

The Performance of a Service for Network-Aware Applications

Katia Obraczka and Grig Gheorghiu

USC Information Sciences Institute

University of Southern California

4676 Admiralty Way suite 1001

Marina del Rey, CA 90292

katia, grig@isi.edu

(310)822-1511 (voice) (310)823-6714 (fax)

Abstract

This paper evaluates the performance of *topology-d*, a service that estimates the state of networked resources by periodically computing the end-to-end latency and available bandwidth among them. Using its delay and bandwidth estimates, topology-d computes a fault tolerant, minimum-cost spanning tree connecting participating sites.

We deployed topology-d on 27 Internet sites throughout the world and collected data for a period of two and a half months. The results of these wide-area experiments show that topology-d's estimates compare quite well with latency and bandwidth measurements from existing tools. We observe that the logical topologies computed by topology-d are consistent with current latency and bandwidth estimates. Topologies are also responsive to changes in network and server load, as well as in group membership.

Robustness and fault-tolerance distinguish topology-d from other measurement services and prove to be invaluable in a distributed, administratively decentralized environment like the Internet. Participating sites maintain a weakly-consistent view of the group which provides the basic infrastructure to automatically handle group membership dynamics, including failure and recovery.

1 Introduction

The growth of the Internet motivated the development of a variety of distributed applications, such as information dissemination, multimedia, and metacomputing services. These applications need to be *network-aware* to ensure they get appropriate service from the underlying communication and computing infrastructure. For example, a client's request to a replicated information service should be automatically directed to a close-by, lightly loaded server. Being able to choose adequate servers requires that applications have access to information about network and

server load. We argue that a service that provides knowledge about the dynamics of the underlying communication and computing infrastructure is essential to *network-aware* applications.

This paper evaluates the performance of *topology-d*, a service that estimates the state of networked resources by periodically computing the end-to-end latency and available bandwidth among them. Since these are end-to-end measurements, they capture both network and server load. Based on these estimates, topology-d computes a low-delay, high-bandwidth spanning tree with additional edges for fault-tolerance. Topologies are periodically re-computed to take into account network and server load dynamics.

Topology-d evolved from the Harvest [1] replication service [9], whose goal was to minimize the network bandwidth and delay required to replicate broker archives used in Harvest. Motivated by the need of other network-aware applications, we have implemented the replicator's network estimation and topology computation tool as a stand-alone service we call *topology-d*. This paper presents the results of the experimental performance evaluation study we conducted by deploying topology-d on 27 hosts across the Internet. The goal was to validate the tool's estimates and logical topologies. These experiments also allowed us to showcase our tool's robustness and fault-tolerance, which are essential in a distributed, administratively decentralized environment like the Internet. Through topology-d's membership protocol, sites can dynamically leave (voluntarily or due to failures) and join a group without disrupting the group's normal operation.

Topology-d's estimates can be used in different ways. They can be logged for future assessments of the network status. Alternatively, topology-d's data can be either directly consumed by applications or it can be periodically published in a directory service for consultation by interested third-parties. The latter approach is the one adopted in the Globus metacomputing infrastructure [4]. We give more details about topology-d's integration into Globus in Section 2. Other potential clients of our tool include replicated databases and distributed information dissemination services, such as the World Wide Web and Internet archives. We anticipate that other applications can also benefit from topology-d's services. Topology-d's logical topologies can be tailored to fit the requirements of different distributed applications. Currently, topology-d

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SPDT 98 Welches OR USA

Copyright ACM 1998 1-58113-001-5/98/ 8...\$5.00

generates fault-tolerant topologies that try to minimize both update propagation cost and time in replicated information services. Logical topologies for metacomputing systems can simply connect participating sites with a pre-specified number of close-by, well-connected, lightly-loaded computation resources.

2 Related Work

2.1 Internet Measurement

Internet measurement is the subject of the IP Provider Metrics (IPPM) subgroup of the IETF's Bench Marking Working Group (BMWG). IPPM's main goal is to provide and standardize metrics and methodologies for Internet performance measurement [10]. The Cooperative Association for Internet Data Analysis (CAIDA)'s taxonomy surveys measurement tools currently available both as free and commercial software.

TReno [8] is one tool being considered for IPPM-endorsement. It aims at accurately measuring the bulk transfer capacity of network links by implementing its own TCP algorithm with Selective Acknowledgements (SACK). TReno measures the throughput of a given link independent of the particular TCP implementation on the end hosts. Topology-d and netperf, on the other hand, measure the throughput of an end host's particular TCP implementation since this is the performance perceived by applications running on that host.

Pathchar [6] is a recently released tool which estimates bandwidth, delay, average queue and loss rate of every hop between a given source-destination pair of Internet hosts. Pathchar uses the ICMP protocol's Time Exceeded response to packets whose TTL has expired.

bprobe and cprobe [2] probe the network by sending several pairs (bprobe) or a short train of packets (cprobe). They estimate network bandwidth by measuring packet inter-arrival time. The goal of bprobe is to measure the bottleneck bandwidth along a network path, whereas cprobe attempts to estimate the effective bandwidth available to an application in the presence of competing network traffic. One advantage of these tools is that they do not load the network with their own traffic (although bprobe does have a tendency to send packets of sizes up to 8000 bytes for very short periods of time). Unfortunately, at the time we conducted our experiments, the tools were only available for SGI machines and were hard to port to other platforms since they depend on SGI's accurate timer mechanisms.

While topology-d provides its own network bandwidth and latency estimates, it could just as well use the estimates from an IPPM-approved tool (which for the moment does not exist). We envision our tool as a "middleware" service that provides information about the underlying communication and computing infrastructure. In fact, topology-d could use its group management infrastructure to provide estimates gathered from a variety of network measurement tools to applications which depend on this kind of knowledge for adequate performance. Before looking at some examples of such applications, we discuss two other existing middleware services similar in certain aspects to our tool.

The Network Weather Service (NWS) [14] forecasts the end-to-end throughput and latency of TCP/IP-based ap-

plications. NWS collects data using netperf and then applies a set of forecasting methods, including mean-based, median-based, and autoregressive models. NWS then dynamically selects the "best" forecast according to a specified metric, which can be based on measuring either the mean square prediction error or the mean percentage prediction error. One drawback of NWS is that it does not adjust to failures and only functions well as long as all monitored machines stay up and running. Unlike NWS, topology-d was designed to adjust to group membership dynamics.

There is an increasing interest in measuring performance aspects of the Internet specifically related to Web-browsing. For example, Lachesis [12] was designed to assess the performance of Internet Service Providers (ISPs). It benchmarks a particular ISP by taking a list of so-called "landmarks" (important sites across the Internet, including DNS root servers, well-known FTP servers and popular WWW servers) and measuring the packet loss and network latency to each landmark.

2.2 Applications

One current client of our tool is the resource broker used in the Globus metacomputing infrastructure [4]. Globus aims at enabling the deployment of high-performance, computing-intensive distributed applications by taking advantage of resources such as high-speed networks and supercomputers. The Globus Metacomputing Directory Service (MDS) [3] provides information about the current state of available resources. This information is used by the Resource Broker when scheduling resources for specific tasks. Topology-d's robustness and distributed group membership protocol made it an ideal candidate for the job. Topology-d is now deployed at all the sites participating in the Globus project and it provides its network measurements to the MDS. We are currently looking at ways to customize the logical topology algorithm to make it more suitable for Globus.

Smart clients [15], which is part of the Berkeley NOW (Network Of Workstations) project, is another application that can benefit from topology-d's services. It was developed as a collection of Java applets and take a client-side approach to providing transparent access to multi-server Internet services such as HTTP and FTP. The *director* applet chooses the "best" machine from a pool of servers providing a particular service requested by the client. The choice is transparent to the user and is based on the kind of network state information topology-d provides.

3 Topology-d

For performance and robustness, topology-d was implemented as a Unix daemon that does not fork, but instead uses non-blocking I/O for all communication. The daemon can be queried in two ways: from a Web browser or via a command-line utility called *td-client*. The HTTP interface is better suited for interactive sessions: it displays both current estimates and the group's logical topology. The command-line interface is useful for "batch-mode" queries. We wrote a series of Perl scripts as wrappers around this utility and we used these scripts for statistics gathering and parsing.

3.1 Estimate Collection and Topology Generation

Each machine running topology-d periodically computes available bandwidth and RTT between itself and all the other machines in the group. For RTT estimates, a timestamped UDP packet is sent to another member of the group, which simply returns it back to the originator. For bandwidth estimates, each machine sends a block of data using TCP (the default for the block size is 32KB). Available bandwidth is then computed as:

$$\text{bandwidth} = \text{bytes_sent} / (\text{time}_{\text{last_byte}} - \text{time}_{\text{first_byte}})$$

where the two timestamps are taken by the destination machine. When computing the actual estimates to be reported to the master, previous history is taken into account in order to avoid transient changes. This “damping” effect is computed using the following exponential average function¹:

$$\text{new_est} = \alpha * \text{old_est} + (1 - \alpha) * \text{current_est}$$

We currently set α to 0.5. It is important to note that topology-d’s bandwidth and RTT estimates take into account machine load. While a tool like TReno is more accurate for transport-layer measurements, topology-d tries to measure the actual delay and bandwidth seen by an application.

A group member designated as the *master* collects the estimates reported by group members into a cost matrix for the group which the master uses to compute the group’s logical topology. Each entry $C_{i,j}$ in the cost matrix corresponds to the communication cost between machines i and j and is given by $B_{i,j}/D_{i,j}$, where $B_{i,j}$ and $D_{i,j}$ are the estimated bandwidth and RTT between nodes i and j , respectively. Topology-d then generates the logical topology by invoking a topology generator program, which uses as input the cost matrix and a connectivity requirement k for each node (currently we have $k = 2$). The topology generator first computes a minimum cost spanning tree connecting all the nodes, and then, for each node whose degree d is less than the required connectivity k , adds the current cheapest edge until $d = k$. The master periodically sends the current logical topology to all the machines in the group.

3.2 Group membership

When a site joins a group, it sends a join request to the master, which adds it to the list of known sites. However, the machine is not “officially” part of the group until the master distributes a new topology that contains the site.

There is no protocol for leaving the group. Machines leave a group silently and if the master has not heard from a member after a pre-determined period of time, it drops the machine from the group membership. The silence period is configurable and is currently set to one hour.

Topology-d was designed to be fault-tolerant with respect to group membership changes. A group continues to function normally as machines leave or join. Even if the master is temporarily disconnected from the rest of the group, the members will continue to compute the estimates, without receiving topology and group membership updates. This weakly consistent membership protocol al-

lows topology-d to be robust to failures of group members, including the master.

4 Experiments

We installed topology-d at 27 Internet sites including 17 locations in the US, 6 in Europe, 1 in South America, 3 in Asia and the South Pacific. We configured all participating sites as members of a single topology-d group, whose master was `excalibur.usc.edu`.

Conducting such a large scale, widely distributed experiment required considerable effort. Topology-d’s robustness proved to be invaluable in such an autonomously decentralized environment since it was quite common that a machine got rebooted or crashed during the experiments. Other than missing data for the unavailable sites, the experiments proceeded normally. These temporary site failures demonstrated topology-d’s ability to adjust to membership changes caused by a site voluntarily leaving the group and joining in later, and by temporary failures.

Table 1 describes the configurable group parameters and shows their values for the experiments. The `master` site parameter points to the master of the group. The `ping period` and `bandwidth period` parameters determine how often sites estimate latency and bandwidth. The `update period` and `estimates period` determine how often the logical topology is updated and how frequent sites report their estimates to the group.

4.1 Data Collection

We sampled topology-d’s bandwidth, latency and topology information every hour over a period of two and a half months, starting beginning of June 1997 and ending in mid September. To validate topology-d’s estimates, we compared them against measurements obtained from running `ping` and `netperf` on a subset of the machines in the group. In order to restrict the N^2 network paths we had available, we ran `ping` and `netperf` measurements on four sites chosen based on their geographic location, how well they were connected, and how stable (in terms of uptime) they were. We limited the data collection period to one week; during these seven days, we ran `ping` and `netperf` every hour while running topology-d.

`Netperf` [7] estimates the available network throughput by sending packets of configurable length over a configurable period of time (10 seconds by default). This bandwidth measuring technique yields more accurate estimates than topology-d. On high bandwidth-delay paths, topology-d’s single 32 Kbyte packet may not be enough to stretch the TCP transmission window beyond slow-start. This is likely the main reason why `netperf` reports consistently higher throughput values than topology-d. Clearly, there is a tradeoff between estimate accuracy and overhead. One issue for future work is to dynamically set the size of the bandwidth probe according to the path being measured.

We also ran `traceroute` at three of the data collection sites: `redondo.ece.uci.edu`, `cosmo.mcs.anl.gov`, and `mirage.irdi.nus.sg`. `Traceroute` information is useful in diagnosing transient problems as well as providing insight to validate the topologies generated by topology-d.

¹This well-known damping mechanism is used in TCP’s congestion control round-trip time computation [5]

Parameter	Value	Description
master-site	excalibur.usc.edu	points to group master
ping-period	15min	latency estimation period
bandwidth-period	60min	bandwidth estimation period
update-period	60min	topology update period
estimates-period	30min	estimate report period

Table 1: Group parameters and their values.

4.2 Data Analysis

It has been shown [13] that the data population obtained by doing network measurements over the Internet does not present a normal or Gaussian distribution. Hence, the mean and standard deviation are not meaningful statistics for Internet measurements. Following the recommendations of the IPPM community, we then chose to present the 10th and 90th percentiles together with the median and the interquartile range (IQR, i.e. the 75th percentile minus the 25th percentile) to summarize our measurements.

5 Results

We used ping’s and netperf’s measurements of latency and bandwidth to validate topology-d’s RTT and bandwidth estimates, respectively. Once we validate topology-d’s estimates of network and server load, we use them to validate the logical topologies generated.

5.1 Latency and Bandwidth

Our results are presented in the form of graphs showing topology-d’s latency (in milliseconds) and bandwidth (in kilobits/sec) estimate samples taken between a given pair of hosts over time (in hours). Latency graphs also show the corresponding ping measurements, while bandwidth graphs present netperf data.

Figures 1 and 2 show latency and bandwidth measurements taken from redondo.ece.uci.edu to the three other data collection sites. We use redondo.ece.uci.edu’s estimates as example; measurements taken from the other data collection sites present similar behavior. The statistics in Figure 3 summarize measurements from all data collection machines.

Figure 1(a) shows that topology-d’s and ping’s RTT measurements to cosmo.mcs.anl.gov are quite similar. Topology-d’s ability to take previous history into account dampens the effects of transients, which were captured by ping. For farther hosts connected through bottleneck connections, such as trans-oceanic links, both topology-d and ping report higher variability in RTT. This is the case for itia.math.uch.gr and mirage.irdu.nus.sg (Figures 1(b) and (c)). We observe higher RTT variability in the case of the link to mirage.irdu.nus.sg. To confirm this variability, we inspected the traceroute logs we collected and we found indeed that there were numerous problems with this network path and multiple cases of unreachability. Generally, topology-d’s and ping’s measurements compare quite well. This similarity is corroborated by the statistics presented in Figure 3(a).

Topology-d’s and netperf’s bandwidth measurements taken from redondo.ece.uci.edu are presented in Fig-

ure 2(b) and summarized by the statistics presented in Figure 3. They show that although the magnitudes of the measurements taken by each tool differ considerably (as explained in Section 4), the shapes of the curves are similar. This means that both topology-d and netperf capture similar trends in bandwidth variability over time, although topology-d’s use of damping prevents it from capturing transients. Again, one can notice the very high variability for the trans-oceanic link to mirage.irdu.nus.sg as opposed to the relative stability of the trans-continental link to cosmo.mcs.anl.gov.

We also notice from the graphs that there is no clear pattern in network and server usage over time, with the possible exception of the bandwidth graph from redondo.ece.uci.edu to mirage.irdu.nus.sg. This graph presents a certain periodicity, with peaks and slopes roughly every 20 hours². This pattern was consistent across several sets of bandwidth measurements that we took between these two sites and it can also be observed on the reverse path, from mirage.irdu.nus.sg to redondo.ece.uci.edu. However, the pattern is not apparent for the link between cosmo.mcs.anl.gov and mirage.irdu.nus.sg, which prompts us to speculate that this is an isolated characteristic of the inter-continental link between the US and Asia. For all the other graphs, “peak” and “off-peak” times are indistinguishable. This is due to the fact that our experiments included sites that span various time zones and thus the results show the current trends in network and server usage which are driven by globally distributed applications such as the World Wide Web.

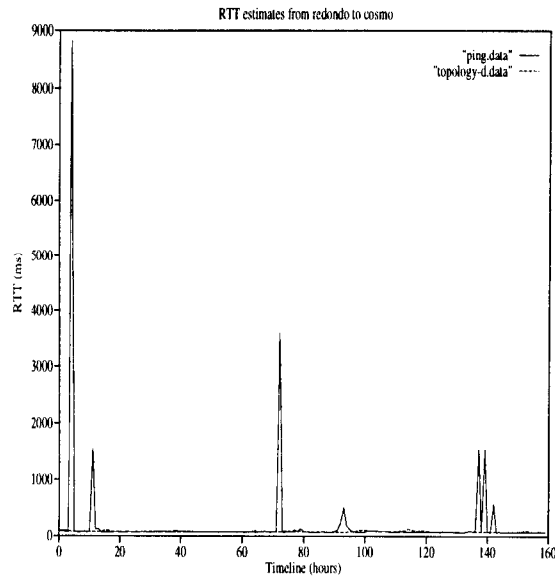
One other observation concerns the asymmetry of network paths. The fact that network links are generally asymmetric has been stated in [11] and is confirmed by our statistics.

Discussion

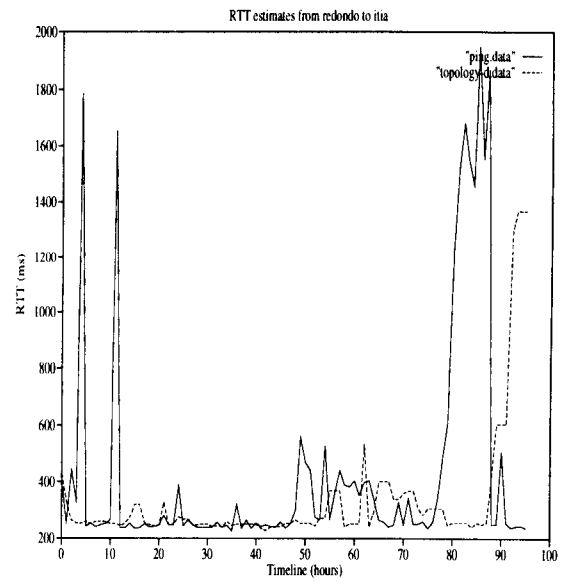
Topology-d was not designed to provide accurate network bandwidth and delay measurements, but an estimate of the state of the network and its resources. One of the goals of our experiments was to demonstrate topology-d’s ability to capture network dynamics so that client applications can adjust to changes in network and server load.

One issue for future work is to investigate other measurement tools, such as the ones mentioned in Section 2. Experimenting with other tools that use alternate ways of measuring network bandwidth and latency will allow us to identify topology-d’s strengths and weaknesses regarding network state estimation. It will also help us identify tools that could be incorporated into topology-d. This

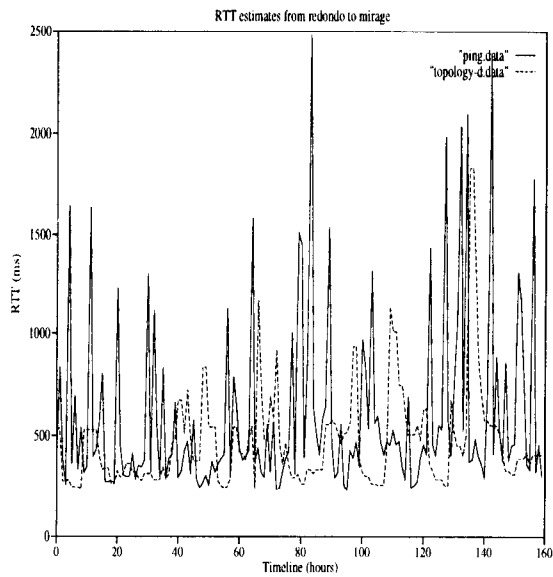
²This periodicity is more noticeable for netperf’s measurements than for topology-d’s due to the latter’s use of damping.



(a) RTT to `cosmo.mcs.anl.gov`

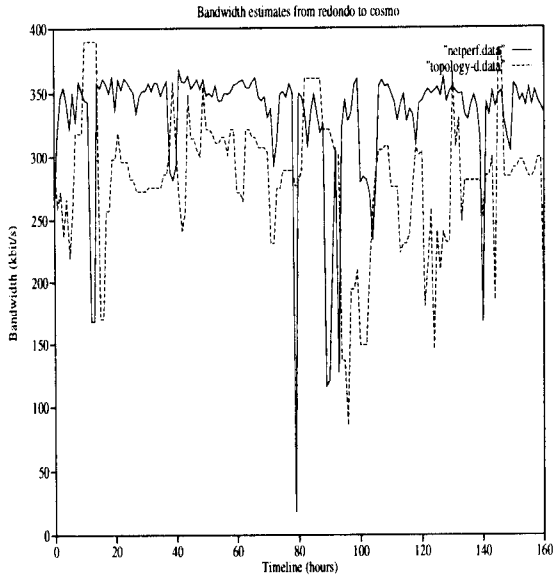


(b) RTT to `itia.mcs.anl.gov`

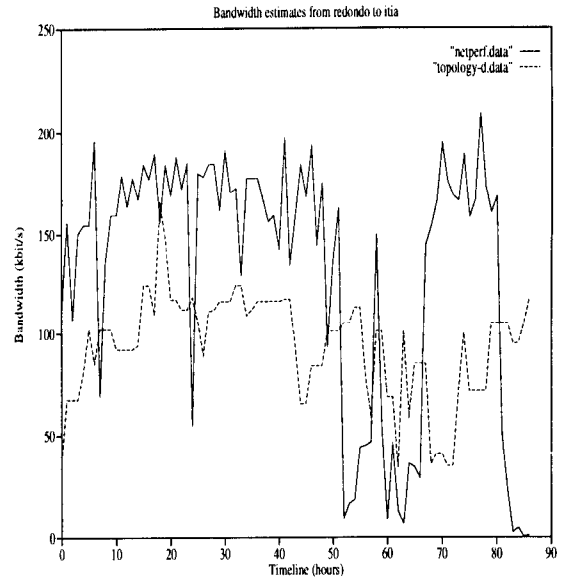


(c) RTT to `mirage.irdu.nus.sg`

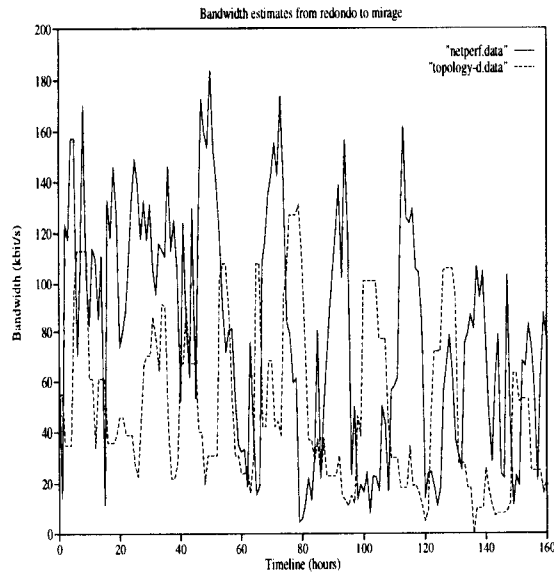
Figure 1: Latency (RTT in ms) measurements taken from `redondo.ece.uci.edu` over time (in hours). Dotted lines show topology-d estimates, while continuous lines show ping measurements.



(a) BW to `cosmo.mcs.anl.gov`



(b) BW to `itia.mcs.anl.gov`



(c) BW to `mirage.irdu.nus.sg`

Figure 2: Bandwidth (BW in kbits/sec) measurements taken from `redondo.ece.uci.edu` over time (in hours). Dotted lines show topology-d estimates, while continuous lines show netperf measurements.

Sites	cosmo.mcs.anl.gov	itia.math.uch.gr	mirage.irdu.nus.sg	redondo.ece.uci.edu
cosmo.mcs.anl.gov	N/A N/A	0.00/238(78.5)/392 200/221(110)/576	333/443(209)/892 337/436(151)/935	72.0/75.0(6.00)/98.0 70.0/74.0(7.00)/91.4
itia.math.uch.gr	0.00/236(87.0)/356 203/216(66.0)/522	N/A N/A	0.00/565(198)/862 487/592(213)/1,470	0.00/248(55.0)/401 233/250(40.0)/411
mirage.irdu.nus.sg	320/419(101)/678 340/463(219)/801	0.00/652(356)/1,110 488/678(326)/1,280	N/A N/A	251/399(195)/771 275/424(291)/834
redondo.ece.uci.edu	71/78(10)/92 70/74(8)/98	0.00/253(63.2)/410 237/256(145)/921	251/391(249)/708 289/418(284)/1,250	N/A N/A

(a) Latency (in milliseconds).

Sites	cosmo.mcs.anl.gov	itia.math.uch.gr	mirage.irdu.nus.sg	redondo.ece.uci.edu
cosmo.mcs.anl.gov	N/A N/A	0.00/53.0(65.5)/97.0 0.00/96.1(122)/140	10.0/28.0(24.0)/52.0 14.0/42.0(39.0)/79.0	163/260(66.5)/299 224/381(82.9)/407
itia.math.uch.gr	0.00/50.0(76.5)/94.0 0.00/63.1(47.2)/81.3	N/A N/A	0.00/24.0(40.0)/52.6 5.79/67.2(53.6)/86.2	0.00/95.0(74.0)/124 40.3/169(49.3)/187
mirage.irdu.nus.sg	12.0/30.0(26.0)/60.0 6.24/16.4(21.9)/68.6	0.00/31.0(28.5)/65.0 750e-3/15.0(34.2)/78.9	N/A N/A	10.6/42.0(49.0)/91.0 4.87/13.3(22.3)/123
redondo.ece.uci.edu	117/284(72.5)/321 288/348(20.3)/359	0.00/92.0(48.5)/117 9.34/159(123)/186	8.00/36.0(49.0)/103 18.0/80.1(82.4)/142	N/A N/A

(b) Bandwidth (in kbits/sec).

Figure 3: Statistics summarizing latency and bandwidth measurements. Rows and columns list source and destination machines. Each cell shows the 10th percentile/median(IQR)/90th percentile for our sample population. The numbers on the top line refer to topology-d’s estimates, while the numbers on the bottom line refer to ping estimates in (a) and to netperf estimates in (b).

will provide applications with a range of network measurement tools available under the same measurement infrastructure.

Topology-d employs an *active* estimation strategy³ since it generates overhead traffic when performing estimates. One way of making active measurement tools less invasive is to keep the measurement frequency low. There is a clear tradeoff between the overhead generated by active measurement tools, their accuracy and ability to detect changes (and avoid transients) in the underlying infrastructure. Topology-d’s estimation frequency is a configurable group parameter that can be set by system administrators to satisfy the needs of specific applications, yet keep the overhead traffic to a minimum given the characteristics of the underlying administrative domain’s resources.

As the number of participating sites grow, organizing them into hierarchical clusters help limit the amount of estimation overhead generated. Note that a n -member cluster performs $O(n^2)$ estimates. Therefore, small clusters generate less estimation traffic than larger ones. A designated cluster member can be responsible for performing inter-cluster estimation. In [9] we report the overhead generated by topology-d for different group sizes.

5.2 Topologies

For our experiments, we configured topology-d to re-compute the group’s topology every hour and sampled the group’s topology every hour.

³This classification is according to Vern Paxson’s framework [10].

We evaluated the topologies generated by topology-d using two metrics: (1) correlation between bandwidth and RTT estimates and the observed topologies, and (2) topology’s adaptability to the dynamics of the underlying network and computing infrastructure, including group membership changes.

5.2.1 Estimates and Logical Topologies

We verify that the logical topologies the master computes for the group reflect indeed the estimates collected by the group members. For each host, we calculate the median for the RTT and bandwidth estimates taken from that machine to all the other group members. We order the resulting list in decreasing order based on the ratio *bandwidth/RTT* which represents the cost of a given link. Also for each host we count how many times any given member of the group was seen among the host’s best neighbors and we order the resulting list in decreasing order. Tables 2 and 3 show the top machines in these two lists for redondo.ece.uci.edu over the week we collected topology-d, ping and netperf data. As expected, there is a strong correlation between the two lists, with the same machines dominating both lists. This close correlation between logical topologies and latency and bandwidth estimates is observed for all the other machines in the group.

We notice that the order of the five best machines according to redondo’s estimates is the same as the order in which they occur in the list of best neighbors, with one exception: *syc.isi.edu* appears more times in the list of neighbors than *excalibur.usc.edu*. It can also be ob-

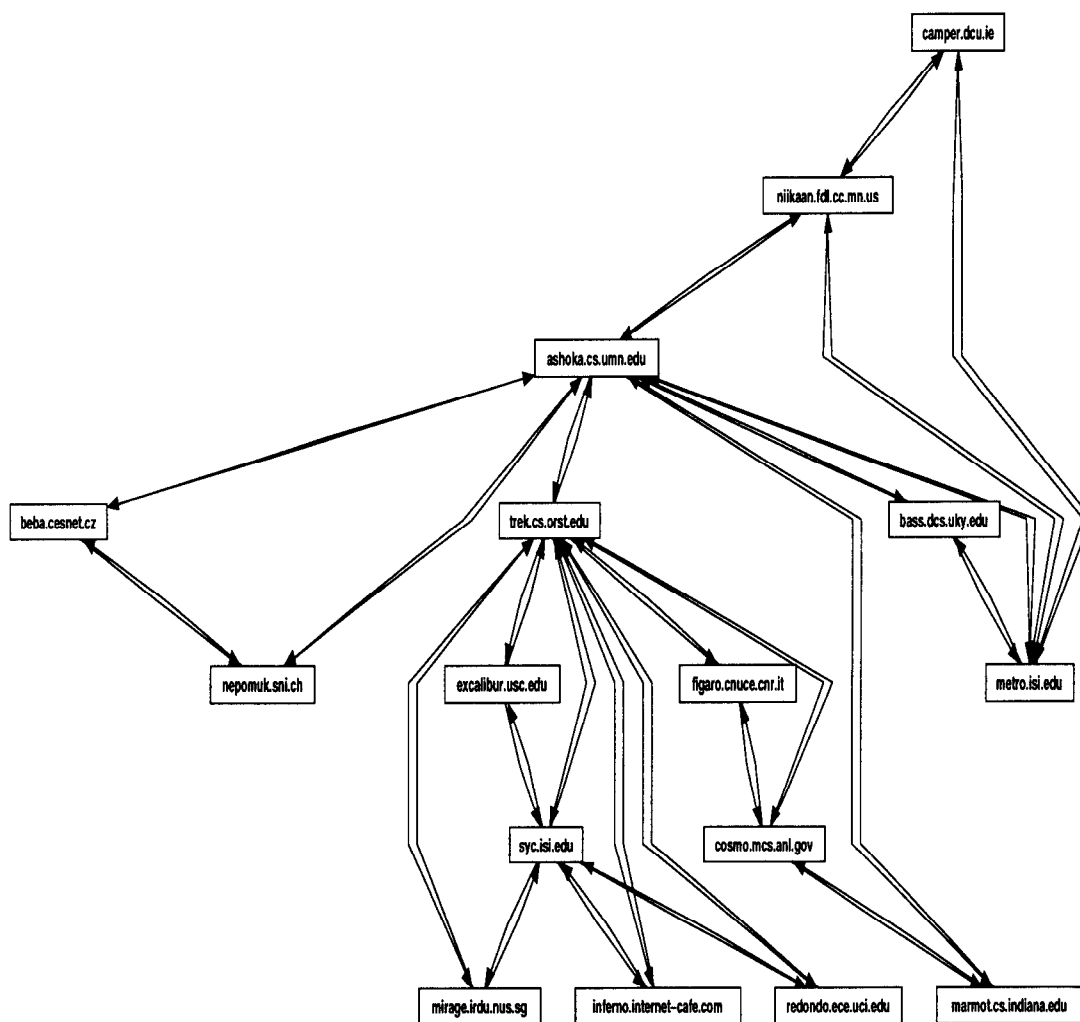


Figure 4: Adaptability to changes in group membership. Topology snapshot taken on Sept. 9 1997 at 5 PM Pacific Time.

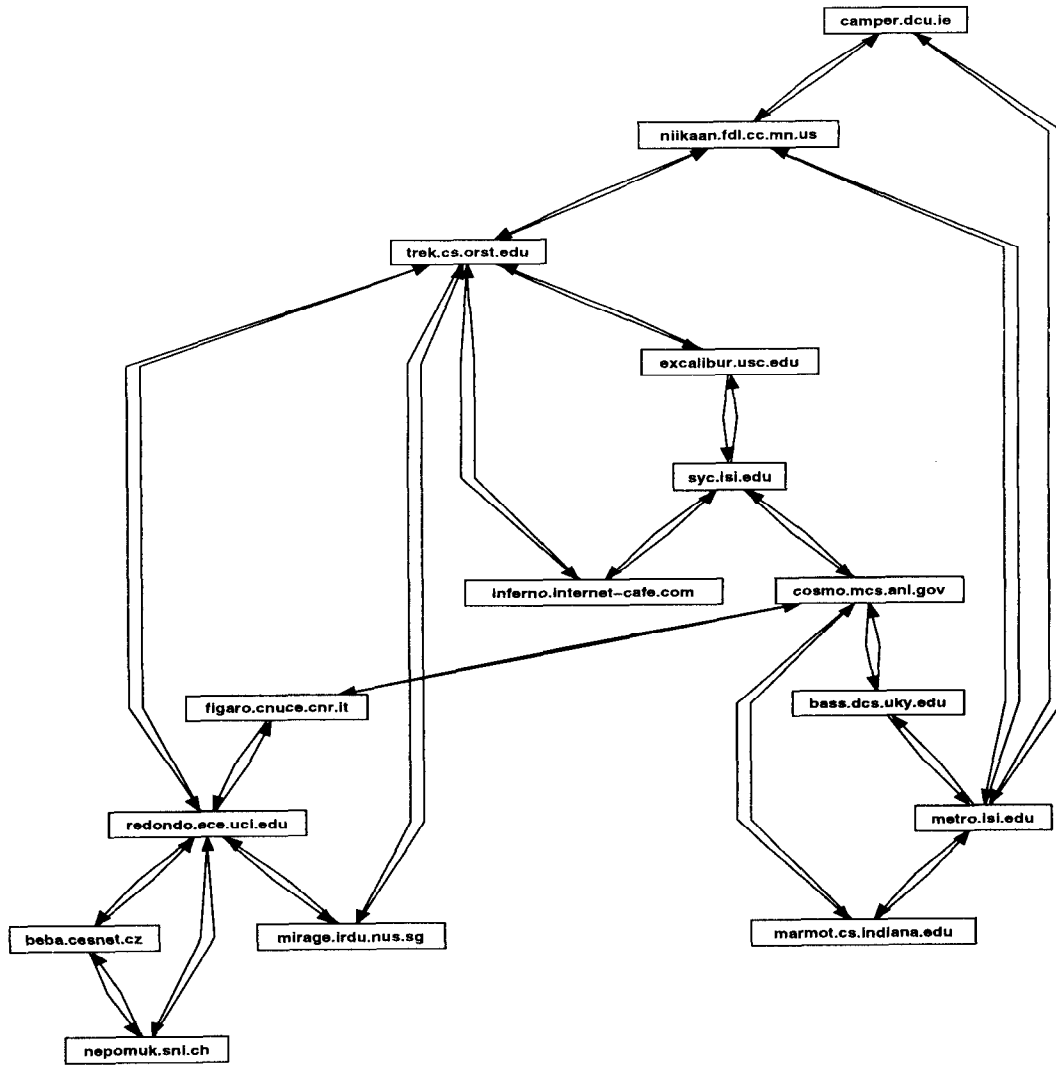


Figure 5: Adaptability to changes in group membership. Topology snapshot taken on Sept. 9 1997 at 6 PM Pacific Time.

Site	Median RTT	Median BW	BW/RTT
excalibur.usc.edu	28 ms	507 Kb/s	18
syc.isi.edu	30 ms	506 Kb/s	16
trek.cs.orst.edu	29 ms	488 Kb/s	16
ashoka.cs.umn.edu	69 ms	499 Kb/s	7
niikaan.fdl.cc.mn.us	93 ms	333 Kb/s	3
pharos.bu.edu	94 ms	332 Kb/s	3

Table 2: Estimates for `redondo.ece.uci.edu`

Site	Number of occurrences
syc.isi.edu	113
inferno.internet-cafe.com	107
excalibur.usc.edu	94
trek.cs.orst.edu	81
ashoka.cs.umn.edu	72
mirage.irdn.nus.sg	68
niikaan.fdl.cc.mn.us	31
pharos.bu.edu	28

Table 3: Topology neighbors for `redondo.ece.uci.edu`

served that two other machines (`inferno.internet-cafe.com` and `mirage.irdn.nus.sg`) appear in the top of the list of neighbors, although they are not among the top machines according to the estimates. This is due to the fact that the current topology computation algorithm does not account for link asymmetry. So if `inferno.internet-cafe.com` and `mirage.irdn.nus.sg` see `redondo.ece.uci.edu` among their best neighbors, then they will appear in the list of `redondo`'s neighbors. This sometimes can lead to situations where the topology resembles a star having in its center a powerful and/or well connected machine (refer to Section 5.2.2 for an example). We plan to modify the topology computation algorithm to take into account asymmetric links. We will also specify a maximum node degree.

5.2.2 Adaptability

Figure 4 shows the “star” effect mentioned in the previous section. Host `ashoka.cs.umn.edu` is connected to six other hosts. Note that the degree parameter the topology computation algorithm uses specifies the minimum node degree. We plan to include a maximum node degree to limit the number of neighbors assigned to a host and therefore eliminate star topologies.

One aspect of topology adaptability concerns changes in group membership. Figures 4 and 5 show two snapshots taken one hour apart during our experiments. We observe a very clear transition. When the host `ashoka.cs.umn.edu` went down, the group reconfigured itself quickly. This topology change illustrates topology-d's fault-tolerance to (voluntary or involuntary) membership changes, which is extremely valuable in a distributed and unpredictable

We also observe that topologies are responsive to changes in latency and bandwidth estimates. This statement is supported by the same type of evidence we showed for adaptability to group membership changes. Whenever the “cost” of a link (i.e. the ratio $bandwidth/RTT$) from host A to host B changes significantly enough so that host B is not seen as a neighbor anymore by host A, the new recom-

puted topology reflects this change. In general, topology snapshots tend to be very similar except for a few links. This is a trend we observed for the entire experiment. It is very uncommon to find two identical topologies, but changes from one topology to the next are very small and generally limited to two or three machines.

6 Conclusions

This paper reported the results of a study we conducted to evaluate the performance of topology-d, a service that estimates the state of networked resources. Based on these estimates, topology-d computes a fault tolerant, minimum-cost spanning tree connecting participating sites. We validated topology-d's estimates by comparing them with the ones obtained from other well-known tools such as `netperf` and `ping`. We showed that topology-d's logical topologies were consistent with its estimates of the current network state. We also showed that the topologies adjust to changes in network and server load, as well as in group membership.

The Internet-wide experiments proved that topology-d's fault-tolerance and robustness are extremely important in the constant presence of reboots and shutdowns; these are essential features in wide-area, administrative decentralized environments.

References

- [1] C.M. Bowman, P.B. Danzig, D.R. Hardy, U. Manber, and M.F. Schwartz. Harvest: A scalable, customizable discovery and access system. Technical Report CU-CS 732-94, Department of Computer Science, University of Colorado-Boulder, July 1994.
- [2] R. Carter and M. Crovella. Dynamic server selection using bandwidth probing in wide-area networks. Boston University Computer Science Dept. TR-96-007, March 1996.

- [3] S. Fitzgerald et al. A directory service for configuring high-performance distributed computations. *Proceedings of the 6th IEEE International Symposium on High Performance Distributed Computing*, August 1997.
- [4] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *International Journal of Supercomputer Applications*, Summer 1997. Available at Globus web page <http://www.globus.org/>.
- [5] V. Jacobson. Congestion avoidance and control. *ACM SIGCOMM 88*, pages 273–288, 1988.
- [6] V. Jacobson. Pathchar: A tool to infer characteristics of internet paths. Available from <ftp://ftp.ee.lbl.gov/pathchar/>, April 1997.
- [7] R. Jones. netperf. Available from <http://www.cup.hp.com/netperf/NetperfPage.html>.
- [8] M. Mathis and J. Madhavi. Diagnosing internet congestion with a transport layer performance tool. *Proceedings of the INET 1996*, 1996.
- [9] K. Obraczka, P.B. Danzig, D. DeLucia, and E. Tsai. A tool for massively replicating internet archives: Design, implementation, and experience. *Proceedings of the 16th. IEEE ICDCS*, pages 657–664, May 1996.
- [10] V. Paxson. Towards a framework for defining Internet performance metrics. *Proceedings of the INET 1996*, 1996.
- [11] V. Paxson. End-to-end routing behavior in the Internet. *Proceedings of the ACM Sigcomm 1996*, pages 25–38, August 1996.
- [12] J. Sedayao and K. Akita. Lachesis: A tool for benchmarking internet service providers. *Proceedings of the LISA IX Conference*, 1995.
- [13] J. Madhavi V. Paxson, G. Almes and M. Mathis. Framework for IP Performance Metrics. Internet Draft available at <ftp://ftp.ietf.org/internet-drafts/draft-ietf-ippm-framework-02.txt>, expiration date: July 1998.
- [14] R. Wolski. Forecasting network performance to support dynamic scheduling using the network weather service. *Proceedings of the 6th IEEE International Symposium on High Performance Distributed Computing*, August 1997.
- [15] C. Yoshikawa, B. Chun, P. Eastham, A. Vahdat, T. Anderson, and D. Culler. Using smart clients to build scalable services. *Proceedings of the USENIX 1997 Technical Conference*, 1997.