第5部

ネットワークトラフィック統計情報の収集と解析

第 1 章　MAWI ワーキンググループ

MAWI(Measurement and Analysis on the WIDE Internet) ワーキンググループは、トラフィックデータの収集と解析、また、データの保存と利用に関する活動を行なっている。

回線利用率やプロトコルの分布を示すマクロなトラフィックデータは、故障や問題の検出を容易にし、バックボーントラフィックの傾向の分析に役立つと同時に、ネットワーク機器選定や将来のバックボーンの構成、容量設計に欠かせない重要なデータである。

一方、パケットトレースに代表されるミクロなトラフィックデータは、生の記録をもとにさまざまな情報を抽出することが可能であるため、プロトコル設計、トラフィック理論をはじめ、あらゆるネットワーク研究にとって貴重なデータとなる。その反面、生データにはユーザのプライバシ情報も含まれるため、取り扱いには十分な注意が必要とされる。

今回の報告書では、まず、第 2 章で、現在進めているトラフィックデータ・レポジトリについて報告する。本プロジェクトでは、複数のサンプリングポイントでの定期的なトレースデータの収集、収集したデータをアーカイブするサーバの設置、収集データを自動処理するツールの開発等を行っている。また、パケットトレースをより有効に利用するため、プライバシ情報をスクリーニングした、安全に取り扱えるトレースデータの作成を実験中である。

第 3 章では、(株) KDD 研究所との共同研究として行なっている、WIDE 国際線のパケットトレース収集について報告する。本研究では、1999 年 5 月 13 日に増強された国際線におけるトラフィック変化について報告する。

第 4 章では、多地点で同時に収集されたデータに関して、トラフィック・パターンの特徴から時間同期を得る手法について報告する。

第 5 章では、パケットトレースをもとにウェブサーバの性能解析を行なう手法について報告する。

第 2 章　WIDE Traffic Data Repository

2.1 Introduction

In this paper, we introduce an on-going effort within the WIDE project to collect a set of free tools to build a traffic data repository containing detailed information of our backbone traffic. The WIDE project makes the resulting data sets publicly accessible so that this project is not only on freely-redistributable software but also on freely-redistributable traffic data sets.

The WIDE project is a research consortium in Japan established in 1987. The members of the project include network researchers, engineers and students of universities, industries and government. The focus of the project is empirical study on a live large scale internet. Thus, WIDE runs its own internet testbed carrying both commodity traffic and research experiments. WIDE is also responsible for various Internet operations including M-root name server, NSPIXP(Network Service Provider Internet eXchange Point), AI3(Asian Internet Interconnection Initiatives), and 6Bone in Japan.

The goals of our traffic repository are to promote traffic analysis research as well as to promote development of tools. Traffic characteristics in a backbone network are considerably different from those in a local area network but few people have access to traffic traces from backbone networks. Obtaining details of backbone traffic is getting harder as more backbone networks are shifting to commercial ISPs, which motivate us to build a traffic repository [8].

Traffic traces are collected at several points within the WIDE backbone. Traces are in *tcpdump* raw format so that all header information is available and can be used for detailed analysis.

We use commodity hardware and the existing freely-available tools for building our traffic repository so that it has nothing technically fancy. Our

focus is rather continuity in making the latest traces available. At this writing, daily traces at one sample point have added up to the record of more than a year.

2.2 Related Work

Packet Monitoring

Packet capturing were brought with the advent of Ethernet. The first personal computer, Xerox Alto, already had programs to monitor Ethernet. As Ethernet came into wide use, dedicated network monitors became indispensable to developers and operators. The CMU/Standford enet packet filter is the first UNIX based packet filter developed in 1980 [9]. It eventually evolved into the Ultrix Packet Filter at DEC, NIT under SunOS, and BPF.

Userland programs that prints the headers of packets appeared with UNIX workstation. Sun implemented NIT (Network Interface Tap) to capture packets and *etherfind* to print packet headers. The advantage of UNIX-based monitoring tools is that users can use other software tools available on UNIX for manipulating and analyzing packet traces.

tcpdump [10] is probably the most popular packet capturing tool in the UNIX community. *tcpdump* first appeared in 1989 and merged into BSD Net Release2 in 1991. *tcpdump* is based on a powerful filtering mechanism, the BSD packet filter (BPF) [11]. The packet capturing and filtering facilities of *tcpdump* are implemented in a separate library, *pcap* [12]. The pcap library became independent from *tcpdump* in 1994, and there are a wide range of network monitoring or analysis tools which integrate the *pcap* library. In 1999, tcpdump.org [13] was organized by volunteers to maintain the *tcpdump* code.

High-performance monitoring systems are explored by OC3MON [14] and its successors that are based on a PC hardware but exclusively for ATM . CoralReef [15] is a package developed at CAIDA to analyze the output of OCxMON.

Packet monitoring techniques have been used to gather long-term statistics. A pioneering work is *statspy* [16] in the NNStat package developed at ISI. As SNMP becomes widely available, network statistics tools are geared toward SNMP. MRTG [17] and its successor RRDtool [18] are popular tools to collect traffic counters from routers through SNMP. More recently, cflowd [19] is developed at CAIDA to make use of Cisco's NetFlow [20] that exports statistics of flow cache entries.

Traffic Archive

The Internet Traffic Archive (ITA) [21] was created in 1995 by Danzig *et al.* to promote research on network analysis. ITA has several traces studied in published papers as well as unstudied traces. ITA is an important step towards open traffic data sets because research based on open data sets can be confirmed or further analyzed by someone else, which leads to deeper studies.

There are several different formats in the ITA archives but the majority of the available traces are in the *tcpdump* ascii output format. A set of shell scripts, called *sanitize*, are written by Paxson and used to scramble addresses in the *tcpdump* ascii output format to provide anonymity to network users.

Our traffic repository was motivated in part by the effort at ITA. We employ automatic traffic sampling at regular intervals since the archives at ITA do not seem to be updated much. We also thought that the *tcpdump* raw format is preferable to the ascii format because the raw format has more information and powerful tools are available to manipulate the raw format.

2.3 Motivation

WIDE installed several traffic sampling points within the backbone since traffic data has been essential to both network research and operation. However, traffic information tend to be confined to a small set of members, and it is difficult to

share detailed information without a framework to support sharing. This leads to the idea of a traffic trace repository in which detailed traffic traces are archived and easily accessible to everyone.

In order to build a traffic trace repository and make good use of traces, we had to solve two problems. One is to create a safety measure for handling traces that include privacy information. The other is automation of the trace acquisition process.

Traffic traces include private information of the network users. Special care is needed to handle traces, and thus, only limited members are allowed to handle raw traces. Still, there is always a risk of accidents when we handle raw traces. Hence, even if traces are available only for limited members, it is important to make traces safe enough to prevent possible accidents. On the other hand, if traces are made free from user privacy, we can make the traces open to the public since WIDE does not need to worry about its impact to stock prices.

Automation of the maintenance process is the other important factor. Collecting traffic traces in a long term needs perseverance, and cannot be achieved unless most of the work are automated. Not only automation of acquisition but also automation of summarization and visualization are essential to maintaining the repository because people tend to run out of energy if no feedback is given.

There are strong concerns about security and privacy with regard to making traces publicly available. After a long discussion, we have reached a conclusion that the benefits outweigh the risks. Or, at least, it is worth a challenge.

2.4 Privacy Issues

Traffic traces contain privacy information including network addresses and application payload so that it is important to understand issues involved in user privacy.

There are 2 major issues regarding user privacy.

Removing user data: User private data must be removed from traces. Traffic traces should have only protocol headers and protocol payload which contains user data should be removed.

Providing anonymity: IP address is unique and can be used to identify a user, and thus, addresses should be scrambled to provide anonymity to users.

There are a wide variety of research purposes that have different requirements for traces. No single method will satisfy all the requirements and still keep user privacy. We are trying to provide traces which can be used for a wide range of research. For research which has specific requirements, our traces will provide a starting point, and can be used to narrow down its requirements. Then, it will be easier to find a specific method to meet the requirements.

2.4.1 Removal of Payload

As a general rule, we should remove the payload of TCP or UDP that contains users' private information. If another protocol header exists on top of a TCP or UDP header and the inner header does not contain user private information, the inner header may be maintained. If it is difficult to judge whether a header contains user private information or not, the header should be removed as a precaution.

Once protocol payload is removed, the risk of jeopardizing user privacy is considerably reduced. It would be safe enough for use within a closed group. However, in order to make traces open to the public, we need a further level of security. That is, we need to provide anonymity to network users.

2.4.2 Address Scrambling

We should provide anonymity to individuals and organizations by scrambling source and destination addresses in IP headers. IP addresses, however, have hierarchical structures and special

addresses such as broadcast addresses, multicast addresses and private addresses. It is not easy to provide anonymity but still keeping the structures and special meanings. We should chose an appropriate method according to the importance of anonymity in traces and the purpose of the data set.

Address Scrambling Methods

Address scrambling maps one IP address to another IP address. There are a number of methods to scramble addresses.

1. the sequential numbering method maps each IP address occurrence to a sequential number. Although this method is easy to understand, it is difficult to preserve other meanings of addresses.
2. the hash method maps an IP address to another IP address using a hash function in order to provide random mapping. It is also possible to preserve the common address prefix between 2 addresses by maintaining an ordered tree of addresses similar to a routing table. In this method, if 2 IP addresses have a common address prefix, they are mapped to addresses with a common address prefix of the same length. Note that, although it preserves routing information, this method has a risk of being reverse-engineered. For example, one can use a well-known server's address as a clue to de-scramble the address prefix [22]. The impact of this threat, however, depends on the importance of hiding the network topology.

There are several choices regarding address consistency between two or more data sets.

1. all occurrences of an address are to be mapped to a single address within a data set.
2. all occurrences of an address are to be mapped to a single address across different data sets.

Longer consistency is convenient for analysis but it also makes reverse-engineering easier.

Address Issues

non-unique addresses Addresses not containing user identifiers may be left without scrambling. Those addresses include broadcast addresses, multicast addresses, and private addresses. In the case of IPv6, link-local addresses and site-local addresses could contain unique interface identifier (e.g., MAC address). A solicited-node multicast address contain lower bits of the global address. Therefore, these IPv6 addresses should be scrambled as well.

addresses in upper layers IP addresses could be contained in an upper protocol message. For instance, ICMP and DNS contain IP addresses in the protocol payload. These addresses must be scrambled in the same manner, or removed.

MAC addresses Link-layer headers (e.g., Ethernet headers) contain MAC addresses. A MAC address contains vendor and model information which could be part of user privacy or lead to a security hole. However, traces from backbone networks do not contain MAC addresses of user nodes since MAC addresses recorded in the trace are only from local nodes on the same segment.

IP/TCP options An IP options can contain IP addresses. Addresses in IP options should be scrambled in the same manner. Otherwise, IP options should be replaced by NOP options, or removed.

On the other hand, TCP options do not contain privacy information. TCP options carry useful information to analyze TCP behaviors so that TCP options may be preserved.

2.5 Methods

We use several tools to automatically maintain the traffic repository. The details of these tools are described later in this section. New trace data is collected from sampling points to the repository during the night. A web page for the new trace is automatically created.

At a sampling node, a script is invoked from *cron* to run *tcpdump* and compress the trace. The raw trace file is placed under a certain directory.

At the repository node, another script is invoked from *cron* to fetch the raw trace and process it. The script copies the compressed raw trace from the sampling point over a secure session using *scp*. Then, the script uncompresses the trace and invokes *tcpdpriv* to remove privacy information from the trace. The trace is fed into *tcpdstat* to get a summary output. The script creates a web page for the trace, and updates the index page to include the newly created page. Finally, the script compresses the trace data again, and place it for ftp.

2.5.1 tcpdump

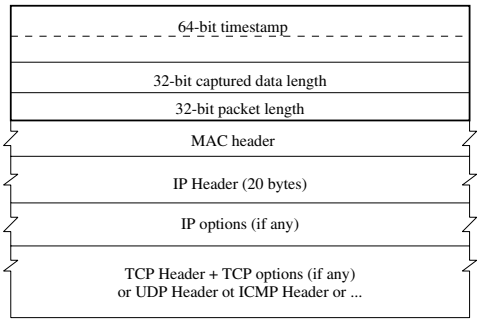

図 2.1 Pcap header format

We use *tcpdump* to obtain traffic traces because *tcpdump* is widely used, and installed as part of the default tools on many systems. In addition, there are many tools that integrates the *pcap* library and be able to read *tcpdump* output files. Those tools include *tcptrace*, *tcpslice*, *tcpdstat* and *ttt*.

tcpdump, by default, puts the network interface into *promiscuous mode* to capture every packet going across the wire. In the BSD-derived kernel, BPF is implemented as a packet capture mechanism. When BPF is enabled, the network driver in the kernel passes both sending and receiving datalink level frames to BPF. BPF performs packet filtering if necessary, adds timestamp, and copies the fixed length from the head of the frame into the store buffer. *tcpdump* can read multiple frames in a single read from the store buffer in the kernel in an efficient manner. *tcpdump*, by default, prints the header information of each packet in a text format. With *-w* option, *tcpdump* writes out the packet frames into a specified file. With *-r* option, *tcpdump* reads from a saved file instead of a network interface to replay a saved file. The *pcap* library is used to read or write data in the raw format. Thus, it is easy to write a program to read or write packets in the *tcpdump* format.

Figure 2.1 shows the format of raw *tcpdump* output. In the BSD systems, the kernel uses *microtime()* for timestamp so that the precision of the timestamp depends on the machine architecture. Also, the timestamp is taken when a packet is passed to BPF from the network driver so that it is the time that the driver sees that packet.

2.5.2 tcpdpriv

We use *tcpdpriv* to remove user data and scramble addresses. *tcpdpriv* was developed by Greg Minshall at Ipsilon Networks in 1996. *tcpdpriv* removes privacy information in a raw *tcpdump* output. *tcpdpriv* uses the *pcap* library to read and write *tcpdump* output files. *tcpdpriv* removes the payload of TCP and UDP, and the entire IP payload for other protocols. *tcpdpriv* implements several address scrambling methods; the sequential numbering method and its variants, and the address prefix preserving method.

However, the original *tcpdpriv* lacks several features we need:

- it does not support IPv6.
- it does not preserve TCP options that are

essential to analyzing TCP behaviors.

- we also want to preserve other protocols such as ICMP, ARP and DNS.

Thus, we have modified the original *tcpdpriv* to support these features. The default settings are also changed to meet our requirements since the options seem to be too complex and a mistake of option selection could be fatal to user privacy.

2.5.3 tcpdstat

We developed *tcpdstat* to get summary information of a *tcpdump* file. *tcpdstat* reads a *tcpdump* file using the *pcap* library and prints the statistics of a trace. The output includes the number of packets, the average rate and its standard deviation, the number of unique source and destination address pairs, and the breakdown of protocols.

tcpdstat is intended to provide a rough idea of the trace content, and to be processed for a web page. It provides helpful information to find anomaly in a trace. For example, if the traffic volume of ICMP or DNS is unusually large, or if the traffic volume of a specific address pair is unusually large, it is a sign of some form of a DoS attack.

2.5.4 Other Tools

There are other tools that are not used to create the traffic repository but can read *tcpdump* files and useful for analyzing traces afterwards.

tcpslice by Vern Paxon extracts portions of a trace. *tcptrace* by Shawn Ostermann produces detailed information about each TCP connection in a trace. *tracelook* by Greg Minshall provides *xgraph* plots of TCP connections in a trace. *flstats* also by Minshall prints flow statistics. *ethereal* by Gerald Combs is a traffic analizer with a graphical user interface. *ethereal* uses the *pcap* library, and thus, can replay a *tcpdump* file. Our *ttt* (Tele Traffic Tapper) tool displays composition graphs of protocols and host addresses in real time. *ttt* can replay a trace file at a given speed so that it is possible to replay a 1-hour trace in 1 minute.

2.6 Current Status

Currently, we are collecting daily-traces from the following sampling points.

trans-pacific is a 1.5Mbps T1 line, one of the several international links of WIDE. The sampling point is on an Ethernet segment one hop before the T1 line. The incoming traffic (from U.S. to Japan) of this link is fairly congested.

6Bone is located on a FastEthernet segment connected to NXPIXP-6 (An IPv6 internet exchange point in Tokyo) [23]. The segment located at an AS boundary, and the traffic includes only native IPv6 and does not include IPv4 except tunneled IPv4 over IPv6. Because NXPIXP-6 is built on a FastEthernet switch, only a small portion of the total traffic can be captured.

Traces are sampled at a fixed time of day. This is obviously not desirable and we need to find a better sampling method.

We started daily data collection at the *trans-pacific* point in February 1999. Since WIDE has a number of connections to the Internet exchange points, the return path of a session does not necessarily go through the same link.

The size of each trace file is about 100M bytes (about 40MB when compressed). We believe 100MB is an appropriate size for handling on a commodity PC as well as for fetching over the network, still has enough information for statistical analysis.

Figure 2.2 is a sample output of *tcpdstat* from the *trans-pacific* point on February 12, 2000. This 1-hour-long trace contains about 2 million packets, the number of unique address pairs is about 56K. HTTP is dominant in the trace, 70% of the total packets and 62% of the total bytes.

Among the collected traces, some data sets contain traces of DoS attacks such as *portscan* and *smurf*. These traces could be useful for development of tools to detect such attacks.

```
DumpFile: 200002121359.dump
FileSize: 140.35MB
Id: 200002121359
StartTime: Sat Feb 12 13:59:00 2000
EndTime: Sat Feb 12 15:06:29 2000
TotalTime: 4048.47 seconds
TotalCapSize: 108.37MB CapLen: 76 bytes
# of packets: 2095754 (449.89MB)
AvgRate: 932.21Kbps stddev:312.68K

Packet Size Histogram (including MAC headers)
 [   32-   63]:    1315693
 [   64-  127]:     258761
 [  128-  255]:     121532
 [  256-  511]:     113037
 [  512- 1023]:     137300
 [ 1024- 2047]:     149431

IP flow (unique src/dst pair) Information
# of flows: 56157 (avg. 37.32 pkts/flow)
Top 10 big flow size (bytes/total in %):
8.0% 6.0% 4.8% 2.8% 2.6% 2.0% 1.4% 1.2% 0.8% 0.7%

Protocol Breakdown
 protocol     packets                bytes        bytes/pkt
------------------------------------------------------------
 total    2095754 (100.00%)   471744043 (100.00%)  225.10
 ip       2095736 (100.00%)   471743089 (100.00%)  225.10
  tcp     1768533 ( 84.39%)   400258906 ( 84.85%)  226.32
   http   1474686 ( 70.37%)   292981631 ( 62.11%)  198.67
   squid    42778 (  2.04%)    36118457 (  7.66%)  844.32
   smtp     74280 (  3.54%)    29130167 (  6.17%)  392.17
   nntp      1270 (  0.06%)      101659 (  0.02%)   80.05
   ftp      23779 (  1.13%)     7180413 (  1.52%)  301.96
   pop3      5601 (  0.27%)     2537763 (  0.54%)  453.09
   telnet     995 (  0.05%)       88678 (  0.02%)   89.12
   ssh       1950 (  0.09%)      230243 (  0.05%)  118.07
   dns       1169 (  0.06%)       94179 (  0.02%    80.56
   bgp       6090 (  0.29%)      419164 (  0.09%)   68.83
   other   135935 (  6.49%)    31376552 (  6.65%)  230.82
  udp       264967 ( 12.64%)   62915121 ( 13.34%)  237.45
   dns     187377 (  8.94%)    29884111 (  6.33%)  159.49
   rip        135 (  0.01%)        8910 (  0.00%)   66.00
   other    77455 (  3.70%)    33022100 (  7.00%)  426.34
  icmp      51228 (  2.44%)     5609707 (  1.19%)  109.50
  igmp        801 (  0.04%)       48060 (  0.01%)   60.00
  ospf       8909 (  0.43%)     2283318 (  0.48%)  256.29
  ipip          3 (  0.00%)         246 (  0.00%)   82.00
  ip6        1295 (  0.06%)      627731 (  0.13%)  484.73
  frag        112 (  0.01%)      157111 (  0.03%) 1402.78

tcpdump file: 200002121359.dump.gz (45.94 MB)
```

図 2.2 sample output of tcpdstat at trans-pacific

The *6bone* point has been added in January 2000. The traffic volume of the *6bone* point is still low; the average rate is around 100Kbps and the mojority of traffic is BGP and ICMPv6. However, we expect IPv6 traffic will increase in a few years as major router or OS vendors have started shipping IPv6 in their base systems. Our intention is to record the evolution of IPv6 traffic in a long term.

Figure 2.3 is the output of *tcpdstat* from the *6Bone* point on the same day. This 3.5-hour long trace contains about 200K packets, the number of unique address pairs is about 270.

We expect that IPv6 traces will be useful for development of tools to support IPv6 since IPv6 traffic traces, especially on a backbone link, are

```
DumpFile: 200002120900.dump
FileSize: 17.64MB
Id: 200002120900
StartTime: Sat Feb 12 09:00:00 2000
EndTime: Sat Feb 12 12:33:45 2000
TotalTime: 12825.20 seconds
TotalCapSize: 14.69MB CapLen: 94 bytes
# of packets: 193424 (48.84MB)
AvgRate: 40.31Kbps stddev:63.46K

Packet Size Histogram (including MAC headers)
 [   64-  127]:     100654
 [  128-  255]:      66924
 [  256-  511]:       1179
 [  512- 1023]:       2322
 [ 1024- 2047]:      22345

IP flow (unique src/dst pair) Information
# of flows: 270 (avg. 716.39 pkts/flow)
Top 10 big flow size (bytes/total in %):
53.9% 4.0% 3.8% 3.7% 3.6% 3.6% 3.1% 2.9% 2.9% 2.9%

Protocol Breakdown
protocol     packets              bytes     bytes/pkt
-----------------------------------------------------
 total   193424 (100.00%)  51210692 (100.00%) 264.76
 ip6     193424 (100.00%)  51210692 (100.00%) 264.76
  tcp6   184430 ( 95.35%)  49453242 ( 96.57%) 268.14
   smtp     402 (  0.21%)     54893 (  0.11%) 136.55
   ftp    51229 ( 26.49%)  29569720 ( 57.74%) 577.21
   ssh       53 (  0.03%)      6820 (  0.01%) 128.68
   bgp   132476 ( 68.49%)  19798783 ( 38.66%) 149.45
   othe     270 (  0.14%)     23026 (  0.04%)  85.28
  udp6      469 (  0.24%)     36610 (  0.07%)  78.06
   othe     469 (  0.24%)     36610 (  0.07%)  78.06
  icmp6    7346 (  3.80%)   1489628 (  2.91%) 202.78
  ip4      1179 (  0.61%)    231212 (  0.45%) 196.11

tcpdump file: 200002120900.dump.gz (2.99 MB)
```

図 2.3　sample output of tcpdstat at 6bone

not widely available.

Although we started collecting traces and made them available, we have not studied the traces throughly. Rather, one of the purposes of our open traffic repository is to leave analysis to those who are interested in doing it.

If other organizations start building similar but possibly closed traffic repositories, it would be possible to share experiences and development of tools. Especially, packet traces could be a counter measure against the ever-growing threat of DoS attacks.

2.7 Future Work

Our focus at this moment is long-term data collection. So far, we have set sampling points only on relatively slow connections. Data collection from faster links is an obvious direction, but we have limited storage capacity and network capacity.

As for high-performance packet capturing, we can benefit from advanced research such as OC3MON [14]. OC3MON uses a DOS-based capturing tool to monopolize CPU, and takes advantage of the processor on the ATM card for offloading.

However, today's commodity PC is already quite powerful: Gigabit Ethernet is about

125MB/sec. The bus bandwidth of 32bit PCI at 33MHz is 132MB/sec, and 64bit PCI at 66MHz is 528MB/sec. The disk interface is getting faster as well. Ultra160 SCSI provides 160MB/sec. A single high-end disk has sustained rate of about 30MB/sec but disks can be used in parallel so that 4 disks provide about 120MB/sec. CPU power itself seems to be catching up, but it would be tricky to efficiently run *tcpdump* on non-realtime UNIX. It seems to be doable by a commodity PC to capture packets even at a gigabit network if the system is correctly tuned.

2.8 Conclusion

We have presented the WIDE traffic repository, an on-going effort to create archives of *tcpdump* files collected at several points within the WIDE backbone. Our attempt is a challenge to the legitimacy of concerns about revealing detailed traces for privacy and security reasons. We hope our repository will be useful for traffic analysis and for development of tools.

第 3 章　国際線トラフィックの解析

3.1 トラフィック収集および解析の重要性

トラフィックデータは、ユーザの数やネットワークトポロジ、経路などといったネットワークの状態によって変化する。このため、トラフィックデータの解析は、回線の切断や経路制御の不具合など、故障や問題の検知を容易にする。また、回線容量の不足が示せるため、ネットワークの設計や設備投資の目安にする事ができる。このため、トラフィックの収集と解析はとても重要である。

さらに、本報告書で述べるような、帯域の変化が起きた時点でのトラフィックデータの解析は、帯域容量とユーザへの影響の関連を述べるときの根拠としても必要とされる。

3.2 収集したデータについて

WIDE プロジェクトは、5 月 13 日に、アメリカへの国際線の帯域を 4Mbps から 11Mbps に増加した。帯域の増加による影響を調べるため、5 月 11 日から 5 月 15 日までのトラフィックデータを蓄積し、解析を行った。

トラフィックデータは、tcpdump で読むことのできるパケットのバイナリダンプとして蓄積した。

プライバシーを保護するため、蓄積時にはパケットの始点アドレスと終点アドレスを意味を持たない数字に変換した。アドレスを数字に変換するとき、アドレスの一意性を保つようにしたため、コネクションは識別できるようになっている。

3.3 解析手法

まず、帯域の変化を明確にするため、また帯域の利用効率を調べるため、スループットの変化について解析した。これを 3.4 章に示す。ここで、TCP と UDP の全体に対する比率が示される。3.5 章では、TCP に特化した解析を行ない、TCP コネクションの振舞いについて考察する。3.6 章では、UDP に特化した解析を行なう。さらに、最近のアプリケーションが使用するポートとデータ量の関係についても述べる。

3.4 スループット解析

3.4.1 全体データのサンプリング

帯域の変化を明確にするため、また帯域の利用効率を調べるため、スループットの変化を解析した。

図 3.1 では、tcpslice を用い、1 時間毎のデータの中から 3 分間のデータを抽出しグラフを作成した。このグラフから以下の事がわかる。

- 帯域が切り替わる以前も以後もデータのほとんどを TCP が占めている。
- 帯域が切り替わると TCP のスループットが著しく変化する。
- 帯域が切り替わっても UDP のデータ量はほとんど変化しない。

また、帯域が切り替わった後のスループットが著しく変化している事から、帯域が切り替わる以前は潜在的なサービスの要求に対して、帯域が不足していた事が推測される。

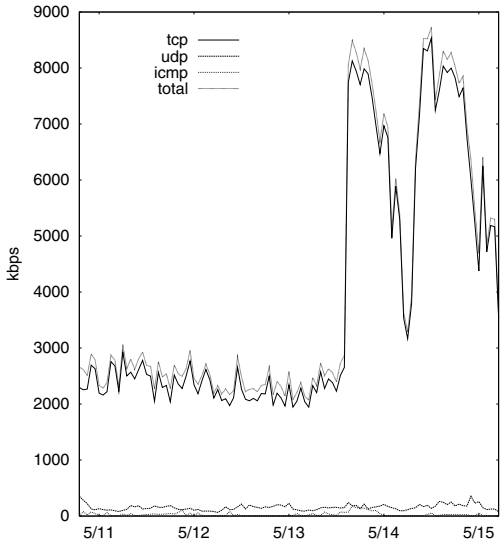

図 3.1　サンプリングによるスループット

図 3.2 は、tcpslice を用いて帯域が切り替わった前後 5 分間のデータを抽出しスループットを示したグラフである。図 3.3 は帯域が切り替わった前後 1 分間の全てのパケットをサイズ別に散布図にしたものである。

図 3.2 において、パケットの量が一時的に減少している。そのことから、帯域変更を行う際に何らかの作業が行われ、一時的にパケット転送が行われなかった事が推測できる。この 2 つのグラフから、以下の事がわかる。

- 実際に帯域が切り替わったのは 5 月 13 日の 15 時 28 分から 15 時 29 分の間である。
- 帯域が切り替わった瞬間は一時的にデータ量が少なくなる。
- 帯域が切り替わった直後からスループットは著しく増加する。

3.4.2 帯域が切り替わった前後の 5 分の tcptrace

図 3.4 は帯域が切り替わった前後 5 分間のデータ

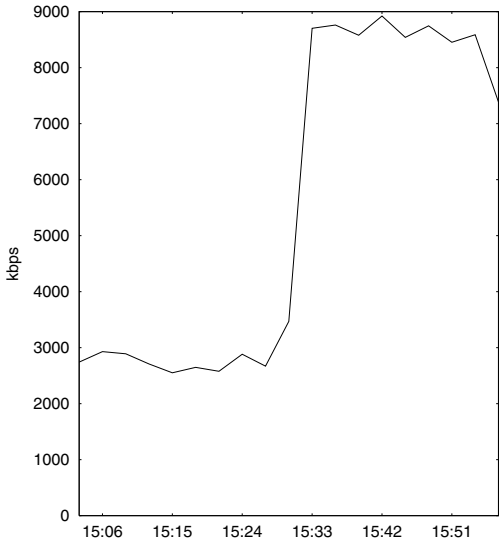

図 3.2　帯域変化前後のスループット

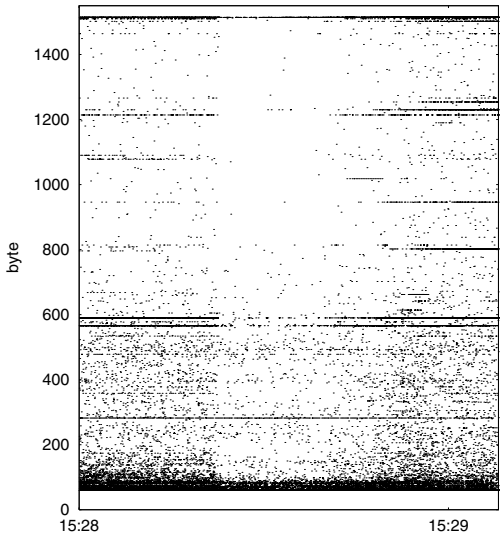

図 3.3　パケット散布図

に対し tcptrace を使用し、各コネクションを抜き出しカウントしたものである。コネクションとは送信側、受信側の IP アドレスとポート番号のペアを指し、ユーザやアプリケーションの要求とそのレスポンスのペアを判別する事が出来る。このグラフから、帯域が切り替わってもコネクションの数は大きく変化していない事がわかる。このことから、図 3.1 で示された帯域変化によるスループットの変化は、コネクション数の増加によってもたらされたものではなく、各コネクションが送受信するデータ量の増加によって引き起こされたものである事がわかる。

帯域変化をまたがって通信が行われたコネクションは、before,after の両方に数えられている。これは tcptrace を使用してコネクションを検出した為である。
また、tcpdump を使用した場合、SYN パケットが送信された時間からコネクションの開始時間を特定することが出来るが、SYN パケットの再送の場合と本当のコネクションを区別することができない。

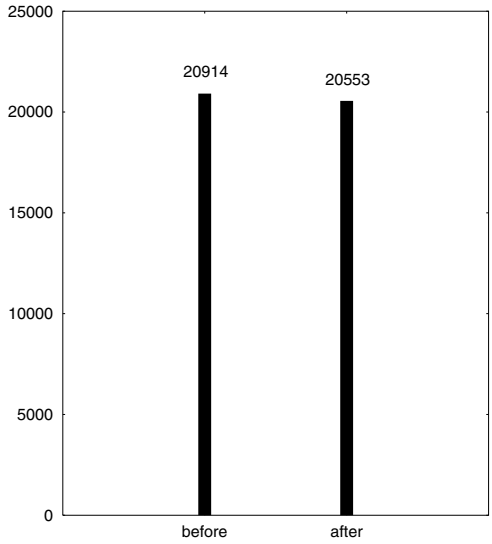

図 3.4 帯域変化前後のコネクション数の変化

3.5 ポート別解析（TCP）

3.5.1 サンプリングによる概要

この章では、TCP に特化した解析を行う。TCP に特化した解析を行うことによって、TCP の振る舞いや TCP を使用したアプリケーションの特性を検証することが出来る。図 3.5 は収集したデータから TCP プロトコルを使用してデータ転送行ったものだけを抜き出したデータのスループットを示している。このデータの抽出には tcpslice を使用しデータのサンプリングを行った後に、tcpdump を使用した。以下にこのグラフからわかる事を示す。
解析に使用した tcpdump の機能には特定のプロトコルのデータを抜き出す機能も付随しているため、ポート別のデータ解析を行うツールとしては優れている。

- HTTP データがトラフィックのほとんどを占めている。
- 帯域変化後、HTTP データが大きく振幅している。
- 帯域変化後、HTTP 以外のデータの顕著な増減は見られない。

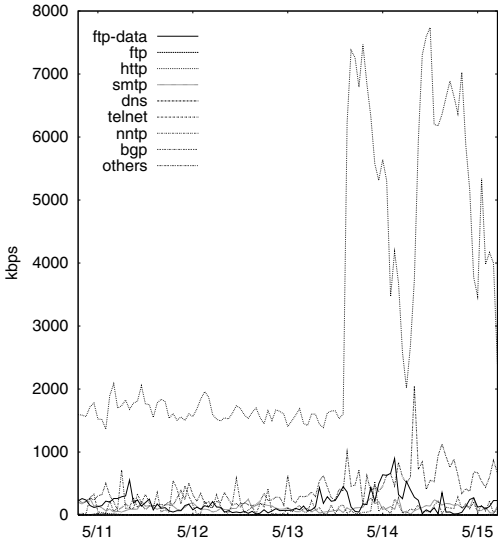

図 3.5 サンプリングによる TCP スループット

3.5.2 1 つのコネクションに対する RTT の変化

RTT とはパケットを送信し応答が返ってくるまでの時間の事である。RTT はネットワークの遅延を把握する上で大きな指針となりうる。図 3.6 は収集したデータから tcptrace を使用しコネクションの詳細な情報を得たものを描画したグラフである。その際、正常終了したコネクションのみを対象とした。これは、送信側と受信側の間で交換されたデータの時間から RTT を計算する。そのため、正常終了したコネクションでなければ RTT の正しい計測が出来ないからである。
また、ここで示される RTT は各コネクションの平均 RTT である。そのため、帯域変化前と変化後の RTT をはっきり分別することができない。以下、このグラフからわかる事を示す。

- 帯域が変化しても RTT は変化しない。
- RTT が 500msec 弱のコネクションが比較的多い。

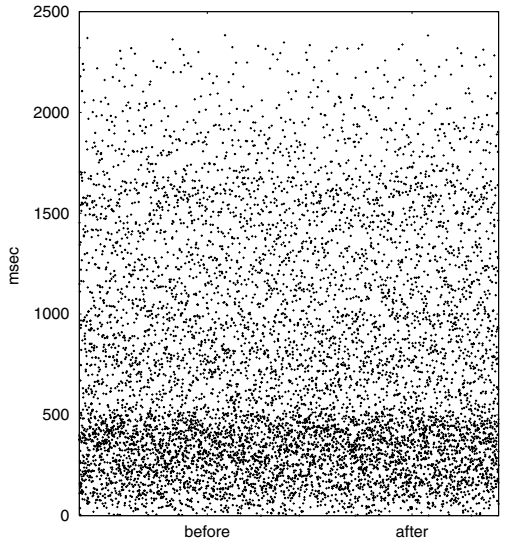

図 3.6　帯域変化前後 5 分間のコネクション平均 RTT

3.5.3 Reset と Fin の変化

ネットワークが混雑している場合、要求に対する応答が遅くなる。そのため Reset で終了するコネクションが増加する。帯域が切り替わる前後 5 分間のデータを抽出し、tcptrace を使用し、その中に含まれる全ての TCP のコネクションを検出した。その中で、Reset で終了したコネクションと Fin で正常終了したコネクションの数をグラフにしたものが図 3.7 である。
以下、このグラフからわかることを示す。

- unidirectional に分類されるデータが多い。
- 帯域が変化しても Fin と Reset の比率はあまり変化しない。

このことによって、対象とした WIDEinternet の国際線を通るデータは非対称な経路を通る事が多いが推測できる。

3.6 ポート別解析（UDP）

3.6.1 サンプリングデータによる概要

図 3.8 は収集したデータから UDP プロトコルを使用してデータ転送行ったものだけを抜き出したデータのスループットを示している。このデータの抽出には

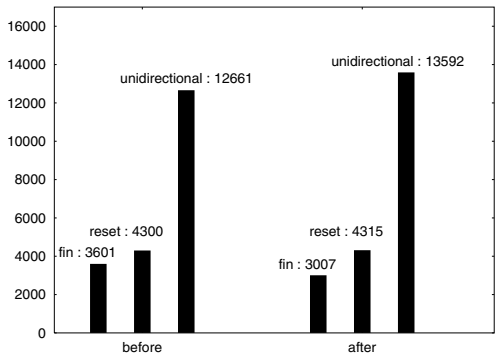

図 3.7　Reset と Fin の変化

tcpslice を使用しデータのサンプリングを行った後に、tcpdump を使用した。解析に使用した tcpdump の機能には特定のプロトコルのデータを抜き出す機能も付随しているため、ポート別のデータ解析を行うツールとして適している。

以下にこのグラフからわかる事を示す。

- DNS データが多い。
- Others に分類されるデータが多い。
- 帯域が変化した後も著しくデータ転送量の増減したポートは見られない。

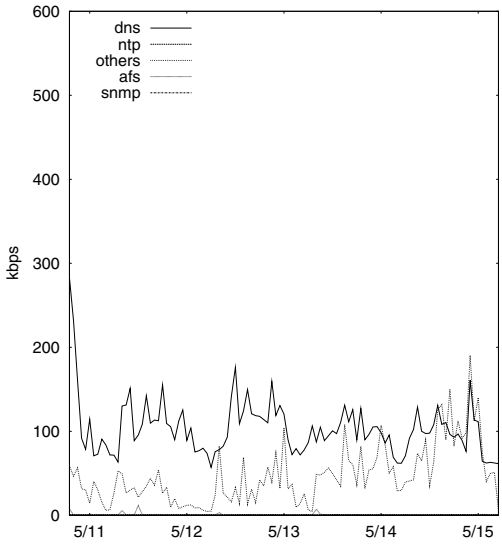

図 3.8　サンプリングによる UDP スループット

表 3.1 最近のアプリケーションと使用ポート番号の一覧

ポート番号	アプリケーション
2300-2400	DirectX 対応ネットワークゲーム
2425	IP Messenger
4000	ICQ
5000	Ultima Online
5001-5500	hotline
6112	Diablo,StarCraft
6970-6999	QuickTime4
7000	VDO Live
7648-7652	CU-SeeMe
27910	Quake

3.6.2 最近のアプリケーションとポートの分布

前章で UDP のポート別トラフィックを表したが、「その他」に分類されてしまうデータが多い。しかも、更に詳しく「その他」に分類されるデータを調査してみたが、特に大量のデータを送受信しているポート番号は存在しない。表 3.1 に最近のアプリケーションと使用ポート番号の表を示す。ただし、これら特定のアプリケーションが使用しているポート番号のデータが多いという事もない。

これらのことから、インターネット上で様々なアプリケーションが使用されるようになった事が推測できる。

3.7 まとめ

本章では、帯域増加時のトラフィックデータから、解析と考察を行なった。解析は、TCP と UDP について、コネクションの数やデータの量、全体からの割合などについて行なった。これにより、帯域の変化が影響を与えるもの、与えないものの二種類に分類できた。

- 影響を受けるもの
  - 全体のデータ量
  - TCP のデータ量
- 影響を受けないもの
  - UDP のデータ量
  - TCP コネクションの数
  - TCP コネクションにおける RTT

さらに、以下の二つの事が考察できた。

- well-known でない UDP ポートを使用するアプリケーションが増えてきた事
- TCP コネクションの RTT は 500ms 以内のものが比較的多い

トラフィックの解析結果には、様々な要因が関わってくるため、一概に理由を述べる事はできない。また、解析手法の焦点を変える事で、考察できることも大きく変化する。更なる解析を行なう事が重要である。

第 4 章 A TRAFFIC PATTERN MATCHING TECHNIQUE

4.1 INTRODUCTION

The measurement and management of the Internet is a challenge that needs to be addressed. Internet traffic Measurement technologies provides the base for effective network management and operation, e.g. locating bottlenecks, planning networks, managing QoS, and etc. Yet there is to date no generic solution to the Internet measurement problem. [24] has attempted to construct a globally scalable Internet measurement facility. The IETF [25] WG is defining the framework for distributed management. The IETF [26] WG is defining the metrics for measuring the Internet. The absence of a measurement nfrastructure has slowed down the deployment of informed and intelligent management and control.

The size, spread and heterogeneity of the Internet means that there will be parts of it which will appear like a black box. E.g. An ATM backbone or the telephone network which supports the IP network. In general one will not have access to management information related to the internal state of the black box. In such cases one can only

build a picture from the input and output characteristics.

One of the major problems of traffic measurement in distributed environment is synchronization of the information observed at different independent point. The time stamps at different point are in general not synchronized. This makes it difficult to build a picture of the functioning of the intervening network. We cannot say how much traffic was handled by the network, at a given interval of time. [26] broached the importance of the clock issue.

Global time synchronization is one of the solutions. For this purpose, NTP [27] can be used. It provides statistical clock synchronization. GPS can also being used for long distant end-to-end measurement across a ocean [28]. Another NTP-like approach, centralized clock management, is considered for a LAN/MAN environment [29]. However, those are either statistical or require special devices and are thus not very appropriate in the context of Internet Measurement.

To avoid this problem, we propose the novel idea for synchronization of measurement information by using the traffic characteristics at each observed point. The technique uses pattern matching of the characteristics. This technology will synchronize the information observed at independent points. In this paper, we explain the concept and show the experimental results.

4.2 BLACK-BOX IN MEASUREMENT

Historically, the structure of Internet was simple. It typically consisted of one broadcast (shared) media and several connected nodes. In such cases, it is relatively easy to measure the network characteristics by using single traditional tools like [10] or [30].

However, the network has evolved to include many types of elements and sub networks. The observation points too are distributed. An observer can at best collect information from several points to know the network usage and characteristics across several segments.

Fig. 4.1 shows the common monitoring environment. The network consists of switches and nodes. Each interface may be monitored. The links may be monitored using devices like RMON.

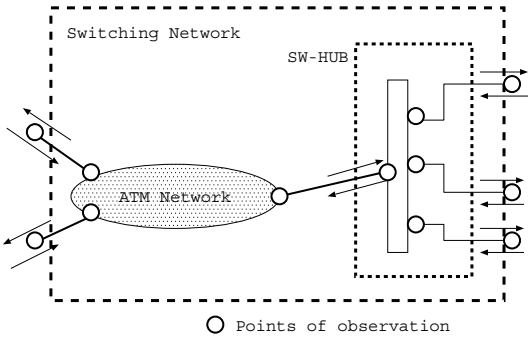

図 4.1　Black-Box in the Internet

4.3 INFORMATION SYNCHRONIZATION

In this section, we explain the necessity and significance of synchronizing the information observed at different independent points. For instance, Fig. 4.2 shows a simple example of the necessity of synchronization. Say one is measuring the traffic from *Net A* to *Net C*. N_1 is the traffic observed at OP_1. N_2 is the traffic observed at OP_2. Now, N_1, N_2 represent the traffic passing through *NetB* (including traffic originating and destined to B), However, to get an exact picture of the traffic through *Net B*, N_1 and N_2 need to be synchronized.

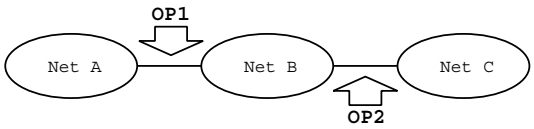

図 4.2　Observation point

Furthermore, we can also find out the number of dropped packets in *Net B* using incoming/outgoing traffic observed at OP_1 and OP_2, if we can know the delay caused by *Net B*.

For these measurements, the accurate time synchronization is indispensable.

Among the candidates for time synchronization, NTP or GPS are worth mention. NTP is a protocol with an algorithm to synchronize the clocks connected to the Internet. GPS provides more accurate time than NTP using the signals from satellites.

However, the former is statistical and the later requires the special devices. Furthermore, some equipment like HUBs, PCs etc. may not have the capability of handling and/or using such special devices.

In this context we propose a way to synchronize independent information by traffic pattern matching. This allows the information observed at independent points to be compared and collated.

4.4 TRAFFIC PATTERN MATCHING

4.4.1 HETEROGENEOUS NETWORKS

Today's networks are characterized by their heterogeneity. For instance, in a LAN environment, there are few server machines and many client nodes typically. So, the amount of traffic generated in such a network leans heavily to the server side. Fig. 4.3 shows the sample of a heterogeneous usage of network. The traffic observed at the different points show similarity in their patterns.

4.4.2 COMPARE THE TRAFFIC PATTERN

Fig. 4.4 shows a network model, which has n links, and each link has paths for both incoming and outgoing. We correlate them by following pattern matching method.

MODELING THE TRAFFIC PATTERN

To capture the transition pattern, we make the vector model, which represents the amount of traffic of each time-slot.

The characteristics for a duration t, which is divided into slots of size. So it has $m = \Delta/\delta$ elements (Fig. 4.5). In the case of the network as illustrated in Fig. 4.5, the characteristics is

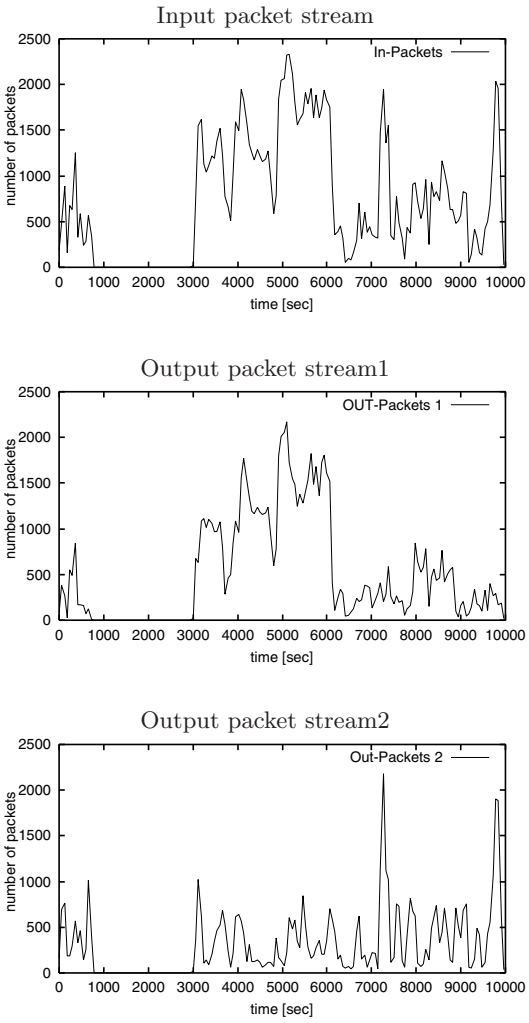

図 4.3 Traffic transition similarity

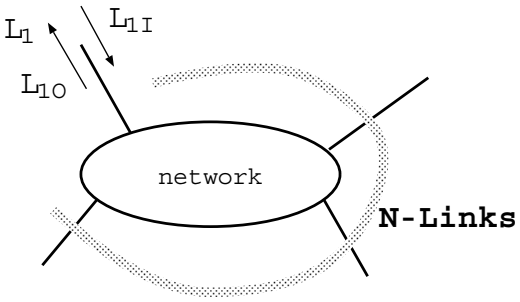

図 4.4 N-Links connected network

defined as follows:

$$L_{i_I} = (a_{1i}, a_{2i}, a_{3i}, \cdots, a_{ti}, a_{\frac{\delta}{\Delta}i})$$

$$L_{j_O} = (b_{1j}, b_{2j}, b_{3j}, \cdots, b_{tj}, b_{\frac{\delta}{\Delta}j})$$

L_{i_I}: number of input packets at Link i

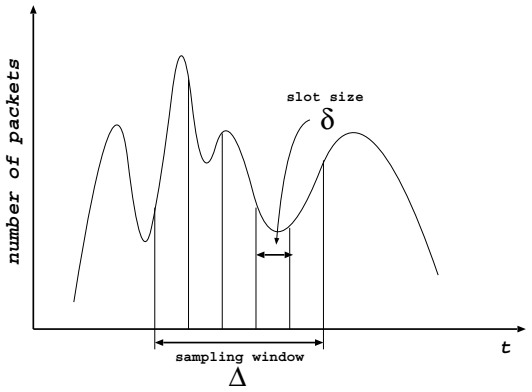

図 4.5 Traffic pattern model

L_{j_O}: number of input packets at Link j

COMPARING THE PATTERNS

To compare the similarities of the characteristics, correlation coefficient between incoming and outgoing traffic is used. That is calculated as follows.

where

σ_{in}: standard deviation of L_{i_I}

σ_{out}: standard deviation of L_{j_O}

$r_{ij}(L_{i_I}, L_{j_O}) =$

$$\frac{1}{n\sigma_{in}\sigma_{out}} \times \sum_{k=-n}^{n} (L_{i_I}(t+k) - \overline{L_{i_I}})(L_{j_O}(t+k) - \overline{L_{j_O}})$$

$\overline{L_{i_I}}$ average of L_{i_I}

$\overline{L_{j_O}}$ average of L_{j_O}

r_{ij} are calculated for all combinations of i, j. Then the max value of r_{ij} is selected from all of them.

4.5 RESULT OF MEASUREMENT

This section shows the remarkable results of our proposed method. We evaluate the method on two types of operational network. The first is a 100Mbps FDDI backbone network, [31] (Tohoku OPen Internet Community), which provides connectivity to some big universities, colleges, and museums. The other is 10Mbps Ethernet Internet-eXchange, [32] (Tohoku Regional Internet-eXchange, which provides connectivity to some commercial ISPs. TOPIC has $n = 20$ links, and TRIX has $n = 5$ links.

The calculation is based on the data at the interfaces, which are entry points of the FDDI/Ethernet. We monitored the traffic in the networks using TCPDUMP to evaluate the accuracy of our technique.

4.5.1 OVERALL EVALUATION

Table 4.1 shows the overall result of our proposed method. The success rate shows the rate of the exact matches between the logged packets and calculated packets by proposed method. The interval is the sampling rate, that is δ in fig. 4.5. The value shown in the table is derived heuristically.

表 4.1 Overall evaluation

	interval δ (sec)	success rate (%)
100M-FDDI	0.01	84.6
10M-Ethernet	0.1	91.9

Fig. 4.6 shows the result of the effect of our proposed method. The top graph represents the result of our method, the next graph shows the logged data. The two graphs are compared in the third graph. They are almost same. The lowest graph is a close-up of the area of mismatch.

4.5.2 WHAT IS THE TRAFFIC IN THE NETWORK?

Fig. 4.7 shows a 5 minutes throughput ob the backbone. This information could not be obtained using existing measurement technology without some synchronizing technique.

4.5.3 WHO IS USING THE BANDWIDTH

Fig. 4.8 shows the ranking of the usage of the bandwidth. Lines with rectangles represent the calculated values. Lines with circles are based on data derived from logs. The two are closely

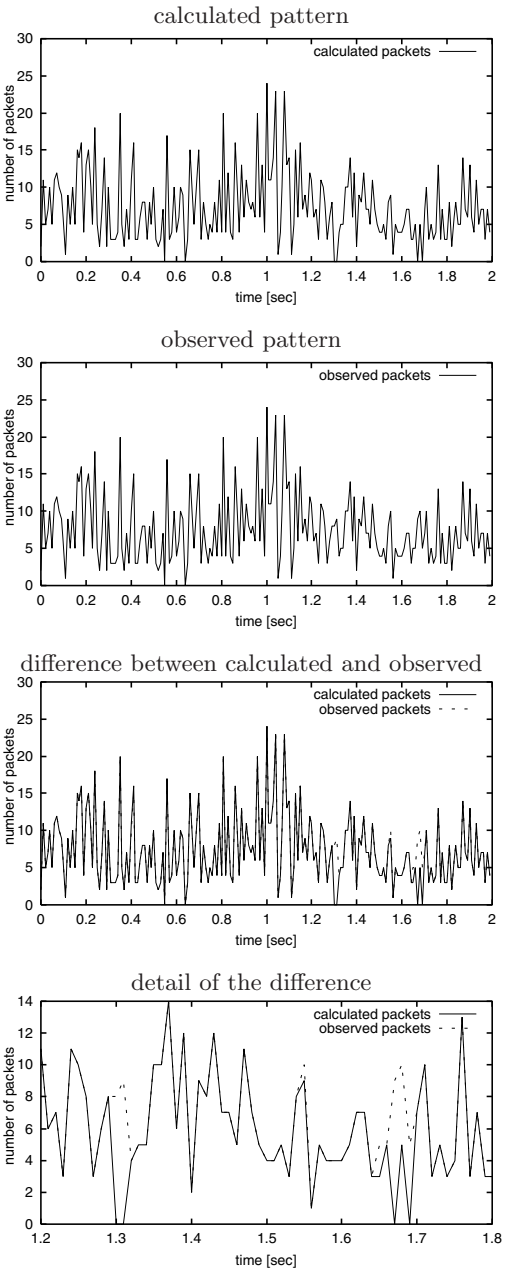

図 4.6 Sample synchronizing result

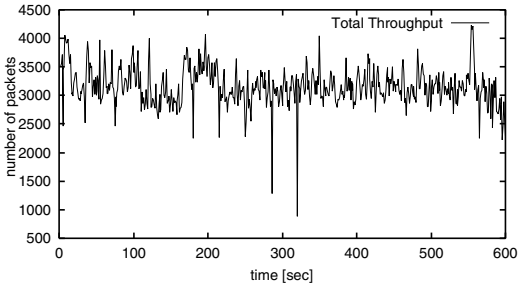

図 4.7 Network throughput

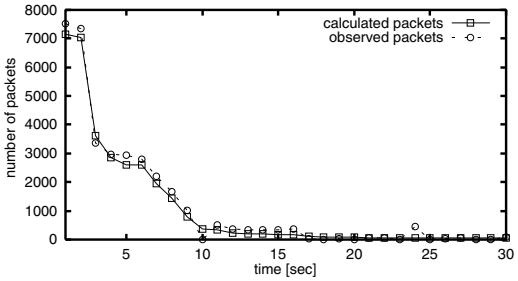

図 4.8 Ranking of the network usage

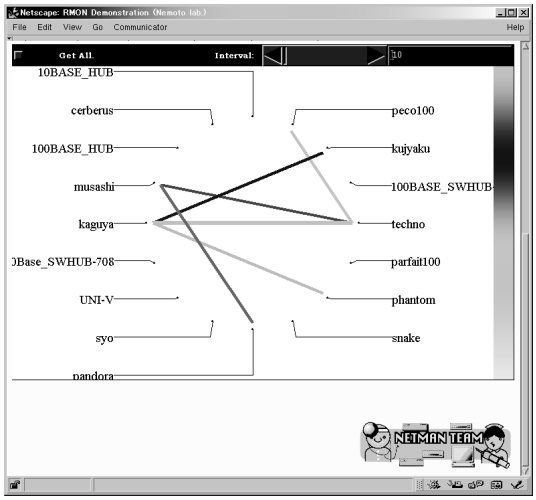

図 4.9 Matrix monitor for SW-HUB

similar. Significantly, the top 10 links are heavy consumer of the network resources, especially top 2, that information is useful for future topology design or dynamic rerouting techniques.

4.5.4 SWITCHING HUB VIEW

Fig. 4.9 shows the sample application of our method. This is a matrix view of a Switching HUB. The information observed at each port is synchronized, so we can see the view of the network.

4.6 ISSUES

This proposed method is a way to synchronize the management information obtained from different points in the network. The method is scalable and works well in the case of networks with small latency. Recent technology has led to very high-speed and high-capacity networks. This technol-

ogy is expected to be useful in the measurements of such networks.

Since the proposal does not involve the addition of additional devices the method can be implemented using current management framework.

4.7 CONCLUSION

In this paper, we discussed strong requirement of measurement of the present day Internet. We have also explained the difficulty in doing the same. This problem makes effective network management and control difficult.

We have proposed a methodology for information synchronization. We have also established its efficacy by deploying a pilot version of our method on an operational network.

第 5 章　パケットモニタによる WWW サーバの性能計測システム

5.1 はじめに

World Wide Web(WWW) は社会に急速に広まっている．現在，3,500 万を超える WWW サーバが存在しており [33]，WWW トラヒックは Internet 上のトラヒックのおおよそ 80%を占めるようになった [34].

この急速な普及にともない，様々な種類の WWW サービスが行われるようになった．例えば，WWW を用いて航空チケットや宿泊施設の予約が可能となった．また，イベントの中継にも WWW が利用されるようになった．例えば，1998 年の長野オリンピックやサッカーのワールドカップなどがあげられる．

WWW サービスの充実にともない，サービス利用者はサービス品質を意識するようになった．サービス品質とは，WWW コンテンツの内容やコンテンツの表示が開始されるまでの時間もしくは表示が終了するまでの時間である．特に，サービス利用者は「より速くコンテンツが表示される」ということを要求するようになった．

一方，WWW サーバ管理者の立場では，利用者の要求を満たすより良いサービス品質を提供するために，サーバシステムの性能を知る必要がある．サービス品質を正確に測定するには，1) サーバシステムの全体の挙動を測定可能，2) 運用中のシステムの性能を計測可能，3) 計測がサーバシステムに影響を与えない，といった要件が必要である．しかし，ログ解析手法やベンチマークテストといった，これまでの性能解析手法ではサービス品質について考慮されていない．そこで，我々はパケットモニタによる WWW サーバ性能計測手法を提案する．パケットモニタによる計測では，外部からサーバシステムの性能計測が可能であるため，上記の要件を満足することができる．

本論文では，パケットモニタによる WWW サーバシステム性能計測手法を提案し，その手法を用いたシステム Enhanced Network Measurement Agent(ENMA) の設計および実装を行う．そして，実際に運用されているサーバシステムの性能計測を行い ENMA の有効性を明らかにする．

5.2 パケットモニタによる WWW サーバ性能計測手法の提案

本章では，サービス品質と WWW サーバシステムの性能との関係について述べ，従来の性能計測手法の問題点および我々が新たに提案する手法について説明する．

5.2.1 サービス品質

WWW サービスの品質には以下に示す 4 つの項目が考えられる．

1. **コンテンツの質と量**
   コンテンツの質と量には，各サイトの特徴が現れる．WWW サーバの性能は，これら 2 つの項目に対して無関係であるので，本研究では考慮しない．

2. **クライアントが WWW コンテンツを要求してから表示が開始されるまでの時間**
   クライアントが WWW コンテンツを要求してから表示が開始されるまでの時間は，様々な要因によって変動する．WWW サーバの性能について考えた場合，この時間が短いということはサーバの性能が高いことを示している．

3. **コンテンツが表示され始めてから表示が終了するまでの時間**
   コンテンツが表示され始めてから表示が終了するまでの時間も，先の項目と同様のことが言える．WWW サーバの性能について考えた場合，この時間が短いということはサーバの性能が高いことを示している．

4. **どれだけ多くの利用者に対して同時にサービスを提供することができるか**
   WWW サーバの性能が高ければ高いほど，多くの利用者に同時にサービスを提供できると考えられる．

以上の WWW サービスの品質についての検討の結果，WWW サーバ管理者に必要な項目は，2,3,4 である．これは，利用者は「より速くコンテンツが表示される」ということを最も期待しているからである．

5.2.2 性能指標

本節では，WWW サーバシステムの性能を代表するいくつかの性能指標について述べる．

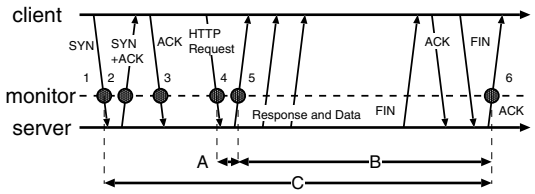

図 5.1 性能指標

同時コネクション数 (N_c)

同時コネクション数は WWW サーバシステムが同時に処理している HTTP コネクションの数である．この項目は，サービス品質の「どれだけの利用者に対して同時にサービスを提供できるか」に関連する．

コネクション継続時間 (T_c)

コネクション継続時間は，TCP コネクション確立から終了までの時間である．すなわち，クライアントによってサーバへ送信された最初の SYN パケット (図 5.1 の 1) を観測してから，最後の ACK パケット (図 5.1 の 6) を観測するまでの時間である (図 5.1 の区間 C)．この項目は「コンテンツが表示され始めてから表示が終了するまでの時間」に関連する．

応答時間 (T_r)

応答時間の定義は，クライアントからの HTTP 要求パケットを観測してから，サーバの最初の HTTP 応答パケットを観測するまでの時間である．我々が定義した応答時間は図 5.1 の区間 A である．この項目は「クライアントが WWW コンテンツを要求してから表示が開始されるまでの時間」に関連する．

データ転送時間 (T_d)

データ転送時間は，最初のデータパケットをサーバが観測してから，最後の HTTP データパケットを観測するまでの時間と定義する (図 5.1 の区間 B)．データ転送時間に FIN パケットとその ACK を含んでいるのは，データが含まれていることが多いからである．この項目は「コンテンツが表示され始めてから表示が終了するまでの時間」に関連する．

5.2.3 従来の性能計測手法の比較

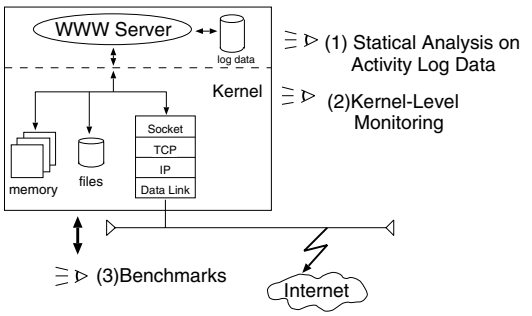

図 5.2 従来の WWW サーバ性能計測方法

WWW サーバの性能計測方法は図 5.2 に示すように以下の 3 手法が考えられる．

1. ログ解析
2. カーネルレベルモニタリング
3. ベンチマークテスト

本節では 5.2.2 における性能指標に着目し，これらの手法の利点および欠点について述べる．

1. ログ解析手法　この手法は，WWW サーバが出力したログの統計解析を行うもので，WWW サーバの性能解析手法として一般的に用いられている．こ

の手法では，クライアントのアクセスパターンやリクエスト処理時間などを知ることができる．しかし，アプリケーション層で測定しているために，サーバ管理者に必要な，応答時間，データ転送時間，同時コネクション数は測定することができない．

2. カーネルレベルモニタリング　この手法では，システムコール内での I/O の数，シグナル処理の数，ファイルシステムオペレーションの数などを明らかにできる．しかし，この手法でサーバ管理者に必要な，応答時間，データ転送時間，同時コネクション数を測定するには，カーネルへの変更が必要になる．商用 OS や専用ハードウェアを用いたサーバなどを用いた場合，カーネルの変更は非常に困難となるので，管理者の必要とする性能指標を測定することは事実上不可能である．また，これは計測がサーバシステムに影響を与えるので WWW サーバシステム自体の性能劣化を引き起こす可能性がある．

3. ベンチマークテスト　WWW サーバ計測の代表的なベンチマークプログラムとして，SPECWeb96 [35] や WebStone [36] がある．ベンチマークテストでは，サーバ管理者の必要とする性能指標を測定することは可能である．しかし，ベンチマークテストでは特別な環境でテストを行うので，実際の運用時のデータとは同じ結果が得られない．また，運用中の大規模 WWW サーバにおいてベンチマークテストのためにサービスを中断することは困難である．

5.2.4 パケットモニタによる WWW サーバ性能計測手法

前節で述べた方法では，サーバ管理者が必要とする性能指標を測定することは困難であった．そこで我々は，WWW サービスが WWW サーバシステムにおいて送受信されるパケットによって構成されることに着目し，パケットモニタによる WWW サーバシステムの性能計測手法を提案する．

WWW サービスは HTTP 上で行われており，HTTP は TCP を用いている．TCP はパケットを観測することで TCP のコネクション数や，その状態遷移を監視することができる．

本手法ではサーバシステムと同一のネットワークセグメント上に計測ホストを設置し，パケットモニタリングを行い，対象となるサーバシステムの外部から性能計測を行う．

表 5.1　性能評価手法の比較

	Log Analysis	Kernel Level Monitor -ing	Bench -mark Test	Packet Monito -ring
A	×	○	△	○
B	○	×	×	○
C	○	△	×	○
D	△	×	○	○

新たに提案する手法には以下に示す特徴がある．

A. トランスポート層での挙動を観測できる　パケットモニタリングを行い，TCP コネクションを解析することによって，トランスポート層での挙動をカーネルレベルモニタリングと同等，あるいはそれに近い精度で観測することができる．

B. WWW サーバシステムに影響を与えない　本手法はパケットモニタリングによる外部からの性能計測である．このため，本手法の計測によって，WWW サーバシステムに影響を与えない．

C. 運用中のサーバシステムを測定できる　本手法は運用中の WWW サーバシステムに対するコネクションの解析を行うことを想定している．このため，性能計測のために運用の停止や，運用状態とは別の特別な環境を用意する必要がない．さらに，運用状態のデータを得ることができるため，より正確なサービス品質の分析を行うことができる．

D. 特定のサーバシステムに依存しない　本手法では WWW サーバシステム自体に変更を加える必要がない．このため，WWW サーバシステムが商用 OS で運用されているといった場合にも計測を行うことができる．

従来の手法および我々の提案した手法の比較を表 5.1 にまとめた．

5.3 システム設計

本章では，提案した計測手法を用いた WWW サーバ計測システムの設計について述べる．また，

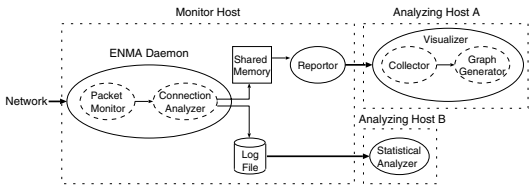

図 5.3 システム構成

本システムを Enhanced Network Measurement Agent(ENMA) と名付けた．ENMA は，ENMA デーモン (ENMA Daemon) と性能解析プログラム群 (Performance Analysis Workbench) の二つの要素から構成されている (図 5.3).

5.3.1 ENMA デーモン

ENMA デーモンはリアルタイムにパケットモニタリングを行う．この ENMA デーモンはパケットモニタモジュールとコネクションアナライザモジュールの 2 つのモジュールから構成されている．

パケットモニタモジュールは ENMA が設置されているネットワーク上の HTTP コネクションを構成するパケットのモニタを行う．モニタされたパケットから，IP データグラムを抽出し，コネクションアナライザへ渡す．

コネクションアナライザは以下の機能を提供する．

- HTTP コネクション上のパケットの到着時間を記録する．
- 各々の TCP コネクションで観測された IP データグラムのペイロード長を記録する．また，IP データグラムのペイロードの総数を TCP コネクションでの上りと下りを区別して記録する．
- TCP ペイロードについてもコネクションの上りと下りを区別して，ペイロード長を記録する．
- 1 つのコネクション情報を 1 つのエントリとしてログファイルに出力する．
- リアルタイムにデータの視覚化と解析を行うために，性能解析プログラム群へ共有メモリを介してデータを提供する．

5.3.2 性能解析プログラム群

性能解析プログラム群は ENMA デーモンが提供するデータの様々な解析を行うプログラムの集合であり，以下に示す 2 種類に分類される．

- コネクション時間や同時コネクション数の分布といった，全体を読み込まないと解析できないデータの算出のために，ENMA デーモンの性能解析プログラムを用意した．これらのプログラムは ENMA デーモンのログファイルを読み込み，データの解析をバッチ形式で行う．
- これまでに処理したコネクション数や，現在処理しているコネクション数といった，現在の WWW サーバの処理状態を知るために，リアルタイムな性能解析を行うプログラムを用意した．このプログラムは ENMA デーモンが用意している共有メモリ領域からデータを読み込む．

5.4 実装

我々は，様々な UNIX プラットフォームで動作するシステムを目標として本システムを実装した．ENMA デーモンは C 言語，性能解析プログラム群は AWK, C 言語で実装した．開発および実行環境は FreeBSD 2.2.7 や IRIX6.3 である．この章では ENMA の実装について説明する．

5.4.1 パケットモニタモジュール

パケットモニタモジュールでは，ネットワーク上のパケットを監視するために，Lawrence Berkeley National Laboratory(LBNL) のパケットキャプチャライブラリ (libpcap [37]) を用いている．このライブラリは BSD 上の Berkeley Packet Filter(BPF [11]) や System V での Network Monitoring Protocol(the packet snooper) の機能を利用し，統一された API を提供している．このライブラリを用いることによって，IP 層でのパケットが得られる．また，多くの UNIX プラットフォームで ENMA を動作させることが可能となる．

5.4.2 コネクションアナライザ

コネクションアナライザでは，各々の TCP コネクションの TCP 状態遷移の監視を行う．このモジュールではシーケンス番号，確認応答番号，TCP ヘッダ

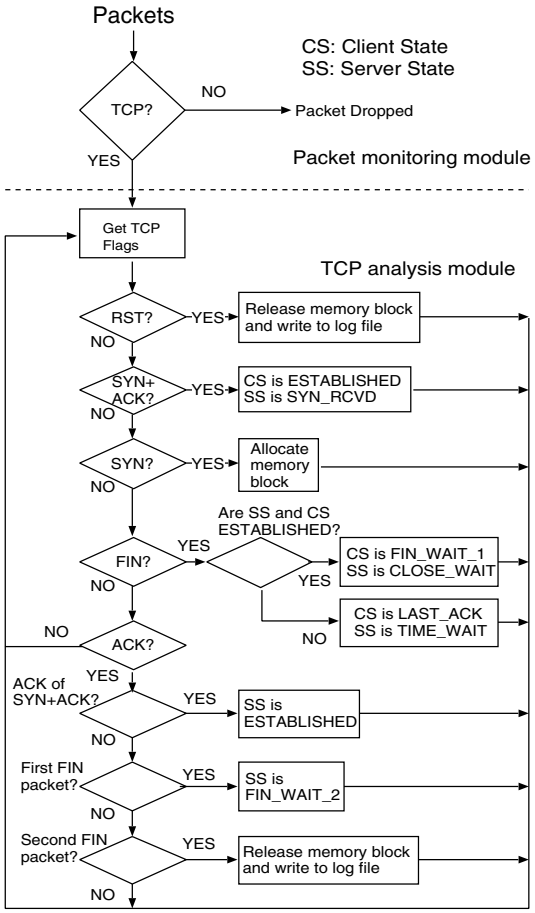

図 5.4　コネクション解析アルゴリズム

のフラグを用いてサーバとクライアントの状態を追跡する．コネクションアナライザの用いるアルゴリズムを図 5.4 に示す．

図 5.4 に示すように，パケットが到着すると，最初に TCP ヘッダのフラグを見る．フラグの種類によって以下のような処理を行う．

1. RST の場合，コネクションが終了すると判断するので，メモリブロックを解放し，ログファイルにコネクション情報を出力する．
2. SYN+ACK の場合，サーバの状態変数を SYN_RCVD に遷移させ，クライアントの状態変数を ESTABLISHED に遷移させる．
3. SYN の場合，メモリブロックを割り当て，状態の初期化を行う．
4. FIN の場合は，サーバの状態とクライアントの状態によって，状態を遷移させる．
   (a) サーバとクライアントの状態が ESTABLISHED の場合，クライアントの状態を FIN_WAIT_1 へ遷移させ，サーバの状態を CLOSE_WAIT へ遷移させる．
   (b) サーバとクライアントの状態が ESTABLISHED でない場合は，クライアントの状態を LAST_ACK へ遷移させ，サーバの状態を TIME_WAIT へ遷移させる．
5. ACK の場合
   (a) SYN+ACK に対する ACK であるなら，サーバの状態を ESTABLISHED へ遷移させる．
   (b) 始めの FIN パケットに対する ACK であるなら，サーバの状態を FIN_WAIT_2 へ遷移させる．
   (c) 2 つめの FIN パケットに対する ACK であるなら，コネクションが終了すると判断して，メモリブロックを解放しデータをログファイルへ出力する．

このアルゴリズムでは，コネクションアナライザは SYN パケットを観測した時に，各々の TCP コネクションの状態を記録するためにメモリブロックを割り当てる．TCP コネクションが正常に終了する場合，メモリブロックは解放されデータファイルへコネクションの情報が出力される．正常に終了しなかった場合，メモリブロックが解放されない．そこで，コネクションアナライザは，ある一定間隔でメモリブロックを検査し，古いメモリブロックを解放する．

5.4.3 性能解析プログラム群

図 5.3 に示すように，性能解析プログラム群には 3 つのプログラムがある．統計データ報告プログラム (Reporter) および視覚化プログラム (Visualizer) は，リアルタイムな解析のために用いられる．ENMA デーモンは ENMA システム内の多くの計算機資源を用いるので，ENMA デーモンと視覚化プログラムの両方を同一の計算機で実行することは，ENMA システムの性能劣化を引き起こす可能性がある．最悪の場合，ENMA デーモンがネットワーク上のパケットを喪失する可能性がある．そのような状況を回避するために，リアルタイム解析を行うプログラムを統計データ報告モジュールと視覚化モジュールの 2 つのモジュールに分割し，視覚化モジュールは異なる計算機上で実行することを可能とした．

統計解析プログラム (Statistical Analyzer) は ENMA デーモンが出力したログファイルの統計解析を行うプログラムである．以下に各プログラムの詳細を述べる．

統計データ報告プログラム

統計データ報告モジュールは図 5.3 に示すように，共有メモリから統計データを得る．後で述べる視覚化プログラムの要求を受け付ける．

視覚化プログラム

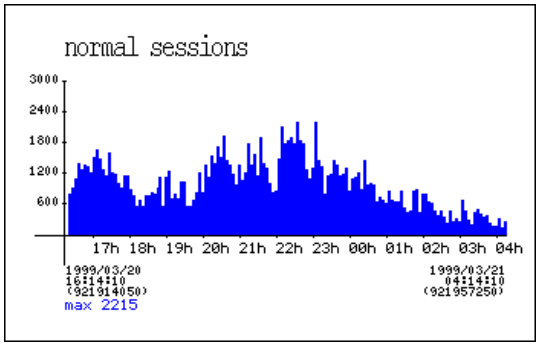

図 5.5 正常セッションの経時変化 (視覚化モジュールの例)

視覚化モジュールは様々なデータを表示するためのプログラムである．データは統計データ報告モジュールに対して要求を発行することで総計データを入手する．その入手した統計データより，図 5.5 のような性能指標に関するグラフを生成する．

統計解析プログラム

統計解析プログラムは，ENMA デーモンが出力したログファイルの統計解析を行い，コネクション継続時間や同時コネクション数などの頻度分布を集計する．

5.5 ENMA の有効性の検証

ENMA の有効性を確かめるために，ENMA を実際に動作している WWW サーバに対して適用し，有効性の検証を行った．

5.5.1 WWW サーバの処理能力差の計測

我々は，ENMA を用いることで WWW サーバの処理能力差を計測することが可能かどうかの検証を行った．2 章で述べたように，応答時間 (T_r) は WWW サーバの性能を表している指標の一つと考えられる．そこで，我々は処理能力の異なる 2 つのサーバシステムに対して ENMA を適用し，これらの値を測定した．

システム構成

この実験では，PentiumII 200MHz のホスト A と 80486DX2 66MHz のホスト B，2 台の WWW サーバを用意した．両ホストとも OS は FreeBSD 2.2.7 で WWW サーバアプリケーションとして Apache 1.3.1 を用いた．サーバ，クライアント，モニタホストを同一のセグメントに接続し実験を行った．

この実験では，ランダムに WWW サーバ上のオブジェクトにアクセスした．そして，サーバに約 200 のオブジェクトを置き，並行してすべてのオブジェクトへアクセスを行った．1 回の測定で約 10,000 アクセス行い，T_c, T_r を測定した．

結果

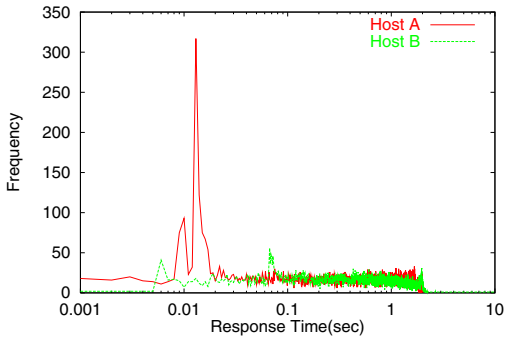

図 5.6 応答時間の頻度分布

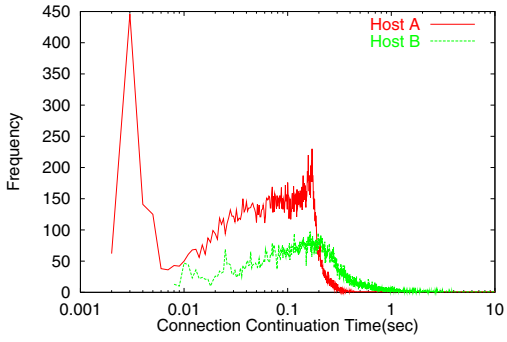

図 5.7 コネクション継続時間の頻度分布

図 5.6,5.7 に測定の結果を示す．図 5.6 は応答時間 (T_r) の頻度分布である．図 5.6 より，ホスト A

では T_r が 20 msec 付近に最も多く分布している．しかしホスト B ではピークは見当たらず，T_r は 50 msec よりも長くなっている．したがって，図 5.6 からはホスト A の方が高速に処理していることがわかる．図 5.7 はコネクション継続時間の頻度分布である．図 5.7 より，ホスト A ではコネクション継続時間 (T_c) が 3 msec 付近に最も多く分布しており，ホスト B の T_c は 200 msec 付近に分布している．これから，ホスト A の方がホスト B よりもコネクションに関する処理を高速に行っていると言える．

これら 2 つのグラフから，ホスト A とホスト B との性能の違いを見ることができる．ENMA により，容易に WWW サーバの性能の違いを示すことができた．

5.5.2 大規模 WWW サーバの計測

ENMA が実際に運用されている WWW サーバの性能計測を行うことができるかどうか確かめるために，第 80 回全国高校野球選手権大会のインターネット中継に用いられた WWW サーバを計測した．この WWW サーバでは，1 日に数千万アクセスが得られた．

システム構成

ターゲットとなる WWW サーバホストは Sun Enterprise 450(CPU は Ultra SPARC II 300MHz 2 個，512MB のメモリ) である．OS は Solaris 2.6 で WWW サーバプログラムは Apache [38] 1.3.1 である．ENMA デーモンが動作しているホストは IBM-PC(PentiumII 300MHz，64MB メモリ)，OS は FreeBSD 2.2.7-RELEASE を用いた．WWW サーバホストと ENMA を同一セグメントに接続し，パケットのモニタを行った．

結果

図 5.8 はコネクション継続時間 (T_c) の頻度分布を示している．WWW サーバによって得られたログの解析では T_c が 1 msec 付近に最も多く分布している．しかし ENMA によって得られたログの解析では，20 msec 付近に最も多く分布している．

WWW サーバによって生成されたログは，そのシステムのアプリケーション層における記録である．そのため，ログにおける T_c は図 5.10 に示したように，accept システムコールが終了してから close システムコールが終了するまでの時間である．この多くが 1 msec 付近に分布しているということは，図 5.10 のこれらのシステムコールが直ちに終了することを示している．例えば，write システムコールに関して言えば，今回測定で用いた Solaris 2.6 の送出ソケットバッファサイズは 8KByte であり，サーバが処理したオブジェクトの大半が 8KByte に収まる (図 5.9) ことから，write システムコールはオブジェクトをソケットバッファに格納した直後に終了することを示している．このようにアプリケーション層における

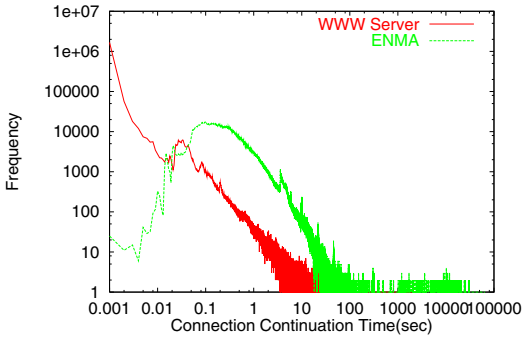

図 5.8　コネクション継続時間の頻度分布

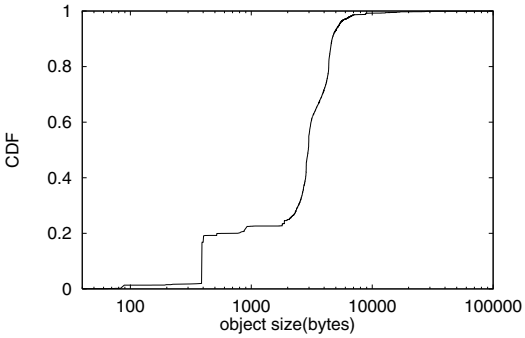

図 5.9　オブジェクトサイズの累積分布

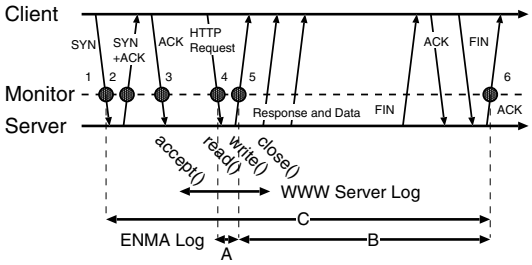

図 5.10　コネクションとシステムコール

サーバログでは正確な T_c は得られない．

一方，ENMA による T_c は，サーバのログによる T_c に加え，TCP 層でのコネクション確立と切断の手続き時間を含んでいる．さらに，close システムコールが呼ばれた後の，OS によるパケット送出処理時間も含んでいる．したがって，ENMA によって測定された T_c は TCP 層でのコネクション継続時間である．よって，ENMA のログファイルの解析結果は WWW サーバのログファイルより正確な性能指標を示したと考えられる．

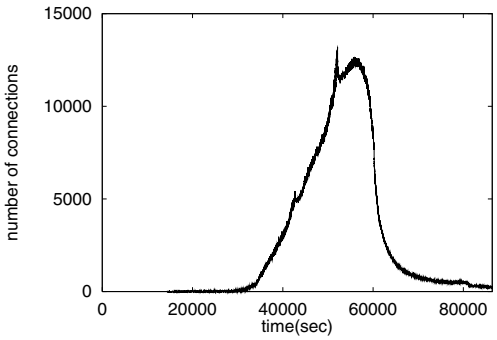

図 5.11 ENMA のログ解析による同時コネクション数

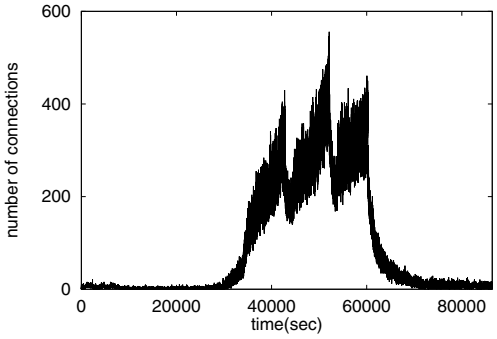

図 5.12 WWW サーバのログ解析による同時コネクション数

同時コネクション数 (N_c) を図 5.11, 5.12 に示す．WWW サーバおよび ENMA によるログのそれぞれの T_c は異なっているので，T_c から算出する N_c の解析結果も異なる．ENMA が報告する T_c と WWW サーバのログによる T_c では，ENMA による T_c の方が長くなっている．T_c が長い場合には時間の重なりが多くなるので，N_c も大きな値となる．

図 5.11 , 5.12 より WWW サーバのログからの解析ではピーク時の N_c は 550 である．しかし，ENMA ではピーク時の N_c は 13,000 である．

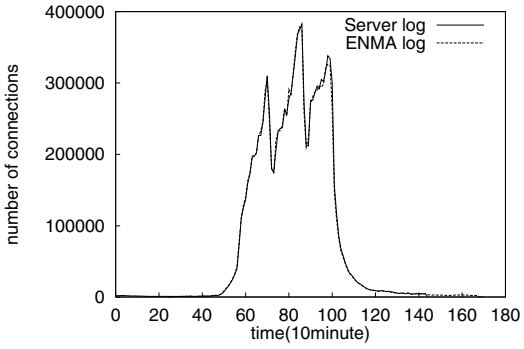

図 5.13 単位時間あたりに受信したコネクション数

ENMA で十分にパケットを受信できたかを検証するために，単位時間あたりに受信したコネクション数を図 5.13 に示す．図 5.13 では，WWW サーバのログと ENMA のログでは，大部分が一致している．したがって，ENMA ではパケット喪失がほとんど発生しなかったと考えられる．

これらの結果より，ENMA を用いることで WWW サーバのログ解析ではわからなかった WWW サーバの性能指標を計測することができた．これより ENMA は実際に運用されている WWW サーバに対して，性能計測を行うことができることが明らかとなった．

5.6 問題点

5.6.1 パケット喪失

モニタリングによるパケット喪失は，正しい性能計測を行うことができなくなるので，パケット喪失は重要な問題である．ENMA における，パケット喪失の解決方法を以下に述べる．

1. サーバホストと同等な高速な計算機を用いる

WWW サーバアプリケーションと ENMA デーモンの計算機に与える負荷を考慮すると，WWW サーバアプリケーションの方が高負荷であると考えられる．したがって同等の計算機を用いた場合には，ENMA デーモンでのパケット喪失を抑制することができると考えられる．

2. オーバヘッドの小さいパケットモニタ機構で動作させる　一般的に UNIX を用いるパケットモニタリングは数百 Mbps や高帯域ネットワークに適用する場合，OS のオーバヘッドに大きく影響される．したがって，Gigabit Ethernet や他の高速ネットワークに ENMA を適用した場合には，オーバヘッドの小さい OS 上で動作させる必要がある．

5.6.2 パケット順序の入れ替わり

別の問題としてパケット到着の順序が入れ替わることがあげられる．現在の実装では，ENMA はシーケンス番号の追跡を行っていない．したがって，パケット順序の入れ替わりを検出することができない．これにより WWW サーバの状態の検出を失敗する可能性があるので，性能計測に問題が生じる恐れがある．

5.7 他のパケットモニタリングシステム

従来のパケットモニタリングシステムとして tcpdump がある．tcpdump はネットワーク上のトラヒックをモニタリングしパケットを観測するための汎用のプログラムである．しかし，汎用のプログラムであるため HTTP に特化した情報を取り出すことができない．また，通信状態をリアルタイムでモニタリングする用途には向かない．そして，汎用であるためプログラムのホストに与える負荷が大きく，高負荷な WWW サーバの計測を行う場合には，パケット喪失の可能性がある．

それに対して，ENMA ではリアルタイムに HTTP に特化したパケットモニタリングを行う．また，コネクション単位でログへの出力を行っているため，ログ出力のオーバヘッドが小さくなる．さらに，ログファイルのサイズを抑制することができる．

5.8 おわりに

WWW サービスではサービスの品質がもっとも重要である．WWW サーバ管理者はサービスの品質を維持するために，サーバシステムの性能を知る必要がある．従来の手法では，サービスの品質に関連する性能計測を行うことは困難であった．例えば，WWW サーバのログ解析では，アプリケーション層での解析であるため，サーバ管理者の要求する性能指標を得ることができなかった．

そこで我々は，WWW サービスが WWW サーバに対するパケットの送受信で構成されていることに着目し，パケットモニタによる性能計測手法を提案した．提案した手法には以下に示すような特徴がある．

- WWW サーバシステム全体の挙動を測定できる．
- WWW サーバシステムに影響を与えない．
- WWW サーバシステムを変更する必要がない．
- 運用中のシステムを測定できる．

我々は，提案した手法を用いたシステム ENMA の設計と実装を行った．ENMA を用いることで，同時コネクション数，応答時間，データ転送時間などの性能指標を測定することができた．そして，ENMA を運用されている WWW サーバに適用し，実際に運用されている WWW サーバシステムの性能計測を行うシステムとしての有効性を示した．