Abstract

Network Tomography is the process of identifying delay and loss rate of the internal links of a network assuming that these internal nodes are not going cooperate. First step in this process is to identify topology. We have tools such as traceroute which could be helpful in finding topology but it relies on internal nodes replying to ICMP requests. Due to security and load reasons, ISP generally disable ICMP and so traceroute might not give us the topology fully.

This project was designed to test the thesis of Mr.Amir Malekzadeh, who proposed a new topology identifying mechanism using Traceroute sandwich probe, which relies on no cooperation from internal nodes.

This algorithm work exceptionally well when the delay along the network in monotonic. My suggestion in this project is how to take TSP in a non-monotonic delay network like internet and how to build topology in such a network.

I found that even in non-monotonic network, with this new suggestion, TSP can be used and it might result in getting the topology right. For this, I have compared results of trace route and the topology inferred.

Acknowledgements

I would like to thank Mr. Mike Macgregor, my mentor for this project for his valuable guidance and support. He did course correction whenever my project went wayward and I credit his valuable suggestions which helped me complete this project.

Since my project involved testing packets with multiple destinations, I needed destination nodes to be distributed. My friends provided their helping hands in me getting their IT resources to do this testing. I thank all of them for this invaluable help.

Table of Contents

CHAPTER 1

INTRODUCTION

There are many applications and services that require information about the internal network characteristics. Network monitoring, high quality media streaming, peer-to-peer and collaborative environments are examples of services that can benefit from knowledge of network topology, bandwidth, and traffic intensity [1, 2, 3].

Obtaining statistical data is not an easy task considering variance that we might expect because of load on the Internet. Applications work on the end nodes and much of important data is only available on the internal nodes[4]. Those who control these internal nodes do not want to share much of this information with end user. Often these internal nodes are configured in such a way that they don't respond to ICMP requests or messages[5,6,7]. Therefore Applications if they rely completely on ICMP, may not work properly over Internet.

At present the most widely used topology identification rely much on ICMP and if the internal nodes have ICMP disabled, then the topology might not be complete. Due to the problems with cooperation from internal nodes, researchers have started to work on network inference without relying on internal nodes. This approach is called network tomography.

Network Tomography is the process of inferring characteristics of a network in which we may not have absolute control. We analyze network characteristics to get a better picture of an entity that is forbidden or unknown to get more valuable information that can help us fine tune our applications or services so that it adapts to the core network in a better way.

With network tomography, we can infer bandwidth, delay and topology of the core network. We can do this either via a passive approach where we don't make any special effort to infer characteristics but use the existing packet flow and analyze it. Or else we can send some sets of

packets and make an active approach and see what these sets of packets are going through to get some vital information.

In passive approach we monitor the packets that are flowing in and out of our network and analyze the flow to get some insight about ISP or core network. Usually this approach is good as we are not creating any additional traffic just f for the sake of analysis. But the results of passive analysis may be erroneous or far from the actual ones. Also to monitor ongoing traffic, we should have some control over the network routers that are there along the path.

By taking active approach we send some packets over the network observe these packets behavior and get to know about the core. But how active can our approach be. We cannot just choke the network with too much tomography packets and hinder the network from the actual traffic flowing between users. We should carefully design an approach where the packets that are sent are either too small or too little or to put it in the other way the burden created by these packets in the network is negligible

Several tomography methods were studied and we found that it is useful in producing a logical topology with very limited information about internal nodes. Our contribution is using the idea of traceroute to get more information about the internal nodes. But when including the idea of trace route, we believe that internal nodes are not going to cooperate with ICMP messages.

Sandwich probe is another idea where a big packet is sandwiched between two small packets and we carefully monitor delay experienced by small packets when the big packet is queued. We combine these two ideas to provide a constructive method of inferring topology. Details of this probe method and how to infer topology from this method are mentioned in this report in subsequent chapters.

CHAPTER 2:

Background work and Studies

In this chapter the tomography problem is modeled and some previous works are explained in order to introduce the major directions that researchers in this filed are working on. The methods proposed to solve network tomography problems can be broken into two major steps: data collection and topology inference. During the first step some data about the network is collected by sending probe messages or monitoring the network's events. In the second step this data is used to infer the network topology or other characteristics of the network. There are several ideas in these two steps that we discuss separately.

2.1 Maximum likelihood Approach[7]

Coates et al. suggested a maximum likelihood model based on pairwise similarity function for nodes of the network [8]. For a tree with an end-node set R, there is a $|R| \times |R|$ matrix X of estimated pairwise similarities. Let Xij be a random variable parameterized by ij for any pair of end-nodes i, j. A sample $x = \{x_{ij}\}$ of X is measured. Let $p(x|)$ be the probability density function of the random variables, which means $X \sim p(x|y)$. X can come from the sandwich probes discussed in section 2.2. Setting $L(x|T)$ as

$$L(x|T) \equiv \sup_{y \in G(T)} p(x|y),$$

where $G(T)$ is the set of all possible y's for the tree T, they defined the maximum likelihood tree as

$$\hat{T}(x) = \operatorname{argmax}_{T \in \mathcal{T}(R)} \log L(x|T), \quad (2.3)$$

where $\mathcal{T}(R)$ is a forest of all trees with the set of leaves R.

Maximum-likelihood methods, e.g.,Markov Chain Monte Carlo (MCMC) idea by Cotes et al. [5], have to face the overfitting problem. Cotes et al. came up with the idea to add a penalty

parameter to their maximization criteria. But tweaking this parameter is a tricky task which is not described how to get done.

2.2 Constructive methods:

Constructive methods are simpler in general. Constructive methods gradually build up the network tree [6].

There are several proposed constructive methods; most of them have a similar bottom-up approach:

1. Choose a pair/group of nodes with the highest similarity.
2. Merge the pair/group into a new single node.
3. Update the similarities between the new node and the other nodes.
4. Continue until only one node remains.

2.3 Issues with Maximum Likelihood Approach and Constructive methods:

Problem with the above mentioned methods is that they are useful for limited types of networks, e.g., binary trees, or are less precise than maximum-likelihood methods. The current methods do not collect much information and there are some preset parameters which are needed to adjust values. But how this preset values are set and how to determine the values are not mentioned in detail.

2.4 Sandwich Probe:

One of the data collection methods discussed is called sandwich probe which is suggested by Cotes et al. [5]. The sandwich probes give estimation for the distance from the source to the common parent of each pair of receivers. Cotes et al. suggested a Maximum-Likelihood approach to infer topology using sandwich data.

It is discussed in that because of the high complexity of this approach it has to deal with the trade-off between accuracy and running resources. In this chapter I introduce a constructive method defined by Amir Malekzadeh, to find the network topology using the information from sandwich probes. This approach is much less complex than Maximum-Likelihood.

Chapter 3

Data collection using TSP

3.1 Approach:

Objective of this project is to infer topology of the network by sending some probe packets from one source to multiple destinations. We are going to infer topology by actively measuring delay up to every subsequent hops and find similarities in delay. Usually when it comes to delay measurements, there are needs for synchronization. Since we are going to measure delay only from the destination end, there is no need for synchronization between source and destination required.

For this delay measurement, Amir is proposing a new probing approach called "Traceroute Sandwich probe", here after referred to as "TSP". For TSP, we send two similar small packets sandwiched by a large packet in between. We use a similar approach to TSP, whereby we set TTL value for the large packet "Q" as 1 for the first probe. Then in subsequent probes we increment TTL value until we reach the destination.

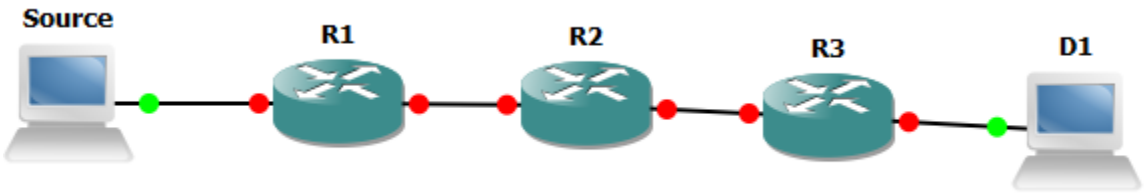
We send small packet P1 first and then after a particular duration of the time called delta, we send large packet Q and a small packet P2 immediately one after another. At the destination we find the time between small packets to find delay of the hop.

Since Q packet gets queued at every router along the path, it creates a delay for the smaller packet that immediately follows it (P2).

3.2 Set 1: Sending all packets to one destination

In the first instance, all the three packets are sent to the same destination, say D1. At the destination D1, we calculate the time difference between P2 and P1 packets. This time difference will more or less be the delay up to that specific hop. We do this delay difference calculation for every hop from source to destination. Usually to avoid spikes or anomalies, we send multiple probes per hop and then take the average to find the delay. For example consider

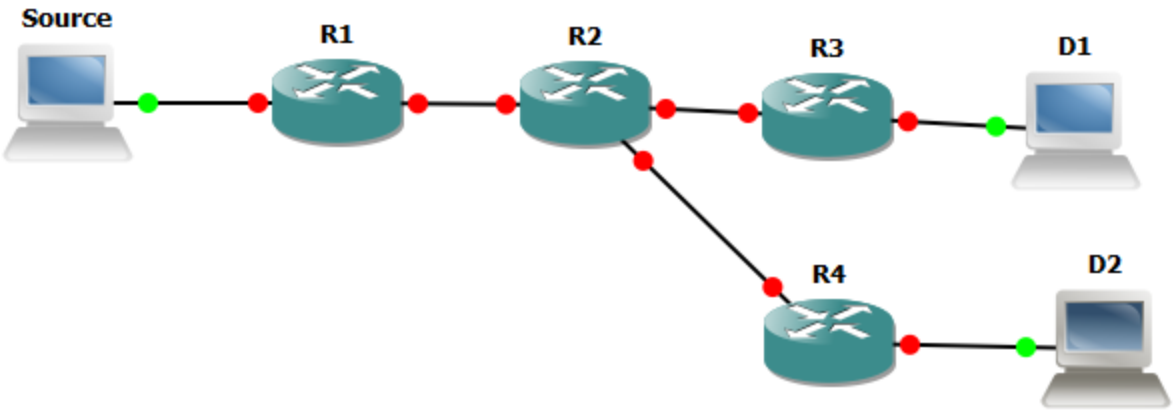
the following scenario. As explained, P1, Q and P2 packets are all sent to D1. First P1 is sent to Destination D1. After some fixed time, say 50ms, Q and P2 are sent immediately one after another. But Q packet will have TTL value 1. So when Q reaches R1, R1 decreases TTL value by 1 and so it becomes 0. R1 cannot forward it to R2. But the second small packet P2 which immediately follows Q, is delayed here in R1. So by the time P2 reaches destination, P2 would have this delay added up. At Destination D1, if we find the time difference between P2 and P1, we fill find the delay up to R1. In the next probe, TTL value for Q is incremented to 2. Then we do the same process and find delay up to R2. We continue this process until Q reaches destination. To put it other way we will TTL value is equivalent to total number of hops between source and destination, when Q reaches D1.



3.3 SET 2: Sending big packet to other destination

Since we have delay difference of every hop till destination D1, we try to find if there are any common path(s) between two destinations. So if there is another destination, say D2, we would like to find if there is any common path between D1 and D2 , if so, which node along the network is that common parent. To do this, we do the same probing scheme with a slight difference. In this case, P1 and P2 will reach Destination D1, but Q will be sent D2

Consider the following scenario:



P1 is sent to Destination D1. After some fixed time, say 50ms, Q and P2 are sent immediately one after another. But Q packet will have TTL value 1 and *is sent to Destination D2*. P2 packet which immediately follows Q, is sent to Destination D1. As explained earlier, when Q reaches R1, R1 decreases TTL value by 1 and so it becomes 0. R1 cannot forward it to R2. But the second small packet P2 which immediately follows Q, is delayed here in R1. So by the time P2 reaches destination, P2 would have this delay added up. At Destination D1, if we find the time difference between P2 and P1, we fill find the delay up to R1. This process is continued as explained in the previous case. We find delay difference for every hop(node) from source to destination D1.

We now have two sets of data. Set 1 where all the three packets are sent to D1. Set 2 where the big packet Q is sent to D2 but the smaller packets that are sandwiched by Q are sent to D1.

Chapter 4

Topology Inference

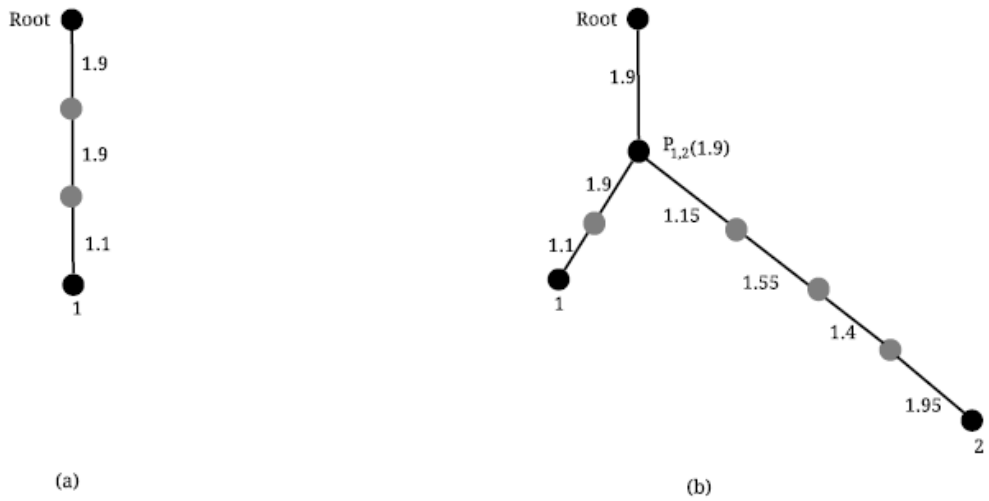
4.1 Existing Topology inference logic with TSP [10]:

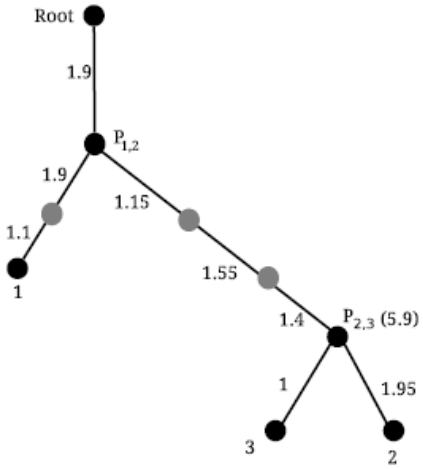
We start with a tree containing only the root, and try to add leaves to the tree one by one. In the process, we construct the internal topology as well. Assume we have a tree, which is a partial tree of the whole network. Now assume we want to add a new leaf n to the tree (see Figure 4.4). First we find the leaf n' in the tree that maximizes $x_{n',n}$. The paths from the nodes n and n' to the root have a common segment, and $x_{n',n}$ is our estimate of the delay along that segment. We need to find out at which node in the path from n' to the root these two paths separate. The delay from the separation node to the root has to be close to $x_{n',n}$. So we find the number k which minimizes

$$|x_{n',n} - y_{n',k}|,$$

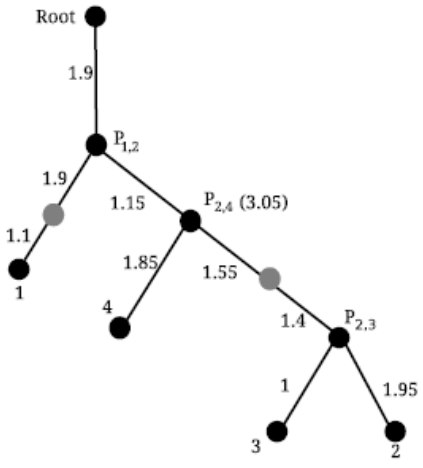
which means the k-th node in the path of the root to n' has the closest estimated delay to x.

Figure 4.1





(c)



(d)

4.2 Issues with this approach:

I found that it is practically tough to get identical delay when sending packets. For example if I send sandwich probe, where set1 in which P1,Q and P2 is sent to destination 1, Set 2 where P1 and P2 are sent to destination1 and Q is sent to destination 2, Hop by hop measurements seldom match. Also hop by hop delay is seldom monotonic.

> For example, if hop1 delay is 0.1520333529s for set 1, for set2, the same can be as drastically different as 0.1172068859s

So if our algorithm is purely dependent on matching delays for these two sets, I would have to wait for too many tests to reach a perfect data, which in internet is quite difficult. So conditions like x(I,j)=y(I,k) or |x(I,J)-Y(I,K)|should be minimum, works very rarely.

So, the TSP with the algorithm mentioned may need too much data collection for fulfilling the above conditions and in the Internet scenario if we leave the algorithm as it is, it would be a difficult task.

4.3 Suggestions and changes:

For example, consider the table below. We have two sets of data and let us call those as Set 1 and Set 2. Here we can see that there is drastic difference in delay in set1 and set 2 for hop2, even though according to the topology, it is a common link (parent). Under these circumstances, we may have to slightly change the approach to find the common parent.

Now we compare values of Set1 and Set2 for every hop. If at all two destinations D1 and D2 have some common parent, Delay up to that common parent would follow similar pattern in both Set 1 and Set2. But when the packet diverges from the common parent, we could find that P2 no longer faces the same delay in subsequent hops as the larger packet Q has already diverged through a different link and no longer delaying P2.

Let me explain this with an example of data collected:

Table : 4.1

Hop	SET 1 Delay when all packets are Sent to Destination 1 (1)	SET 2 Delay when Q is sent to Destination 2 (2)	Standard Deviation of set 1 (3)	Standard Deviation of Set 2 (4)	Difference in the delay between set1 and set 2 in a particular hop (5) = (2)-(1)
2	0.1964666605	**0.1902666728**	0.0532633912	0.0312185252	-0.0061999877
3	0.204666694	**0.1617000182**	0.057130235	0.0364482545	**-0.0429666758**
4	0.1544999838	0.1613793291	0.0382184607	0.0483745515	0.0068793453

We can see at Hop 2, Set 1 and Set 2 values are similar. But at Hop3, in Set 1 there is an increase in delay but in Set 2 there is a decrease in delay. Also the difference in delay between Set 1 and Set 2, is large when compared with Hop 2 and Hop3.

This proves that in SET2, P2 is reaching destination D1 at a faster rate and Q is no longer delaying P2 after hop2. So with this difference, we assume that Hop2 is the common parent for destination D1 and D2.

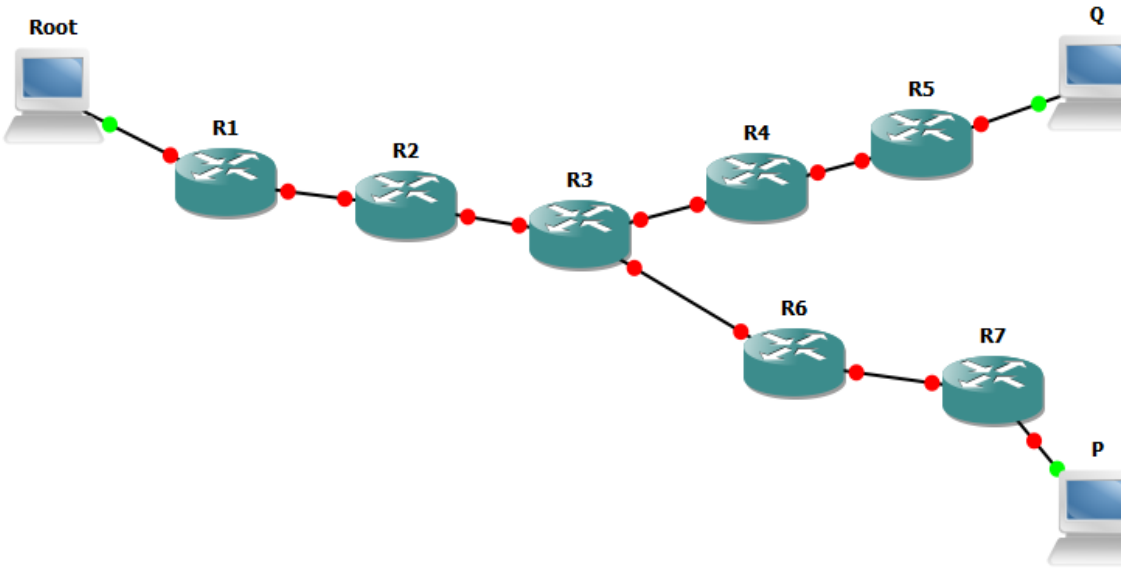
CHAPTER 5

RESULTS

5.1: Testing in Lab Network

5.1.1 Case 1 :

Actual Topology:



Summarized Data as a resultant of Traceroute sandwich probe: (Destination is 6 hops away from source)

Total No of probes sent per destination for each tests: 500

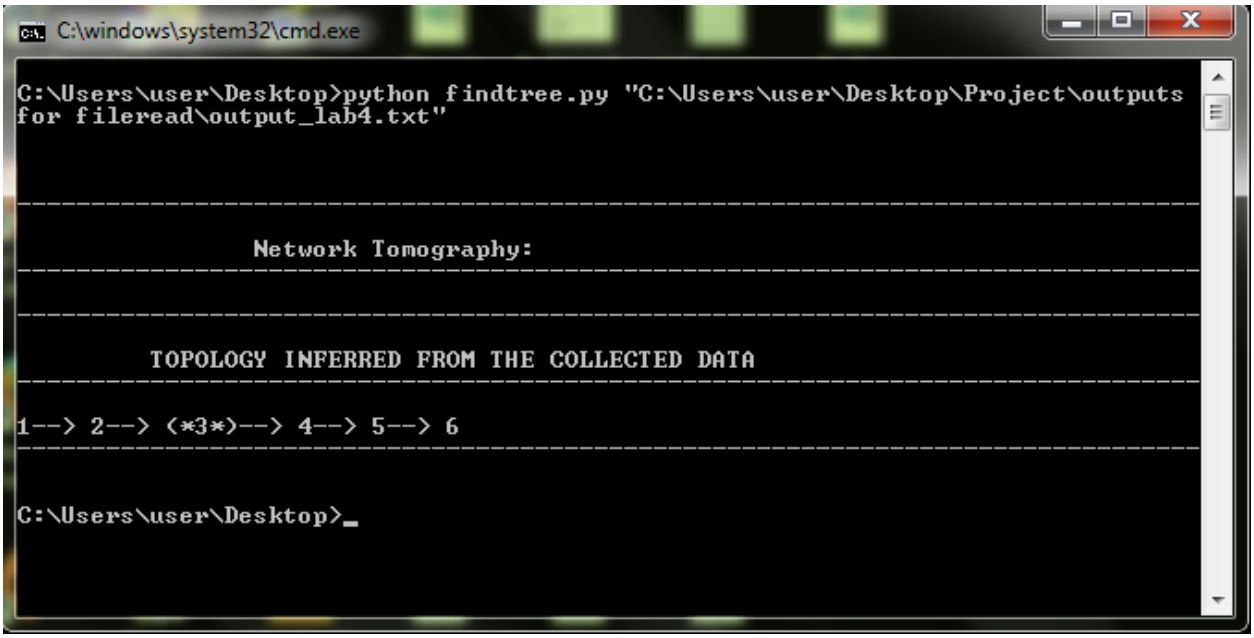
P1, Q and P2 packets are sent to destination 1(P) to collect set 1

P1 is sent to Destination 1(P), Q to Destination 2(Q) and P2 to destination 1(P) to collect set 2

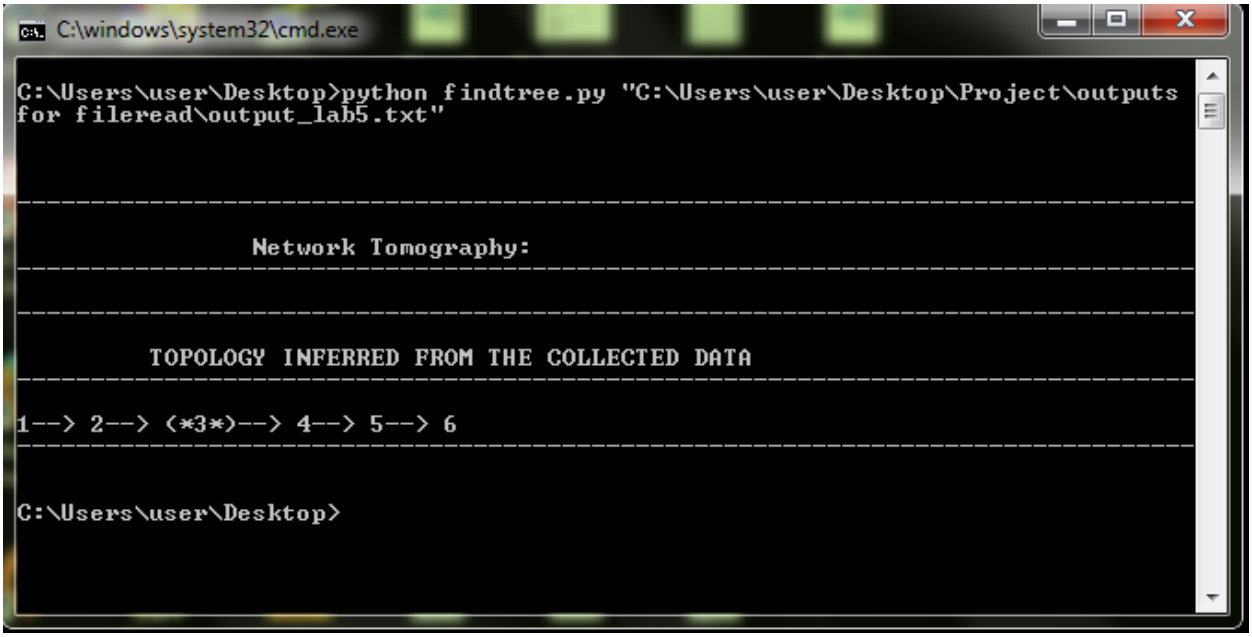
CASE1:TEST1					
Hop	Q Sent to Destination 1	Q sent to Destination 2	Standard Deviation of set 1	Standard Deviation of Set 2	Difference in the delay between set1 and set 2 in a particular hop

1	0.0766399961	0.0764779944	0.0006183809	0.0034942117	-0.0001620016
2	0.0763879967	0.0761200018	0.0006733891	0.0048634973	-0.0002679949
3	0.0763220067	0.0760540004	0.0006183224	0.00486118	-0.0002680063
4	0.0763839917	0.0760499973	0.0006168743	0.0048608128	-0.0003339944
5	0.0763659954	0.0760300045	0.0006164727	0.0048605669	-0.0003359909
6	0.076202004	0.0760620027	0.0006353772	0.0047684533	-0.0001400013

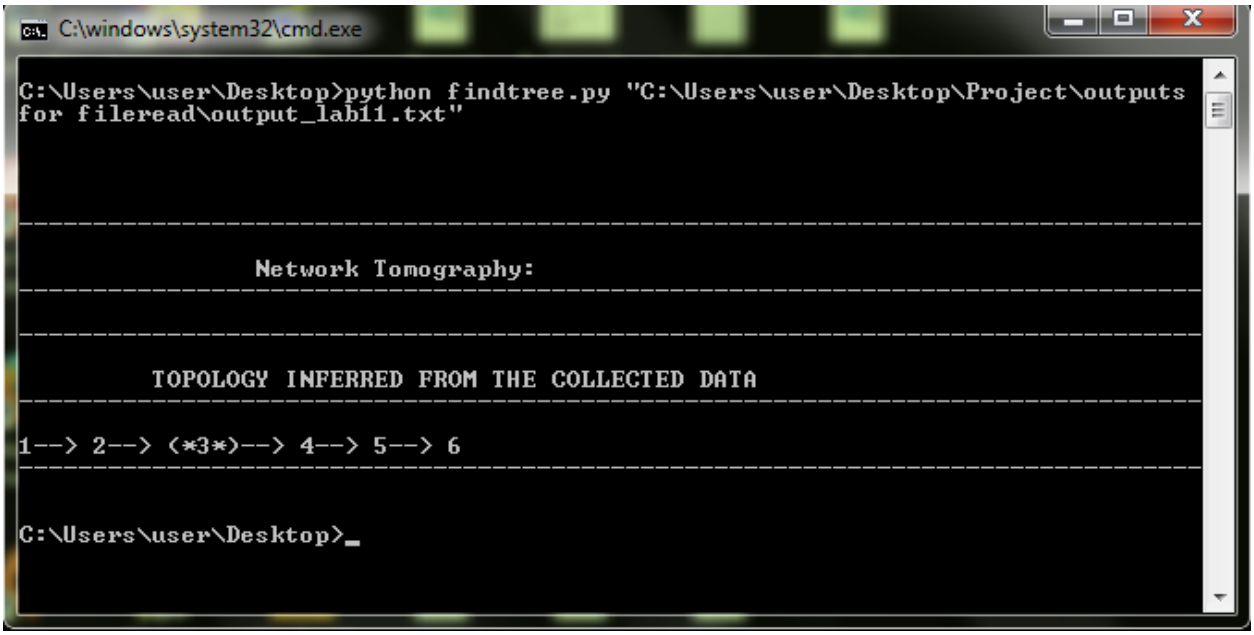
Result:



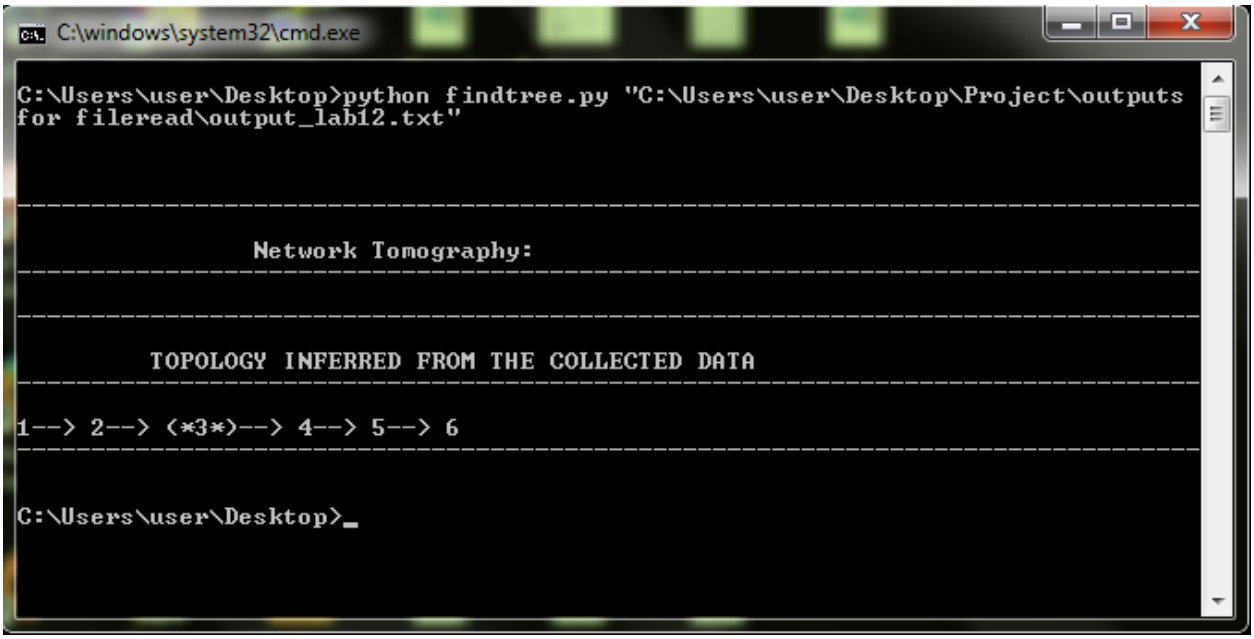
CASE 1 :TEST 2					
Hop	Q Sent to Destination 1	Q sent to Destination 2	Standard Deviation of set 1	Standard Deviation of Set 2	Difference in the delay between set1 and set 2 in a particular hop
1	0.0766259928	0.0763740029	0.0006181589	0.0035005883	-0.0002519898
2	0.0763599982	0.0760860038	0.0006086062	0.0048330759	-0.0002739944
3	0.0763339992	0.0760819993	0.000637524	0.0048640821	-0.0002519999
4	0.0763719964	0.0760180035	0.0006675403	0.0048647384	-0.0003539929
5	0.0763940015	0.0760440059	0.0006121826	0.004861078	-0.0003499956
6	0.0762200022	0.0760720077	0.0006136996	0.0047986233	-0.0001479945



CASE 1 :TEST 3					
Hop	Q Sent to Destination 1	Q sent to Destination 2	Standard Deviation of set 1	Standard Deviation of Set 2	Difference in the delay between set1 and set 2 in a particular hop
1	0.1514479966	0.1512280049	0.0006747584	0.0068278852	-0.0002199917
2	0.1512640009	0.150630003	0.0006799289	0.0095689654	-0.0006339979
3	0.1512580032	0.1506399994	0.0006748545	0.0095700785	-0.0006180038
4	0.1512820053	0.1506339974	0.0006859159	0.0095699548	-0.0006480079
5	0.1512499971	0.1506260009	0.0006924691	0.0095690189	-0.0006239963
6	0.1509900026	0.1505879998	0.0007388783	0.0095069626	-0.0004020028

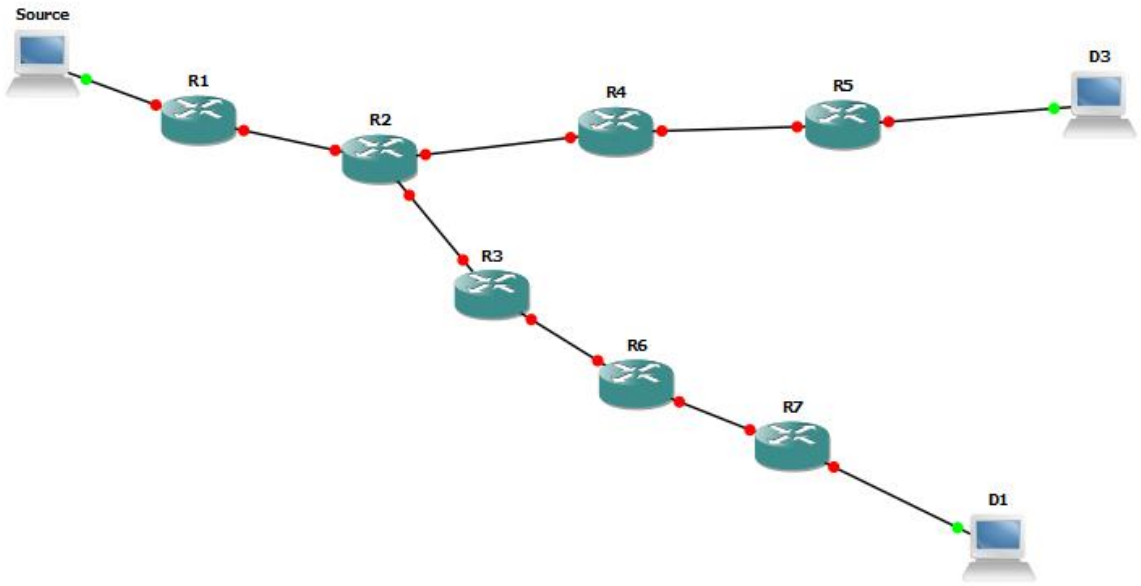


CASE 1 :TEST 4					
Hop	Q Sent to Destination 1	Q sent to Destination 2	Standard Deviation of set 1	Standard Deviation of Set 2	Difference in the delay between set1 and set 2 in a particular hop
1	0.1515119972	0.1510920029	0.0006707069	0.0069612898	-0.0004199944
2	0.151219995	0.1505560026	0.0007068303	0.0095659223	-0.0006639924
3	0.1512280111	0.1506319981	0.000681188	0.0095706101	-0.0005960131
4	0.1512480092	0.1505659986	0.0007710353	0.0095673212	-0.0006820107
5	0.1512440038	0.1505999966	0.0006902626	0.0095674443	-0.0006440072
6	0.1508499985	0.1505840001	0.0007848841	0.0095027832	-0.0002659984



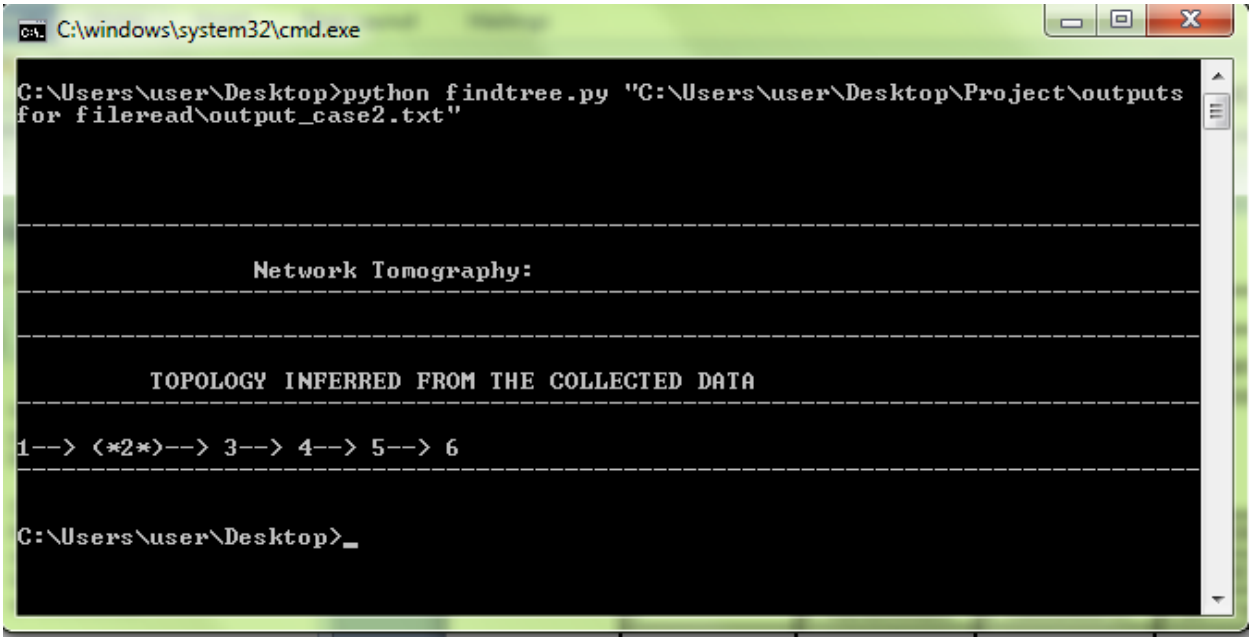
5.1.2 Case 2 :

Actual Topology:



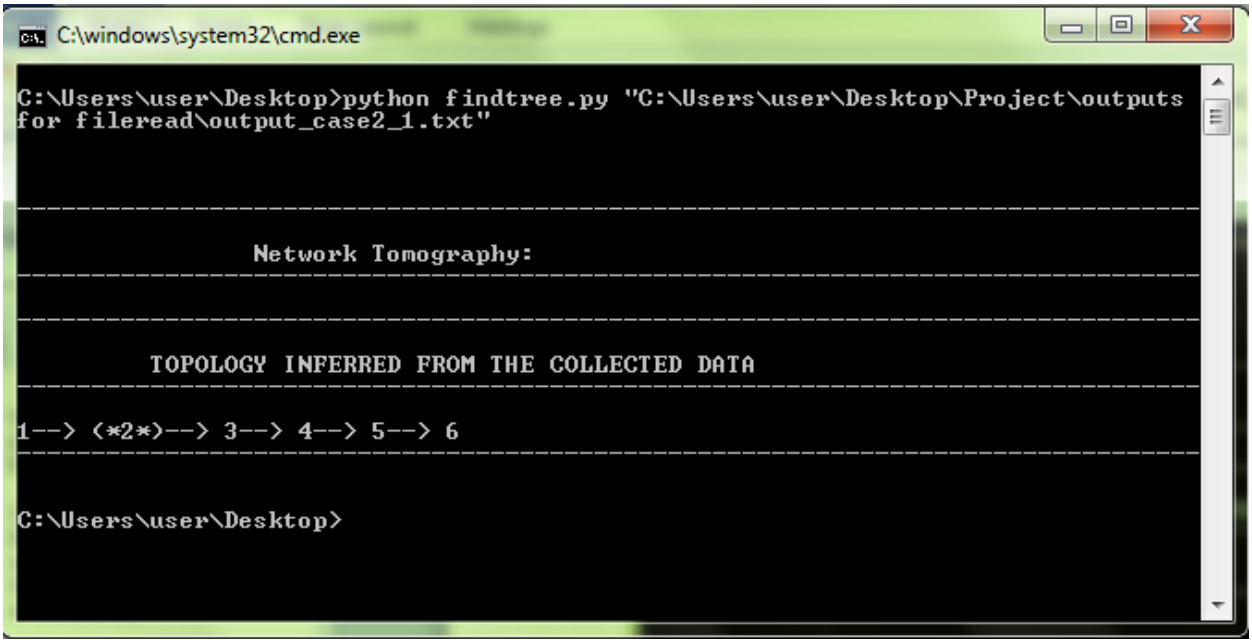
Summarized data as a result of Traceroute Sandwich probe:
Total No of probes sent per destination for each tests: 500
P1, Q and P2 packets are sent to destination 1(D1) to collect set 1
P1 is sent to Destination 1(D1), Q to Destination 3(D3) and P2 to destination 1(D1) to collect set 2

CASE2:TEST1	SET1	SET 2			
Hop	Delay when Q Sent to Destination 1 (1)	Delay when Q sent to Destination 3 (2)	Standard Deviation of set 1 (3)	Standard Deviation of Set 2 (4)	Difference in the delay between set1 and set 2 in a particular hop (5) = (2) –(1)
1	0.1531999898	0.151476007	0.0011063462	0.0068224214	-0.0017239828
2	0.1528939991	0.1509140043	0.0010652502	0.0095878364	-0.0019799948
3	0.1529200025	0.1509040017	0.0011142664	0.0095847164	-0.0020160007
4	0.1529879961	0.150940002	0.0010843743	0.009587722	-0.0020479941
5	0.1528819966	0.1509020014	0.0010198455	0.00958793	-0.0019799953
6	0.1525160079	0.1509040041	0.0011521618	0.0094880412	-0.0016120038

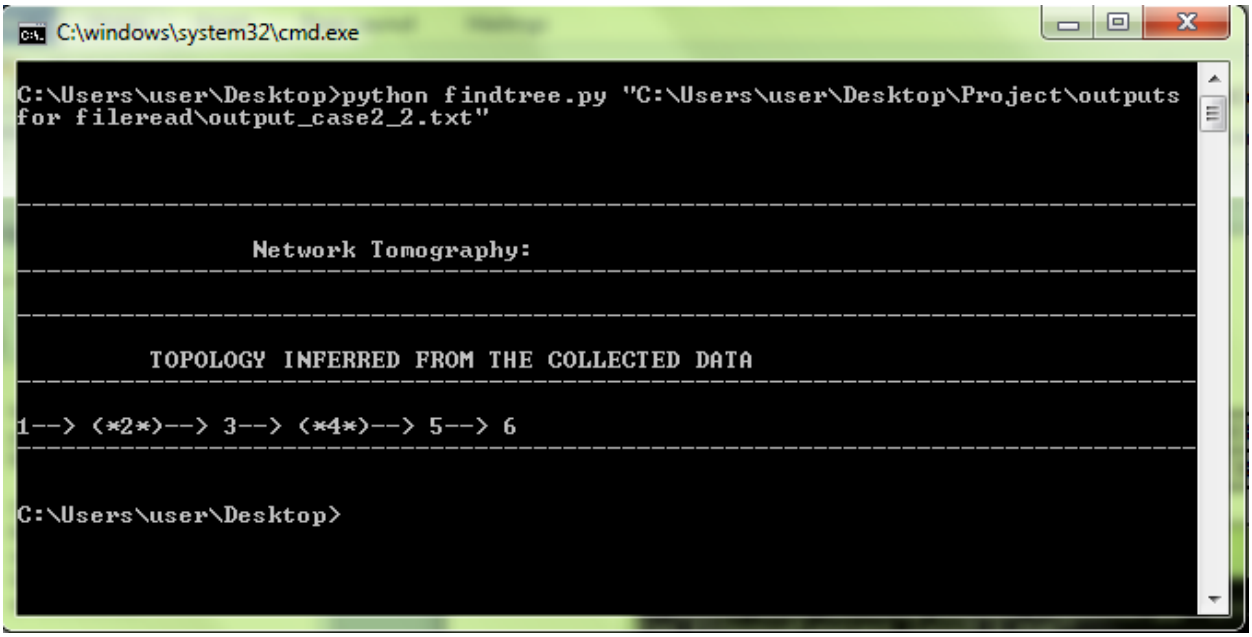


CASE2:TEST2	SET 1	SET 2			
Hop	Delay when Q Sent to Destination 1 (1)	Delay when Q sent to Destination 3 (2)	Standard Deviation of set 1 (3)	Standard Deviation of Set 2 (4)	Difference in the delay between set1 and set 2 in a particular hop (5) = (2) –(1)
1	0.1514879923	0.1511959949	0.0006941488	0.0068101086	-0.0002919974
2	0.1512540007	0.1505959969	0.0006881087	0.0095706211	-0.0006580038

3	0.151278007	0.1506520014	0.0006638591	0.0095698955	-0.0006260056
4	0.1512999983	0.150581995	0.0006767501	0.0095694967	-0.0007180033
5	0.1512700028	0.1506020002	0.000693618	0.009568469	-0.0006680026
6	0.1509440041	0.150662004	0.0007380369	0.0094777502	-0.0002820001



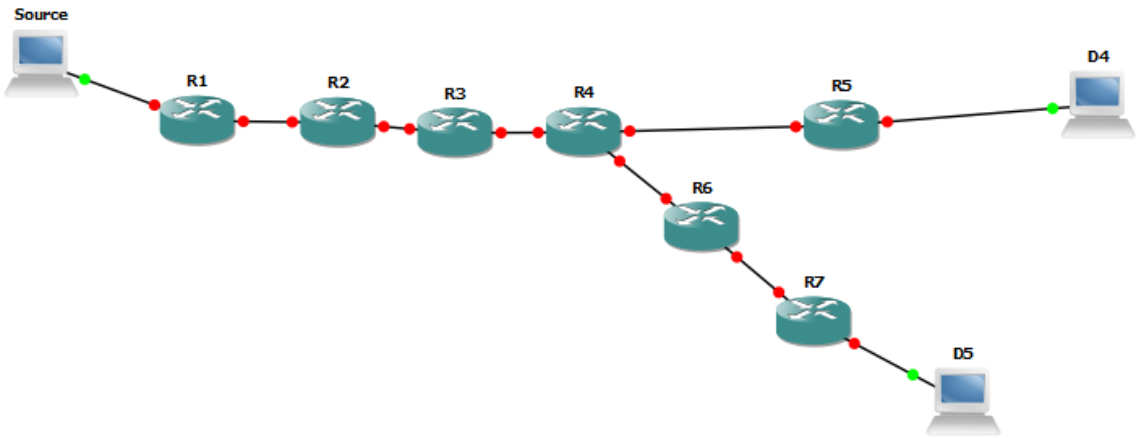
CASE2:TEST3	SET 1	SET 2			
Hop	Delay when Q Sent to Destination 1 (1)	Delay when Q sent to Destination 3 (2)	Standard Deviation of set 1 (3)	Standard Deviation of Set 2 (4)	Difference in the delay between set1 and set 2 in a particular hop (5) = (2) –(1)
1	0.1514779973	0.1511600075	0.000696796	0.0068051759	-0.0003179898
2	0.1512199945	0.150607995	0.0007152655	0.0095769683	-0.0006119995
3	0.1512620039	0.1505579977	0.000705233	0.009567373	-0.0007040062
4	0.151184001	0.1506299982	0.0007029593	0.0095681294	-0.0005540028
5	0.1512680006	0.1505799999	0.0007322411	0.0095676325	-0.0006880007
6	0.1509120002	0.1506060014	0.0007611715	0.0095048856	-0.0003059988



Note: Here based on the data, algorithm is predicting Node 4 to be a parent, but actually it is not. But based on other two tests we can find that only (2) is the parent and not (4)

5.1.3 Case 3

Actual Topology:



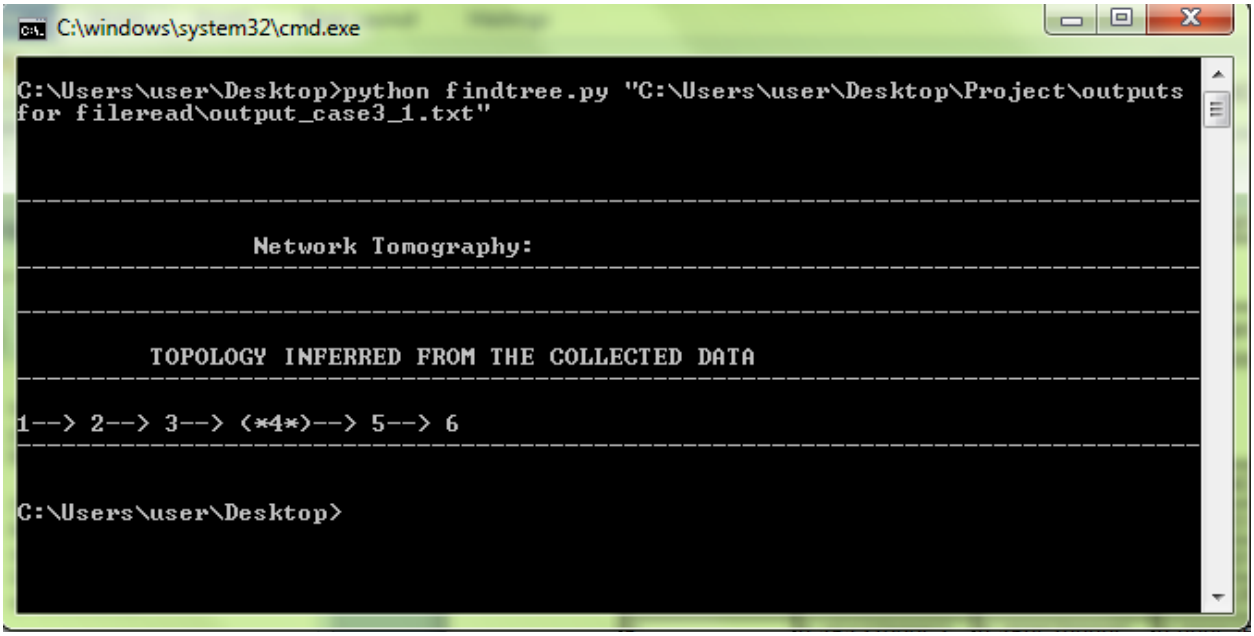
Summarized data as a result of Traceroute Sandwich probe:

Total No of probes sent per destination for each tests: 500

P1, Q and P2 packets are sent to destination 1(D4) to collect set 1

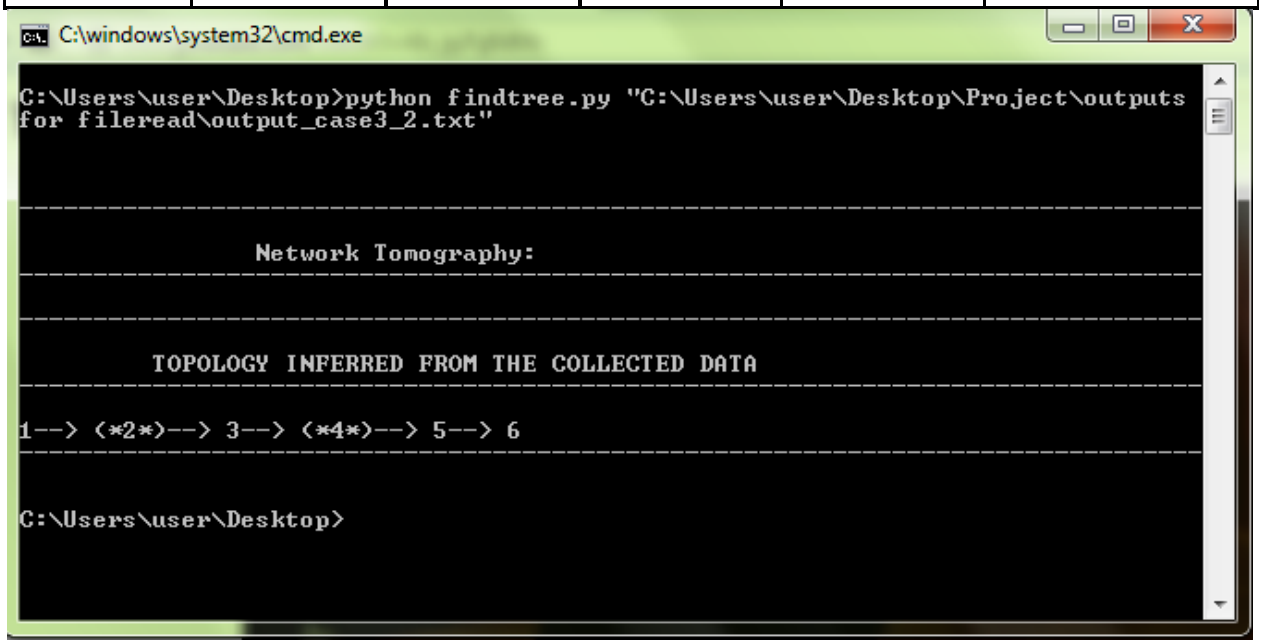
P1 is sent to Destination 1(D4), Q to Destination 3(D5) and P2 to destination 1(D4) to collect set 2

CASE3:TEST1	SET 1	SET 2			
Hop	Delay when Q Sent to Destination 4 (1)	Delay when Q sent to Destination 5 (2)	Standard Deviation of set 1 (3)	Standard Deviation of Set 2 (4)	Difference in the delay between set1 and set 2 in a particular hop (5) = (2) –(1)
1	0.1515159998	0.1512579966	0.0006432298	0.00680907	-0.0002580032
2	0.1513299975	0.1507399998	0.0006519965	0.0095779121	-0.0005899978
3	0.1513440018	0.1507399969	0.0006911287	0.009543607	-0.0006040049
4	0.1513139997	0.1507660027	0.0006288087	0.0095749277	-0.000547997
5	0.1513799963	0.1506700006	0.0006257755	0.0095735624	-0.0007099957
6	0.1510300026	0.1507300019	0.0007674434	0.0095140523	-0.0003000007

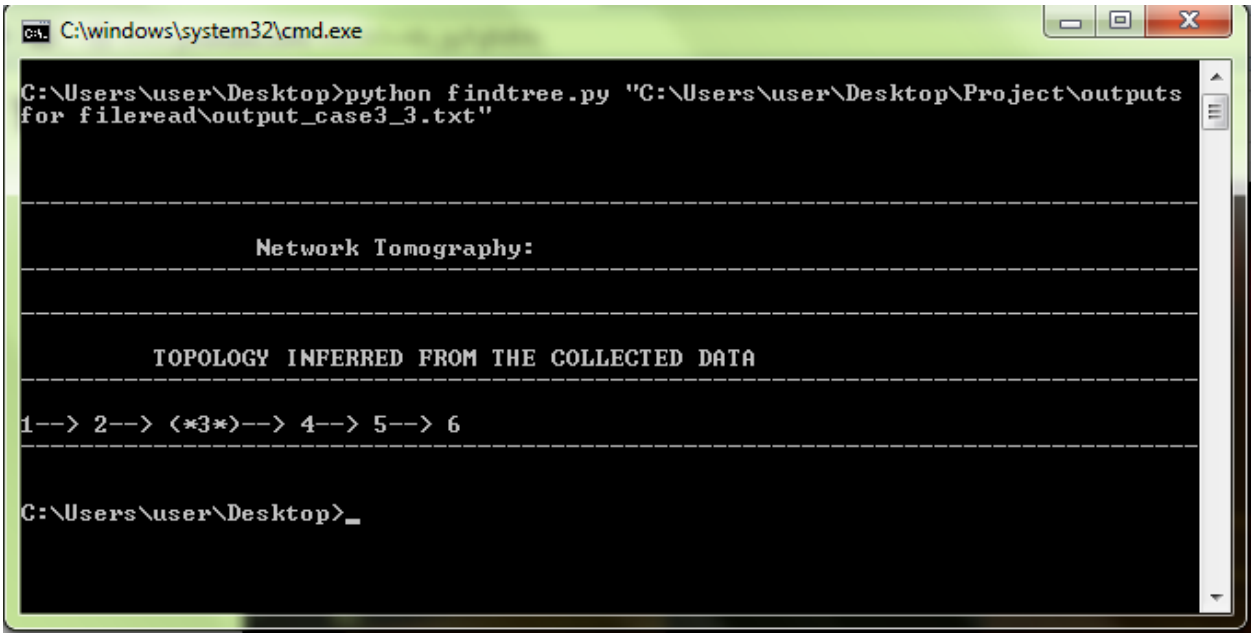


CASE3:TEST2	SET 1	SET 2			
Hop	Delay when Q Sent to Destination 4 (1)	Delay when Q sent to Destination 5 (2)	Standard Deviation of set 1 (3)	Standard Deviation of Set 2 (4)	Difference in the delay between set1 and set 2 in a particular hop (5) = (2) –(1)
1	0.151579999	0.1512539992	0.0006630228	0.0068062822	-0.0003259997
2	0.1513179955	0.1507259984	0.0006639856	0.0095747024	-0.0005919971

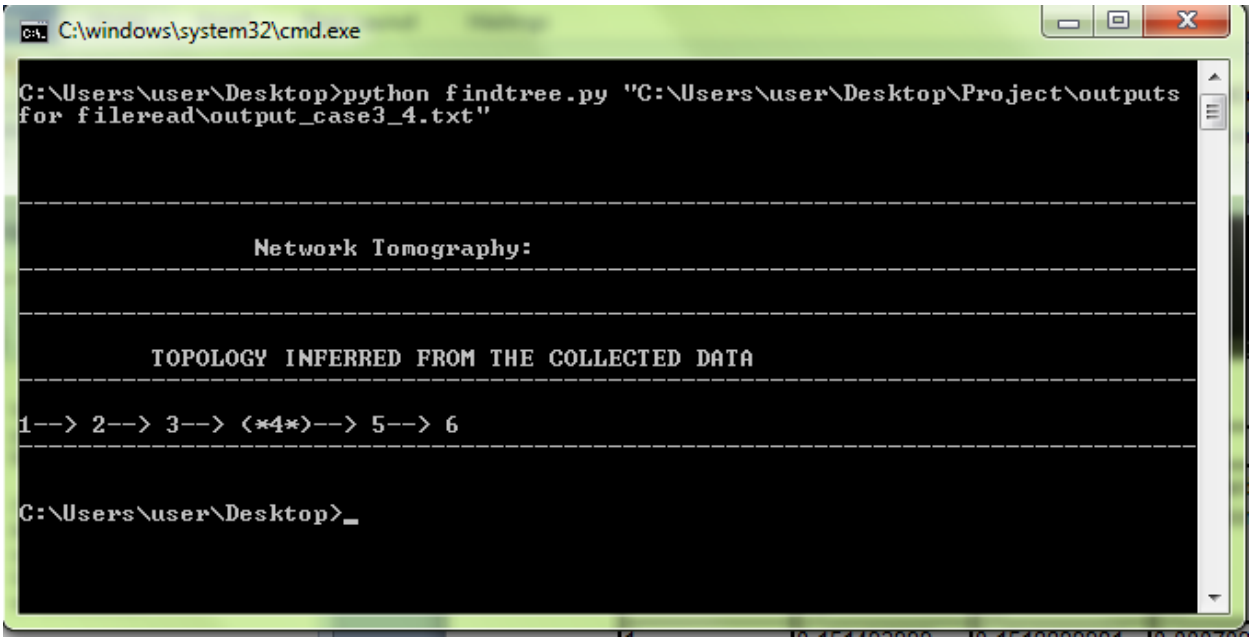
3	0.1513279972	0.1506960015	0.0006666445	0.0095716035	-0.0006319957
4	0.1512999978	0.1507300034	0.0006434285	0.0095741899	-0.0005699944
5	0.1513799973	0.1507039971	0.0006808839	0.0095747777	-0.0006760001
6	0.1509600015	0.1506939998	0.000727021	0.0094796808	-0.0002660017



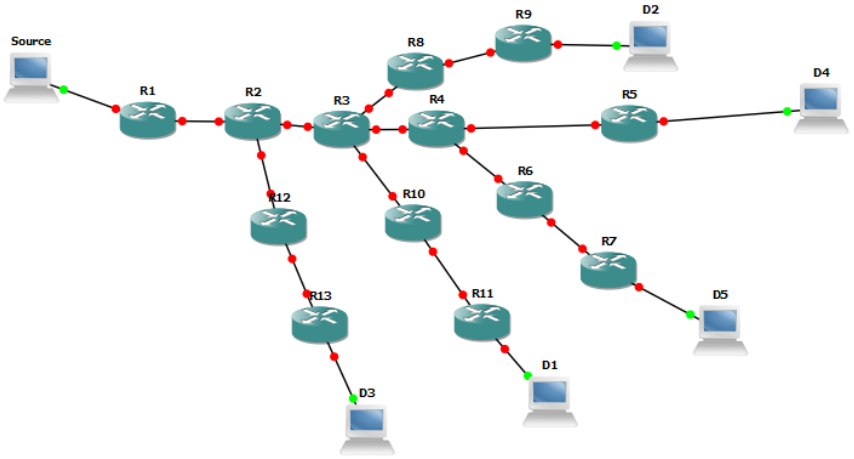
CASE3:TEST3	SET 1	SET 2			
Hop	Delay when Q Sent to Destination 4 (1)	Delay when Q sent to Destination 5 (2)	Standard Deviation of set 1 (3)	Standard Deviation of Set 2 (4)	Difference in the delay between set1 and set 2 in a particular hop (5) = (2) –(1)
1	0.1515099978	0.1511900024	0.0006495399	0.0068069007	-0.0003199954
2	0.1513619981	0.1507279987	0.0006564741	0.0095734016	-0.0006339993
3	0.1513740001	0.1507560019	0.0006527853	0.0095772894	-0.0006179981
4	0.1513940015	0.1507239919	0.0006501978	0.0095745402	-0.0006700096
5	0.1513800025	0.1506800027	0.0006446728	0.0095731707	-0.0006999998
6	0.1509800053	0.1507239952	0.0007250402	0.0095093508	-0.0002560101



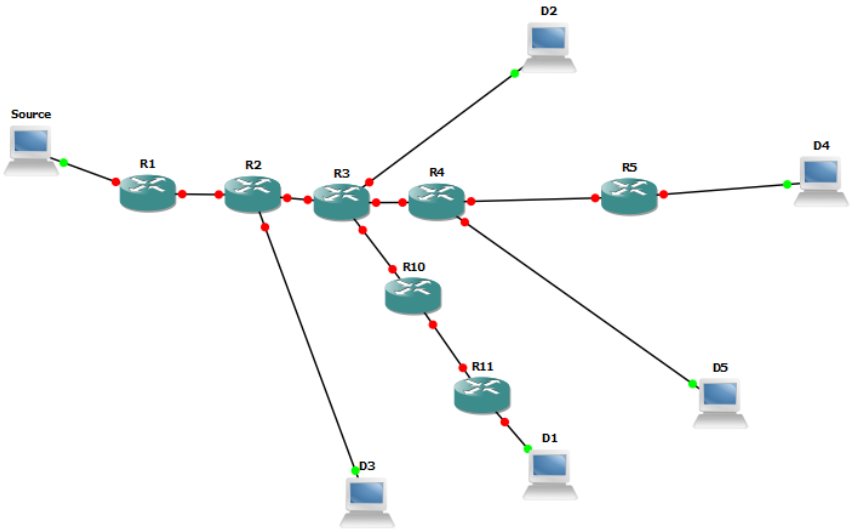
CASE3:TEST4	SET 1	SET 2			
Hop	Delay when Q Sent to Destination 4 (1)	Delay when Q sent to Destination 5 (2)	Standard Deviation of set 1 (3)	Standard Deviation of Set 2 (4)	Difference in the delay between set1 and set 2 in a particular hop (5) = (2) –(1)
1	0.151493999	0.1512299981	0.0007028275	0.0068059602	-0.0002640009
2	0.1513660026	0.1508500032	0.0006197135	0.0081576669	-0.0005159993
3	0.151357996	0.1506599917	0.0006494821	0.0095724815	-0.0006980042
4	0.1512940001	0.1507439976	0.0006508249	0.0095759316	-0.0005500026
5	0.1513359957	0.1507060032	0.0006473797	0.0095412513	-0.0006299925
6	0.1510240035	0.1507199988	0.0007175818	0.0095115479	-0.0003040047



5.1.4 Actual combined topology of all cases:



Topology inferred by TSP, By sending probes from source to
D1,D2 and D1,D3, measuring from D1
D4 and D5, measuring from D4



5.2 Testing over ISP Network:

Source : a PC in Chennai, India

Destinations

1: A PC in Garneau, Edmonton, Alberta

2. A PC in Mckernan, Edmonton Alberta

3. A PC in Millwoods, Edmonton Alberta

4. A PC inside UofA campus public network

5.2.1 Case 1:

Packet sent from India to Destination 1(Garneau) and Destination 3(Millwoods)

IP address of Garneau : 96.52.102.25

IP Address of Millwoods: 68.150.144.25

Trace route results:

C:\Users\pad>tracert 96.52.102.25 ←Traceroute to destination 1

Tracing route to s0106c8fb26592a52.ed.shawcable.net [96.52.102.25]
over a maximum of 30 hops:

```
 1    13 ms     3 ms     4 ms  D-Link.Home [192.168.1.1]
 2   486 ms   245 ms   140 ms  59.92.64.1
 3   406 ms   175 ms   107 ms  218.248.235.213
 4   319 ms   167 ms   128 ms  218.248.235.142
 5   238 ms   146 ms   123 ms  14.141.139.145.static-chennai.vsnl.net.in [14.141.139.145]
 6   218 ms   167 ms   126 ms  172.25.81.134
 7     *       84 ms    48 ms  ix-0-100.tcore1.mlv-mumbai.as6453.net [180.87.38.5]
 8   448 ms     *      193 ms  if-9-5.tcore1.wyn-marseille.as6453.net [80.231.217.17]
 9   490 ms   273 ms   152 ms  if-8-1600.tcore1.pye-paris.as6453.net [80.231.217.6]
10   535 ms   252 ms   246 ms  80.231.154.86
11   252 ms   248 ms   249 ms  prs-bb2-link.telia.net [213.155.131.18]
12   326 ms   248 ms   249 ms  nyk-bb2-link.telia.net [62.115.137.78]
13   291 ms   290 ms     *     chi-bb1-link.telia.net [80.91.248.196]
14   290 ms   352 ms   291 ms  shawbusiness-ic-300303-chi-eqx-i1.c.telia.net [62.115.12.82]
15   372 ms   306 ms   308 ms  66.163.75.118
16   334 ms   324 ms   327 ms  rc1we-tge0-6-0-11.ed.shawcable.net [66.163.77.66]
17   428 ms   331 ms   329 ms  dx3ld-te3.ed.shawcable.net [64.59.184.254]
18     *        *     ^C
```

C:\Users\pad>tracert 68.150.144.25 ← Traceroute to Destination 3

```
Tracing route to s0106602ad0719ce5.ed.shawcable.net [68.150.144.25]
over a maximum of 30 hops:

 1    7 ms     5 ms     4 ms  D-Link.Home [192.168.1.1]
 2   207 ms    17 ms    28 ms  59.92.64.1
 3    21 ms    19 ms    27 ms  218.248.235.141
 4    20 ms    24 ms    33 ms  218.248.235.142
 5    20 ms    19 ms    34 ms  14.141.139.145.static-chennai.vsnl.net.in [14.141.139.145]
 6    58 ms    56 ms    57 ms  172.25.81.134
 7    52 ms    48 ms     *     ix-0-100.tcore1.mlv-mumbai.as6453.net [180.87.38.5]
 8   153 ms     *      154 ms  if-9-5.tcore1.wyn-marseille.as6453.net [80.231.217.17]
 9   159 ms   156 ms   159 ms  if-8-1600.tcore1.pye-paris.as6453.net [80.231.217.6]
10   257 ms   276 ms   256 ms  80.231.154.86
11   258 ms   278 ms   254 ms  prs-bb2-link.telia.net [213.155.131.16]
12   271 ms   265 ms   303 ms  nyk-bb2-link.telia.net [213.155.135.7]
13   297 ms   299 ms   295 ms  chi-bb1-link.telia.net [213.155.131.243]
14   297 ms   296 ms   298 ms  shawbusiness-ic-300304-chi-eqx-i1.c.telia.net [62.115.12.86]
15   312 ms   310 ms   314 ms  rc2nr-hge0-9-0-0.wp.shawcable.net [66.163.77.201]
16   332 ms   334 ms   327 ms  rc1we-tge0-6-0-3.ed.shawcable.net [66.163.76.109]
17   339 ms   335 ms   338 ms  dx2ni-te3.ed.shawcable.net [64.59.184.150]
18     *     ^C
C:\Users\pad>
```

C:\Users\pad>tracert 68.150.144.25 ← Traceroute to Destination 3 (Again)

Tracing route to s0106602ad0719ce5.ed.shawcable.net [68.150.144.25]
over a maximum of 30 hops:

```
 1    4 ms     4 ms     4 ms  D-Link.Home [192.168.1.1]
 2   44 ms    24 ms    21 ms  59.92.64.1
 3   69 ms    20 ms    69 ms  218.248.235.213  ← 3rd hop is sometimes same for D1 and D3
 4  289 ms   180 ms   148 ms  218.248.235.142
 5  297 ms   145 ms    38 ms  14.141.139.145.static-chennai.vsnl.net.in [14.141.139.145]
 6  276 ms   157 ms    66 ms  172.25.81.134
 7    *       51 ms    56 ms  ix-0-100.tcore1.mlv-mumbai.as6453.net [180.87.38.5]
 8  421 ms   222 ms   153 ms  if-9-5.tcore1.wyn-marseille.as6453.net [80.231.217.17]
 9  491 ms   347 ms   198 ms  if-8-1600.tcore1.pye-paris.as6453.net [80.231.217.6]
10  652 ms   459 ms   257 ms  ^C
```

C:\Users\pad>tracert 96.52.102.25 ← Traceroute to Destination 1

Tracing route to s0106c8fb26592a52.ed.shawcable.net [96.52.102.25]
over a maximum of 30 hops:

```
 1    7 ms     1 ms    19 ms  D-Link.Home [192.168.1.1]
 2  145 ms   108 ms   172 ms  59.92.64.1
 3   43 ms    37 ms    29 ms  218.248.235.133 ← 3rd hop is different!!!!
 4   25 ms    20 ms    25 ms  218.248.235.142
 5  143 ms    40 ms   109 ms  14.141.139.145.static-chennai.vsnl.net.in [14.141.139.145]
 6  320 ms   186 ms   138 ms  ^C
C:\Users\pad>tracert 96.52.102.25
```

Note:

1. What we can infer from traceroute is that after hop2, packet may take a different path to reach destination 1 and 3, Or may travel the same path and reach hop3, and so on.
2. At Hop 10, there is again a common parent and then it diverges from hop 11.

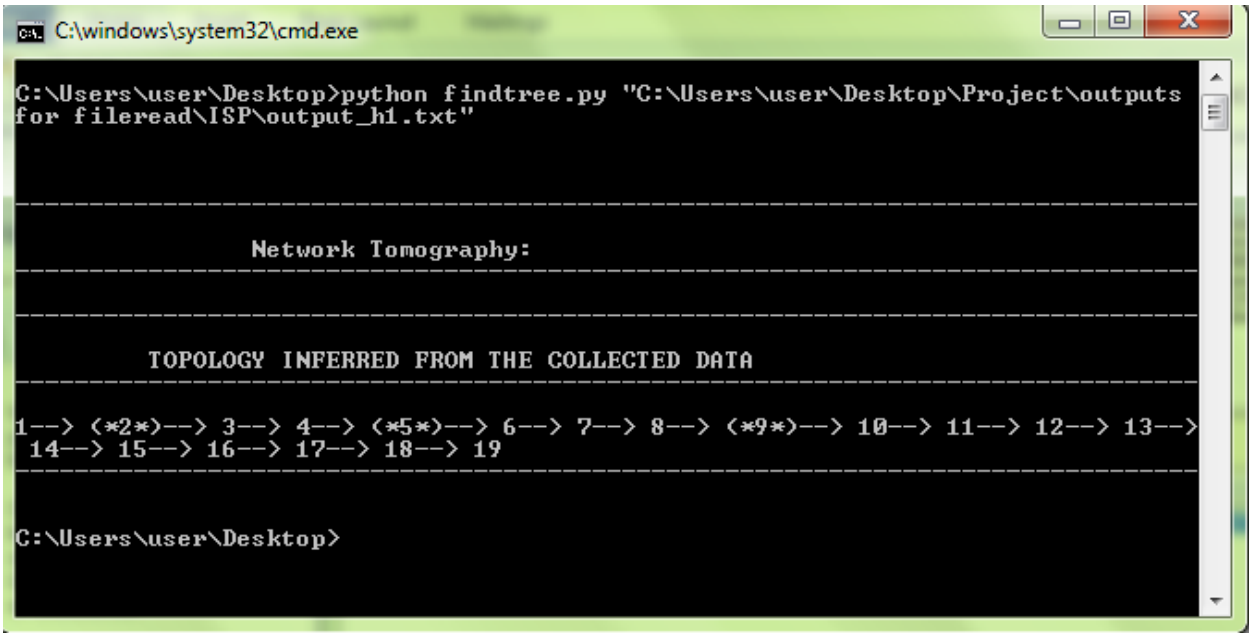
Summarized data as a result of traceroute Sandwich probe:

Total No of probes sent per destination for each tests: 30 per hop

P1, Q and P2 packets are sent to destination 1(D1) to collect set 1

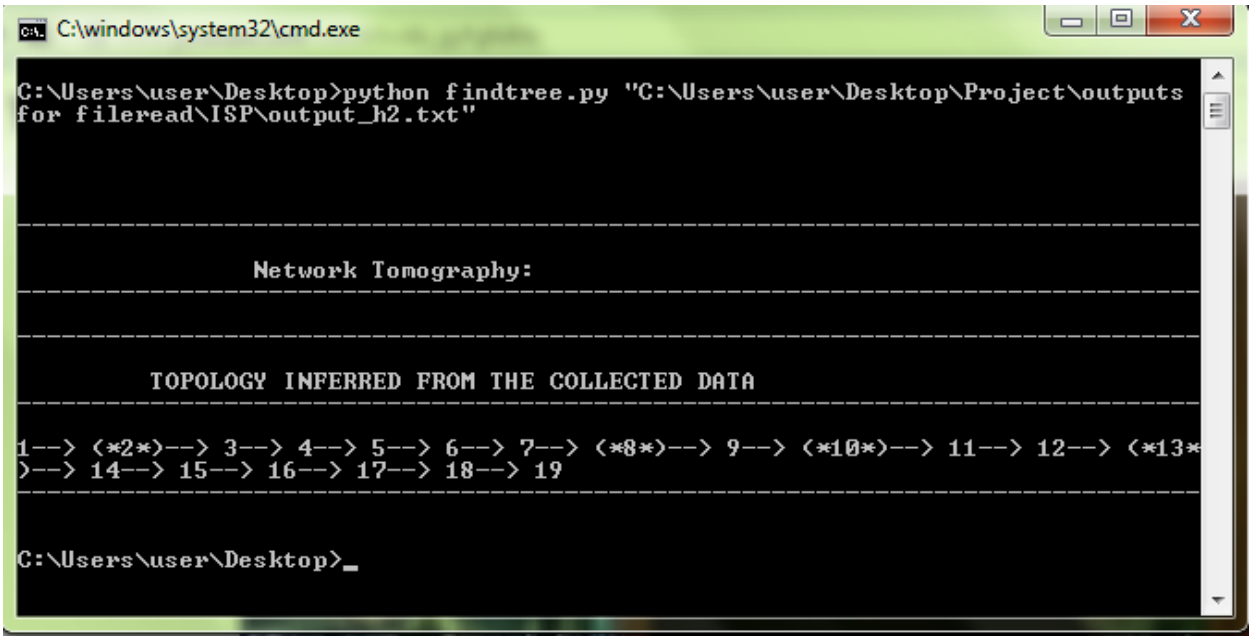
P1 is sent to Destination 1(D1), Q to Destination 3(D3) and P2 to destination 1(D1) to collect set 2

CASE 1: TEST1	SET 1	SET 2			
Hop	Q Sent to Destination 1	Q sent to Destination 3	Standard Deviation of set 1	Standard Deviation of Set 2	Difference in the delay between set1 and set 2 in a particular hop
1	0.1549000104	0.1292068876	0.0685056475	0.0539011753	-0.0256931228
2	0.1964666605	0.1902666728	0.0532633912	0.0312185252	-0.0061999877
3	0.204666694	0.1617000182	0.057130235	0.0364482545	-0.0429666758
4	0.1544999838	0.1613793291	0.0382184607	0.0483745515	0.0068793453
5	0.1385666688	0.1606896499	0.0408196949	0.0485282408	0.0221229811
6	0.1610999982	0.1483792848	0.051721623	0.0486102195	-0.0127207134
7	0.1584333499	0.1475333611	0.0477237426	0.0356143465	-0.0108999888
8	0.1496333281	0.1210999727	0.0435360692	0.0381910525	-0.0285333554
9	0.1413333416	0.1357931022	0.0483959468	0.0379191892	-0.0055402394
10	0.1447000424	0.1350689592	0.0509893962	0.0347954762	-0.0096310832
11	0.1332333167	0.1279666742	0.0420788214	0.0425327268	-0.0052666426
12	0.1268666665	0.1328333298	0.0488069077	0.0334664756	0.0059666634
13	0.1756000121	0.1315999826	0.0649654131	0.0432701317	-0.0440000296
14	0.1480333328	0.1333333254	0.0603816365	0.0348820268	-0.0147000074
15	0.1589333296	0.1461034561	0.0548777089	0.0329632104	-0.0128298735
16	0.1389999866	0.137571437	0.0457112392	0.0425673867	-0.0014285496
17	0.1539666573	0.1433999936	0.0488982993	0.0268248761	-0.0105666637
18	0.1257333358	0.1329230529	0.0388252044	0.0588204585	0.007189717
19	0.1241666635	0.1392666419	0.0326337089	0.0414310283	0.0150999784



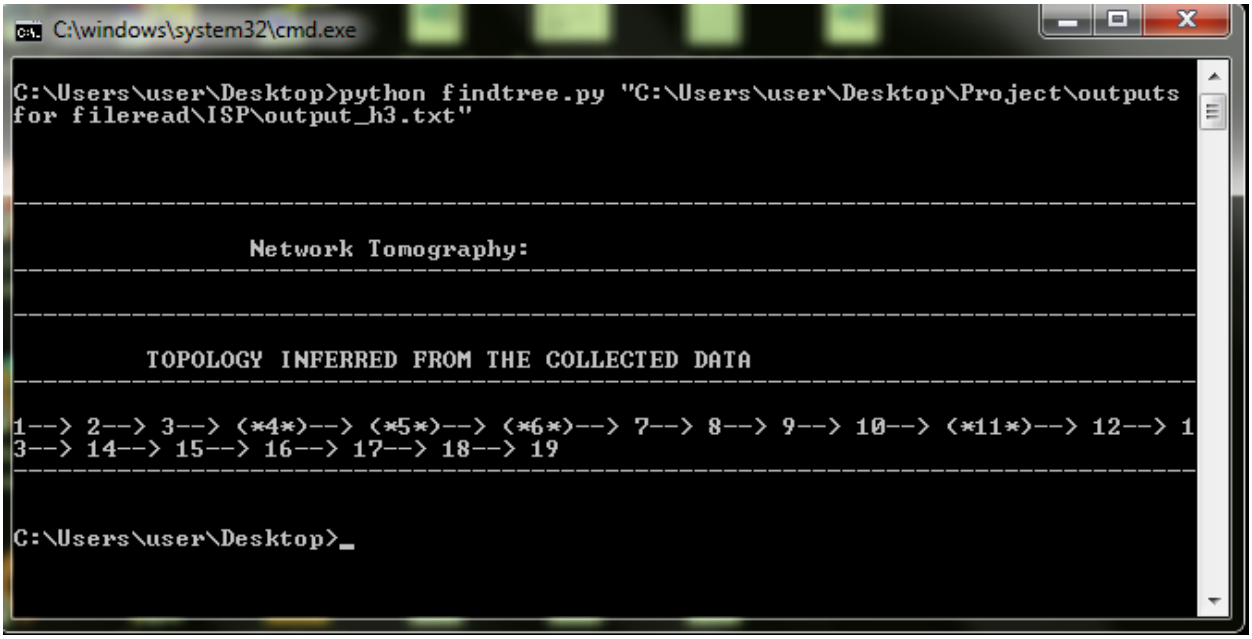
CASE1: TEST 2	SET 1	SET 2			
Hop	Q Sent to Destination 1	Q sent to Destination 3	Standard Deviation of set 1	Standard Deviation of Set 2	Difference in the delay between set1 and set 2 in a particular hop
1	0.1326000214	0.1126538423	0.0848164538	0.0705233247	-0.019946179
2	0.1731000026	0.167678552	0.0640423871	0.0520376919	-0.0054214506
3	0.1762333155	0.164928564	0.0496989172	0.0555621095	-0.0113047515
4	0.143533357	0.1325000127	0.0548499124	0.0443491649	-0.0110333443
5	0.1409000079	0.1586206946	0.0655593194	0.0544531924	0.0177206867
6	0.1076000134	0.1485714316	0.0448193349	0.061077381	0.0409714182
7	0.1754000107	0.1498999993	0.061486198	0.054894062	-0.0255000114
8	0.1368333101	0.130392858	0.0446184861	0.0565380646	-0.0064404522
9	0.1455333153	0.1109999788	0.0560361837	0.0361998104	-0.0345333365
10	0.13379999	0.1325666587	0.0392229703	0.0421921474	-0.0012333314
11	0.1632000128	0.1225714428	0.0543263673	0.0462682258	-0.04062857
12	0.1486000141	0.1403448088	0.0373062894	0.0550343257	-0.0082552052
13	0.1348000288	0.1399310211	0.0522981187	0.0503981627	0.0051309923
14	0.1482666651	0.113499982	0.0518047149	0.0452883392	-0.0347666831
15	0.2150999943	0.1261034423	0.075247362	0.0411154556	-0.0889965521
16	0.1568000317	0.1307930946	0.0438927679	0.0386459453	-0.026006937
17	0.1755333344	0.1402666728	0.0526461218	0.0226361386	-0.0352666616
18	0.1477333307	0.1387241462	0.0516446761	0.0482014318	-0.0090091845

19	0.1417333523	0.1349666834	0.0383816018	0.0330358042	-0.006766669



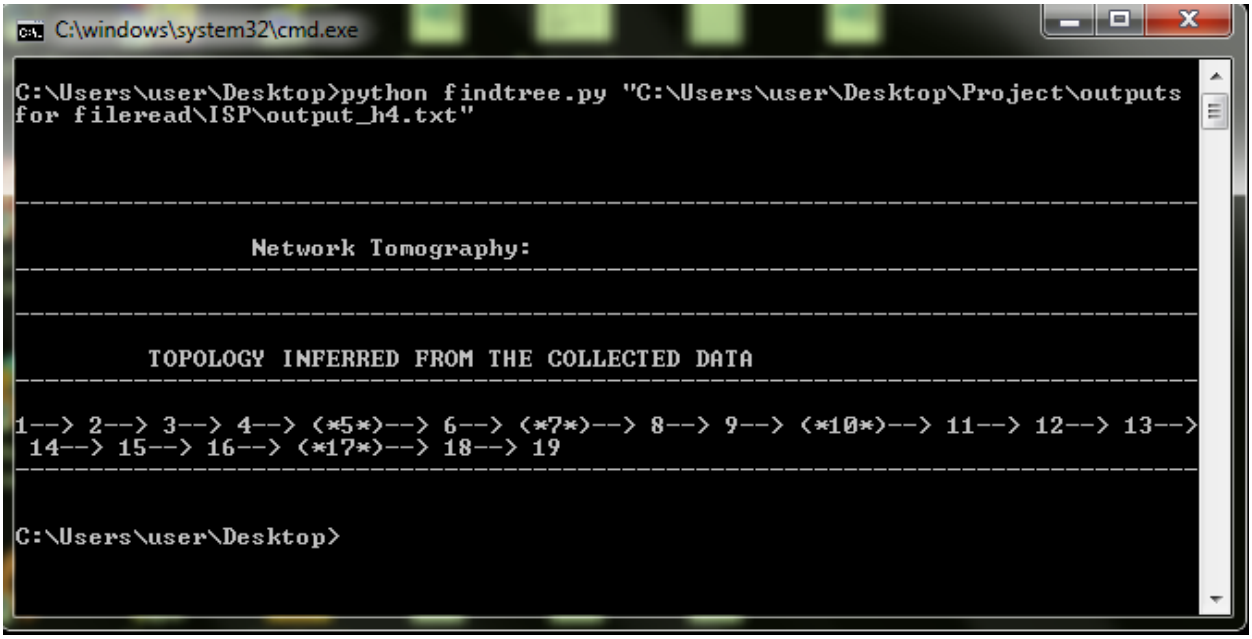
CASE1: TEST 3	SET 1	SET 2			
Hop	Q Sent to Destination 1	Q sent to Destination 3	Standard Deviation of set 1	Standard Deviation of Set 2	Difference in the delay between set1 and set 2 in a particular hop
1	0.1520333529	0.1172068859	0.0634891222	0.0639248057	-0.034826467
2	0.1836333513	0.1791034731	0.0546213694	0.0588035488	-0.0045298782
3	0.1827666601	0.1600384529	0.0464872174	0.0698392861	-0.0227282072
4	0.154766647	0.1672758563	0.0390961927	0.0497605278	0.0125092093
5	0.1494000117	0.1462068969	0.0408187877	0.0603947043	-0.0031931148
6	0.1494000435	0.1382500018	0.0507212068	0.0771489819	-0.0111500416
7	0.1499333223	0.1374138142	0.0321348451	0.063126751	-0.0125195081
8	0.1473332961	0.1245357054	0.0482542143	0.0644140463	-0.0227975908
9	0.1419999917	0.1181666692	0.041819393	0.05116581	-0.0238333225
10	0.1486332973	0.1232758637	0.04263948	0.0424910755	-0.0253574336
11	0.1351999839	0.1249999918	0.0367525475	0.0582402169	-0.0101999921
12	0.1551999966	0.1167585685	0.0345428862	0.0433191614	-0.0384414281
13	0.1634333293	0.1161111019	0.0449917544	0.0663849543	-0.0473222273
14	0.1633666595	0.1209666888	0.045906914	0.0534019337	-0.0423999707
15	0.1731666803	0.1221034609	0.0470846868	0.0468835891	-0.0510632194
16	0.1644333522	0.139199988	0.0421887864	0.0387611656	-0.0252333641

17	0.138199989	0.1389285752	0.0355344037	0.0595419132	0.0007285862
18	0.1308333556	0.1424138382	0.0370306842	0.0335262729	0.0115804826
19	0.1329666376	0.1221333106	0.0464373705	0.041681089	-0.010833327



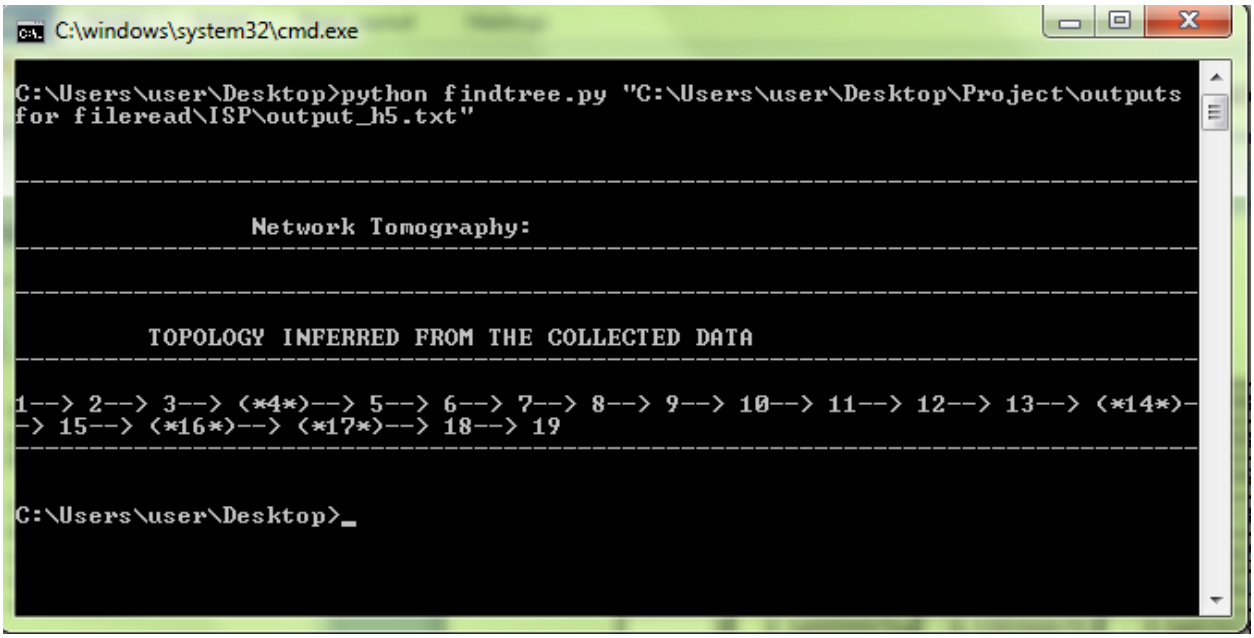
CASE1: TEST 4	SET 1	SET 2			
Hop	Q Sent to Destination 1	Q sent to Destination 3	Standard Deviation of set 1	Standard Deviation of Set 2	Difference in the delay between set1 and set 2 in a particular hop
1	0.1635000149	0.1108076756	0.0875888675	0.1044109277	-0.0526923394
2	0.1868666728	0.1504999825	0.0568094539	0.0627071676	-0.0363666903
3	0.1790999969	0.1577999989	0.0601122916	0.0560252207	-0.021299998
4	0.1596333186	0.1340666612	0.0667360554	0.0710632972	-0.0255666574
5	0.1447333336	0.1466551731	0.0592271835	0.0657890749	0.0019218395
6	0.1622666518	0.1234999816	0.0730631569	0.065458753	-0.0387666702
7	0.1633000374	0.13666666	0.0543891155	0.0629986583	-0.0266333774
8	0.1746000131	0.1278666655	0.0724500854	0.0591505544	-0.0467333476
9	0.1520666599	0.1384333372	0.0491077648	0.0528044671	-0.0136333227
10	0.1363333146	0.1339286055	0.0493642344	0.0679510798	-0.0024047091
11	0.1540999889	0.1284642475	0.0668310011	0.054294201	-0.0256357414
12	0.1251333396	0.1283793038	0.0426875386	0.0443690773	0.0032459643
13	0.1250333468	0.1157586328	0.0330378389	0.0416261144	-0.009274714
14	0.1456000169	0.128700002	0.0450383744	0.04514286	-0.0169000149
15	0.1300666491	0.1204666694	0.0429177028	0.0402299561	-0.0095999797
16	0.1777999798	0.1490689804	0.0661661918	0.0635069615	-0.0287309995

17	0.1268333197	0.1376333396	0.0458898002	0.0323722934	0.0108000199
18	0.1444999933	0.1348965579	0.035823899	0.0439578332	-0.0096034354
19	0.1442333539	0.129076921	0.0372924956	0.0633636311	-0.0151564329



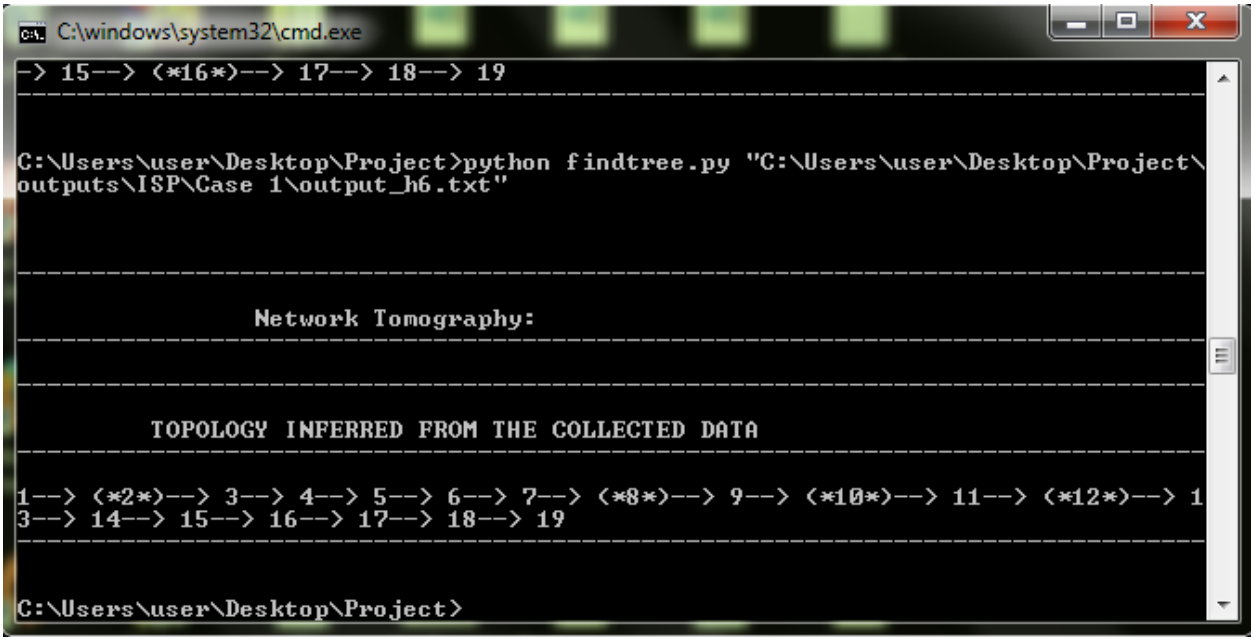
CASE1: TEST 5	SET 1	SET 2			
Hop	Q Sent to Destination 1	Q sent to Destination 3	Standard Deviation of set 1	Standard Deviation of Set 2	Difference in the delay between set1 and set 2 in a particular hop
1	0.1571999947	0.1099285824	0.0819216869	0.0724926891	-0.0472714123
2	0.2538000027	0.1762222714	0.068398504	0.0666369172	-0.0775777313
3	0.1854667028	0.1721724148	0.0566188114	0.0535551926	-0.0132942879
4	0.1688333273	0.15872416	0.0528496358	0.0567419568	-0.0101091673
5	0.1785333474	0.139964274	0.060513257	0.0569210412	-0.0385690735
6	0.1522666613	0.1422068826	0.0638342654	0.0621867215	-0.0100597787
7	0.1494666974	0.1388275952	0.0513964145	0.0474882277	-0.0106391022
8	0.1488999764	0.1282666763	0.0482923039	0.0440340422	-0.0206333001
9	0.1354333401	0.1216551929	0.046465628	0.0448125064	-0.0137781472
10	0.146566693	0.132758601	0.0585927204	0.0602829544	-0.013808092
11	0.129366676	0.1241723916	0.0402396925	0.0423053227	-0.0051942845
12	0.1152333339	0.1329333464	0.035262535	0.0493753743	0.0177000125
13	0.1142333349	0.1363793077	0.0397799784	0.0463462537	0.0221459729
14	0.1229000251	0.135466663	0.03905753	0.0533864961	0.012566638
15	0.131033357	0.1268275935	0.0306816994	0.0444214289	-0.0042057635

16	0.1396666765	0.1401666562	0.0268779241	0.0497024833	0.0004999797
17	0.1422332923	0.1370689705	0.0241753208	0.0510135805	-0.0051643218
18	0.1480333408	0.1313103396	0.0261068255	0.0371496917	-0.0167230011
19	0.1540999969	0.1351379115	0.0338904561	0.0342404938	-0.0189620854



CASE1: TEST 6	SET 1	SET 2			
Hop	Q Sent to Destination 1	Q sent to Destination 3	Standard Deviation of set 1	Standard Deviation of Set 2	Difference in the delay between set1 and set 2 in a particular hop
1	0.1454000076	0.126407429	0.0747141548	0.071417993	-0.0189925785
2	0.204099973	0.1883103108	0.0543758374	0.0258772026	-0.0157896623
3	0.1840666771	0.1681333383	0.0401426438	0.0311648478	-0.0159333388
4	0.1673666557	0.1683999936	0.0513861376	0.0513508046	0.0010333379
5	0.1450666746	0.1482666651	0.0425017198	0.0533884829	0.0031999906
6	0.1563333352	0.1517333666	0.0515456348	0.0452820075	-0.0045999686
7	0.1786333323	0.1535666625	0.0469017446	0.0329976604	-0.0250666698
8	0.1398000081	0.1365666866	0.044681704	0.0385786625	-0.0032333215
9	0.1445666472	0.1315714121	0.0336620041	0.0405980023	-0.0129952351

10	0.1313666582	0.1359666665	0.042055644	0.038290544	0.0046000083
11	0.139233319	0.1282499773	0.0406658679	0.0431778003	-0.0109833416
12	0.1366666635	0.1374333143	0.0586346598	0.0282544407	0.0007666508
13	0.1665000041	0.1322666725	0.0652990487	0.0394849468	-0.0342333317
14	0.1499000311	0.1418999831	0.0406561115	0.0417271667	-0.008000048
15	0.1189999978	0.1357333183	0.0384820219	0.0359489768	0.0167333206
16	0.1296333075	0.1403333505	0.0415174483	0.0336613944	0.010700043
17	0.1517333269	0.1406666835	0.044488842	0.0272131625	-0.0110666434
18	0.1444666783	0.1457586206	0.0426807735	0.0289507677	0.0012919423
19	0.1650333246	0.1460333427	0.0416757158	0.0249312171	-0.0189999819



Observation of Case 1 :

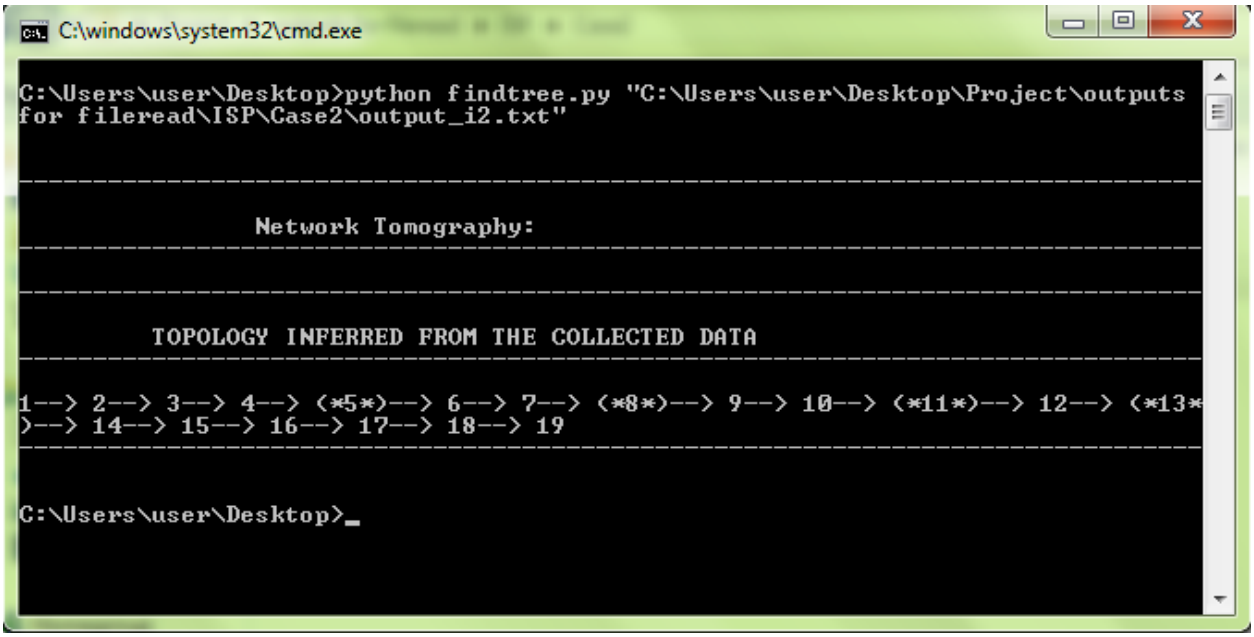
From the tests, we could infer that though the path taken changes, sometimes, it goes through same common parents, For example, Node2, Node 5, Node 8 and Node 10 are repeated as parents in multiple tests.

So we can get a picture like there might path divergence in those nodes but since the result is too dynamic, output we get depends on routing decisions at particular instance of tests.

5.2.2 Case 2 :

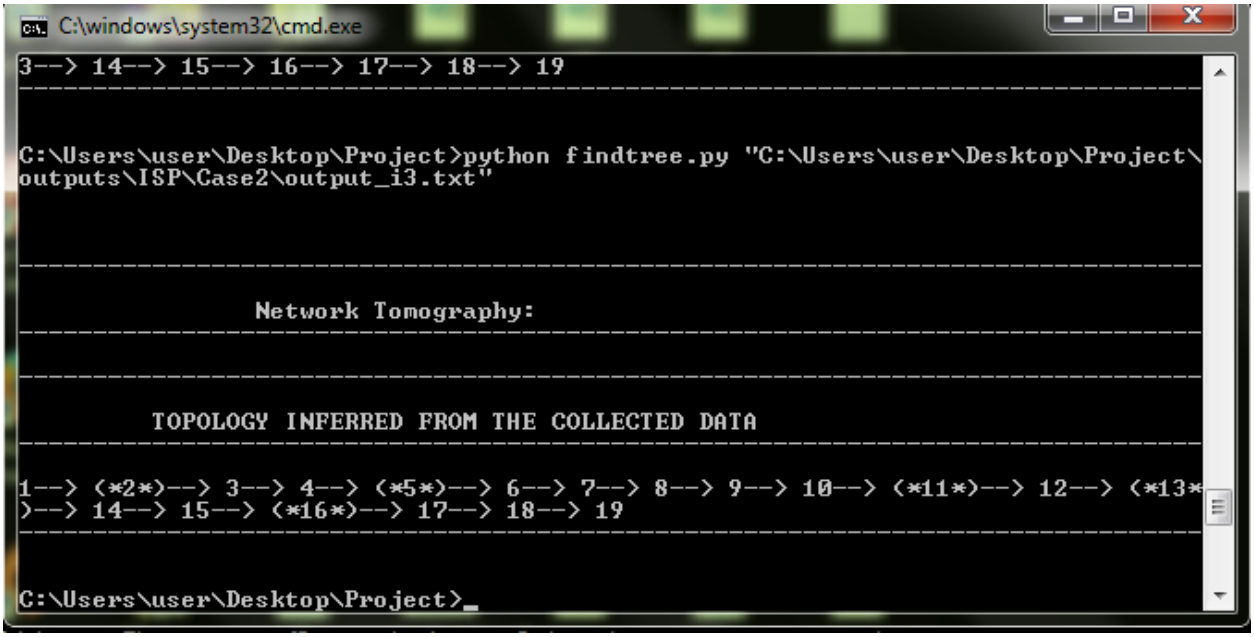
Packet sent from India(Source) to Destination 1(Garneau) and Destination 2(Mckernan)
IP address of Garneau : 96.52.102.25
IP Address of Mckernan: 96.52.127.104

CASE2: TEST 1	SET 1	SET 2			
Hop	Delay when Q Sent to Destination 1	Delay when Q sent to Destination 2	Standard Deviation of set 1	Standard Deviation of Set 2	Difference in the delay between set1 and set 2 in a particular hop
1	0.1828000069	0.1297931096	0.0928650024	0.0960635969	-0.0530068973
2	0.1846000036	0.1543846314	0.0670629353	0.0817377765	-0.0302153722
3	0.1840666533	0.1634642652	0.0490021518	0.0554858185	-0.020602388
4	0.1698666573	0.1605714474	0.0767168233	0.0749232555	-0.0092952098
5	0.1479000092	0.1546785576	0.056791665	0.0800872776	0.0067785484
6	0.1719000022	0.1428276095	0.0649189455	0.0596354125	-0.0290723927
7	0.1582000017	0.1473571488	0.0560915524	0.0694025398	-0.0108428529
8	0.1403333426	0.1298620783	0.0514164073	0.0591518038	-0.0104712642
9	0.1350333214	0.1079285485	0.0462489153	0.0496293788	-0.0271047728
10	0.1719666719	0.1172333638	0.0600864947	0.0422631972	-0.0547333082
11	0.1221666813	0.1271034438	0.054359348	0.0697857838	0.0049367625
12	0.1649666627	0.1187856793	0.0571569391	0.0573771022	-0.0461809834
13	0.1291999737	0.1172069023	0.048446538	0.0522970567	-0.0119930714
14	0.131099995	0.1091200161	0.0408398085	0.0862733638	-0.0219799789
15	0.1552999814	0.1140999635	0.0448856248	0.0443962323	-0.0412000179
16	0.1499000072	0.1136551808	0.0488556039	0.0567669795	-0.0362448265
17	0.1540666978	0.1159000079	0.0515534145	0.0450283848	-0.0381666899
18	0.1784666618	0.1327931059	0.0766513472	0.0523834107	-0.0456735559
19	0.1361333211	0.1284666538	0.0408854326	0.0383890032	-0.0076666673



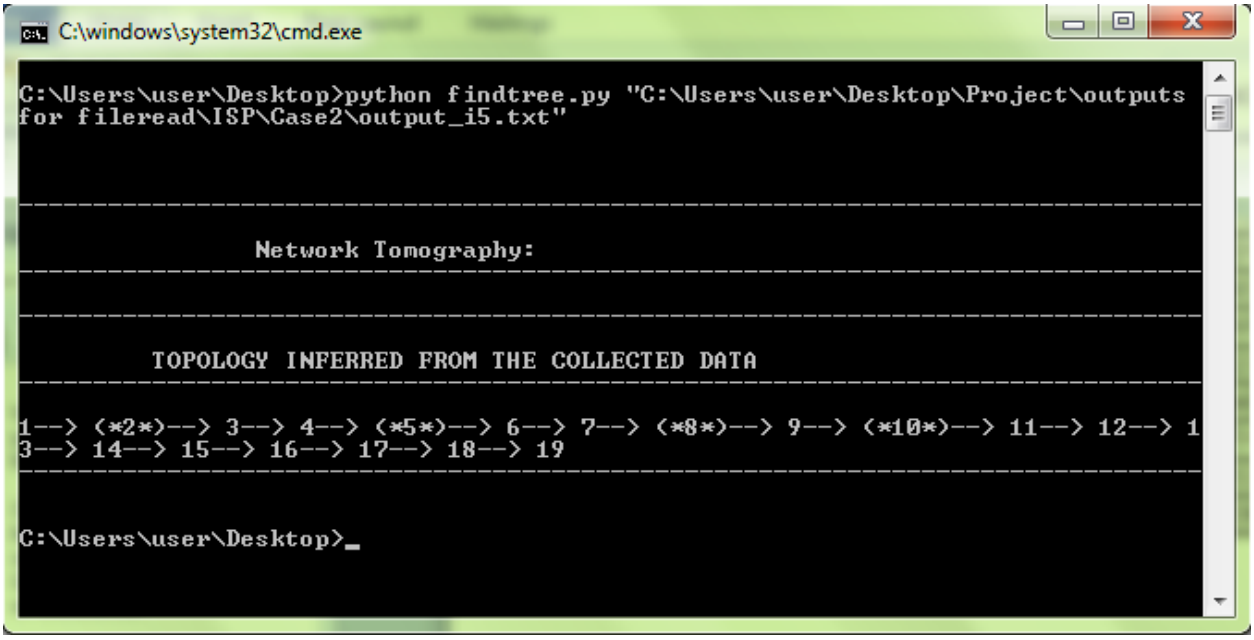
CASE2: TEST 2					
Hop	Delay when Q Sent to Destination 1	Delay when Q sent to Destination 2	Standard Deviation of set 1	Standard Deviation of Set 2	Difference in the delay between set1 and set 2 in a particular hop
1	0.1448999723	0.1150769087	0.0757605706	0.1081266727	-0.0298230636
2	0.2197666883	0.1771111047	0.045590461	0.0783201249	-0.0426555837
3	0.211866649	0.1591379478	0.0549081702	0.0598390976	-0.0527287012
4	0.1635666688	0.1366551745	0.0493311069	0.0516878692	-0.0269114944
5	0.1550333261	0.1490000281	0.0545187921	0.0647575277	-0.0060332981
6	0.1591666619	0.1310689778	0.0488441392	0.0651244321	-0.0280976841
7	0.1650333643	0.1382758535	0.0359377809	0.0536494588	-0.0267575108
8	0.1147333066	0.1128571544	0.0372736993	0.0570025154	-0.0018761521
9	0.1376999855	0.1194642867	0.0334875241	0.0548064403	-0.0182356988
10	0.1432999929	0.1191071527	0.0462658766	0.0550238233	-0.0241928401
11	0.1214667002	0.1231851843	0.0399764692	0.0783262063	0.001718484
12	0.1384000222	0.1032500352	0.0384579348	0.0574598917	-0.035149987
13	0.1365000089	0.1264482695	0.0353843137	0.05120663	-0.0100517394
14	0.1481999954	0.11599999	0.0415607444	0.0630931999	-0.0322000053
15	0.152699995	0.1238276054	0.0328120023	0.0487666866	-0.0288723896

16	0.1447666804	0.1390345014	0.0332837731	0.0707079566	-0.005732179
17	0.1482999643	0.1225666682	0.0391694012	0.0440149026	-0.0257332961
18	0.154433314	0.123896533	0.0433104625	0.0521283937	-0.030536781
19	0.1287000259	0.1223214609	0.03680334	0.0535022197	-0.0063785649



CASE2: TEST 3	SET 1	SET 2			
Hop	Delay when Q Sent to Destination 1	Delay when Q sent to Destination 2	Standard Deviation of set 1	Standard Deviation of Set 2	Difference in the delay between set1 and set 2 in a particular hop
1	0.1596666495	0.1219999981	0.0728690995	0.1044411712	-0.0376666514
2	0.1833999952	0.1739655363	0.0257935327	0.0568166649	-0.0094344589
3	0.1906333447	0.1564444083	0.0540417471	0.0673694737	-0.0341889364
4	0.1607666651	0.1372333527	0.044513692	0.0566755202	-0.0235333125
5	0.1446999947	0.1464999755	0.0545931555	0.056545976	0.0017999808
6	0.1546332995	0.1191333135	0.0448865122	0.0492332578	-0.035499986
7	0.1433999936	0.1458275811	0.0452169756	0.045448804	0.0024275876
8	0.132066679	0.1303571548	0.0424523402	0.0428390264	-0.0017095242
9	0.1381666581	0.1242413685	0.0465546816	0.0612981739	-0.0139252896
10	0.1479666948	0.1280666749	0.0399751803	0.0519589124	-0.01990002
11	0.154399991	0.1139999505	0.0341253592	0.0427171306	-0.0404000406
12	0.1614000161	0.1247857128	0.0476839697	0.0488124876	-0.0366143034

13	0.1289999803	0.1299655109	0.0343229779	0.0436203331	0.0009655306
14	0.1799666723	0.1409642696	0.0489660097	0.0654080811	-0.0390024026
15	0.1429000219	0.1238666932	0.0400698678	0.0450827159	-0.0190333287
16	0.1340333462	0.1356071319	0.0307468341	0.0543251253	0.0015737857
17	0.1407333295	0.1361666679	0.0232306395	0.0417085741	-0.0045666615
18	0.1345333099	0.1289999896	0.0371526234	0.044978209	-0.0055333203
19	0.1635333379	0.1283703822	0.0421407096	0.0691932354	-0.0351629557



Observation of Case 2 :

Note: Where it went wrong!

Node 2 is a common parent where there is a divergence, which is found in test 2 and 3.

We can find that Node 5, 8 and 11 are often inferred as parent nodes whereas node 2 and node 10 pops up as parent in one test.

But when I did trace route I cannot substantiate that node 5 is a common parent and there is a path diverging at node 6. However repeated tests gave results contrary to traceroute. So very hard to prove that Node 5 is a common parent, though there was a significant drop in delay after node5 in multiple tests.

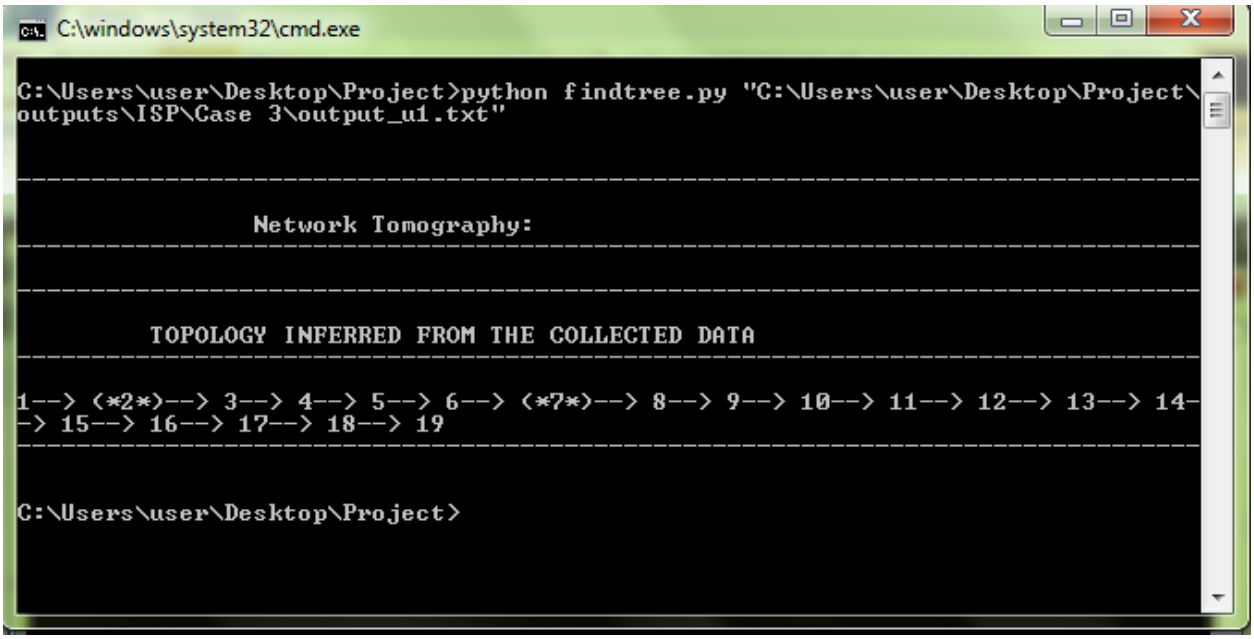
5.2.3 Case 3:

Total No of probes sent per destination for each tests: 30 per hop

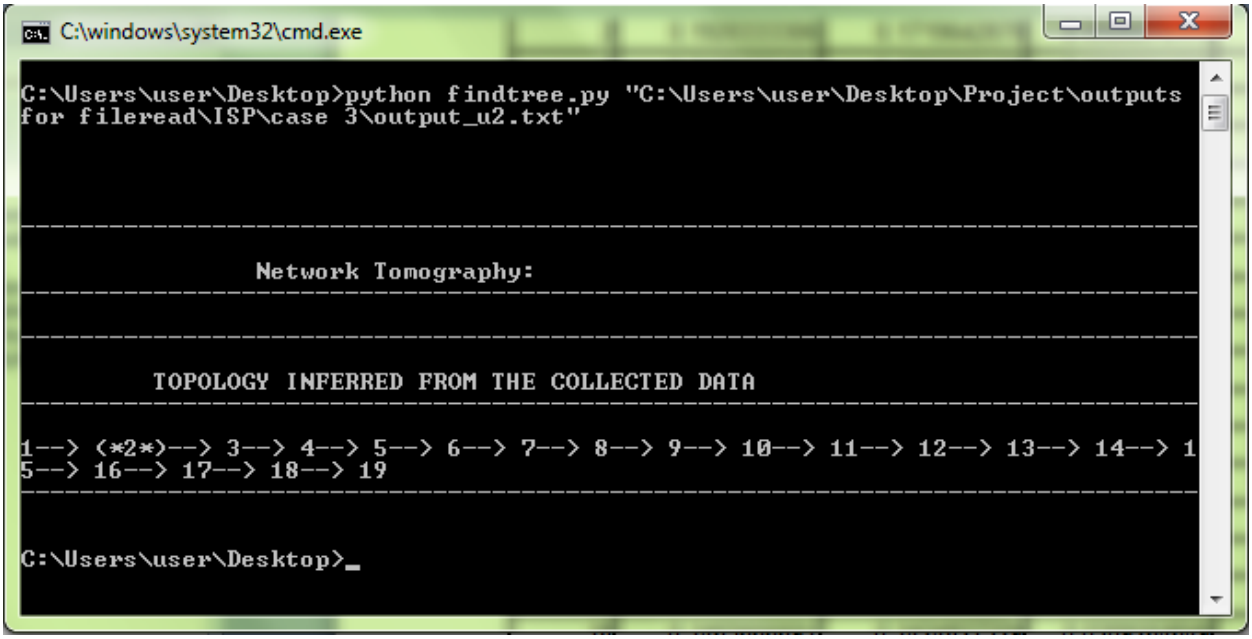
P1, Q and P2 packets are sent to destination 1(D4) to collect set 1

P1 is sent to Destination 1(D4)(University), Q to Destination 3 (D3)(Millwoods) and P2 to destination 1(D4) to collect set 2

CASE3: TEST 1	SET 1	SET 2			
Hop	Delay when Q Sent to Destination 4	Delay when Q sent to Destination 3	Standard Deviation of set 1	Standard Deviation of Set 2	Difference in the delay between set1 and set 2 in a particular hop
1	0.1381000121	0.1475517092	0.078916591	0.0846306538	0.0094516971
2	0.1874333143	0.1788518694	0.0588440574	0.0723032867	-0.008581445
3	0.1849333127	0.1561379104	0.0645709516	0.0412847118	-0.0287954024
4	0.1642000278	0.1512333155	0.0672528229	0.0449063425	-0.0129667123
5	0.1596333186	0.156785718	0.0734039993	0.0506099044	-0.0028476006
6	0.1586333434	0.1382000287	0.0656578677	0.0564390167	-0.0204333146
7	0.1479666869	0.1456785713	0.0505656317	0.0428852474	-0.0022881156
8	0.1360666672	0.1225517043	0.0446701142	0.0492334066	-0.013514963
9	0.1183666706	0.1318999926	0.0395461782	0.0353396287	0.013533322
10	0.1432000001	0.1476428424	0.0463107869	0.0652514888	0.0044428422
11	0.1264000018	0.1425714152	0.0403415437	0.0751785761	0.0161714134
12	0.1478333314	0.1443103264	0.0610489423	0.0549985293	-0.003523005
13	0.1289666335	0.1265999953	0.0547664791	0.0375850807	-0.0023666382
14	0.1546000083	0.1237999996	0.0575709678	0.0481604448	-0.0308000088
15	0.1716999769	0.1224666437	0.0666934471	0.0407166994	-0.0492333333
16	0.1188333035	0.1294000149	0.0466962611	0.0412476854	0.0105667114
17	0.1827333212	0.1296428697	0.0691321719	0.0546663305	-0.0530904514
18	0.1995000045	0.1352000157	0.0803577036	0.0377354035	-0.0642999887
19	0.1297666788	0.1351666927	0.0530672609	0.0332295861	0.0054000139

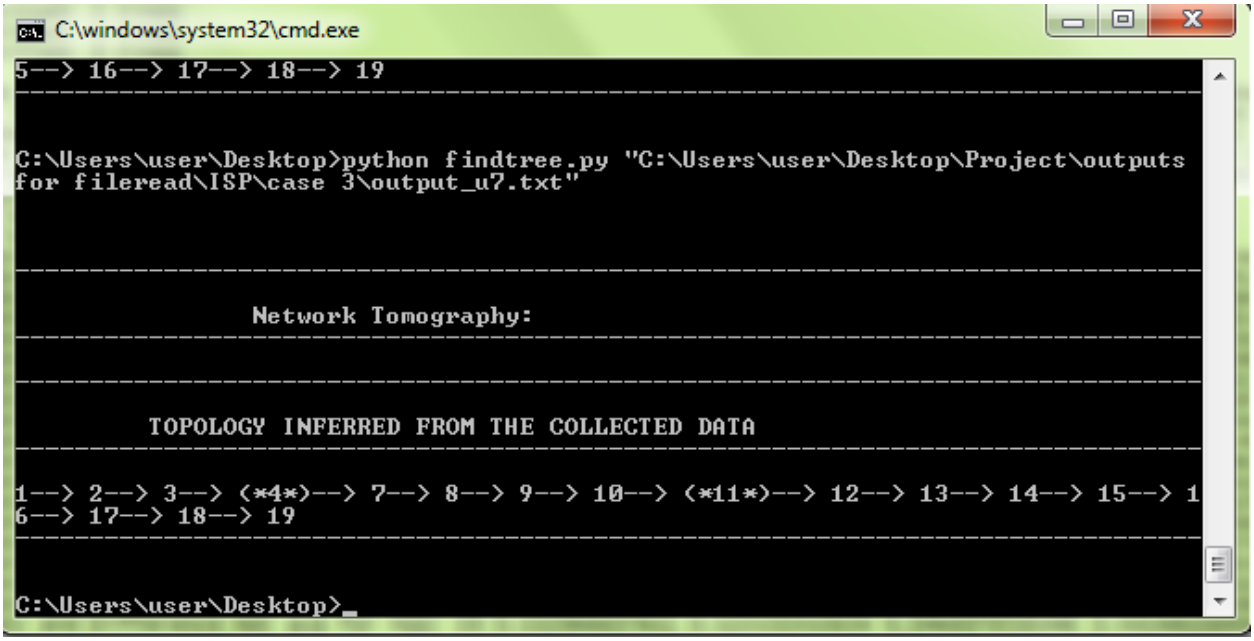


CASE3: TEST 2	SET 1	SET 2			
Hop	Delay when Q Sent to Destination 4	Delay when Q sent to Destination 3	Standard Deviation of set 1	Standard Deviation of Set 2	Difference in the delay between set1 and set 2 in a particular hop
1	0.1564666589	0.1354074037	0.080630765	0.0892308939	-0.0210592552
2	0.1920333306	0.1719642878	0.0506074175	0.058917327	-0.0200690428
3	0.2175666491	0.1652000109	0.0689504069	0.0283377557	-0.0523666382
4	0.1620333433	0.1525861888	0.0582451397	0.0490194764	-0.0094471545
5	0.177700003	0.1556333621	0.0564070696	0.0524579069	-0.0220666409
6	0.1339333455	0.1286896508	0.0554959152	0.0611212768	-0.0052436946
7	0.1591332833	0.1359310315	0.0422458051	0.0471416045	-0.0232022518
8	0.1418333054	0.1142332951	0.0416634638	0.0415625451	-0.0276000102
9	0.1372000297	0.1239655018	0.0361080706	0.0417252051	-0.0132345279
10	0.1254000028	0.1272666772	0.0390560088	0.0417100774	0.0018666744
11	0.1542999983	0.1332999706	0.0490584523	0.0460565877	-0.0210000277
12	0.1295666615	0.1249999841	0.0480573728	0.0510091845	-0.0045666774
13	0.1438333432	0.133066686	0.0419326837	0.0430851123	-0.0107666572
14	0.1327666759	0.1245172106	0.0327217325	0.0460171126	-0.0082494654
15	0.1504333576	0.1438333193	0.0325046867	0.048028514	-0.0066000382
16	0.1236333211	0.1340357321	0.0348228573	0.0456008672	0.0104024109
17	0.1526000261	0.1418620554	0.0379285088	0.0395988484	-0.0107379708
18	0.1465666691	0.1424827329	0.0309264649	0.0333792302	-0.0040839362
19	0.1409333706	0.1418965521	0.0364726717	0.0347285864	0.0009631815



CASE3: TEST 3	SET 1	SET 2			
Hop	Delay when Q Sent to Destination 4	Delay when Q sent to Destination 3	Standard Deviation of set 1	Standard Deviation of Set 2	Difference in the delay between set1 and set 2 in a particular hop
1	0.1520333529	0.1172068859	0.0634891222	0.0639248057	-0.034826467
2	0.1836333513	0.1791034731	0.0546213694	0.0588035488	-0.0045298782
3	0.1827666601	0.1600384529	0.0464872174	0.0698392861	-0.0227282072
4	0.154766647	0.1672758563	0.0390961927	0.0497605278	0.0125092093
5	0.1494000117	0.1462068969	0.0408187877	0.0603947043	-0.0031931148
6	0.1494000435	0.1382500018	0.0507212068	0.0771489819	-0.0111500416
7	0.1499333223	0.1374138142	0.0321348451	0.063126751	-0.0125195081
8	0.1473332961	0.1245357054	0.0482542143	0.0644140463	-0.0227975908
9	0.1419999917	0.1181666692	0.041819393	0.05116581	-0.0238333225
10	0.1486332973	0.1232758637	0.04263948	0.0424910755	-0.0253574336
11	0.1351999839	0.1249999918	0.0367525475	0.0582402169	-0.0101999921
12	0.1551999966	0.1167585685	0.0345428862	0.0433191614	-0.0384414281
13	0.1634333293	0.1161111019	0.0449917544	0.0663849543	-0.0473222273
14	0.1633666595	0.1209666888	0.045906914	0.0534019337	-0.0423999707
15	0.1731666803	0.1221034609	0.0470846868	0.0468835891	-0.0510632194
16	0.1644333522	0.139199988	0.0421887864	0.0387611656	-0.0252333641

17	0.138199989	0.1389285752	0.0355344037	0.0595419132	0.0007285862
18	0.1308333556	0.1424138382	0.0370306842	0.0335262729	0.0115804826
19	0.1329666376	0.1221333106	0.0464373705	0.041681089	-0.010833327



Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\pad>tracert 142.244.159.71

Tracing route to uws69644usr.uws.ualberta.ca [142.244.159.71]
over a maximum of 30 hops:

1 3 ms 4 ms 4 ms D-Link.Home [192.168.1.1]
2 21 ms 18 ms 28 ms 59.92.64.1
3 24 ms 19 ms 19 ms 218.248.235.129
4 18 ms 19 ms 31 ms 218.248.235.214
5 24 ms 19 ms 19 ms 14.141.139.145.static-chennai.vsnl.net.in [14.141.139.145]
6 29 ms 24 ms 19 ms ix-4-2.tcore1.cxr-chennai.as6453.net [180.87.36.9]
7 271 ms 264 ms 265 ms if-3-3.tcore2.cxr-chennai.as6453.net [180.87.36.6]
8 269 ms 269 ms 269 ms if-9-2.tcore2.mlv-mumbai.as6453.net [180.87.37.10]
9 274 ms 273 ms 273 ms if-6-2.tcore1.l78-london.as6453.net [80.231.130.5]
10 274 ms 269 ms 269 ms if-2-2.tcore2.l78-london.as6453.net [80.231.131.1]
11 272 ms 278 ms 269 ms if-20-2.tcore2.nyy-new-york.as6453.net [216.6.99.13]
12 270 ms 268 ms 269 ms if-12-6.tcore1.ct8-chicago.as6453.net [216.6.99.46]
13 269 ms 269 ms 267 ms if-22-2.tcore2.ct8-chicago.as6453.net [64.86.79.1]
14 270 ms 269 ms 266 ms 64.86.79.78
15 * * * Request timed out.

16 328 ms 325 ms 322 ms h-207-148-129-186.biz.sta.cadvision.com [207.148.129.186]
17 333 ms 330 ms 333 ms core1-gsb-asr.backbone.ualberta.ca [129.128.0.20]
18 330 ms 328 ms 357 ms 129.128.0.11
19   *   *   *   Request timed out.
20 ^C

C:\Users\pad>tracert 68.150.144.25

Tracing route to s0106602ad0719ce5.ed.shawcable.net [68.150.144.25]
over a maximum of 30 hops:

1   9 ms   13 ms   10 ms D-Link.Home [192.168.1.1]
2 24 ms 19 ms 21 ms 59.92.64.1
3 21 ms 18 ms 45 ms 218.248.235.141
4 20 ms 19 ms 28 ms 218.248.235.142
5 19 ms 40 ms 23 ms 14.141.139.145.static-chennai.vsnl.net.in [14.141.139.145]
6 59 ms 93 ms 54 ms 172.25.81.134
7 49 ms 47 ms 45 ms ix-0-100.tcore1.mlv-mumbai.as6453.net [180.87.38.5]
8 153 ms 153 ms 153 ms if-9-5.tcore1.wyn-marseille.as6453.net [80.231.217.17]
9 159 ms 160 ms 157 ms if-8-1600.tcore1.pye-paris.as6453.net [80.231.217.6]
10 267 ms 254 ms 254 ms 80.231.154.86
11 259 ms 257 ms 254 ms prs-bb2-link.telia.net [213.155.131.18]
12 257 ms 302 ms 255 ms nyk-bb2-link.telia.net [213.155.135.7]
13 337 ms 334 ms 364 ms chi-bb1-link.telia.net [80.91.248.196]
14   *   337 ms 333 ms shawbusiness-ic-300304-chi-eqx-i1.c.telia.net [62.115.12.86]
15 354 ms 352 ms 350 ms 66.163.75.118
16 373 ms 370 ms 373 ms rc1we-tge0-6-0-4.ed.shawcable.net [66.163.78.69]
17 444 ms 375 ms 377 ms dx2ni-te3.ed.shawcable.net [64.59.184.150]
18   *   *   *   Request timed out.
19   *   ^C

C:\Users\pad>tracert 142.244.159.71

Tracing route to uws69644usr.uws.ualberta.ca [142.244.159.71]
over a maximum of 30 hops:

1   3 ms   2 ms   4 ms D-Link.Home [192.168.1.1]
2 21 ms 20 ms 35 ms 59.92.64.1
3 22 ms 31 ms 34 ms 218.248.235.141
4 20 ms 19 ms 102 ms 218.248.235.214
5 20 ms 18 ms 18 ms 14.141.139.145.static-chennai.vsnl.net.in [14.141.139.145]
6 24 ms 19 ms 19 ms ix-4-2.tcore1.cxr-chennai.as6453.net [180.87.36.9]
7 268 ms 265 ms 268 ms if-3-3.tcore2.cxr-chennai.as6453.net [180.87.36.6]
8 269 ms 269 ms 287 ms if-9-2.tcore2.mlv-mumbai.as6453.net [180.87.37.10]
9 271 ms   *   273 ms if-6-2.tcore1.l78-london.as6453.net [80.231.130.5]
10 270 ms 269 ms 269 ms if-2-2.tcore2.l78-london.as6453.net [80.231.131.1]
11 272 ms 269 ms 269 ms if-20-2.tcore2.nyy-new-york.as6453.net [216.6.99.13]

```
 12   267 ms   264 ms   264 ms  if-12-6.tcore1.ct8-chicago.as6453.net [216.6.99.46]
 13   269 ms   267 ms   268 ms  if-22-2.tcore2.ct8-chicago.as6453.net [64.86.79.1]
 14   274 ms   269 ms   269 ms  64.86.79.78
 15    *        *        *     Request timed out.
 16   320 ms   319 ms   320 ms  h-207-148-129-186.biz.sta.cadvision.com [207.148.129.186]
 17   329 ms   328 ms   327 ms  core1-gsb-asr.backbone.ualberta.ca [129.128.0.20]
 18   328 ms   328 ms   324 ms  129.128.0.11
 19    *     ^C
```

C:\Users\pad>tracert 68.150.144.25

Tracing route to s0106602ad0719ce5.ed.shawcable.net [68.150.144.25]
over a maximum of 30 hops:

```
  1    2 ms     2 ms     4 ms  D-Link.Home [192.168.1.1]
 2   21 ms    30 ms    18 ms  59.92.64.1
 3   29 ms    26 ms    19 ms  218.248.235.209
 4   22 ms    17 ms    34 ms  218.248.235.142
 5   19 ms    24 ms    18 ms  14.141.139.145.static-chennai.vsnl.net.in [14.141.139.145]
  6   56 ms   107 ms    64 ms  172.25.81.134
  7   48 ms     *       48 ms  ix-0-100.tcore1.mlv-mumbai.as6453.net [180.87.38.5]
  8    *      153 ms     *     if-9-5.tcore1.wyn-marseille.as6453.net [80.231.217.17]
  9  160 ms   522 ms   158 ms  if-8-1600.tcore1.pye-paris.as6453.net [80.231.217.6]
 10   258 ms   254 ms   255 ms  80.231.154.86
 11   257 ms   253 ms   254 ms  prs-bb2-link.telia.net [213.155.131.18]
 12   256 ms   254 ms   254 ms  nyk-bb2-link.telia.net [213.155.135.7]
 13   335 ms   335 ms   332 ms  chi-bb1-link.telia.net [80.91.248.196]
 14    *      339 ms   337 ms  shawbusiness-ic-300304-chi-eqx-i1.c.telia.net [62.115.12.86]
 15   440 ms   351 ms   349 ms  66.163.75.118
 16   373 ms   367 ms   377 ms  rc1we-tge0-6-0-0.ed.shawcable.net [66.163.78.133]
 17   431 ms   375 ms   373 ms  dx2ni-te3.ed.shawcable.net [64.59.184.150]
 18    *        *        *     Request timed out.
 19    *        *        *     Request timed out.
 20  ^C
C:\Users\pad>
```

Observations of Case 3:

Note: We can see from the traceroute that much of the divergence happens between 2nd and 4th hops.

But because delay is building up in identical fashion, algorithm feels that nodes down the line might be parents, which is not proven by traceroute.

So the problem with this algorithm is that if the delay is mimicked by diverged links, algorithm assumes that it might be the same link in both sets. Since we don't have any information to differentiate link other than delay, it is very tough to remove errors if there is mimicking.

CHAPTER 6:

Conclusion & Future work

In this project, we discussed so far about Network Tomography which is study or inferring network bandwidth, delay or topology by either observing existing traffic or by actively probing. This report gave a brief explanation about existing data collection methods like Maximum likelihood approach and Constructive methods.

Our new idea of traceroute sandwich probe was discussed in detail. This report also explains on how to collect data using TSP and how to infer topology from the data collected. To test this approach, we setup lab network with multiple routers and destinations diverging at various internal nodes. We used our approach to find whether the topology is observed correctly.

Then we used this same algorithm to check if it can be used with ISP Core network. We found out why the existing algorithm proposed by Amir Malekzadeh has to be slightly modified. We found out that delay similarities rarely occur in ISP network and so the algorithm by Amir which relies on similarities in delay will need to be modified to find topology.

Then we discussed about new suggestions to improve the existing algorithm and discussed the results.

One of the biggest problems that I think could be solved with this new solution is that TSP can be used with a non-monotonic network where delay is wayward.

6.1 Future work:

Since we are using UDP to send probes, we expect packets to be lost. Even though the program at present can sense this loss, it asks sender to resend every probe again. This is done purely because, we don't know for which hop a certain P1 or P2 is not received.

I suggest that we modify code in such a way that there is a mechanism to identify packets and for which hop that packet is received. Even if there are some packets lost, we find average values of each hop

with the existing data. This avoids unnecessary transmission of probes again and also utilizes whatever data that is collected irrespective of loss.

I suggest that even though we are assuming that ICMP is disabled in the core network, if at all routers reply with ICMP messages, we can also try to include them in our results. For example, if we could get IP address of some of the nodes and if there are similarities, we can use that just to emphasize that the program is working fine. This will just work as cross check mechanism. Also when we build a topology tree, we will also have IP information, which might bring more clarity.

If we get some information about the IP address, we can build topology with multiple sources and destinations and club the tree to form a single topology. At present, there is no way we can say that two nodes in topology are one and the same as we cannot distinguish it.

Bibliography

[1] A. Rowstron and P. Druschel, “Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems,” in *Middleware 2001*, pp. 329–350, Springer, 2001.

[2] M. Hefeeda, A. Habib, B. Botev, D. Xu, and B. Bhargava, “Promise: peer-to-peer media streaming using collectcast,” in *Proceedings of the eleventh ACM international conference on Multimedia*, pp. 45–54, ACM, 2003.

[3] J. Ni, H. Xie, S. Tatikonda, and Y. Yang, “Efficient and dynamic routing topology inference from end-to-endmeasurements,” *IEEE/ACMTransactions on Networking (TON)*, vol. 18, no. 1, pp. 123–135, 2010.

[4] R. C´aceres, N. Duffield, J. Horowitz, and D. Towsley, “Multicast-based inference of networkinternal loss characteristics,” *Information Theory, IEEE Transactions on*, vol. 45, no. 7, pp. 2462–2480, 1999.

[5] M. Coates, R. Castro, R. Nowak, M. Gadhiok, R. King, and Y. Tsang, “Maximum likelihood network topology identification from edge-based unicast measurements,” *ACM SIGMETRICS Performance Evaluation Review*, vol. 30, no. 1, pp. 11–20, 2002.

[6]. Amir Malekzadeh**,** NETWORK TOPOLOGY INFERENCE WITH END-TO-END UNICAST MEASUREMENTS, Pg 11-13

[7]. Amir Malekzadeh**,** NETWORK TOPOLOGY INFERENCE WITH END-TO-END UNICAST MEASUREMENTS, Pg 9

[8] M. Coates and R. Nowak, “Sequential monte carlo inference of internal delays in nonstationary data networks,” *Signal Processing, IEEE Transactions on*, vol. 50, no. 2, pp. 366–376, 2002.

[9] Amir Malekzadeh**,** NETWORK TOPOLOGY INFERENCE WITH END-TO-END UNICAST MEASUREMENTS, Chapter 4, Pg 29

[10] Amir Malekzadeh**,** NETWORK TOPOLOGY INFERENCE WITH END-TO-END UNICAST