Logistical Multicast for Data Distribution

Jason Zurawski, Martin Swany
Department of Computer and Information Sciences
University of Delaware, Newark, DE 19716
{*zurawski, swany*}*@cis.udel.edu*

Micah Beck, Ying Ding
Department of Computer Science
University of Tennessee, Knoxville, TN 37996
{*mbeck, ying*}*@cs.utk.edu*

Abstract

This paper describes a simple scheduling procedure for use in multicast data distribution within a logistical networking infrastructure. The goal of our scheduler is to generate a distribution schedule that will exploit the best network paths by using historic network performance information. A "spanning tree" is constructed between available logistical depots to help reduce the overall time of data movement.

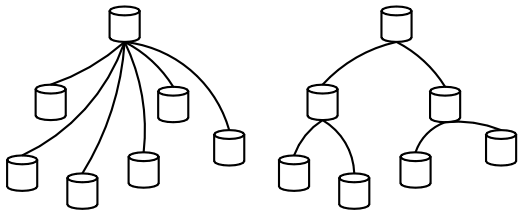
Our hypothesis is that we can generate appropriate schedules from historical network measurements. In order to evaluate the scheduling procedure we have employed the multicast operation used in the Internet Backplane Protocol (IBP) middleware suite. Investigation into the merits of such a scheduling procedure involved a control group that performs a broadcast to a set of logistical depots and an experimental group that is configured to perform a multicast via schedules generated based on historical network data. All testing was conducted on PlanetLab, a distributed network service testbed.

1 Introduction

One of the chief sources of overhead in Grid environments is that of data movement. Computing systems and networks are growing ever faster, but the relative cost of data movement remains high. Logistical Networking [9] (LN) was developed to leverage storage and processing in the network to improve performance and functionality in distributed computing environments. The LN mode positions "depots" at strategic locations in the network. These depots may then be used on an as needed basis to store information "en route" across a network [19]. One obvious goal of this activity is to bring data closer to the places where it will be used [8].

A common data distribution scenario is a "multicast" of data from a single host to multiple recipients [24, 12]. Usually a tree is formulated such that data is transmitted to certain nodes that in turn will send to other nodes [21, 7, 2, 26]. Distribution trees, for the most part, are constructed utilizing some notion of underlying network topology; this information is often gleaned from network measurement data [25, 11].

In general, the performance of a broadcast operation as shown in Figure 1a, in which a single source distributes data to each destination, can be improved upon greatly. One way to do so is to assemble a multicast tree with the initial data source as the root of the tree as shown in Figure 1b.



(a)Broadcast **(b)Multicast**

Figure 1. The broadcast operation requires the root node to do all of the work, whereas the multicast approach allows intermediate nodes to assist

This work presents a basic scheduler for constructing multicast trees that utilize LN infrastructure. To evaluate our approach, we utilize intermediate storage in the network as enabled by the Internet Backplane Protocol (IBP) [19]. The goal of IBP is to provide LN capabilities to client applications [9]. Although IBP offers other avenues to improving performance through data locality, this work focuses only on using IBP as a vehicle for executing synchronous, one-to-many data movement.

Previous incarnations of the logistical scheduler used

for this work were found in various implementations of the Logistical Session Layer (LSL) [21, 20]. In addition to modifying the scheduler to communicate with IBP, the representation of network performance information has been generalized. Whereas the former scheduler depended on data from the Network Weather Service (NWS) [23], the current implementation will consume network information from other sources, as NWS data is not currently available for the entire PlanetLab testbed. To date, it has proven to be difficult to construct a large-scale measurement infrastructure capable of providing network performance information on demand. As such, we focus on commonly-available historical measurement data in an effort to determine its efficiency as a basis for scheduling network data movement.

This paper will proceed as follows. First, we will briefly outline our general approach to schedule construction. Next we will discuss the network measurement data that we utilized. We will then describe our experimental framework and present results from our measurements.

2 Multicast Approach Overview

Multicast is an often investigated topic. Previous work has addressed issues such as multicast within a fast message passing cluster [22, 4], as well as application level multicast on public networks [15,24,5,3,12,26,10]. Our approach is unique in that there is no persistent multicast tree or peer relationships. Rather, we instantiate the multicast structure from the edge of the network utilizing existing LN resources.

A multicast schedule often relies on knowledge of the target resources as well as status information regarding the network. Once network information is known, it is possible to generate a distribution directive in the form of a "spanning tree" [21, 7, 20, 26, 2]. This tree is consumed by the multicast operation to decrease the overall time to distribute some amount of data to all nodes. Our approach is unique in that is considers a very low level of network measurement infrastructure and makes no measurements internally.

Our scheduler takes into account available storage locations as well network performance information gathered over some period of time as part of the PlanetLab project [21, 20, 6, 25, 11, 16, 17]. This derived multicast schedule makes use of the perceived best paths to disperse the data as possible. Although it is quite possible to use a variety of criteria when creating a schedule such as the cost or utilization of specific computational resources, this paper will deal strictly with maximizing the speed of the data transfer and hence minimizing the total time needed.

2.1 Performance Measurements

Frequently requesting data from an on–demand system such as the NWS can easily cause a bottleneck when generating dynamic schedules; this can become especially evident within testbeds the scale of PlanetLab [18, 6]. For the reason that active measurements are hard to scale, this work will show that it is possible to generate acceptable data movement directives through the primary usage of historical measurements; virtually no reliance on current network status (beyond that of connectivity) is required by this system at run time.

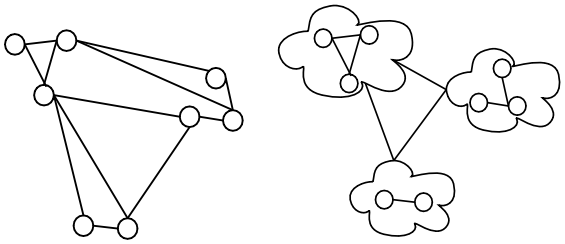
Various tools exist to measure and analyze performance data over a network; a discussion of each available method as well as the inherent traits of each are well beyond the scope of this paper. We will focus on information collected from results of running an Iperf [14] test to measure achievable bandwidth to and from all available computing resources on PlanetLab. Iperf measures the achievable bandwidth between a pair of hosts for a given amount of data.

The distribution spanning tree is created based on this network performance information. Ultimately, the goal of performance measurement is to get as much relevant information as possible while impacting the medium as little as necessary. This has led to reliance upon available measurement sources to avoid unnecessary impact on other network traffic as well as simplify the collection process.

The input to the scheduler is the maximum observed bandwidth between a pair of resources for a given period of time. The use of the maximum value is based on the intuition that the best measurement we have observed comes the closest to representing the link with no cross traffic. While current link conditions may change, our goal is to determine whether a more general notion of the achievable bandwidth is sufficient, as the cost of basing our decision on this data is certainly easier.

2.2 Measurement Collection and Analysis

Each PlanetLab machine runs a client application that gathers Iperf measurements for all other PlanetLab machines; the results are stored in publicly accessible log files [17] and are updated on an incremental basis. A complete set of Iperf data for all PlanetLab resources takes approximately one week to generate. To build an adequate basis of measurements it is necessary to induce several consecutive weeks. These accrued values are gathered and built into a rudimentary N x N matrix of performance information where N is the total number of PlanetLab machines running a client. Despite there being a requisite quantity of data, additional processing steps are necessary.



(a)Standard **(b)Enhanced**

Figure 2. Network sites offer a plurality of resources. It is possible to replicate measurements across similar resources.

Some instantiations of the performance matrix can be sparse due to unresponsive machines. Observation has shown that the algorithm proceeds in a step-wise fashion, never overtaxing a single resource for a given time period. This observation is able to explain the data discrepancies; when the algorithm visits a node early in the week it may be responsive and provide some information, later the resources could be unavailable and unable to provide any performance data. This fact is reason enough to only be concerned with the maximum observed value over the time series.

An example of incomplete measurement data is shown in Figure 2a. The sparsity of the matrix can be an issue when attempting to construct a distribution schedule. Observation of the figure shows that some nodes are far from completely connected. To correct this obvious discrepancy it is possible to exchange information between multiple machines under the assumption that they are in close proximity to each other and share a common link to the outside world. This relationship can be tested by examining both the IP address and netmask configuration; related machines may then be included in the interpolation process to exchange relevant data.

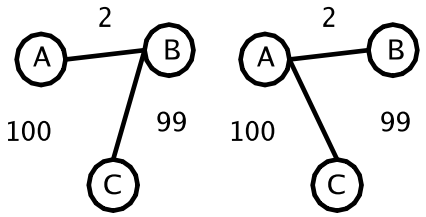
This interpolation technique helps to populate the matrix resulting in a greater variety of information for the eventual scheduling. In essence, we create information on a site by site basis instead of a computer to computer basis [7]. An example of this transformation can be seen in Figure 2b.

The extracted measurements are not intended to provide any sense of symmetry. It is quite possible that measurements observed between nodes a and b can differ greatly from that of their converse. This anomaly can best be addressed by examining the administrative policies of PlanetLab. A resource that decides to set incoming and outgoing bandwidth limits may do so when necessary; if at a given time in the day the resource is nearing the peak of allowed usage the network will "constrict" to allow for less available bandwidth. This permits resources to still function, albeit in a slightly degraded manner.

Exploring the issue of machines within a close proximity a little further reveals a nuance that is generally overlooked: nearby machines often have high available bandwidth. In most situations the machines are connected via the same LAN and should be able to achieve near maximum bandwidth between each other. To draw attention to this fact all machines on the same LAN are given an artificially high bandwidth value. This assures that a multicast will be delivered only once to a single site, thus increasing throughput even further [20, 21].

2.3 Experimental Considerations

Due to small variations in network measurements, machines with functionally similar connectivity have slightly different edge values. To keep the generated schedules simple, we would like to consider measurements within some ε of one another as the same [21]. In practice, this amounts to a modification to the scheduling algorithm such that "better" paths are only added to the tree if they improve on the existing path by more than ε. Thus, the constructed spanning trees can be made more conservative by increasing ε.



(a)MST **(b)MST with ε**

Figure 3. Distribution graph featuring three nodes; By adjusting the value of ε we can limit the depth of the directive.

Consider the graph shown in Figure 3a. The true minimum spanning tree should include the edges from A to B and B to C. A more desirable tree, as shown in Figure 3b, would employ the edge A to C as well as A to B; this effort would further reduce the depth of the generated directive albeit adding to the branching factor.

Schedules bearing a tree structure that is uncommonly deep and with minimal branching are undesirable for a multicast. A similar structure that is shallow as well as wide is equally unappealing. Experiments have been designed to illustrate the effect that adjusting ε can have on the generation of a well balanced data distribution. It will be shown that altering ε can reduce the depth and

increase the width of a spanning tree well within acceptable guidelines.

The ε threshold is rather sensitive in relation to observed bandwidth because of the low variance; throughout the world the difference in achievable bandwidth on any given network differs by no more than a small order of magnitude. This threshold value will be presented as the highest possible ε that can achieve the minimum data movement time.

2.4 Schedule Creation

Data collection and analysis aside, the actual algorithm to calculate a dynamic multicast schedule is the crux of this work. The data we bring to this algorithm is akin to a completely connected graph. Graph vertices are known to be the different computing resources that house a logistical depot. Edges are the known bandwidth figures and connect the various vertices. Because the collected data may be far from perfect it is necessary to fabricate "expensive" edges between hosts that may not be able to communicate due to an absence of data. This expensive property of a given edge will ensure that the algorithm will complete given the basic information provided.

To evaluate the cost from vertex to vertex within the graph it is necessary to examine each edge along the way. The cost of entire path as demonstrated in [20] is not the sum of each link, but simply the cost of the slowest (i.e. most "expensive") link. To produce minimum spanning tree, we need a metric where smaller values are better. Since we are operating with achievable bandwidth measurements, we can convert this to "transfer time" estimates by considering $1/bandwidth$ as the "values" for edges between vertices.

To optimize paths in a graph environment it is necessary to minimize the maximum weights – thus the Minimax approach is appropriate. The minimax algorithm is a well known recursive algorithm, normally used for selecting the next desirable move in intelligent systems. For our purposes we will examine a directed graph $G = (V, E)$ where V is the set of vertices and E is the set of edges. It is our goal to make the move that maximizes the minimum value of the position resulting from subsequent moves. For brevity a step by step description of the minimax algorithm has been omitted, but result of each step of the process gives the best possible move to make for a given node. The end result presents a distribution tree that visits each node utilizing paths with maximum bandwidth.

Minimax is a greedy algorithm and has much in common with Dijkstra's Shortest Path algorithm [13] as well as an enumeration of other similar problems. The cost to build such a tree is understood to have an asymptotic running time of $O(NlogN)$ when keeping the edges of the graph in order.

Number	Name
1	dschinni.planetlab.extranet.uni-passau.de
2	pl1.cs.utk.edu
3	pl2.cs.utk.edu
4	lefthand.eecs.harvard.edu
5	plab1.ee.ucla.edu

Table 1. Hosts used to illustrate directive construction.

The implementation is always aware of the starting node (v_{start}); it is from this node that we construct all eventual paths in the graph. It is of note that any node can be the root node as long as it features network connectivity. Nodes that suffer poor connectivity (possible due to bandwidth restrictions) may still function as the root node, but may delegate most or all of the multicast responsibility to better suited nodes. In experimental settings it was not uncommon, given a slow initial node, that control would be transfered to a more suited environment such as resources located on the Internet2 Abilene backbone [1]. Once the multicast has propagated along the fastest connections it will slowly start to reach leaf nodes in parallel.

2.5 Schedule Example

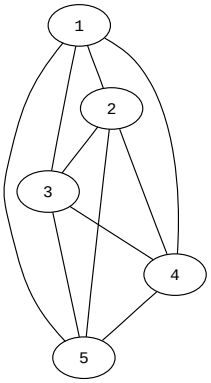


Figure 4. Completely connected graph featuring a subset of the PlanetLab nodes. Each node is understood to be accessible and running an IBP depot.

We will illustrate how a spanning tree is constructed using the hosts in Table 1. These resources exist within PlanetLab in various parts of the world. Over the span of

a week the Iperf data will be gathered from each node to reveal the bandwidth that exists between all other participants. This example will construct a small graph using the listed nodes and extract a spanning tree based on the measurements. The format of such a completely connected graph is in Figure 4.

	Node 1	Node 2	Node 3	Node 4	Node 5
Node 1	∞	5.84	5.84	-1	8.43
Node 2	1.12	∞	∞	7.31	1.05
Node 3	1.12	∞	∞	7.31	1.05
Node 4	4.94	3.63	3.63	∞	2.49
Node 5	-1	5.53	5.53	-1	∞

Table 2. Observed maximum achievable bandwidth between hosts.

Maximum achievable bandwidth times for the nodes mentioned in Table 1 are shown in Table 2. These illustrate the observed measurements between hosts over a finite amount of time, namely a one month period. The original data was far from complete due to connectivity or congestion problems within the network that may have prevented necessary information from being collected. It is of special interest to note that these nodes have been interpolated within site boundaries. Augmentation of values, such as creating an "expensive" edge of -1 where data was absent as well as maximizing the bandwidth between related hosts is also seen in this example.

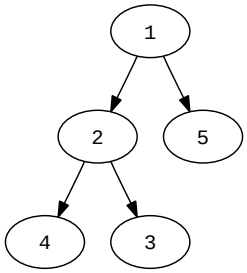


Figure 5. Resultant spanning tree created from the graph in Figure 3.

After the scheduling program has consumed the available data the spanning tree pictured in Figure 5 is created. This tree illustrates the best paths to take from the root node (Node 1) and will ensures a stop at every other listed depot. Note how the algorithm is able to recognize the boundary of a specific site (namely UTK) and take advantage of the proximity of the nodes. It has been observed in [20] that distribution of information to a single node at each site with additional transfer to occur internally is much more efficient than multiple distributions to a site.

3 Experimental Results

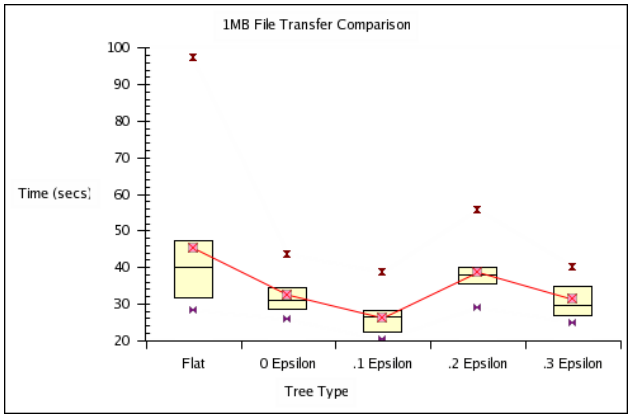


Figure 6. Comparison of 1MB data movement via multicast between logistical resources.

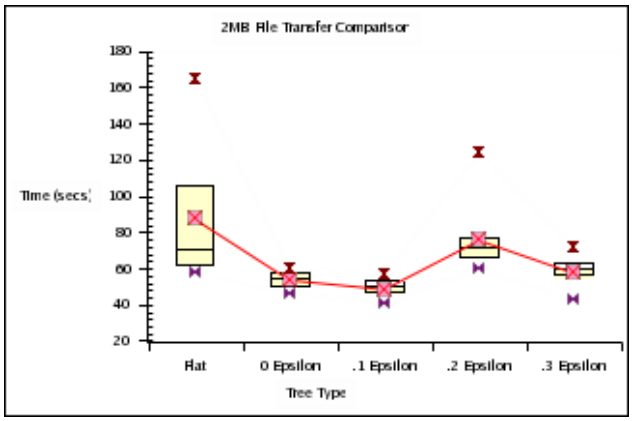


Figure 7. Comparison of 2MB data movement via multicast between logistical resources.

A small set of approximately 40 nodes was considered for this experiment; the root node was varied between a small subset of approximately 5 nodes. It was not always possible to reach all nodes during the trial partially because of the dynamic nature of the PlanetLab testbed. An important realization is that before each trial all nodes are examined; a node that is perceived to be unavailable is always removed from consideration. This

extra step can ensure a properly constructed tree that returns accurate results.

3.1 Experimental Methodologies

Two classifications of tests were performed upon the set of nodes – broadcast from a single node and multicast among nodes. The goal being to deliver information to all available nodes in the shortest amount of time. The multicast experiment is further divided into distinct parts, each part dictated by the variability of ε we choose to consider when making our distribution schedule.

It is our contention that a well chosen ε ensures a tree that is balanced in both depth and width. The alternatives to this of course are trees that become wide and shallow; essentially changing the multicast into a broadcast.

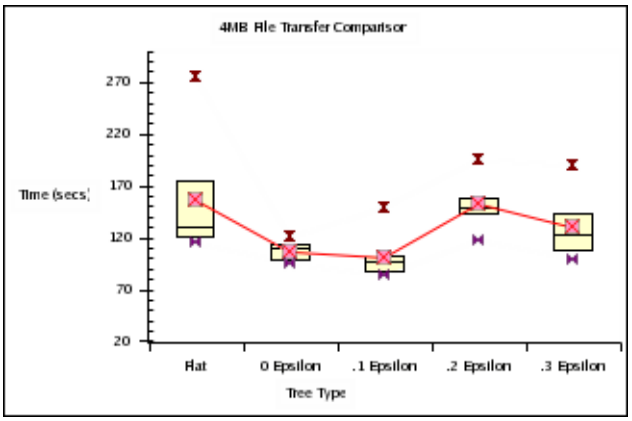


Figure 8. Comparison of 4MB data movement via multicast between logistical resources.

Each part of the experiment aims to transfer files of varying sizes; files ranging from from 1 to 16 megabytes were used for verification purposes. Our intention is to show that no matter the size of a transfer, multicast with adaptive scheduling and appropriate ε choice will always produce a better result than that of broadcast. Ranging the file size will also illustrate that the overhead of our approach is justified even for small file sizes.

Each file size was tested with several ε values to reveal a relationship between multicast performance and tree topology; consideration was given to the following test cases:

- Simple broadcast tree
- Multicast without ε
- Multicast with ε being .1
- Multicast with ε being .2
- Multicast with ε being .3

It is of note to realize that despite the static nature of the data it was always observed that different distribution trees were generated depending on the chosen root node as well as the ε choice. Each distinctly crafted tree was then tested with the various file sizes producing the experimental presented here. Results for each file size are shown in Figure 6 through Figure 10. These graphs show the observed maximum, average, and minimum data transfer times as well as colored regions illustrating the 25^{th}, 50^{th} and 75^{th} percentiles of the observed data.

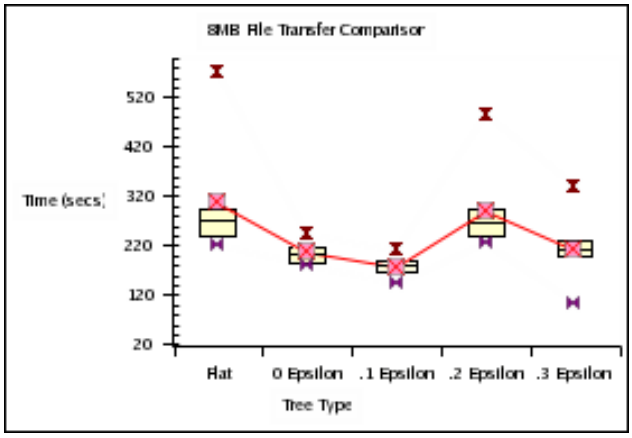


Figure 9. Comparison of 8MB data movement via multicast between logistical resources.

3.2 Analysis

In every instance it is clear to see that having an ε value of .1 will produce the fastest transfer times. It can also be said that broadcast trees produce the most undesirable results, followed closely by the situation where ε is .3. As ε increases it forces the tree to sacrifice depth in favor of greater branching factor at each level. This basically *flattens* the tree making it resemble the overall structure of the broadcast method.

The effect of ε on tree depth is shown in Figure 11. The tree depth as ε increases experiences a steady decline. Having a tree that is too deep is of no benefit in the data movement process because it necessitates additional hops. Each additional hop takes valuable time during the multicast.

Although ε can limit the depth of a data movement directive, it does add to the branching factor at each level. A reasonable branching factor can improve parallel data

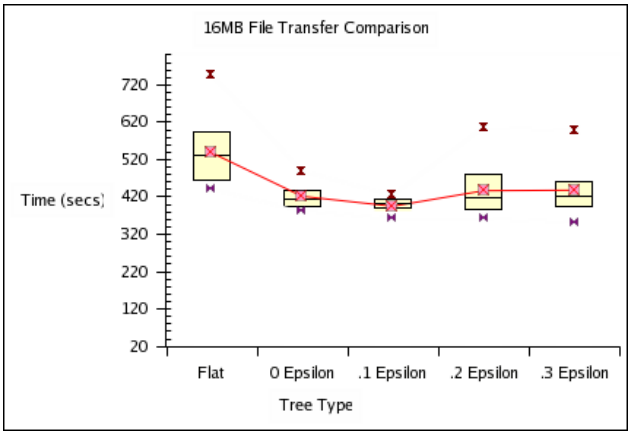


Figure 10. Comparison of 16MB data movement via multicast between logistical resources.

movement. Reduction of tree depth and the corresponding increase in branching can negate much of the potential speedup.

Ideally, it is desirable to branch equally among the various levels and, when applicable, evenly distribute the multicast to better suited nodes. It is not always the case that all nodes are created equal. Trees may become lopsided when a single node dominates all others by achieving exemplary bandwidth values. By twiddling ε we can level the playing field and eliminate the dominating effect that a resource may have on the initial dataset.

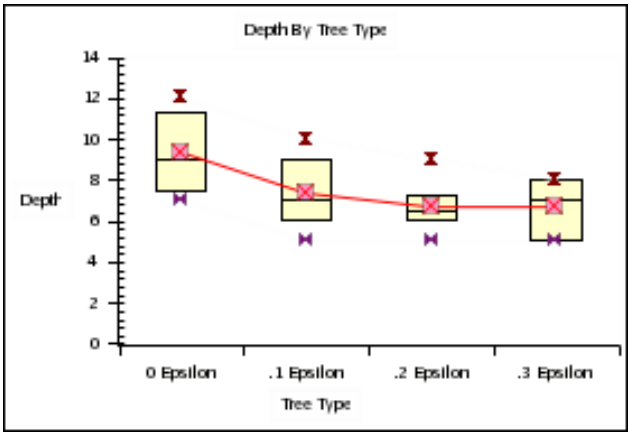


Figure 11. Tree depth as ε increases.

A graph demonstrating the effect of taking the aggregate sum of the branching factors for all nodes is shown in Figure 12. This aggregation is able to demonstrate the steady increase in branching as the value of ε increases. A reasonable amount of branching at the appropriate times is of great benefit to the performance of this system. The trend demonstrated in Figure 12 illustrates an appropriate choice of ε can produce an acceptable branching factor that will consistently perform better than broadcast as well as less optimal choices of ε.

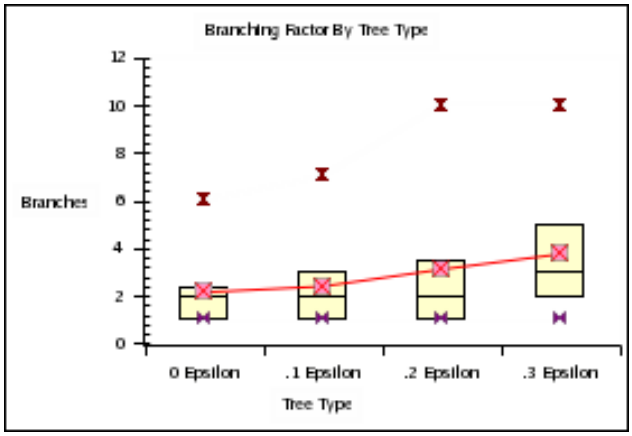


Figure 12. Aggregate branching factor for various values ε.

4 Conclusion

We have shown that it is possible to achieve a significant speedup in logistical networking multicast over that of a broadcast operation using network performance information gathered and archived by an unrelated service. This multicast procedure consumes a distribution directive in the form of a minimal spanning tree constructed from network measurements over the course of a month. These static measurements are sufficient in describing routes that have the potential to provide the fastest transfer times. This approach provides the basis for a general-purpose tool that could be used with minimal effort.

Future plans for this project revolve around the search to create a directive that utilizes bandwidth, latency, and other factors in the formation of the multicast tree. These hybrid directives should be able to increase efficiency by an even larger margin. Additionally, long-lived data transfers may need to adapt to change in conditions over the lifetime of the session.

Because of the scale of this project historical measurements were the primary form of mediation when creating a distribution schedule. Future work centers around issues regarding the availability of network measurements; historic as well as current. A comparison of the two approaches should reveal the value of using historic measurements versus that of dynamic.

References

[1] Abilene. http://www.ucaid.edu/abilene.

[2] M. Adler, T. Bu, R. K. Sitaraman, and D. Towsley. Tree Layout for Internal Network Characterizations in Multicast Networks. In *Networked Group Communication*, November 2001.

[3] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. Resilient Overlay Networks. In *ACM Symposium on Operating Systems Principles.*, 2001.

[4] C. Anglano, C. Casetti, E. Leonardi, and F.Neri. Multicasting at the Host Interface Level in Wormhole Networks. In *Int. Conference on Network Protocols. Austin, TX*, 1998.

[5] S. Banerjee, S. Lee, B. Bhattacharjee, and A. Srinivasan. Resilient Multicast using Overlays. In *ACM SIGMETRICS. San Diego, CA*, June 2003.

[6] S. Banerjee, M. Pias, and T. Griffin. The Interdomain Connectivity of PlanetLab Nodes. In *Passive and Active Measurements. Sophia-Antipolis, France*, 2004.

[7] M. Beck, Y. Ding, E. Fuentes, and S. Kancherla. An Exposed Approach to Reliable Multicast in Heterogeneous Logistical Networks. In *Workshop on Grids and Advanced Networks*, May 2003.

[8] M. Beck, T. Moore, and J. Plank. An End-to-end Approach to Globally Scalable Network Storage. In *Proceedings of ACM Sigcomm 2002. Pittsburgh, PA*, 2002.

[9] M. Beck, T. Moore, J. Plank, and M. Swany. Logistical Networking: Sharing More Than the Wires. In *Proc. of 2nd Annual Workshop on Active Middleware Services*, August 2000.

[10] S. Birrer, D. Lu, F. E. Bustamante, Y. Qiao, and P. Dinda. FatNemo: Building a Resilient Multi-Source Multicast Fat-Tree. In *Workshop on Web Content Caching and Distribution*, October 2004.

[11] R. Castro, M. Coates, M. Gadhiok, R. King, R. Nowak, E. Rombokas, and Y. Tsang. Maximum likelihood network topology identification from edge-based unicast measurements. In *ACM SIGMETRICS*, June 2002.

[12] Y. Chu, S. G. Rao, and H. Zhang. A Case for End System Multicast. In *ACM SIGMETRICS. Santa Clara, CA*, June 2000.

[13] E. W. Dijkstra. A note on two problems in connexion with graphs. Numerische Mathematik, 1, 1959, pp.269-271.

[14] IPerf. http://dast.nlanr.net/Projects/Iperf/.

[15] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O'Toole, Jr. Overcast: Reliable Multicasting with an Overlay Network. In *USENIX Operating Systems Design and Implementation. San Diego, CA*, October 2000.

[16] S. J. Lee, P. Sharma, S. Banerjee, S. Basu, and R. Fonseca. Measuring Bandwidth between PlanetLab nodes. In *Passive and Active Measurements. Boston, MA*, 2005.

[17] Planetlab Iperf Data. http://jabber.services.planet-lab.org/php/iperf.

[18] PlanetLab Overview. http://www.planet-lab.org/consortium/overview.php.

[19] J. S. Plank, A. Bassi, M. Beck, T. Moore, D. M. Swany, and R. Wolski. Managing Data Storage in the Network. *IEEE Internet Computing*, 5(5):50–58, September/October 2001.

[20] M. Swany. Improving Throughput for Grid Applications with Network Logistics. In *Supercomputing 2004*, November 2004.

[21] M. Swany and R. Wolski. Network Scheduling for Computational Grid Environments. In *Adaptive Grid Middleware '03 (in association with PACT)*, September 2003.

[22] K. Verstoep, K. Langendoen, and H. E. Bal. Efficient Reliable Multicast on Myrinet. In *Int. Conference on Parallel Processing. Bloomingdale, IL, Vol. 3*, pages 156–165, 1996.

[23] R. Wolski, N. Spring, and J. Hayes. The network weather service: A distributed resource performance forecasting service for metacomputing. *Future Generation Computer Systems*, 1999.

[24] B. Zhang, S. Jamin, and L. Zhang. Host multicast: A framework for delivering multicast to end users. In *IEEE Infocom. New York, NY*, June 2002.

[25] Y. Zhang, N. Du, E. Paxson, and S. Shenker. The Constancy of Internet Path Properties. In *ACM SIGCOMM Internet Measurement Workshop*, November 2001.

[26] Q. Zhu, M. Parsa, and J. J. Garcia-Luna-Aceves. A Source-Based Algorithm for Delay-Constrained Minimum-Cost Multicasting. In *IEEE INFOCOM*, pages 377–385, 1995.