Performance Prediction Method for IP Lookup Algorithms

Ryo Kawabe †, Shingo Ata ‡, Masayuki Murata †,
Masanori Uga §, Kohei Shiomoto ¶, Naoaki Yamanaka ¶

†Graduate School of Engineering Science, Osaka University
1–3 Machikaneyama, Toyonaka, Osaka 560–8531, Japan
Phone: +81–6–6850–6616, Fax: +81–6–6850–6589
E-mail: {r-kawabe, murata}@ics.es.osaka-u.ac.jp

‡Graduate School of Engineering, Osaka City University
3–3–138 Sugimoto, Sumiyoshi-ku, Osaka 558–8585, Japan
Phone, Fax: +81–6–6605–2191
E-mail: ata@info.eng.osaka-cu.ac.jp

§NTT Network Service Systems Laboratories, NTT Corporation
3–9–11 Midori-Cho, Musashino-Shi, Tokyo 180–8585, Japan
Phone: +81–422–59–4402, Fax: +81–422–59–6387
E-mail: Uga.Masanori@lab.ntt.co.jp

¶NTT Network Innovation Laboratories, NTT Corporation
3–9–11 Midori-Cho, Musashino-Shi, Tokyo 180–8585, Japan
Phone: +81–422–59–4402, Fax: +81–422–59–6387
E-mail: {Shiomoto.Kohei, Yamanaka.Naoaki}@lab.ntt.co.jp

Abstract—**Many address lookup methods for use on IP routers to improve their packet-forwarding capability have been proposed. However, their performance prediction ability is poor because actual traffic characteristics are not considered in their evaluation processes. The actual traffic must be considered in order to predict router performance more accurately, especially for layer 3 and 4 address lookups, whose performances are more affected by the flow characteristics. In this paper, we describe a method for predicting IP lookup algorithm performance that is based on statistical analysis of the Internet traffic. We present an example of its application to an existing IP lookup algorithm and show, based on simulation results, that our method can provide accurate performance prediction for IP lookup algorithms.**

I. Introduction

The rapid growth in the Internet traffic with the spread of multimedia applications such as streaming media has led to an explosive growth in demand for high-speed packet transmission technologies. This has made it necessary to improve the packet forwarding capability of IP routers. A router determines which output interface to use to forward arriving packets based on the packet header information, e.g., the destination address. Other information in the header, such as the source address, source/destination port numbers, and protocol number, may be also used for policy routing and/or layer 4 switching.

IP routers perform two steps for each arriving packet.

1. Look up next-hop of packet from routing table and determine which output interface to use.
2. Forward packet to output interface determined in step 1.

Step 1 greatly affects router performance because the longest prefix matching [1] has become more complicated since classless inter-domain routing (CIDR) [2] was introduced. Address lookup has thus become a performance bottleneck in high-speed routers.

While many approaches have been proposed to overcome this bottleneck, e.g., [1], [3], [4], their performances have not been well studied. Two metrics have generally been used to rate the performance of address lookup algorithms: worst-case and average-case (or actual-case) performance. Worst-case performance is easily derived from the complexities of lookup algorithms, and by using it, a proposed algorithm can easily be compared with existing algorithms. Worst-case performance is also a useful index for describing an algorithm's basic capability. Most papers on IP lookups thus use worst-case performance.

However, actual performance is greatly affected by the sequence of the destination addresses of arriving packets. Furthermore, an algorithm designed to maximize worst-case performance can be very expensive. A closer look at the performance of the target IP lookup algorithm may produce a more elegant solution. For example, we may be able to obtain a much cheaper solution for a limited sacrifice of performance (e.g., introducing a small packet loss probability).

To achieve a more elegant solution, we need a realistic address generation method for use in evaluating the performance of IP routers. However, previous research considered only simulation techniques using random address generation [3], or only a small amount of trace data was used [4]. If only a small amount of trace data is used, the actual performance behavior is likely to be missed. Furthermore, there is a limited amount of trace data available in the public domain, so simulation results lack generality.

Narlikar and Zane recently described an analytical model that accurately estimates the average-case lookup time of an algorithm [5]. Though the average-case lookup time is a useful metric, we also need more detailed network performance, e.g., the behavior of the time-dependent queue length, the average packet loss ratio, and the average packet-processing delay, to find a more appropriate design solution.

In this paper, we describe a method for predicting router performance based on statistical analysis of the Internet traffic. We also present an example of its application to an IP lookup algorithm and show that, based on simulation results, our method can accurately predict performance.

II. Performance Prediction Methods

A. Overview

We will first give an overview of our proposed methods for predicting the performance of address lookup algorithms. As shown in Figure 1, our methods consist of three processes.

1. Pseudo Address Creation
2. Address Sequence Generation
3. Algorithm Performance Evaluation

The *pseudo address creation process* creates a destination address based on the traffic statistics. One way to predict performance is to use the random IP method, which generates a 32-bit random integer as the IPv4 address. However, monitoring of Internet traffic has shown that destination addresses are not uniformly distributed. Instead, they are strongly biased towards certain addresses (e.g., `WWW` servers). The prefix length of the entry generally indicates the size of the organization it belongs to. That is, the address space which includes the longer prefix en-

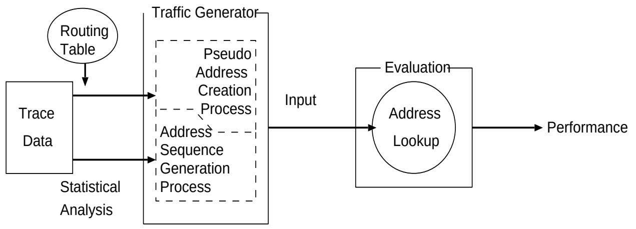


Fig. 1. Overview of Performance Prediction Method

tries tend to include more organizations. Because the most popular traffic in the current Internet is `http`, packets tend to access the destination address space which include more `WWW` servers. Therefore, the headers of the traced packets tend to match the longer prefix entries. Since the method of generating random IP addresses creates uniformly–distributed addresses; the number of accesses of the entry is inversely proportional to the prefix length. A new process is thus needed for creating addresses.

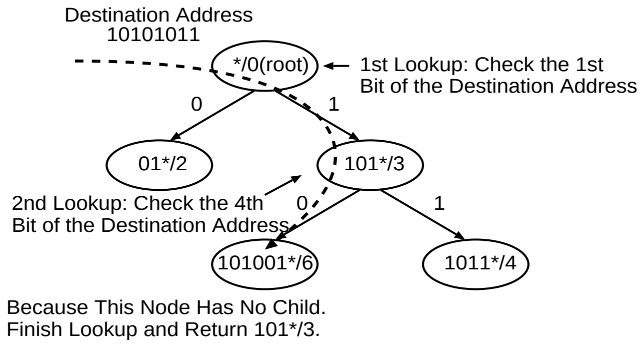
The second process is *address sequence generation*. Let us consider the sequence of packet arrivals in a TCP flow. Because the TCP flow is divided into packets, the probability that the next packet in the flow will arrive at the router tends to decrease as time passes. To model the tendency of packet arrivals, we use the approach to the address sequence generation described by Aida and Abe [6], in which address sequences are generated using a least recently used (LRU) stack and an inverse stack growth function (ISGF). To generate address sequences, we use the LRU stack model and ISGF. The first and second processes generate traffic.

The third process is *algorithm performance evaluation*, which is done through trace-driven simulation using traffic patterns generated by our proposed traffic generation method. We develop the implementation of address lookup algorithms on a trace-driven simulator and evaluate their performance using our traffic generation methods.

We discuss these three processes in more detail in the following subsections. After describing our target IP lookup algorithm in the following subsection, we describe the first two processes in our performance prediction method and an address sequence generation method using an LRU stack based on ISGF. The third process in our performance prediction method is described in Section III.

B. Target IP Lookup Algorithm

Because our pseudo address creation method uses algorithm–dependent parameters, we refer to the IP lookup algorithm to be evaluated by our method as the *target algorithm*. We use a Patricia Tree search as the target algorithm as an example application. A Patricia Tree algorithm is a simple, easily implemented, and well-known algorithm for IP lookup. Legacy routers use a binary tree for longest prefix matching. A Patricia Tree search [7] eliminates nodes having only one child node. In the example Patricia Tree search shown in Figure 2, the nodes for the fourth and fifth bits of the destination address are removed because the routing table has only entries beginning with 101001 when the fourth bit of the destination address is 0. Although a Patricia Tree search is simple, it is relatively slow because the number of removable nodes decreases as the number of entries increased. In the worst case, a Patricia Tree requires up to 32 or 128 memory references in IPv4 or IPv6, respectively. Note here that our performance prediction method is not limited to the Patricia Tree algorithm. It can also be applied to other lookup algorithms by introducing the lookup delay distribution described in the next subsection.



Destination Address
10101011
*/0(root)
1st Lookup: Check the 1st Bit of the Destination Address
0
1
01*/2
101*/3
2nd Lookup: Check the 4th Bit of the Destination Address
0
1
101001*/6
1011*/4
Because This Node Has No Child. Finish Lookup and Return 101*/3.

Fig. 2. Example Patricia Tree Search

C. Pseudo Address Creation Process

In this subsection, we describe our pseudo address creation process. We first describe the characteristics of address lookup delays and then describe our algorithm for creating addresses.

We define D_i as the depth of a Patricia Tree for the matched entry for packet i, and S_i as the address lookup delay for packet i. In a Patricia Tree search, the address lookup delay is given by

$$S_i = D_i \times d_r, \tag{1}$$

where d_r is the delay due to read / comparison operations in RAM. If a read requires 5 nsec and a comparison requires 10 nsec, the address lookup delay is 15 nsec. In the example shown in Figure 2, since two lookups are required to determine the longest prefix entry ($101*$) of the packet, D_i is 2 and S_i is 30 nsec.

To obtain the distribution of S_i, we used a routing table from BGP information (4,669 entries) at the gateway of Osaka University to construct the Patricia Tree. We next collected N_p packet headers by the traffic monitor (OC3MON [8]). Then, we put the N_p traced packet headers into the Patricia Tree to calculate the address lookup delay S_i for each packet i $(1 \le i \le N_p)$ and finally we obtain the distribution of S_i. Figure 3 shows the example of the distribution of S_i (refer to $F(S)$ throughout this paper) derived from 10,000,000 packets obtained on January 24, 2001.

Addresses are created according to $F(S)$ in our proposal methods. The pseudo address creation process works as follows.

1. Obtain the distribution of S_i ($F(S)$) from traced-data.
2. Choose a random number p $(0 \le p < 1)$.
3. Calculate the minimum S_i if $p \le \sum_{S=1}^{S_i} F(S)$.
4. Calculate D_i from S_i.
5. Output a new address as one of arbitrary addresses from matched entries whose depth in the Patricia Tree are D_i.

D. Address Sequence Generation Process

We use the least recently used (LRU) stack and the inverse stack growth function (ISGF) [6] to model the tendency of

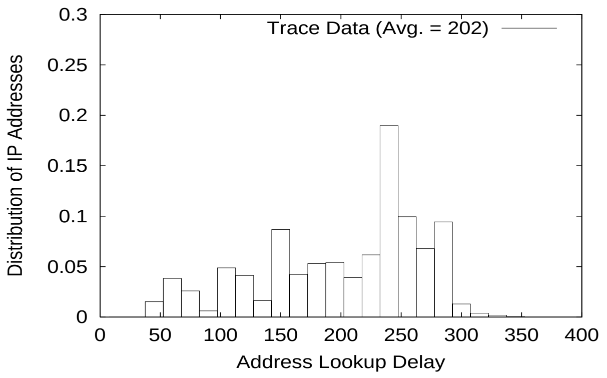


Fig. 3. Distribution of Address Lookup Delays

packet arrivals.

To briefly summarize ISGF, we introduce t_i, denoting the arrival time of the i-th packet (or flow). We define $f(t, T)$ as the expected number of distinct addresses of packets arriving during a period $(t - T, t)$. If we assume that $f(t, T)$ is independent of time t, i.e., $f(t_i, T) = f(t_j, T)$ for all i, j, $f(t, T)$ can be denoted by $f(T)$. This assumption is called a time–translation invariance, and makes it possible to obtain the same value for $f(T)$ whenever traced packet data are gathered. ISGF is a power law function given by

$$f(T) \simeq T^{\alpha} \ (T \gg 1), \tag{2}$$

where α $(0 < \alpha < 1)$ is a constant value.

While ISGF was originally used for predicting the cache hit ratio of computer architectures, the number of distinct destination addresses was found to also satisfy ISGF [6]. That is, if T packets are collected from traced data, the number of distinct destination addresses can be estimated as T^{α}. Thus, by using ISGF, it is possible to derive the probability that the destination address of an arriving packet has already appeared. However, ISGF alone is not enough to model the tendency of packet arrivals because it cannot hold a record of arrival packets. Thus, the LRU stack is applied to ISGF.

The LRU stack is used to store the record of packet arrivals. The distinct addresses arrived are stored in the LRU stack in descending order of arrival time, e.g., latest recently arrived address is stored at the top of the LRU stack. The probability that i-th entry of the LRU stack arrives again is set to a_i which depends only on the location of the LRU stack, e.g., an address of the top element of the LRU stack is arrived at the probability a_1.

By using ISGF, a_i is given by

$$\begin{aligned} a_i &= \{f(g(i-1)+1) - (i-1)\} \\ &\quad -\{f(g(i)+1) - i\}, \end{aligned} \tag{3}$$

where $g(T) = f^{-1}(T)$. From Eq. (2), we get

$$\begin{aligned} a_i &= \{((i-1)^{\frac{1}{\alpha}}+1)^{\alpha} - (i-1)\} \\ &\quad -\{(i^{\frac{1}{\alpha}}+1)^{\alpha} - i\}. \end{aligned} \tag{4}$$

When probability a_i is given, address sequence generation is modeled using the LRU stack model whose probability is given by Eq.(3).

In the LRU stack model, however, the tendency of an IP address to be accessed is determined only by the location of the address in the stack. For example, let us consider two IP addresses, one is located at bottom of the LRU stack, the latter is next to the former one. If the size of LRU stack is very large, the probabilities of each address to be accessed are almost same and very low. Therefore, the LRU stack does not need to have them since these two addresses remain nearly unaffected by ISGF. The extremely large elements of the LRU stack are not meaningful. Consequently, we have to set the LRU stack size to the appropriate size to reduce the processing overhead. We will discuss about the appropriate size of the LRU stack for the accurate performance prediction in Section III.

E. *Traffic Generation Method*

In this subsection, we explain traffic generation using an LRU stack based on ISGF. The following procedure generates time–dependent destination addresses using the LRU stack model with the above quantities: N_p, N_a, α and $F(S)$. We denote m as number of the elements of the LRU stack and M as the LRU stack size.

```
1. Set the number of elements in LRU stack, $m$ ($0 \le m \le M$), to 0.
2. Choose random number $p$ $(0 \le p < 1)$.
3. Calculate minimum $j$ if $p \le \sum_{i=1}^{j} a_i$.
4. If $j \le m$
  (a) Output $j$-th element of LRU stack.
  (b) Shift $k$-th $(1 < k < j)$ element to ($k+1$)-th element.
  (c) Move $j$-th element to top of LRU stack.
 If $j > m$
  (a) Generate an address by *address generation process* described in Subsection II-C.
  (b) Output the new address.
  (c) If $m = M$
      • Remove bottom element of LRU stack.
      otherwise,
      • $m \leftarrow m + 1$.
  (d) Shift $k$-th $(1 < k < m)$ element to ($k+1$)-th element.
  (e) Insert output address at top of LRU stack.
5. Return to Step 2.
```

This procedure produces a series of destination addresses that can be embedded in the simulation program for packet generation. We propose two procedures for applying it: address sequence generation per packet and address sequence generation per flow. Either can be used to generate packets in the simulation.

We first construct the data structure of the target IP lookup algorithm based on the routing table. For example, we construct the Patricia Tree from the routing table. Then, we obtain the distribution of address lookup delays $F(S)$ from the traced-data and data structure, (e.g., Patricia Tree in our example). Next, we apply one of the generation procedures.

- Address Generation per Packet (AGP)
  1. Determines the destination address of packet to be gathered by using the address sequence generation method and $F(S)$ described above.
  2. Generates a packet according to, for example, a Poisson process.

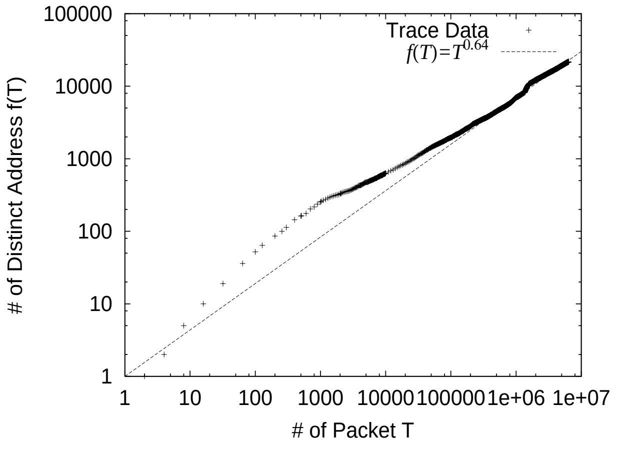


Fig. 4. An Example of ISGF

- Address Generation per Flow (AGF)
  1. Generates a flow, and determines the number of packets in the flow.
  2. Determines the destination address of generated flow by using the address sequence generation method and $F(S)$ described above.
  3. Determines packet interarrival times during which flow is active. (All packets in the flow have same destination address.)

AGP is a simple procedure that assigns an address that LRU stack outputs as each packet. In AGP, the probability that the generated address will appear again in the future depends only on its location in the LRU stack, not on the number of references. This is why the distribution of the number of references in the LRU stack is not heavy-tailed as it is for actual traffic. Therefore, performance predicted using AGP may be defferent from the actual-case performance.

To overcome this weakness, we propose using flow-based address generation, i.e., AGF. In AGF, the number of packets within a flow is determined based on statistical analysis of the traced data. For example, the entire flow duration can be well approximated by the log-normal distribution while the tail–part has a heavy-tailed distributionin [9]. Thus, a combination of the two distributions can be used for the flow duration, which is what we used in our simulations. To accurately combine the two distributions, we estimate the parameters using the maximum-likelihood-estimator (MLE) method [10]. Once the flow is accurately characterized, the packet interarrival times can be modeled as a Poisson distribution [9].

Note that AGP requires less processing overhead and a smaller number of parameters than AGF.

III. Performance Prediction

We used our proposed method to predict the performance of an existing IP lookup algorithms, a Patricia Tree search [7].

A. Traffic Patterns

We first determined the parameters (N_a, N_d, and $F(S)$) from the traced data. We use ten million packet headers gathered by the traffic monitor (OC3MON [8]) at the gateway of Osaka University. The data was collected on January 24, 2001. The ISGF α values were 0.64 for AGP and 0.77 for AGF. The distribution $F(S)$ was shown in Figure 3. We run simulations using both AGP and AGF, and for comparison purpose, two traffic generation procedures.

Actual: A raw sequence of traced packet headers.

Trace-Random: A sequence of addresses generated by the *address generation process*.

We assume that trace-driven simulation using "Actual" traffic provides the actual performance. To judge the validty of "Trace-Random" traffic, AGP or AGF, we compare the results obtained from simulation using them with "Actual" traffic case in Subsection III-C. Note here that due to the time limitations, we do not show the results of AGF in this paper. We will show them in the final version of paper.

B. Performance Metrics

In our simulations, we generated ten million packets based on a Poisson process and input them to the IP lookup algorithms. Their destination addresses are generated by four traffic patterns, "Actual", "Trace-Random", AGP, and AGF. Before running simulations, we have to determine the following three parameters.

1. Buffer size K of the router
2. Traffic intensity ρ
3. LRU stack size M

We denote the buffer size K as the maximum number of packets that can be queued at the router. If a new packet arrives as K packets are queued at the router, it is dropped. Traffic intensity ρ is defined as

$$\rho \quad = \quad \lambda E[S], \tag{5}$$

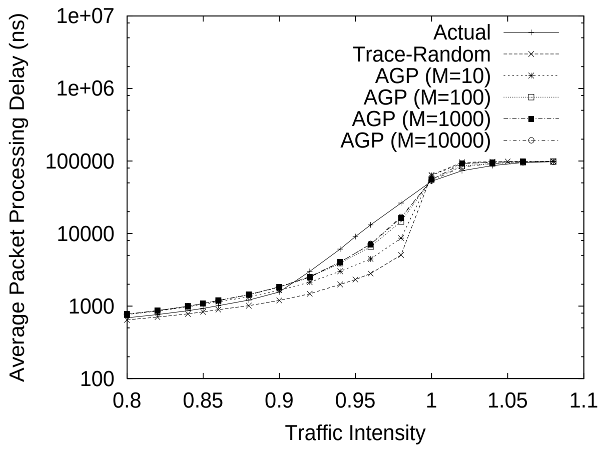
where λ is the packet arrival rate at the router and $E[S]$ is the average value of the address lookup delays of the traced packets. Once we obtain the distribution of the address lookup delays, λ depends only on ρ. The size M of LRU stack is needed when packets are generated using LRU stack in AGP and AGF.

Through simulations, we investigate the average packet processing delay that is defined as the average time duration from when a packet arrives at the router to when the packet is forwarded to the output link. We also investigate the packet loss ratio and the behavior of the time–dependent queue length (the number of packets queued in the buffer) for the performance metrics in the following subsection.

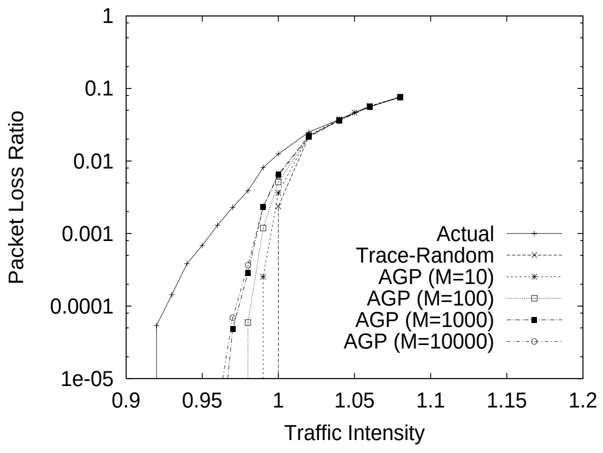
C. Simulation Results

We first examine the adequate LRU stack size M. Figure 5 shows the average packet processing delay and the packet loss probability according to the traffic intensity. As M is large, the average packet processing delay becomes converged and the results of $M = 1,000$ and $M = 10,000$ are almost same. The packet loss ratio of both $M = 1,000$ and $M = 10,000$ are almost same. Accordingly, we set the value M to be $1,000$ in following experiments.

Figure 6 compares the average packet processing delay dependent on the traffic intensity among Actual, Trace-Random, AGP. Comparisons of packet loss ratio are shown in Figure 7. In Figures 6 and 7, we varied the traffic intensity from 0.80 to 1.08,



(a) Average Packet Processing Delay (K=500)



(b) Packet Loss Ratio (K=500)

Fig. 5. Transition of LRU Stack Size M

and two values of the queue length K are examined ($K = 100$ and $K = 500$). From these figures, we can observe that AGP provides better estimation than Trace-Random. While in case of lower traffic intensity ($\rho = 0.82$), the results of Trace-Random is closer to the Actual line than the one of AGP. However, results in the region of low intensity is not important to predict the maximum performance. Moreover, Trace-Random is not "accurate" even in this case because Trace-Random underestimates the average packet processing delay, which causes a critical prediction misleading.

In Figure 8, we show the behavior of the time-dependent queue length when the traffic intensity ρ is set to 0.95. From Figure 8(a), it is observed that the queue length in the "Actual" case sometimes increases significantly. It is caused by the characteristic of packet arrivals. Probably due to the window flow control of TCP, traffic of the TCP connection contains a burst of packets. Then, a significant increase (called as *spike* below) appears when the packets with the long-prefix-matched address arrive in a bursty fashion. On the other hand, in the "Trace-Random" case (Figure 8(b)), any spike does not appear during the simulation. AGP (Figure 8(c)) considers the characteristic that the same address tends to arrive bursty shows many spikes, but their spikes are frequently observed and their amplitudes are not so high, compared with "Actual" case. This is the reason why AGP underestimates the packet loss probability in Figure 7(b).

In the actual traffic case, the number of packets in the flow is heavy-tailed (combination of log-normal and Pareto distributions). On the contrary, in AGP, all flows have the same statistics since the generation probability of addresses depends not on the characteristics of the flow but position of the LRU stack. Therefore, the effect on the behavior of queue length is "averaged" to some extent. We expect that AGF resolves this problem because it considers the flow characteristics (i.e., the number of packets in the flow, interarrival times of flows, and the number of active flows).

IV. Conclusion

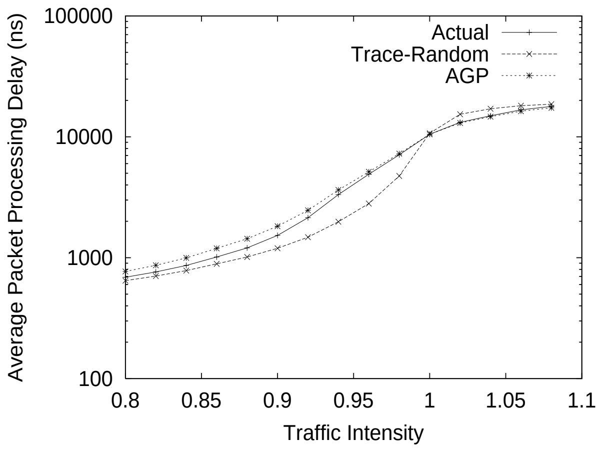
In this paper, we have proposed performance prediction methods of IP lookup algorithms based on the statistical model obtained from the real traffic. We have used a Patricia Tree search as the target algorithm as an example application. We have found out that through a packet driven simulation, we can evaluate them accurately with AGP. However, AGP has a weakness that the flow characteristic is not considered explicitly. To overcome it, we propose using flow-based address generation, i.e., AGF. Due to the time limitations, we do not show the results in this paper, but we will show the results of AGF in the final version of the paper.

Acknowledgements

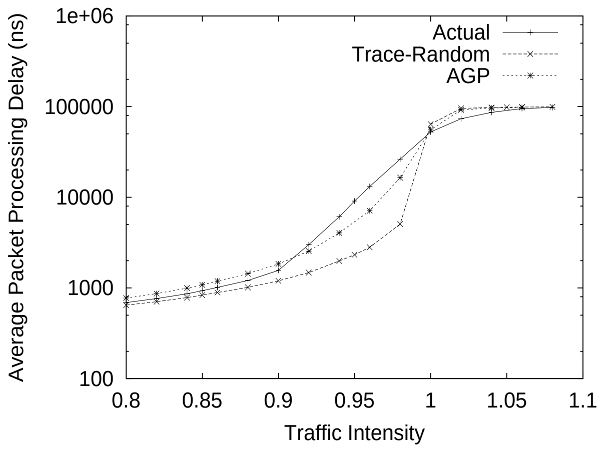
This work was supported in part by Research for the Future Program of Japan Society for the Promotion of Science under the Project "Integrated Network Architecture for Advanced Multimedia Application Systems" (JSPS-RFTF97R16301).

References

[1] M. A. Ruiz-Sánchez, E. W. Biersack, and W. Dabbous, "Survey and taxonomy of IP address lookup algorithms," *IEEE Network*, vol. 15, pp. 8–23, Mar./Apr. 2001.
[2] Y. Rekhter and T. Li, "An architecture for IP address with CIDR." RFC1518, Sept. 1993.
[3] M. Uga and K. Shiomoto, "A fast and compact longest prefix look-up method using pointer cache for very long network address," in *Proceedings of IEEE ICCCN '99*, pp. 595–602, Oct. 1999.
[4] P. Gupta, B. Prabhakar, and S. Boyd, "Near-optimal routing lookups with bounded worst case performance," in *Proceedings of IEEE INFOCOM 2000*, pp. 1184–1192, Mar. 2000.
[5] G. Narlikar and F. Zane, "Performance modeling for fast IP lookups," in *Proceedings of ACM SIGMETRICS 2001*, pp. 1–12, June 2001.
[6] M. Aida and T. Abe, "Pseudo-address generation algorithm of packet destinations for Internet performance simulation," in *Proceedings of IEEE INFOCOM 2001*, pp. 1425–1433, Apr. 2001.
[7] D. E. Knuth., *The Art of Computer Programming*, vol. 3. Addison-Wesley, 1973.
[8] K. Thompson, G. J. Miller, and R. Wilder, "Wide-area Internet traffic patterns and characteristics," *IEEE/ACM Transactions on Networking*, pp. 10–23, Nov. 1997.

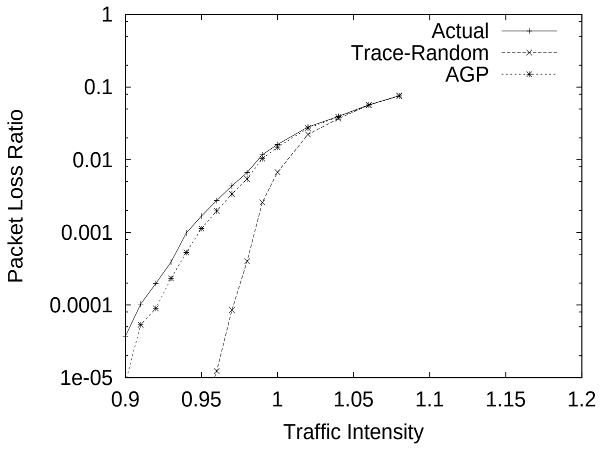


(a) K=100, M=1000

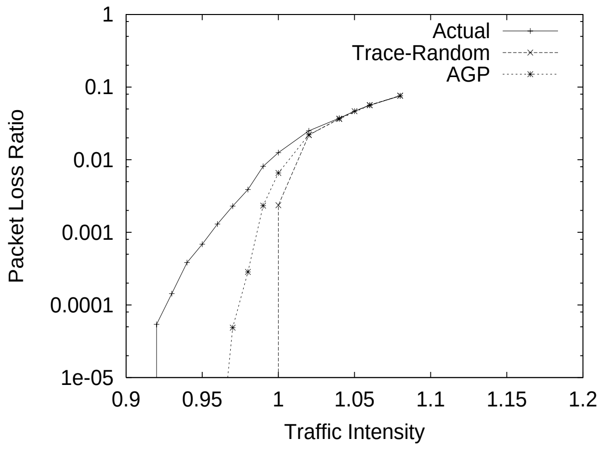


(b) K=500, M=1000

Fig. 6. Average Packet Processing Delay



(a) K=100, M=1000

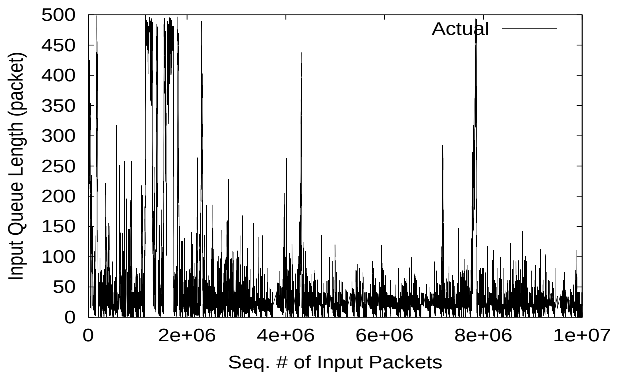


(b) K=500, M=1000

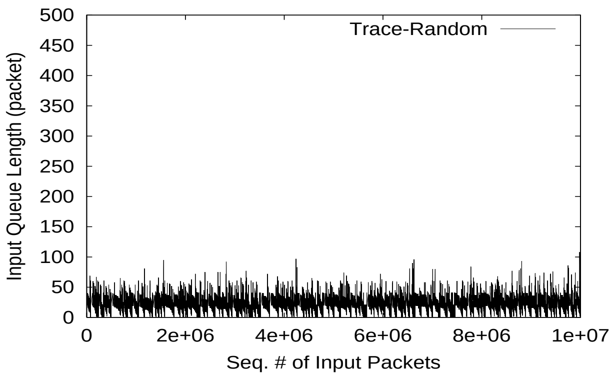
Fig. 7. Packet Loss Ratio

[9] S. Ata, M. Murata, and H. Miyahara, "Analysis of network traffic and its application to design of high-speed routers," *IEICE Transactions on Information and Systems*, vol. E83-D, pp. 988–995, May 2000.

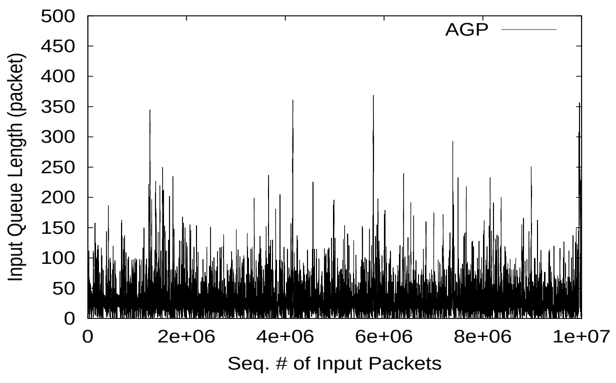
[10] V. Brazauskas and R. Serfling, "Robust and efficient estimation of the tail index of a one-parameter Pareto distribution," *North American Actuarial Journal* available at `http://www.utdallas.edu/˜serfling/`, pp. 12–27, Apr. 2000.



(a) Actual (K=500, ρ=0.95)



(b) Trace-random (K=500, ρ=0.95)



(c) AGP (K=500, ρ=0.95, M=1000)

Fig. 8. Time–dependent Behaviors of Input Queue Length