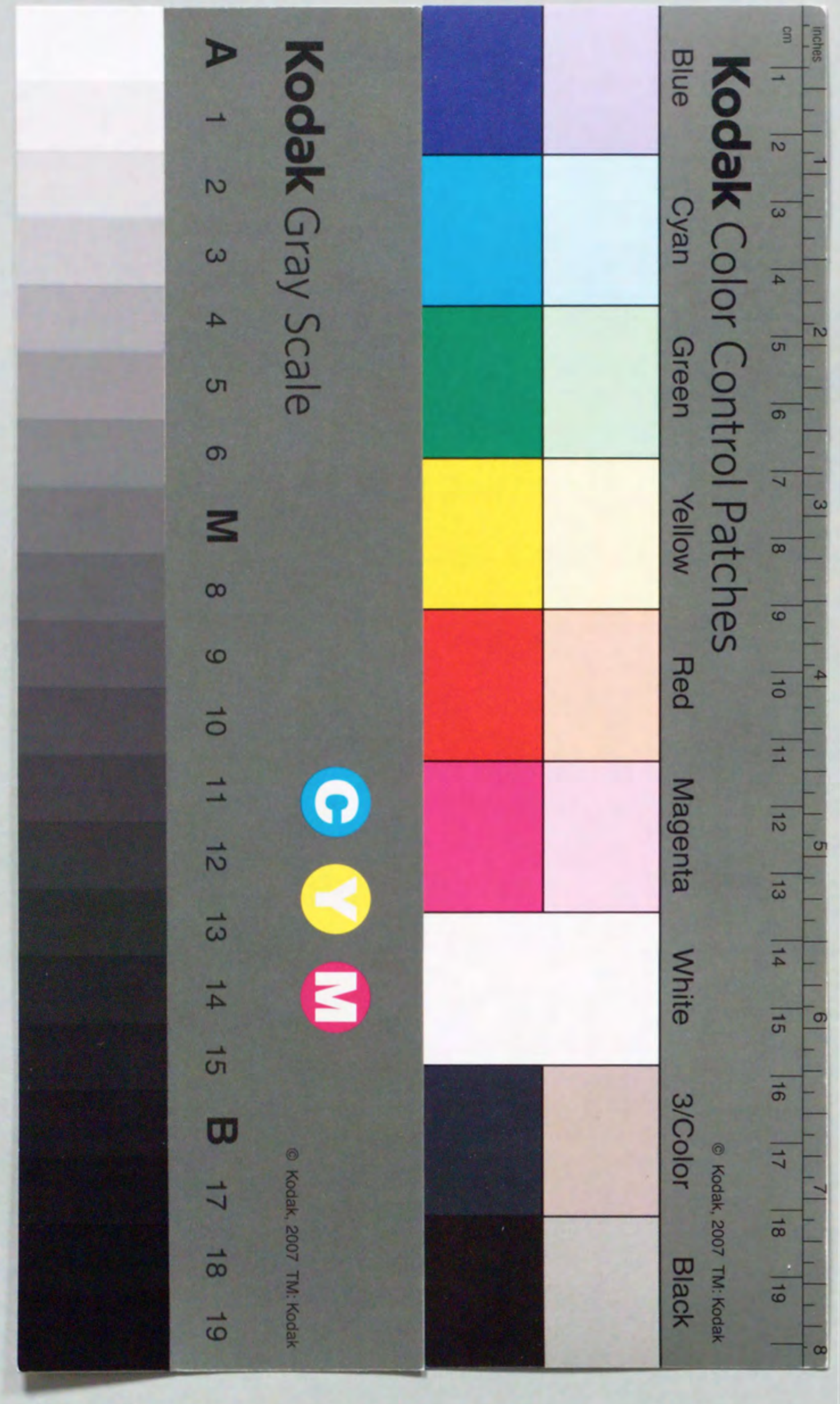

# Doctoral Thesis

## Studies on Scalability Improvement Techniques for the Internet with Access Manager and Performance Manager

Youki Kadobayashi

April 17, 1997

Department of Informatics and Mathematical Science
Osaka University

①

# Doctoral Thesis

## Studies on Scalability Improvement Techniques for the Internet with Access Manager and Performance Manager

by

Youki Kadobayashi

Submitted to the Department of Informatics and Mathematical Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at

Osaka University

April 1997

Thesis Supervisor: Hideo Miyahara, Professor

# Abstract

Protocols and software systems that constitute computer networks have been facing the scaling and management problems. Scalability can be limited by improperly designed protocols or inappropriate use of protocols. Protocols and systems that do not implement automatic configuration and diagnosis lead to management overhead. These two problems may result in computer networks that cannot scale, or large computer networks that cannot be managed.

In this dissertation we examine the problems in scalability and manageability of computer networks that are based on TCP/IP protocol suite. A number of solutions are proposed to these problems. The essence of the ideas is to deal with transient failure in some part of networks by transparently introducing new functions into networks.

A new approach for improving scalability and manageability of information access was introduced, which is based on the concept of "access manager". Its implementation inside distributed file system was presented. Experimental results show that the proposed approach eliminates many of the scaling and management problems that existing information access systems face.

A general method for automatic detection of network performance problems was proposed. A data collection and processing architecture to monitor network performance is described. Experiments with a prototype system show that the proposed techniques can reduce the management overhead in large computer networks.

# Acknowledgments

I would like to thank my supervisor, Professor Hideo Miyahara, for his help and encouragement during the course of my research. I am indebted to Associate Professor Suguru Yamaguchi, for giving me all the great opportunities to work with a number of gifted people, and for sharing his far-sighted vision with us.

I would also like to thank the various members at the Nara Institute of Science and Technology for their assistance and advice during my stay at NAIST: thanks are due to Professor Heiichi Yamamoto, Kazumasa Kobayashi, Tomomitsu Baba and Kiyohiko Okayama.

I have benefited a great deal from the stimulating discussions with knowledgeable and insightful people at the WIDE project: Professor Jun Murai (Keio), Susumu Sano (NEC), Atsushi Onoe (Sony), Osamu Nakamura (Keio), Yoichi Shinoda (JAIST), Hiroyuki Kusumoto (Keio), Akira Kato (University of Tokyo) and Hideki Sunahara (NAIST).

My life as a researcher did not begin without many people who have been at the Miyahara Lab. My special thanks to those people whom I could share creative atmosphere with.

Finally, I would like to thank my parents and my wife Rieko for their love and support during my study, and for making life more pleasurable.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

The integration of computer and communication technologies has had a profound impact on the way people communicate. Computer networks have been the essential infrastructure in scientific and commercial fields, and their interconnection is ever accelerating to form a global communication infrastructure, the Internet. In the future, many of the day-to-day activities may be closely tied to such an ubiquitous computer network.

This dissertation examines two of the most fundamental problems in large computer networks: scalability and manageability. Previous work succeeded to solve these problems to some extent, at certain layers of the OSI reference model. The work described in this dissertation attempt to address these problems from the perspective of the application layer.

Scalability is an essential property to construct computer networks of arbitrary size. Since scalability of a computer network is closely tied to its underlying architectural components, improperly designed protocols or inappropriate use of protocols may lead to bottlenecks that limit the computer network to scale up beyond certain threshold.

Manageability is a key to maintain and operate large computer networks. Forthcoming installation of computer networks to consumer equipments demand simple, failure-resilient, and secure networks. Technology developments are expected to enable administrators manage computer networks in scalable way. Protocols and systems that do not configure and diagnose themselves will need large number of administrators, hence limiting the scalability of the network.

In the rest of this chapter, we look at the development of internetworking technology and the research on scalability and manageability of internetworks from both software and hardware perspective.

## 1.1 Evolution of Internetworking Technology

### 1.1.1 Early History

Internetworking technology, which signifies sheer aggregation of technologies that make it possible to interconnect different kind of networking devices and software suite, has its roots in a single computer network that are intended to support research activities. Sometime around 1967, a project was initiated to connect timesharing computer systems through communication systems at leading universities and research institutions under the sponsorship of Advanced Research Project Agency[1, 2]. In 1969, Bolt Beranek and Newman, Inc. developed the first packet switch called Interface Message Processor, or IMP. Following the success of interconnecting several nodes with IMP, the ARPANET grew rapidly, connecting more than 100 hosts in 1977.

The growth of the ARPANET and the emergence of disparate local area networking devices stimulated technology development for interconnecting networks with different transmission schemes. ARPA then initiated internetworking project to establish a model and a set of rules which will allow data networks of diverse hardware technologies to be interconnected[3]. The internetworking project laid the basis of today's Internet, including IP (Internet Protocol)[4] and TCP (Transmission Control Protocol)[5]. IP specification defined standard procedures to translate packets between networking devices with different restrictions (e.g., maximum transmission unit). IP also introduced the concept of global address space, that makes the Internet look like single large cloud surrounding users. TCP was carefully designed so that virtually any number of logical connections between communication peers can be multiplexed on the Internet.

Due to its global addressing and flat communication model, the Internet required additional developments that complement its model, including protocols for: 1) autonomously routing packets without centralized control, 2) isolating subnetwork failure from the rest of the Internet, 3) gracefully degrading performance without suddenly falling into non-operational state, 4) globally naming hosts on the Internet without single point of failure, and 5) implementing security framework without relying on single cryptographic technology. Many universities and research institutions have contributed to these technology developments and standardization activities, which resulted in key routing protocols like BGP[6] and OSPF[7], network management protocols like ICMP[8] and SNMP[9], and name resolution protocols like DNS[10] and LDAP[11]. These standard protocols, combined with TCP and IP protocols, are collectively called TCP/IP protocol suite.

### 1.1.2 Recent Trends in Internetworking Software

Before 1990's, TCP/IP protocol suite was available only on Unix, VMS and other high-end operating systems. Its limited availability confined internetworking technology into higher education and research areas. However, as soon as the TCP/IP protocol suite was made readily available on commodity platforms like Windows and Macintosh, the technology transfer to mass market had begun quickly. The emergence of WWW browsers on commodity platforms further accelerated mass deployment of internetworking technology.

Throughout the technology developments in the TCP/IP protocol suite, scalability, simplicity and manageability issues have been the major concern, since the Internet has been expanding at the pace nobody has initially intended, and since the technological spectrum of the Internet is very wide. Although worldwide Internet scenario was already envisioned and talked about in the 1980's, there still remains many scalability and manageability issues today.

### Networking Software for Mass Deployment

Delivering advanced software and hardware technologies to commodity platforms require significant amount of engineering, and often paradigm shift, since system designers cannot expect users to understand detailed internal workings of the system. Networking software and devices are no exception.

Current status of the system, either failing or normal, must be presented to users in easily understandable form, such as status indicators. More detailed information on failure can be encoded in diagnostic code, which remote operators can use to perform simple diagnostics over phone line. Several dozens of configuration options and diagnostic commands, usually available in laboratory environments, are not necessary in commodity products that are sold in bulk.

The ability to simplify administrative and troubleshooting tasks, collectively called manageability in this dissertation, is an important property that any networking software or hardware must possess.

Networking software that are intended to be used in the Internet environment must cope with the growth of client installations and the increase in round trip delay. The system must not suddenly stop working when the number of clients exceeds certain threshold, or when several hundred milliseconds are added to the round trip time. There have been several commercial protocols and commercial networking software that behave exactly like this, due to their flawed design.

The TCP/IP protocol suite is one of few exceptions that have been carefully designed with scalability as the primary goal, so that although the performance degrades as the scale of the Internet becomes larger, the Internet itself does not stop functioning. It is important to retain such scalability not only in TCP/IP protocol suite but also in internetworking software built on top of it.

### Scalability in Internetworking Software

Scalability of internetworking software can be evaluated with the following criteria:

- the maximum number of clients one server can serve within acceptable response time,
- the maximum number of clients that can be multiplexed onto 1Mbps line with acceptable response time,
- the maximum of allowable round trip time between server and client for acceptable response time,
- and the maximum number of clients entire system can serve within acceptable response time. There may be several servers in the system.

Also, since internetworking usually involves cooperation between multiple administrative domains, the software must allow cooperation of clients and servers that have different naming authority or numbering authority, different access restriction policy, or different security policy.

### Manageability in Internetworking Software

In this dissertation, we quantify manageability of internetworking software with the following criteria:

- Number of users that can be served by single system administrator within tolerable turn around time (U/A ratio). Here, we define turn around time of a system administration job as the time required to satisfy user request for the internetworking software since the request was issued.
- Number of systems running the internetworking software that one system administrator can manage (S/A ratio). If the total time required to configure, monitor and troubleshoot these systems exceed the total time that can be allocated to the specific internetworking software, we can say that these systems are not manageable by one administrator.

- Number of times that the internetworking software can work around with random combination of failure in lower layers of the TCP/IP protocol stack and in the cooperating machines. This criterion is usually called failure resilience.

- Number of times that configuration changes in the cooperating machines can be made transparent to users. Systems with this property are said to be network transparent.

- Number of times that security incidents in other administrative domains can be prevented from propagating into the administrative domain that has been running the internetworking software.

Although it is possible to further elaborate evaluation criteria for manageability, only these five criteria are defined in this dissertation for simplicity.

### 1.1.3 Recent Trends in Internetworking Devices

The hardware technologies that are designed for internetworking purposes must also provide for management issues and scaling issues in the Internet. Among these issues, the manageability issue has been half solved by implementing SNMP, a protocol for network management. By implementing SNMP on every routers, switches, bridges and on general-purpose computers, administrators can monitor status of each hardware component from central location without any human intervention. Since hardware support is provided for network management, it is the software's role to improve manageability of internetworking devices.

The rapidly growing population of users, together with the emergence of traffic intensive applications for the masses, called for networking industry's focus on scalability issues in internetworking devices.

**Scalability in Internetworking Devices**

Since networking devices are getting faster every year, the Internet could be made scalable just by interconnecting fast networking devices with routers, where IP packets are processed and forwarded in software. However, the performance bottleneck of today's internetworks are routers.

IP packet processing, when implemented in software, can easily saturate bus backplane of router hardware, since only simple processing are necessary at routers. Since bus backplane is shared among all high speed links attached to the router, the total bandwidth requirements for the bus can easily exceed the maximum bandwidth that can be achieved by the bus (e.g., 2Gbps).

In response to this scalability problem of router hardware, many hardware devices have been developed to improve packet processing speed. Many of these devices process most of IP packets in hardware, thus offloading software overhead in IP packet processing.

Two key technologies used in these devices are multistage switch and associative memory. Multistage switches, such as butterfly switches and banyan switches, convey IP packets directly from input port to output port, thus eliminating the shared bus bottleneck. Content addressable memory, which is a kind of associative memory, makes it possible to translate IP address into switch circuit number without software overhead.

These two technologies, widely used in today's commercial routers and switches, provide scalability in aggregate bandwidth.

However, once bus bottleneck is eliminated, high speed links such as FDDI or ATM will be the next bottleneck. Emerging high speed links, such as HIPPI-64 or OC-48 ATM may be able to saturate today's improved routers and switches. After all, the scalability problem persists somewhere in the Internet.

The key to maximizing scalability of internetworking devices lies in appropriate techniques for detecting performance bottleneck. Since the Internet consists of diverse networking technologies and can be configured in arbitrary topology, sophisticated techniques must be developed and established to automatically detect performance bottleneck.

## 1.2 Research on the Scalability of Networking Software and Hardware

Scalability of networking software and hardware has never acquired so much interest as it acquires today, both from the research community of internetworking technology and from the industry. Since most of the networked applications did not demand much bandwidth, their scalability was not a serious problem. However, the advent of traffic-intensive applications like WWW browsers and video-conferencing tools changed the situation completely, giving acute impact on research activities on internetworking.

### 1.2.1 Scalability of Client-Server Model

Networking software that are based on client-server model have most limited scalability, since the processor cycles, I/O bandwidth and the network bandwidth of servers can be easily consumed by a large number of client requests that attempt to transfer data in bulk.

Several studies on caching in the client-server model have been done to explore its scalability. Numerous early work on this subject have been done within the context of distributed file system, since it was the only traffic-intensive networking software based on the client-server model, before recent inception of the web.

A trace-driven simulation study by Blaze and Alonso[12] reported that relatively small client-side cache on disk storage (ranging from 64Kbytes to 512Kbytes) can reduce the traffic between client and server by 60% to 90%. Danzig et al.[13] observed that 30–50% of the Internet backbone traffic were redundant transfer of same files via FTP protocol.

Possibility of alternative architectures have been explored to break the inherent bottlenecks in the client-server model, namely, I/O, network, and processing bottlenecks at server. Hartman and Ousterhout prototyped a distributed file system that stripes file

over multiple servers[14]. Anderson, Patterson et al. proposed a new approach called server-less network file system, which attempts to eliminate these bottlenecks[15].

Recent progress in computer architecture may totally eliminate the three bottlenecks in terms of hardware. Research on scalable multiprocessor architectures, most notably cc-NUMA, made it possible to break the memory bottleneck, which was the major obstacle for realizing scalable systems in traditional SMP architecture. Combined with multiple I/O channels, processors and scalable networking devices like ATM, advanced parallel systems may be able to realize scalable server.

### 1.2.2 Network Scalability and Network Performance

Many research activities and associated standardization efforts are ongoing to explore, establish and implement universal methods for measuring scalability of various networking devices. Scott Bradner of Harvard University established the Network Devices Test Lab, a facility for measuring performance of networking devices under various circumstances. The techniques established by the Bradner's group is being standardized at the benchmark methodology working group of the IETF. Murayama developed DBS[16], a distributed benchmark tool, that makes it possible to measure performance of networking devices under realistic workloads, from the perspective of internet transport layer.

It is also important to assure the performance of operational internetworks by making various measurements in operational environments and collecting statistics therein. The operational statistics working group of IETF attempts to standardize measurement methods. Some of the internet exchange points perform measurements of route stability and end-to-end performance[17].

## 1.3 Organization of Dissertation

The rest of the dissertation is organized as follows:

Chapter two introduces a new approach for achieving scalability and manageability in information access software, based on the concept of access manager. The design and experimental implementation based on this approach, the cluster server, is presented.

Chapter three discusses service availability problem and its solution with access manager approach. Limitations of current networking software implementations are described, followed by the description of implementation framework that overcomes their limitations.

Chapter four proposes a new method to identify performance bottlenecks in the Internet. A data collection and processing architecture for monitoring network performance is presented, that makes it possible to identify failing or saturated network devices without manager's attendance.

Chapter five summarizes the contributions of this research and outlines the future work.

# Chapter 2

# Access Managers for Scalable Information Access in the Internet

This chapter introduces a new approach for information access based on the concept of access manager. The design and experimental implementation based on this approach, the cluster server, is presented. Experimental results from its deployment at several universities and research institutions suggest that the access manager contributes to scalability and manageability in the information access application.

## 2.1 Introduction

As has been discussed in chapter one, the user spectrum of the Internet is no longer limited to people within higher education and research community. However, existing tools and services assume user's knowledge about network configuration and service policy, which makes it difficult for casual users to access wide variety of services in the Internet. Among these services, this work focuses on problems that have been occurring and will occur when casual users attempt to access information services. Other services such as remote computation services and remote printing services are not considered here, since that kind of services are not universally available to everyone in the Internet.

Existing information retrieval clients require users to specify logical location of information servers when accessing remote information in the Internet. There are many potential problems that will arise when casual users use this kind of software. More specifically, there are potential security problems when users mistakenly execute programs with Trojan horse built in. The possibility of encountering this problem will be greater than before, since casual users will not be able to tell servers carefully replicated by malicious organizations from original servers built by some university or company. There is another problem of increased communication cost that might occur if users without network topology information mistakenly access servers far apart. Also, casual users may not be able to adapt to diverse access methods that are necessary to access various information in the Internet.

In this chapter, the concept called "access manager" is introduced. Access manager makes various complexities invisible from users and presents simple "resources". By advertising itself as server to clients and advertising itself as client to servers, access manager can be transparently introduced between client and server, making it possible to introduce additional functions into simple client-server software.

The following functions can be implemented in access manager: cope with diverse access methods and present simple access interface to user, choose optimal servers based on either subnetwork trouble information, network topology information, cost or administrative policies, minimizes potential security risks, and cache retrieved information to reduce redundant network traffic. These functions would contribute to the scalability and manageability of the Internet. By consolidating these complex functions into one particular component, it is made possible to improve information access architecture, while at the same time keeping clients and servers simple.

In this chapter, the design and implementation of access manager inside distributed file system is described. The applicability and the effectiveness of access manager is verified through the implementation and experimental results.

## 2.2 Problems of Information Access in the Internet

Existing tools require users to understand network topology, scheduled network outages, availability of equivalent servers, credibility of each server, usage policies of each server, and detailed service pricing. Casual users may have difficulty in understanding these detailed operational information, thereby experiencing problems described below:

1. *Security issue.* Casual users may not be able to tell credible organizations from malicious organizations. They may not be able to tell "infected" or "contaminated" program from original program that are not tampered. Since distributed document indexing service is widely used in these days, it is really easy to replicate some useful programs from the Internet, embed Trojan horse in them and advertise changed programs at the distributed document indexing service. Being Internet so large, some cases have been reported so far and at least one security advisory was issued for a specific case[18].

2. *Cost issue.* Existing information access software require users to specify logical location of servers like hostname or IP address. Such an approach rules out any opportunity to facilitate alternative servers that provide same contents. These servers, often called mirror servers, are intended to minimize response time from nearby users and offload access to particular server. By requiring users to directly specify one particular server, existing software are introducing possibility that casual users incorrectly specify servers that are far apart from user's location. Suboptimal selection of server without appropriate knowledge incur increase in access cost. If data transmission at the backbone are charged by distance or by traffic volume, the impact of suboptimal selection would be much higher.

3. *Failure issue.* Even when alternative servers are prepared for failure resiliency, casual users may not know the availability of such alternative servers. It is not realistic to assume that users keep track of alternative server's location and its contents.

4. *Diverse interface issue.* Since various operating systems and server software are deployed in the Internet, there can be many access methods to information stored at information servers. Casual users may have difficulties in understanding diverse access methods.

5. *Policy issues.* Although some of the information servers are made accessible from the Internet, their usage policy may vary. Some information servers are intended to serve local community, such as university students or subscribers to particular commercial network, and access from far distant location might be discouraged unless other servers become totally unavailable. However, it has been difficult to enforce this kind of policy on existing software.

If some system could guide casual users to right places, user environments would be more secure, information access would be cheaper, and response time would be shorter than that of today's Internet, thereby contributing to scalability of the Internet. If some system could guide casual users to alternative servers before accessing servers in failing subnetwork, network failure would be invisible to users while making the Internet more manageable. By making scheduled network outages transparent to users, system managers and network operators can reconfigure their systems and networks for improvement at any time.

Information access software may work without solving these problems, but these problems make information access cumbersome from user's point of view and unmanageable from manager's point of view. If these problems persist, the worst scenario is that every user must become Internet expert. Internet, ultimately intended to be a mere communication medium, must solve these problems and present simple interface to users.

## 2.3 Access Manager

This work attempts to address issues described in the previous section by introducing access manager into existing simple information access software. Access manager can

make the following improvements:

- *Hide operational complexities from users.* Since access manager can select appropriate server among other alternatives, any users can facilitate sparsely distributed equivalent servers without understanding network topology and server usage policy.

- *Maximize the availability of service.* Since users can be transparently guided to alternative servers upon subnetwork failure near the primary server, the availability of service perceived by users can be much improved.

- *Minimize differences in access method.* Slight differences in access method, such as differences of database name or filename, can be made transparent at access manager.

- *Reduce latency and access cost.* By selecting nearest server and by caching retrieved information, access manager can contribute to faster, cheaper access and reduced network traffic.

- *Reduce security risk.* Since maliciously replicated and disguised servers can be avoided a priori by applying access control at the access manager, administrators can reduce the security risk of accidentally installing Trojan horse at casual user's computers.

It's possible to achieve the same improvements by implementing similar functions inside client software, but the client-side approach is limited in that all applications must be upgraded to change its behavior like server selection algorithm, failure detection techniques and caching algorithm. Such an assumption, that all client software in entire computers can be upgraded, is not realistic in today's Internet, since the Internet comprises of multiple administrative domains and deployed operating systems are truly diverse. Implementing similar functions in server side does not make sense, since unavailable server cannot instruct users to go to alternative servers.

One important property that are required to achieve improvements described above are location transparency. Location transparency means that location changes of particular resource are invisible from user's point of view. This work selected NFS for experimental implementation, since NFS filenames do not contain either hostname or IP address, hence achieving location transparency.

## 2.4 Design of an Access Manager

This work designed and implemented an access manager inside distributed file system. Many of the existing applications built on top of distributed file system can benefit from improvements that are made possible by access manager. The implementation described in this chapter also makes it possible to extend small-scaled distributed file systems to much larger scale.

There are similar system called AFS, that extends file system to networks larger than local area networks such as campus networks and further supports their interconnection by grouping all servers in each organization into an aggregate, which is called "cell" in the AFS terminology. However, AFS lacks location transparency for inter-cell access in that it requires users to specify hostnames. Lacking location transparency, AFS does not hide operational complexities from users and may not be able to maximize the availability of service. Although AFS does not meet our goal, AFS is much more richer in its functionality compared to this implementation because it is a full-featured commercial product.

In this work, an access manager for distributed file system is designed and a prototype for evaluation purposes are implemented. The prototype, called cluster server, has been implemented as an NFS server, thereby providing uniform interface for use in both local area network and wide area network.

### 2.4.1 Accessing from Client

Many of the file systems used in local area networks adopt a common file system interface called vnode interface[19]. Vnode interface comprises of primitive operations like open, read, write and close. Since many of the applications are built on top of file systems, that are in turn based on vnode interface, access manager for distributed file systems must be able to interface with vnode. The cluster server was designed to be compliant with NFS protocol, which complies to vnode interface, thereby making the system widely applicable. In fact, the cluster server is designed to be visible simply as another server of NFS protocol.

### 2.4.2 Accessing File Servers

Although it is possible to communicate with distant file servers with NFS protocol, it makes access latency much worse than other protocols, since NFS cannot cope with large round trip delay of packets. Since round trip delay of packets are usually ten to hundred times larger in wide area networks compared with local area networks, adopting NFS for wide area access is not realistic. The cluster server attempts to address this problem by adopting NFS for access from local clients, and by using other protocols (e.g., FTP) for access to distant file servers. By implementing FTP for wide area access, most of the public file access can benefit from caching, server selection and other features that are made available at the cluster server.

The design of the cluster server is based on the idea of protocol translation engine, as depicted in Figure 2-1. A protocol translation engine comprises of: stub modules for diverse protocols that have different delay, bandwidth or processing requirements, cache modules for accommodating with disparate protocol granularity, and translation modules for converting commands and data representation between these protocols. The implementation of protocol translation between NFS and FTP is described in Section 2.5.

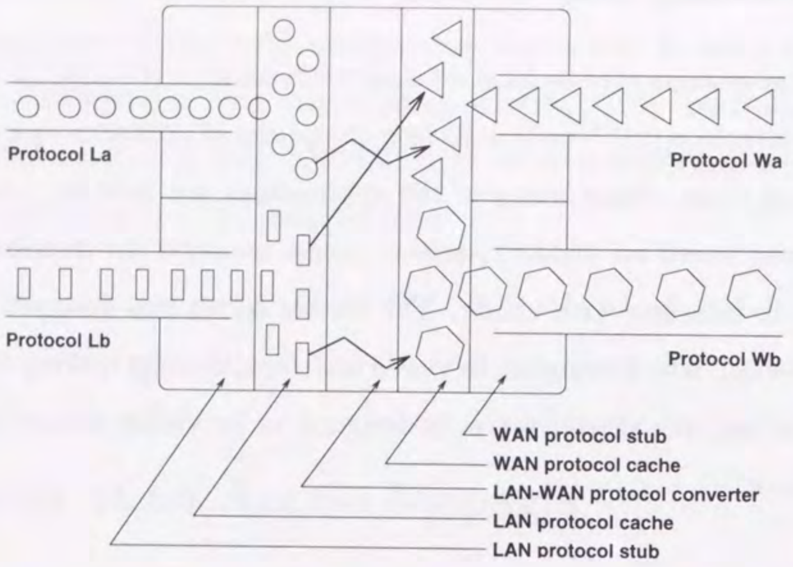

Figure 2-1: A protocol translation engine.

### 2.4.3 File Naming

Cluster server achieves location transparency by encapsulating location information and other administrative information into volume configuration file. A volume in cluster server represents a collection of equivalent part of file system tree in equivalent servers. Volume configuration file is designed to contain various administrative information, such as location of alternative servers, administrative contact person for each server, names of the top directory within each server where equivalent information are made available, slight differences in access methods such as username and password, and policy information such that access to particular server is discouraged in the office hours.

Volume configuration file makes it possible to transparently switch to alternative servers upon subnetwork failure and to select optimal server based on network topology or on accumulated statistics on server response time.

There are several approaches for sharing name space among users. One approach, implemented in Prospero[20], is that each user constructs his own name space on shared resources and makes the name space sharable to other users. Another approach is that users share single name space. Since it is difficult to comprehend non-uniform name space built on top of shared name space, a single shared name space is chosen for the cluster server.

The name space provided by the cluster server comprises of name spaces presented by each volume and single glue of these name space at the top directory. This kind of name space glue is very similar to mount operation provided in conventional Unix file systems.

### 2.4.4 Security

Volume configuration files are intended to be administered by the cluster server's administrator, where it is possible to prohibit access to carefully disguised servers that are built for malicious intent. Although techniques are not established to identify servers with carefully tampered programs or documents, it may be technically feasible to identify them, given that there are many established algorithms to verify integrity of remote files. For example, there are established techniques and their implementations to combine secure hash functions [21]. Also, since digital signatures are commonly used in the Internet, efforts are under way to build software framework that automates verification of software integrity[22]. Data integrity services, with help from widely available certificate authorities, can actively and automatically identify tampered software, enabling automatic generation of the list of malicious or infected servers. Once such list of servers is made available, administrator can prohibit access to them by instructing access manager not to do so.

### 2.4.5 File Sharing

Distributed file system must define the behavior for resolution of read/write contention. In particular, files must not be destroyed by simultaneous writes by multiple clients. Also, integrity of accessed file must be ensured by locking file or by making separate copy for simultaneous modification by other clients. This kind of operations that ensure consistency of file contents are called consistency control operations.

Although consistency control operations are effective in local area networks, its extension to wide area networks may incur significant increase in access latency, hence impairing scalability of the service. Consistency control between clients require at least three phases of interaction between clients or between clients and servers, which results in unacceptably large response time in situations where clients and servers are far apart.

In cluster server, it was decided not to implement write operation and consistency control, since this work focuses on issues in information access. Changes to files are assumed to be made directly at the file server, where locking and other consistency control functions can be implemented in much simpler way. The cluster server invalidates cache when changes are made at the file server.

### 2.4.6 Caching

Cluster server caches file and directory information in the disk storage, aiming at shorter access time and reduced redundant traffic. Alternative methods such as caching files at clients are considered but not adopted since clients can be disk-less workstations or mobile computers with limited disk capacity. Memory caching of entire files and directories are not attempted, since memory caching may not be able to provide sufficient room, potentially causing thrashing. Previous trace-driven simulation study by Blaze et al.[12] revealed that relatively small client-side cache on disk storage (ranging from 64Kbytes to 512Kbytes) can reduce network traffic by 60% to 90%.

### 2.4.7 Cache Consistency

Since the cluster server does not support write operation, the only operation required to achieve cache consistency is cache invalidation. When change notification process is running at the file server, cluster server can simply invalidate cached counterpart of notified files or directories. Also, the cluster server perform heuristic consistency control using FTP protocol. Cluter server compares current directory information with that of

previous access remaining in the cache, discarding files if file size changed and discarding directories if subdirectories are removed.

### 2.4.8 Load Balancing

Although traffic and processing load on file servers can be reduced by performing caching at the cluster server, the cluster server in turn can be overloaded by overwhelming access by a number of clients. In this case, it would be effective to divide clients into several clusters, and then serving each cluster with one cluster server. The division of cluster might have impact on cache hit rate, and therefore one may want to define parent server for all cluster servers. However, previous simulation study by Muntz et al.[23] using AFS shows that implementing caching mechanism in hierarchical fashion does not improve cache hit rate so much; they report that only 7% improvement are observed in the case of 80Mbytes of cache. For this reason, hierarchical caching was not adopted in the cluster server.

Traffic and processing load that are concentrated on the cluster server may not be the big problem in current Internet, where wide area links are much slower than local area links. The processing bandwidth required on the cluster server is nearly equal to that of NFS server.

### 2.4.9 Server Selection

The cluster server can select optimal server based on network topology and geographical information stored in the domain name server. Network topology information consists of the name of network organization, geographical location within the network, and ISO country code. These information are stored as a resource record called GTR record in the domain name system, as shown in Figure 2-2. Administrators can express server selection policy based on network organization, geographical location and country code. The priority configuration file, shown in Figure 2-3, describes server selection policy. Cluster server sorts list of servers in each volume based on the priority configuration file.

```
0.0.221.163.in-addr.arpa.   TXT   "GTR=Nara-WIDE-JP"
0.0.1.133.in-addr.arpa.     PTR   osakau-net.osaka-u.ac.jp.
                            TXT   "GTR=Osaka-SINET-JP"
```

Figure 2-2: Representation of network topology in the domain name system.

```
region  Tokyo-WIDE-JP
region  Tokyo-IIJ-JP
region  Osaka-WIDE-JP
region  -WIDE-JP
region  -JP
```

Figure 2-3: A sample priority configuration file.

## 2.5 Implementation

### 2.5.1 Implementation Overview

The cluster server is implemented in the user space, not in the kernel space, so that the software can be ported to many of the operating systems based on Unix. Since the cluster server works as a server speaking NFS protocol, it must coexist with NFS server running in the kernel. This coexistence was accomplished by binding the cluster server to UDP port different from kernel NFS server. Performance may also be a potential problem since protocols running in the user space usually involves extra buffer copying overhead compared with servers in the kernel. However, several benchmarks revealed that running the server in user space does not exhibit major performance penalty, since disk I/O seems to be the performance bottleneck in the system.

Binding of the cluster server and the client machines are accomplished with modified version of mount program, since most of the ordinary mount programs available on many operating systems are not capable of specifying the UDP port number. The mount program is the only component that needs to be replaced. After the completion of mount system call, directories beneath the mount point can be accessed in the same way

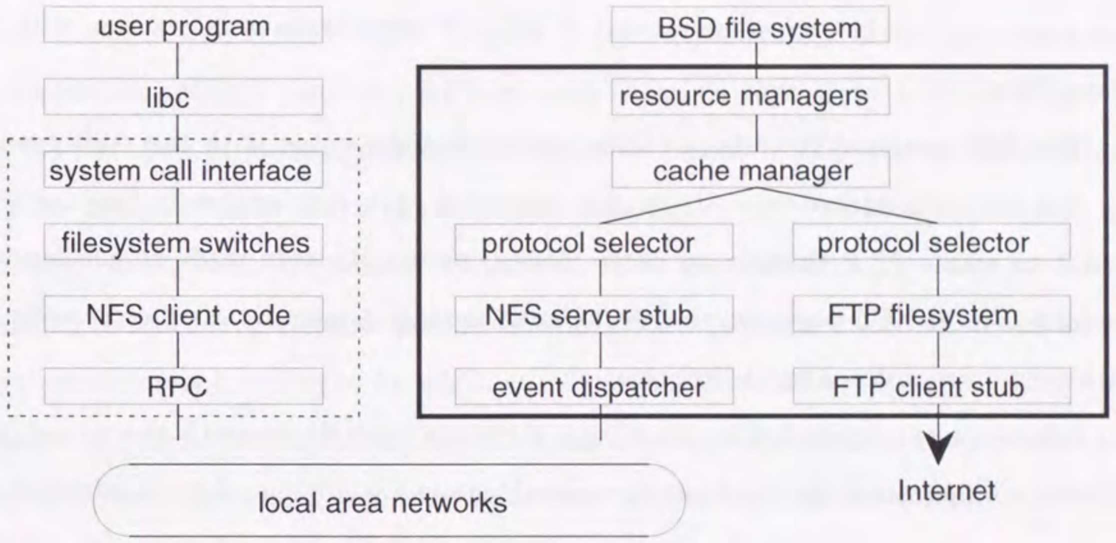

Figure 2-4: Internal structure of the cluster server.

as NFS-mounted directories.

The cluster server performs protocol translation between NFS and FTP, because file servers on the Internet do not provide information service via NFS and because clients can access information in the Internet transparently by extending NFS to the Internet. Translation of these two protocols are not feasible task however, since NFS is connectionless, stateless protocol built on top of UDP and FTP is connection-oriented, stateful protocol built on top of TCP.

This implementation experimented with translation of protocols that are based on completely different models. Although the implementation was not easy and the resulting code is far from elegant, the experiment has shown that such translation can be done with careful design of intermediate data representation and with some tradeoffs.

The implementation of the cluster server is based on non-preemptive threads, where each interaction with clients or servers is implemented in a thread. It required completely different coding style, such as event-driven representation of stateful protocols. In particular, it was difficult to implement FTP with threads, since the socket programming interface does not take the case of asynchronous communication into account. The programming experience gained through this implementation suggests that the socket programming interface must be revised to support asynchronous interaction with TCP connections.

The NFS protocol module of cluster server was generated with the stub generator for Sun's remote procedure call. In the generated stub routines, code fragments were added to create FTP threads on cache misses. Attempts were made to accelerate the performance of NFS protocol module by maintaining in-memory copy of recently used directory information and file information.

Volumes are represented to users as a directory directly beneath the mount point. Volume configuration file, that has the same filenames as volume, contain administrative information that are intended to implement security, policy and optimal server selection. Server selections are accomplished by preprocessing volume configuration files, sorting the list of servers described according to GTR records, with perl scripts that are capable of accessing domain name system.

Files and directories are cached in a dedicated disk partition for later reuse. Files are kept intact in the cache, since storage allocations are performed by Unix file system. Directories are not created exactly like the server; they are converted to unique identifiers within the belonging volume and directories that have identifiers as their name are created. The mapping of name space to flat identifiers was motivated by its potential to adapt to alternative file servers having different name space. This capability is not fully exploited in this implementation however.

Cluster server implements a technique called connection caching, which attempts to accelerate access to file servers by maintaining unused but active connections for a short period of time. By reusing connections already open, it is possible to avoid several steps of FTP login sequences and three-way handshakes that are required for establishing TCP connection.

Figure 2-4 shows a detailed overview of the internals of cluster server, whose components are described in the following sections.

### 2.5.2 User-level NFS Server

Our implementation of user-level NFS server is based on that of `unfsd`[24] and `amd` version 5.3[25]. Our implementation is different from `unfsd` in that `unfsd` cannot coexist with the kernel NFS server, while ours can, although some additional commands were necessary on each client machine. Cluster server is also different from `amd` in that our implementation interprets and handles incoming NFS requests itself, while `amd` handles them uninterpreted, trying to be reliable NFS request forwarder. However, these differences do not mean functional superiority of our implementation since these predecessors serve their own purpose, which is different from that of the cluster server.

We have developed a separate mount protocol named CS protocol from scratch, since the mount protocol described in the NFS protocol specification has been already used by `rpc.mountd` and therefore mount requests cannot be caught by or forwarded to user-level NFS server. Due to this incompatibility, we had to develop a separate `mount` and `umount` command specifically for the cluster server. By specifying the port number of the cluster server in `addr` argument of the `mount` system call, operating system kernel associates particular mount point with the cluster server.

User-level NFS server establishes relationship with clients as follows. First, the server creates an UDP socket for RPC and associates the service dispatch procedure with the *(program name, version number)* pair, so that the NFS stub function will be called upon remote procedure call to the RPC address *(NFS_PROGRAM, NFS_VERSION)*. Next, the server associates another service dispatch procedure with the RPC address *(CS_PROGRAM, CS_VERSION)* so that mount requests can be forwarded to this process.

A client first contacts the server via CS protocol to obtain port number of the NFS service socket. Next, the client issues another RPC to obtain the file handle of the mount point, which is then used to construct arguments for `mount` system call. The kernel then associates the specified mount point with *(socket address, file handle)* pair so that subsequent access to filesystem beneath it will be forwarded to the cluster server.

```
typedef struct wf_fh {
    long     world_id;
    long     vol_id;
    union {
        struct {
            long     u_dir_id;
            long     u_file_id;
        } u_file;
        struct {
            long     u_parent_id;
            long     u_child_id;
        } u_dir;
    } u;
    char     padding[16];
} wf_fh;
```

Figure 2-5: The file handle data structure.

### 2.5.3 File Handle

Since the file handle is a primary means used to perform operations on files and directories on the NFS server, it must be guaranteed to be unique. In other words, an individual file in a filesystem must be uniquely distinguished with a file handle. Unlike other implementations, it was difficult task to assure uniqueness of the file handle since we could not rely on inode numbers or other host-specific ID allocation mechanism; transparent server-switch would be otherwise impossible. Therefore, we had to develop a mechanism in the cluster server, that maps pathnames into file handles.

Our file handle consists of four elements: magic number, volume ID, directory ID, and file ID, each occupying four bytes (see Figure 2-5). An unique volume ID is assigned by a perl script when the volume is first introduced into the system. Cluster server allocates a directory ID from volume's ID-allocation bitmap, when a new sub-directory is seen. File ID is generated from filename with 32-bit CRC algorithm, that obviates the necessity of name-to-ID mapping table, while maintaining consistency and minimizing the likelihood of file ID collision.

### 2.5.4 NFS-FTP Gateway

All network events are detected using select system call and then handled at the dispatcher. UDP and TCP events are handled on a first-come first-serve, non-preemptive basis, and RPC events are immediately dispatched to stub functions.

Stub functions for NFS and CS protocols are generated using rpcgen[26]. NFS stub functions constructs internal data structure from the file handle, then forwards incoming requests to the protocol selector, that selects a protocol switch from available ones. The selection is based on the volume ID and possibly other policy information found on the volume data structure. The protocol selector then dispatches the request to the corresponding function pointed by the protocol switch.

If volume ID is zero, the protocol selector dispatches the request to the root filesystem, where other volumes are joined together and form an independent namespace. Current implementation provides a flat namespace where all volumes are glued, thereby forming a directory directly beneath the mount point of cluster server, where all volumes are visible as sub-directories.

If the volume ID in the file handle is not zero and the FTP filesystem is selected, cache management file system is invoked, where availability of cached directories or files are checked. In case of cache miss, corresponding delegate function is called via the protocol selector, that causes actual file transfer to take place.

FTP filesystem implements three delegate functions: ftp_mount for the lookup fault on the root filesystem, ftp_readdir_miss for readdir fault, and ftp_read_miss for read fault. Basically, ftp_mount and ftp_readdir_miss does the same thing: open new FTP connection if not available, request a directory listing, receive and parse the listing, and convert it into internal data structure. The only difference of these two are that ftp_mount must interface with the root filesystem. Ftp_read_miss is different from these functions in that it must accept a new data connection to initiate the file transfer. Figure 2-6 illustrates the control flow in the cluster server.

The FTP delegate functions request the timeout scheduler to call the timeout function so that the client kernel will receive an error status for nfs_read and nfs_readdir if one second elapses before the completion of requested transfer, thus avoiding the kernel thread to block on a slow file transfer. After the file or directory transfer completes, these requests succeed.

### 2.5.5 Exception Handlers

Two functions are defined to send and receive ICMP packets. Icmp_send constructs an ICMP packet and then sends it to the specified address. Icmp_recv receives an ICMP packet and interprets it as follows. If the packet is an ICMP *echo reply* and it is a response to the previously sent ICMP *echo* packet, RTT is computed from the time stamp and it is reflected to the server statistics. If the packet is either ICMP *network unreachable* or *host unreachable*, and if it contains an original IP header with correct source and destination, the corresponding server is marked unreachable and all threads accessing the server is notified.

Upon receiving TERM or INT signal, all threads are notified of the service shutdown, and all connections and log files are closed. After all connections are assured to be closed, the daemon terminates.

### 2.5.6 Scheduling Facilities

The network event dispatcher serves both TCP and UDP socket events on a first-come first-serve, non-preemptive basis. It also contains several lines of code for signal masking, RPC sockets, thread scheduling and timeout scheduling, since the function embraces the main loop. This code was derived from amd version 5.3.

The thread scheduling facility is also based on the amd implementation. Threads can be blocked and resumed with sched_task and wakeup functions without requiring an individual stack for each thread, since the scheduler implementation is based on the idea of continuation, as in [27].

Timeout scheduler maintains timeout events sorted by the time of call, and provides three major functions to manipulate and access this data structure: timeout_set, timeout_clear and softclock. Timeout_set inserts the requested timeout event to the appropriate place in the list. Timeout_clear cancels previously requested timeout. Softclock removes timeouts older than or equal to the current time and invokes callback function with corresponding closure, returning the number of seconds to the next timeout. Softclock is called from the network event dispatcher to compute how long it can block on the select system call.

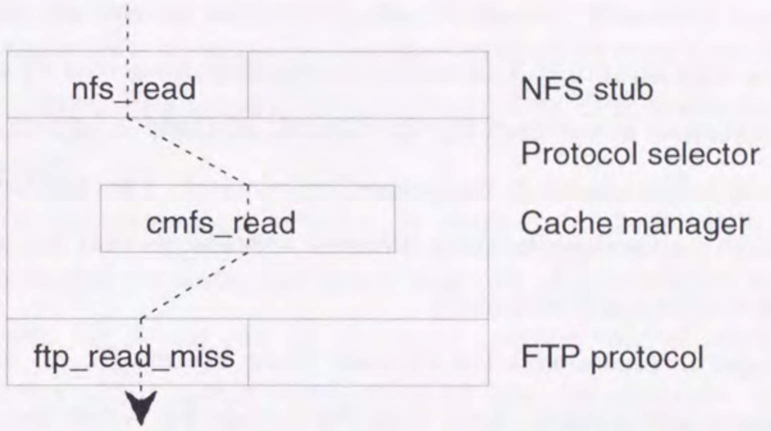

Figure 2-6: Control flow of NFS read request in the cluster server.

The status of each thread is scanned every minute to detect frozen threads. If a thread is flagged as erroneous, give_up entry in the thread data structure is invoked. If a thread is marked as risky, idle field in the thread data structure is incremented. If the idle field exceeds *time to live* constant, the connection used by the thread is marked *timed out* and then the give_up entry is invoked. The idle field is cleared every time the thread is activated, hence active threads can be distinguished from inactive or frozen threads.

### 2.5.7 Resource Managers

Cluster server implements fundamental resource managers for connection, server, volume, directory and file.

Connection manager detects inactive connections with idle field in the connection data structure — as is done with threads — along with the storage allocation/deallocation functions for the data structure. Connection manager scans across all connections every minute; if a connection is not used by any thread, idle field is incremented. If the idle field exceeds *time to live* constant, the connection is closed. This pool of idle connections enable us to share connections between different threads, so that file access requests to the same server can be satisfied quickly.

Server manager interacts with the Domain Name System[28, 29] and manages each server's error state information, along with the storage for server data structures. If a server is flagged as unavailable for some reason, server manager requests timeout scheduler so that the status will be cleared 12 hours later and then its reachability is tested again with ICMP *echo*.

Volume manager allocates/deallocates connection, directory ID and the volume data structure. Volume manager limits the number of connections allocated to one volume, since the total number of file descriptor is limited by the operating system. Directory ID is allocated using an ID allocation bitmap, which is maintained for each volume so that uniqueness of a directory ID is guaranteed within a volume.

Directory manager maintains directory information on both memory and disk. On-memory cache is expired when its idle field increments to the *time to live* constant, as is done in connection manager. When searching a directory and the on-memory cache misses, directory information is fetched from the disk. If the time stamp of the fetched directory is older than two days before, the information is marked as obsolete so that filesystems can update it. The cache manager filesystem calls the delegate function if the directory is flagged as obsolete.

File manager provides memory allocation functions, lookup functions and cache management functions for file data structures.

### 2.5.8 Debug and Administrative Facilities

Cluster server implements a logging facility independent from `syslog`, since `syslog` cannot separate the output of a specific daemon from others, that makes inspection of debug output cumbersome.

Cluster server incorporates a continuous event-trace mechanism, that collects all nfs_read and nfs_readdir requests, and the completion of file transfer so that various aspects of wide area file access can be measured: arrival rate of requests, spectrum of users, predictability of file access and cache miss rate, for example. Trace data is kept in memory, where each event occupies 44 bytes in the current implementation. They are flushed onto the local disk every minute. Stored event occupy 36 bytes for each. The traces are sent to our statistics program at Osaka University via e-mail once per day, while at the same time locally maintained data is erased to conserve disk space. The extra mechanism does not impair the performance of the file sharing service provided by the cluster server, since operations required for each event are few pointer traversal and register/memory assignments.

The current implementation of cluster server accepts connection to TCP port 8002 via `telnet` so that internal data structure can be inspected while the file transfer and other network activities are present, which cannot be easily achieved with existing debuggers. For example, internal data structure for network connection, thread, directory, file, and file handle can be examined from a terminal using `telnet`.

## 2.6 Evaluation

Through the implementation of cluster server, it has been verified that most of the improvements can be achieved by access manager as initially intended. The implementation of cluster server demonstrated that it is possible to hide operational complexities from users by selecting appropriate servers. The availability of service has improved at experiments performed at several participating organizations. This experiment is described in the next chapter. Protocol translation and simple name space translation performed at the cluster server demonstrated that it is possible to minimize difference in access method at access manager. Caching and topology-based server selection made access latency smaller at several lab-based measurements. Security issues are not addressed in this implementation.

## 2.7 Discussion

Although the access manager approach turned out to be effective, there are several limitations and issues that need to be addressed. Since most of client access goes through access manager, access manager can be the performance bottleneck depending on the application. For example, it is not realistic in today's hardware architectures to relay very high speed information flow at the application layer, because shared bus is not fast enough to relay bulk traffic that demands very large bandwidth. Also, access manager can be the single point of failure if clients cannot adapt to failure. However, the impact of these problems may vary depending on future developments on computer architecture. For instance, commodity products are emerging that realize very high speed data transfer using memory crossbar, eliminating the bottleneck of shared bus. Server hardware is getting more robust with the advent of hot-pluggable processor modules and advanced multiprocessor operating systems that tolerate failure in subsystems.

Someone might prefer to re-design distributed filesystem for large scale internetworks from ground up. There have been several research efforts along this direction, for example Echo file system[30]. While that would be more straight approach for realizing globally scalable filesystem, their deployment overhead would be much larger. We took different approach, since we intended to explore the possibility of transparently introducing additional functions into existing services with minimal overhead and minimal changes.

The rapid growth of the web diminished the impact of this implementation, since the web defined HTTP as the new standardized protocol for accessing distant information, making FTP and NFS obsolete. Due to the location-dependent model deployed by the

web, most of the techniques proposed in this chapter, particularly those techniques for improving availability and access latency, cannot be applied to the web. However, some of the techniques proposed in this chapter are applicable to the web. For example, efforts are under way to establish caching proxy for the web[31].

Recently introduced programming languages, such as Java[32] and Obliq[33] that support migration of code fragments across internetworks, can be the alternative way for incorporating intelligence into internetworking software. If these mobile code can be automatically downloaded to every client, and if extension of client internetworking software can be accomplished without compromising security, internetworking software can be made more malleable.

## 2.8 Summary

A new approach for information access based on the concept of access manager is presented. Access manager can be introduced transparently into internetworking software, by advertising itself as a client to servers and as a server to clients. Access manager is intended to hide operational complexities of accessing geographically dispersed resources from users. It can be used to address diverse issues such as security, cost, policy, service availability, and interface uniformity. Through its implementation and experiments in the Internet, its ability to solve these problems has been demonstrated.

# Chapter 3

# Improving Availability of Information Access in the Internet

This chapter examines the implementation method of networking software for the Internet. The failure at the transport layer was measured and analyzed, which turned out to be unacceptably high. In traditional software implementation method, the availability of services cannot be improved further, since failure at the transport layer is made visible to users. In this chapter, we tackle this specific problem with access manager. Through the deployment of cluster server, we verified improvements to the availability of information access. The implementation method for increasing availability is described.

## 3.1 Introduction

In the gateway model of the Internet architecture[34], each layer of the protocol stack implements distinct reliability functions. The protocol stack as a whole provides reliability of end-to-end data stream, by combining these reliability functions.

TCP/IP protocol suite, based on the gateway model, provides reliability of end-to-end data stream to some extent, but it does not guarantee end-to-end availability of service.

Our measurement in some organizations in the Internet revealed that the probability of failure in TCP connection establishment ranges from 5% to 20%. Since TCP is the most commonly used transport protocol by many networked applications, and since most of networked applications barely rely on reliability functions of the transport layer, failure at the transport layer will be directly visible to users. As a result, users will not be able to connect to servers, having difficulty in accomplishing their own task. For example, users will not be able to access necessary information, or users will not be able to send mail.

Since the availability of service is unacceptably low in the Internet, the Internet needs substantial development for its deployment as a new communication infrastructure. A mechanism is needed to improve the availability of service. Furthermore, such mechanism should be applicable to as many applications as possible.

In previous chapter, we introduced access manager, where various functions can be added to existing client-server applications in a transparent manner. Access manager can be used to improve the availability of service, by having alternative servers on distant locations and by switching to one of them upon failure.

In this chapter, we describe new implementation method of networking software that facilitates the use of alternative servers. We intended to improve availability by establishing techniques for constructing alternative servers and by establishing mechanism for facilitating the use of alternative servers. Using this implementation method, server selection and server switching functions are implemented inside cluster server, the access manager built on top of distributed file system. Through the operational experiment in the Internet for two and a half months, we verified that availability of service can be improved by access manager.

### 3.1.1 Failure Modes of the Internet

TCP is the important transport protocol in the TCP/IP protocol suite, on which many standard application-layer protocols like TELNET (remote login) and SMTP (mail delivery) rely on.

Table 3.1: TCP connection failure rates.

| Organization | Failure rate (%) | Days measured | Connections total |
|---|---|---|---|
| NAIST | 4.9 | 52 | 1,817,927 |
| UEC | 8.6 | 11 | 558,071 |
| U. of Tokyo | 19.1 | 86 | 28,488 |
| Company 1 | 5.6 | 27 | 228,282 |
| Company 2 | 14.6 | 141 | 108,879 |

```
p_t = (tcps_drops + tcps_conndrops
      + tcps_timeoutdrop + tcps_keepdrops)
      / tcps_closed
```

Figure 3-1: The formula for calculating $p_t$.

The failure probability of the Internet is considered equal to the failure probability of TCP, assuming that networked applications does not fail. In other words, the availability of service from user's viewpoint equals with the reliability of TCP connections.

In operational networks, there are high probability of failure in TCP connection establishment and abrupt disconnection of TCP connections. We made several measurements on errors occurring at TCP in several organizations including Nara Institute of Science and Technology, the University of Electro-Communications, and the University of Tokyo. Since destination hosts are limited in the case of TELNET and FTP, we have chosen WWW, which connects to various computers in the Internet. Measurements were made on the WWW proxy server using `netstat -s` (Table 3.1).

Measurement results show that the sum of TCP connection drops and failure in establishing TCP connections ranges from 5% to 20%. This probability of failure at the transport layer, $p_t$, can be computed by dividing total number of dropped connections by total number of closed connections (including drops). The formula for calculating $p_t$ is shown in Figure 3-1. The variable in Figure 3-1 can be obtained by `struct tcpstat` inside the operating system kernel[35]. The meanings of each variable is shown in Table 3.2.

Such an unacceptably high connection failure rate stems from the way the Internet

Table 3.2: Meanings of each variable in struct tcpstat.

| Name of variable | Meanings |
|---|---|
| tcps_drops | number of connections dropped after SYN |
| tcps_conndrops | number of connections dropped before SYN |
| tcps_timeoutdrop | number of connections dropped due to retransmission timeout |
| tcps_keepdrops | number of connections dropped due to keepalive timeout |
| tcps_closed | closed connections (including drops) |

is constructed. The path between any two hosts usually consists of many routers, implying that the reliability of transport connections between the two hosts depends on the reliability of each router in the path. If any of the router in the path stops functioning, then the path will be disconnected unless all routers in alternative path is functioning properly. This kind of failure can be reduced to some extent by making the network "tightly connected" in the sense of graph theory.

The another reason for high failure rate is the architectural nature of routing protocols. Routing protocols used in today's Internet assume that advertised routes are equally credible. Since they don't have any provision for human errors or bugs in routing protocol implementation, new and illegal routes can be easily installed into each router, making some part of the Internet unreachable from the other. This kind of problem is unavoidable in today's Internet unless new failure-resilient routing protocols are standardized and widely deployed.

Since network layer does not guarantee end-to-end reachability, TCP cannot guarantee connection establishment. TCP may suffer from connection drops when destination host does not respond or when it is restarted.

## 3.2 Utilizing Alternative Servers

As has been discussed in previous section, the most widely used transport protocol in the Internet does not guarantee enough availability. Since many networked applications

are built on top of TCP, the availability of service is unacceptably low. We attempt to apply access manager approach to improve the availability of service.

Our method assume that two or more servers with the same content are available at geographically distant location. If alternative servers were unavailable for specific service, its availability cannot be improved. We also assume that these alternative servers have nearly equivalent contents, updating each other if necessary. Since alternative servers are widely available in information access services, this research attempts to improve availability of information access services.

### 3.2.1 Utilizing Alternative Servers with Access Manager

Access manager resides between servers and clients, relaying their requests and responses at the application layer. Access manager keeps track of the status of each server so that properly functioning server can be selected.

Access manager can improve the availability of service in the following situation: troubles in the network component that are close to server, or troubles occurring inside the server. Access manager may not be able to improve availability when some network component fails near the client side, since client beneath the failing component cannot communicate with access manager. This problem can be eliminated by placing access manager near the client.

To minimize the effect of failure, the following requirements must be satisfied:

1. Provide the same contents at the original server and at the alternative servers by replicating contents from the original server,
2. Minimize server down-time by detecting failing components as early as possible,
3. Avoid selecting the inaccessible server by sharing fault information with other connections. Fault information can be obtained through interaction with the failing server.
4. Maximize server availability by detecting recovery from failure as early as possible.

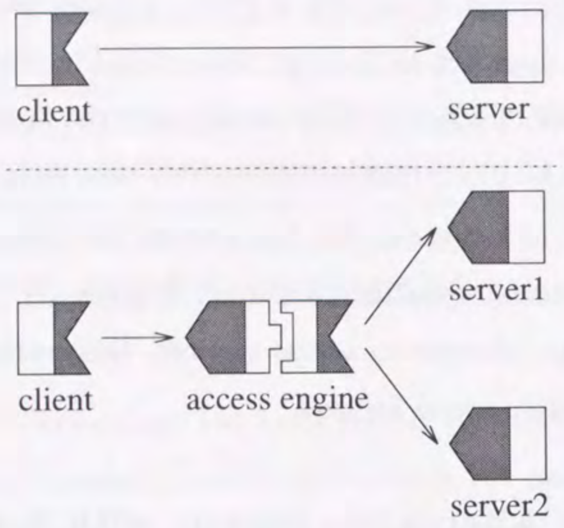

Figure 3-2: Transparent introduction of access engine.

Since conventional implementation method of networking software cannot satisfy these requirements, a new method for implementing networking software is needed to realize an access manager that can improve availability. Its detail is described in Section 3.4.

### 3.2.2 Integration of Access Manager into Existing Services

By making the service interface of access manager the same as that of existing server, it became possible to incorporate access manager into existing networked applications without making major changes (Figure 3-2).

For example, the cluster server provides NFS as access interface to clients, and it selects properly functioning server when it relays initial access request to server. Since server switching function is implemented inside cluster server, it is effective on many of existing services that are built on top of filesystem, such as FTP server and SMB server. By stacking existing networked applications on top of the service interface provided by the access manager, it becomes possible to improve the availability of service on existing networked applications.

```
iij.ad.jp.  MX  10   ns.iij.ad.jp.
iij.ad.jp.  MX  100  ns1.iij.ad.jp.
```

Figure 3-3: The use of alternative servers in sendmail.

### 3.2.3 Describing Alternative Servers

The technique of alternative server is commonly used in sendmail[36], the de-facto standard mail delivery system for the Internet. For each mail domain, primary mail server and one or more secondary mail servers can be defined, and preference for each mail server can be given. These information are stored in the name server for the Internet, the Domain Name System[10, 28, 29]. For example, records shown in Figure 3-3 describe that two mail servers are defined for `iij.ad.jp` domain, and the mail server named `ns.iij.ad.jp` has higher preference than `ns1.iij.ad.jp`.

We consider application of similar techniques to the information access service. In information access service, it is common to replicate a set of contents to improve average response time from various part of the Internet. Replicated contents are usually placed at distant sites. If alternative servers are defined for each file, the total amount of information that must be stored in the name server becomes unacceptably huge. Therefore, there must be some mechanism for reducing the amount of alternative server information.

### 3.2.4 Volumes for Describing Alternative Servers

We use volumes to briefly describe alternative servers for a set of files. Here, we define volume as a part of filesystem provided at the file server. Contents of volumes are replicated on one or more file servers in the Internet. The substance of volume can be described by the list of file servers that provide the same set of files and the path path names in each file server.

By describing alternative servers for a set of files, the total amount of alternative server information can be made small. The administrative overhead of maintaining volume is considered to be minimal, since volume information is common to all organizations connected to the Internet. In other words, it can be put under centralized administration, alleviating the need for each administrator to repeatedly configure each access manager.

## 3.3 Technical Challenges in Implementation Method

In conventional implementation method for networking software, only one thread is running inside one process, which connects to client or server. In other words, the entire process is dedicated to interaction with the communication peer. Any fault on the connection or on the network are detected using retransmission timer or keepalive timer in the operating system. Most of implementations terminate process when it encounter any fault, requiring users to restart the session.

There are following problems in this implementation method. First, users must explicitly reconnect to the same server, or switch to alternative server upon failure. Second, since the operating system tries to retransmit request for 15 minutes by default, the usability of the system may be considered quite bad. Third, there are no provision for sharing the fault information among processes that access the same server. Fourth, recovery from failure cannot be detected without explicit retry request from users.

Therefore, the conventional implementation method for networking software does not satisfy requirements described in Section 3.2.1. There is a need to establish new implementation methods for networking software that can solve these problems.

## 3.4 Implementation of Server Switching

The cluster server attempts to solve the problems that are inherent in conventional implementation method, by implementing networking software in a totally different method. The implementation method described in the following sections can satisfy requirements described in Section 3.2.1.

### 3.4.1 Event-driven Implementation

Since access manager relays client-server interactions at the application layer, it must be capable of handling application layer protocols, while at the same time sensing any symptom of failure. If the application layer protocol use TCP as its transport, failure must be detected while TCP packet processing is in progress.

To maximize server availability by early detection of failure, the connect and accept system calls must be processed in non-blocking fashion. The connect and accept system calls usually make the entire process block while the connection is being established. In this blocking mode of connection establishment, it is impossible detect failure without relying on the failure detection algorithm implemented inside the kernel.

The cluster server handle TCP connections in a event-driven fashion, using socket and ioctl system calls. A thread is associated with each TCP connection. All events are handled through one central event loop, for example: arrival of data on TCP connections, arrival of UDP packets, receipt of signals and expiry of timer. This event-driven structure makes it possible to detect failure while TCP connections are being established.

### 3.4.2 Fault Detection with ICMP

Failure in the Internet can be categorized as failure beneath network layer and failure in the application layer. They can be further categorized as transient failure and long-term failure.

Failure beneath the network layer can sometimes be identified by ICMP packets. RFC1812[37] requires routers to send an ICMP *destination unreachable* packet to the source address whenever the next hop to the destination of the processing packet has one or more troubles beneath the network layer. By collecting ICMP packet, cluster server is capable of detecting failure on the path to the server and is capable of switching to the alternative server. However, since the delivery of ICMP packets are not guaranteed at the network layer, cluster server cannot totally rely on ICMP for detecting troubles.

The cluster server is capable of distinguishing transient failure from long-term failure, by performing reachability test at the network layer after certain time has elapsed since the first observation of the trouble. The reachability test is the same as the standard ping program. If the cluster server receives response, it switches back from the alternative server to the original server, assuming that the trouble was transient one. If reachability test failed, it assumes that the problem will persist for a while. In this case, it schedules reachability test again after certain time.

### 3.4.3 Fault Detection with Timer

Whenever any trouble occur at the application layer, the cluster server detects them by monitoring activities on each connection. An activity timer is kept for each connection, which is cleared whenever the thread associated with the connection issues an access request or receives an response. If there were no response for two minutes, the connection is closed, assuming that some trouble has happened on the connection or on the server process. In the mean time, reachability test is performed at the network layer.

If a new connection is needed to relay an access request, and if the reachability to the server is verified, the cluster server attempts to establish connections without switching to alternative servers. If the reachability test failed, it switches to an alternative server.

By detecting failure without relying on TCP connection time-out, it is made possible to detect failure early, and to minimize the period of service unavailability.

### 3.4.4 Sharing Fault Information with Other Threads

To minimize the attempt to connect to inaccessible servers, fault information, obtained through ICMP and timer, is stored as flags in the data structures for servers and connections. The server data structure makes it possible to accumulate past failures and to share failure information among many connections to the same server.

## 3.5 Evaluation

In this section we evaluate the improvements made to the availability of service by statistics obtained through operation of the cluster server. Measurements were made for 74 days, since January 18, 1996 to March 30, 1996, at the Nara Institute of Science and Technology.

Attempts were made to establish 2284 FTP control connections during the measurement period. 5184 FTP data connections were established. Of the FTP control connections, 119 connection establishments were denied at the application layer for a number of reasons: FTP server was improperly configured, or the number of active connections exceeded allowable maximum. 8 control connections could not be established because the server does not respond at the transport layer. Of FTP data connections, 42 data connections were terminated during the data transfer. There were 15 failures on the name resolution due to the failure near the name server, making it impossible to obtain IP address from hostname. Of these troubles, the cluster server made 181 attempts to switch to alternative server. Among these, 5 attempts failed since all servers were inaccessible.

The probability of failure in this measurement is $p_t = 5/2284 \leq 0.0022$. In other words, the availability of service achieved by the cluster server is more than 99%. If alternative server was not used, the probability of failure would be $p_t = 186/2284$, or 92%.

We can conclude that the availability of service can be improved with access manager.

## 3.6 Discussion

If the availability of service can be improved without the access manager, the architecture of information access would be much simpler. However, if server selection was incorporated into the client, the client has to implement server selection algorithm and protocols for failure detection.

The drawback of this client-extension approach is that client software becomes complex, since many of the functions embodied at the cluster server must be implemented in it. For example, it is difficult to share the current status of each server by exchanging availability information across several subnetworks. It might be difficult to improve server selection algorithm by totally replacing the client software.

Access manager makes it possible to extend the information access architecture without making client software complex. Furthermore, access manager is well suited to common Internet subscriber model, where each Internet service provider needs to improve reliability of services and to improve the overall performance, while at the same time making the configuration of clients and servers simple. By sharing volumes among all access managers, the configuration tasks for using alternative servers are reduced to the minimum.

The techniques illustrated in this chapter assume that the resources can be replicated and that they can be handled in an aggregated fashion. The availability of such resources would be improved by the application of access manager approach.

However, if the contents provided by the service changes so fast that their replication to alternative servers generate huge amount of traffic, then the use of alternative servers may not be realistic.

Security issue is not considered in this approach. If confidential information are replicated, the risk of security incidents increases, since it may be difficult to secure all of the alternative servers. This risk can be reduced by establishing methods for securing servers and data transmission between servers.

## 3.7 Related Work

Other distributed file systems that implement some facility for using alternative servers are AFS from Transarc[38] and ServerGuard from Auspex[39]. OSF DCE/DFS is the same as AFS. In AFS, alternative servers cannot be placed across different cells (the AFS terminology for administrative domain). In other words, AFS cannot improve the

availability of service across wide area networks. ServerGuard aims at transparent failover to the other server upon failure on the active server. There are no provision for server switching when the failure occurred in the Internet.

Many software have been developed for performing replication from the primary server to mirrored servers, most notably `mirror`, `rdist`[40], and `sup`[41].

Several studies have been done to analyze the inherent instability in the Internet routing protocols. Chinoy[42] analyzed dynamics of routing information in the US backbone. He reports that 3% of connected organizations experience more than 10 transitions within 12 hours. IEPG (Internet Engineering and Planning Group) issued an advisory on stabilizing routes[43].

Many efforts are being done to stabilize routes. Route dampening algorithm[44] has been developed for suppressing advertisement of unstable routes in BGP. Tada[45] developed a system for detecting unstable routes, changes in interface status and changes made to router configuration.

Research efforts have been made to design new routing protocols that are more failure-resilient. Perlman[46] proposed a new routing protocol that can recover to the normal state within finite time period. In the new routing protocol, even if incorrect routes are advertised or misbehaving routers are installed into the network, the protocol can recover to the normal state after the source of the problem (malfunctioned routers or misconfigured routes) are removed from the internetwork.

## 3.8 Summary

This chapter first pointed out that most of networked applications are built on top of TCP, and that the reliability of TCP is less than 90% on average. As a result, the availability of service perceived by users are unacceptably low.

Access manager can improve the service availability by transparently interposing itself into clients and servers. However, existing implementation method for networking software does not satisfy technical requirements for solving this particular problem.

A new implementation method for networking software was developed for this purpose. This method makes it possible to detect failure in its early phase, to share failure information, and to recover from failure. The implementation of cluster server was described.

Through the operation of cluster server in the Internet, its ability to improve the availability of service in real-world environments has been verified.

# Chapter 4

# A Network Performance Management System for Maximizing the Scalability of Networking Devices

This chapter proposes commonly applicable techniques to identify performance bottlenecks in the Internet, regardless of underlying networking technologies being deployed. A data collection and processing architecture for monitoring network performance is presented, that makes it possible to identify failing or saturated networking devices without manager's attendance.

## 4.1 Introduction

The increasing demand for Internet connectivity resulted in its growth in scale, variety of networking devices, and in increased number of possible combinations of networking devices. These growth in number and added complexity in technology spectrum make it difficult to manage even small part of the Internet. Also, the emergence of new devices based on new datalink standards such as ATM and 100baseT makes management tasks more difficult. Typical problems observed in those enterprise or campus internetworks that deployed these advanced technologies are slow response from servers, abrupt disconnection, and servers that do not respond at all. Such problems are getting more clearly exposed in today's internetworks with the advent of performance-sensitive applications, like WWW and video-conferencing, that need to carry traffic across several subnetworks.

Problems that occur in internetworks can be categorized as connectivity problem and performance problem. Connectivity problems exhibit when packets cannot be successfully delivered to hosts or routers because of their failure. Performance problems occur when there are frequent bit errors due to failure in physical layer, or when many packets are dropped due to excess amount of traffic flowing into one particular device.

Most of existing network management systems have focused on the connectivity problem, aiming at early detection of critical failure in hosts, routers or other networking devices. These systems ensured packet reachability by periodically sending probe packets. By the deployment of network management systems, administrators can make sure that whole components of particular internetwork are functioning at the network layer.

Due to the proliferation of performance-sensitive applications, administrators of today's internetworks are required to solve performance problems. Specifically, administrators are required to ensure that internetworking software, including applications that interface to users and underlying subsystems that support applications, respond within acceptable delay. If any internetworking software does not achieve expected performance, administrators must improve its performance to meet the requirement by reconfiguring some part of the internetwork. For example, performance of a router can be improved either by adding memory cards, offloading traffic by moving some subnetworks to other routers, or by changing buffer allocation parameters. In this dissertation, these tasks of solving performance problems are collectively called performance management.

### 4.1.1 Performance Management in the Internet

Performance management is a difficult task to accomplish in the Internet for many reasons. Performance cannot be guaranteed in the Internet, since important performance characteristics such as communication throughput and response time are not guaranteed, unlike telephone networks. Since transport protocols hide most of troubles occurring beneath them, users are unaware of those troubles. However, users notice performance problems quickly, since transmission problems in the network layer or datalink layer usually require packet retransmission at the transport layer, resulting in the slower system response to users.

Because users cannot directly observe real problem in the network layer or datalink layer, the real reason of the performance problem is not reported to administrators. Typically, administrators get vague reports from users, such as "I cannot send mail" or "files are inaccessible", from which they must deduce the real reason of performance problem. Since there are several layers and several subsystems involved in end-to-end communication, it is difficult to identify which component is failing, misbehaving, misconfigured or saturated.

Users are unaware of troubles because of architectural nature of TCP/IP protocol suite. Many of the troubles that occur beneath transport layer or network layer are made invisible. Failure in subnetworks are made invisible by routing protocols, that automatically select alternative routes after detecting subnetwork failure. Some routing protocols, most notably OSPF, permit transitive routing loop. The Internet maintains its operational state even if some routers advertise incorrect routes. Moreover, other routers blindly accept these route advertisements, often forming routing loops. Many of these routing troubles are invisible to users because of the failure resilience of routing protocols. Also, when physical links are made unavailable for a short period of time, transport protocols attempt to maintain connections by retransmitting lost packets, that are visible to users as longer round trip delay or slower data transfer.

Users notice performance problems very quickly, primarily because of nonlinear performance characteristics of TCP. TCP, the most widely used windowing flow control protocol that many applications rely on, responds to packet losses very sensitively. The performance of TCP degrades quickly when two or more packets are dropped within the send window, since very long timeout for double packet loss stops send window from advancing.

The biggest problem in managing network performance is that performance problems reported by users do not help reveal the real source of problems in many cases. Administrators have difficulty in correlating phenomena described by users with possible source of problems. Also, administrators may fail to enumerate complete list of routers, switches, fibers or protocols that may be failing, misbehaving or misconfigured. This is not feasible task, since there are many components in typical internetworks.

In this chapter, attempts are made to establish common techniques for managing network performance that are independent of networking devices. Proposed techniques comprise of: periodical measurements of application performance in several combinations of subnetworks, periodical collection of performance and error characteristics at both datalink and network layers, and analysis of correlation between measured application performance and various characteristics. Since these measurement and analysis requires many steps, a system is needed to assist or automate the procedure.

A network performance management system called ipsh (internet performance shell), aimed at assisting the measurement and analysis tasks in proposed techniques, is designed and experimental implementation is being done. A procedural programming language has been designed for ipsh to describe measurement and analysis procedures briefly.

Ipsh is considered to be better than other approaches for performance management in the following aspects. First, ipsh can minimize the amount of collected data on large internetworks, since it is possible to briefly describe data reduction strategy such that error characteristics are not collected when performance observed at the application layer does not change since last measurement. Second, ipsh can probe the internetwork in detail when performance problems are detected. When application performance suddenly drops, performance and error characteristics can be collected for various combinations of

subnetworks to determine failing subnetwork. Third, language constructs like alias and function make it possible to tersely describe complex operations and parameters. Default behavior is defined so that optional parameters can be omitted for common expressions, that simplifies commands in many cases at interactive mode.

## 4.2 Network Management Technologies

We briefly look at existing network management technologies. Although most of them are designed to solve connectivity problems, some of them can be used as building blocks for network performance management.

SNMP is a simple and general purpose protocol for network management. SNMP provides common interface for monitoring devices like bridges or routers, and software like name server or mail server. Information that are commonly available from networking devices are defined in MIB-II[47]. The state information defined in MIB-II are limited to information that are independent of vendors or implementation methods. To satisfy these requirements, these information are often simplified, thus limiting its usefulness. Performance management in the real world therefore relies heavily on enterprise MIB, which contains more detailed information on performance and error status (Table 4.1). Enterprise MIB is defined by each vendor specifically for their products.

There are several commonly available software across many operating systems that have been used to diagnose problems in the Internet. These tools, named `ping`, `traceroute`, `telnet` and `ttcp`, are useful for testing end-to-end reachability and end-to-end performance. Because these tools can be used on most of the user terminals, and because these tools use the same protocol stack as users rely on, they are appropriate for evaluating performance improvements or degradation from user's perspective. Various information can be obtained through execution of these tools, as shown in Table 4.2. However, obtained information have not been fully exploited for the purpose of improving performance.

Network management software, such as HP OpenView and Sun NetManager, can perform many tasks, since they have many functions built in. These tools seem to be used

| Name of variable | Meanings |
|---|---|
| ifInDiscards | number of discarded packets |
| ifInErrors | number of error frames |
| ifInUnknownProtos | frames with unknown protocol type |

Datalink-layer errors that can be obtained with MIB-II

| Name of variable | Meanings |
|---|---|
| locIfInRunts | short frames |
| locIfInCRC | CRC errors |
| locIfInOverrun | interface overruns |
| locIfResets | interface resets |
| locIfCarTrans | carrier signal transitions |
| locIfCollisions | collisions |

Datalink-layer errors that can be obtained with cisco MIB

Table 4.1: Comparison of MIB-II and enterprise MIB.

| Command name | Information obtained |
|---|---|
| ping | round trip time (min, max and average)<br>variance of RTT<br>packet loss rate |
| traceroute | route to the destination<br>RTT to each router (average and variance) |
| telnet | response time of the server |
| ttcp | throughput of services (ftp, web etc.) |

Table 4.2: Information provided by common diagnostic tools.

for limited purposes, such as continuous display of backbone usage statistics using SNMP, and monitoring and graphical status display of routers, name servers and file servers. Reflecting its usage, these tools have many functions to continuously monitor critical components of internetworks. After all, these tools are inappropriate for evaluating performance from user's perspective, since they are primarily intended to monitor part of the Internet, putting focus on network connectivity. Possible usage of these tools were not fully investigated in this work.

It is helpful to optimize traffic flows by obtaining traffic matrix using traffic measurement tools like NNStat[48] and NeTraMet[49]. If redundant traffic flows can be identified, and if relocation of server or reconfiguration of network topology is possible, there are opportunity to reduce redundant traffic flows, which may improve network performance. However, the traffic load on internetworks are dependent on applications deployed in it; if newly introduced applications contribute to significant portion of traffic matrix, reconfiguration of network topology may not make sense. Also, it must be pointed out that traffic measurement tools cannot measure network performance from user's perspective.

## 4.3 Performance Management Procedures

Many existing networks have been using a heuristic technique, which detects the occurrence of problem just by collecting performance and error characteristics using SNMP. Such technique is described in detail by Leinwand[50]. Although this technique is applicable to networking devices whose characteristics are well understood (e.g., Ethernet), it cannot be applied directly to advanced networking devices like ATM and 100baseT, whose performance management procedures are not established. Recent introduction of fair queueing, rate queueing and resource reservation makes it more difficult to establish simple procedures for identifying the source of problem.

This section describes new techniques for managing network performance, that are independent of datalink technology and router hardware architecture.

Proposed techniques consist of following procedures. First, by executing applications between computers located in different subnetworks, accumulate application performance measurements for various combination of subnetworks. Next, compare the combination of subnetworks and the application performance measurements to identify one or more subnetworks where the source of performance problem may be present. For example, if measurement of file transfer throughput among several subnetworks revealed that performance degradation always occur for traffic going through subnetwork N, the performance problem can be considered to be somewhere inside N.

Next, the layer at which the performance problem is occurring must be determined by collecting various statistics at multiple layers. SNMP can be used to investigate traffic volume and error rates. If similar statistics were collected before at the same subnetwork, or if there are subnetworks with same configuration, they can be used for comparison with current status of the subnetwork under investigation, that may help identify specific layer the problem is occurring. For example, if there were surge in error rate at the network layer although error rate at the datalink layer did not change, the problem resides in network layer. Similar reasoning can be done for datalink and application layer cases.

Since these measurement and analysis tasks require many steps, it is desirable to automate some part of them by developing supporting software. Such systems are required to possess the following properties: ability to measure and accumulate error characteristics and performance characteristics for each layer, and the ability to perform statistical analysis on collected data, generate summarized reports, and visualize results.

Periodic measurements, analysis and report generation can help administrators to pinpoint the day the performance problem first exhibited. In some cases, the failing component can be detected simply by comparing records of network configuration changes with generated reports. For example, if a network performance problem has occurred immediately after adding an workstation to FDDI ring, the source of the problem may be either incorrect FDDI cabling, misbehaving network interface card of the workstation, or the burst traffic generated by the workstation.

## 4.4 A Network Performance Management System

Most of the existing MIB give status of specific component of a system, the number of discarded packets for example. In this respect, MIB gives us microscopic characteristics. In contrast, experienced network managers need macroscopic performance measures like those offered by `ping` and `traceroute`. When experienced network managers attempt to solve network performance problems, they first determine the path on which performance problems are observed, and then investigate routers or switches on the path, using telnet or SNMP.

Considering the established procedures of experienced network managers, a procedural programming language was designed which is capable of integrating macroscopic performance measurement tools and microscopic measurement tools. A network performance management system, called ipsh, was built around the language. Ipsh is similar to command shells commonly available on Unix operating systems in that execution of macroscopic process, such as ping, can be described briefly. At the same time, ipsh resembles C and other procedural language in that obtained data can be processed with microscopic operations. Ipsh attempts to integrate macroscopic and microscopic measurement tools by blending desirable properties of these two entirely different language families.

Ipsh makes measurement results from various tools accessible within single language. Measurement results from SNMP, `ping` and `traceroute` are converted to common data format, that makes it possible to easily compare application performance with performance characteristics of network and datalink layers.

In ipsh, procedure can be defined for periodic measurements across multiple layers and multiple subnetworks, which is called watchpoint. By setting watchpoints, performance and error characteristics can be automatically collected without human intervention. Accumulated statistics can eliminate many of the steps in problem diagnosis.

Watchpoint facilitates identification of unusual behavior by comparing current characteristics of networking devices with the ones collected before. Periodic probe makes it possible to collect statistics not only for unusual cases but also for normal cases.

The language is designed to permit reference to these performance characteristics from within various procedures. Ability to apply statistical filters, such as mean, median or variance, to these values at the language level is essential for automated detection and diagnosis of performance problems.

Ipsh simplifies diagnostic interface by providing uniform syntax and coherent interface for probing various functional components of internetworks. Managers can perform probing simply by "probe component-name", where disparate commands had to be used in conventional environments. This simplicity can potentially improve manageability of internetworks, since disparate command names, such as `netstat -s`, `ping`, `nfsstat` or `rpcinfo -p`, have been the major obstacle in probing various subsystems.

### 4.4.1 Functions

Ipsh is intended to be a workbench for network performance management, supporting various tasks as follows: periodic data collection and storage, detailed data collection when symptoms of performance problems are observed, interactive probing of internetworks, interactive analysis of accumulated data, and report generation.

Since many of these tasks require various operations such as data storage, extraction, conversion, translation and visualization, ipsh has many functions that support seamless interaction between data processing subsystems, while at the same time providing uniform interface to users. Ipsh also has a number of functions for processing collected data in detail, at the language level. These functions together make it possible to generate reports or visualize results depending on the current status of internetworks.

### 4.4.2 Module Organization

Ipsh comprises of the following components and interfaces between them: command shell for interaction with users, data collector for gathering various performance and error statistics, filesystem, data slicer, mathematical filter, visualizer, and report generator

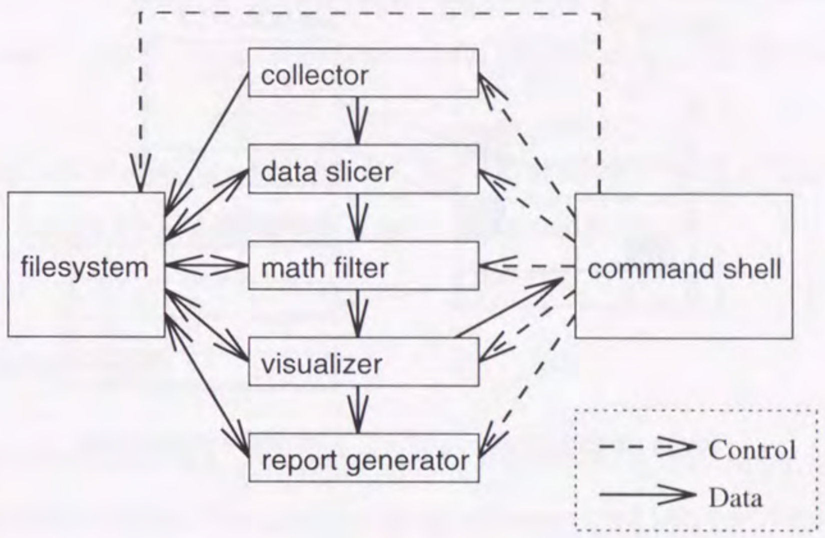

Figure 4-1: The structure of ipsh.

(Figure 4-1). The flow of control is depicted in dashed lines in the figure.

Data collector records statistics obtained with SNMP, `ping`, `traceroute`, or `netstat`. A common data structure used to store these statistics are described in the next section. Data slicer extracts data, using user specified range (time period, subnetworks etc.), from one or more datasets. Data slicer also merges different datasets if necessary. Mathematical filter is then applied to matrices and vectors of time-series data. Visualizer takes data of arbitrary dimension as an argument, and feeds it to visualization program specified by the user. Report generator converts resulting graph, performance index or logged messages to MIME format and then sends them by e-mail.

### 4.4.3 Data Structure

A common data structure is used among components of ipsh. First, a matrix is prepared for each measured subsystem. Measured subsystems are uniquely identified by the hostname and the measured layer. A matrix comprise of a series of records (time, V, detail) that are sorted by time stamps. The time interval of adjacent records may not be a fixed constant. V is a vector of data, either of integer, floating or string, that are obtained

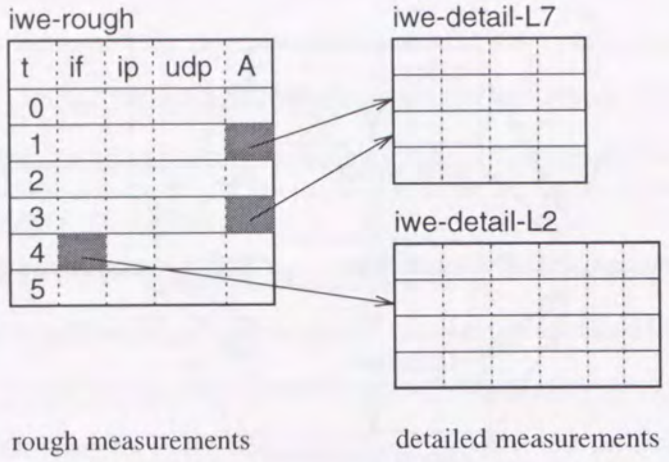

Figure 4-2: The data format of ipsh.

through measurements of datalink, network or application layers. The detail field points at particular record in other matrix. The detail field is used when unusual conditions are detected, pointing at the data collected with detailed probe. An conceptual example is illustrated in Figure 4-2.

The data structure is designed so that traversal from rough measurements to detailed measurements are possible. There are close relationships between the structure of watchpoints and the structure of matrices. An watchpoint that probes various subsystems conditionally will produce hierarchically organized matrices for one target host.

Since the time interval between measurements need not be fixed, it is possible to perform fine-grained measurements when performance problems are detected. Also, collected data can be reduced by making the time interval larger on non-working days, for example.

## 4.5 Implementation of the System

Ipsh has been implemented in C, using yacc and lex for language parser. The current implementation of data collector interfaces with SNMP, `ping` and `traceroute`. The

current version of data slicer is capable of slicing data by time period, hostname, and by the column name in the matrix. In the visualizer, interfaces to xgobi[51] and xgraph are implemented.

Current implementation use text format for representing common data format. Standard scientific format such as HDF[52] may be used in the future.

## 4.6 Discussion

There are a number of alternative approaches to implement a system for network performance management. Aside from incorporating management features into the language, it is possible to implement necessary functions as a library of general-purpose scripting languages like Perl, Tcl or Python. Although extending programming language with a library would be much easier to implement than designing a new language, a programming language was designed for interactive prototyping, since most of the general-purpose scripting languages are does not support interactive prototyping.

Interactive prototyping of administrative tasks, perhaps the most useful feature of Unix shell environments, are considered to be important in performance management tasks, since the ability to prototype programs in interactive programming environment makes it possible to capture patterns from repetition of similar commands.

Another important property of Unix shell environments are their syntactic constructs that make it possible to simplify expressions. For example, alias and function can give a name to a sequence of frequently used commands, while at the same time reducing the number of characters typed. Also, their syntax are designed with reduced syntactic complexity, eliminating the need to explicitly indicate the scope with special symbols like `()`, `[]` or `::`, in most of the cases (Figure 4.3). Many commands define commonly used patterns as default behavior, making it possible to omit optional parameters that change their behavior.

There seems to be a common misconception that seamless extension of existing programming language outweighs those benefits introduced by designing a new language

Table 4.3: Comparison of lexical simplicity.

| Language | Syntax of function call |
|---|---|
| Perl | `&func(a, b, c);` |
| Bourne Shell | `func a b c` |

that supports specific problem domain, primarily because learning a new language is considered to be a major overhead. However, seamless extension of existing language does not completely eliminate the overhead of learning new concepts that supports the problem domain.

If syntactic constructs, such as if, while and for, and mathematical operators are similar to C or other popular programming languages, learning a new language would not be a major overhead. A new language based on this approach is Java. Since the syntactic constructs of Java is quite similar to C or C++, it is easy to learn the language. Writing complex programs in Java requires significant amount of efforts however, since it requires at least several hours to understand the structure of class libraries and usage of many of the useful methods.

## 4.7 Evaluation

We can verify the following characteristics of ipsh that facilitate performance management tasks by the two examples presented here: i) performance problems can be identified without manager's attendance by automated and conditional probing of subsystems, ii) programs can be written as simple as shells in the Unix environment, iii) detailed data analysis can be written as much as C, iv) collected data can be minimized by conditional probes, v) network performance can be measured from user's viewpoint by the use of end-to-end performance measurement tools like `ping` and `traceroute`, as well as SNMP.

In the code fragment presented at Figure 4-3, the reachability and delay variance of several routers in widen.ad.jp domain were measured periodically, using ICMP echo.

```
watchpoint widen-all {
    # ping all NOCs
    probe ip ips1.nara.widen.ad.jp;      ..
    probe ip fuj4.kyoto.widen.ad.jp;       :  (1)
    probe ip bay5.tokyo.widen.ad.jp;     ..
    afterprobe {
        # this part applied to all probes
        if (ip.loss == 100) {            ..
            alert terse;                   :  (2)
        } else {                         ..
            if (ip.rtt >                         ..
                2 * median(ip.past.rtt[1..7])) {   :
                alert "rtt surge";                 :  (3)
                log ip.rtt;                        :
            }                                    ..
        }
    }
    mailto noc-ops@widen.ad.jp;              ----- (4)
}
```

Figure 4-3: Periodic measurements of reachability and delay variance.

The code works as follows. First, it sends ICMP packets to sampled routers using probe commands and then wait for their reply (1). For all probes performed, sentences enclosed by afterprobe clause is executed. If there were no response within certain period, leave simple message describing the symptom to the log file (2). If the measured round trip time exceeds twice the median of previous seven measurements, the trend and the current value is logged (3). If any abnormal conditions are observed during execution of the code, electronic mail will be sent to the address specified by mailto command (4).

The code fragment of Figure 4-4 periodically monitors the availability of web servers. When any of the web servers does not respond, it attempts to determine which layer, either DNS, IP, TCP or application, is causing the problem. The code works as follows. First, it checks the availability of web servers using http protocol (1). For each web server, the response time is logged (2). If the web server does not respond via http protocol, availability of DNS server is checked, and then the reachability at the IP layer is tested. If the IP layer was the problem, associated route at the nearby router, cisco-itc1 in this example, is retrieved with SNMP and then recorded (3). If the web server responds at the IP layer but not at the http subsystem, an attempt is made to connect to the server with echo protocol running on top of TCP (4), to see if some systems at the application layer are functioning.

## 4.8 Summary

Failure modes of the internetworks running TCP/IP protocol suite are analyzed. It was pointed out that existing network management system assist the task of ensuring connectivity, leaving other problems to be solved. Failure in delivering expected performance to users are becoming major problem with the emergence of traffic-intensive applications and new networking devices.

Techniques for automated identification of performance problems in the Internet are presented, that are commonly applicable regardless of underlying networking technologies. A performance management system and underlying programming language were

```
watchpoint iwe-web-all {
    # L7 reachability to all web servers
    probe http park.org;                    ...
    probe http japan.park.org;                 :  (1)
    probe http www.expo96.ad.jp;            ...
    afterprobe {
        log http.rtt;                       ..... (2)
        if (http.unreach) {
            # what's the problem?
            probe dns ns.expo96.ad.jp;
            probe ip _;
            if (ip.unreach) {
                log snmp cisco-itc1 {                  ...
                    ipRouteEntry ipRouteDest _ {          :
                        ipRouteIfIndex,                   :  (3)
                        ipRouteMetric1                    :
                    }                                     :
                }                                      ...
            } else probe tcp(echo) _;       ..... (4)
        }
    }
    mailto noc@expo96.ad.jp;
}
```

Figure 4-4: Monitoring availability of web servers.

developed to easily apply these techniques under various circumstances. The system can help administrators automate management tasks that have been done manually.

# Chapter 5

# Conclusions and Further Work

## 5.1 Lessons Learnt

In this dissertation we examined scalability and manageability in the Internet environment. Although the TCP/IP protocol suite was designed with scalability and manageability, there are some problems around the protocol suite, not in the protocol suite itself, that might fundamentally affect the scalability of the Internet. One of these problems are simplicity of information access architecture, which cannot cope with complex issues such as network outages, security, access cost and access policy. Another problem is the simplicity of networking software implementation, which relies directly on the TCP transport protocol and hence cannot properly handle subnetwork failure. The solution proposed in this dissertation is to introduce access manager between servers and clients, where most of these problems can be addressed in seamless and transparent way.

The last problem described in this dissertation is the network performance problem. Since the TCP/IP protocol suite makes it possible to interconnect many kinds of networking devices in arbitrary topology, some part of them often become performance bottleneck. The scale and complexity of constructed networks have been the major obstacles to maximize their performance. The solution to this problem proposed in this dissertation is a device-independent performance management method and a software system that supports proposed method.

## 5.2 Summary of Contributions

This dissertation has made a number of contributions to the areas of scalable networking software architecture and scalable network performance management architecture. We summarize the major contributions in this section.

- A discussion on the characteristics of protocols and underlying models of the Internet was presented. Three aspects were examined in detail: scalability of networking software, manageability in networking software and scalability in networking hardware. Two problems were identified, that scalability has been achieved only in transport, routing and name resolution protocols, and that scalability in the hardware cannot be maximized without established techniques for detecting performance bottlenecks.
- Evaluation criteria for scalability and manageability in networking software were proposed.
- A new approach for improving scalability and manageability of information access was introduced, which is based on the concept of access manager. An implementation of distributed file system based on the new approach, the cluster server, was presented, which eliminates many of the problems that existing information access systems face.
- The mechanism for improving service availability in the face of machine crashes or network troubles were implemented. Its effectiveness has been evaluated through its deployment on operational internetworks.
- A general method for automated identification of network performance problems was proposed. A data collection and processing architecture to monitor network performance is described.

## 5.3 Future Work

*A. Application of the Scalability Improvement Techniques to the Web*

Most of the schemes proposed in this dissertation are directly applicable to the Web. Although some of the techniques are implemented in several Web proxies, the automated server selection based on performance and reliability statistics has not been incorporated yet. We believe that the application of these schemes can greatly enhance scalability of the Web on the Internet.

*B. Stable Routing in Operational Internetworks*

Current routing protocols are vulnerable to human errors in router configuration and incorrect behavior in routing protocol implementation, resulting in their inherent instability. Technology developments are expected to realize ultimate stability in operational internetworks.

*C. Zero-Administration Internet Server*

If the inherent instability in operational internetworks persist in the future, efforts must be made to standardize the use of alternative servers. An area for further work is to reduce the management overhead required for configuring, maintaining and re-configuring each server. For example, an integrated system for configuring and replicating mail server, directory server and name server would greatly reduce the administration cost.

*D. Development of Design Patterns for the Internet*

Most of instability in particular Internet sites stems from some of the following problems: incorrect combination of internetworking devices, incorrect configuration of particular internetworking devices, or excess load on some internetworking devices. An interesting study would be to develop a design pattern for large-scale installations, with correct combination of internetworking devices, scalable topology, and configuration guidelines.

# Bibliography

[1] L. G. Roberts. The Evolution of Packet Switching. *Proceedings of the IEEE*, 66(11):1307–1313, 1978.

[2] J. M. McQuillan and D. C. Walden. The ARPA Network Design Decision. *Computer Networks*, 1:243–389, 1977.

[3] Vint Cerf. The Catenet Model for Internetworking, IEN-48. Technical report, DARPA/IPTO, July 1978.

[4] Jon Postel. Internet Protocol, RFC791. September 1981.

[5] Jon Postel. Transmission Control Protocol, RFC793. September 1981.

[6] Yakov Rekhter and Tony Li. A Border Gateway Protocol 4 (BGP-4), RFC1771. March 1995.

[7] John Moy. OSPF Version 2, RFC1583. March 1994.

[8] Jon Postel. Internet Control Message Protocol, RFC792. September 1981.

[9] Jeffrey D. Case, Keith McCloughrie, Marshall T. Rose, and Steven Waldbusser. Protocol Operations for version 2 of the Simple Network Management Protocol (SNMPv2), RFC1448. May 1993.

[10] Paul V. Mockapetris and Kevin J. Dunlap. Development of the Domain Name System. In *Proceedings of ACM SIGCOMM'88*, pages 123–133, August 1988.

[11] Wengyik Yeong, Tim Howes, and Steve Kille. Lightweight Directory Access Protocol, RFC1777. March 1995.

[12] Matt Blaze and Rafael Alonso. Long-term caching strategies for very large distributed file systems. In *Proceedings of Summer USENIX Conference*, pages 3–15, Nashville, TN, June 1991.

[13] Peter Danzig, Michael Schwartz, and Richard Hall. A case for caching file objects inside internetworks. In *Proceedings of ACM SIGCOMM'93*, pages 239–248, September 1993.

[14] John H. Hartman and John K. Ousterhout. Zebra: A striped network file system. Technical report, University of California, Berkeley, 1992.

[15] Thomas E. Anderson, Michael D. Dahlin, Jeanna M. Neefe, David A. Patterson, Drew S. Roselli, and Randolph Y. Wang. Serverless network file systems. In *Proceedings of the 15th ACM Symposium on Operating Systems Principles*, December 1995.

[16] Yukio Murayama, Youki Kadobayashi, and Suguru Yamaguchi. Design and Implementation of DBS: a Performance Evaluation System for TCP. In *Proceedings of Internet Conference '96*, pages 39–44, July 1996.

[17] Vern Paxson. End-to-end routing behavior in the internet. In *Proceedings of ACM SIGCOMM'96*, August 1996.

[18] CERT Coordination Center. wuarchive ftpd Trojan Horse. April 1994. CERT Advisory CA-94:07.

[19] Steven R. Kleiman. Vnodes: An architecture for multiple file system types in Sun UNIX. In *Proceedings of Summer USENIX Conference*, pages 238–247, Atlanta, GA, 1986.

[20] B. Clifford Neuman. The virtual system model for large distributed operating systems. Technical Report TR-89-01-07, Dept. of Computer Science, University of Washington, April 1989.

[21] Gene H. Kim and Eugene H. Spafford. Experiences with Tripwire: Using Integrity Checkers for Intrusion Detection. In *USENIX Systen Administration, Networking and Security Conference III*, pages 89–101, April 1994.

[22] Aviel D. Rubin. Trusted Distribution of Software Over the Internet. In *Symposium on Network and Distributed System Security.* Internet Society, 1995.

[23] David Muntz and Peter Honeyman. Multi-level caching in distributed file systems. Number Proceedings of USENIX Winter Conference, pages 305–313, January 1992.

[24] Mark Shand. *unfsd - user-level NFS server.* University of New South Wales, May 1988. Appeared on comp.sources.unix, Volume 15, Issues 1–2.

[25] Jan-Simon Pendry and Nick Williams. *Amd: The 4.4BSD Automounter Reference Manual.* Imperial College of Science, Technology & Medicine, March 1991.

[26] Sun Microsystems. rpcgen programming guide. In *Network Programming Guide*, chapter 3. Sun Microsystems, 1990.

[27] Richard P. Draves, Brian N. Bershad, Richard F. Rashid, and Randall W. Dean. Using continuations to implement thread management and communication in operating systems. In *Proceedings of the 13th ACM Symposium on Operating System Principles*, pages 122–136, October 1991.

[28] Paul Mockapetris. Domain names - concepts and facilities, RFC1034. November 1987.

[29] Paul Mockapetris. Domain names - implementation and specification, RFC1035. November 1987.

[30] Andrew D. Birrell, Andy Hisgen, Chuck Jerian, Timothy Mann, and Garret Swart. The echo distributed file system. Technical report, DEC Systems Research Center, September 1993.

[31] Anawat Chankhunthod, Peter Danzig, Chuck Neerdaels, Michael F. Schwartz, and Kurt J. Worrell. A Hierarchical Internet Object Cache. In *Proceedings of USENIX 1996 Annual Technical Conference*, January 1996.

[32] James Gosling, Bill Joy, and Guy Steele. *The Java Language Specification*. Addison-Wesley, 1997.

[33] Luca Cardelli. Obliq: A language with distributed scope. Technical Report 122, DEC Systems Research Center, June 1994.

[34] D. Clark. The Design Philosophy of the DARPA Internet Protocols. In *Proceedings of ACM SIGCOMM'88*, pages 106–114, August 1988.

[35] Gary R. Wright and W. Richard Stevens. *TCP/IP Illustrated: the implementation*, volume 2. Addison-Wesley, 1995.

[36] Eric Allman. Sendmail – An Internetwork Mail Router. In *4.4BSD System Manager's Manual*, pages SMM:9–1 – SMM:9–12. The USENIX Association, 1994.

[37] Fred Baker. Requirements for IP Version 4 Routers, RFC1812. June 1995.

[38] Transarc Corporation. *AFS-3 Programmer's Reference.*

[39] Auspex Systems, Inc. *Auspex version 1.8M1 Software Release Note*, 1995.

[40] Michael A. Cooper. Overhauling Rdist for the '90s. In *LISA VI*. USENIX Association, October 1992.

[41] Steven Shafer and Mary Thompson. The SUP Software Upgrade Protocol. Technical report, School of Computer Science, Carnegie Mellon University, September 1989.

[42] Bilal Chinoy. Dynamics of Internet Routing Information. In *Proceedings of ACM SIGCOMM'93*, September 1993.

[43] Internet Engineering and Planning Group. Operational Advisory Note on Route Flapping. Draft, March 1994.

[44] Curtis Villamizar. Controlling BGP/IDRP Routing Traffic Overhead, IETF BGP Working Group. 1993.

[45] Takuo Tada. Studies on route management – design and implementation of a network management system for the large scale internet. Master's thesis, Graduate School of Information Science, Nara Institute of Science and Technology, February 1997.

[46] Radia Perlman. *Network Layer Protocols with Byzantine Robustness.* PhD thesis, MIT, 1988.

[47] Keith McCloughrie and Marshall T. Rose. Management Information Base for Network Management of TCP/IP-based internets: MIB-II, RFC1213. March 1991.

[48] Robert Braden and Annette DeSchon. *NNStat: Internet Statistics Collection Package.* USC/ISI.

[49] Nevil Brownlee. *NeTraMet and NeMaC Reference Manual.* The University of Auckland, June 1995.

[50] Allan Leinwand. Accomplishing Performance Management with SNMP. In *Proceedings of INET'93*, August 1993.

[51] Deborah F. Shwayne, Dianne Cook, and Andreas Buja. *User's Manual for XGobi: a Dynamic Graphics Program for Data Analysis.* Bellcore, November 1991.

[52] National Center for Supercomputing Applications. *HDF User's Guide*, April 1996.