Real-Time IP Flow Measurement Tool with Scalable Architecture

著者	Kitatsuji Yoshinori, Yamazaki Katsuyuki, Tsuru Masato, Oie Yuji
journal or publication title	IEICE Transactions on Information and Systems
volume	87
number	12
page range	2665-2677
year	2004-12-01
URL	http://hdl.handle.net/10228/00006350

PAPER Special Section on New Technologies and their Applications of the Internet

Real-Time IP Flow Measurement Tool with Scalable Architecture

Yoshinori KITATSUJI[†a)], **Katsuyuki YAMAZAKI**[†], **Masato TSURU**[††], ***Members***, ***and*** **Yuji OIE**[††], ***Fellow***

SUMMARY There is an emerging requirement for real-time flow-based traffic monitoring, which is vital to detecting and/or tracing DoS attacks as well as troubleshooting and traffic engineering in the ISP networks. We propose the architecture for a scalable real-time flow measurement tool in order to allow operators to flexibly define "the targeted flows" on-demand, to obtain various statistics on those flows, and to visualize them in a real-time manner. A traffic distribution device and multiple traffic capture devices processing packets in parallel are included in the architecture, in which the former device copies traffic and distributes it to the latter devices. We evaluate the performance of a proto-type implementation on PC-UNIX in testbed experiments to demonstrate the scalability of our architecture. The evaluation shows that the performance increases in proportion to the number of the capture devices and the maximum performance reaches 80 K pps with six capture devices. Finally we also show applications of our tool, which indicate the advantage of flexible fine-grained flow measurements.

key words: *IP flow, passive measurement, measurement tool*

1. Introduction

There is an emerging requirement for real-time flow-based traffic monitoring, which is vital to detecting and/or tracing DoS attacks as well as troubleshooting and traffic engineering in the ISP networks, instead of the existing IP-layer traffic volume monitoring or off-line flow analysis of collected traffic data. For example, fine-grained and user-defined flow monitoring is of practical importance for performance sensitive services such as Grid applications, while such monitoring on very high-speed links is a challenging task due to the large overheads to investigate the contents of every packet passing through the monitoring point.

The contents of traffic passing through Internet Service Providers (ISPs) are becoming diverse since various applications such as peer-to-peer, VoIP and so on, are widely used and DoS attacks occur frequently in their networks. It is getting more difficult in such networks to monitor the traffic of these applications as well as to detect and/or to trace DoS attacks with tools showing graphs of the whole IP layer traffic. Monitoring in such environments is required to classify traffic into flows.

On the other hand, the MPLS based traffic engineering requires monitoring flows to optimize usage of an entire administrated network. Managed networks such as Grid Computing research networks are expected to progress the optimization of flow controls based on both operational administrative policies and QoS requirements. A key issue is to detect what hinders a flow from reaching its target throughput. This kind of monitoring requires highly accurate flow measurement up to microsecond resolutions to understand how much bandwidth a flow consumes.

It is much more useful that flows can be specified by any field from the IP header up to application data in the payload as network operators require. Especially in the ISP operations, the following functions are very useful to understand the characteristics of problems and to reduce troubleshooting time: 1) extracting traffic flows flexibly and impromptuly specified by the operators on demand; 2) visualizing them in a real-time manner. However, the existing tools for such flow measurements have several limitations and drawbacks as mentioned in Sect. 2. In general, the existing software-based systems are suitable for relatively slow links, while the existing hardware-based systems cannot achieve sufficient flexibility.

In this paper, we propose the architecture for a scalable real-time flow measurement tool in order to allow operators to flexibly define "the targeted flows" on-demand, to obtain various statistics on those flows, and to visualize them in a real-time manner. The system we implemented based on the proposed architecture consists of the multiple capture devices, the manager device and the user interface devices. We also propose a bit-pattern-based flow definition method and its data structure to measure multiple flows with flexible flow definitions. Finally we report on the performance evaluation that the proposed system performs to measure flows with up to 80 K pps traffic with 6 capture devices with multiple flow definitions.

2. Related Works

MRTG is a tool for collecting Management Information Base (MIB) information (typically byte counters of router interfaces every five minutes) from remote network devices by using Simple Network Management Protocol (SNMP) and for visualizing time-varying characteristics of the information. MRTG is widely adopted in IP network operations because it is easy to use and automatically generates visual graphs and their HTML pages. MRTG theoretically can visualize flow-based traffic information based on RMON2-MIB [5] in cooperation with RMON2 enable devices. However, there are several limitations: e.g., RMON2-MIB is not

Manuscript received March 31, 2003.
Manuscript revised June 25, 2004.

[†]The authors are with KDDI R&D Laboratories, Inc., Kamifukuoka-shi, 356–8502 Japan
[††]The authors are with the Department of Computer Science and Electronics, Kyushu Institute of Technology, Iizuka-shi, 820–8502 Japan
a) E-mail: kitaji@kyushu.jgn2.jp

so flexible, RMON2 enabling devices are not so common, and collecting information cannot be performed in a short time interval due to the architectural limitation of SNMP.

NetFlow [6] provided by Cisco or Cflowd [7] developed by CAIDA [8] collect flow statistics generated by sampling packets passing through the router. Real-Time Flow Measurement Working Group of IETF suggests that per-flow basis counters kept in router to export statistics to collectors. The keeping per-flow counts consume considerable memory as well as processing power if the number of flows becomes large.

sFlow [3], which is also discussed in Network Working Group of IETF, is a specification to export raw data from sampling the traffic arriving at the switch besides the statistics. Some MIB of its statistics are shared with NetFlow MIB. The basic behavior of sflow export is that the sampled packets are chopped into a certain length and sent to multiple collectors. Per-flow network traffic measurement with sampled traffic has a trade-off between the sampling rate and the accuracy of measurement result as shown in [9], [10]. Some routers or switches have a function to capture every packet passing an interface, however such processes often cause performance degradation in the packet forwarding process, and thus, are adopted only to slow speed interfaces.

Anritsu Cooperation provides hardware-based traffic monitoring tools which measures flows passing through a GigabitEthernet link [2]. Tools can store measured flows at millisecond resolution timestamps and their playback function generates traffic as captured. The system limits to efficiently measuring up to four flows at the same time.

In this paper, we discuss the requirements of the flow measurement tool especially for network operation in ISP and propose architecture with scalability to meet high speed traffic and flexible flow definitions.

3. Requirements in Network Operations

3.1 Flow Measurement

Flow is defined as a set of packets passing an observation point in a network during a certain time interval and generally having the same 5-element-tuple of source IP address, destination IP address, protocol, source port number and destination port number [11]. However, we take flow in a wider sense to define as a set of packets having common properties specified by not only fields in the header but also application data in the payload.

We define flow measurement as clarifying the traffic properties derived from traffic changes, statistical length (total number of packets or bytes) and existing time (interval between first and the last packets) of flows. The flow measurement process consists of capturing traffic, flow identification, statistic processing and data preservation which described in Sect. 4. The flow identification process requires high performance and often expensive hardware for high speed links.

3.2 Requirements

In ISP operations, The traffic monitoring while classifying traffic based on applications is required to meet DoS attack, P2P utilization limitation and so on. The monitoring is required an ability to distinguish a certain application traffic from various ones. However some of applications cannot be distinguished based on port number of transport protocol. Therefore application data fields in a packet are also used in that case.

Additionally, ISPs are required to monitor the quality of traffic for customers who make a SLA contract with them. The ISP operations are required abilities to detect and monitor the precise change of traffic to troubleshoot and to analyze the depression of such customer's traffic performance regardless of the traffic volume. The followings are the requirements for flow measurement to support such ISP operations:

1. *Processing Scalability*: The measurement system should be extensible in terms of its packet processing power. Flow identification processes require processing power as traffic increase.
2. *Flexible Flow Definition*: To diagnose and troubleshoot the customer traffic, the ability to define a flow regardless its volume is important. The flexible flow definition by specifying a flow with application data fields in packets leads to smooth operations in the recent Internet carrying various application traffic.
3. *Operational Flexibility*: To support multiple users to obtain various statistics for flows, the system should be able to accept to update the flow definitions on-demand and, to visualize them in a real-time manner. Additionally these operations are done by one interface.
4. *Long-term Operation*: System should be able to keep working as long as possible. Any replacement such as storage to meet the limited room, updates of system configuration and parameters.
5. *Flexibility of Visualization Resolution*: The system should give operators a spatially fine-grained view such as traffic volume per routes or applications, and a temporally fine-grained view such as millisecond-order traffic behavior, which exposes the burst of traffic which looks flat rate under the low (coarse) resolution. Additionally the ability to change such resolutions as operators required is useful for the operation in troubleshooting.

We define the scalability for real-time flow measurement system as to realize all of these requirements in this paper.

3.3 Issues

In this section, we survey existing flow measurement system or techniques and compare them with the requirements in the previous section.

Table 1 Comparison of requirements enabled by different measurement systems.

	NetFlow	sFlow	NeTraMet	Anritsu	proposed system
Processing Scalability	○	○		•	•
Operational flexiblity	○	•	•	○	•
Flexible Flow Definition			○	○	•
Long-term Operation	•	•	•		•
Resolution Flexibility		•		•	•

•	enabled
○	limited

NetFlow consumes memory and make main processor overwork in routers as the increase of interface speed. Therefore sampling technology is recently deployed in NetFlow to follow such a high speed. Although some collectors such as FlowScan [12] provide multiple flow definitions, NetFlow running background of collectors exports all flow information and its overhead is not small between collectors and routers.

sFlow forwards sampled packets to collectors. The sampling rate is enlarged to follow high speed link. Although some collectors such as InMon Traffic Server [13] provide multiple flow definitions, it's hard to detect flows having small traffic volume compared to the sampling rate.

NeTraMet provides the flexible flow definitions by any combination of addresses or ports of Datalink, Network and Transport layer. The collection of measurement information is done with SNMP. Therefore it cannot be performed in a short time interval due to the architectural limitation of SNMP.

The hardware-based traffic monitoring tools from Anritsu Cooperation support various link speeds by replacing the network interface cards. The system limits to measure up to four flows at the same time and the high resolution visualization is done in the off-line manner.

Table 1 summarizes our overview of existing systems and the requirements realized by them. For proposed system we discuss in Sect. 7.1.

In the ISP operations, multiple tools are used because any system cannot provide the all requirements previously described. It is effective that a real-time flow measurement tool supporting all requirements is deployed in ISP operations.

4. Proposed System

Generally, flow measurement system consists of the following components.

- Packet Capture Component
- Flow Identification Component
- Status Preservation Component
- Analysis Component
- Data Preservation Component
- User Service Component
- User Interface Component

In order to realize the scalability which discussed in Sect. 3.2, we propose the role of each component in Sect. 4.1 and the architecture which combines these components in Sect. 4.2.

4.1 Enhancement of Component

We propose the enhancement of basic components to meet the scalability.

- *Packet Capture Component*: Copying traffic from Network device is done with the mirror function provided high-end Ethernet switches or network taps. The optical tap can follow the increase of link speed due to no conversion between optic and electricity. The system should have a buffer for input traffic to absorb burst input. The system should also have the mechanism which independently handles process buffering input traffic and post-process. To prevent to make the timestamp inaccurate, buffering should be done after obtaining the timestamp in a packet arrival.
- *Flow Identification Component*: This component examines every packet matching with the flow definitions. Generally, the burden of flow identification process increases as the increase of traffic rate and the number of flow identifications. Both are not avoidable to support the Processing Scalability and the Flexible Flow Definition. The improvement the process performance (e.g. by hardware implementation) is worried to cost enormously. As the other solution, deployment of multiple non-high-end devices to distribute burden of flow identifications process is expected lower cost such as in proportion to the traffic rate or so.
- *Status Preservation Component*: This component checks the flow status by inactivity timeout which defined by user.
- *Analysis Component*: This component calculates statistics as the way of the each measurement attribute described in Sect. 4.4. The calculation is done based on the flow granularity defined on-demand to meet the Flexibility of Visualization Resolution. The statistics are sent to Data Preservation Component.
- *Data Preservation Component*: As the increase of the number of flows, the system is required to meet the limitation of storage for measured data. The system should place the measured data in multiple places and change the place when the total amount of data stored in a place reaches a certain threshold which given by a user. Additionally the system should provide the information where the system currently accesses and history of its accesses. This enables recognize which data is safe to move.
- *User Service Component*: This component has two fea-

tures. One is to update flow definitions used in the Status Preservation Component, the Analysis Component and the Data Preservation Component. The flow definitions are updated form the User Interface Component. The other is to send stored measurement data to the one of User Interface Components. The User Service Component manages the multiple accesses from User Interface Components and should prevent that the flow definitions become inconsistent.

- *User Interface Component*: This component communicates with the User Service Component to exchange flow definitions and measured data, and visualizes graph in various resolutions requested by operators. Traffic graphs are automatically updated by periodically collecting measurement data.

4.2 Proposed Architecture

We propose the architecture that pipeline the process of components and deploy Flow Identification Component to distribute burden of flow identification process over multiple devices to have good scalability. To manage the multiple Flow Identification Components in the system, we propose the Distribution Components and additional functions.

- *Distribution Components*: This distributes monitoring traffic to the multiple Flow Identification Components. Distribution Component is a key that the system follows the input traffic rate. Therefore its process should be as simple as possible to be implemented on hardware.
- It's not expected that the Distribution Component distributes traffic based on the flow definitions, which makes the distribution process complex. Therefore the Flow Identification Components should maintain the all flow definitions as considering the Distribution Component adopts a round-robin distribution for its simplicity.
- The system should be able to add and drop the Flow Identification Components to follow the requirements of process power while the system is running. Both the Distribution Components and the Analysis Component detect an addition of the Flow Identification Components. The information of Flow definition should be shared between the Status Preservation Component and the Flow Identification Components. The Distribution Component should be able to detect to stop distribution to the Flow Identification Components which stops its process or is removed. This function meets the Processing Scalability and the Long-term Operation.
- The Status Preservation Component re-orders the measured data based on the data timestamp. The measurement data is asynchronously sent from the Flow Identification Components.

Figure 1 shows the combination of the components described in Sect. 4.1 and dataflow through them to pipeline

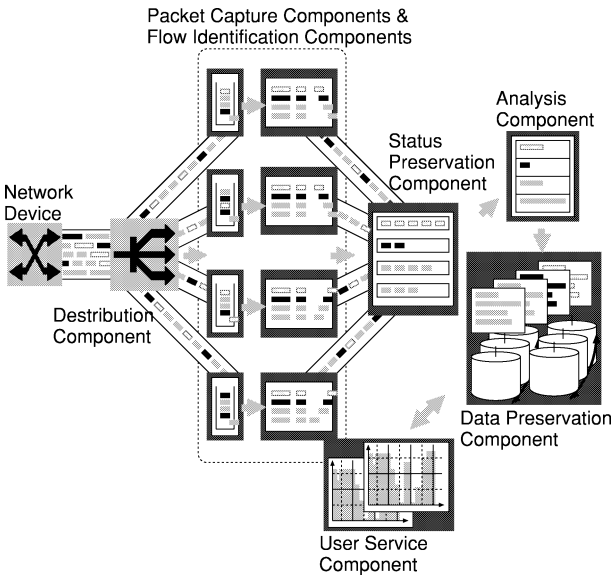

Fig. 1 Architecture of a distributed real-time flow measurement tool.

the measurement process. The basic flow of process is as follows:

1. Traffic is copied by an Ethernet switch or an optic tap and forwarded to the Distribution Component.
2. It then forwards packets to one of devices having the Packet Capture Component and the Flow Identification Component.
3. The Flow Identification Component updates measurement data if the packet matches with flow identifications.
4. The Flow Identification Component periodically send the local measurement data to the Status Preservation Component.
5. Partial measurement data from the Flow Identification Component is reordered and flow status is checked in the Status Preservation Component.
6. Statistics of measurement attribute calculated in the Analysis Component and it sent to Data Preservation Component.

This architecture enables the system to update flow definitions on-demand (the Operational Flexibility), to move the accumulated data and to add or drop the Flow Identification Components without stopping the system (the Long-term Operation), to accept the various measurement granularities of time-scale (the Flexibility of Visualization Resolution).

The Flexible Flow Definition discussed in the following section.

4.3 Flow Definition and Data Structure

We define a flow definition as a set of chained bit-patterns. The bit-pattern is defined by elements as described in Table 2. A flow definition consists of the multiple bit-patterns. The 'chain' of bit-pattern implies the 'AND' operation for

pattern matching.

When the capture device receives a packet, it picks up a bit field specified by parameters of a bit-pattern. We call this bit filed a flow identifier (FID). The original packet is judged as matching with the bit-pattern if its FID is between a range specified by the minimum and maximum pattern. The FID obtained from the last bit-patterns chained for a flow definition is used as a key to search and update measurement attribute described in Sect. 4.4. The combination of a flow definition and the last FID specifies a flow in our architecture.

By taking a flow definition into multiple bit-patterns, the multiple flow definitions can be converted into hierarchical chained bit-patterns to eliminate duplicate pattern-matching per packet. Figure 2 shows an example of hierarchical chained bit-pattern for measuring WEB and NEWS traffic. Bit-pattern: "IPv4 TCP" is used when the packet is judged as IPv4 packet by bit-pattern: "IP version" which checks version field of IP header. Both "IPv4 TCP DST WWW" and "IPv4 TCP DST NEWS" share "IPv4 TCP" and "IP version." Although the total number of bit-patterns from two flow definitions (WEB and NEWS) is six, that of hierarchical bit-patterns becomes four. For instance, when a UDP packet is examined with those bit-patterns, it doesn't match with "IPv4 TCP." Hence, the following pattern matches aren't done.

Table 2 Elements specifying the bit-pattern.

Pattern ID	Identifier used for reference by other bit-patterns
Position	Position of bit-pattern from a top of IP packet
Length	Length of bit-pattern
Mask	Specifies a valid and invalid bits
Minimum Pattern	Minimum value to specify range of bit-pattern
Maximum Pattern	Maximum value to specify range of bit-pattern
Child Pattern ID	Reference list of chained bit-patterns

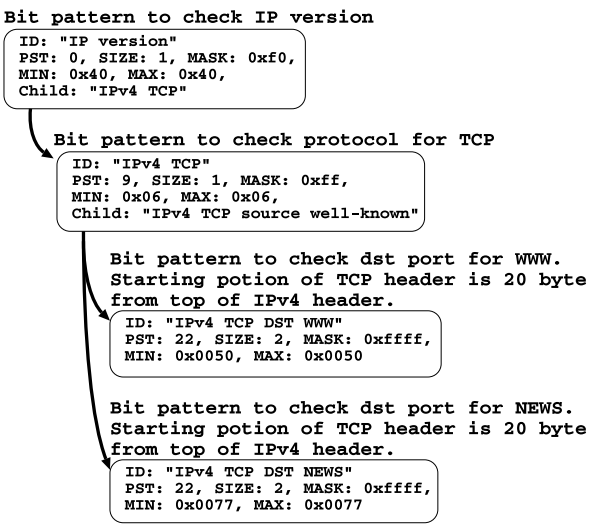

Fig. 2 Example of hierarchical chained bit-patterns.

4.4 Measurement Attribute

We define four measurement attributes for a flow definition:

- *Flow Count*: Number of packets and bytes at certain intervals defined by users.
- *Flow Length*: Statistics on total number of bytes, packets and duration of a flow.
- *Packet Gap*: Statistics on timestamp intervals of consecutive packets composing a flow.
- *Association Packet Gap*: Statistics on timestamp intervals of two packets matching two different flow definitions.

When the Flow Count or the Flow Length are enabled, each of the multiple flows detected by a flow definition has its counters or length values. The Packet Gap is assorted the only flow definitions of exact pattern matching in which maximum and minimum patterns are the same. A flow definition can have multiple measurement attributes. The intervals for the Flow Counter, the Packet Gap, the Association Packet Gap and values of durations of the Flow Length have microsecond precision.

The Association Packet Gap calculates statistics from timestamp intervals of two packets that are expected to pass an observation point in order and that need two different flow definitions to detect respectively. E.g., a pair of packets having a flag SYN and FIN in the TCP header respectively to measure the duration of the TCP connection. The first flow definition is used to match the first packet, and then FIDs and timestamps are collected. The FID is substituted for the minimum and/or maximum patterns of the last bit-pattern for the second flow definition as users previously define. The second flow definition is generated with first packet FID and a lifetime whenever a packet matches with the first flow definition. Hereafter, the second flow definition begins to be used to detect the second packet expected to appear after the first one. The second flow definitions are released when a packet matches with it or its lifetime expires before a packet matches. After a packet matches with the second flow definition, timestamp intervals are computed from both the first and the second packets.

The user can limit the number of FID collected from the first flow definition to prevent the FIDs consuming resources in case that a long lifetime is given. When the number of FIDs reaches its limit, the following FIDs are discarded until the active FIDs are released.

5. Implementation Issues

In this section, we argue the implementation issues for implementing the system with the PC-UNIXs as the examples based on the proposed architecture, time accuracy, time synchronization and FID search.

5.1 Implementation with PC-UNIXs

In this section we discuss the implementation of proposed architecture with PC-UNIXs. The following shows the composition and physical devices:

- *Distribution device*: As explained in Sect. 4, we assume to use a hardware-based general-purpose distribution device as the Distribution device.
- *Capture device*: The Capture device consists of the Packet Capture Component and the Flow Identification Component.
- *Manager device*: The Manager device consists of the Status Preservation Component, Data Preservation Component and User Service Component.
- *User Interface device*: The User Interface device consists of the User Interface Component.

The capture device consists of three thread processes; Packet Buffering Process, Flow Identification Process, Report and Definition Update Process. Report and Definition Update Process periodically reports measurement data to the manager device and receives update of flow definitions. The Report and Definition Update Process derives or releases the Packet Buffering Process and the Flow Identification Process after that the capture device starts. All processes share flow definitions and measured flow information constructed in a memory space. For capturing packets, PCAP library [14] was adopted It provides timestamp and an interface to collect capturing loss to application programs attempting to detect and count. It is also supports both IPv4 and IPv6, and makes it easier to implement/port a capture device on/to the variety of OSs.

The manager device consists of four thread processes; Main Process, Definition Advertisement and Collection Process, Save Process, User Interface Service Process. Main Process manages the connections established from the capture devices or the user interface devices and derives other three processes. Definition Advertisement and Collection Process receives reports from the capture devices and advertises flow definitions when detecting any difference between registered definitions and definitions in a report. The Status Preservation Component and the Analysis Component are realized in this process.

Time synchronization is required between the multiple capture devices to prevent inconsistent reports between them. The accuracy of timestamp and method of time synchronization are discussed in the following sections.

5.2 Time Synchronization

In case that timestamp is given by the capture devices, the packet arrival time becomes inaccurate against the actual arrival time at the distribution device. The factors of this are differences of 1) packet forwarding delay and 2) process delay to take in packets in the capture device. The timestamping by the capture devices cannot avoid these factors even time synchronizes between capture devices completely.

1 can be minimized if the number of hop between the distribution device and the capture devices is few (e.g. 1 hop) and the connections between them consists of the same length cables and high speed device (e.g. GigabitEthernet switch). Although 2 is caused by the burden of flow identification or other tasks running concurrently, the computer which exchanges gigabit class traffic is expected to have small delay. E.g. A computer which receives 1 Gbit/s (MTU: 1500B) takes in 12 microsecond per packet treatment in average. The roughly estimated accuracy is up to 1 millisecond, since the implementation is based on software in this proposal.

Above discussion is based on that time synchronizes between the capture devices. For the time synchronization, there are several clock sources are available as follows:

- *GPS*: GPS requires a sky view for an antenna. The establishment of antenna may be limited the building maintenance or security policies of data-centers where the system installed.
- *Atomic Fountain Clock*: The facility locations are limited. Wide area deployment is difficult.
- *CDMA*: CDMA (Code Division Multiple Access) is becoming a widely popular air interface for mobile telephony. The ntp (a free NTP server [16]) supports CDMA as its clock source. CDMA brings better chance to deploy in building than GPS if its location is in the area of CDMA.
- *Clock Generator*: It's hard for the clock generator to deploy in wide area. The ntp also supports a few clock generation devices.

Note that each scheme can take both forms to receive clock signal from source and distribute PPS (pulse per second) to multiple receivers and then to computer, or to receive clock signal by receiver through an antenna and distribute PPS to multiple computers. In the implementation of proposed architecture, any clock source is available because capture devices should be installed in the same LAN segment or in few hop topologies to get better timestamp accuracy as discussed above. The selection of clock source is done with the other criteria, Wide area deployment, establishment of antenna and so on.

5.3 Flow ID Search

The range definition of a flow definition results detecting multiple flows in the system. Each flow matched with a flow definition is distinguished by the last FID of chained bit-patterns.

When a packet matches a flow definition, the same FID is searched from the previously registered FIDs. If the same FID does not exist at that moment, a new FID is registered. In case that many FIDs are registered due to large range size of bit-patterns, The FID search tends to take more time.

For searches with a certain length key such as the FID, a binary search can reduce the processing cost to $O(\log n)$

where n is the total number of FIDs registered. However the Flow Identification Component registers FID and the search tree grows gradually thus the search tree must be well balanced. The AVL tree [17], which keeps the search tree balanced, is one of solutions. However the insertion of a new FID is costly as opposed to normal list process, thus it is better to discover in what range size of bit-pattern the AVL tree shows better efficiency.

6. Evaluation

Firstly we evaluated if the system works as we expected. The follows are confirmed.

- Flow definitions specifying any fields of a packet are accepted by the system and correctly match packets with appropriate flow definitions.
- The system accepts updates of flow definitions with any measurement attributes on-demand and hereafter it measures and stores statistics of requested attributes.
- The system can keep to run while the capture devices are added or dropped. The measurement data preservation function works correctly.
- Traffic graphs are shown on the user interface device and its auto update function works correctly.
- The system accepts multiple access from the user interface devices and prevents conflicts of flow definitions updates.

These evaluations imply that the system satisfies the requirements except the Processing Scalability. In the rest of this section, we evaluate the performance and scalability with follows experiments:

1. *Evaluation of capture device process performance when the Packet Buffer Process is used or not.* The distribution device asynchronously send packets to the capture device. The capture device is not ready to take a packet when it arrives, due to the previous packet process engaged. We evaluate how the deployment of packet buffering process is effective in the packet capturing process.
2. *Evaluation of the performance difference of capture device in deploying AVL Tree Search or the Sorted List Search.* We evaluate how the performance of a capture device becomes difference in deploying AVL Tree Search and Sorted List Search. The overhead of FID search may have an influence on the performance of capture device.
3. *Evaluation of how the entire system performance increase based on increment of the number of the capture devices.* We evaluate the improvement of measurement performance against the number of capture devices.

6.1 Evaluation Environment

We performed two kinds of test for the evaluation 1 and 2, and for the evaluation 3. The former examined the process performance of a capture device by measuring capturing loss which is caused by an overwork of processing. The traffic is sent with 500, 2 K, 4 K, 8 K, 10 K, 12 K, 15 K, 18 K, 20 K, 24 K, 27 K, 30 K packets per second (pps) respectively from a traffic generator to a capture device in Fig. 3 configuration. Flow definitions are registered in a manager device through a user interface device. The manager device advertises flow definitions to the capture device. The capture device makes Flow Counter reports for all FIDs to the manager device every 10 seconds. The performance is defined as the maximum speed in which there is no capturing loss for 60 seconds generation over 10 times examinations. The traffic generation is shaped by an application that authors developed to keep the short jitter at the user program level. However small burst traffic was observed during generation because the final packet treatment was controlled by a kernel.

The other evaluation is a performance test of the entire system for given 4, 8, 16 and 32 bit-patterns and the number of capture devices.

Figure 4 shows a configuration of the performance test. Although the configuration does not match with proposed architecture, each traffic generated from a generator is assumed as the traffic from distribution device in the practice. This can evaluate the entire system performance by the total the generated traffic. In this test, we used GPS for time synchronization clock source for all capture devices. The clock

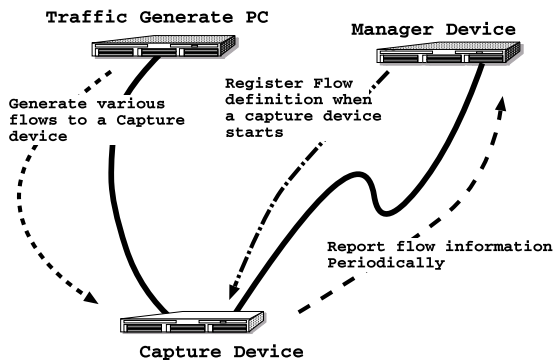

Fig. 3 Configuration of performance evaluation.

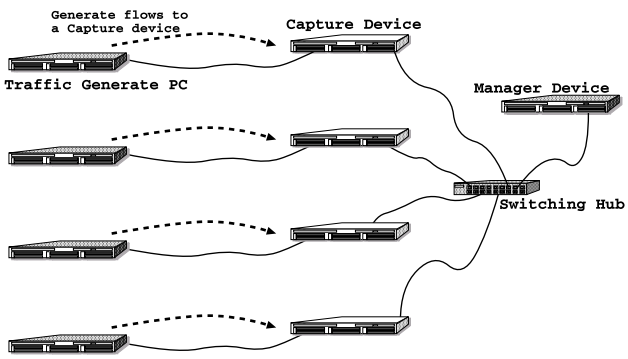

Fig. 4 Configuration of entire system performance test.

signal received at an antenna was divided up to 6 receivers and they provided plus per second to the corresponding capture devices.

We define the performance as the maximum value of total packet speed generated by the generators for 60 seconds without any measurement loss. The maximum value was searched by binary search of packet speed. Bit-patterns were advertised to capture devices from the manager device every 10 second. Measured values of counter attribute were reported to the manager device every second. The precision of counter attribute was 10 milliseconds.

All computers in both evaluations had the same specifications as shown in Table 3.

6.2 Performance Difference in Deploying or Not Deploying the Packet Buffering Process

Figure 5 shows the process performance of capture device in which 1 up to 64 bit-patterns are registered respectively. Traffic which had bit-patterns to match with one of flow definitions was sent to the capture device. The bit-pattern appearance in generated packets was done in a round-robin manner. The length of a bit-pattern was four bytes and an exact match defined. This implies that the result is either matched or unmatched.

The maximum performance was 27 K pps where the number of bit-patterns was 1 and that the minimum performance was 2 K pps where the number of patterns was 64 when Packet Buffer Process was deployed in the capture device. The appropriate number of patterns is expected to be between 4 and 64 in practice, therefore the performance per capture device would be between 2 K to 15 K pps. As described in Sect. 5.1, we adopted PCAP library for capturing packets. The timestamp is already obtained before the packet is buffered. The delay while packets are buffered doesn't cause inaccuracy on their arrival time.

On the other hand, the packet loss was observed on 500 pps traffic when the Packet Buffer Process was not deployed. From 2 experiments, the Packet Buffering Process is effective.

Table 3 Specification of PC-UNIXs used for evaluation.

CPU	Xeon 2.8 GHz
Memory	2 GB
HDD	73 GB
Bus	PCI-X (64 bit, 133 MHz)
Network Interface	2 ports, 10/100Base-TX
Operating System	RedHat 9 Linux kernel 2.4.20

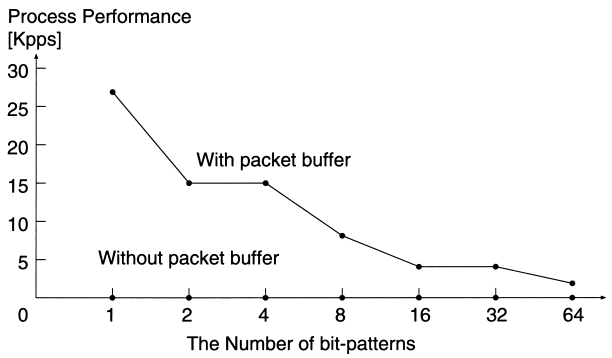

Fig. 5 Process performance per number of bit-patterns with or without the packet buffer.

6.3 Performance Difference in Deploying AVL Tree Search or Sorted List Search

Figure 6 shows the capture device process performance when the range size of the bit-pattern is between 64 and 8192 respectively. The FID in packets which matched with the flow definitions was generated in a round-robin manner, so the number of each FID in packets became the same between FIDs. The length of the bit-pattern was four bytes.

From the results, the criterion to select a search from the AVL tree search or the sorted list search on the proposed system was 1024 in point of range size view. It may be effective if each flow definition statically selects one of searches such as the AVL tree search and the sorted list search with a criterion of the range size of the bit-pattern when the flow definition is registered in a capture device. The range size of a bit-pattern composing a flow definition is previously obvious when the flow definition is registered.

6.4 Scalability Evaluation

Figure 7 shows the average performance from 10 examinations for each number of bit-patterns and the capture devices. The exact pattern matching with 4-byte-length bit-patterns was used. All bit-patterns was specified at the same position in a packet. The FID field of generated traffic picked up by the capture device is generated in round-robin manner to match with the bit-patterns with the same frequency. Therefore every bit-pattern matched with 1 nth of total traffic, where n is the number of given bit-patterns. The Flow Counter for the measurement attribute was used and the sorted list for the FID search to increment counters was adopted.

The performance increased in proportion to the number of capture devices and it reached 80 K pps with six capture

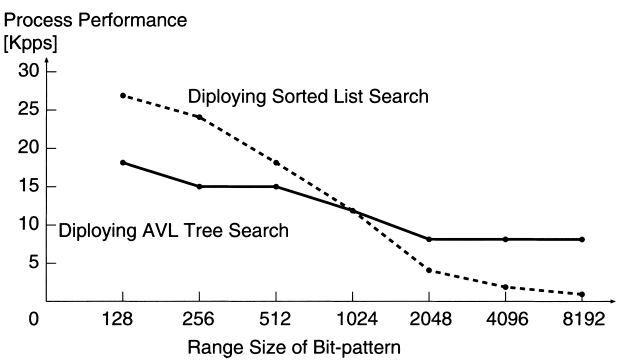

Fig. 6 Process performance per range size of bit-pattern with AVL tree search or sorted list search.

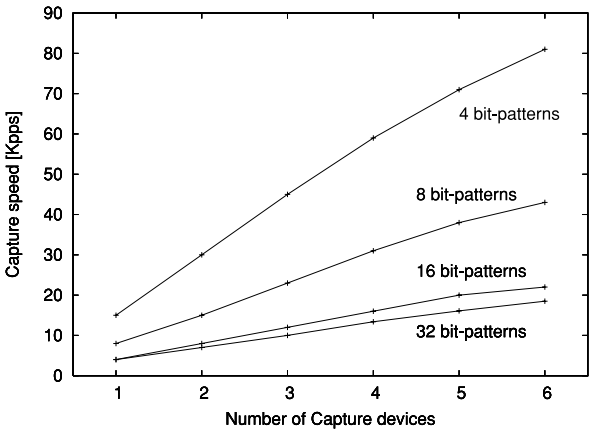

Fig. 7 Performance of entire system.

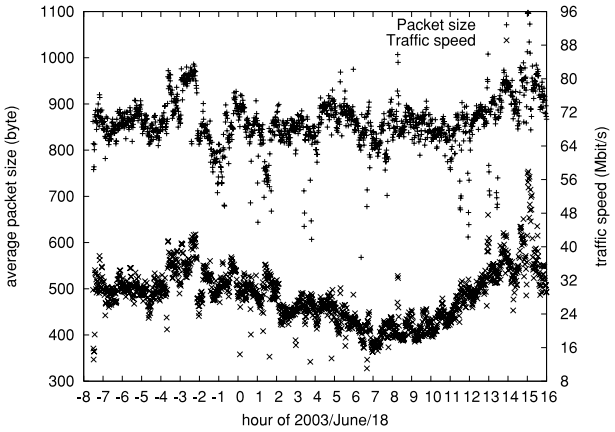

Fig. 8 Packet length of traffic passing between APAN and WIDE.

devices.

7. Discussions

7.1 Evaluation Results

From the results of Fig. 5 in case the packet buffering process deployed, it is clear that the number of bit-patterns per packet significantly impacts on performance. We assume that the proposed system performs with the multiple flow definitions and that each of them consists of the multiple bit-patterns. The hierarchical bit-pattern structure is expected to prevent the performance degradation as the number of reduction of bit-patterns.

In order to improve the performance, it would be better to modify the order of chained bit-patterns operating as 'and' operations between bit-patterns while the system is in progress. This tuning to reduce the average number of pattern matching per packet depends on the traffic measured. The dynamic matching order method is expected to reduce the entire pattern matching overheads.

For the reference of the proposed system performance, we show an example of average packet length from a real world network that we investigated. Figure 8 shows the 1 minute average packet length ('+') and average traffic ('×') of the traffic passing through a GigabitEthernet link between APAN Tokyo XP [18], [19] and WIDE [20] measured in June 18, 2003. The average packet length on that day was 861 bytes. The results of this packet length are only an example, yet the performance of 2 k–15 K pps per the capture device is estimated at 13 M–100 M bit/s. 80 K pps performed by 6 capture devices of Sect. 6.4 is estimated at 600 M bit/s if observed traffic have 861 bytes which is same as the average packet size.

As we showed in the Sect. 6.4 the performance of the entire system increases in proportion to the number of the capture devices and the proposed architecture addresses scalability by deploying multiple capture devices. However, we ascertained the reason of a little performance falls at six capture devices. It caused by once or twice of significant performance failure in 10 examinations. The performance failure came from the rise of the delay of data processing at the threads of the Definition Advertisement and Collection Process in the manager device, due to the large volume reports for Flow Counter from six capture devices. The long delay caused significant gap of data processing between capture devices and manager devices and thus resulted to discard delayed reports beyond a given threshold.

Using multiple capture devices in the proposed system is a key to achieving high-performance measurement of the IP flow on a high-speed link. However, the manager device can be a bottleneck for the performance of the entire system. The one of solution may be further to separate the components, the Status Preservation Component and the Data Preservation Component, to multiple devices.

Through all evaluations, we could show that the proposed system provides scalability (Table 1) by achieving the requirements defined in Sect. 3.2. To improve the further performance scalability, we investigate and develop the distribution device and the manager device as future works.

7.2 Performance

As discussed in Sect. 3.2, existing hardware-based flow measurement system has limitations such as the number of flow definitions given at the same time to follow wire speed supported by its NICs. E.g. MD1230A series from Anritsu products accepts up to 4 flows [22]. On the other hand, our proposed system provide the interface to define flows flexibly and have no limitation the number of the flow definition and thus give users the convenience and flexibility in their operations. Additionally, although the proposed system composed by a few computers is inferior to the hardware-based system in the performance aspect, the proposed system can bring the performance close to hardware-based system by adding computers.

Comparing to existing software-based systems, Argus [21], for instance measures flows specified by addresses from Datalink to Transport layers. Although Argus basically can measure all flows passing the observation point, it con-

straints users to obtain measurement data aggregated by addresses or port numbers such as IP prefix in the offline manner with proper scripts because it doesn't provide the interface for user to define flows used in the on-line manner. On the other hand, NeTraMet can collect measurement data aggregated by the combination of addresses and/or port numbers from Datalink to Transport layers due to the interface for user to define flows using these addresses. NeTraMet adopts hashing algorithms to handle flow identification process particularly for these address treatment to follow high traffic rate [1].

As described in Sect. 4.3, besides the collection of aggregated flow data as NeTraMet provides, our proposed system provides better interface for the flow definition that user can flexibly specify flows using any field of packets. This flexibility enables us various operational applications such as to measure round trip time for TCP flows (described in Sect. 8.2), to monitor SYN flag of TCP to detect Syn Flood attack, to monitor connections carrying abnormal volume traffic and so on. To improve measurement performance in the condition supporting such flexibility in the flow definition, we design the data structure and architecture which reduce the number of pattern matching per packet.

For compare these software-based systems including the proposed system, it's hard to find the significance from the performance comparisons under the identical conditions because these systems have different design and implementation originating in their different usage fields. However, note that the performance of all systems rather depends hardware specification of computers on which the software of the systems implemented, at least. Under this point of view, the proposed system can follow the traffic rate without the dependency of the computers performance by adding the number of computer in the system, as the evaluation showed in Sect. 6.4.

7.3 Timestamp Accuracy and Distribution

We proposed that the capture device use its own clock in the implementation. In the Sect. 5.2 we discussed that timestamp becomes inaccurate in the software-based system. To measure high speed links, high time precision is required (e.g. at least 38 nanosecond to detect 48 byte frame on OC-192 link). Both delay variations of packet forwarding and software-based packet capturing are hardly accepted to meet such high speed links. Therefore timestamp process should be implemented in the distribution device of the hardware based implementation and both delay variation should be removed.

The timestamp function realized in a distribution device requires the capture device to obtain a timestamp from a packet and to shift the start or end point of the packet for the timestamp field. This also eliminates the need for clock synchronization between the capture devices and hence system becomes simpler.

8. Application and Advantage of Flow Measurement

8.1 Example Real World Environment

Figure 9 shows an example of the flow measurement of packet speed on the same GigabitEthernet link between APAN Tokyo XP and WIDE as shown in Fig. 8. The measurement was performed with a 2-capture-device system. The capture devices made reported every second. The distribution was performed in a round-robin manner using a prototype we developed with PC-UNIXs for this measurement. The total average and maximum traffic speed was 0.92 K pps and 2.8 K pps. The bit-pattern definitions were for checking the protocol field of the IP packet (1-byte length) and the destination port field of TCP header (2-byte length). There was no capturing loss detected during measurement. The resolution of the graph is in milliseconds.

The procedures for graph generation of Fig. 9 with the proposed system were as follows:

1. Register flow definitions with the manager device
2. Measure flows with the multiple capture devices and store date to a manager device
3. Check graph for each flow definition by using the user interface device
4. Dump flow information in a specific period to text file
5. Generate a graph with an appropriate tool such as RRDTool [23], Gnuplot [24], Microsoft Excel, etc.

Steps 1 to 3 were repeated until the graph that the user wanted was shown on the user interface device. It was easier to extract certain flows at the observation point because measurement based on the flow identification definition registered started just after its registration, and the graph showed up without delay. The operation with the proposed system effects to help operators look and feel for the traffic status without complex procedures.

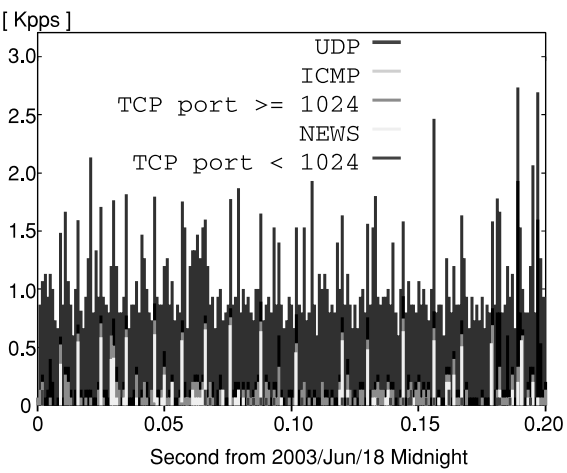

Fig. 9 Flows of outgoing traffic to WIDE.

8.2 RTT Measurement

RTT measurement is an example of the use of the Associated Packet Gap. RTT measured passively is not common because there is not always traffic passing a path of interest nor is it able to specify the probe packet size, interval of consecutive probes and measurement period. RTT itself is essential for the buffer size decision of high performance TCP transmission in a delay environment or for path selection for delay sensitive real-time applications such as VoIP or contents streaming. However RTT inference with an active measurement tool may cause competition between application traffic and the measurement traffic when measurement is performed before the application traffic is sent. The passive approach is more scalable and becomes more appropriate if the RTT inference for optimization of application traffic becomes more common [25].

A TCP connection is suitable to measure RTT passively because a packet pair of data and its acknowledgment (ACK) can be the probe. To collect timestamps of the pair, the proposed system needs to receive bidirectional traffic at an observation point and to have two flow definitions to specify data packets and its ACK packets, which have replaced source/destination addresses and ports between them. The first definition is used to collect the timestamp of a data packet and the FID matching with the sequence field bit-pattern. The second is for the timestamp used to compute interval between the data packet and itself matching with the ACK sequence field bit-pattern. Both definitions are connected by the Association Packet Gap attribute described in Sect. 4.4

Receiving the ACK sequence at a data source host implies that all the data sequences previously sent less than ACK sequence are received at a destination host, even though the number of ACK sequences doesn't match with one of the data sequences exactly. Therefore the FIDs (sequence of data packet) collected by the first flow definition should be applied to the minimum value for range matching of the second flow definition to match with an ACK packet corresponding to the data packet. The maximum value of the second flow definition should be "$0xffffffff$." FIDs of the first flow definition around the highest portion of a sequence before rounding back 0 may not match the ACK sequence, because the highest ACK sequence portion may be omitted by the Delayed ACK [26] (DACK) and round back to the low value. Therefore it is a good idea that the first flow definition uses a range matching not to collect the highest portion of the data sequence, such as the multiple size of MTU as an example. Figure 10 shows an example of two flow definitions bound by the Association Packet Gap attribute to measure RTT from a TCP flow.

Figure 11 shows an example of RTT phase plot [27] showing minimum RTT realm, the correlation between two adjacent RTT values and congestion transition. The RTTs for this phase plot was collected by the proposed system from the traffic of both ways passing the same GigabitEthernet link between APAN Tokyo XP and WIDE shown in Fig. 8. The target flow lasted for 4.589 seconds and 1128 RTT data was collected. The X-axis ticks in that plot represent RTT_n and the Y-axis ticks represent RTT_{n+1}, where n and $n+1$ are indexes of RTT data. Figure 11 shows that minimum RTT including TCP process time at end system is 8.3 ms, minimum RTT realm (E) is 0.6 ms. [27] describes that three regions of *I*, *II* and *III* derived from the phase plot imply *no congestion*, *transient congestion* and *persistent congestion*, respectively, as follows:

- *Region I* contains probe pairs that experience minimum RTT and minor random overhead E due to router, media or end host's TCP processing.
- *Region II* contains probe pairs that experience the beginning of congestion on the path.
- *Region III* contains probe pairs that experience persistent congestion due to packets are queued in routers or an end host raising process burden.

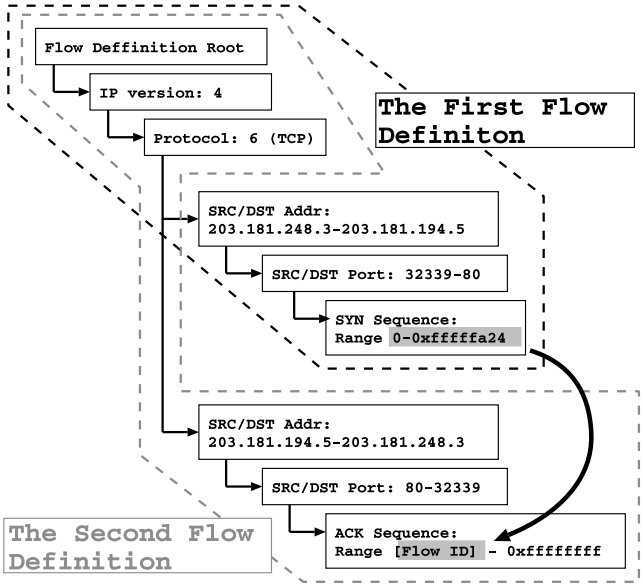

Fig. 10 An example of flow definitions to collect RTTs from a TCP connection.

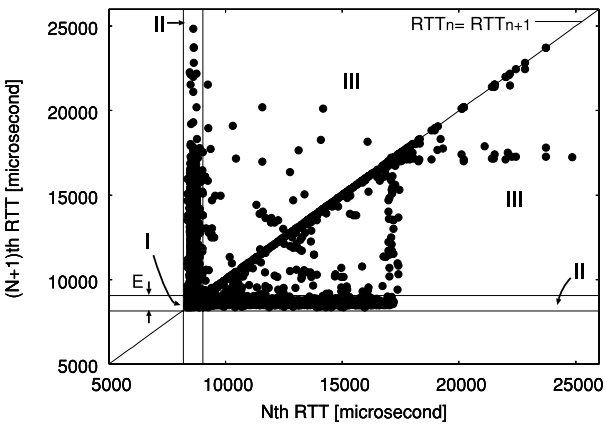

Fig. 11 An example of phase plot showing different congestion region.

We should consider DACK and retransmission to use TCP's data and its ACK packets for the passive RTT measurement. DACK is done when a receiver acknowledges every multiple data packets received or when no acknowledgment is sent of more than 200 milliseconds. Therefore the region *II* and line $R_n = R_{n+1}$ become filled by plots of enlarged RTT caused by DACK besides congestions on the path even though there is no loss with the target flow. The retransmission also shows large RTT in Region *II* for the beginning of loss and *III* for the rest of the following. For the accurate inference to detect congestion on the path, It is a good idea to extract minimum RTT from multiple RTTs computed from an ACK packet arrival. This extension requires small modification on the to the Association Packet Gap attribute, but a greater effect is expected.

9. Conclusions

We discussed the requirements for flow measurements that help us understand how the network is being used in a spatially fine-grained view (e.g., not only total traffic volume but traffic volume per route or applications) and in a temporally fine-grained view (e.g., not only five-minute average but millisecond-order behavior) even on high-speed links. Based on them, we proposed an architecture of scalable and flexible real-time flow measurement tools, which adopts multiple capture devices for processing packets in parallel, hierarchical chained flow definitions for reducing average overheads of pattern-matching, and an adaptive tree data structure for dealing with a large number of active flows.

To evaluate the proposed architecture, we then developed a proto-type implementation of the tool on PC-UNIX and examined its performance in testbed experiments, demonstrating that it could measure statistics on flows defined by complex and on-demand definitions on a gigabit class link. Even by the proto-type system on nominal PC-UNIX machines, the maximum performance reached 80 K pps with six capture devices.

We also showed applications of our tool in real environments. The real-time fine-grained flow visualization enables us to understand what types of traffic are dominant and, at the same time, how they are behaving over a short time-scale. Monitoring bidirectional TCP flows enables us to infer RTT behavior without sending any probe packet.

The development of the distribution device and the evaluations of its performance and that of entire system require further investigation.

Acknowledgments

The authors would like to thank the Asia-Pacific Advanced Network and the WIDE projects that accepted traffic measurement. This work is partially supported by the National Research Grid Initiative (NAREGI) Network Project, Ministry of Education, Culture, Sports, Science and Technology, Japan.

References

[1] NeTraMet web page, http://www2.auckland.ac.nz/net/NeTraMet/
[2] MD1230A-20, MX123001A-20, "Application traffic monitor," Anritsu Corporation, http://www.anritsu.com/
[3] P. Phaal, S. Panchen, and N. McKee, "InMon corporation's sFlow: A method for monitoring traffic in switched and routed networks," Request For Comments 3176, Informational, Sept. 2001.
[4] Multi Router Traffic Grapher, http://www.mrtg.org/
[5] S. Waldbusser, "Remote network monitoring management information base," IETF Request for Comments 2819, Standards Track, May 2000.
[6] Cisco Systems, Inc., "NetFlow service and applications," White paper, http://www.cisco.com/warp/public/cc/pd/iosw/ioft/neflct/tech/napps_wp.htm, 2002.
[7] CAIDA, "cflowd: Traffic flow analysis tool," http://www.caida.org/tools/measurement/cflowd/
[8] The Cooperative Association for Internet Data Analysis, http://www.caida.org/
[9] N. Duffield, C. Lund, and M. Thorup, "Properties and prediction of flow statistics from sampled packet streams," Proc. ACM SIGCOMM IMW 2002, pp.159–171, Marseilles, France, Nov. 2002.
[10] N. Duffield, C. Lund, and M. Thorup, "Charging from sampled network usage," ACM SIGCOMM IMW, San Francisco, CA, Nov. 2001.
[11] J. Quittek, T. Zseby, B. Claise, and S. Zander, "Requirements for IP flow information export," Internet Draft (draft-ietf-ipfix-reqs-10.txt), http://www.ietf.org/internet-drafts/draft-ietf-ipfix-reqs-10.txt, June 2003.
[12] FlowScan – Network Traffic Flow Visualization and Reporting Tool, http://www.caida.org/tools/utilities/flowscan/
[13] InMon Corporation, http://www.inmon.com/
[14] Tcpdump/Pcap, http://www.tcpdump.org/
[15] D.L. Mills, A. Thyagarajan, and B.C. Huffman, "Internet timekeeping around the globe," Proc. PTTI Applications and Planning Meeting, pp.365–371, Long Beach, CA, Dec. 1997.
[16] NTP: Network Time Protocol, http://www.ntp.org/
[17] G.M. Adelson-Velskii and Y.M. Landis, "An algorithm for the organization of information," Soviet Math. Dokl., vol.3, pp.1259–1262, 1962.
[18] Asia-Pacific Advanced Network, http://www.apan.net/
[19] K. Chon, "Asia-Pacific advanced network (APAN)," http://apan.net/home/index1.html, 1999.
[20] WIDE Project, http://www.wide.ad.jp/
[21] Argus web page, http://www.qosient.com/argus/
[22] PRODUCT INTRODUCTION — MD1230A Familiy, http://www.anritsu.co.jp/Products/pdf_e/MD1230Afamily_EI1100.pdf
[23] Round Robin Database Tool (RRDTool), http://people.ee.ethz.ch/ oetiker/webtools/rrdtool/
[24] Gnuplot Central, http://www.ucc.ie/gnuplot/
[25] M. Zangrilli and B.B. Lowekamp, "Comparing passive network monitoring of grid application traffic with active probes," Fourth International Workshop on Grid Computing, pp.84–91, Phoenix, AZ, Nov. 2003.
[26] D. Clark, "Window and acknowledgment strategy in TCP," IETF Request for Comments 813, July 1982.
[27] A. Zeitoun, Z. Wang, and S. Jamin, "RTTometer: Measuring path minimum RTT with confidence," IEEE Workshop on IP Operations and Management (IPOM 2003), Kansas City, MO, Oct. 2003.

Yoshinori Kitatsuji received his B.E. and M.E degrees in 1995 and 1997, respectively, from the department of science engineering, Osaka University, Japan. In 1972, he joined the KDD Co., Ltd. He was a research fellow of Telecommunications Advancement Organization of Japan from 2001 to 2003. He is presently an expert researcher in National Institute of Information and Communications Technology.

Katsuyuki Yamazaki received B.E. and D.E degrees from the University of Electro-communications and Kyushu Institute of Technology in '80 and '01, respectively. At KDD Co., Ltd., he had been engaged in development of ISDN and S.S. No.7, R&D and international standards of ATM networks, consultation for a new telecommunication company, and R&D of Internet QoS and networking. He is currently at KDDI R&D Labs. Inc., and responsible for R&D strategy.

Masato Tsuru received B.E. and M.E. degrees from Kyoto University, Japan in 1983 and 1985, respectively, and then received his D.E. degree from Kyushu Institute of Technology, Japan in 2002. He worked at Oki Electric Industry Co., Ltd. (1985–1990), Information Science Center, Nagasaki University (1990–2000), and Japan Telecom Information Service Co., Ltd./Telecommunications Advancement Organization of Japan (2000–2003). Since April 2003, he has been an Associate Professor in the Department of Computer Science and Electronics, Kyushu Institute of Technology. His research interests include performance measurement, modeling and analysis of computer communication networks. He is a member of the IPSJ and JSSST.

Yuji Oie received B.E., M.E. and D.E. degrees from Kyoto University, Kyoto, Japan in 1978, 1980 and 1987, respectively. From 1980 to 1983, he worked at Nippon Denso Company Ltd., Kariya. From 1983 to 1990, he was with the Department of Electrical Engineering, Sasebo College of Technology, Sasebo. From 1990 to 1995, he was an Associate Professor in the Department of Computer Science and Electronics, Faculty of Computer Science and Systems Engineering, Kyushu Institute of Technology, Iizuka. From 1995 to 1997, he was a Professor in the Information Technology Center, Nara Institute of Science and Technology. Since April 1997, he has been a Professor in the Department of Computer Science and Electronics, Faculty of Computer Science and Systems Engineering, Kyushu Institute of Technology. His research intrests include performance evaluation of computer communication networks, high speed networks, and queuing systems. He is a member of the IEEE and IPSJ.