**Computer Science 4457A – Networks II**

**2017-18 Fall Semester**

**Group 3:**

**Jameel Kaba**

**Pearson Radu**

**Pedro Ortega**

**Table Of Contents**

**Section Page #**

**Problem Definition 1**

**Project Importance 1**

**Related Works 1**

**Comparison with Related Works 2**

**How the solution was designed 3**

**Challenges Faced 3**

**Solution Details 4**

**Data used 5**

**Overview of Our Proposed Idea to Address the Problem 10**

**Testing and Results 10**

**Future Improvements 11**

**Wishlist 11**

**Timeline 12**

**Problem Definition**

For our project we are working towards creating a program which will address the problem of proactive Quality of Service (QoS) monitoring. As such, given a network parameters, we intend on developing a prototype for proactive QoS monitoring which will analyze and sort information between two points on a network.

**Project Importance**

The problem of proactive QoS monitoring is important to network technology. By addressing this problem it will allow users to obtain more accurate information in regards to how constant and stable flow of networks which are available by IP are made to the public. This problem is also important as it can help with the need for developing connection solutions and persistence in data transfers.

For our motivation we focus on 2 key points. The first is that as technology becomes more integrated in society, aspects such as internet availability are becoming more vital for business and people to stay connected. The second point is that by using our solution, we hope to help ISPs find areas in their networks which require immediate attention, so that they can take action before users suffer from outages.

**Related Works**

Companies run and maintain different software that is similar to ours in order to provide proactive QoS monitoring. For our project we will look at two existing solutions which are used by other companies. These solutions are Synthetics by New Relic and LiveNX by LiveAction.

Synthetics by New Relic

Synthetics provides the service of QoS monitoring specifically for small companies and applications. Performance and factors such as the flow of data, packet loss, load times, etc. are show using graphs and charts. Synthetics also simulates potential traffic, giving users the ability to see the robustness of their app/website.

1

LiveNX by LiveAction

LiveNX provides proactive QoS monitoring through the use of a network topology. This topology contains the status between endpoints while allowing the user to analyze the path in order to see where problems may currently be or where problems may arise. Another feature of LiveNX is that users can preview CLI changes to see the effect it will have on the network topology before sending it to the router. Users also have the ability to see simulations of data flow through the use of specific custom files.

**Comparison with Related Works**

Synthetics v. Our Program

Differences:

Synthetics’s systems focuses more on websites performance, where their features work in giving users the ability to see content loading and average duration and effects of payload size on performance. Meanwhile, our own systems are more attuned to a general network connection and interactions between two specific endpoints on a network. Synthetics also allows for simulated browser interactions from a browser script written in modified Javascript, while we use a randomly generated range of data, set for various components in between two network points. Lastly, synthetics uses charts and graphs to display information, while we use a network topology to show our gathered information.

Similar:

Both programs use familiar sections to handle and display to the user. The first familiar component is the network status display, where synthetics shows this to the user and their website status throughout various parts of the world, our program displays similar information but about specific network regions in a more localized scale (IE: a local network IP like Bell, Rogers or start.ca stationed in a particular city). Another similarity is the detailed metrics showing where potential issues may arise from.

2

LiveNX v. Our Program

Different:

In contrast, LiveNX allows the user to preview any changes made to the router by simulating change, whereas our program does not include router change since we do not directly control the components of the network, but instead work around changes in data flow. As well, LiveNX allows the user to generate fake network traffic to view the effect on a network, while we have a predetermined set of data.

Similar:

Both LiveNX as well as our product share network QoS parameter and network status display, as well the user can use either program to create a network topology map to show the network status and QoS parameters.

**How the solution was designed**

Our solution is a Java program which provides proactive QoS monitoring for a path which is based off given parameters, considering the shortest path between two endpoints via an internet source.

Our solution was designed based on two questions.

1) How can we draw a network topology that shows the required information?

Answer: By using an open source drawing software.

2) How can we find the best path between two endpoints in a topology?

Answer: By using a shortest path algorithm such as dijkstra’s or a breadth first search.

**Challenges Faced**

Time

Programming is a time consuming and very detailed oriented task. Along with our other courses which also demanded programming and time to invest on them, we at first struggled in fitting

3

completing this task. As well, we did not expect the great amount of time required to figure out the open source program JUNG and the implementation of a Network simulation with the proper parameters

Coding (Bugs)

Initially, we had thought that JUNG was somewhat reasonable to work with, but as time went along we realized it wasn’t as friendly as we thought it was. The documentation is poorly written, and is not much help when looking further into it. Support-wise, though it was updated somewhat frequently, it is not easy to find help on certain questions as the user base is low. In retrospect, we probably should have gone with a different drawing software, or perhaps even build our own.

Testing and deciding approach

In regards to testing, it was not easy trying to find information and possible ranges in regards to the metrics that we chose. Trying to find information in regards to reasonable ranges for throughput and link utilization was a difficult task, and took almost double the amount of time to find in comparison to the rest of the metrics.

**Solution Details**

Description of how the solution was developed:

We decided to go with an Open source drawing software, as it would lessen the overall work needed to display network topology if we had to create it from scratch. We decided on JUNG, as we thought (at the time) that it seemed to be one of the easier drawing software available to modify into our network topology map. We then created a shortest path algorithm that could find the shortest path between 2 nodes via an internet source. In order to display the path taken, we once again had to alter JUNG so that the path taken was highlighted. Next, we had to incorporate the network parameters into our program. So JUNG had to be altered once more so that we can show the network status for the path taken as well as the parameters for selected points along the path.

4

How Our Solution Works

The program returns a network topology that contains the network status along a path between two endpoints. This topology is drawn using JUNG which is an open source software. The path is determined by two endpoints provided by the user.

The program will allow the user to select a starting and ending location. By going this route, the program will determine the best possible path from the first endpoint to the internet source and from the internet source to the second endpoint. This will be done by a shortest path algorithm which will try to find the shortest path between endpoints. The program will return a network topology that contains the network status along this path. The user will also be able to select vertices on the path, to see the network parameters at that point.

**Data Used**

For our data, we tried to look at things from the perspective of an ISP.

The metrics that we are focusing on are: Packet loss, Throughput, Latency, Link Utilization, and Number of Hops. Each metric is discussed in more detail below, followed by a table containing the ranges for each metric, and a brief description on why we chose these ranges.

These metrics are based on the use of a router port, TCP port 22 which is used for Secure File Transfer Protocol.

**Packet loss**

Definition: Packet loss occurs when one or more packets of data travelling across a computer network fail to reach their destination. Packet loss is generally caused by network congestion. Packet loss is measured as a percentage of packets lost with respect to packets sent.

Data: We have 3 ranges set for packet loss which are noted in the table below the discussion

5

**Link Utilization**

Definition: Link Utilization is the percentage of a network’s bandwidth that is currently being used by network traffic.

Data: There are 3 ways to calculate Link Utilization, we attempted to use method 2, for which the calculations will be discussed in more detail compared to the other methods.

1. Calculating utilization given throughput and link speed

Link Utilization = (throughput / link speed) \* 100%

1. Stop and Wait Flow Control

Link Utilization = T1 / T2

Where T1 is the Transmission Time for a Single Packet

T1 = Packet Size / Link Speed

Packet Size: Varies, but for testing purposes we’ve set it between 1000-2500

Link Speed: Depends on the connection

* + Ethernet: 10, 100, 1000
  + Wifi: 150, 300

And T2 is the Time span between beginning transmission of a packet to the time that the same sender can transmit the next packet

T2 = T1 + Propagation Delay + Acknowledgement Frame Size + Propagation Delay

Propagation Delay: distance / transmission speed

Distance is the physical distance between two points, in this case how many 100’s of miles

Transmission speed, assuming we are using fibre optic cables is 3 \* 10^8

Acknowledgement Frame Size: Depending on what type of link we are using

* + Ethernet: 46
  + Wifi: 112

6

1. Sliding Window Flow Control

IF (T1 ≥ T2) THEN Link Utilization = 100%

ELSE Link Utilization = T1 / T2

Where T1 is the Transmission Time for a Window of Packets

And T2 is the Time Span between the beginning of transmission of packet to time the same sender begins transmitting the next packet window

**Throughput**

Definition: Throughput is a measure of how many units of info system can process in a set amount of time.

Data: Throughput can be calculated in 1 of 2 ways, we will be using method 1 to find throughput

1. If Link Utilization is known then Throughput can be calculated as below

Throughput = Link Utilization \* Link Speed

Link Utilization calculated and discussed above

Link Speed can vary, based on the type of connection which is used

1. If Link Utilization is not known, then Throughput can be calculated using a Receiving Window and Round Trip Time

Throughput ≤ RWIN / RTT

Where RWIN is a TCP Receiving Window, which has max size of 524,280 bits when there is no option for window scale

And RTT is the Round Trip Time which is calculated as follows

RTT = α \* old\_RTT + (1 - α) \* new\_round\_trip\_sample

where α is constant weighting factor ( 0 ≤ α < 1 ).

Once a new RTT is calculated, it is plugged into the equation above to obtain an average RTT for that connection, this repeats for every new calculation.

7

**Latency**

Definition: Latency is the amount of time that a message takes to travel through a system. In a network, Latency is the amount of time it takes for a packet to get from a starting point to an ending point. Sometimes, Latency is described as the amount of time required for a packet to be returned to the sender.

Data: Latency is 0.82 ms for every 100 miles it must travel, as such each node is assigned a distance which is used for determining the latency.

**Number of Hops**

Definition: The number of hops refers to the number of devices which data must pass through between the source and the destination.

Data: The number of hops is used a basic measurement of distance which is used for our shortest-path algorithm. Each router passed is technically considered as 1 hop. Because the number of hops is always 1 for passing a router, it is not displayed as a metric.

**Data Ranges:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Metric** | **Good** | **Acceptable** | **Bad** |
| Packet Loss | < 1% | 1% - 5% | 5% - 10% |
| Link Utilization | < 40% | 40% - 70% | 70%+ |
| Throughput | < 50% of Link Speed | 50% - 80% of Link Speed | 80%+ of Link Speed |
| Latency | 0 ms - 80 ms | 80 ms - 170 ms | 170 ms+ |

8

**Reasoning for Data Ranges**

For Packet Loss we found that most examples and sources would have a good packet loss at less than 1%, while for an acceptable range tended to vary between 1% and 5%, and bad packet loss would be anything above the acceptable range.

In regards to the data ranges, it was difficult to find information in regards to link utilization and throughput, as there was no set clear range that is used for either metric. In regards to link utilization, we decided to aim on the ranges based on the fact that if the utilization is high it could cause network slowdown or failure, which ISPs would like to avoid. Because we are looking at low utilization, the tradeoff would be a low throughput.

We were not too sure about what could be considered as good throughput (whether it should be high or low) however our reasoning is as follows. From an ISPs perspective, the reason we would want a low throughput. For example, if a user has a router that can support up to 300 Mbps of link speed but they are only paying for say 5 Mbps, then you would not expect them to make full usage of throughput. In a way you could think of it as, based on the type of link speed that is being used, if the throughput is high then that could mean there is a lot of traffic currently on the network and as such an throughput on the ISPs side could be high. For an ISP it is better if the throughput and link utilization are low, this way in case if there is an error or an outage, it can be tended to right away with the free throughput and link utilization.

For Latency, we decided to go with ranges that are generally used for gaming. This was done, for if ISPs wanted to accommodate gamers, or perhaps offer internet with potentially lower latency. We decided to go with the latency as .82ms per 100 miles as more recent sources tended to agree on this point. There were some older sites that suggested 1ms for 100km, but we decided it would be better to use something more recent. The ranges we used for Latency are the general ranges which are used in game development that the developer expects.

9

**Overview of Our Proposed Idea to Address the Problem**

Our idea will help solve the problem by analyzing data flow from two endpoints and will be displayed to the user using a network topology diagram. This diagram shows all off the existing points on the network and their connections, as well as the status and connectivity information for the path between selected nodes.

**Testing and Results**

What we expected to see while testing

Test: The network topology displays on screen. It would be a bit of an issue if the topology would not show up.

Result: PASS.

Test: Metrics are displayed on screen when a node from the path is selected. If a node somewhere in the path did not have the metrics appear then the user would not be able to see the stats for that area which could lead to long term problems.

Result: PASS (mostly), metrics which require calculations may not be accurate, as well percentages are represented in decimal form.

Test: Path color changes based on the network status between edges. This should be shown so that the user can see the network status along the path.

Result: PASS

Test: User can select two endpoints and the program will return the path, metrics for selected nodes, and path status for the path. If the user can’t select two endpoints, then the program is literally just a display of a network topology. If the path or metrics or both are not shown, then there is an issue as the program isn’t working correctly.

Result: PASS

10

**Future Improvements**

Reaching the end of this assignment, we reflected on areas we could improve on when making further projects of this size. One particular consideration is comparative testing with a known method or solution (use another company's software). If possible, we would look to attain access to another program like Synthetics or LiveNX, where we could have done comparative testing to see how our program fairs against another and what we could improve in our program by seeing how an established QOS monitoring program works.

Another task we could have implemented would be, testing our software on an actual network as opposed to using simulated data and a predetermined network topology. By using our software on an actual network, it would allow us to see how reliable our program actually is when used in a real life scenario. With our simulated data, we can see how our program works and corrects potential problem areas and adapts the connection path, but since we have a simulated network topology map, in the real world we wouldn’t necessarily know what it would look like or what the data could potentially be as it is constantly changing and is not predetermined.

Lastly, we would consider working on more time to make the program look nice and clean. Due to time restrictions with other courses, jobs, and more, we would have made the program more concise and compact. We would do this by either using another open source program for drawing software or by making our own drawing software to make a topology map.

**Wishlist**

Below are some of the features that we wanted to implement, but unfortunately ran out of time before we could fully implement them.

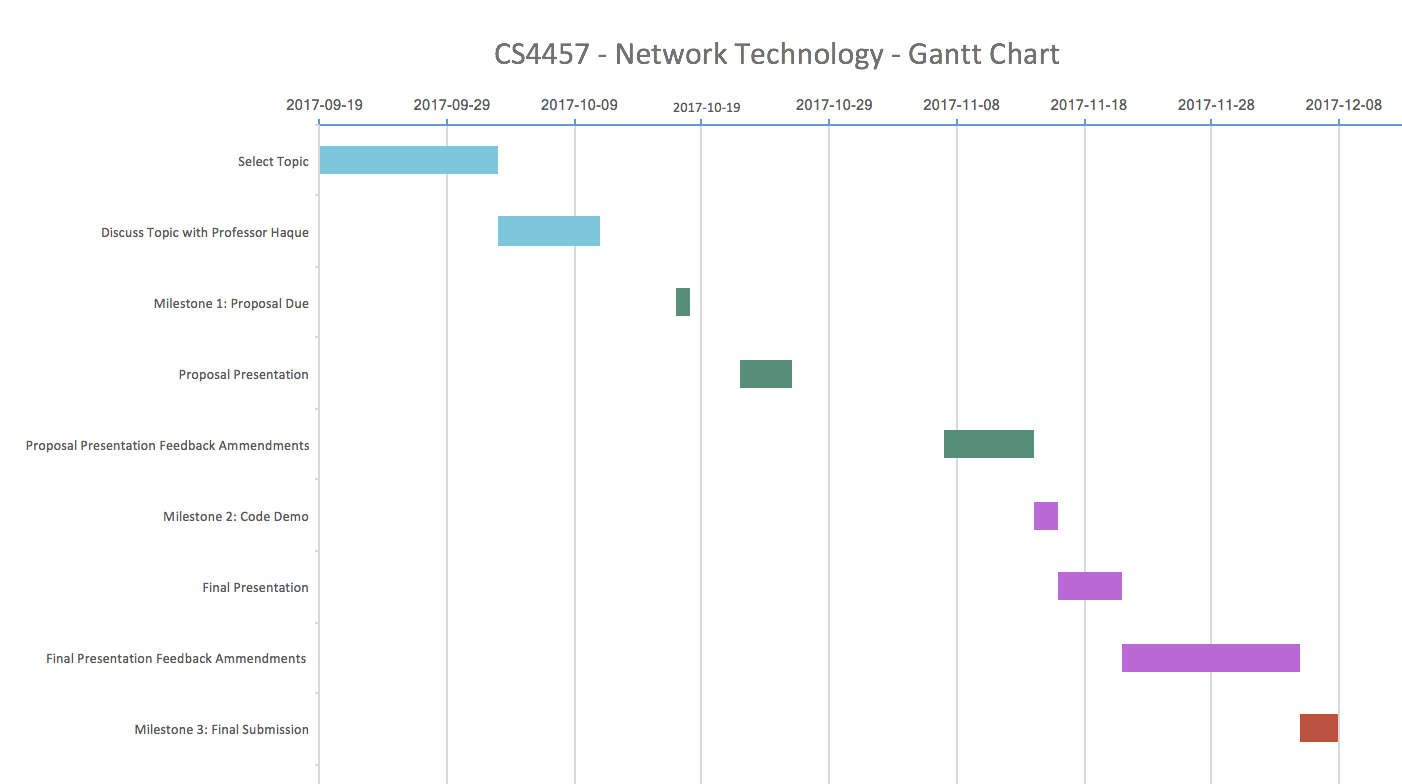
Modify the shortest path algorithm to consider the network parameters when finding the shortest path. This would have been beneficial for ISPs to see if data was being re-routed due to poor network status elsewhere.

11

Allowing the User to choose their own path, would have allowed for better usability. If a user wanted to check a specific path, they could have been able to pick and choose which routers or edges they want information about. This feature is not fully implemented as the path and metrics do not get displayed.

Ensuring that calculations are correct and that the steps/equations used are shown. This would help ensure that metric data is accurate, and that could be helpful when the user is trying to find problems. Unfortunately, due to difficulties with trying to implement the calculations in JUNG, the calculations for link utilization were unable to be implemented properly. As such, we had to settle with assigning a percentage to each node.

**Timeline**

****

12