IPwatch: A Tool for Monitoring Network Locality

Mark J. Lorence
Information Technology Center
Carnegie Mellon University

M. Satyanarayanan
Department of Computer Science
Carnegie Mellon University

Abstract

In this paper we introduce the concepts of *Logical* and *Physical Network Locality* and point out their importance to the performance of distributed systems. We then describe the design of *IPwatch*, a simple and inexpensive tool for monitoring logical network locality. IPwatch exploits short-term locality to enable monitoring of medium- and long-term locality of large networks using modest computational resources. We describe experiments at Carnegie Mellon University to validate our ideas and to calibrate IPwatch. The results confirm the existence of substantial short-term locality in this environment. Less than 5 percent of the possible host pairs account for 75 percent of the traffic, and less than 15 percent of them account for 90 percent. Comparative measurements on another network in our environment show even stronger short-term locality.

This work was supported by the National Science Foundation (Contract No. CCR-8657907), Defense Advanced Research Projects Agency (Order No. 4976, Contract F33615-84-K-1520) and the IBM Corporation. The views and conclusions in this document are those of the authors and should not be interpreted as representing the official policies of the funding agencies or Carnegie Mellon University.

1. Introduction

In the course of the last decade local area networks (LANs) have grown from simple single-segment Ethernet cables into multi-segment topologies composed of diverse physical media. Today, a LAN in a large organization is typically composed of a shared *Backbone* to which a number of semi-independent *Subnets* are attached via devices called *Routers* that perform packet switching. Multiple backbones may be present for enhanced reliability and performance. A variety of factors account for this increase in complexity. First, electrical considerations limit the lengths of individual LAN segments and the density of machines on them. Second, maintenance and fault isolation are simplified if a LAN is decomposable. Third, administrative functions such as the assignment of unique host addresses can be decentralized if a LAN can be partitioned. These considerations will increase in importance as distributed systems become more pervasive.

During the same period of time, distributed systems have been evolved to to provide *Network Transparency* for a variety of services such as remote file access [13], electronic mail [12] and remote program execution [6]. Network transparency masks underlying network complexity. Neither users nor application programs need be concerned with the details of the network traffic they generate. Unfortunately, performance inhomogeneities cannot be hidden even when a network is rendered functionally homogeneous. Routers, which introduce load-dependent transmission delays, are the primary source of performance inhomogeneity. Uneven loading of subnets is a secondary cause. Ignoring the effects of network topology can result in suboptimal use of a network by a distributed system. Nonuniform performance is already observed by users of large distributed systems such as Andrew [5]. Further growth will exacerbate the problem.

This paper describes the design, implementation and calibration of a tool called *IPwatch* that will assist implementors in building and maintaining distributed systems that make optimal use of the underlying network. Section 2 examines the nature of locality in distributed systems and explains why it is relevant to the concerns expressed in the preceding paragraph. Section 3 outlines the considerations that influenced the design of the tool and describes its design. Section 4 presents results from a series of experiments conducted to validate the tool and calibrate it. Section 5 concludes the paper with a discussion of work in progress.

2. Network Locality

2.1. Definition

We define *Logical Network Locality* as the property whereby a small fraction of many possible host pairs account for most end-to-end network traffic. It is influenced by the behavior of users and application programs and by the design of distributed systems. *Physical Network Locality*, on the other hand, is defined as the property whereby most network traffic traverses as few routers as possible, preferably none. It is influenced by logical network locality, network topology, and the placement of hosts. Since physical locality minimizes network delays, an optimal mapping of a distributed system on to a network will exhibit the highest possible physical locality.

Our primary goal is to understand and measure logical network locality. We are convinced that such knowledge is critical for tuning a distributed system for optimal performance. Information about logical network locality can be used to improve physical network locality by modifications to one or more of the following areas:

1. Network topology
2. Placement of hosts on the network
3. Design of distributed system software.

For brevity we use the term *Locality* to mean logical network locality in the rest of this paper. Physical network locality will always be referred to by its full name.

2.2. Origin

The origin of locality can be best understood by an example. The Andrew system, which motivated this work, consists of over 500 workstations accessing files from a small collection of file servers. Workstations cache copies of most-recently-used files to improve performance and minimize workstation-server interactions. Most communication between a workstation and servers is to fetch files missing from the cache or to write back a file that has been modified. Files are fetched and stored in their entirety, using a specialized bulk transfer protocol. Since Andrew provides true location transparency, a user at a workstation can use any file stored in the shared file system, totally unaware of the specific servers that he is interacting with.

At the next level of detail, however, the architecture of Andrew substantially complicates an understanding of the resulting network traffic patterns. First, system files that are read often but rarely modified can have read-only replicas at multiple servers. It is indeterminate which one of these replicas get used by a workstation. Second, since the files of a user are typically located on one server, this server is likely to participate in most of the traffic with the user's workstation. However, when a user at a public workstation logs out and a new user starts work, most of the workstation references will be to the file server that stores the files of the new user. Even users who have workstations dedicated to them may localize their references to different servers as they proceed with different aspects of their computing activities. Third, when a file is modified, all workstations with currently valid cache copies are notified by the server responsible for that file. Although write-sharing is rare for private files, it is common for shared writable directories such as bulletin boards. The effect of this aspect of the design on locality is an open question. Finally, low-function machines such as the IBM-PC use the Andrew file system via surrogate workstations called *PCServers* [11]. PCServers are located on the same subnet as the PCs, but may access servers on other networks. Thus the logical simplicity of the Andrew file system at the user level does not carry over to the next level of detail. Although the existence of locality is undeniable, it is impossible to characterize this locality without observation and measurement.

Though we have used Andrew as an example, locality is not specific to it. Even in a network used only for terminal emulation and user-initiated file transfers, there will be locality arising from the fact that users deal with a small number of hosts at a time, typically one. In general, if we define a logical *Connection* as a source-destination pair, only a subset of possible connections will contribute significantly to traffic at any given time. How small this subset is, and how rapidly its membership changes are measures that characterize the locality of a network. A complete understanding of a distributed system requires data on its short, medium and long-term locality.

Locality in a network is analogous to locality of virtual memory references. If one views the concatenated source and destination addresses of a packet as defining an address space, only a small fraction of this address space is likely to be heavily referenced at any given time. But the analogy is imperfect. Virtual memory references exhibit both *Temporal* and *Spatial* locality. Networks do exhibit temporal locality, but not spatial locality. If a pair of machines communicate it is likely they will communicate again in the near future. However, there is no reason to believe that if two machines communicate, machines whose network addresses are close to theirs are also likely to communicate. Spatial locality is therefore not a meaningful concept here.

2.3. Measurement

Existing tools are inadequate for measuring locality. Simple tools [15] merely present network utilization, packet counts and byte counts. More sophisticated tools allow selective filtering of packets [2], but are not capable of dynamically modifying their behavior based on what connections seem "interesting" at any given time.

Early in our design of a tool to measure locality, it became clear that a brute force technique for monitoring all possible pairs of machines was inappropriate. First, such an approach may require excessive amounts of memory. On a large network of over 5000 machines, such a tool would have to keep track of over $\binom{5000}{2}$ connections, double this number if communication is expected to be asymmetric. Equally important, such a tool cannot track the shifts in locality that occur over time. What is needed is a tool that is adaptive to the current traffic patterns and can focus its attention on those pairwise communications that currently contribute to the bulk of the traffic.

3. Design of the Tool

3.1. Hardware

The hardware we used for IPwatch had to possess certain important characteristics. First, it had to be usable with Ethernet [3] as well the IBM token-ring [4] since both these technologies are prevalent in our environment. Second, it had to possess adequate memory and processing power to perform monitoring without any loss of information even during periods of intense network activity. Third, it had to be easily transportable so that it could be used anywhere on campus as well as at other sites.

As far as possible we wished to use off-the-shelf hardware that met our requirements. Fortunately such hardware is readily available. Our choice was an IBM AT with 640 Kbytes of memory, running at a clock speed of 6 Mhz. Besides meeting the criteria listed above, it also uses a lightweight operating system, MS-DOS, that provides us with easy access to all parts of the system. At the same time it has a rich collection of robust software development tools.

To interface the AT to the network we chose the Ungermann-Bass PC/NIC Ethernet adapter and the IBM 6100 token-ring adapter. These adaptors are fast and possess adequate buffering to handle bursts of traffic. They are capable of operating in *Promiscuous Mode*, which allows them to receive all packets on the network, not just packets specifically addressed to them. Our initial work has been entirely on the Ethernet. We intend to extend our implementation to the token ring shortly.

3.2. Software Overview

As defined in Section 2.2, a connection is a source-destination pair. IPwatch makes short-term locality visible by separating *Interesting* connections from *Uninteresting* ones. Each interesting connection has an associated entry in a *Long-Term Cache*. At present this entry is used to keep track of packet and byte counts. It would be simple to modify the tool to monitor other information such as interarrival time distributions or packet traces.

IPwatch uses a small *Short-Term Cache* to keep track of connections that it suspects may be interesting, but is not certain about. This cache is periodically swept and each entry is *Promoted* to the long-term cache or *Flushed*. The criteria for promotion is discussed in detail in Section 3.4. If a network exhibits significant short-term locality only a few of the connections in the short-term cache will be promoted. Further, the connections which are flushed will only account for a small fraction of the total traffic.

An interesting connection may become uninteresting over time. To maintain the invariant that the long-term cache contains only connections that are currently interesting, the long-term cache must also be swept periodically. The tool does not yet incorporate such a mechanism, but we are in the process of developing it. Fortunately, the long-term cache has never been completely filled in our limited use of the monitor.

When IPwatch receives a packet, it first checks the long-term cache to see if this connection is an interesting one. If not, it examines the short-term cache to to check if the connection is being considered for promotion. If a match is found in either cache, the corresponding entry is updated. Otherwise a new entry is created in the short-term cache, since the current packet could be the first of a long stream that renders this connection interesting.

3.3. Software Implementation

IPwatch consists of two modules: an interrupt handler written in assembler and a processing program written in C. When a packet arrives, the interrupt handler places it in a circular buffer. The processing program is triggered by the placement of packets in this buffer and by a timer that initiates sweeps of the short-term cache. The two modules are thus organized as a producer and a consumer, with the buffer allowing packets to be received even while the short-term cache is being swept or a previous packet is being processed.

Both short-term and long-term caches are organized as hash tables, indexed by connection. The short-term cache is 1024 entries long. This size has proved adequate for the range of two parameters that we have considered (10 to 45 seconds). In our experiments with IPwatch there has never been an occasion when the short-term cache has become full. The long-term cache consists of five independent tables, each containing 2000 entries. This organization is imposed by the addressing limitations of the IBM PC-AT that restrict the maximum size of contiguous data structures to 64 Kbytes. Although this restriction complicates programming it has not noticeably affected the performance of the tool. Entries in both caches are identical. Each entry is 22 bytes long and consists of the following fields:

```
struct STPD_entry
    {
    unsigned char ip_dest[4];   /* dest IP address */
    unsigned char ip_src[4];    /* src IP address */
    long pkts;                  /* no. pkts - this connection */
    long bytes;                 /* no. bytes - this connection */
    short front_ptr;            /* front ptr for chaining */
    short back_ptr;             /* back ptr for chaining */
    short table_no;             /* table number for chaining */
    short no_ticks_yet;         /* flag for new conns */
    short ticks_left;           /* counter - promotion/flushing */
    };
```

3.4. Promotion Algorithm

The promotion algorithm specifies the criteria for a connection to be considered interesting. The criterion we use is short-term traffic intensity, where the time period of observation is a few seconds or minutes. This criterion represents our personal bias, based on our experience of factors that affect performance in distributed systems. We believe that an event such as 10 packets being transmitted in one second between two machines is of greater significance than an event where those 10 packets are sent at the rate of one packet a day.

We measure traffic intensity on a connection using two parameters. The first parameter, *Time-to-Live* (τ), specifies the amount of time a connection spends in the short-term cache before being considered for promotion. The second parameter, *Packet Threshold* (π), specifies the minimum number of packets that must be seen on a connection during the time it is in the short-term cache. We ensure that each short-term entry lives at least τ seconds. If the first packet on an entry arrives just before a cache sweep (χ), the entry will have τ seconds before being considered for promotion; if it arrives just after a sweep it will have $\tau+\chi$ seconds. On the average, therefore, a connection must accumulate at least π packets in

τ+0.5χ seconds if it is to be promoted.

In assessing the effect of τ and π we need to consider their impact on two dependent variables. The first variable, *Packet Capture Ratio*, is defined as the ratio of the number of packets represented in the long-term cache to the total number of observed packets. If we view the packets corresponding to flushed connections as lost information, the packet capture ratio is a measure of the accuracy of the tool. With infinite resources we could keep a trace of all network traffic and not lose any information at all. The second dependent variable, *Promotion Ratio*, is defined as the fraction of connections entering the short-term cache that get promoted. This is a direct measure of short-term locality. The number of connections promoted is a direct measure of resource usage. If many connections are promoted a large long-term cache will be necessary.

The qualitative relationship between the input parameters of IPwatch and its dependent variables is easy to establish. With τ fixed, lower values of π will cause more connections to be promoted. Fewer connections will be flushed, so the packet capture ratio will be higher. With π fixed, larger values of τ will cause more connections to be promoted, again increasing the packet capture ratio.

If our hypothesis about short-term locality is true, these relationships will not be linear, though they will be monotonic. We should find it initially easy to improve packet capture ratio by lowering π or increasing τ. Beyond a certain threshold, intuitively corresponding to the "working set" of connections, such improvement should become much more difficult. How sharp the discontinuity in behavior is and where it occurs in the parameter space cannot be analytically determined. We therefore performed a number of experiments with the tool to empirically determine this relationship. These experiments are described in the next section.

4. Experiments with the Tool

Our experimental work was motivated by a number of distinct goals. First, our intuition about the presence of short-term locality in real networks had to be validated. Second, we were curious to know the degree of short-term locality in our environment. Was 80% of the traffic accounted for by 20% of the connections, as the classic 80/20 rule would imply? Or was the distribution skewed differently? Third, we needed to calibrate IPwatch by observing its behavior with respect to its input parameters τ and π. This information is important for our future work using the tool. Finally, we wished to understand how sensitive our observations were to the specific network being observed.

The following sections present our experimental work on short-term locality. Section 4.1 describes the environment in which we conducted our work. Section 4.2 describes the experimental setup and procedure, Section 4.3 analyzes the data, and Section 4.4 analyzes the sensitivity of this data with respect to the network.

4.1. Environment

The LAN at Carnegie Mellon University, where this work was performed, spans an area of about 1 square mile and encompasses 42 buildings. Figure 1 shows the details of this network. The LAN is composed of three different kinds of media: Ethernet, IBM token-ring, and Appletalk [1]. About 42% of the machines on the LAN are on Ethernet, 55% on token-ring, and 3% on Appletalk. There are approximately 60 subnets, usually one subnet per building, and a single Ethernet backbone.

TCP/IP [8, 10] is the dominant protocol in use, accounting for over 85 percent of the packet traffic. The routers that connect subnets to the backbone support a homogeneous IP address space. IP packets can be sent between any pair of machines at CMU without explicit routing information. Of the non-IP traffic, about 8 percent is Decnet [16]. The remaining 7% is composed of a miscellany of other protocols such as ARP [7], ICMP [9] and proprietary Apple protocols. Since TCP/IP has become a *de facto* standard at CMU we anticipate that its share of traffic will increase in the future. Consequently our tool only monitors IP traffic and ignores other packets.

There are approximately 2000 machines on the CMU LAN, spanning a diversity of types and using the network in a variety of ways. An approximate breakdown of these machines is as follows:

- 15 mainframes (IBM 3083, DEC20, etc.)
- about 70 minicomputers (VAX/780, PDP11/45, etc.)
- about 700 workstations (IBM PC-RT, DEC MicroVax, Sun-2, Sun-3, etc.)
- about 700 personal computers (IBM PC, Apple Macintosh, etc.)
- other miscellaneous machines, such as print servers and routers

Andrew workstations, of which there are over 500, interact primarily with file servers. The traffic they generate consists of machine-initiated file transfers using a non-TCP bulk transfer protocol, and remote procedure calls with relatively small arguments and results. Backups of Andrew file servers account for a large portion of the traffic. At night, 75 large disks from 17 file servers are backed up to tape, generating about a gigabyte of data.

The Computer Science Department (CSD) accounts for approximately 400 of the 2000 machines on campus. Few of these machines use Andrew. The workstations in CSD use the Software Update (SUP) protocol [14] to keep their files up to date. SUP traffic consists primarily of machine-initiated TCP-based file transfers from data servers. Paging traffic from diskless Sun workstations also contributes to traffic within the CSD subnet.

Human-initiated terminal emulation sessions, human-initiated file transfers, and electronic mail delivery are contributors to network traffic on the backbone and all subnets of the CMU LAN. On a few subnets , personal computers accessing PCServers (mentioned in Section 2.2) generate significant amounts of page-at-a-time file access traffic.

As one might expect, the backbone is the most heavily used network segment. It exhibited an average daily utilization of about 14% during the week when our experiments were performed. During the same period of time, the average daily backbone traffic amounted to over 44 million packets and 1.7 gigabytes.

4.2. Experiment Design

We first ran a number of preliminary experiments to debug our tool and to probe the parameter space for a range of interesting values for τ, the time-to-live and π, the packet threshold value. Based on the observations from these experiments we chose four values of π (2, 3, 5, and 30 packets) and four values of τ (10, 15, 30, and 45 seconds). This gave us sixteen different parameter combinations to investigate in detail.

We then set up a series of carefully controlled experiments on the CMU backbone, to observe the behavior of IPwatch for these parameter combinations. To achieve perfect experimental control we would have had to simultaneously run 16 instances of IPwatch, one for each parameter combination. To reduce the hardware requirements, we used four machines and ran four tests at a time, holding τ constant. Thus, with τ set to 10 seconds, we simultaneously ran the tests for π equal to 2, 3, 5, and 30 packets. This was then followed by the tests for τ set to 15 seconds, and so on. The short-term cache sweep time, χ, was constant at 5 seconds. Each test was run for 100 sweeps of the short-term cache.

IPwatch keeps counters of total packets, packets in promoted connections, total connections, connections promoted, and number of connections examined on each sweep of the short-term cache. From these counters we computed the following quantities:

- Packet capture ratio = packets in promoted connections / total packets
- Promotion ratio = connections promoted / total connections
- Memory usage = number of promoted connections + average number of entries in short-term cache

We express memory usage in terms of connections rather than bytes. Translating this value to bytes is trivial since a fixed amount of data is recorded for each connection.

We were interested in finding out whether short-term locality changes significantly from day to day and at different times of the day. Independent measurements of backbone utilization had indicated a peak at about 11am every day, a minimum around 7am, and a distinct period of activity around 2am when tape-backups were being performed. So we repeated the entire experiment at those three times every day, for a week that was representative of the academic year at CMU. A separate monitor was used to measure network utilization during our experiments. Since the data from our experiments was substantial (sixteen combinations run three times a day for seven days) we stored it in a relational database for convenient postprocessing.

An assumption implicit in our work is that the monitor is fast enough to observe every packet. Since traffic is bursty and there is limited buffering capacity in the network adaptors it is possible to miss packets during periods of intense network activity. Our adaptors have a status bit that is set whenever such an event occurs. IPwatch uses this status bit to keep track of the number of such events. Fortunately, there was no occasion when a packet was missed during our entire series of experiments.

4.3. Data Analysis

Initial analysis of the data showed that the variation between days was relatively small, particularly if we focused on weekdays. The weekends were similar to the weekdays in all respects except for lower utilization. We therefore focused on the weekdays and averaged the data for each time slot over 5 days. The resulting standard deviations are small relative to the mean, confirming that averaging across days is meaningful. This data is presented in Tables 3, 4, and 5. For the convenience of the reader, a subset of the data from Table 5 is graphically displayed in Figures 2 through 5. The network utilization on the backbone during these experiments is shown in Table1.

The existence of short-term locality is made apparent by the presence of a sharp knee in Figure 2 and Figure 4. Figure 2, for instance, shows that a 90% capture ratio is attained with only a 15% promotion ratio. In other words, 90% of the packets are accounted for by 15% of the connections. Beyond this, small increases in the capture ratio require large increases in promotion ratio. Within the limits of statistical variation, this relationship is true for all the data presented in Tables 3 to 5. Coarsely summarizing the data in these tables it appears that less than 5% of the connections account for 75% of the traffic, less than 15% connections for 90% traffic, less than 60% connections for 98% traffic, and less than 85% connections for 99% traffic. This is a rough characterization of the short-term locality observed on the CMU backbone.

Examination of Tables 3 to 5 indicates that for a given capture ratio there is little variation in promotion ratio between Tables 3, 4, and 5. However there is substantial difference in memory usage. In other words, the relative fraction of connections that are interesting does not vary significantly during the course of a day, although the total number of connections does. This is consistent with an interpretation that says that the factors contributing to short-term locality do not change even when the utilization of the network changes substantially.

This data also suggests the memory resources needed by IPwatch for medium-term monitoring of our LAN. For example, consider the data for 10 seconds in Table 5. As the threshold is increased from 2 to 3 packets, the packet capture ratio decreases from 98.6% to 93.1%. Memory usage, however, drops from 7041 entries to 3190. A relatively small sacrifice in accuracy thus leads to large savings in memory usage. The current implementation of IPwatch can have up to 10000 22-byte entries in its long-term cache. However, since the cache is organized as a hash table it should be operated well below capacity for good performance. It appears possible to operate it at about 30% capacity with an accuracy of over 90%. Alternatively we could double the amount of information maintained per connection and operate at the same accuracy using about 60% of the long-term cache capacity. The figures for memory utilization we have used here are an upper bound since they are obtained from Table 5, which corresponds to the most active part of the day. Medium-term monitoring at other times of the day will require less memory for the same accuracy.

For the set of parameter combinations we have studied, the tool is substantially more sensitive to the packet threshold π, than the time-to-live, τ. This observation holds quite strongly in all cases except for largest value of π (30 packets). Tables 3 to 5 show that a small change to π with τ fixed affects the capture ratio substantially. But changes to τ with π fixed barely change the capture ratio. The insensitivity to τ may be interpreted as meaning that a connection does not have to be monitored for long to determine if it is interesting.

For future work on our LAN we will probably use a value of 3 packets for π and 15 seconds for τ. This combination yields a capture ratio of over 90% with a promotion ratio of about 15% at all times of the day. The total memory utilization never exceeds 3000 entries, below a third of our long-term cache size.

4.4. Sensitivity of Data

Since locality is closely related to user and program behavior, it is necessary to observe a variety of networks to gain a comprehensive understanding of this property. We have taken an initial step in this direction by repeating the experiments described in Section 4.2 on the CSD subnet at CMU. As mentioned in Section 4.1, this subnet has software and usage patterns that differ significantly from the rest of CMU. It contributes little to the backbone traffic, since most traffic generated in it is addressed to hosts on the same subnet. To a first approximation, therefore, we may view the CSD subnet as an independent source of data.

The results of our experiments are shown in Tables 6, 7, and 8. As before, Figures 2 to 5 graphically depict a subset of this data. Table 2 presents the network utilization on the CSD subnet when these experiments were being run.

The presence of short-term locality is once again confirmed by the presence of a knee in Figures 2 and 3. In comparison to the backbone, the CSD subnet data shows significantly lower memory usage and promotion ratios for a given capture ratio. The lower memory usage is attributable to the fact that there are fewer machines using the CSD subnet than the backbone. This corresponds to fewer total connections in the CSD subnet. But the lower promotion ratio indicates higher short-term locality. Less than 5% of the connections account for over 90% of the data. This is in contrast to the backbone where the top 5% of the connections accounted for only 75% of the data.

Our comparison indicates that there is indeed substantial difference in short-term locality between networks. We intend to pursue this issue further by monitoring networks outside CMU.

5. Conclusion

The work described in this paper is only the beginning of an effort to understand the nature and extent of locality in distributed systems. We view the tool described here as a common back end for a number of more specialized tools. For example, we intend to build a a real-time graphical monitor that allows operations personnel to easily keep track of those pairs of communicating hosts contributing most to network activity. The monitor would be a simple extension of the tool described here, with mechanisms for aging long-term cache entries and for displaying the most active entries. Abnormalities caused by defective software or incorrect actions by operations personnel can be detected at an early stage by the use of this monitor. Such abnormalities have seriously inconvenienced Andrew users in the past.

To understand medium-term locality, we will augment the tool described in this paper with an interface to a database system. Long-term cache entries will be entered in the database when they are flushed by an aging mechanism. The database of locality information can be used to model and improve physical network locality. As discussed in Section 2.2, such improvement may involve changes to network topology, placement of hosts, and modifications to software.

A recurring issue in empirical work is the specificity of results to the environment in which the work was done. How unique are the results presented here to CMU? Do other sites exhibit similar locality properties? To answer these questions we intend to repeat the experiments described in this paper at other organizations that use large LANs. The simplicity and portability of our tool are important virtues in this context.

In conclusion, we believe that measurement of locality is critical for tuning the performance of a distributed system. The goal of our work was to produce a tool to assist in this task. We are pleased that our implementation is accurate yet frugal, and look forward to using it to enhance our understanding of distributed systems.

Value	Weekday	Weekend	Total
Avg. packets per day	44.4 million	28.3 million	39.8 million
Avg. bytes per day	1.7 gigabytes	1.1 gigabytes	1.5 gigabytes
Avg. utilization per day (24 hours)	15.5%	10.4%	14.1%
Avg. utilization at 2 am	12.9%	5.0%	10.7%
Avg. utilization at 7 am	10.8%	7.2%	9.7%
Avg. utilization at 11 am	14.8%	7.5%	12.7%

Table 1: Utilization values for the campus backbone, Dec. 1 through Dec. 8, 1987

Value	Weekday	Weekend	Total
Avg. packets per day	20.2 million	10.8 million	17.5 million
Avg. bytes per day	1.1 gigabytes	0.4 gigabytes	1.0 gigabytes
Avg. utilization per day (24 hours)	11.4%	3.5%	9.1%
Avg. utilization at 2 am	8.2%	5.0%	7.3%
Avg. utilization at 7 am	5.8%	4.0%	5.3%
Avg. utilization at 11 am	21.6%	5.0%	16.9%

Table 2: Utilization values for the CSD subnet, Dec. 8 through Dec. 16, 1987

Threshold (pkts)	Time to live (sec)	Pkt Capture Ratio (%)	Promotion Ratio (%)	Memory Usage (conns)
2	10 seconds	98.0 (1.0)	53.8 (16.9)	6157 (1739)
3		92.1 (3.1)	16.5 (2.9)	2662 (398)
5		88.5 (4.8)	9.8 (1.5)	2057 (369)
30		64.1 (10.0)	1.1 (0.3)	861 (135)
2	15 seconds	98.1 (0.9)	56.9 (17.3)	5931 (1710)
3		92.8 (3.2)	13.7 (2.8)	2043 (328)
5		90.6 (4.0)	8.6 (1.9)	1536 (361)
30		71.1 (9.2)	1.6 (0.3)	913 (160)
2	30 seconds	98.8 (0.9)	66.9 (23.6)	6104 (1687)
3		93.2 (3.0)	16.0 (5.5)	2177 (527)
5		91.8 (3.4)	10.8 (4.2)	1776 (419)
30		76.8 (8.7)	2.3 (0.6)	1178 (166)
2	45 seconds	99.2 (0.9)	76.8 (24.5)	6639 (1929)
3		94.4 (2.2)	25.4 (4.0)	3321 (438)
5		91.3 (3.7)	15.8 (3.2)	2663 (595)
30		74.8 (11.8)	2.6 (0.6)	1451 (37)

Average network utilization during this period was 12.9 %. Each data point is the average of 5 weekday trials. Figures in parentheses are standard deviations. A memory connection is 22 bytes long.

Table 3: Campus backbone results at 2 am

Threshold (pkts)	Time to live (sec)	Pkt Capture Ratio (%)	Promotion Ratio (%)	Memory Usage (conns)
2	10 seconds	98.0 (0.3)	57.8 (6.5)	5281 (1470)
3		90.7 (2.4)	15.1 (5.2)	1944 (608)
5		86.0 (2.8)	8.6 (1.3)	1486 (195)
30		61.4 (9.4)	0.6 (0.2)	563 (68)
2	15 seconds	98.1 (0.3)	57.0 (7.2)	5289 (1519)
3		91.5 (2.4)	10.9 (3.6)	1499 (363)
5		88.9 (2.8)	6.0 (2.6)	1063 (357)
30		71.4 (3.4)	1.1 (0.2)	604 (119)
2	30 seconds	99.3 (0.3)	70.6 (10.7)	5711 (1522)
3		93.1 (1.5)	15.0 (3.5)	1820 (446)
5		90.9 (1.9)	7.4 (2.5)	1257 (227)
30		80.8 (2.1)	1.7 (0.1)	830 (162)
2	45 seconds	99.6 (0.3)	79.5 (13.0)	6144 (1594)
3		94.8 (1.7)	23.8 (6.4)	2444 (281)
5		92.3 (1.9)	15.8 (3.2)	2078 (240)
30		78.1 (3.9)	1.8 (0.4)	1084 (179)

Average network utilization during this period was 10.8%. Each data point is the average of 5 weekday trials. Figures in parentheses are standard deviations. A memory connection is 22 bytes long.

Table 4: Campus backbone results at 7am

Threshold (pkts)	Time to live (sec)	Pkt Capture Ratio (%)	Promotion Ratio (%)	Memory Usage (conns)
2	10 seconds	98.6 (0.2)	58.7 (2.8)	7041 (769)
3		93.1 (1.6)	20.8 (2.7)	3190 (364)
5		89.3 (2.5)	13.3 (2.1)	2746 (183)
30		57.6 (4.2)	1.5 (0.1)	1080 (26)
2	15 seconds	98.5 (0.1)	59.6 (2.9)	6972 (697)
3		93.5 (0.8)	19.8 (2.8)	2868 (156)
5		90.9 (1.4)	13.2 (1.7)	2399 (156)
30		67.0 (2.2)	2.4 (0.2)	1233 (67)
2	30 seconds	99.1 (0.3)	74.4 (4.0)	7278 (1021)
3		94.1 (1.2)	23.2 (3.4)	3467 (444)
5		92.2 (1.0)	15.9 (1.5)	2664 (173)
30		73.6 (2.3)	3.7 (0.4)	1635 (112)
2	45 seconds	99.0 (0.9)	82.3 (5.5)	7459 (501)
3		94.9 (0.6)	31.3 (1.4)	4117 (150)
5		92.3 (0.8)	21.6 (1.0)	3604 (223)
30		73.7 (0.7)	4.0 (0.0)	1932 (110)

Average network utilization during this period was 14.9%. The data for 10 seconds is also presented in Figures 2 and 3. The data for 45 seconds is also presented in Figures 4 and 5. Each data point is the average of 5 weekday trials. Figures in parentheses are standard deviations. A memory connection is 22 bytes long.

Table 5: Campus backbone results at 11am

Threshold (pkts)	Time to live (sec)	Pkt Capture Ratio (%)	Promotion Ratio (%)	Memory Usage (conns)
2	10 seconds	99.5 (0.2)	20.7 (2.0)	449 (43)
3		98.9 (0.4)	16.0 (0.8)	365 (18)
5		97.0 (1.5)	12.9 (4.9)	339 (66)
30		86.3 (6.7)	2.9 (2.0)	150 (39)
2	15 seconds	99.5 (0.1)	24.4 (3.3)	527 (66)
3		98.6 (0.4)	15.6 (1.3)	355 (26)
5		97.6 (1.2)	12.8 (6.2)	303 (119)
30		94.7 (1.3)	4.6 (2.2)	134 (32)
2	30 seconds	99.5 (0.2)	22.1 (0.8)	499 (22)
3		99.3 (0.2)	19.3 (3.1)	446 (64)
5		98.3 (0.8)	13.8 (3.7)	343 (66)
30		94.7 (1.8)	5.5 (1.9)	194 (47)
2	45 seconds	99.7 (0.2)	32.1 (13.5)	702 (276)
3		99.4 (0.2)	20.4 (0.9)	478 (23)
5		98.7 (0.6)	17.4 (6.5)	424 (124)
30		96.4 (1.4)	6.6 (2.1)	228 (63)

Average network utilization during this period was 8.2%. Each data point is the average of 5 weekday trials. Figures in parentheses are standard deviations. A memory connection is 22 bytes long.

Table 6: CSD subnet results at 2am

Threshold (pkts)	Time to live (sec)	Pkt Capture Ratio (%)	Promotion Ratio (%)	Memory Usage (conns)
2	10 seconds	99.1 (0.2)	16.9 (0.7)	368 (14)
3		98.2 (0.4)	11.3 (1.5)	256 (29)
5		96.8 (1.0)	8.1 (0.9)	196 (16)
30		92.1 (2.4)	3.0 (0.7)	91 (17)
2	15 sections	99.2 (0.3)	17.9 (5.9)	391 (121)
3		98.3 (0.5)	11.5 (2.1)	266 (46)
5		97.2 (0.4)	10.6 (0.5)	258 (11)
30		92.4 (0.6)	3.1 (0.8)	105 (19)
2	30 seconds	99.5 (0.2)	22.0 (6.6)	497 (140)
3		99.1 (0.3)	16.8 (6.2)	389 (128)
5		97.7 (0.6)	9.5 (0.9)	248 (20)
30		93.0 (0.9)	3.3 (0.5)	124 (15)
2	45 seconds	99.9 (0.0)	30.9 (3.4)	676 (67)
3		99.4 (0.2)	22.7 (4.8)	521 (106)
5		98.5 (0.6)	13.9 (5.9)	359 (130)
30		94.3 (0.3)	3.8 (0.2)	152 (5)

Average network utilization during this period was 5.8%. Each data point is the average of 5 weekday trials. Figures in parentheses are standard deviations. A memory connection is 22 bytes long.

Table 7: CSD subnet results at 7am

Threshold (pkts)	Time to live (sec)	Pkt Capture Ratio (%)	Promotion Ratio (%)	Memory Usage (conns)
2	10 seconds	99.6 (0.1)	24.0 (2.0)	521 (41)
3		99.0 (0.5)	15.6 (1.5)	357 (28)
5		98.2 (0.8)	11.2 (1.6)	277 (31)
30		94.4 (2.2)	4.6 (0.1)	150 (0)
2	15 seconds	99.6 (0.1)	20.0 (1.9)	443 (39)
3		99.3 (0.2)	18.3 (1.5)	418 (28)
5		98.5 (0.5)	13.1 (0.8)	320 (14)
30		96.3 (1.3)	5.9 (0.3)	179 (14)
2	30 seconds	99.5 (0.1)	21.8 (3.5)	504 (70)
3		99.4 (0.2)	16.5 (0.6)	399 (14)
5		99.1 (0.4)	14.2 (1.6)	363 (34)
30		94.1 (2.0)	5.5 (0.7)	213 (0)
2	45 seconds	99.9 (0.0)	32.6 (4.5)	726 (95)
3		99.5 (0.2)	21.1 (2.2)	505 (43)
5		98.9 (0.4)	13.8 (1.0)	364 (20)
30		95.9 (0.7)	5.8 (0.8)	214 (13)

Average network utilization during this period was 21.6%. The data for 10 seconds is also presented in Figures 2 and 3. The data for 45 seconds is also presented in Figures 4 and 5. Each data point is the average of 5 weekday trials. Figures in parentheses are standard deviations. A memory connection is 22 bytes long.

Table 8: CSD subnet results at 11am

Figure 1: The CMU LAN

This data is obtained from Tables 5 and 8. Four data points are shown, corresponding to the four threshold values. The left-most point corresponds to a threshold value of 30 packets; the right-most point corresponds to a threshold value of 2 packets.

Figure 2: Promotion ratio versus packet capture ratio

This data is obtained from Tables 5 and 8. Four data points are shown, corresponding to the four threshold values. The left-most point corresponds to a threshold value of 30 packets; the right-most point corresponds to a threshold value of 2 packets.

Figure 3: Memory usage versus packet capture ratio

This data is obtained from Tables 5 and 8. Four data points are shown, corresponding to the four threshold values. The left-most point corresponds to a threshold value of 30 packets; the right-most point corresponds to a threshold value of 2 packets.

Figure 4: Promotion ratio versus packet capture ratio

This data is obtained from Tables 5 and 8. Four data points are shown, corresponding to the four threshold values. The left-most point corresponds to a threshold value of 30 packets; the right-most point corresponds to a threshold value of 2 packets.

Figure 5: Memory usage versus packet capture ratio

References

1. *AppleBus Link Access Protocol Specification Version 1.0.* Apple Computer, Inc., 1984.

2. *LANalyzer EX 5000E Ethernet Network Analyzer User Manual.* Excelan Incorporated, San Jose, California, 1986.

3. Technical Committee Computer Communications of the IEEE Computer Society. *Carrier Sense Multiple Access Method and Physical Layer Specifications (CSMA/CD) ANSI/IEEE Std 802.3.* The Institute of Electrical and Electronic Engineers, 1985.

4. Technical Committee Computer Communications of the IEEE Computer Society. *Token-Ring Access Method and Physical Layer Specifications, ANSI/IEEE Std 802.5.* The Institute of Electrical and Electronic Engineers, 1985.

5. Morris, J. H., Satyanarayanan, M., Conner, M.H., Howard, J.H., Rosenthal, D.S. and Smith, F.D. "Andrew: A Distributed Personal Computing Environment". *Communications of the ACM 29*, 3 (March 1986).

6. Nichols, D.A. Using Idle Workstations in a Shared Computing Environment. Proceedings of the 11th ACM Symposium on Operating System Principles, November, 1987.

7. Plummer, David C. An Ethernet Address Resolution Protocol. Tech. Rept. RFC826, Network Working Group, 1982.

8. Postel, Jon, Editor. Internet Protocol DARPA Internet Program Protocol Specification. Tech. Rept. RFC791, DARPA, 1981.

9. Postel, Jon. Internet Control Message Protocol DARPA Internet Program Protocol Specification. Tech. Rept. RFC792, Network Working Group, 1981.

10. Postel, Jon, Editor. Transmission Control Protocol DARPA Internet Program Protocol Specification. Tech. Rept. RFC793, DARPA, 1981.

11. Raper, L.K. The CMU PC Server Project. Tech. Rept. CMU-ITC-051, Information Technology Center, Carnegie Mellon University, February, 1986.

12. Rosenberg, J., Everhart, C.F. and Borenstein, N.S. An Overview of the Andrew Message System. Proceedings of the ACM Sigcomm ’87 Workshop, Stowe, Vermont, August, 1987.

13. Satyanarayanan, M., Howard, J.H., Nichols, D.N., Sidebotham, R.N., Spector, A.Z. and West, M.J. The ITC Distributed File System: Principles and Design. Proceedings of the 10th ACM Symposium on Operating System Principles, December, 1985.

14. Shafer, Steven. The SUP Software Upgrade Protocol. Carnegie Mellon University, Computer Science Department, 1985.

15. Striemer, Bryan L., Lorence, Mark J. Monitoring Local Area Networks at Carnegie-Mellon University: Tools for Network Planning. Tech. Rept. 07.885, IBM Corporation, 1988.

16. Wecker, S. "DNA: the Digital Network Architecture". *IEEE Transactions on Communications COM-28* (April 1980).

Table of Contents

List of Figures

List of Tables