International Journal of Networked and Distributed Computing, Vol. 4, No. 1 (January 2016), 22-31

On the Empirical Evaluation of Multicasting with Minimum Delay Variation

Nicklaus Rhodes
Department of Mathematics and Computer Science, The Citadel,171 Moultrie Street
Charleston, SC 29409, USA

Shankar Banik
Department of Mathematics and Computer Science, The Citadel,171 Moultrie Street
Charleston, SC 29409, USA
E-mail: shankar.banik@citadel.edu
www.citadel.edu

Abstract

In certain types of Internet applications, multicasting with Quality of Service (QoS) optimizations are required on both the end-to-end delay from the source to the destinations, as well as the difference in the end-to-end delay between the destinations. Examples of these applications include video-conferencing, online games, and distributed database replication. These applications require that the messages should reach all the destinations within a specified period of time and all the destinations should receive the message from the source almost at the same time. The solutions to the Delay and Delay Variation Bounded Multicast Network (DVBMN) problem involve generating a network which spans the source and the group of destinations such that the end-to-end delay from source to each destination is within a bound and the delay variation among all the destinations is minimum. The Chains heuristic proposed in the literature provides such a multicast network which achieves the tightest delay variation with minimum execution time. To test the performance of multicasting using the multicast network generated by Chains heuristic on the Internet, we have implemented a prototype of a multicast network using Chains and experimented multicasting with the prototype on PlanetLab. PlanetLab is an overlay testbed which connects more than 1000 nodes all over the world. We have analyzed the performance of multicasting using the prototype on PlanetLab. Results show that the measured value of maximum delay variation from the prototype approximately matches with the value calculated by Chains for a small number of destination nodes. However, as the number of destination nodes increases, the measured value of maximum delay variation from the prototype quickly increases..

Keywords: Multicast, Delay Variation, Shortest Path, Evaluation.

1. Introduction

Multicasting is an efficient group communication mechanism where one source node sends the same message to a group of destinations called the multicast group. Compared to unicast where a source node sends multiple copies of same message to the multicast group, multicasting conserves network bandwidth by sending single copy of the message to the multicast group. Multicasting is usually performed by constructing a multicast tree which spans the source node and the nodes in multicast group. Due to limitations of IP multicast, the recent trend in multicasting is to provide the service in the application layer by constructing overlay multicast trees [1][2][3][4].

Applications like video-conferencing, online games, distributed database replications use the advantage of multicasting. These applications require that each message should arrive at the destination within a certain period of time. These applications also require that

messages should reach all the destinations almost at the same time. In a video-conference, when a speaker speaks, everybody wants to hear the speaker at the same time. In an online games, when a player makes a move, all the players in the game want to see the move almost at the same time. When the stock database is updated, we want to make sure that it is replicated at all places almost at the same time.

The problem of designing a multicast tree which ensures end-to-end delay bound for each destination and the smallest delay variation among the destinations is known as the DVBMN (Delay and Delay Variation Bound Multicast Network) problem. Rouskas et. al. [5] have shown that the DVBMN problem is NP-Complete. Several heuristics have been presented in the literature [5][6][7][8] for the DVBMN problem. The Chains heuristic presented by Banik et. al. [8] outperforms all other heuristics by achieving the tightest delay variation with minimum execution time. In [8], the authors have demonstrated the performance of Chains by simulation experiments.

The criticism with any simulation environment is that it provides a controlled environment. The controlled environment does not resemble the Internet. Virtual overlay testbeds like PlanetLab [9] are used by researchers for testing the performance of network protocols and algorithms on the Internet. PlanetLab provides an environment where researchers can test network algorithms and protocols in an overlay network built with live nodes which are geographically dispersed and independent. These nodes will experience such interference that might normally be encountered by a live system, including packet delays and loss, CPU overloading, node outages and link breakage. Currently there are 1341 number of nodes available at 657 locations in PlanetLab. In our research we have used PlanetLab to study the performance of multicasting using the multicast network generated by Chains.

The Chains heuristic [8] uses *k*-shortest path algorithm HREA [10]. The algorithm is based on an acyclic, weighted, directed graph, with edge weights representing the transmission delay between nodes. Given a network, a source node, a group of destination nodes, and a delay bound, Chains heuristic finds a set of shortest paths for each destination such that the delay of each of these paths is within the end-to-end delay bound. Next, the heuristic selects a path for each destination such that the difference in delay of the paths among all destinations is minimum.

To observe the performance of multicasting using the multicast network generated by Chains, we have endeavored to create a prototype of multicast network using Chains and tested it on the PlanetLab. Our goal is to have a robust test of multicasting using Chains in the face of factors that are difficult to build into a simulation. This will allow for the measurement of actual delay variation that would be expected to occur when an application uses Chains for multicasting.

The rest of this paper is organized in the following way. In Section 2, we discuss the System Model that is used to describe the DVBMN problem space. We will describe Chains and HREA from the literature in this section. Section 3 describes the detailed design of the prototype. In Section 4 we discuss the experimental setup in PlanetLab. Section 5 provides discussion on the results that were obtained. Section 6 concludes the paper where we discuss the implications of the results, as well as opportunities for future work.

2. System Model

We consider the overlay network as a fully connected, weighted graph. Each node corresponds to a host in the network, and each link weight is the one-way delay between hosts connected by that link. On this network, we designate one node as the source node which sends data to a group of designated nodes called a multicast group. To enable multicasting, a network is created that includes the source node and each destination node from the multicast group and possibly other nodes, along with a selected set of links. For each destination node, a path from source node is defined as the set of links that connect the source node to that destination node. The delay of the path is defined as the sum of delay of all the links in that path. The source node will transmit packets that will be distributed to the destination nodes along these paths.

The problem of deriving a multicasting network for DVBMN problem is that of finding a set of paths in the overlay network from the source node to each of the destination nodes in the multicast group that will satisfy the following two bounds: the end-to-end delay bound which denotes the maximum allowable delay from source node to any destination node in the multicast group, and the delay variation tolerance which denotes the maximum allowable variation between the delay of

the paths from the source node to any two destination nodes in the multicast group. The problem is defined formally as follows:

Given an overlay network $G = (V, E)$, a source node $s \in V$, a multicast group $M \subseteq V$, a link delay function $d(e) = \Re+$ for each $e \in E$, an end-to-end delay bound Δ, and delay variation tolerance δ, find a multicast network $T = (V', E')$ which contains S and all nodes in M such that,

$$\sum_{e \in P(s, v)} d(e) \leq \Delta \text{ for each } v \in M \quad (1)$$

and

$$\left| \sum_{e \in P(s, v)} d(e) - \sum_{e \in P(s, u)} d(e) \right| \leq \delta, \quad \forall v, u \in M \quad (2)$$

where $P(s, v)$ and $P(s, u)$ denote the paths from s to v and s to u in T respectively.

Chains heuristic proposed by [8] finds such a T which achieves the minimum δ for a given Δ. Chains use k-shortest paths generated by HREA algorithm proposed by [10]. The following subsections provide an outline of Chains and HREA from the literature.

2.1. *Chains*

Chains[8] is a heuristic that efficiently generates a multicast network which meets QoS demands on Delay and Delay Variation. The authors in [8] have compared Chains heuristic with other heuristics for DVBMN problem in terms of time complexity and the delay variation achieved by the heuristic. This complexity is found to be the best among the other heuristics analyzed, while still offering the highest degree of delay variation optimization. The Chains heuristic has a time complexity of:

$$O(|E| + nk \log(|E| / n) + m^2 k)$$

where n is the number of nodes in the network, E is the number of links in the network, m is the number of destinations in the multicast group, and k is the number of shortest paths considered by Chains.

The Chains heuristic first computes the k-shortest paths using HREA algorithm [10] from source to each destination such that the delay of these paths are within the end-to-end delay bound. Then "chains" are formed by selecting a path for each destination from these shortest paths. Each chain will contain exactly m elements where m is total number of destinations in the multicast group. The value of a "chain" is defined as the maximum difference between the delay of any two paths in the "chain". Next Chains selects the "chain" which provides the minimum value.

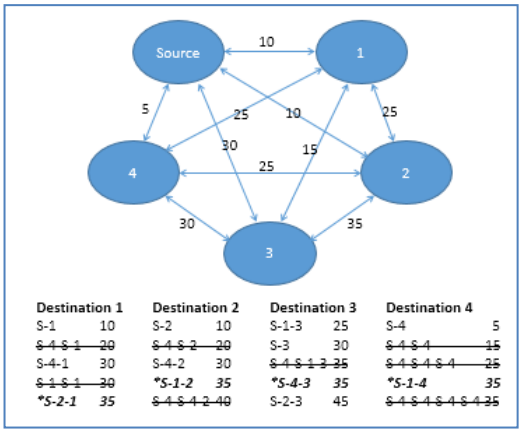

Fig. 1. Example of an overlay network with k-shortest paths for each destination

"chain1": 10 (Destination1) – 10 (Destination2) – 25 (Destination 3) – 5 (Destination 4)
Value of "chain1" is 20

"chain2": 10 (Destination1) – 10 (Destination2) – 25 (Destination 3) – 35 (Destination 4)
Value of "chain2" is 25

Fig. 2. Example of "chains"

For the example network in Figure 1 with a value of k=5, delay bound = 50, the resulting k shortest paths are shown for destinations 1, 2, 3, and 4. Notice that, as a result of the HREA algorithm, cyclical paths are generated. These paths are discarded by Chains. The remaining paths are then analyzed, starting with the least costly paths. To create a "chain", a set of paths is selected, one for each destination. Some examples of "chains" are shown in Figure 2. The value of a "chain" is calculated as the difference between the maximum delay and minimum delay in the "chain". The value of "chain 1" in Figure 2 is 25-5 = 20. The Chain heuristic finds the "chain" which gives the minimum value. For the example network in Figure 1, the "chain" with the minimum value is **35 (Destination 1) – 35 (Destination 2) – 35 (Destination 3) – 35 (Destination 4)** and the value of the chain is 0. The set of paths that form the "chain" with minimum value are marked with * in Figure 1. The multicasting network is formed using these paths.

2.2. *HREA*

Jimenez et. al. [10] provides an efficient algorithm, Recursive Enumeration Algorithm (REA), to calculate k shortest paths from a source to a destination in a weighted graph. The algorithm is a recursive function that uses the last found k-1 shortest paths to compute the shortest path k. The worst case computational complexity of REA is shown to be:

$$O(|E| + nk \log(|E| / n))$$

A further refinement to the REA algorithm is HREA. HREA uses a list data structure to record the length of the backpath and a pointer to that path for each step in the calculation. This allows for a more efficient implementation that will have the same worst case complexity but will perform better in most cases.

2.3. *Network Latency Graph*

The HREA algorithm requires as part of its input a weighted, directed graph. For our case, each node in this graph will represent a node in the overlay network, and each edge represents the connection between two nodes in the overlay network. The weight of each edge needs to be representative of the one-way transmission delay between the nodes connected by that edge.

Accurately generating this graph is absolutely essential to ensuring the validity of our experimental process. Any errors present in the latency measurements will propagate through the Chains calculation and potentially deliver a sub-optimal result.

We have assumed that the graph is fully connected because of the nature of our overlay network which lies on top of the Internet. With this topology, the only difficult part is the measurement of the inter-node delays. Existing implementations for measuring latency were considered. Because of the overhead of the existing solutions, we have developed our program for calculating the latency between two nodes in the overlay network. Given the rapidly changing nature of the underlying connections, the latency measurement program must be invoked relatively frequently to ensure an accurate representation of the one-way delay between nodes in the overlay network.

3. Design of the Prototype

The computation of finding Chains is delegated to a central node that does not participate in multicasting. To normalize the timing data, we use a time server that each node must query when initialized. The time server node also does not participate in multicasting. Each node uses a Multicast Forwarding Table to forward the multicast data through the multicasting network.

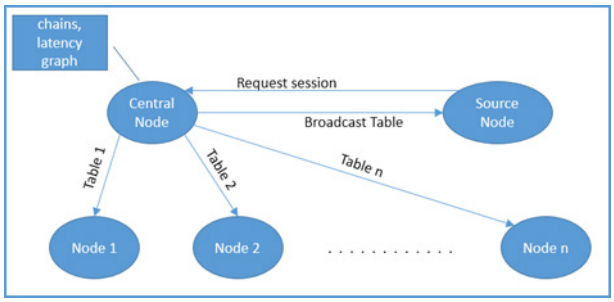

Fig. 3. Initializing Session

Figure 3 shows the setup for initializing a multicast session. To initialize a session, the source node will signal the central node that a new session is required and send a list of destination nodes. The central node will analyze the current overlay configuration, including the measured latency graph of the network, and run the Chains heuristic for that configuration and set of destination nodes. The central node will then send each node the Multicast Forwarding Table for that session.

Figure 4 shows the setup for multicast transmission and delay measurement. The source node will transmit a

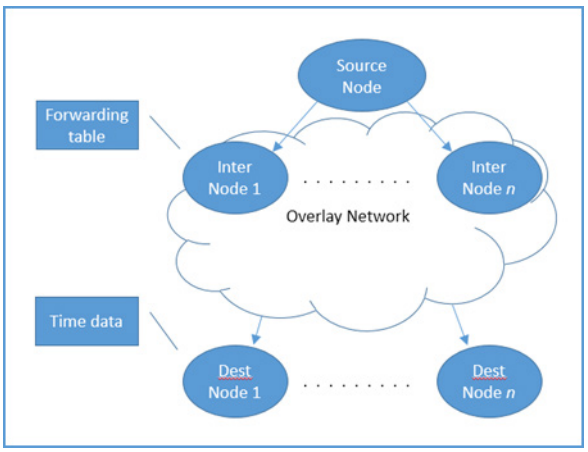

Fig. 4. Multicast Network

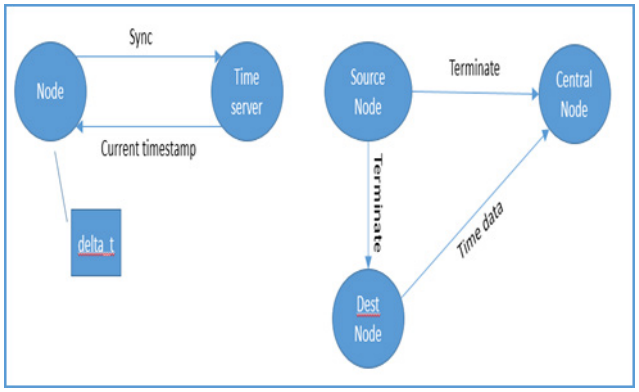

Fig. 5.Timeserver and Session Termination

series of packets. For each packet, the source node will look up each destination node in the multicast forwarding table and get the calculated next hop node. The current system time will be read and it will be included in the packet. Each intermediate node will forward the packet according to its multicast forwarding table for that session. When the packet reaches the destination node, that node will calculate the arrival time and store transmit and arrival times for that packet.

To terminate the session (Figure 5), the source node sends a final packet to each destination node, signaling the end of the multicast session. Each destination node will then send the time data it has accumulated, including the time offset and the set of time measurements for each packet received. After waiting for sufficient delay to allow each destination node to transmit the final session data, the source node sends the central node a signal to terminate the multicast session. At this point the central node will calculate the delay for each packet according to the following formula:

$$t\_delay = t\_dest - t\_source + (\Delta\_dest - \Delta t\_source), \quad (3)$$

where t_dest is the timestamp at the destination, t_source is the timestamp at the source, Δ_dest is the correction factor at the destination and Δt_source is the correction factor at the source.

The architecture of the prototype for the nodes is shown in Figure 6 and Figure 7. Each node implements a UDP socket listener that will be used to connect to the network. All incoming packets are parsed and processed by a pool of worker threads. The pool of threads was kept at 100 to allow for sufficient resources to reduce processing delay in the node.

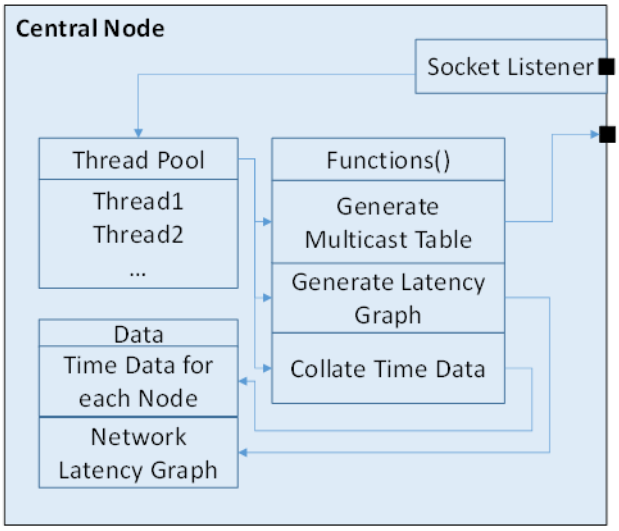

Fig. 6. Architecture of Central Node Program

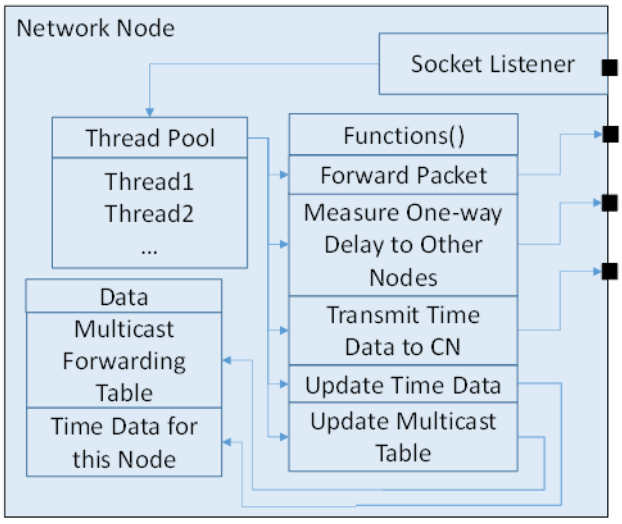

Fig. 7. Architecture of Network Node Program

The central node has three main modules: *Generate Multicast Table*, *Generate Latency Graph*, and *Collate Time Data*.

- *Generate Latency Graph* signals each node to measure the one-way delay to each of its neighbors and then builds the *Network Latency Graph* based on that data. The measurements are split evenly for each node in the network, with each node measuring $\frac{n-1}{2} + (n+1)\%2$ connections. These measurements are taken in parallel, and the output is saved to disk for reuse.

- *Generate Multicast Table* uses the Chains heuristic to build the Multicast Forwarding Table for each node in the multicast network. This function takes as input the network latency graph and eventually calls the Chains program. Finally, the Multicast Forwarding Table for each node for this session is sent to the corresponding node.
- *Collate Time Data* function gathers the measured transmit times from each destination node when the session ends. The gathered time data includes transmit and receive time for each packet, as well as the time offset for source and destination nodes.

Each network node has the following modules: Update Multicast Forwarding Table, Measure Delay in Network, Forward Packet, Update Time Data, and Transmit Final Time Data.

- *Update Multicast Forwarding Table* records a new table for that session. The table includes keys for the current session and for the packet destination, and gives the next hop for the incoming packet based on that information. This design will enable scaling for multiple source multicast for future research.
- *Measure Delay in Network* will record time to transmit to neighboring nodes and send that data back to central node. To measure the delays, we use an averaging scheme, where a stream of packets are sent from node to node, with the round trip time being measured and used to estimate the one-way delay. The measurements are continued until the successive averages converge to 3% of the predecessor.
- *Forward Packet* will send the packet to the next node in the destination's path based on the entry in Multicast Forwarding Table.
- *Update Time data* will record the time sent and time received at the destination node. The times are generated using Python time function which returns the system clock time as a floating point number in seconds. As such, the precision of our measurements will be limited by the time reported by the system. We assume that our maximum resolution of 1ms will be adequately supported in this case.
- *Transmit Final Time Data* will signal the end of a session, then the node will send time data back to the central node for processing. The time data in this case is the transmit and receive times for each of 30 packets and the time offset of the node as related to our time server.

4. Experimental Setup

4.1. *PlanetLab*

PlanetLab is a networking testbed designed to allow individual researchers to have access to arbitrarily many nodes which are geographically dispersed. On these nodes, the researcher can build and test network applications. The number of nodes available is over 1000, and these nodes are geographically dispersed all over the world. Individual accounts are called "slices" and are available through a Principle Investigator, usually within a school or organization. The Principle Investigators also contribute nodes, so that the organization is supported by the researchers who use it.

Our PlanetLab slice was initialized and populated with 10-50 nodes. Nodes were selected from the North America region for testing purposes. The number of destinations was varied from 5-25 nodes. 30 test packets were transmitted and the delay for each packet was recorded. Using the average value of delay for these 30 packets, the maximum delay variation between destination nodes was measured.

4.2. *Multicast Session*

The multicast session was populated with varying number of destinations: 5, 10, 15, 20, and 25. For each test, the size of the overlay network was kept at twice the number of destinations. So, for 5 destination nodes, the total size of the network is 10. The rationale behind this design choice was to try and eliminate the unnecessary complexity that might arise from having an excessively large network for a relatively small number of nodes. On the other hand, we wanted to provide a network of sufficient size to provide significant optimization from Chains.

For each test run, a total of 30 packets were transmitted, one after the other, to simulate a real

application sending a bulk data transfer, such as in a file transfer or database replication.

5. Results

The prototype was successfully implemented on the PlanetLab nodes. The prototype was tested using various values for the number of destination nodes: 5, 10, 15, 20 and 25. For each set of destination nodes, the central node generated a multicast network using Chains and calculated the maximum delay variation for the multicast network. Then the source node performed actual multicasting using the multicast network (derived by Chains) and measured the actual value of maximum delay variation. Table 1 shows the maximum *delay variations* that were calculated by central node by running Chains and the values measured by performing actual multicasting using the multicast tree generated by Chains in PlanetLab. Figure 8 plots the maximum delay variation VS number of destination nodes for the calculated the measured values.

Table 1: Comparison of Maximum Delay Variations for Calculated and Measured Values

Nodes	Calculated Max Variation (ms)	Measured Max Variation (ms)	Difference (measured - calc)	Correlation Coefficient
5	13	18.5	5.5	0.925
10	36	88.3	52.3	0.905
15	11	81.5	70.5	-0.167
20	8	158.1	150.1	-0.328
25	10	207.4	197.4	0.107

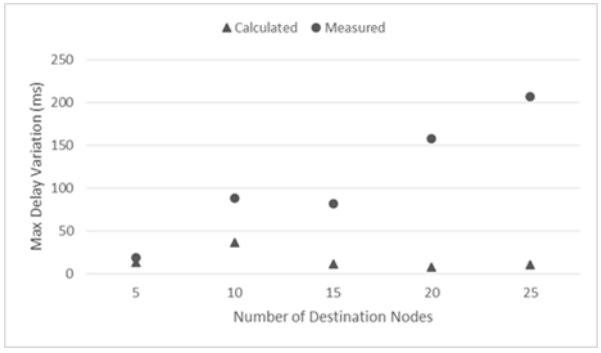

Fig. 8. Plot of Maximum Delay Variations VS Number of Destination Nodes

The differences between measured and calculated values are also given in Table 1. We observe that the difference between the calculated and observed measured values are quite small for 5 destination nodes (5 *ms*), then the difference increases dramatically as the number of destination nodes increases. The correlation between the calculated average delay for each node and the measured average delay were also calculated. We observe that there is a strong correlation for 5 and 10 nodes, but 15, 20 and 25 nodes have a very weak correlation between calculated and measured values.

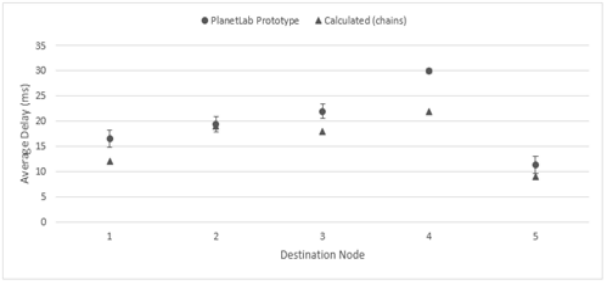

Fig. 9. Average Delay for Multicast Network with 10 Nodes and 5 Destinations

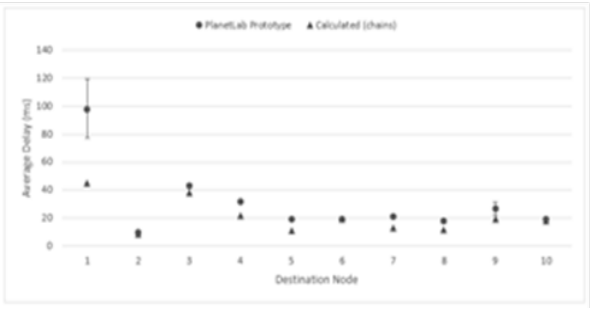

Fig. 10. Average Delay for Multicast Network with 20 Nodes and 10 Destinations

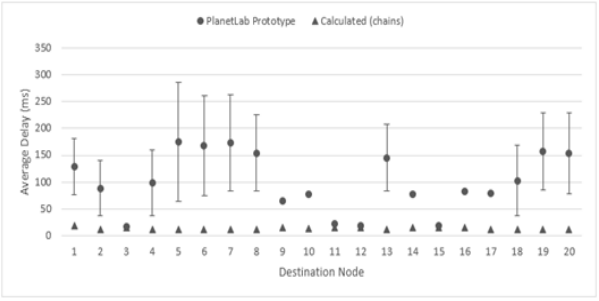

Fig. 11. Average Delay for Multicast Network with 30 Nodes and 15 Destinations

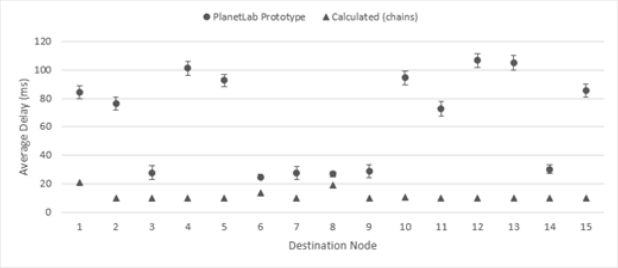

Fig. 12. Average Delay for Multicast Network with 40 Nodes and 20 Destinations

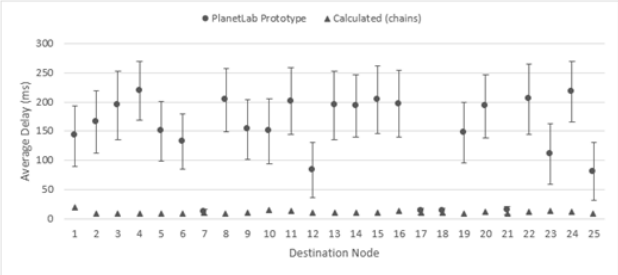

Fig. 13. Average Delay for Multicast Network with 50 Nodes and 25 Destinations

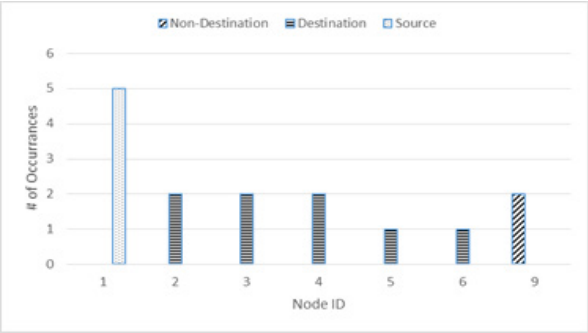

Fig. 14. Number of Paths Each Node Participates in for a Multicast Tree with 10 Nodes and 5 Destinations

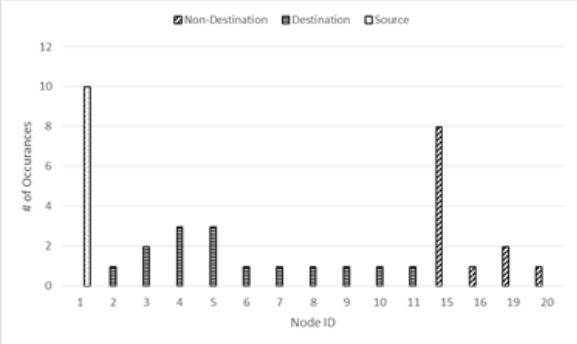

Fig. 15. Number of Paths Each Node Participates in for a Multicast Tree with 20 Nodes and 10 Destinations

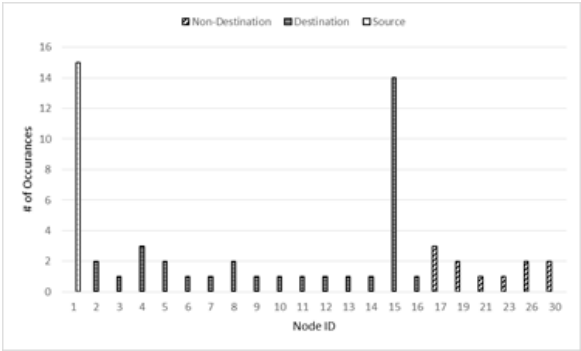

Fig. 16. Number of Paths Each Node Participates in for a Multicast Tree with 30 Nodes and 15 Destinations

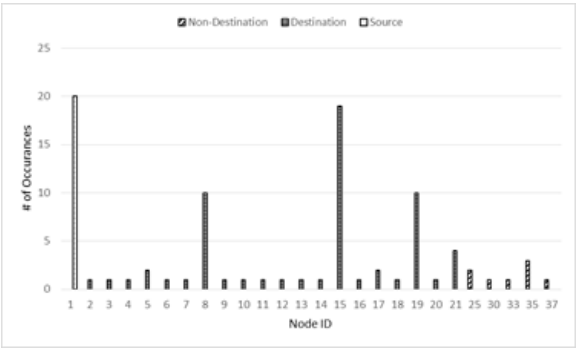

Fig. 17. Number of Paths Each Node Participates in for a Multicast Tree with 40 Nodes and 20 Destinations

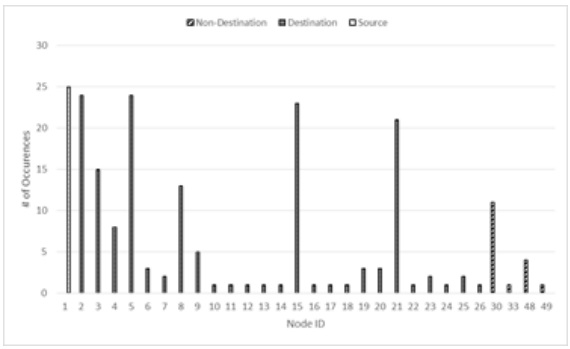

Fig. 18. Number of Paths Each Node Participates in for a Multicast Tree with 50 Nodes and 25 Destinations

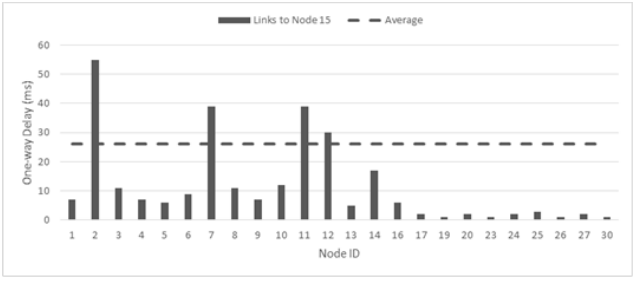

Fig. 19. Link Latencies for Links Connecting Node 15 of 30

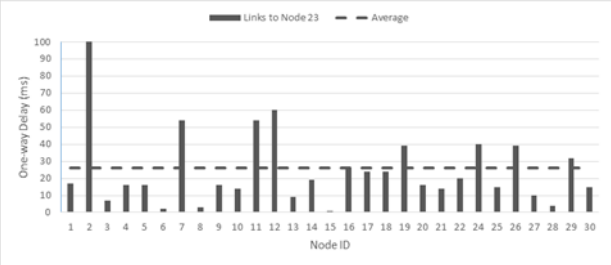

Fig. 20. Link Latencies for Links Connecting Node 23 of 30

The resulting average delay for each node for different values of destination nodes (5, 10, 15, 20, 25) are shown in Figures 9 – 13. The vertical error bars on the measured averages represent the standard deviation in the sampled data set. We observe that the measured versus calculated values match quite well with a small number of destination nodes.

However as the number of destination nodes increases, the delay experienced by the destination nodes increases dramatically. The maximum delay variation between nodes also increases which we have observed from Figure 8.

It is interesting to observe that the average delay and the delay variation both increase, and taking a closer look at Figures 9 – 13, we notice that while some destination nodes have dramatic increase in delay, others will still closely match the calculated values. Upon examining the destinations that experience dramatic increases in delay, we note that they share common intermediate nodes. These nodes typically have low latency links with some other node in the path, and therefor are more commonly selected by the shortest paths algorithm. The nodes that more closely match the calculated delays typically do not share their intermediary nodes with other paths, or they are not shared with many paths.

One probable cause for the increase is that there is an inordinate amount of processing time associated with the packets being routed through those very common nodes. This might be due to strictly I/O time in the interface, or insufficient threads being assigned to the processing of incoming packets. Figure 14-18 show the number of occurrences for each node in the paths that are used in the multicast tree with various numbers of destination nodes. These results reinforce the earlier observation that a single node might become heavily burdened in the selection of paths during the construction of the multicast tree by Chains. The simulation result in Figure 15 shows that the destination Node 15 takes part in 14 out of 15 multicast paths. To gain insight into this phenomenon, we have assembled a chart of the latency values for Node 15 and compared them to the average latency measurement across all nodes in Figure 19. As we can see, there are a significant number of links that are much faster than the average. By comparison, Figure 20 shows the link latencies for Node 23, which is only present as an intermediate node in one path. Most of the links for Node 23 have close to the average latency, with few outliers spread evenly amongst higher and lower than average.

6. Conclusion

In this paper we have considered the problem of designing a delay and delay variation bounded multicasting network. The Chains heuristic proposed by authors in [8] outperforms other heuristics in terms of time complexity, yet achieves the tightest delay variation. In this research, we have built a prototype for creating a multicast network using Chains and tested its performance on PlanetLab Network, which is a virtual testbed that connects more than 1000 machines all over the world. The results clearly show that there are more considerations to account for in the actual multicasting than just the delay when building the multicast network. Although the simulation shows that Chains builds a multicast network efficiently and quickly, in practice we observe that the delay from the prototype quickly diverges from the calculated values as the number of destination nodes increases.

One possible source of the unwanted delay from the prototype is the excessive processing time in each node. As the number of destination nodes increases, the number of packets being multicast increases linearly. When the number of packets reaches and exceeds the ability of a node to process them, those packets will be buffered until resources become available to process. Therefore, once this point is reached any new packets arriving will experience additional delay. We can assume that the packet buffering time at each node will increase at a roughly linear rate, i.e. a packet arriving with 30 packets buffered at that node will experience at least 30 times longer processing delay than a packet arriving with one packet buffered.

We have also observed at least one possible cause for the apparent preference of Chains for choosing paths

with certain nodes over others. Although the underlying *k*-shortest path algorithm is efficient, it provides cyclic paths which are discarded by Chains. As a result, the number of useable paths for Chains reduces. This increases the likelihood of selecting the paths containing nodes with low-latency links more frequently. This situation makes those nodes overloaded which slows the performance of the multicast with Chains.

Future enhancements of this research include extending our testing of the prototype to multisource multicasting. In addition we would like to allow nodes to join and leave the multicast session and study the impact of this on test results. Also, we would like to develop an efficient multicasting protocol to allow efficient use of network links. Presently, for each destination, a new packet is generated during each step. As we have shown, many destinations share common paths, so it should be possible to eliminate many packets along these paths by properly utilizing the existing multicast forwarding tables and path data.

7. References

[1] S. Baneerjee, C. Kommareddy, K. Kar, B. Bhattacharjee, and S. Khuller, “Construction of an Efficient Overlay Multicast Infrastructure for Real-Time Applications,” Proc. IEEE INFOCOM ’03, Mar. 2003.

[2] E. Brosh and Y. Shavitt, “Approximation and Heuristic Algorithms for Minimum Delay Application-Layer Multicast Trees,” Proc. IEEE INFOCOM ’04, Mar. 2004.

[3] A. Riabov, Z. Liu, and L. Zhang, “Overlay Multicast Trees of Minimal Delay,” Proc. 24th IEEE Int’l Conf. Distributed Computing Systems (ICDCS ’04), 2004.

[4] S.Y. Shi and J.S. Turner, “Multicast Routing and Bandwidth Dimensioning in Overlay Networks,” IEEE J. Selected Areas in Comm., vol. 20, no. 8, pp. 1444-1455, Oct. 2002.

[5] G.N. Rouskas and I. Baldine, “Multicast Routing with End-to-End Delay and Delay Variations Constraints,” IEEE J. Selected Areas in Comm., vol. 15, no. 3, pp. 346-356, 1997.

[6] P.-R. Sheu and S.-T. Chen, “A Fast and Efficient Heuristic Algorithm for the Delay and Delay Variation Bound Multicast Tree Problem,” Proc. 15th Int’l Conf. Information Networking (ICOIN ’01), pp. 611-618, Feb. 2001.

[7] S. Kapoor and S. Raghavan, “Improved Multicast Routing with Delay and Delay Variation Constraint,” Proc. Global Telecomm. Conf. (GLOBECOM ’00), vol. 1, pp. 476-480, 2000.

[8] S.M. Banik; S. Radhakrishnan; C.N. Sekharan, "Multicast routing with delay and delay variation constraints for collaborative applications on overlay networks," Parallel and Distributed Systems, IEEE Transactions on , vol.18, no.3, pp.421,431, March 2007.

[9] https://www.planet-lab.org/

[10] V. Jimenez and A. Marzal, “Computing the k shortest paths: a new algorithm and an experimental comparison,” Proc. Third Workshop, Algorithm Eng., pp. 15-29, 1999.

[11] O. Gurewitz; I. Cidon; M. Sidi, "One-way delay estimation using network-wide measurements," Information Theory, IEEE Transactions on , vol.52, no.6, pp.2710,2724, June 2006.

[12] G. Bela; H. Piroska, “Using PlanetLab to implement multicast at the application level,” International Journal of Computer Networks & Communications (IJCNC) Vol.3, No.1, January 2011.

[13] S.Y. Shi; J.S. Turner, "Routing in overlay multicast networks," INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE , vol.3, no., pp.1200,1208 vol.3, 2002.