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

Passive Packet Loss Measurement Employing the IP Packet Feature Extraction Technique

Satoru OHTA[†a)] *and* **Toshiaki MIYAZAKI**[†], *Members*

SUMMARY Performance measurements are indispensable for managing the Internet. Among the performance measurement techniques known, passive measurement is attractive because of its accuracy; user traffic is observed without inserting additional test traffic. However, the technique is handicapped by its large storage and bandwidth costs. This paper proposes a passive packet loss measurement technique that effectively avoids the difficulty of the conventional passive measurement approaches. Its key advance is utilizing the packet feature computed by a hash function. Since the feature can identify a packet with a short length of data, it becomes possible to greatly decrease the storage and bandwidth costs of passive measurements. The paper details the measurement procedure and assesses the design parameters used in the method. In addition, the validity of the proposed method is confirmed through experiments. The experiments also show the advantage of the method over the conventional active measurement.

key words: *Internet, performance, passive measurement, packet loss*

1. Introduction

The measurement of network performance is critical to properly manage the Internet. This need is strengthening due to the emergence of services that require wide bandwidth and low degradation. To provide such services with a satisfactory level of quality, the network must be checked by an appropriate performance measurement method. Accordingly, a considerable number of studies have examined measurement metrics [1]–[4], measurement methods [5]–[8] and described measurement results [9], [10].

Performance measurement methods are based on either active measurement or passive measurement [8]. Active measurement evaluates performance by injecting test traffic into the network. Passive measurement estimates the network state by observing the user traffic without inserting any additional traffic into the network. Passive measurement has the advantage of not perturbing the network by test traffic and the performance actually experienced by the users is measured. Current passive measurement techniques, however, suffer from unduly large storage and bandwidth costs. Because of this difficulty, passive measurement is seldom used for measuring metrics such as delay or packet loss.

This paper focuses on estimating packet loss rates by using two-point passive measurements. Packet loss is an important performance metric particularly for applications that use the TCP protocol, since their throughput is greatly degraded by packet loss [11]. The paper proposes a new method that minimizes the excessive storage and bandwidth costs. Namely, the proposed method records the feature of each packet instead of the packet itself or a portion of the packet. The feature, which is computed by a hash function, is small in length but sufficiently informative to identify the packet. Because of this characteristic, the proposed method effectively reduces the required storage and bandwidth costs while providing accurate estimations. Through the experiments on the prototype, the effectiveness of the proposed method was confirmed.

The paper first overviews related studies in Sect. 2. Section 3 introduces the basic idea of the proposed method. Section 4 details design issues, and Sect. 5 gives the results of experiments on the prototype.

2. Related Work

Passive measurements are often used to evaluate traffic load or protocol profiles while active measurements are used to evaluate performance metrics such as packet loss and delay. For example, Ref. [9], which reports various behaviors of the Internet, uses passive measurements for estimating the traffic matrix, packet size distribution, and protocol distribution. Meanwhile, they use active measurements for evaluating round trip time and packet loss. In addition, most commercial and freely available performance measurement tools employ various active measurement methods [5]. Nevertheless, passive measurement can estimate performance metrics as well. For example, Ref. [10] utilizes passive measurements to evaluate the round trip time and the out-of-sequence rate. Such evaluations are achieved by recording packets at the sender/receiver nodes and comparing the packet records. This approach is referred to as two-point passive measurement in the following.

Passive measurements work without inserting additional test traffic into the network. Active measurement injects test traffic into the network which may cause congestion, so the network state may become quite different from that experienced by users. Thus, active measurement may estimate performance inaccurately. In addition, the test traffic of active measurement differs from the traffic of the user applications. This difference in traffic characteristic also leads to inaccurate performance evaluations. By contrast, passive measurement can estimate the performance actually experienced by the user.

A shortcoming of passive measurement is the storage volume required to record the packet information [14]. Two-

Manuscript received March 30, 2004.
Manuscript revised June 23, 2004.

[†]The authors are with the NTT Network Innovation Laboratories, NTT Corporation, Yokosuka-shi, 239–0847 Japan.
a) E-mail: ohta.satoru@lab.ntt.co.jp

point passive measurement has another disadvantage: excessively large bandwidth is needed to gather the packet records stored at the sender and receiver for comparison. In the passive measurement system reported in Ref. [10], 330 GB disk arrays are installed at monitoring points; a 12 TB tape library and a 10 TB disk array are used to gather and analyze packet data. It is difficult to widely introduce such systems because of the high capital expenditures needed. The costs of storage and bandwidth will grow with the traffic rate and longer measurement times. This makes it difficult to widely deploy passive measurements for performance estimation.

In [8], Aida et al. proposed a measurement method that combines active and passive measurements. The resulting method effectively lowers the storage and bandwidth requirements for passive measurement. Their method, however, still injects test traffic into the network. This may disturb the network state if many samples must be collected to achieve sufficient accuracy. Additionally, the performance metrics inferred through calculation on the active and passive measurement results do not strictly identify the performance that the user experiences. Namely, the measurement result may be inaccurate. Considering these points, it is concluded that a better alternative method must be established though the method of Ref. [8] offers some improvement over the conventional methods.

The recording of IP packet routes is another subject that is being comprehensively studied. Reference [12] proposes a concept called "trajectory sampling," in which the routes of randomly selected packets are recorded for traffic management. Another example of packet route recording is seen in the "IP traceback" technique [13]. The main purpose of the IP traceback technique is to defend hosts from DoS (Denial of Service) attacks by identifying the sources and routes of hostile packets. Trajectory sampling [12] and IP traceback [13] use a common powerful technique that relies on a hash function. Namely, these techniques utilize the hash function to identify a packet. This hash-based packet identification is very useful and effective because it recognizes the same packet from among many packets using a very limited amount of information.

Hash-based packet identification is considered to have application areas other than trajectory sampling and IP traceback. As such an application, this study examines the technique for passive measurement. Owing to its characteristic of representing packets with a small amount of data, the technique effectively solves the storage and bandwidth consumption problem of conventional passive measurement methods.

3. Basic Concept

This study focuses on two point passive measurements. Namely, the transmitted packets are recorded at the sender host while the received packets are recorded at the receiver host. The packet records captured at these hosts during a specified period are gathered and compared. The results of this comparison allow the performance parameters to be estimated. This procedure is illustrated in Fig. 1. With the two point scheme, the following performance parameters can be estimated.

- one-way packet loss [3]
- delay variation
- one-way delay [2]

Packet loss is estimated by checking whether each packet recorded at the sender is also recorded at the receiver or not. If packet recorded at the sender is not recorded at the receiver, the packet is judged to have been lost. Thus, packet loss can be counted. The one-way delay is measured by recording packets and adding a synchronized timestamp and comparing the timestamp of the sender packet record with that of the receiver packet record. The delay variation is also evaluated from timestamp data.

In the following, this study addresses one-way packet loss measurements. As stated above, the packet loss is detected by checking whether the same packet exists in the sender and receiver records. As mentioned in Sect. 2, very large storage and bandwidth costs are incurred by conventional techniques.

This study employs the hash technique to overcome the above difficulty. With the proposed method, a portion of data bytes, $b(p)$, are extracted from the transmitted/received packet, p, at the sender/receiver, and the extracted portion feed to a hash function, $h(x)$, as the input. Hash function $h(x)$ maps the byte sequence $b(p)$ into an integer in $[0, A-1]$, where A is a positive integer. The byte length of the output is $\lceil(\log_2 A)/8\rceil$. Thus, by selecting an appropriate value of A, the hash output can have much smaller byte length than the whole packet or the extracted portion. Meanwhile, the hash output still includes sufficient information to identify the packet. Therefore, the hash output values can be recorded at the sender and receiver and compared for loss detection. Due to this hash output representation, packet

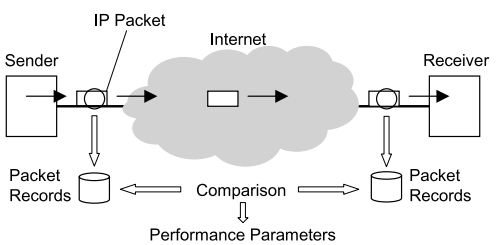

Fig. 1 Block diagram of two point passive measurement.

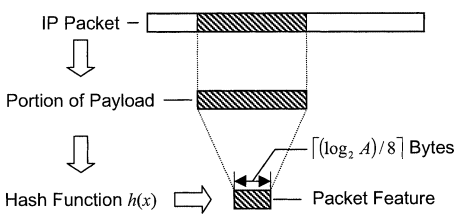

Fig. 2 The hash-based feature that identifies an IP packet.

loss can be estimated with a fairly small storage and bandwidth requirements. The hash output value is called the "feature" of the packet because the small byte length value identifies the packet. The procedure of computing the packet feature from an IP packet is illustrated in Fig. 2.

4. Proposed Method

As mentioned above, the proposed method of measuring packet loss consists of the following two phases.

Phase 1) The hash-based features of the transmitted and received packets are recorded at the sender and receiver.

Phase 2) The two data sets of packet features recorded at the sender/receiver are gathered at one place and compared to count packet loss.

In these phases, it is necessary to assess the sequence of recording packet features, the loss counting procedures, and the determination of the parameters associated with the packet feature computation.

4.1 Packet Loss Estimation Sequence

Let S be the set of packets whose features are recorded at the sender. Similarly, let R denote the set of the packets whose features are recorded at the receiver. Additionally, let Q be the set of the packets lost between the sender and the receiver. Then, the next relation must be hold for accurate packet loss measurement.

$$S \subseteq Q \cup R. \tag{1}$$

If the above relation does not hold, some packets in S are correctly delivered to the receiver but not recorded at the receiver. Such packets are counted as lost though they are not lost. Thus, Eq. (1) must be hold to avoid overestimating packet loss.

For Eq. (1) to be hold, the sequence of starting or stopping the packet feature recording should be determined carefully. This is achieved in the following way.

1. Start recording the features of the packets arriving at the receiver.
2. After the start of recording at the receiver is confirmed, start recording the features of the packets that leave the sender.
3. After a specified measurement time has passed, stop recording the packet features at the sender.
4. If a specified time T has passed after stopping recording at the sender, stop recording at the receiver.
5. After recording stops at the receiver, gather the recorded data at one place and count the packet loss.

With the above sequence, the receiver starts capturing features before the sender transmits a packet in S. Thus, no packets of S can arrive the receiver before the receiver begins to record packet features. Additionally, if time T is set larger than the delay from the sender to the receiver, no packets in S arrive at the receiver after the end of packet feature recording. Thus, Eq. (1) is exactly satisfied. Assuming that the receiver is controlled by the sender, the above sequence is illustrated by Fig. 3.

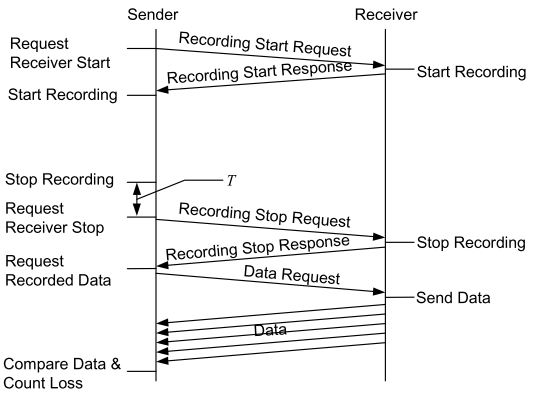

Fig. 3 Sequence of sender host controlled packet feature recording and loss calculation.

4.2 Packet Loss Counting Procedure

Executing the recording sequence described in Sect. 4.1 yields two lists of packet features. One list, L_S, includes the data captured at the sender while the other list, L_R, includes the data captured at the receiver. In addition, assume that we have a hash table that takes a packet feature as a key and stores an integer as the entry associated with each key. Let c_x denote the hash table entry for key x. Let N be the number of the lost packets. The following procedure successfully counts the packet loss.

Procedure *losscount*
1. $N := 0$;
2. **for each** element x in list L_R **do**
3. **if** c_x exists in the hash table
   then $c_x := c_x + 1$
   else put x in the table and $c_x := 1$;
4. **for each** element y in list L_S **do**
5. **if** c_y exists in the hash table and $c_y > 0$
   then $c_y := c_y - 1$
   else $N := N + 1$;
6. **return** N;

With utilizing the hash table data structure, the procedure *losscount* can be executed efficiently in terms of computational time. With the hash table, the search, creation and value update of an element for a given key are executed in $O(1)$ time. Thus, steps 3 and 5 are executed in $O(1)$ time. As a result, it is obvious that *losscount* is executed in $O(|S|+|R|)$ time. This implies that the procedure can handle large sets of packet features with practical computational times.

If no two recorded packets have the same feature, procedure *losscount* returns the correct packet loss. In this situation, if a packet in S is transmitted and not received, table

entry c_x is not created in step 3 for the feature x of the packet. Thus, N is correctly incremented for the packet in step 5. On the other hand, if a packet in S is received at the receiver, c_x is set at 1 in step 3. Since c_x exists and is equals to 1, step 5 does not increment N. Thus, value N finally equals the number of lost packets.

The above procedure gives correct packet loss values in cases that some duplicated feature values exist in L_S and L_R. To see this, let us define packet sets R^* as follows.

$$R^* = \{p | p \in R \text{ and } p \notin S\} \quad (2)$$

For this definition, if there is no duplication between the packets features from R^* and the elements of L_S, *losscount* counts packet loss correctly independently of duplication within L_S. To see this, consider that q packets in S have the same feature value x_0 and no elements of R^* have feature value x_0. For this situation, suppose that $r(r \leq q)$ of these q packets are lost. Namely, $q - r$ packets that have feature value x_0 will arrive at the receiver as elements of R. Then, step 3 in procedure *losscount* is executed $q - r$ times for $x = x_0$. This computation sets value c_{x_0} to $q - r$ if $r < q$. Clearly, table element x_0 is not created if $q = r$. Next, step 5 is executed q times for $y = x_0$. If $r < q$, the "then" clause of step 5 is executed for $y = x_0$ until c_{x_0} becomes 0. Since the initial value of c_{x_0} is $q - r$, the "then" clause is repeated $q - r$ times. After c_{x_0} reaches 0, the "else" clause is executed r times for $y = x_0$. This increments N by the number of lost packets, r. For the case of $q = r$, the "then" clause is not executed, and the "else" clause is repeated r times. Therefore, the number of lost packets is exactly counted for $q = r$ as well as $q > r$.

If the feature of an element in R^* equals that of some element in L_S, the counted packet loss may not be correct. Suppose that q packets in S and q^* packet in R^* have the same feature value x_1. In the following, such feature coincidence is called "collision." For these packets, suppose that r packets ($r \leq q$) in S are lost. Then, collision will occur among $q + q^* - r$ packets in R assuming that q^* packets in R^* are not lost. Thus, step 3 of *losscount* is executed $q + q^* - r$ times for $x = x_1$, which sets c_{x_1} to $q + q^* - r$. Then, step 5 is repeated q times for $y = x_1$. For the case of $q^* < r$, the "else" clause of step 5 is executed $r - q^*$ times and thus N is incremented by $r - q^*$, which is smaller than the actual number of lost packets, r. For the case of $q^* \geq r$, the "else" clause of step 5 is not executed at all because c_{x_1} does not reach 0 after q repetition of the "then" clause. Thus, N is not incremented in spite of packet loss for $q^* \geq r$. For both cases, therefore, the loss of r packets in S is not counted correctly.

To avoid the above counting error, the probability of collision must be minimized. Let P_c denote the collision probability between the features of two distinct packets. Then, for a small value of P_c, the feature of a packet in S collides with that of an element of R^* with probability $P_c|R^*|$. In the sequence shown in Fig. 3, let t_{ss} and t_{sr} be the recording start times at the sender and receiver, respectively. Assume that the maximum delay of a packet is D. Then, packets transmitted in time period $[t_{sr} - D, t_{ss})$ may be recorded at the receiver as elements in R^*. Similarly, let t_{es} and t_{er} be the recording stop time at the sender and receiver, respectively. Let the minimum delay of a packet be 0. Then, packets transmitted in time period $(t_{es}, t_{er}]$ may be recorded at the receiver as elements in R^*. It is not difficult to see that a packet transmitted in the periods other than $[t_{sr} - D, t_{ss})$ and $(t_{es}, t_{er}]$ can not be an element of R^*. Thus, $|R^*|$ is bounded by the product of the length of these periods and the maximum packet rate.

From the delay characteristic observed in the current network [9], it is possible to estimate that the length sum of $[t_{sr} - D, t_{ss})$ and $(t_{es}, t_{er}]$ is less than a few seconds. As a realistic numerical example, assume that the length sum of the periods is 2 seconds and the maximum packet rate is 5×10^4 (packets/s). Then, the upper bound of $|R^*|$ is 10^5. For this example, if the collision probability of two packet features is less than 10^{-7}, the probability that an element of L_S collides with a packet feature from R^* is upper bounded by 10^{-2}. This result implies that collision probabilities less than 10^{-7} offer an acceptably low relative error in measured loss and so permit realistic operation.

4.3 Packet Feature

The next parameters must be determined to obtain an effective hash-based packet feature.

- The packet portion to be extracted as the input of the hash function.
- The bit length of the hash function output.

These decisions should be made from several viewpoints. First, the hash output computed from a packet should not coincide with that computed from another packet for accurate packet loss measurement. Second, the bit length of the hash function output should be as small as possible to reduce the storage and bandwidth costs. Finally, the extracted part of a packet should not be changed during transfer across the network. Since the hash value computed at the sender must be equal to that computed at the receiver for the same packet, the hash input should not include the fields modified in the network such as TTL or checksum.

Which part of a packet is optimal in terms of extraction was evaluated in past studies that examined hash-based packet identification [12], [13]. Reference [12] reports that collisions occur with high probability if only the first 20 bytes of a packet is given to the hash function but the collision probability decreases quickly as the extracted byte length is increased. They state byte lengths exceeding 40 bytes do not improve the performance much. Similarly, in [13], it is reported that the first 28 invariant bytes are sufficient to distinguish almost all different packets.

However, these results are not necessarily applicable to this study. This is because the past studies aimed at identifying particular packets among all packets in the network while this study aims at identifying packets in a particular traffic flow. The traffic flow measured by the proposed

method may well be packets of a particular protocol or application used between a specified pair of hosts. Because of this difference, some fields found to be informative in the past studies are not informative in this study. For example, the source and destination addresses of the IP header are included in the methods of [12], [13]. However, the addresses will be the same for all packets in our measurements. For the measurement of a particular protocol flow, fields in the IP header are not very useful for packet identification.

Given the above argument, the proposed measurement method uses a part of the payload of the IP packet as the input for the hash function. Throughout this paper, the payload is defined as the data left after removing the IP header from the packet. Thus, the payload includes the TCP (or UDP) header if the higher layer protocol is TCP (or UDP). An experiment was carried out to see which part should be extracted. The experiment employed two PC's that ran applications based on the TCP/UDP protocols (http, ftp, rtp, nfs), and the payload portions of the 4×10^6 transmitted packets were captured. Sets of byte sequences were then made by extracting m bytes from the n-th byte in each captured payload. For each of these byte sequence sets, the probability of collision between two byte sequences was evaluated. The parameters m and n, which specify the extracted part, must be selected to minimize this probability. Additionally, it should be noted that a large value of m will increase the computational time of the hash function.

Figure 4 shows the results of this experiment. The figure plots the collision probability between two byte sequences derived from different packets against the byte length m for the cases of $n = 1, 3, 5, 7$. The characteristics for $n = 2, 4, 6$ were basically similar to those for $n = 1, 3, 5$ and are thus omitted from the graph. The figure shows that increasing the byte length greatly decreases the collision probability. However, byte lengths larger than 20 bytes do not improve the collision probability significantly. The collision probability depends on the start position n of the extracted part as well. Among the examined values, $n = 5$ showed better performance than the others. Namely, the collision probability decreases more steeply against the increase of m when $n = 5$ than with any other value of n. For $n = 1$, the collision probability does not reduce greatly for $1 < n \leq 4$ but decreases quickly for $n > 4$. This result suggests that the first four bytes of the payload are not informative. This is because the first four bytes represent the source and destination port numbers in TCP and UDP packets. Since the port numbers are fixed for the traffic flow measured, the first four bytes are not effective in identifying the packets. By contrast, the eight bytes from the 5th byte are very informative. With TCP, these bytes include the sequence number and the acknowledge number, which differ among the packets in a particular traffic flow. With UDP/RTP, the portion includes the sequence number and the timestamp [15], which differ greatly among packets. Therefore, the collision probability is effectively decreased by eliminating the first four bytes but including the next eight bytes in the input of the hash function. From the

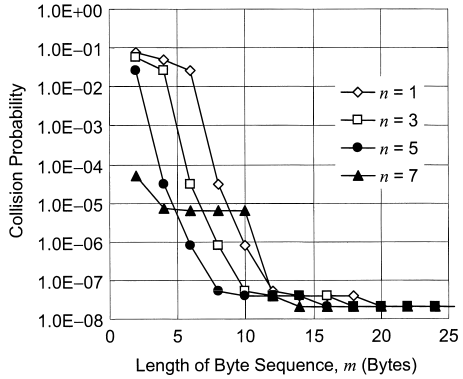

Fig. 4 Collision probability versus the length m of bytes extracted from the payload: the extraction start point, n, is varied.

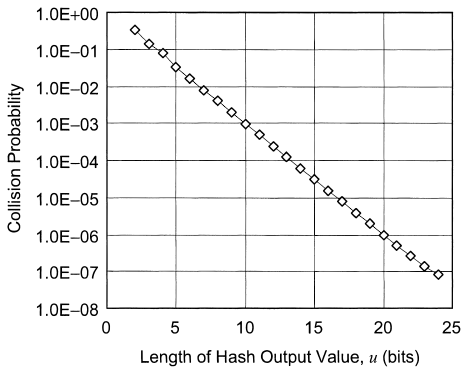

Fig. 5 Collision probability versus the bit length of hash function output.

above observation and Fig. 4, it is seen that the parameter setting of $n = 5$ and $m = 16$ is a good choice because it makes the collision probability sufficiently low ($p_c < 10^{-7}$) and the byte length is quite short.

Next, it is necessary to determine how many bits are necessary for the hash function output. To examine this point, the collision probability of the hash function output was evaluated for the above data set, which includes byte sequences of 4×10^6 packets. In the evaluation, the hash function input was the byte sequence created by $m = 16, n = 5$. The employed hash function $h(x)$ is as follows:

$$h(x) = x \bmod A. \tag{3}$$

where x is an integer represented by the byte sequence and A is a positive integer. The value of integer A was selected as a prime number that is close to but less than $2^u, u = 2, 3, \ldots, 24$. Obviously, this setting results in the output bit length being u.

The results of this evaluation are shown in Fig. 5. In the figure, the x-axis shows the bit length while the y-axis shows the collision probability of the hash values of two packets. The figure shows the collision probability decreases monotonously against the bit length of the hash function output. The figure also shows that the output length of 24 bits will give sufficiently low collision probability. Namely, a 3 Byte packet feature can identify a packet almost exactly. Thus, the space needed to store the feature is much

smaller than the extracted payload (16 Bytes) or the packet itself.

5. Evaluation

A prototype of the proposed method was implemented as a set of Linux software programs. One program employs `pcap` library [19] to record packet features. Another program implements the *losscount* procedure and so estimates packet loss from the packet feature records. These programs follow the sequence shown in Fig. 3 through an `expect` [20] script. The purpose of implementing the prototype was to confirm the effectiveness of the method.

Figure 6 shows the network configuration used in the experiments. The packet loss was measured by the proposed method and by two conventional active measurement methods for the traffic transmitted from the source host to the destination host. The transit host ran network emulation software (`NIST Net` [16]) to set the packet loss ratio and the bandwidth to the specified values.

With using the prototype, two scenarios were examined. The first scenario, referred to as scenario 1, clarified that the proposed method was accurate. In this scenario, the transit host set the packet loss to values from 0.002% and 10% while the bandwidth was set to a sufficiently large value, 400 Mb/s. Packet loss was set for one direction from the source host to the destination host. For this setting, the packet loss was measured by the proposed method as well as two conventional active measurement techniques, `ping` and the UDP measurement by `Iperf` [18]. Both measured values were compared to the values set by the emulation software. The measured user traffic was emulated by the Poisson arrival UDP packet flow created by `mgen` [17]. The average bit rate of the measured traffic was 100 Mb/s; packet size was 1250 Bytes. To prevent the active measurements from disturbing the network condition, the volume of the `Iperf` traffic was set at 1 Mb/s and the interval of `ping` was 1 second. The measurement time was 300 seconds. The measurements were repeated 5 times and the average of the measured values was obtained. In this setting, it is expected that the packet loss will exactly follow the loss value set by the emulation software because the traffic volume is considerably lower than the link capacity.

The second scenario, referred to as scenario 2, showed that the proposed method yielded accurate packet loss measurements where the active methods could not because of the difference between the test traffic and the measured user traffic. To emulate this situation, a bursty packet flow was used as the measured user traffic. The packet flow is a repeated sequence of a busy period and an idle period. The bit rate during the busy period was a varied parameter. If the bit rate of the busy period is higher than the available bandwidth, packets will be lost. No packets are generated during an idle period. The test traffic packets created by the active methods are also lost during the busy periods. However, since the test traffic packets will not be lost during the idle period, the active methods may underestimate the packet loss. To clarify this point, the second scenario compared the packet loss values measured by the proposed method with those output by the active measurement methods.

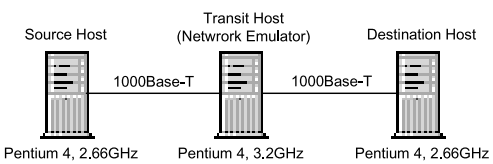

Fig. 6 Experimental network configuration.

For the user traffic used in the second scenario, the length of the busy period was set at 0.01 seconds while the interval between the busy periods was exponentially distributed with the average of 0.1 seconds. The bandwidth between the source and the destination was set at 40 Mb/s by the emulation software on the transit host. The `Iperf` traffic was set at 1 Mb/s so as to be sufficiently low for the bandwidth. The `ping` interval was 1 second. The measurement time was 300 seconds. The measurement was repeated 5 times and the average of measured values was obtained.

Figure 7 shows the results of scenario 1. In the figure, the x-axis shows the packet loss rate set on the network emulation software (denoted as P_{set}) while the y-axis shows the measured packet loss rate (denoted as P_{meas}). For `Iperf` and `ping`, the 95% confidence intervals are also plotted; the confidence interval for the proposed method was too small to be plotted. The figure also shows the line $y = x$, which represent exact measurement. As seen in the figure, the proposed method follows the ideal value closely. Namely, the proposed method estimates the packet loss accurately.

As seen in Fig. 7, the values output by `Iperf` and `ping` have larger variance than those of the proposed method except for large P_{set}. Namely, the values by these active methods include greater measurement errors. In addition, `ping` failed to detect packet loss for $P_{set} \leq 5 \times 10^{-4}$. These results were brought about because the number of the measured packets was too small to evaluate the loss rate accurately. Since `ping` generates fewer test packets than `Iperf`, the loss measurements by `ping` are more inaccurate. Need-

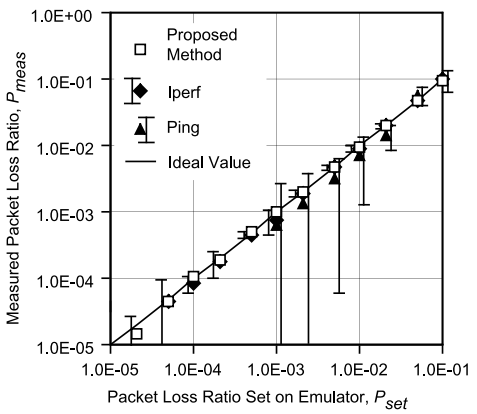

Fig. 7 Measured packet loss ratio versus the packet loss ratio specified in the network emulator.

less to say, the active measurement methods can easily create a sufficient number of packet loss events simply by increasing test traffic volume. However, heavy test traffic may congest the network and will obviously change the performance. Thus, it is difficult to employ such test methods in an operating network. In conclusion, Fig. 7 shows the limitation of the active measurement methods as well as the accuracy of the proposed method.

In scenario 1, the features of about 3×10^6 packets were recorded over the 300 second period. The feature records occupied about 9 MB. This size is easily handled by current storage systems, and thus the proposed method can be easily applied to higher packet rates or longer measurement times.

Another advantage of the proposed method is illustrated in Fig. 8, which shows the results of scenario 2. In Fig. 8, the measured packet loss ratio is plotted against the peak rate of the measured traffic. The figure shows that the measured values differ greatly among the proposed method and the active measurement methods. Namely, the values output by `Iperf` and `ping` are much smaller than that indicated by the proposed method. This is explained as follows.

In scenario 2, the packets of user traffic are generated in busy periods and the rate exceeds the bandwidth. Thus, packets are likely to be lost during the busy periods due to network congestion; packets are not likely to be lost during the idle periods. This means that the packet loss occurs periodically following the periodicity of the traffic bursts. Since the active measurement methods generate test packets at a constant rate, only some of the test packets are dropped due to congestion; the measurement packets issued during the idle periods are not likely to be lost. By contrast, all user packets are issued during the congestion periods and thus are lost with high probability. Thus, the packet loss measured by the active measurement methods is much smaller than that seen in the user traffic. In particular, since `ping` issues fewer test packets than `Iperf`, a `ping` packet experiences congestion with smaller probability. Thus, the packet loss measured by `ping` is smaller than that measured by `Iperf`.

From the above observation, the active measurement methods that issue constant rate test traffic do not accurately estimate the packet loss of bursty traffic. Thus, it is concluded that the proposed passive measurement method is essential to obtain exact packet loss values for such traffic.

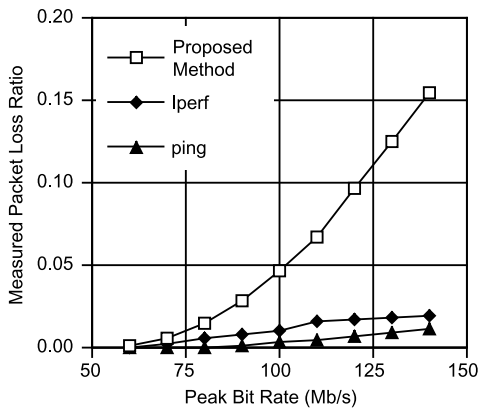

Fig. 8 Packet loss ratios measured by the proposed method and active measurement methods for scenario 2.

6. Conclusion

This paper presented a passive measurement method that can accurately estimate packet loss in the Internet. Using a hash function to compute packet features effectively eliminates the problems experienced by conventional passive measurement methods. We detailed the control sequence determining packet feature recording and loss counting as well as the parameters used for feature extraction. The proposed method was implemented as software programs and its accuracy was confirmed by experiments. The experiments also clarified its advantages over the current active methods. Namely, the proposed method can measure packet loss accurately even if the user traffic is bursty unlike the active methods. Since the proposed method is inexpensive to implement, it will be easy to deploy in the Internet.

Acknowledgment

The author would like to thank Professor Kimio Oguchi of Seikei University and the members of the High-Speed Protocol Processing Research Group of NTT for their encouragement.

References

[1] V. Paxon, G. Almes, J. Mahdavi, and M. Mathis, "Framework for IP performance metrics," IETF RFC2330, May 1998.
[2] G. Almes, S. Kalidindi, and M. Zekauskas, "A one-way delay metric for IPPM," IETF RFC2679, Sept. 1999.
[3] G. Almes, S. Kalidindi, and M. Zekauskas, "A one-way packet loss metric for IPPM," IETF RFC2680, Sept. 1999.
[4] G. Almes, S. Kalidindi, and M. Zekauskas, "A round-trip delay metric for IPPM," IETF RFC2681, Sept. 1999.
[5] CAIDA, "Internet tools taxonomy," http://www.caida.org/tools/taxonomy/
[6] T. Hansen, J. Otero, T. McGregor, and H.-W. Braun, "Active measurement data analysis technique," Proc. International Conference on Communications in Computing (CIC 2000), paper CIC-66, Las Vegas, June 2000.
[7] S. Savage, "Sting: A TCP-based network measurement tool," Proc. 1999 USENIX Symposium on Internet Technologies and Systems, pp.71–79, Boulder Col., Oct. 1999.
[8] M.Aida, K. Ishibashi, and T. Kanazawa, "CoMPACT-Monitor: Change-of-measure based passive/active monitoring — Weighted active sampling scheme to infer QoS," Proc. IEEE SAINT 2002 (Workshop on Measurement Technology for the Internet Applications), pp.119–125, Nara, 2002.
[9] k. claffy, "Internet measurement and data analysis: Topology, workload, performance and routing statistics," Proc. NAE'99 Workshop, Los Angels Ca, March 1999. available at http://www.caida.org/outreach/papers/1999/NAE/
[10] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T Seely, and C. Diot, "Packet-level traffic measurements from the Sprint IP backbone," IEEE Netw., vol.17, no.6, pp.6–17, Nov. 2003.

[11] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The macroscopic behavior of the TCP congestion avoidance algorithm," ACM SIGCOMM Computer Communication Review, vol.27, no.3, pp.67–82, July 1997.
[12] N.G. Duffield and M. Grossglauser, "Trajectory sampling for direct traffic observation," IEEE/ACM Trans. Netw., vol.9, no.3, pp.280–292, June 2001.
[13] A.C. Snoeren, C. Partridge, L.A. Sanchez, C.E. Jones, F. Tchakountio, S.T. Kent, and W.T. Strayer, "Hash-based IP traceback," Proc. SIGCOMM'01, pp.3–14, San Diego, Aug. 2001.
[14] N. Brownlee, k. claffy, M. Murray, and E. Nemeth, "Methodology for passive analysis of a university internet link," Proc. PAM 2001 Workshop, April 2001. available at http://www.caida.org/outreach/papers/2001/MethodAnalyseLink/.
[15] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A transport protocol for real-time applications," IETF RFC1889, Jan. 1996.
[16] NIST Net Home Page, http://dns.antd.nist.gov/nistnet/
[17] MGEN The Multi-Generator Toolset, http://manimac.itd.nrl.navy.mil/MGEN/
[18] Iperf, http://dast.nlanr.net/Projects/Iperf/
[19] LBNL's Network Research Group, http://ee.lbl.gov/
[20] D. Libes, Exploring Expect, O'Reilly & Associates, Sebastapol CA, 1995.

Satoru Ohta received the B.E., M.E., and Dr. Eng. degrees from the Tokyo Institute of Technology, Tokyo, Japan, in 1981, 1983, and 1996, respectively. Since 1983 he has been with NTT, where he worked on the research and development of cross-connect systems, broadband ISDN, network management, and telecommunication network planning. He is currently involved in research on network protocols and network performance evaluation at NTT Network Innovation Laboratories, NTT Corporation. Dr. Ohta is member of the IEEE. He received the Excellent Paper Award in 1991 from IEICE.

Toshiaki Miyazaki is a senior research engineer, supervisor of NTT network innovation laboratories. He received the B.E. and M.E. degrees in applied electronic engineering from the University of Electro-Communications, Tokyo, Japan in 1981 and 1983, and the Dr. Eng. degree from the Tokyo Institute of Technology in 1994. Since joining NTT in 1983, he has been engaged in research on VLSI CAD systems including high-level synthesis, logic design tools, and CAD frameworks. Since 1993, he has been involved in developing telecommunications-oriented FPGAs and their applications, and next generation networking protocols. Since 2003, he has been engaged in research on peer-to-peer communications and ubiquitous network environments. His research interests are in programmable hardware systems, adaptive networking technologies, and autonomous systems. Dr. Miyazaki is a member of IEEE and ISPJ.