PAPER Special Issue on Next Generation Internet Technologies and Their Applications

Development of the Flexible and General IP Evaluation Tool STAGE

Yoshihiro ITO†, Yuichiro HEI†, Masami ISHIKURA†, *and* Tohru ASAMI†, *Members*

SUMMARY Recently, because of the expansion of TCP/IP networks, the evaluation of IP networks is being considered as a major task for network operators and administrators. As a result, many IP evaluation tools have appeared, but almost all of them can only be applied to a limited purpose. The authors have already developed an IP evaluation tool, called KITS (KDD Internet Test System) [1] that can generate flexible load traffic. Based on KITS, the authors have now developed a new IP measurement tool. This paper describes the flexible and general IP evaluation tool, STAGE (Simulated Traffic Analyzer/generator for General Evaluation).

key words: *network evaluation, IP, KITS*

1. Introduction

The expansion of the Internet has led many networks that use TCP/IP protocols in Intranet or Extranet environments. Consequently, the evaluation of IP networks has become important and many tools for IP evaluation have now been developed.

Before we discuss IP network measurement in this paper, we will classify IP network evaluation methods into two classes. The first is a method wherein measurement and analysis of traffic are performed without injecting a load to the network or a device. We call this class of methods, passive measurement. The major passive IP measurement tool is a LAN analyzer, which captures data-link frames over the medium and decodes them. In addition, other passive methods using SNMP [2] are RMON [3] or Meter MIB [4]. With the passive method, we cannot select an arbitrary load traffic as an input, so it is difficult to analyze the network characteristics. However, the passive method can evaluate networks without affecting the present network, so is suitable for monitoring the present network or examining the faults in the network.

The second is a measurement and analysis of the network which generates load traffic. We call this class of methods, active measurement. Software or hardware-based tools using the active measurement method already exist. Hardware-based active measurement tools can generate heavy and exact load traffic, but generally are very expensive and inflexible. Major active IP measurement software-based tools are Ttcp [5] and Netperf [6]. These tools send and receive pseudo-traffic using TCP/UDP to measure the end-to-end TCP/UDP performance. Ttcp can measure the maximum throughput, and Netperf can measure the maximum throughput and the minimum delay time. KITS, which the authors previously developed, can also measure TCP/UDP performance using arbitrary load traffic which a user can define. On the other hand, Ping [7] and Traceroute [8] measure the roundtrip time or the route path using ICMP [9] messages. Similarly using ICMP messages, Pathchar [10] estimates available bandwidth over the path and Treno [11] estimates TCP performance. The active IP measurement tool can select load traffic as input so that it performs a more exact measurement than the passive tool. However, because of the effect on the network, the operator must be careful when using this tool.

Generally, in order to evaluate IP networks more exactly, it is better to use active tools rather than passive tools, but almost all existing active IP measurement tools only evaluate specific applications or services.

This paper describes about the general and flexible IP evaluation tool, STAGE(Simulated Traffic Analyzer/Generator for General Evaluation), which the authors developed based on KITS. Section 2 describes the characteristics of STAGE, and Sect. 3 shows the implementation of STAGE. Section 4 shows typical measurement using STAGE. Section 5 shows a survey and a comparison with other tools. Finally, Sect. 6 gives the conclusion.

2. STAGE

2.1 Design of STAGE

Almost all the existing IP active measurement tools are used to measure the performance of specific applications or services, so that these tools can only generate very simple or specific traffic. KITS can generate all kinds of traffic which emulate any application or network by defining the generation functions of the packet length and interval. Although KITS is very efficient for evaluating many types of IP networks, we found some problems using KITS as follows. First, KITS can generate traffic only over RTP/TCP or RTP/UDP. Some measurements may need to generate traffic using rawIP packets or Datalink frames. And some routers may differentiate RTP packets from other packets. Second, KITS cannot be controlled from a remote terminal, therefore measurements with a complex procedure like RFC 2544 based measurements must be done manually. Third, the result shown by KITS are network QoS metrics like throughput, latency or packet loss rate. However users needs more familiar applica-

Manuscript received September 30, 1999.
Manuscript revised November 30, 1999.

† The authors are with KDD R&D Laboratories Inc., Kamifukuoka-shi, 356-8502 Japan.

tion QoS metrics like file transfer rate or quality of VoIP.

Therefore, the new tool that we developed based on KITS should have the following: flexibility, scalability, conformity to standard measurement, comprehension and familiarity. Flexibility means that the tool must be applicable to any type of application or network. Scalability means that the tool can extend easily. Conformity to standard measurement means that this tool must be able to measure standard metrics like RFC 2544 based metrics. Comprehension means translation to application QoS metrics from network QoS metrics. Familiarity means that this tool must support a user friendly graphical user interface (GUI).

2.2 Functions of STAGE

To satisfy the requirements as described above, STAGE has many functions as follows, 1) Flexible pseudo-traffic generation, 2) Arbitrary protocol for transmission, 3) Remote control of many sender or receiver nodes from one node, 4) Applicable IP multicast, 5) Auto measurement, 6) Mapping from network QoS metrics to application QoS metrics. The functions 1), 2), 3) and 4) support flexibility. Functions 1) and 5) can be defined by users so that scalability can be satisfied. Using the functions 3) and 5), STAGE provides scripts for standard measurement like RFC 2544, so the conformity to the standard measurement is satisfied. The function 6) supports Comprehension. To support familiarity, STAGE provides a web-based graphical user interface. As described before, STAGE was developed based on KITS. We discuss the differences between KITS and STAGE in later section.

2.3 Web-Based Measurement Architecture

In order to provide a new service like SLA (Service Level Agreement), networks should be constantly monitored. STAGE was developed on a UNIX-based OS and it can boot from an httpd daemon. So the receiver node is controllable from the other node via an http control connection at any time. Therefore, an administrator of networks does not need to use a specific terminal and can measure from anywhere in the network. Before describing this architecture, we would like to define some terms. We call a node which runs STAGE, a measurement node. We classify this measurement node into three categories, sender, receiver and configurator. A sender generates pseudo-traffic, a receiver receives and analyzes it. A configurator set up senders or receivers via http by an administrator. In many cases, a configurator is the same node as a sender or a receiver.

A general measurement using STAGE is as follows. An administrator of network uses a web browser to configure and control measurement process of a sender via http. In this case, a configurator is the same node as a sender or a receiver. Next, the measurement process of a sender configures and controls the measurement processes of receivers via http. Then, a sender starts to transmit measurement pseudo-traffic. After the measurement, the receiver sends its own result to a sender. Finally, a sender analyzes these collected

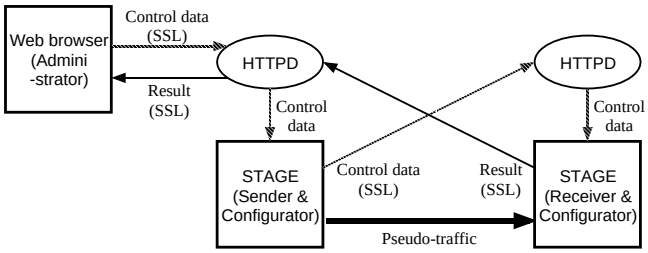


Fig. 1 Web-based architecture.

results and sends the analysis to an administrator. Figure 1 shows this architecture. Here, by using SSL (Secure Socket Layer), we can maintain data security among a specific group of nodes, that is, between a web browser and a configurator or between a configurator and a sender (receiver). An administrator can control many senders and receivers simultaneously from one node.

On the other hand, measurements generally need to be repeated to increase reliance of measured value, and a measurement based on RFC 2544 or BER estimation [1] needs specific methods for measurement. STAGE supports many web-based scripts to perform RFC 2544-based measurement or BER estimation, so that these metrics can be obtained with one measurement. These script can be modified by the users.

STAGE can work manually when a control connection cannot be established according to the asymmetrical path. For example, many Intranets connect to the Internet via firewalls, so that the traffic through these firewalls may be restricted to some application or source/destination pairs. Also, when the control traffic may affect the measurement, it is not desirable to use control connections.

2.4 Traffic Generation

In this subsection, we describe the traffic generation function of STAGE.

2.4.1 Traffic Pattern

Almost all tools for the IP evaluation, such as Ttcp or Netperf, can generate only very simple traffic or traffic-emulating specific applications. KITS, however, can generate pseudo-traffic that emulates traffic of any type of application or network by defining two generation functions for packet length and packet interval. Similarly, like KITS, STAGE can use user-defined generation functions. Also, generic functions, like the normal distribution function or the Poisson distribution function, and generation functions from the captured LAN/WAN traffic are supported as parts of a function library.

2.4.2 IP Header Field

KITS and STAGE normally generate pseudo-traffic for measurement as IP packets. Although KITS can generate any type of traffic, KITS cannot edit all IP header fields. Here, in order to evaluate devices that control traffic per flow, such as layer 3 switches or policy control routers, the source IP ad-

dresses of pseudo-traffic for measurement need to be variable. On the other hand, to evaluate routers that perform QoS/CoS control using MPLS [12] or IP precedence bits [13], the TOS field in the IP header must be variable.

STAGE can set any source address of a pseudo-traffic only if it works as a traffic generator, and can edit IP header fields of pseudo-traffic like TTL, TOS, protocol as well as source address fields.

2.5 Protocols

Figure 2 shows the STAGE protocol stack.

2.5.1 Encapsulation of Pseudo-Traffic Data

KITS uses the RTP [14] packet as pseudo-traffic in order to carry certain information (timestamp, sequence number, etc.) for calculating the many statistical end-to-end metrics. However, some routers which handle low-rate serial lines perform header compression [15]. This compression may cause the router to discriminate the pseudo-traffic from other traffic, in which case STAGE cannot generate pseudo-traffic using the RTP packet and another packet format specifically designed for STAGE is used. We call such a packet format, a STAGE packet (Fig. 3). In this packet format, PT indicates the payload type used to control data transmission. The Sequence number is used to measure the packet loss rate. The Time stamp (1) indicates the date (day, hour, minute) at which packets are created. The Timestamp (2) indicates the time (microseconds). Also in this packet format, the Cumulative sending data octets indicates the number of cumulative data already sent and length indicates the length of this packet.

Although, actual applications rarely use RTP over IP, or RTP over a Datalink layer protocol stack, the pseudo-traffic data encapsulated in RTP or the STAGE packet can be transmitted over any of the following protocols: TCP, UDP, ICMP, rawIP and Datalink. In this implementation, we use (Fast) Ethernet as Datalink. Although other media can be used, almost no terminals use ATM or SONET except for routers and switches. The main purpose of STAGE is measurement of end-to-end application performance and is therefore considered sufficient to support (Fast) Ethernet.

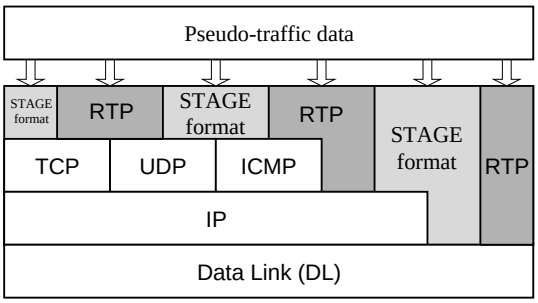


Fig. 2 Protocol stack.

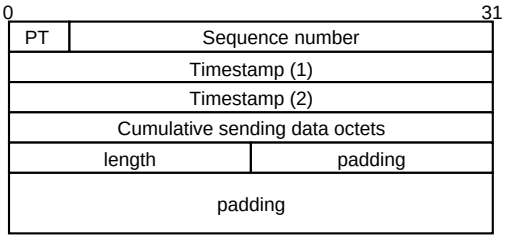


Fig. 3 STAGE packet format.

2.5.2 Pseudo-Traffic Transmission Protocol

KITS generates pseudo-traffic over TCP or UDP. Many existing applications and services definitely use TCP or UDP however some applications directly use IP in order to transmit data. In order to evaluate these applications or layer 2 switch devices, it is necessary to generate rawIP packets or Datalink frames as pseudo-traffic. STAGE cannot only generate pseudo-traffic for measurement using TCP or UDP, but also can generate pseudo-traffic if also having rawIP packets or Datalink frames.

2.5.3 Stand-Alone Measurement Using ICMP Messages

An active measurement tool injects a load onto the objective network for evaluation purposes so generally, a sender and receiver pair work simultaneously. However, when one end of the target network is a router which cannot work with any measurement tools, then measuring end-to-end characteristics between two nodes is difficult. Therefore, measurement tools such as Ping or Traceroute are available which estimate characteristics using ICMP messages. Similar techniques are used in Treno which estimates TCP performance or Pathchar which estimates the available bandwidth over the path. STAGE can also generate pseudo-traffic using ICMP messages so that it measures network QoS metrics with only one end node.

2.6 Multicast

Of those services which will appear over the Internet in the near future, IP multicast communication will certainly become more important. Although KITS can measure in a multicast environment, the sender and the receivers must be working simultaneously. Since in a general, multicast environment, receivers are able to join or leave a session, STAGE can make measurements from the joining through the leaving of a receiver as shown in Fig. 4. KITS, however, can measure only the one receiver A as shown in this figure.

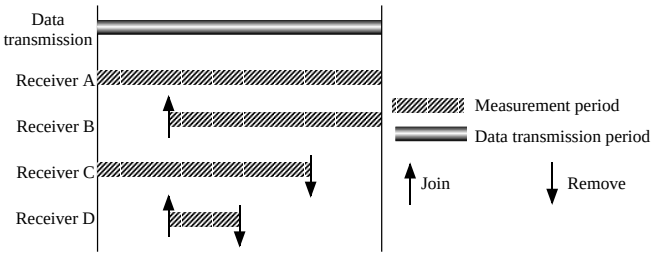


Fig. 4 Measurement in a multicast environment.

2.7 Mapping to Application QoS Metrics

In general, the metrics measured by IP measurement tools like KITS, are network QoS metrics such as Throughput or Latency, etc. These metrics differ from the application QoS metrics that the user actually recognizes. Much researches is being done on mapping from the network QoS metrics to the application QoS metrics are proceeding [18], thus making it preferable to translate from the network QoS metrics to the application QoS metrics. STAGE supports this mapping function by defining user translation functions or using some of the mapping functions obtained in [18].

3. Implementation

3.1 Platform

STAGE was developed on FreeBSD 2.2.8. In order to use STAGE, users can select the method with the command line parameters or configuration file. STAGE can be controlled via a Web based GUI using a web server on the same node.

3.2 Configurable Parameters

This subsection shows the configurable parameters.

3.2.1 Pseudo-Traffic Parameters

IP addresses
Both the source and the destination address of a packet can be set except for generation of Datalink frames. In order to simulate multiple senders, source addresses can be set for more than one address for IP and UDP modes.
Traffic pattern
The generation functions and their packet length and packet interval parameters can be set. A generation function can be selected within 1) fixed length (interval), 2) existing support functions or 3) user defined functions.
Transmission protocol
We can choose the protocol for transmission from IP, ICMP, TCP or UDP. We can also generate pseudo-traffic using Datalink frames. In this implementation, STAGE can generate (Fast)Ethernet frames.
Field values of IP packet
Besides the IP address, we can set the TOS, TTL and protocol field values.

3.2.2 Execution Parameters

Measurement topology
As the measurement topology, we can select from one-to-one, one-to-many, many-to-one, many-to-many or stand-alone using ICMP.
Multicast
Measurement with IP or UDP can use multicast.
Display of results
The display format of the results can be either the network QoS metrics or application QoS.

3.3 Functions

The traffic pattern of the pseudo-traffic and the mapping from the network QoS metrics to the application QoS metrics can be defined as a C language function by the user.

3.3.1 Generation Functions

Figure 5 shows an example of pseudo-traffic generation function.

The function shown in this figure is a packet length generation function with four arguments. The number of arguments of the generation function is actually variable. The arguments given to the function are handled as command options.

3.3.2 QoS Mapping Function

The mapping function which translates from network QoS metrics to application QoS metrics can be defined by users. The network QoS metrics that STAGE calculates are usable via the API. Applicable network QoS metrics are the mean and standard deviation of the packet length and delay time, packet loss rate, error rate, throughput and transmission rate.

For example, [16] shows the relationship between the quality of VoIP and the network metrics. Using this result, we can estimate the quality of VoIP using the network metrics.

3.4 User Interface

In general, it is difficult to measure the performance between many nodes such as in a multicast environment. If the purpose of the measurement is explicit, the measurement procedure can be automatic as shown in Fig. 6. First, we must design the measurement and define the number of nodes, a trans-

```
long    normal_dist(int usage, long *param[]) {
        long    val, mean, stddv, min, max;

        if(param[0] != NULL)    mean = *param[0];
        else if(usage == LENGTH_FUNC)    mean = LENGTH_MEAN;
        else    mean = DELAY_MEAN;

        if(param[1] != NULL)    stddv = *param[1];
        else if(usage == LENGTH_FUNC)    stddv = LENGTH_STDDV;
        else    stddv = DELAY_STDDV;

        if(param[2] != NULL)    min = *param[2];
        else if(usage == LENGTH_FUNC)    min = LENGTH_MIN;
        else    min = DELAY_MIN;

        if(param[3] != NULL)    max = *param[3];
        else if(usage == LENGTH_FUNC)    max = LENGTH_MAX;
        else    max = DELAY_MAX;

        val = nrnd() * stddv + mean;
        if(val > max)
                return(max);
        else if(val < min)    return(min);
        else    return(val);
}
```

Fig. 5 Example of a generation function of packet length.

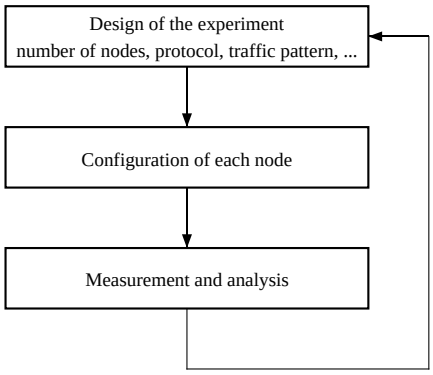


Fig. 6 Experiment procedure.

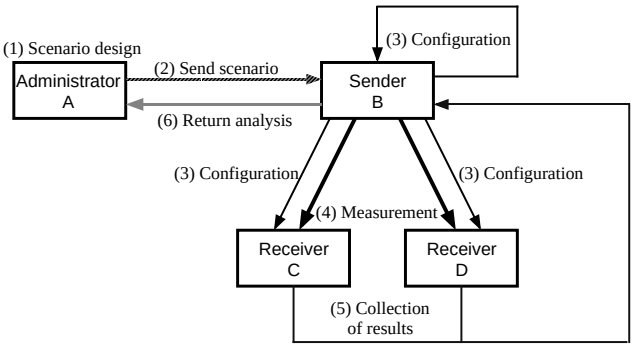


Fig. 7 Example of a measurement scenario.

mission protocol, traffic pattern, etc. Second, we configure the measurement tool according to this design. Finally, we start to measure, and collect and analyze the results. As the number of nodes in the experimental environment increases, the configuration of each node and the collection of results become difficult.

For example, consider the experimental scenario shown in Fig. 7.

The administrator of this network wants to measure the performance between sender B and receivers C and D with multicast using his terminal A. According to the measurement scenario, first, administrator A creates scenario (1), then sends the scenario to sender B (2). Second, sender B sends the configurations to receiver group (C and D) and to itself B, according to scenario (3). Third, B starts to transmit data to C and D (4). Then C and D send the result to B (5). Finally, B analyzes these collected data and sends the result to A (6).

As described before, STAGE provides auto measurement and remote control functions, so that procedures (2), (3), (4), (5) and (6) as described above can be performed. It is easy to provide a web-based GUI in order to support procedure (1). The reasons are as follows: STAGE is generally implemented on UNIX-based OS, so web service can be installed easily. Further almost all of the web browser supports security functions, so that the security of data between an administrator and a sender is maintained. For these reasons, STAGE can provides web-based GUI. Figure 8 shows a typical GUI for setting up a QoS mapping function.

Using auto measurement, remote control functions and GUI, we can automatically measure according to the experimental design with STAGE. To perform this automatic measurement, each node in Fig. 7 must be set as follows: 1) Administrator A provides web service. 2) Sender B, receiver C and D provide Inetd (Internet services daemon) or httpd (hypertext transfer protocol server daemon) service.

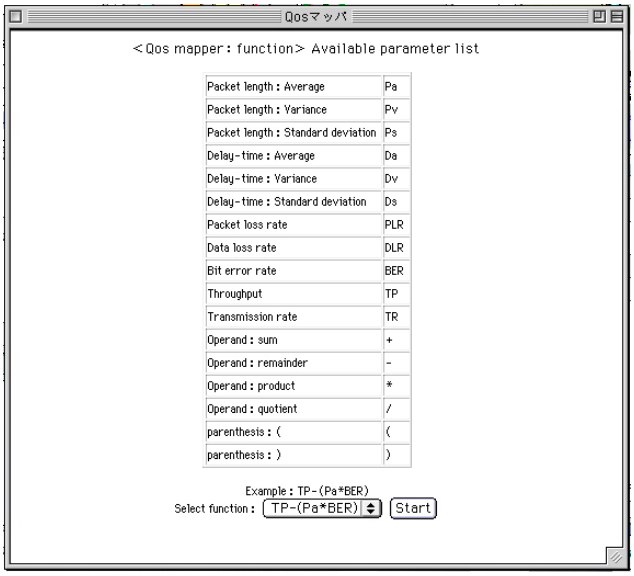


Fig. 8 Example of GUI (QoS mapping function).

STAGE uses the control TCP connection to configure other nodes. But, in some cases, it is undesirable to use this connection throughout the measurement. Using JAVA, STAGE can be controlled dynamically in greater detail depending on the current status.

3.5 Precision

The precision of STAGE depends on its operating system. To synchronize the time between a sender and a receiver, the current implementation uses NTP [19], so the accuracy of the measurement is ten milliseconds at most. To improve the precision of the system, it needs to use a real-time operating system. However, many current applications over the Internet do not need more accuracy. As another solution, we can use GPS to synchronize the time.

4. Measurement Example

In this section, we shows some examples of measurement using STAGE.

4.1 Multicast Performance

We measured one-to-two performance with multicast over a 10BASE-T based LAN using STAGE. In this experiment, we first made a data file of pseudo-traffic. Here we used pseudo-traffic whose packet length is the normal distribution with 250 bytes average, 200 bytes standard deviation, 0 bytes minimum, 500 bytes maximum length and a fixed packet interval (100 milliseconds). Figures 9–13 shows the GUI window to set up this experiment. First, we set up transmission

protocol (Fig. 9). Here we selected UDP as the transmission protocol. Second, we selected the topology (Fig. 10). In this experiment, we used one sender and two receivers. Third, we defined the receiver hosts (Fig. 11). Here we set the addresses of receivers and a multicast address. Fourth, we configured the parameters of a sender (Fig. 12). We set the filename of traffic pattern, port, packet format and log file name. Finally we configured the parameters of receivers similar to those of a sender (Fig. 13).

Figures 14 and 15 show the result. From this result, both nodes could receive all of the packets and both throughputs are nearly equal. The delay time however differs between the two nodes.

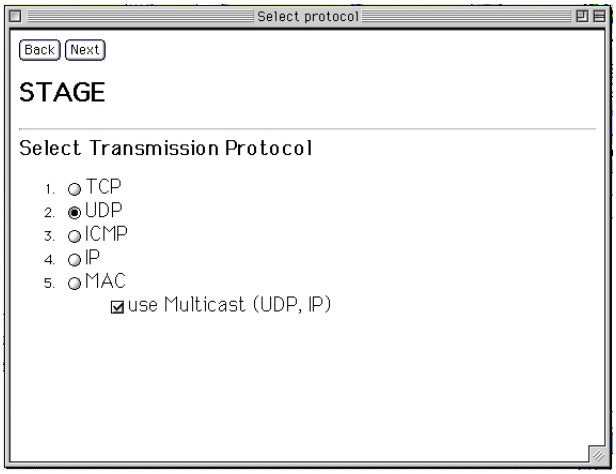


Fig. 9 Configuration (protocol).

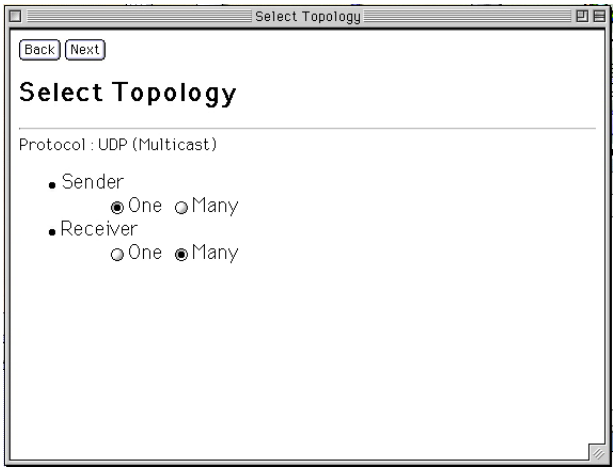


Fig. 10 Configuration (topology).

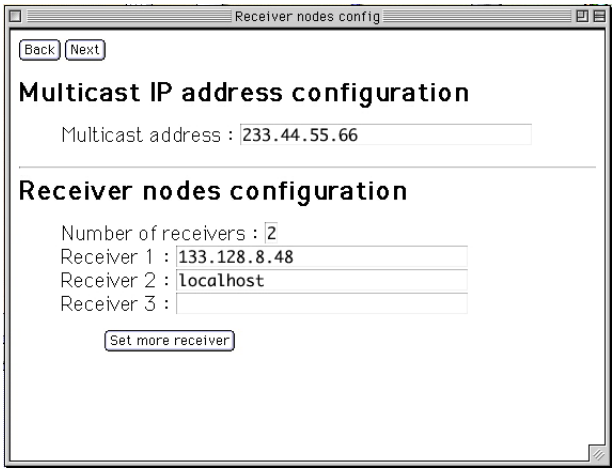


Fig. 11 Multicast configuration (receiver(s)).

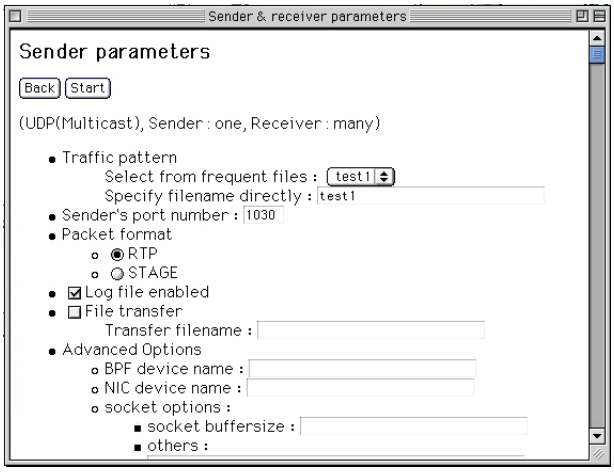


Fig. 12 Configuration (parameters of sender).

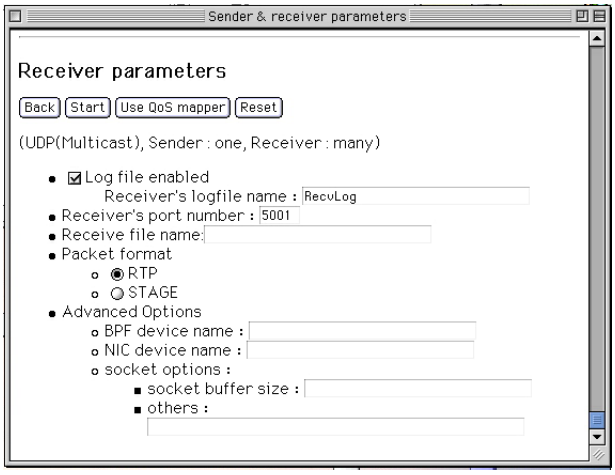


Fig. 13 Configuration (parameters of receivers).

5. Comparison with Other Tools

This section describes a comparison of STAGE with other active measurement tools.

We classify active measurement tools into two types; hardware-based and software-based. Software tools can moreover be classified into two categories according to their transport protocol. Some tools use ICMP to transfer pseudo-traffic. Other tools however use UDP/TCP to transfer the pseudo-traffic such as for general data transfer applications. These classifications are described below.

5.1 Hardware-Based and Software-Based Tools

In general, hardware-based tools can generate pseudo-traffic for measurement in a heavier volume and greater precision than is possible for software-based tools. Hardware-based tools however have a much greater cost than software-based tools. Hardware-based tools typically generate a heavy load pseudo-traffic of more than 10 gigabits per second and accuracy on the order of 10 nanoseconds. Software-based tools however generate a traffic at most of hundreds megabits per second and an accuracy on the order of ten microseconds. Hardware-based tools however, generally have a cost more than ten times that of software-based tools. Software-based tools are sufficient for measuring end-to-end performance in some applications because most applications on TCP/IP do not require a high bandwidth or high accuracy. However measureing devices, routers and switches requires a very high load and high accuracy so that hardware-based tools must be used.

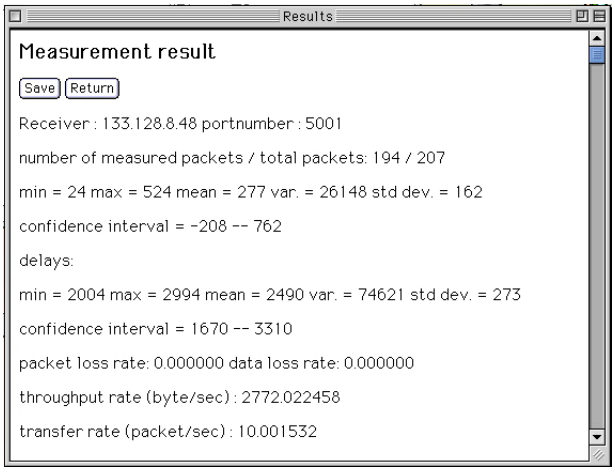


Fig. 14 Result (receiver 1).

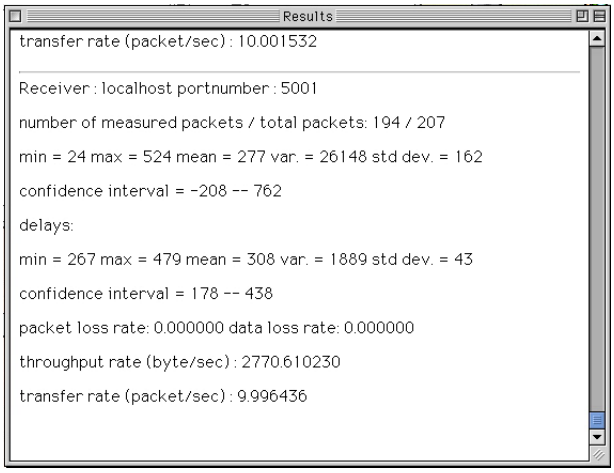


Fig. 15 Result (receiver 2).

5.2 Software-Based Tools

In this subsection, we discuss several software-based tools.

5.2.1 ICMP Tools

Major ICMP tools and their common characteristics are described next. All IP nodes support ICMP so in many cases, the receiver node does not need to execute such tools.

The most popular tool that uses ICMP is Ping. Ping sends ICMP echo request messages to the destination node, and the destination returns the ICMP echo reply message to the sender. Consequently, the sender can calculate the round trip time (RTT) and the number of reply messages losses. Ping is now implemented in almost all IP nodes.

Traceroute like Ping is often used to examine the network connectivity. By sending ICMP messages changing the TTL value, Traceroute can measure the RTT of all nodes over the path from source to destination.

Treno can estimate TCP throughput using the user-level to implement TCP-like protocol, Treno uses UDP to an unused port, ICMP error response or ICMP echo/reply so that Treno can measure TCP throughput independently of the TCP implementation of the end hosts.

Table 1 Comparison with other tools.

	Hardware-based Tools	Ping	Traceroute	Treno	Pathchar	Ttcp	Netperf	KITS	STAGE
Protocol	Datalink, IP	ICMP	ICMP	ICMP	ICMP	TCP/UDP	TCP/UDP	TCP/UDP	ICMP, IP, TCP/UDP
Metrics	throughput, delaytime, packet loss, etc.,	RTT	RTT	TCP throughput	Bandwidth	TCP/UDP throughput, packet loss rate	Throughput, delaytime, packet loss, etc.,	Throughput, delaytime, packet loss, etc.,	Throughput, delaytime, packet loss, etc.,
Remote control	yes	no	no	no	no	no	yes	no	yes
Flexible traffic	somewhat	no	no	no	no	no	no	no	yes
Multicast	yes	no	no	no	no	no	no	somewhat	yes
cost	very high	low	low	low	low	low	low	low	low

Pathchar can estimate the performance of each node along a path from a source to destination. Pathchar estimates the performance by using ICMP protocol's Time Exceeded response for various sizes of UDP packets whose TTL has expired.

5.2.2 UDP/TCP Tools

The major characteristics of active measurement tools using UDP/TCP are as follows. Almost all applications over TCP/IP networks use UDP/TCP to transmit their own data, and these tools therefore generate measurement pseudo-traffic using UDP/TCP. This pseudo-traffic can emulate general application traffic. However, both the sender and the receiver must execute the pseudo-traffic in order to measure the end-to-end performance of an application.

Ttcp is an active measurement tool using UDP/TCP. Ttcp is the legacy throughput benchmark or load generator. Ttcp can only measure maximum TCP/UDP throughput and packet loss rate.

Netperf is an active measurement tool for measuring the maximum TCP/UDP throughput, minimal, latency, TCP transaction speed (e.g., connection, request, response, disconnection), and the CPU utilization during test. Netperf supports many transport mechanisms (TCP, UDP, Unix domain, DLPI, Fore ATM, HIPPI, XTI, others).

KITS is a TCP/UDP performance measurement tool the authors have developed. KITS is able to generate any type of TCP/UDP traffic and measure the network throughput, delay time, jitter and data loss.

5.3 Comparison

Table 1 shows a summary comparing active measurement tools including STAGE.

6. Conclusion

This paper describes the flexible and general IP measurement tool, STAGE. Using STAGE, users can generate arbitrary traffic by choosing the transmission protocol and defining the generation functions of pseudo-traffic for measurement. Also, STAGE can boot from an httpd daemon from any node, and is applicable to stand-alone measurement using ICMP. Through these functions of STAGE, we can measure the IP performance generally and flexibly.

As a future topic, STAGE will be extended to support other functions, such as applicability to IPv6 and MPLS. Now we are examining use of a new specification to support IPv6 and MPLS.

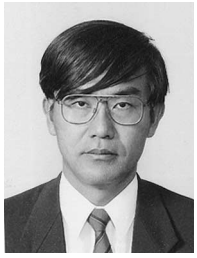
References

[1] M.Ishikura, Y.Ito, and T.Asami, “A traffic measurement tool for IP-based networks,” IEICE Trans. Inf. & Syst., vol.E82-D, no.4, pp.756-761, April 1999.
[2] SNMPv2 Working Group, J.Case, K.McCloghrie, M.Rose, and S.Waldbusser, “Structure of management information for version 2 of the simple network management protocol (SNMPv2),” RFC 1902, Internet Engineering Task Force, Jan. 1996.
[3] S.Waldbusser, “Remote network monitoring management information base,” RFC 1757, Internet Engineering Task Force, Feb. 1995.
[4] N.Brownlee, “Traffic flow measurement: Meter MIB,” RFC 2064, Internet Engineering Task Force, Jan. 1997.
[5] URL: ftp://ftp.arl.mil/pub/ttcp/
[6] URL: http://www.cup.hp.com/netperf/NetperfPage.html
[7] URL: ftp://ftp.arl.mil/pub/ping.shar
[8] URL: ftp://ftp.ee.lbl.gov/traceroute.tar.Z
[9] S.W.Richard, TCP/IP Illustrated, Volume 1: The Protocols. Reading, Addison-Wesley, 1994.
[10] URL: ftp://ftp.ee.lbl.gov/pathchar/
[11] URL: http://www.psc.edu/networking/treno_info.html
[12] E.Rosen, A.Viswanathan, and R.Callon, “Multiprotocol label switching architecture,” Internet-Draft, Internet Engineering Task Force, April 1999.
[13] K.Nichols, S.Blake, F.Baker, and D.Black, “Definition of the differentiated services field (DS field) in the IPv4 and IPv6 headers,” RFC 2474, Internet Engineering Task Force, Dec. 1998.
[14] S.Casner and F.Jacobson, “RTP: A transport protocol for real time applications RFC 1889,” Internet Engineering Task Force, Jan. 1996.
[15] S.Casner, “Compressing IP/UDP/RTP headers for low-speed serial links,” RFC 2508, Feb. 1999.
[16] S.Bradner, “Benchmarking terminology for network interconnection devices,” RFC 1242, Internet Engineering Task Force, July 1991.
[17] S.Bradner and J.McQuaid, “Benchmarking methodology for network interconnect devices,” RFC 2544, Internet Engineering Task Force, March 1999.
[18] O.Maeshima, Y.Ito, M.Ishikura, and T.Asami, “A method of service quality estimation with network measurement tool,” IEEE IPCCC’99, pp.201-209, 1999.
[19] D.Mills, “Network time protocol (v3),” RFC 1305, Internet Engineering Task Force, April 1992.
[20] V.Paxson, G.Almes, J.Mahdavi, and M.Mathis, “Framework for IP performance metrics,” RFC 2330, Internet Engineering Task Force, May 1998.
[21] G.Almes, S.Kalidindi, and M.Zekauskas, “A one-way delay metric for IPPM,” RFC 2679, Internet Engineering Task Force, Sept. 1999.

Yoshihiro Ito was born on March 25, 1968 in Nagoya, Japan. He received B.E. and M.E. degrees from Nagoya Institute of Technology, Aichi in 1991 and 1993. He joined Research and Development Laboratories of Kokusai Denshin Denwa Company Ltd. (KDD) in 1993 and has engaged in research on TCP/IP networks. He received the Young Engineer Award in 1999 from IEICE. He is currently a research engineer at the Network Engineering Support Laboratory of KDD R&D Laboratories Inc.

Yuichiro Hei was born in 1972. He received B.E. and M.E. degrees from Univ. of Tokyo in 1996 and 1998. He joined KDD R&D Laboratories Inc. in 1998 and has engaged in research on TCP/IP networks. He currently works at Network Engineering Support Laboratory of KDD R&D Laboratories Inc.

Masami Ishikura received B.E. and M.E. degrees in electrical engineering from Keio University, Kanagawa Japan, in 1984 and 1986. He joined R&D Laboratories of Kokusai Denshin Denwa Company, Ltd. (KDD) in 1986 and has been engaged in research on data link protocols, OSI, LAN and ISDN.



Tohru Asami received B.E. and M.E. degrees from Kyoto University, Kyoto, Japan. In April 1976, he joined Kokusai Denshin Denwa Co., Ltd., and has engaged in research on natural language processing, computer networks, knowledge acquisition, machine learning and expert systems. He is presently Vice President, Managing Director of KDD R&D Laboratories, Inc. He received the Young Engineering Award in 1984 from IEICE, Japan and the Maejima Award in 1988. Mr. Asami is a member of IEEE, AAAI and IPSJ.