The NSFNET Backbone Network[1,2]

David L. Mills
Electrical Engineering Department
University of Delaware

Hans-Werner Braun
Computer Center
University of Michigan

Abstract

The NSFNET Backbone Network interconnects six supercomputer sites, several regional networks and ARPANET. It supports the DARPA Internet protocol suite and DCN subnet protocols, which provide delay-based routing and very accurate time-synchronization services. This paper describes the design and implementation of this network, with special emphasis on robustness issues and congestion-control mechanisms.

1. Introduction and Background

The NSFNET is a loosely organized community of networks funded by the National Science Foundation to support the sharing of national scientific computing resources, data and information [7]. NSFNET consists of a large number of industry and academic campus and experimental networks, many of which are interconnected by a smaller number of regional and consortium networks. The NSFNET Backbone Network is a primary means of interconnection between the regional networks and is the subject of this report.

The NSFNET Backbone Network, called simply the Backbone in the following, includes switching nodes located at six supercomputer sites: San Diego Supercomputer Center (SDSC), National Center for Supercomputer Applications (NCSA) at the University of Illinois, Cornell National Supercomputer Facility (CNSF), Pittsburgh Supercomputer Center (PSC), John von Neumann Center (JVNC) and the National Center for Atmospheric Research (NCAR). The six nodes are interconnected by 56-Kbps internode trunks (see Figure 1).The Backbone is extended for regional interconnects (not shown) to the University of Michigan and the University of Maryland, with a further one planned at Rice University. Additional nodes (not shown) are used for program development and testing, bringing the total to about thirteen.

Each Backbone node is connected to an onsite Ethernet, which serves as the attachment point for supercomputers and other local hosts. Most sites have an extensive system of local networks and gateways, including high-speed bus, ring and point-to-point links, which serve to concentrate traffic from throughout the site. Other gateways connect to regional and consortium networks, which

1. Sponsored by: Defense Advanced Research Projects Agency contract number N00140-87-C-8901 and by National Science Foundation grant number NCR-86-12015.

2. Reprinted from: Mills, D.L., and H.-W. Braun. The NSFNET Backbone Network. *Proc. ACM SIGCOMM 87 Symposium* (Stoweflake VT, August 1987), 191-196.

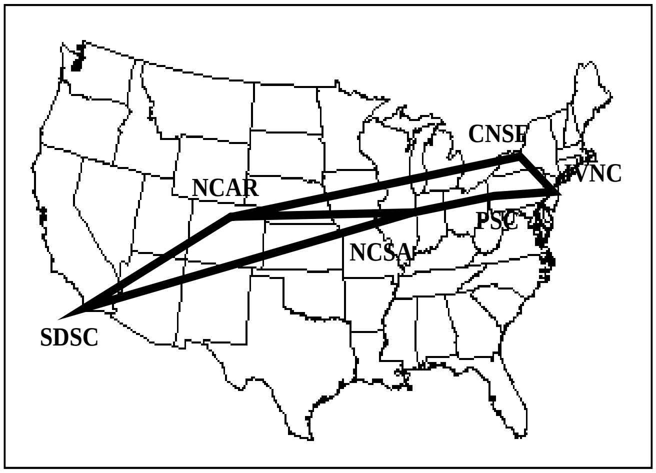

Figure 1. NSF Backbone Network

in some cases span large regions of the country. Some sites are connected to other backbone networks such as ARPANET and public data networks as well.

The Backbone uses the DARPA Internet architecture, which is based on the IP and TCP protocols [8]. Most of the regional and consortium networks, as well as the campus networks they connect also use these protocols. There are several thousand service hosts and gateways connected to the Internet, as well as many more personal computers and workstations. In late July, 1987, there were 4625 hosts on 676 networks interconnected by 184 gateways listed at the Department of Defense Network Information Center alone, which by itself is only a small fraction of the overall Internet. There are presently about 63 networks either directly connected to the Backbone or by means of gateways and other regional and consortium networks, while over 250 networks are in regular operation on the Internet system as a whole.

In following sections the Backbone subnet architecture and protocols are described along with its hardware and software components. Its design features are summarized, including factors related to robustness, congestion control and services. Operation and maintenance issues are described, including system control, monitoring and performance measurement. Finally, plans for further expansion are summarized.

2. Network Architecture

The Backbone, as well as the onsite local-net complexes, regional networks and the campus networks they connect, are part of the Internet System developed by the Defense Advanced Research Agency (DARPA) over the last several years and conform to its architecture and protocols. The Internet operates in connectionless mode using the Internet Protocol (IP) [20] as the basic internetworking mechanism. End-to-end reliability is maintained using the Transmission Control Protocol (TCP) [22], which assembles and reorders datagrams (protocol data units) received over possibly diverse and unreliable paths using retransmissions as necessary. The User Datagram Protocol (UDP) [19] provides direct IP datagram access for transaction services, including routing and network control in some cases.

Since the basic service expected of the Backbone is connectionless, no provision for end-to-end reassembly, reordering or retransmission is necessary. The network does not support end-to-end virtual circuits and has no implicit connection-setup or other resource-binding mechanisms as does, for example the ARPANET. However, in order to improve overall service, reliable (retransmission) services are provided on selected internode trunks, in particular the 56-Kbps trunks interconnecting the Backbone sites, which use the DEC Digital Data Communications Message Protocol (DDCMP) for the pragmatic reason that the hardware interfaces happen to support this protocol.

2.1 Subnet Architecture

The Backbone subnet protocols are based on the Distributed Computer Network (DCN), which uses Internet technology and an implementation of PDP11-based software called the Fuzzball. DCN networks of hosts and gateways are now in regular service in the INTELPOST facsimile-mail system, which was built by COMSAT Laboratories and operated by the U.S. Post Office and international affiliates, as well as the Backbone and about a dozen campus networks in the U.S. and Europe, including the Universities of Maryland, Michigan and Delaware, Ford Scientific Research Laboratories and M/A-COM Linkabit.

The DCN architecture is intended to provide connectivity, routing and timekeeping functions for a set of gateways, service hosts and personal workstations using a specialized protocol called HELLO [10], which is based on IP. HELLO services include delay-based routing and clock-synchronization functions in an arbitrary topology including point-to-point links and multipoint bus systems. However, the DCN architecture is not intended for use in very large networks such as ARPANET, since it does not include load-adaptive routing algorithms and comprehensive congestion controls.

A brief description of the process and addressing structure used in the DCN may be useful in the following. A physical host is a PDP11-compatible processor which supports a number of cooperating sequential processes, each of which is given a unique identifier called its port ID. Every physical host contains one or more designated internet processes, each of which supports a virtual host assigned a unique identifier called its host ID. Virtual hosts can migrate among the physical hosts at will, as long as their host IDs remain unchanged, since the routing tables are automatically updated by the HELLO protocol.

The physical host also supports other processes for input/output devices (disks, terminals and network-interface devices), as well as spooling systems, various network daemons and users, which are provided with separate virtual address spaces. The physical host is identified by a host ID for the purpose of detecting loops in routing updates, which establish the minimum-delay paths between the virtual hosts. Additional host IDs are assigned dynamically by the operations of other routing and address-binding protocols such as the Internet Control Message Protocol (ICMP) [21], Address Resolution Protocol (ARP) [18], Exterior Gateway Protocol (EGP) [11] and related protocols.

Each virtual host can support multiple transport protocols, connections and, in addition, a virtual clock. Selected virtual hosts can act as gateways to other networks as well. Each physical host contains a physical clock which can operate at an arbitrary rate and, in addition, a 32-bit logical clock which operates at 1000 Hz and is assumed to be reset each day at 0000 hours UT. Not all physical hosts implement the full 32-bit precision; however, in such cases the resolution of the logical clock may be somewhat less.

DCN networks are self-configuring for all hosts and networks; that is, the routing algorithm will automatically construct entries in the various tables, find minimum-delay paths and synchronize logical clocks among all virtual hosts and gateways supporting the DCN protocols. For routing beyond the span of the DCN routing algorithm, the tables can be pre-configured or dynamically updated using the ICMP, ARP and EGP protocols. In addition, a special entry can be configured in the tables which specifies the gateway for all address ranges not explicitly designated in the tables.

2.2 Subnet Addressing and Routing

The correspondence between IP addresses and host IDs is determined by two tables, the Local Mapping Table and the Global Mapping Table, which are structured in the same way. Each entry in these tables defines a range of IP addresses which map onto a specified host ID and thus a virtual host. There is no restriction on the particular range or ranges assigned a virtual host, so that these hosts can be multi-homed at will and in possibly exotic ways. The mapping function also supports the subnetting and filtering functions outlined in [2]. By convention, one of the addresses assigned to a virtual host in each physical host is declared the base address of the physical host itself. Entries in these tables can be pre-configured or dynamically updated using the HELLO, ICMP, ARP and EGP protocols.

Datagram routing is determined entirely by IP address - there is no subnet address as in the ARPANET. Each physical host contains a table called the Host Table, which is used to determine the port ID of the network-output process on the minimum-delay path to each virtual host. This table also contains estimates of roundtrip delay and logical-clock offset for all virtual hosts indexed by host ID. For the purpose of computing these estimates the delay and offset of each virtual host relative to the physical host in which it resides is assumed zero. The single exception to this is a special virtual host associated with an NBS radio time-code receiver, where the offset is computed relative to the broadcast time.

Host Table entries are updated by HELLO messages exchanged frequently over the links connecting physical-host neighbors. At present, the overhead of these messages is controlled at about 3.4 percent of the aggregate network traffic. They include data providing an accurate measurement of delay and offset between the neighbors on the link itself, as well as a list of the delay and offset entries in the Host Table for all virtual hosts. There are two list formats, a short format with indexed entries used when the neighbors share the same subnet and a long format including the IP address used in other cases.

The routing algorithm is a member of the Bellman-Ford class [1], which includes those formerly used in the ARPANET and presently used in several Internet gateway systems. The measured roundtrip delay to the neighbor is added to each of the delay estimates in its HELLO message and compared with the corresponding delay estimates in the Host Table. If the sum is less than the value already in the Host Table or if the HELLO message is received on the next-hop interface, as previously computed by the routing algorithm, the sum replaces the value and the routing to the corresponding virtual host is changed accordingly. In other cases the value in the Host Table remains unchanged.

Each entry in the Host Table is associated with a time-to-live counter, which is reset upon arrival of an update for the entry and decrements to zero otherwise. If this counter reaches zero, or if an update specifying infinite distance is received on the next-hop interface, the entry is placed in a hold-down condition where updates are ignored for a designated interval, in the Backbone case two minutes. The hold-down interval is necessary for old routing data, which might cause loops to form, to be purged from all Host Tables in the system. In order to further reduce the incidence of loops, the delay estimate is set at infinity for all hosts for which the next-hop interface is the one on which the HELLO message is sent, regardless of the value in the Host Table.

3. Switching Nodes

A Backbone node consists of a Digital Equipment Corporation LSI-11/73 system with 512K bytes of memory, dual-diskette drive, Ethernet interface and serial interfaces. One or two low-speed asynchronous interfaces are provided, as well as one to three high-speed synchronous interfaces. All Backbone nodes include crystal-stabilized time bases. One node (NCAR) is equipped with a WWVB radio time-code receiver providing a network time reference accurate to the order of a millisecond.

Other nodes connected to the Backbone and running DCN protocols use LSI-11 and other PDP11-compatible systems with from 256K to 2048K bytes of memory, plus various hard disks and serial interfaces, including ARPANET interfaces, X.25 interfaces and terminal multiplexors. Most of these nodes also include crystal-stabilized time bases, while two are equipped with WWVB time-code receivers and one with a GOES time-code receiver. Some of these systems are used for general-purpose network access for mail, word-processing and file staging, as well as packet-switching and gateway functions.

The software system used in the Backbone nodes, called the Fuzzball, includes a fast, compact operating system, comprehensive network-support system and a large suite of application programs for network protocol development, testing and evaluation. The Fuzzball software has been rebuilt, modified, tinkered and evolved over several generations spanning a twenty-year period [9]. It has characteristics similar to many other operating systems, in some cases shamelessly borrowing their features and in others incorporating innovative features well before other systems made these features popular.

Originally, the Fuzzball was designed primarily as an investigative tool and prototyping workbench. Many Fuzzballs have been deployed for that purpose at various locations in the U.S. and Europe, including Norway, United Kingdom, Germany, Holland and Italy. Various organizations use Fuzzballs as terminal concentrators, electronic-mail and word-processing hosts, network monitoring and control devices and general-purpose packet-switches and gateways. For the Backbone the Fuzzball is used primarily as a packet switch/gateway, while the application programs are used for network monitoring and control.

The Fuzzball implementation incorporates complete functionality in every host, which can serve as a packet switch, gateway and service host all at the same time. The system includes host and gateway software for the complete DARPA Internet protocol suite with network, transport and applications-level support for virtual-terminal and file-transfer services, along with several mail systems with text, voice and image capabilities. In order to provide a comprehensive user interface and platform for program development and testing, a multiple-user, virtual-machine emulator supports the Digital Equipment Corporation RT-11 operating system for the PDP11 family, so that RT-11 program-development utilities and user programs can be run along with network-application programs in the Fuzzball environment.

3.1 Robustness Issues

When the Internet was small and growing rapidly there was great concern about its potential vulnerability to destructive routing loops or black holes that could form when more than one routing algorithm was used or when an protocol misbehaved because of algorithmic instability or defective implementation. The solution to this problem was to partition the Internet into multiple, independent systems of gateways, called autonomous systems, where each system could adopt any routing algorithm it chose, but exchange routing information with other systems using the Exterior Gateway Protocol (EGP).

The expectation was that the Internet would evolve into a relatively small, centrally managed set of backbone gateways called the core system, together with a larger set of unmanaged gateways grouped into stub systems with single-point attachments to the core system. In this model the stub systems would normally not be interconnected to each other, except via the core system, with exceptions handled on an ad-hoc, engineered basis.

As the Internet evolved into a richly interconnected, multiple-backbone topology with large numbers of regional and campus networks, the stub-system model became less and less relevant. Requirements now exist in NSFNET where gateways within and between autonomous systems need to interoperate with different routing algorithms and metrics and with different trust models. Backbones now connect to backbones, while regional systems now connect wily-nily to each other and to multiple backbones at multiple points, so that the very concept of a core system as effective management tool has become obsolete.

As specified, EGP by is designed primarily to provide routing information between the core system and stub systems. In fact, only the core system can provide routing information for systems not directly connected to each other. The enhancements to EGP described in [14] suggest restructuring the Internet as a number of autonomous-system confederations, as well as an outline for a universal metric. Neither the baseline or enhanced EGP model is adequate to cope with the evolving requirements of NSFNET.

A great deal of study was given these issues during the design phase of the Backbone. One issue is the vulnerability of NSFNET as a whole to routing loops, either due to adventurous, unstable configurations or defective implementations. Another is the robustness of the various metrics (e.g. hop-count based and delay based) with respect to the various transformations required between them. Still another is protection from false or misleading addressing information received from or transmitted to neighboring systems. Each of these issues will be discussed in following sections.

3.2 Metric Transformations

Since it is not possible for the Backbone routing algorithm to have unlimited scope, there exist demarcations where the algorithm must interoperate with other routing algorithms, protocols and metrics. In order to support multiple routing algorithms in a single autonomous system or confederation, it is necessary to explore how they can safely interoperate without forming destructive routing loops.

Consider two Bellman-Ford algorithms one with metric R, which might for example represent hop count, and the other with metric H, which might represent measured delay. Nodes using each metric send periodic updates to their neighbors, some of which may use a different metric. Each node receiving an update in a different metric must be able to transform that metric into its own. Suppose there are two functions: F_h, which maps R to H, and F_r, which maps H to R. In order to preserve the non-negative character of the metrics, both F_h and F_r must be positive and monotone-increasing functions of their arguments.

It is not difficult to show [16] that loops will not occur if both of the following conditions are satisfied:

$$x \le F_r(F_h(x)) \quad \text{and} \quad x \le F_h(F_r(x)) \tag{1}$$

As long as these conditions are satisfied and both the domains and ranges are restricted to non-negative values, mutually inverse functions for F_h and F_r can readily be found, such as linear transformations $Ax + B$, powers x^n and exponentials e^x, together with their inverses. Note that these conditions require careful analysis of the finite-precision arithmetic involved and the errors inevitably introduced.

Several of the Backbone nodes are connected to extensive regional networks, some of which use a routing protocol called the Routing Information Protocol (RIP) [6]. In some cases a regional network is connected to more than one Backbone node. A typical case involves the translation between RIP and HELLO at both sites, in which case the above conditions come into play. Note that these conditions do not guarantee the shortest path relative to either metric, just that whatever path is chosen, no loops will form.

3.3 Fallback Routing

Ordinary routing algorithms compute shortest paths on a directed, labeled graph. If there are multiple paths between given endpoints, the algorithm will select the one with minimum total distance, but will make an arbitrary choice when more than one path exists with that distance. A reachability algorithm is defined as a routing algorithm in which all paths between given endpoints have the same distance; therefore, the algorithm will select one of them arbitrarily. In practice such algorithms are useful mainly in tree-structured topologies where autonomous systems with only a few reachable networks are interconnected by one or at most a few gateways, such as the stub-system model commonly associated with EGP.

Cases exist in NSFNET where several autonomous systems with many reachable networks are haphazardly interconnected by multiple gateways. In order to insure stability, it is desirable to hide the internal routing details of each system; however, for reasons of load balancing it is desirable to control which gateway is to be used for normal traffic in to and out of the system and which is to be used as backup should the normal gateway fail. A fallback algorithm is defined as a routing algorithm in which two sets of paths exist between given endpoints, one intended as primary paths and the other as fallback paths should all primary paths fail. However, in both the primary or fallback set, the algorithm will select one of them arbitrarily.

In reachability algorithms it is not necessary to know the distance along the path selected, only that it exists (i.e. the distance is less than infinity); therefore the metric has only two values: zero and infinity. In fallback algorithms a finer distinction is necessary in order to determine whether a primary or fallback path is in use; therefore, the metric has three values: zero, one and infinity. It is not difficult to invent metric transformations which preserve this distinction without introducing loops [16].

Fallback routing is now used by the EGP-speaking gateways between the various Backbone site networks and the core system. For each Backbone network one of these gateways is designated primary and uses an EGP metric of zero, while the remaining gateways are designated fallback and use a nonzero metric. The primary gateway is assigned on the basis of pre-engineered configurations and traffic forecasts. As a special experimental feature, the core-system EGP implementation incorporates an ad-hoc form of fallback routing. The effect is that, if the primary gateway for a particular network fails, the load is nondeterministically shared among the fallback gateways.

4. Routing Agents

Since the Backbone nodes are connected directly to Ethernets serving a general population of potentially defective hosts and gateways, the Backbone design includes a special routing agent which filters information sent between the switching nodes and other gateways in the local autonomous system. In order to conserve resources in the node itself, the agent is implemented as a daemon in a trusted Unix-based host attached to the same Ethernet. The agent, now installed at all Backbone sites, mitigates routes computed by other routing systems, such as RIP and/or EGP, and communicates with the Backbone node using the HELLO protocol. It consists of a portable C-language program for the Berkeley 4.3 Unix system distribution [5].

Among the features implemented in the routing agent are exclusion lists, which delete selected networks known to the local routing algorithm from HELLO messages sent to the Backbone node. Others include calculation of the metric transformations, when required, and management of the various data bases involved. At present, the resources necessary to operate the routing agent are provided by the sites themselves, while configuration control of the data bases is maintained by the network operations center.

5. Congestion Control

Like many networks designed for connectionless-mode service, the Backbone does not bind resources to end-to-end flows or virtual circuits. In order to deal with traffic surges, the Internet architecture specifies the ICMP Source Quench message, which is in effect a choke packet sent to the originating host when a downstream gateway experiences congestion. While the choke packet can be an effective mechanism to control long-term flows; that is, when the flow intensities are relatively stable over periods longer then the nominal transit time of the network, it is usually not an effective mechanism in other cases.

Therefore, when a short-term traffic surge occurs, the only defense possible is to either drop arriving packets or selectively preempt ones already queued for transmission. Previous designs drop arriving packets when the buffer pool becomes congested, which has unfortunate consequences for fairness and end-to-end performance. Simply increasing the size of the buffer pool does not help [17]. In addition, it has long been suspected that a major cause of Internet traffic surges is defective transport-level implementations or antisocial queueing policies, resulting in large, uncontrolled bursts of packets. Thus, an effective preemption strategy must take fairness into account in order to avoid capture of excessive network resources by reckless customers.

Extensive experience in the design, implementation and experimental evaluation of connection-less-mode networks suggests an interesting preemption strategy which has been implemented in the Fuzzball system. It is based on two fairness principles:

1. Every customer (IP source host) has equal claim on buffer resources, so that new arrivals can preempt other customers until the space claimed by all customers is equalized.

2. When a preemption is necessary for a customer with buffers on multiple queues, the preemption rates for each of these queues are equalized.

The intent of the first rule is to identify the customer capturing the most buffer space, since this customer is most likely a major contributor to the congestion. The intent of the second rule is to spread the preemptions evenly over the output queues in case of ties.

It is not possible without a heavy performance penalty to implement the above rules in their purest form. In the Fuzzball implementation an input buffer is almost always available for an arriving packet. Upon arrival and inspection for correct format and IP checksum, the (sometimes considerable) unused space at the end of the buffer is returned to the buffer pool and the packet inserted on the correct output queue, as determined by the routing algorithm. A preemption is necessary when an input buffer must be allocated for the next following packet.

When preemption is necessary, each output queue is scanned separately to find the customer with the largest number of 512-octet blocks. Then the queue with the largest number of such blocks is determined and the last buffer for the associated customer is preempted, even if the buffer preempted was the one just filled. In case of ties, the queue with the most packets transmitted since the last preemption is chosen. The entire process is repeated until sufficient buffer space is available for the input buffer request.

The experience with the Fuzzball implementation has been very satisfying, as shown below and in Section 7. Table 1 Illustrates the performance of the policy over a typical period of several days.

Line	Rate	Timeout	Preempt	Total
1	0.32	.767	.0	.767
2	0.62	.504	.0	.504
3	1.56	.058	.0	.058
4	1.91	.020	.0	.020
5	0.30	.059	.0	.059
6	0.58	.141	.0	.141
7	2.23	.018	.025	.044
8	3.02	.045	.018	.063

Figure 2. Dropped Packet Rate

Line	Rate	Timeout	Preempt	Total
9	1.82	.110	.026	.137
10	1.61	.056	.0	.056
11	2.20	.021	.162	.184
12	3.41	.059	.071	.130
13	3.79	.034	.027	.061
14	3.98	.027	.0	.027
15	2.79	.033	.0	.033
16	1.39	.052	.004	.056
Average	1.97	.060	.028	.088

Figure 2. Dropped Packet Rate (Continued)

Therre sixteen internode trunks in the Backbone (including the SURA regional network). The Rate column shows the mean packets per second sent on the trunk, while the Timeout and Preempted columns show the percentage of packets deleted from the trunk queue due these causes.

These data should be compared with the weekly statistics collected for the seven ARPANET/MILNET gateways operated by Bolt Beranek Newman for the Defense Communication Agency [3]. These gateways, which operate in an environment similar to the Backbone, drop an arriving packet when the output queue for an ARPANET/MILNET destination subnet address exceeds eight packets. On a typical week in late July 1987 these gateways carried an aggregate of 56.74 packets per second for an equivalent of 14 lines, with a mean drop rate of 7.035 percent, almost two orders of magnitude greater than the Backbone. The busiest gateway carried an estimated 6.44 packets per second per line and dropped 12.5 percent of these packets.

6. Network Services

The Backbone nodes are intended primarily to serve as IP packet switches and gateways for NSFNET client networks. However, There are several other services available, some for the general user population and others for monitoring and control purposes. These include some applications based on TCP and some on UDP (see [4] for service descriptions and protocols, unless indicated otherwise):

1. TCP-based virtual-terminal (TELNET), file-transfer (FTP) and mail (SMTP) services intended for system monitoring and control purposes.

2. UDP-based name-lookup (NAME and DOMAIN-NAME), file-transfer (TFTP) and time (TIME, NTP) services, as well as a special statistics (NETSPY) service [15] intended for network monitoring.

3. IP-based utilities (ECHO, TIMESTAMP), primarily intended for system monitoring and fault isolation.

The UDP-based time services TIME and NTP are unique features of the Fuzzball. The physical-clock hardware and Fuzzball software, as well as the DCN protocols, have been specially designed to maintain network time synchronization to an unusual precision, usually less than a few milliseconds relative to NBS broadcast standards. The Network Time Protocol (NTP) [13] implemented by every Fuzzball provides accurate timestamps in response to external requests, as well as providing internal synchronization and backup for a network of NTP servers spanning the entire Internet.

There are presently five Fuzzball systems with WWVB or GOES time-code receivers on the Internet, with at least one attached via high-speed lines to the Backbone, ARPANET and MILNET. A conforming NTP daemon program has been written for the Berkeley 4.3 Unix system distribution. A discussion of the synchronization algorithms used can be found in [12].

7. Network Operations

The NSFNET Backbone Network Project is presently managed by the University of Illinois. Network operations, including configuration control and monitoring functions, are managed by Cornell University. Additional technical support is provided by the Information Sciences Institute of the University of Southern California, and the Universities of Michigan, Delaware and Maryland. The NSFNET Network Services Center (NNSC), operated by Bolt Beranek Newman, provides end-user information and support.

The NSF Information and Services Center (NISC) at Cornell University is presently responsible for the day-to-day operations and maintenance functions of the Backbone. They are assisted by staff at the various sites and regional operating centers for hardware maintenance, as well as the resolution of node and trunk problems. Most of the software maintenance, including bugfixes, version updates and general control and monitoring functions, are performed remotely from Cornell.

The Fuzzball includes event-logging features which record exception events in a log file suitable for periodic retrieval using the standard Internet file-transfer protocols FTP and TFTP. In addition, a special UDP-based server has been implemented [15] so that cumulative statistics can be gathered from all nodes with minimum impact on ongoing service. At present, statistics are gathered on an hourly basis from every node and incorporated in a data base suitable for later analysis. About nine months of history data are now available in the data base, which is used to produce periodic management reports with performance statistics similar to those shown in this report.

A great deal of additional information is available from the Backbone nodes, the Unix-resident routing agent (gated) and various other sources. This information, which is available via remote login (TCP/TELNET), includes the contents of various routing tables, the state of system resources such as the buffer pool, state variables for the various protocols in operation and so forth. An interesting sidelight is that the time-synchronization function, which requires precise measurement of network delays and logical-clock offsets, serves as a delicate indicator of network stability. If the network becomes congested or routing loops form, the delays and offsets usually become unstable and are readily noticed by an experienced operator. In fact, the precision of the system is so exquisite that the temperature of the machine room can be estimated from the drift-compensation term of the logical-clock corrections computed by each node.

The growth in traffic carried by the Backbone over the nine-month period since October 1986 is clearly apparent in Figure 2, which shows the number of packets delivered to the destination Ethernets per week. Figure 3 shows the preemption rate (percentage of packets preempted per packet delivered) per week. The dramatic reduction in preemption rate at about week 27 was due to an expansion in buffer space together with adjustments to system parameters such as retransmission

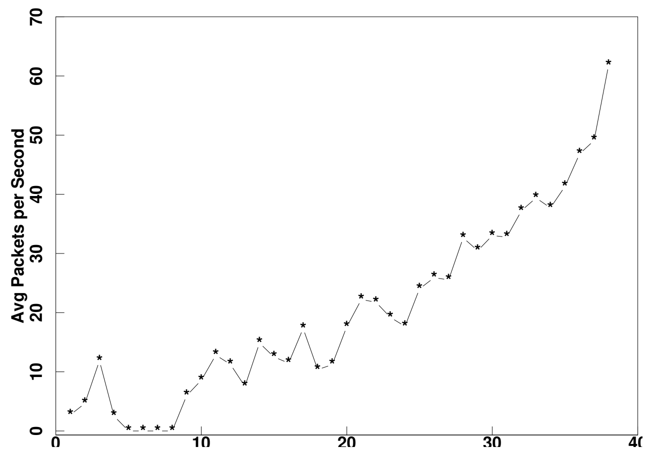

Figure 3. Packets Delivered, Averaged by Week

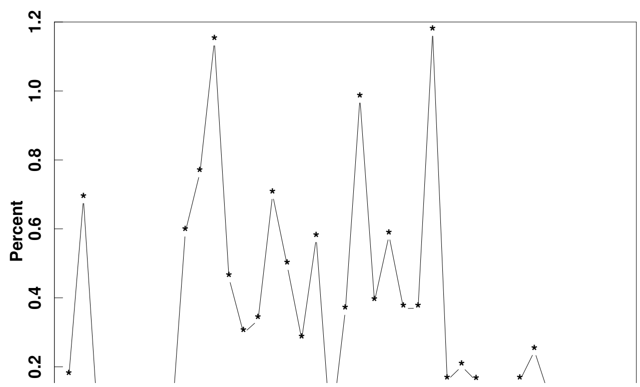

Figure 4. Percentage of Packets Dropped, by Week

limits. A second dramatic drop in preemption rate at about week 33 was due to the introduction of the new preemption policy described previously. The effectiveness of this policy is evident by the fact that, during a period in which the packets delivered rose by 50 percent, there was a six-fold decrease in the number of packets preempted.

8. Future Plans

Today the Backbone is an integral part of the Internet system; in fact, over one-fourth of all Internet networks are reachable via this network. As evident from the previous section, the aggregate traffic carried by the Backbone is currently approaching that of the ARPANET/MILNET gateways, which are overloaded and soon to be replaced. Moreover, although the preemption policy is working well and suggests that additional node and trunk capacity remains, the alarming rate of growth indicates the current Backbone configuration will be inevitably overwhelmed within a short time.

Current plans are to augment Backbone service by the addition of high-speed nodes and additional trunking capacity. While no decision has been made on the node configuration or trunk speeds, it is likely that T1 speeds (1.544 Mbps) and new high-speed packet switches will become available in 1988. The migration path from the existing Backbone to a new one using this technology is now under review.

It is anticipated that the current interim network-management structure will be replaced by a permanent one. The National Science Foundation has solicited a Cooperative Agreement for "Project Solicitation for Management and Operation of the NSFNET Backbone Network," with award expected by November of 1987. The awardee will have primary responsibility for designing, installing and operating upgrades to the Backbone. The emerging OSI protocols will become a very important factor for the future evolution of the NSFNET. The migration of NSFNET to an OSI connectionless-mode environment will become imperative as the OSI protocols mature and implementations become widely available. A most likely migration strategy will be to support both Internet IP and OSI connectionless-mode (CNLS) protocols in all NSFNET gateways, including the Backbone. This will allow hosts supporting either or both protocol suites to coexist in the same internetwork. Changes in subnet protocols and addressing mechanisms necessary to implement this strategy are already in progress. In addition, it is likely that application-level gateways may be installed at strategic points in order to support essential services such as mail during the migration period.

9. Acknowledgments

Doug Elias of Cornell University and Mike Minnich of the University of Delaware provided invaluable assistance in the generation, analysis and presentation of the performance data in this report.

10. References

1. Bertsekas, D., and R. Gallager. *Data Networks*. Prentice-Hall, Englewood Cliffs, NJ, 1987.

2. Braden, R. Requirements for Internet gateways. DARPA Network Working Group Report RFC-1009, USC Information Sciences Institute, June 1987.

3. Chao, J. Weekly Throughput Summary for the BBN LSI-11 Gateways. Report distributed via electronic mail by Bolt Beranek Newman.

4. Defense Communications Agency. *DDN Protocol Handbook*. NIC-50004, NIC-50005, NIC-50006, (three volumes), SRI International, December 1985.

5. Fedor, M. Gated - network routing daemon. Unix manual description pages, Cornell University, 1987.

6. Hedrick, C. Routing Information Protocol. DARPA Network Working Group Report (number to be assigned), Rutgers University, July 1987.

7. Jennings, D.M., L.H. Landweber, I.H. Fuchs, D.J. Farber and W.R. Adrion. Computer networks for scientists. *Science 231* (28 February 1986), 943-950.

8. Leiner, B., J. Postel, R. Cole and D. Mills. The DARPA Internet protocol suite. *Proc. INFOCOM 85* (Washington DC, March 1985). Also in: *IEEE Communications Magazine* (March 1985).

9. Mills, D.L. An overview of the Distributed Computer Network. *Proc. AFIPS 1976 NCC* (New York, NY, June 1976).

10. Mills, D.L. DCN local-network protocols. DARPA Network Working Group Report RFC-891, M/A-COM Linkabit, December 1983.

11. Mills, D.L. Exterior Gateway Protocol formal specification. DARPA Network Working Group Report RFC-904, M/A-COM Linkabit, April 1984.

12. Mills, D.L. Algorithms for synchronizing network clocks. DARPA Network Working Group Report RFC-957, M/A-COM Linkabit, September 1985.

13. Mills, D.L. Network Time Protocol (NTP). DARPA Network Working Group Report RFC-958, M/A-COM Linkabit, September 1985.

14. Mills, D.L. Autonomous confederations. DARPA Network Working Group Report RFC-975, M/A-COM Linkabit, February 1986.

15. Mills, D.L. Statistics server. DARPA Network Working Group Report RFC-996. University of Delaware, February 1987.

16. Mills, D.L. Metric Transformations. Memorandum distributed to the Internet Activities Board, Internet Architecture Task Force and Internet Engineering Task Force, June 1987.

17. Nagle, J. On packet switches with infinite storage. DARPA Network Working Group Report RFC-970, Ford Aerospace, December 1985.

18. Plummer, D. An Ethernet address resolution protocol. DARPA Network Working Group Report RFC-826, Symbolics, September 1982.

19. Postel, J. User datagram protocol. DARPA Network Working Group Report RFC-768, USC Information Sciences Institute, August 1980.

20. Postel, J. Internet Protocol. DARPA Network Working Group Report RFC-791, USC Information Sciences Institute, September 1981.

21. Postel, J. Internet control message protocol. DARPA Network Working Group Report RFC-792, USC Information Sciences Institute, September 1981.

22. Postel, J. Transmission control protocol. DARPA Network Working Group Report RFC-793, USC Information Sciences Institute, September 1981.