DBS: a powerful tool for TCP performance evaluations

Yukio Murayama[a] and Suguru Yamaguchi[a]

[a] Graduate School of Information Science,
Nara Institute of Science and Technology
Takayama 8916-5, Ikoma, Nara, 630-01, Japan

ABSTRACT

Network performance measurement tools, mainly used for TCP performance evaluations, have serious problems that their applications are limited to single point-to-point configuration and only the mean throughput for long term are measurable. Their major drawback is that they covers only a subset of the entire TCP functions and small variations of network systems.

In this paper, we propose a TCP performance measurement tool called DBS (Distributed Benchmark System) which is aiming to give performance index with multi-point configuration and also in order to measure changes of throughput. It measure the performance of entire TCP functions in various operational environments.

Experiments had been conducted using DBS to measure TCP end-to-end performance with various kinds of networks such as both LANs including FDDI, ATM, and HIPPI over ATM networks, and WANs including VSAT satellite channels. In short, DBS has the capability of both measuring and analyzing TCP performance more in details. Through these measurements, DBS unveiled details of TCP performance which cannot be realized by other existing tools.

Keywords: TCP, internet, paformance evaluation, benchmark, transport layer, computer network

1. INTRODUCTION

TCP/IP is a fundamental communication protocol suite for the Internet. It contributes to make the rapid growth of the Internet in 1990s. The reason why the Internet has so succeeded is that the Internet protocol suite was well designed and composed with excellent philosophy.[1] However, as the Internet has been growing, several serious problems on its scalability such as running out of IP address space were pointed out. In order to resolve these problems, the Internet community has kicked off new activities such as IPv6 development and deployment.[2] Through these new activities, the Internet community is aiming to make improvements on the current Internet environment, and as its result, it is expected that the Internet can continue its expansion to cover the all over the world.

One of the activities is now focused on the TCP[3] (Transmission Control Protocol) which provides reliable transport layer service for many Internet applications including WWW, E-mail, news and other major ones. Since its development in 1980's, many researchers pointed out its drawbacks and fix them to improve its performance and capability. However, there still exists several problems in it. For example, the RFC1323[4] stated that an original TCP could not provide good performance even with broad bandwidth networks. The other major drawback is that TCP may transfer data segments intermittently. Thus, it can achieve enormously low throughput while the underlying data link remains enough bandwidth.[5,6] Various kinds of solution for these problems have been proposed and some of them were integrated into the major implementation of TCP such as its implementation in 4.4BSD UNIX. However, many researchers still believe that TCP still has many unknown problems inside. Therefore, both identifying and fixing remaining problems in the TCP are very important.

In order to achieve this goal, there are two approaches: simulation and benchmarking. The simulation approach is well known and several simulation tools for protocol evaluation are available. However, we consider that using protocol simulators is not adequate for our purpose, because the modeling for its simulation may eliminate important elements of TCP specifications and implementations. Especially, evaluations on actual TCP implementations cannot

Other author information: (Send correspondence to Y. M.)
Y. M.: Email: yukio-m@is.aist-nara.ac.jp; WWW: http://shika.aist-nara.ac.jp/member/yukio-m; Telephone: +81-0743-72-5784; Fax: +81-0743-72-5349
S. Y.: Email: suguru@is.aist-nara.ac.jp; WWW: http://shika.aist-nara.ac.jp/member/suguru; Telephone: +81-0743-72-5341; Fax: +81-0743-72-7349
This paper was published in the Proceedings of the conference "Performance and Control of Network Systems", part of SPIE's International Symposium Voice, Video, and Data Communications, SPIE Volume 3231, pp. 570-581, November 1997.

be done through simulations. The performance of the TCP is affected not only by its protocol specification and internal algorithms, but also by its implementation, memory management mechanism in its underlying operating system, system architecture and characteristics of communication channels we use. In fact, the current existing simulators such as "REAL"[7] and "ns"[8] cannot handle simulation on computer architecture or data links.

On the other hand, the benchmarking approach is anther popular way to evaluation protocol implementations. In this approach, monitoring processes are observing actual data transfers over networks. This approach has several advantages. One of them is that this approach can evaluate actual protocol implementations. The other is that this method can be applied very easily to evaluation of the end-to-end protocol performance. If we evaluate the end-to-end protocol performance through the simulation technique, we have to develop large scale simulation model which includes the target protocol and its underlying network elements. Therefore, the simulation may not be feasible in many cases. In the future, network will certainly become much higher than ever, but its mechanism will also become more complicated, comparing with what we have today. Hence, it is very important to establish fundamentals for evaluations of the end-to-end protocol performance. In this sense, developing more powerful benchmarking system has become very important today.

2. NECESSARY FUNCTIONS FOR TCP EVALUATION TOOLS

We consider that TCP evaluation tool should have the following functions:

1. The most important role of packet network is to share underlying communication channels and networking hardware. In the Internet, many TCP connections are sharing the network resources simultaneously. Also, TCP connections are set up to span many sites in the Internet. Therefore, the TCP evaluation tool should handle multiple TCP connections and simultaneous data transfers over the connections.

2. The evaluation tool should create both congested and low loaded conditions. Also, the tool should measure the changes of the throughput every moment. In the actual Internet, the load of the network is changing every moment. When network load is changing, TCP may change it's functioning mode, therefore, its performance is surely changed by changes of network load.

3. There are three control mechanisms related to TCP performance closely: flow control, retransmission control, and congestion avoidance control. These control mechanisms make strong impacts on TCP performance. If these three control mechanisms do not work well, we cannot utilize the network bandwidth effectively, thus we cannot obtain maximum throughput on the TCP connection. We need a tool which can evaluate these control mechanisms simultaneously.

4. The evaluation tool should handle several kinds of data links. The goal of improving TCP performance is to archive maximum throughput in the end-to-end application. Network monitoring using network analyzer or packet dumping using 'tcpdump' command cannot measure application level throughput. Network monitoring or packet dumping are not evaluation method but a good method for analyzing network behavior. TCP/IP has been used on a variety of data link such as Ethernet, Token Ring, FDDI, ATM, X.25, ISDN, Frame Relay, Fiber Channel and so on. It will continue to be used on new data link that is not available today.

Today we can find measurement tools such as ttcp, Netperf,[9] nettest, and NetSpec.[10] However, these tools do not fulfill the required functions listed above. Netperf, ttcp, and nettest can measure only the mean throughput on a single point-to-point connection. Furthermore, they evaluate only the flow control mechanism while it should evaluate all three controls. Although NetSpec can handle multiple TCP connections simultaneously, it can measure only the mean throughput. Thus, TCP/IP network traffic cannot be analyzed in detail by using NetSpec. Moreover, all these tools were designed with a wrong assumption that both the changes of throughput and the lost of TCP segments never occur in the network. Therefore, they measure the TCP performance under the special condition. For funding out problems around the TCP implementation, these tools cannot be used, because we cannot evaluate TCP protocol processing in details with these tools. For this purpose, we need more better tools.

3. PROPOSAL AND GOALS OF DBS

A goal of our development of this system called "DBS" (Distributed Benchmark System) is to provide a useful tool chest for evaluating entire TCP functions. Since there are three TCP control mechanisms that are essential for evaluating TCP performance, i.e. flow control, retransmission control and congestion avoidance control, we introduce the following properties as the fundamental of DBS functions:

- The DBS can treat multiple TCP connections spanning multiple hosts simultaneously, and

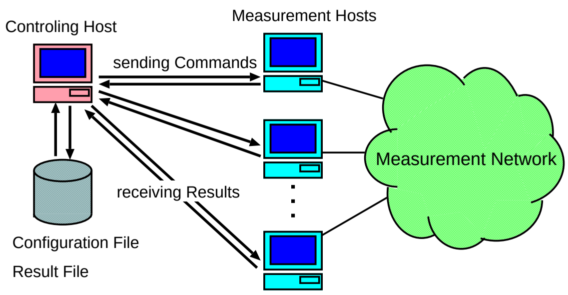

Figure 1. DBS System Structure Overview

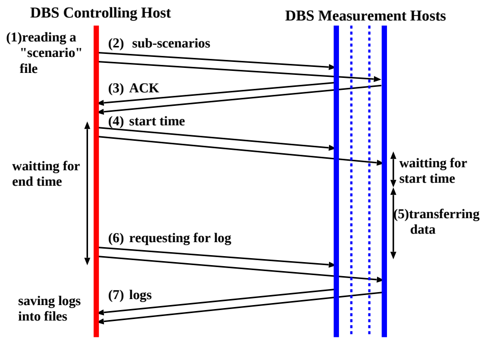

Figure 2. DBS Process Flow

- The DBS has the capability of measuring the changes of application level throughput at every data transmissions.

Having these properties, the DBS will be able to evaluate the three mechanisms of TCP mentioned above. In a congested network, data transmissions may cause TCP segment lost. In this case, retransmissions should be done for recovering the lost segments. As its result, both the retransmission control and the congestion avoidance control are invoked. On the other hand, segment transmission in a less crowded network may invoke the TCP flow control mechanism.

The DBS can generate various situations where the three controls are working together. Moreover, the DBS can cope with the fluctuation of the network traffic in the case that the network load changes from high to low or from low to high respectively.

With these functions, the DBS can evaluate TCP implemented in each OS and evaluate various kinds of network components which integrated into a single network system.

4. IMPLEMENTATION

4.1. System Structure

The DBS is composed with 3 programs: dbsc, dbsd, and dbs_view. Figure 1 shows an overview of the DBS System Structure.

The dbsc is a control program to handle monitoring program launches and terminations on observed hosts. dbsd is a program for sending and receiving data among observed hosts. These two programs (dbsc and dbsd) are used for actual benchmarking. On the other hand, dbs_view is used for data analysis. The details of these programs are described below:

- dbsc : DBS controller (Controlling Host)

  The DBS controller is a program controlling the experiment of TCP/UDP data transfer. Functions of the DBS controller are that (1) the DBS controller reads command from a command file, then (2) it asks the DBS daemons to start data transfer experiments, and (3) after receiving results from the daemons, the DBS controller will save them into the local files.

- dbsd : DBS daemon (Measurement Host)

  The DBS daemon is a daemon program that is launch on the experimental hosts. It sends and receives network traffic according to the commands instructed by the DBS controller.

- dbs_view : DBS viewer

  The DBS viewer is a program for analysis data which is gathered by the DBS controller. It draws graphs to reveal the transitions of sequence numbers, changes of throughput, changes of delay times or other performance indexes.

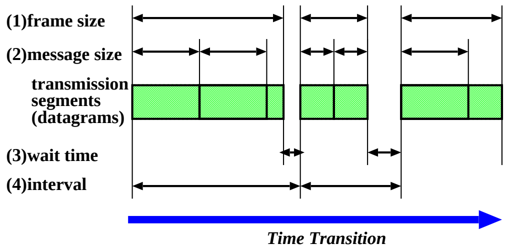

Figure 3. DBS Traffic Pattern

4.2. Process Flow

The DBS process flow is shown in Figure 2. The DBS performs benchmarking in distributed manner as follows:

(1) When the DBS Controller starts up, it reads a "scenario" file written in the DBS command language. The file specifies how the benchmark is carried out. After reading the file, the DBS Controller splits it in to "sub-scenarios" for each host where the DBS daemon is running.

(2) The DBS Controller sends the sub-scenarios to each DBS Daemons.

(3) When the DBS Daemon receives a sub-scenario, it sends an ACK back to the DBS controller in order to notify its acceptance.

(4) If all the DBS Daemons who participate the benchmark send ACK back to the DBS controller, then the DBS Controller announces start time of the benchmark to all the DBS daemons.

(5) After the start time, the DBS Daemons send/receive data to/from other DBS Daemons as the sub-scenario specifies. Simultaneously, each DBS Daemon logs time and other important data when both sending and receiving TCP segments are occurred. What is recorded in this log file is also specified in the scenario which the DBS Controller reads.

(6) Just after the benchmark is over, the DBS Controller asks the DBS Daemons to send back the log files.

(7) The DBS Daemons send the log files back to the DBS Controller. The DBS Controller saves them into a "result" file.

Our DBS implementation assumes that clocks on all the hosts participating to the benchmark are synchronized. This assumption can be fulfilled by using of the network time synchronization system such as NTP.[11] Since popular implementations of TCP use 500ms and 200ms timer inside,[12] the time drift and error caused by NTP (around 10ms) is small enough to evaluate TCP performance. In the case more precise benchmark is required, more accurate clock system such as GPS based global clock synchronization can be used.

During a benchmark, the DBS tries to make influences as few as possible. For example, during the benchmark, both the DBS Controller and Daemons are implemented very carefully in order to use less CPU time and to make less disk I/O. Furthermore, the DBS Daemon receives the sub-scenario from the specific hosts where the DBS Controller is running. This mechanism is implemented to avoid a security problem. If the DBS Daemon accepts scenarios from any hosts, villains called "cracker" can use the DBS Daemon as a way to generate network congestion. This simple mechanism prevents malicious uses of the DBS daemon.

4.3. Traffic Pattern

The DBS can generate UDP traffic for more realistic benchmarking. Since the UDP traffic is purely managed by applications directly, the DBS uses a model of UDP traffic and generates quasi-traffic pattern which can be applied to several UDP applications such as MPEG data streams. This UDP traffic model is depicted in the Figure 3.

The traffic is modeled as a sequence of data chunks called "frames." The size of each frame may vary. Each frame may consist of several messages. In this model, a single message is defined that it can be transferred in a single UDP datagram. Since the UDP has its maximum transfer unit (64K octets), the size of the message is limited by

Table 1. comparison DBS with other tools in functions

	DBS	Netperf	ttcp	nettest	NetSpec
Can it measure the throughput in TCP and UDP?	Yes	Yes	Yes	Yes	Yes
Can it establish multiple connections between two hosts?	Yes	No	No	Yes	Yes
Can it establish multiple connections between multiple hosts?	Yes	No	No	No	Yes
Can it specify the traffic pattern?	Yes	Yes	No	No	Yes
Can it measure the time variation of throughput?	Yes	No	No	No	No
Can it measure the overhead of connection establishment?	Yes	Yes	No	No	No
Can it specify the starting time of transfer for each connection?	Yes	No	No	No	No
Can it record the value of TCP control block?	Yes	No	No	No	No
Can it measure the behavior of slow start?	Yes	No	No	No	No
Can it measure RTT between sending segment and ACK?	Yes	No	No	No	No
Is it take some type of firewall and security into consideration?	Yes	No	No	No	No

Table 2. Environment and Status

CPU	Twinhead Subnote (i486/33MHz)
OS	BSD/OS V2.1
Ethernet Card	PCMCIA EtherLink III(3C589C)
TCP Buffer Size	8196byte
Message Size	2048byte
Send Times (DBS,ttcp)	2048 times
Send Time (Netperf)	10s

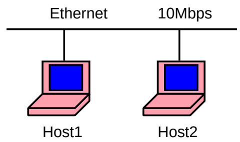

Figure 4. Evaluation Environment

Table 3. Results (Mean Throughput)

	Mean Throughput	DBS = 100
DBS	6223Kbps	100.0
Netperf	6357Kbps	102.1
ttcp	6172Kbps	99.1

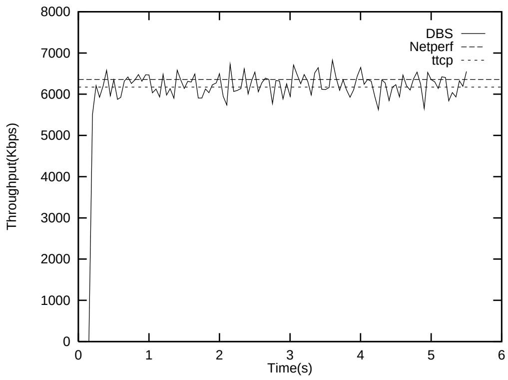

Figure 5. Throughput

this UDP's limitation. Therefore, if a frame is longer than the UDP maximum transfer unit, the frame is split into several messages.

Between each frames, there is a time gap called "wait time" which implies the application overhead. The preparation time of each frame can be treated as this wait time. Moreover, this model controls the frame intervals. For some multimedia applications, an application rate control mechanism is employed. This frame interval can be used for modeling of this kind of application level rate control.

4.4. TCP Trace Function

In the BSD UNIX, an useful trace function called "tcp_trace" is implemented. If the function is enabled, the UNIX kernel records each changes of values in the TCP control block (TCB).[12] The DBS running on the BSD UNIX systems can use this trace function in order to make more precise analysis. Analysis using this TCB values is a great help to know how TCP is working. Through this function, several TCP internal information such as TCP/IP pseudo-headers, TCP congestion window size, round trip time, retransmission timeout and other values in the TCB structure can be obtained.

5. EVALUATION

As one of its evaluation, we compared the DBS and existing tools from the view of their functions, as shown in Table 1. For measuring TCP performance, several functions are required to TCP benchmark systems. In the table, we summarize these required functions and check which tools fulfill them. It is obvious that the DBS has several

Table 4. Environment and Status

CPU	SGI INDY (R4600 100MHz)
OS	IRIX R5.3
TCP Buffer Seize	65535byte
Message Size	8192byte
Send Start Time	HOST1→HOST6: 0.00s
	HOST2→HOST7: 0.50s
	HOST3→HOST8: 1.00s
	HOST4→HOST9: 1.00s
	HOST5→HOST10: 1.50s
Send Times	1000 times

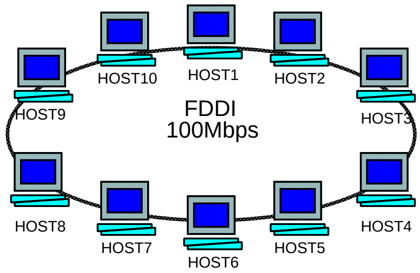

Figure 6. Environment

advantages because it can provide several performance indexes which can only be measured so far through simulation approach.

As its verification and evaluation, we conducted several benchmarks using both our DBS and existing tools in order to compare what we got as their results. During a benchmark, the DBS is logging the time when both sending and receiving TCP segments are occurred as mentioned in Section 4. In order to avoid overheads for this logging as much as possible, we implemented the very carefully. Hence, it is very important to know how much the log recording in the DBS makes influences on the benchmark results.

For this verification, we applied our DBS, Netperf, and ttcp to the benchmark environment shown in Figure 4 where the Host1 send data to the Host2 as much as possible. The configuration of these hosts are summarized in Table 2. With the DBS and the ttcp, the Host1 tries to send data 2048 times, while the Host1 send data continuously in 10 seconds with the Netperf. As shown in Table 3, there is quite a small deference on the mean throughput among these three tools. From this result, we can conclude that influences on benchmark results by the time logging in the DBS is fairly small or just no effect.

Changes of the throughput between two hosts in this benchmark are depicted as a graph shown in Figure 5. This graph is a plot of the mean throughput at the Host2 (TCP segment receiver) in every 50ms. This can be calculated easily from the log files generated by the DBS. As a reference, the mean throughput obtained by both the Netperf and the ttcp are also plotted in the same graph. As shown in the figure, the throughput is 0Kbps from its start time (0ms) to about 200ms. This 0Kbps period is caused by both the sender's Slow-Start algorithm and the receiver's Delayed ACK technique. At the beginning of the benchmark, the Host1 immediately sends only a single TCP segment to the Host2 because of its Slow-Start mechanism. After a very short period (about 1ms), the Host2 completely receives the segment. Because of the TCP's Delayed ACK mechanism, the Host2 does not reply an ACK back to the Host1 instantly, and it waits for the next TCP segment. However, by the sender's Slow-Start mechanism, the Host1 is also waiting an ACK from the Host2. Thus, these two hosts fall into the deadlock situation. However, this deadlock is resolved by the Delayed ACK timer mechanism. When the first segment was received by the Host2, it enabled the Delayed ACK timer. When the Delayed ACK timer (UNIX fast timer) is expired, the Host2 replied the ACK back to the Host1. This is the mechanism where the 0Kbps period was generated. It is obvious that only the DBS can help this kind of analysis, because only the DBS has a function to evaluate changes of the throughput.

6. CASE STUDIES

A series of benchmarks was conducted to measure the performance of several types of networks such as an FDDI network, an ATM network, a HIPPI connection over an ATM network, a VSAT satellite network, and a part of the WIDE Internet which is an academic nation wide Internet in Japan.

6.1. FDDI Network

The benchmark environment of the FDDI network is shown in Figure 6. The configuration of hosts in the network and the benchmark scenario are shown in Table 4.

In the ideal case, if the number of TCP streams is N, the throughput of each streams is nearly 1/N of the data link level throughput. Furthermore, when the number is changed from N to M, the throughput should be changed from 1/N to 1/M smoothly. By the results of this benchmark shown in Figure 7 and 8, we can say that the behavior of TCP over the simple FDDI network is almost an ideal case. The results suggest that the FDDI network works well with the TCP mechanism.

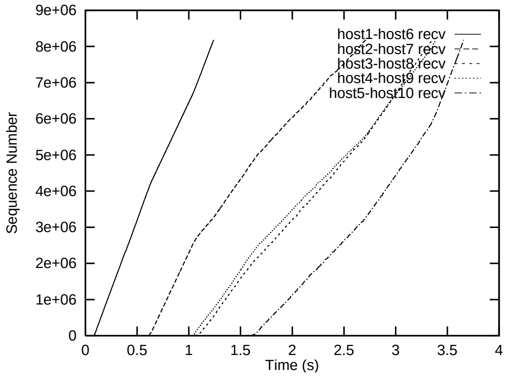

Figure 7. Sequence Number

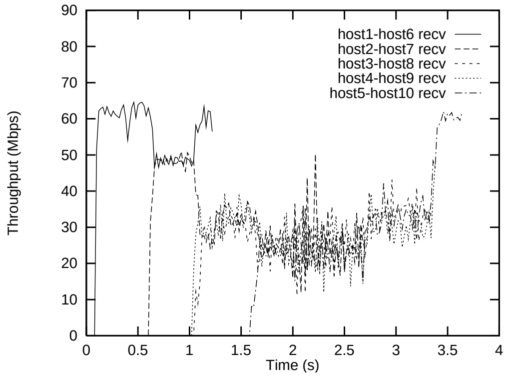

Figure 8. Throughput

Table 5. Environment and Status

CPU	Sun SPARCstation10
OS	SunOS 4.1.3
ATM Switch	NEC ATOMIS5
ATM Card	Fore SBA-200
Router	NEC IP 45/650 (Cisco7000)
AAL Type	AAL5
MTU	9188octet
TCP Buffer Size	52428byte
Message Size	8196byte
Send Start Time	HOST1→HOST3: 0.00s
	HOST2→HOST3: 2.00s
Send Times	HOST1→HOST3: 2000
	HOST2→HOST3: 500

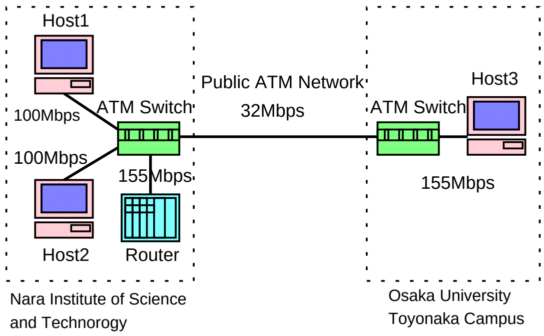

Figure 9. Environment

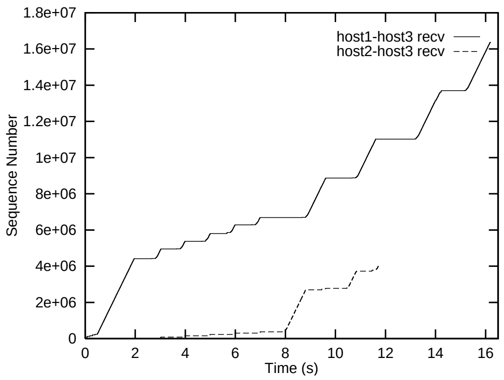

Figure 10. Sequence Number

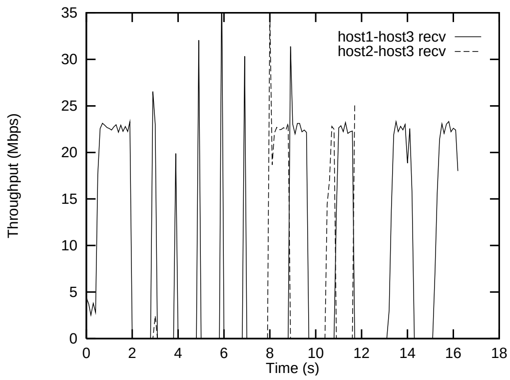

Figure 11. Throughput

6.2. ATM Network

The target environment of the benchmark is shown in Figure 9. In this network, there are two ATM switches; one is located in our campus and the other is in Osaka University. These two ATM switches are connected together by the wide area public ATM service provided by a local telephone company. The assigned bandwidth to this connection is 32Mbps. ATM OC-3 (155Mbps) or TAXI (100Mbps) interfaces connect hosts in this network to the ATM switches. The router is also attached to this ATM network via OC-3 interface. In this network, the classical IP over ATM method is used for IP datagram transfers. ATM VC's are configured between (1) the Host1 and the Router, (2) the Host2 and the Router, and (3) the Host3 and the Router. Assigned bandwidth to these VC's are (1) 100Mbps UBR, (2) 100Mbps UBR and (3) 32Mbps CBR (shaped), respectively. The benchmark scenario and the configurations of these host and the router are shown in Table 5.

Table 6. Environment and Status

CPU	SGI ONYX RE2, CHALLENGE XL
OS	IRIX 5.3
ATM Switch	NEC ATOMIS7
HIPPI-ATM TA	NEC ATOMIS2HA
TCP Buffer Size	256000byte
Message Size	8192byte
Start Time	ONYX(20)→CHALLENGE(36): 0.0s
	ONYX(12)→CHALLENGE(36): 1.0s
	ONYX(20)→CHALLENGE(36): 2.0s
	ONYX(12)→CHALLENGE(36): 3.0s

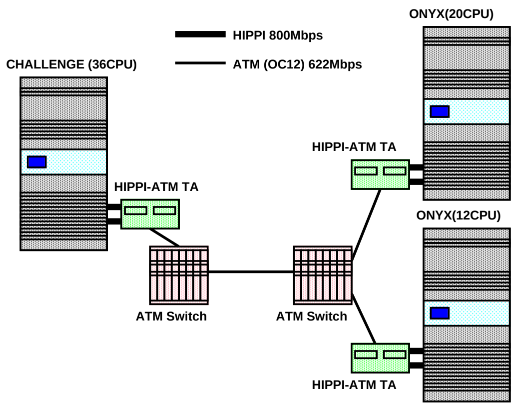

Figure 12. Environment

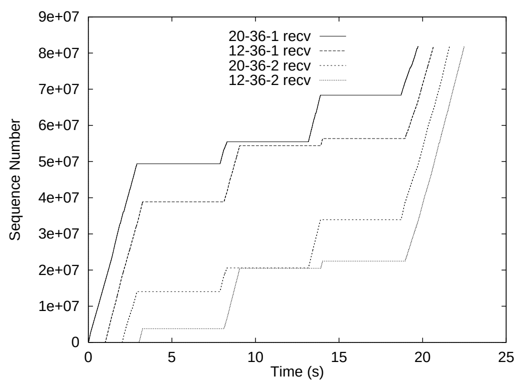

Figure 13. Sequence Number

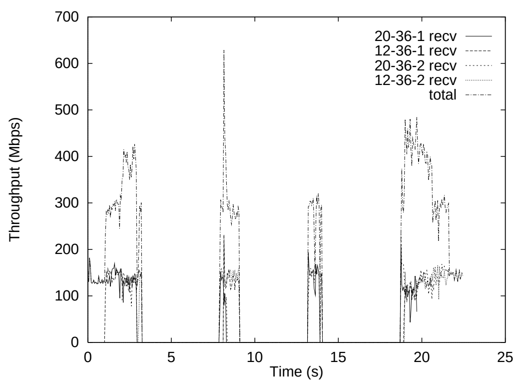

Figure 14. Throughput

When there is a single TCP stream between the Host 1 and the Host3 via the Router in the period between 0 and 2 sec., its throughput is approximately 23Mbps. By the other benchmarks not described here, we found that the bottleneck is the Router.

In the case we added one more TCP stream between the Host2 and the Host3 at 2 sec., we expected that these two TCP streams split the available bandwidth equally. However, the throughput of these TCP streams is awfully bad, because TCP data segments are transferred intermittently as shown in Figure 11. Through our detailed analysis of the benchmark results, we found that several TCP segments were lost during this period. There are two possible reasons why the TCP segments were lost: (1) the Cell buffer at the ATM switch was overflowed, or (2) the input buffer at the Router is overrun. We consider that both might be occurred by the ATM VC configuration where Host1 and Host2 were connected to the Router via 100Mbps UBR VC. The total bandwidth of VC at the Router interface is exceeded over 155Mbps. This kind of consideration can be lead only by the DBS.

6.3. HIPPI Over ATM Network

The HIPPI (High-Performance Parallel Interface) was developed as a high-speed channel (800Mbps) mainly for supercomputers. Since the HIPPI can be applied to a point-to-point connection within 25m (standard HIPPI cable), it has been applied mainly to supercomputer interconnection in machine rooms. In the recent year, the HIPPI-ATM TA has been available. The TA can be used as a way to make point-to- multi-point HIPPI connections.

In 1996, we got a chance on working with NEC Corp. to develop their new HIPPI-ATM TA. When we got the first version of the TA, our group evaluated its performance using our DBS. The benchmark environment is shown in Figure 12. In this network, 3 SGI's boxes are connected each together using the HIPPI-ATM TA. The underlying ATM network is based on OC-12 (622Mbps). The host configuration and the benchmark scenario are shown in Table 6. The result is depicted in Figure 13 and 14. As shown in these figures, the performance we obtained was just like

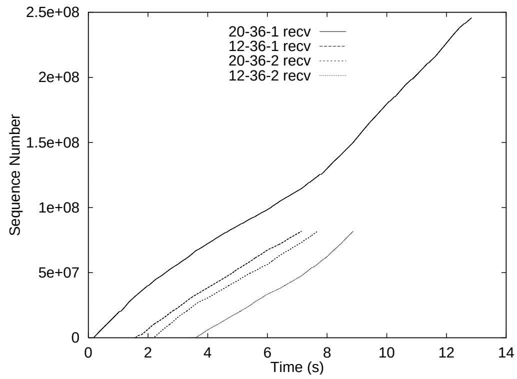

Figure 15. Sequence Number

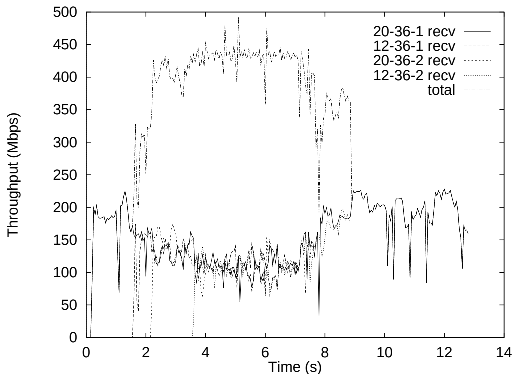

Figure 16. Throughput

Table 7. Environment and Status

CPU (HOST1,2,3,4)	Twinhead Subnote (i486/33MHz)
CPU (HOST5)	GATEWAY2000 (i486/66MHz)
OS	BSD/OS V2.1
Router	SPARCstation10 (SunOS4.1.4)
TCP Buffer Size	65535byte
Segment Size	512octet
Message Size	2048byte
Send Start Time	HOST1→HOST5: 0.00s HOST2→HOST5: 2.00s HOST3→HOST5: 4.00s HOST4→HOST5: 6.00s
Send Times	300
Notes	TCP trace is enabled.

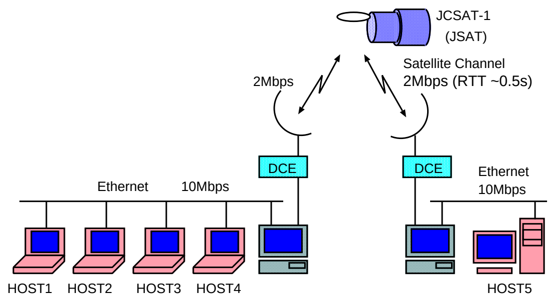

Figure 17. Environment

a nightmare. The network is idle in 60% of the benchmark period. We reported this result to NEC and ask their engineers to improve the TA. The NEC team rechecked its design and implementation, and finally they decided to redesign their controller chip and released its new version again. After updating our TA to the new version, the same benchmark was carried out again. The result of the second benchmark is shown in Figure 15 and 16. The performance has been much improved. We can get almost 450Mbps as an average throughput.

This is one of good examples where our DBS helps design and performance verification process of the network device developments.

6.4. VSAT Satellite Network

In 1990's, several groups have been trying to use VSAT satellite channels as an underlying communication channel of the Internet. However, because of its large RTT, many engineers worried about its end-to-end level performance. Our group evaluated it using the DBS. The target environment is depicted in Figure 17. As a joint research program between our group and the JSAT Inc. who is a communication satellite operator in Japan, we can use 2Mbps VSAT point-to-point channel provided by the JCSAT-1 communication satellite. We used 2 VSAT ground stations located in our campus and configure the target environment with several hosts and gateways listed in Table 7. The benchmark scenario is that at we add one more TCP stream over the satellite channel at every 2 seconds. The purpose of this benchmark is to show how their overall throughputs were getting worse in this scenario and to know if the satellite channel can work well with the current implementation of TCP.

Since the RTT (Round Trip Time) on this environment is about 500ms, an ACK message for each TCP segment is received every 500ms. Therefore, the congestion window size at the sender can be increased every 500ms. Since the TCP window size is limited up to 64Kbytes in this scenario, 5 or 6 seconds is required to provide 1Mbps as its maximum throughput over this network in the ideal situation. Because of its huge RTT and the TCP Slow-Start mechanism, the size of the TCP congestion window is increased very slowly.

The data transfer between HOST4 and HOST5 is finished within 15 seconds. But the other TCP streams need over 40 seconds to finish their data transfer. The throughputs of these three streams are fairly low comparing with

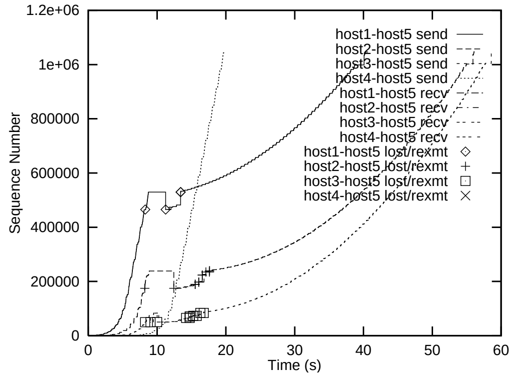

Figure 18. Sequence Number (Kernel Level)

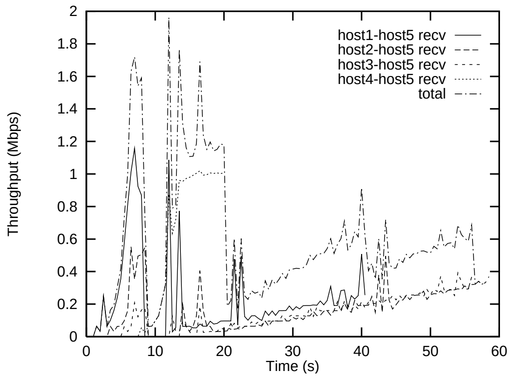

Figure 19. Throughput

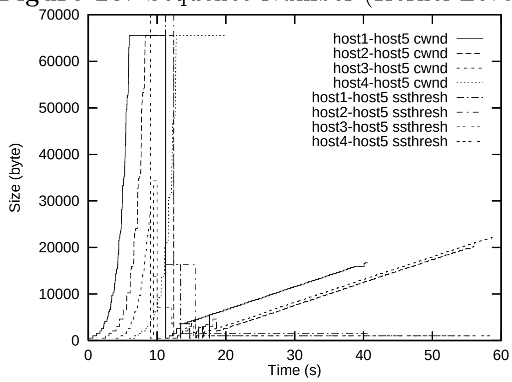

Figure 20. Congestion Window Size (cwnd) and Slow-Start Threshold (ssthresh)

Table 8. Environment and Status

CPU	SPARCstation (SUN)
OS	SunOS 4.1.3
Data Transfer	exp1:HOST1(Nara)→HOST2(Fujisawa) exp2:HOST2(Fujisawa)→HOST1(Nara)
TCP Buffer Size	exp1:8196byte exp2:16384byte
Message Size	exp1:1024byte exp2:1024byte
Date	exp1:Feb 9 16:10 (JST) 1996 exp2:Feb 3 19:40 (JST) 1996
Notes	TCP trace is enabled.

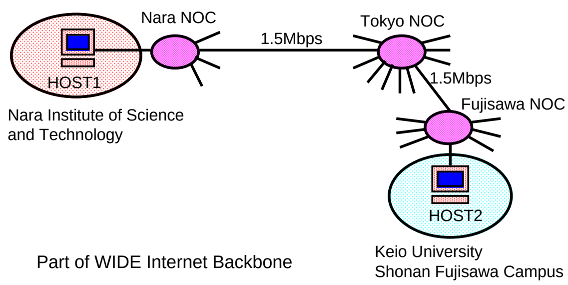

Figure 21. System Structure

the stream between HOST4 and HOST5. This is unfair. The reason why this unfairness was happened is the TCP slow-start threshold for these three low-throughput streams became too small.

The mechanism how the threshold became too small is slightly complicated. Around 8 seconds, several TCP segments for these three low-throughput streams were lost. By these segment losses, TCP at the Host5 transferred number of duplicated ACK messages back to their sender process. In the current implementation of TCP, the slow-start threshold is decreased to half every when three duplicated ACK messages are received. Therefore, by this mechanism, the slow-start threshold for these three streams was dropped down suddenly. Hence, throughputs for these three streams had been degraded rapidly. This problem is know as a side effect of the TCP fast retransmit implemented in 4.3BSD UNIX Reno version.[13]

6.5. WAN

Even though TCP was designed for wide area networks, it still has several drawbacks that are exposed in the

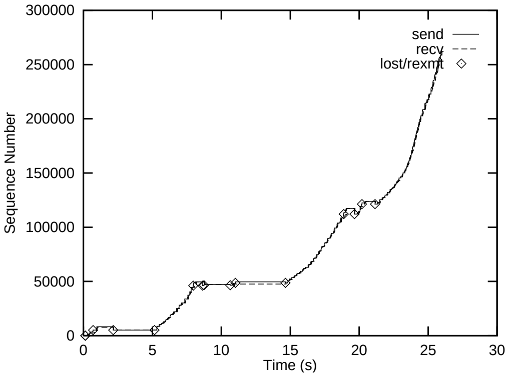

Figure 22. Exp.1 Sequence Number and lost segment (Kernel Level)

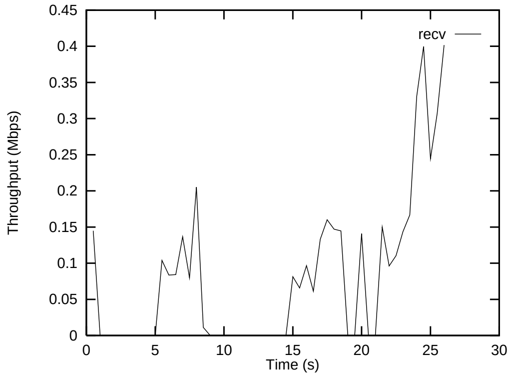

Figure 23. Exp.1 Throughput

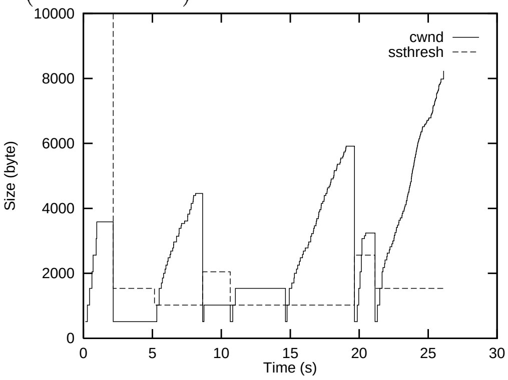

Figure 24. Exp.1 Congestion Window Size (cwnd) and Slow-Start Threshold (ssthresh)

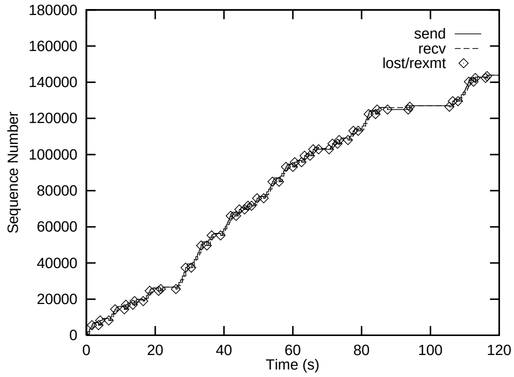

Figure 25. Exp.2 Sequence Number and lost segment (Kernel Level)

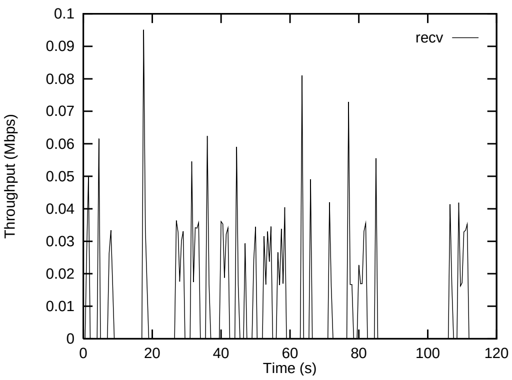

Figure 26. Exp.2 Throughput

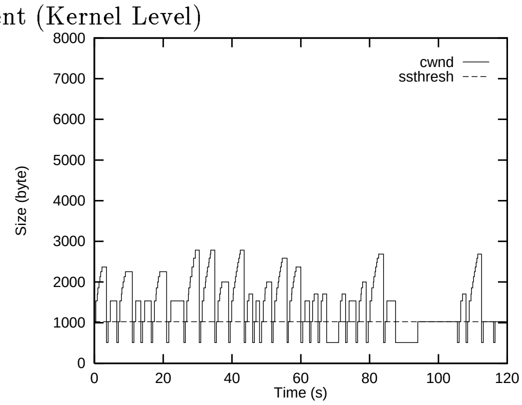

Figure 27. Exp.2 Congestion Window Size (cwnd) and Slow-Start Threshold (ssthresh)

large scale Internet. One of them is that TCP generates short burst traffic in the Internet, thus it can't provide good performance in such an environment. In order to confirm this behavior, we conducted a benchmark. The target environment is depicted in Figure 21. This is a part of the WIDE Internet backbone in Japan. The WIDE Internet is a large-scale testbed for network researches, however, it also carry ordinary Internet traffic for organizations connected to the WIDE Internet. We installed the DBS on a host in our campus and the other in Keio University located near Tokyo area. Using the DBS, we tried to evaluate how the TCP acts in this kind of heavy-loaded wide area network. The configurations of the hosts and benchmark scenario are shown in Table 8.

We did this benchmark twice in 1996. The results in the first benchmark where the rate of lost segments is about

1% are shown in Figure 24, 22 and 23. The results in the second benchmark in which the loss rate is around 30% are also shown in Figure 25, 26 and 27. As shown in Figure 23 and 26, TCP generated a sequence of short bursts periodically. Since the TCP congestion avoidance mechanism does not control the data transfer rate but the number of TCP segment to be sent, these kind of short bursts are generated.

7. FUTURE WORK

In recent years, several new techniques and implementations to improve the TCP performance have been proposed such as The SACK (Selective Acknowledgment),[14] the VEGAS,[15] and the FACK with SACK.[16] We are planning evaluations of them using the DBS.

Many protocols in the Internet protocol suite have been improved by their deployments to the actual Internet environment. In this improvement process, there exists concrete philosophy that not only protocol specifications but also their implementations or "working code" are very important and vital for protocol developments for the Internet. In this sense, we believe that the DBS, which is benchmark software for actual implementations of TCP, can help for improving TCP and development of "TCPng" (TCP Next Generation) for the future Internet.

8. SUMMARY

Since the TCP is a vital transport protocol for Internet applications, the TCP performance evaluation is required in order to improve it for the actual Internet environment which characteristics are changing rapidly. We proposed a new TCP performance evaluation system called DBS based on the benchmarking approach. In this paper, we show the design and implementation of the DBS. Furthermore, we report several case studies in which the DBS is applied to several kinds of network system. The DBS provides rich functions and it can unveil TCP properties which could not be analyzed by the other existing tools. We are going to use the DBS in order to evaluate newer TCP implementations, and will use it as a tool for supporting the TCPng design in the near future.

APPENDIX A. HOW TO GET DBS?

The DBS is available at the following URL: `<http://shika.aist-nara.ac.jp/member/yukio-m/dbs/index.html>`

REFERENCES

1. David D. Clark, "The Design Philosophy of the DARPA Internet Protocols," Proc. SIGCOMM '88, Computer Communication Review Vol. 18, No. 4, August 1988, pp. 106-114.
2. S. Deering, R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," RFC 1883, Jan 1996.
3. J. Postel, "Transmission Control Protocol", RFC 793, September 1981.
4. V. Jacobson and R. Braden and D. Borman, "TCP Extensions for High Performance," RFC 1323, May 1992.
5. Douglas E. Comer and John C. Lin, "TCP Buffering And Performance Over An ATM Network," Purdue Technical Report CSD-TR 94-26, 1994.
6. Jon Crowcroft, Ian Wakeman, Zheng Wang, and Dejan Sirovica, "Is Layering Harmful?," IEEE Network Magazine, vol.6, pp 20-24, January 1992.
7. Srinivasan Keshav, "REAL: A Network Simulator," Computer Science Department Technical Report, University of California, December 1988
8. S. McCanne and S. Floyd, "ns-LBNL Network Simulator," Obtain via: http://www-nrg.ee.lbl.gov/ns/.
9. Hewlett-Packard Company, "Netperf: A Network Performance Benchmark Revision 2.1," 1995.
10. Roel Jonkman, "Net Spec: A Network Performance Evaluation and Experimentation Tool," http://www.tisl.ukans.edu/Projects/AAI/products/netspec/, 1995.
11. D. Mills, "Network Time Protocol version 2 specification and implementation," RFC 1119, Sep 1989.
12. Gary R. Wright, W. Richard Stevens, "TCP/IP Illustrated, Volume2 The Implementation," Addison-Wesley Publishing Company, 1995.
13. S. Floyd, "TCP and successive fast retransmits," Lawrence Berkeley Laboratory, Technical Report, 1995. Available from anonymous ftp from ftp.ee.lbl.gov:papers/fastretrans.ps.
14. M. Mathis, J. Mahdavi, S. Floyd, A. Romanow, "TCP Selective Acknowledgment Options," RFC 2018, October 1996.
15. Lawrence S. Brakmo, Sean W. O'Malley, Larry L. Peterson, "TCP Vegas: New Techniques for Congestion Detection and Avoidance," National Science Foundation Grant IRI-9015407 and ARPA Contract DABT63-91-C-0030, February 16, 1994.
16. Matthew Mathis and Jamshid Mahdai, "Forward Acknowledgment: Refining TCP Congestion Control," ACM SIGCOMM, volume 26, number 4, October 1996.