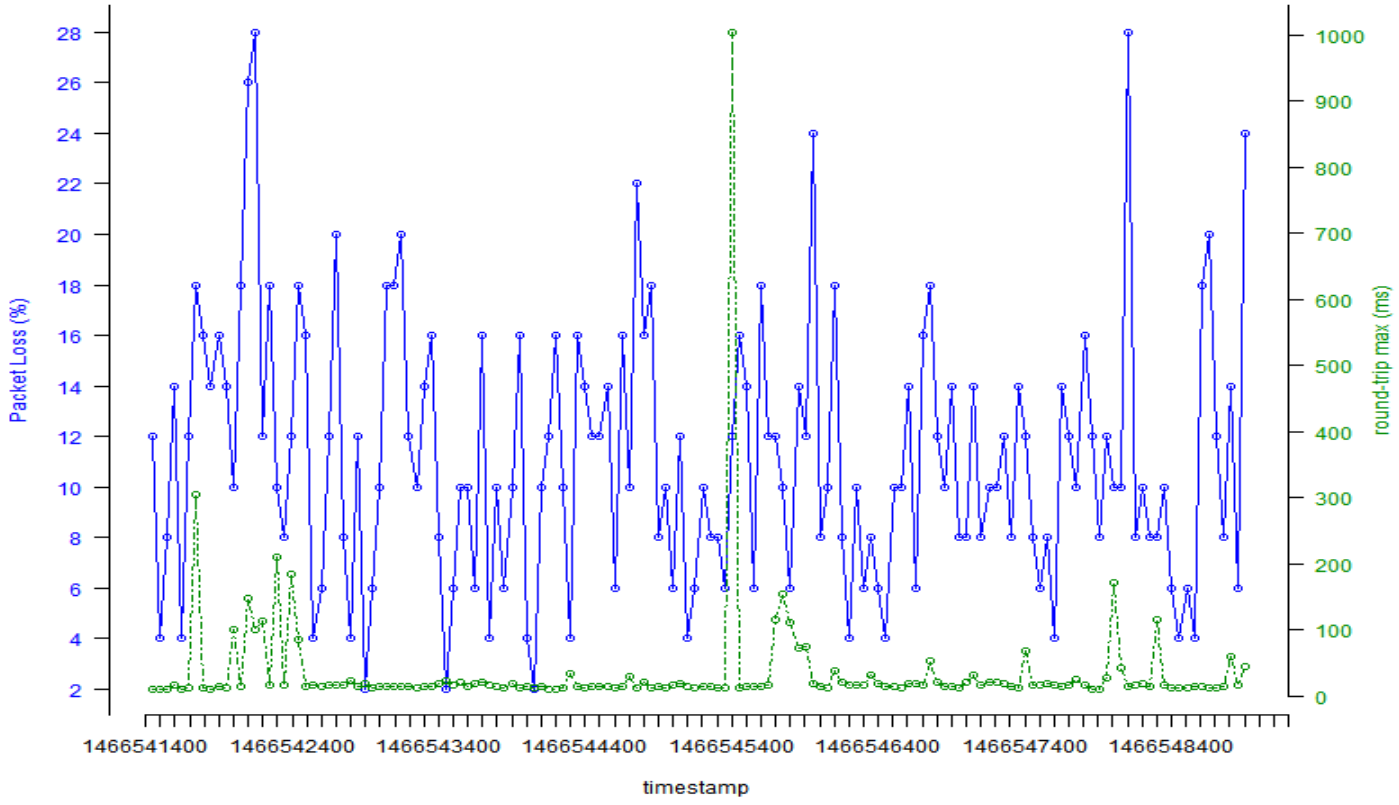


Figure 4: Percentage of Packet Loss and Max. of Round-trip Time

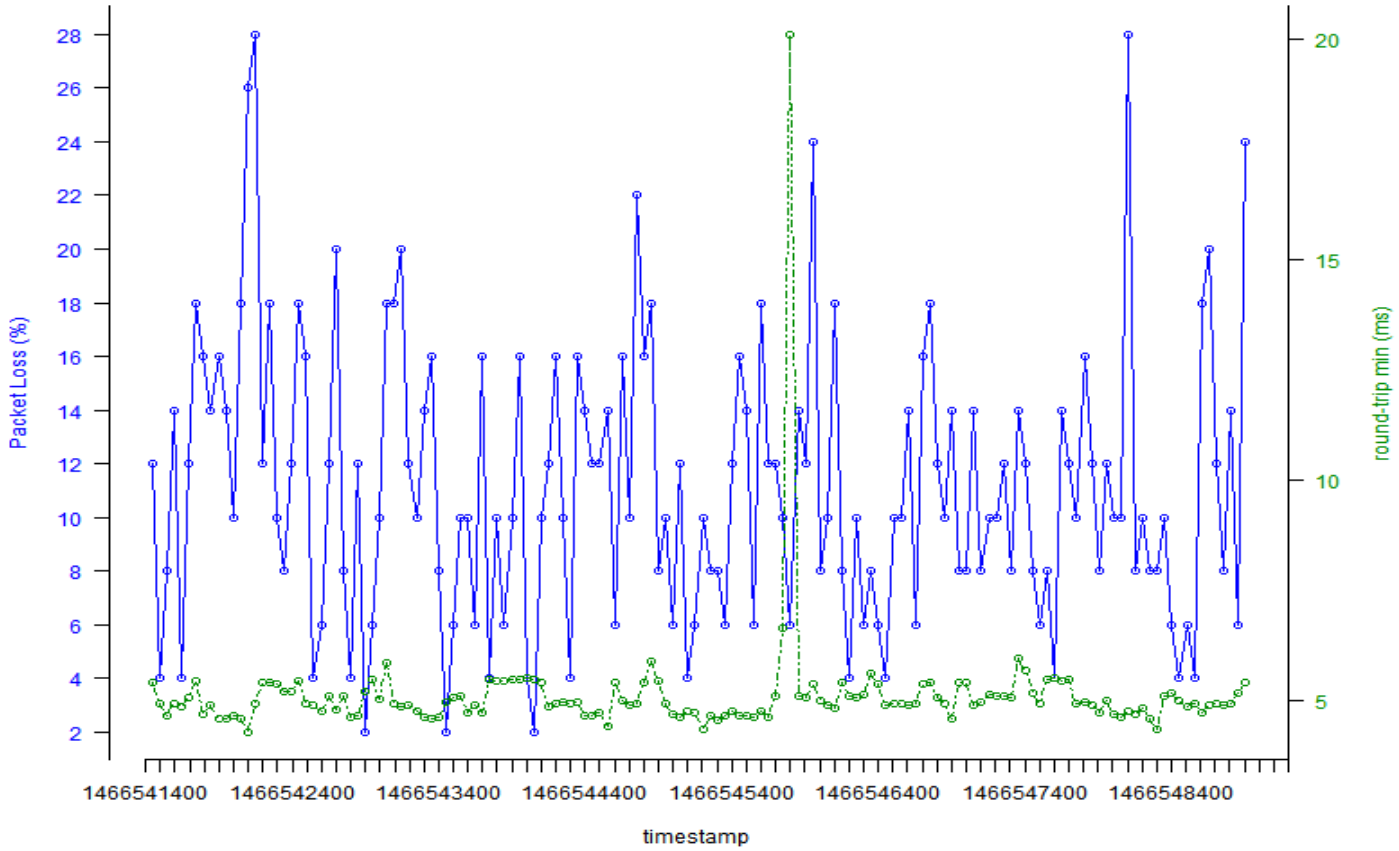


Figure 5: Percentage of Packet Loss and Min. of Round-trip Time

Reference:

1. Packet loss - https://en.wikipedia.org/wiki/Packet_loss#Acceptable_packet_loss