Master's Thesis

Title

Performance Prediction Method for Address Lookup Algorithms based on Statistical Traffic Analysis

Supervisor

Prof. Masayuki Murata

Author

Ryo Kawabe

February 13th, 2002

Department of Informatics and Mathematical Science

Graduate School of Engineering Science

Osaka University

Master's Thesis

Performance Prediction Method for Address Lookup Algorithms based on Statistical Traffic Analysis

Ryo Kawabe

Abstract

Many address lookup methods for IP routers to improve a packet forwarding capability have been recently proposed, but evaluation on their performance prediction is very limited because of lack of consideration on actual traffic characteristics. It is necessary to consider the actual traffic characteristics to predict more realistic performances of routers, and in this thesis, we propose methods for predicting the router's performance based on the statistical analysis of the Internet traffic. Our method consists of three steps: (1) creates destination addresses based on traffic statistics derived from traced data, (2) models the tendency of packet arrivals by using ISGF, whose parameter is determined by analyzing the actual traffic, and generates address sequences by arranging addresses created in step (1), and (3) implements address lookup algorithms on a trace-driven simulator and evaluates their performance using traffic patterns generated in step (2). From simulation results, we show that our method can provide accurate performance prediction for address lookup algorithms. Furthermore, we use an approximate mathematical approach for quick performance prediction and discuss its accuracy compared with simulation results.

Keywords

Address Lookup, Performance Prediction Method, Statistical Traffic Analysis, Traffic Generation, ISGF (Inverse Growth Stack Function), Flow Characteristics

Contents

List of Figures

List of Tables

1 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 has become more complicated since classless inter-domain routing (CIDR) [1] was introduced. Figure 1 gives an example of the longest prefix matching. In the routing table, each routing entry consists a pair of sub network address and its prefix length. When packet whose destination address is 10101011 arrives at the router, the second and third entries are matched in the routing table. The router then selects the third entry because the prefix length on the third entry is longer and outputs the packet to port C. The longest prefix matching requires more complicated process than simple fixed length comparison, since it is necessary to select the longest prefix out of some matching prefixes. Address lookup has thus become a performance bottleneck in high-speed routers.

While many approaches have been proposed to overcome this bottleneck, in e.g., [2-8], 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

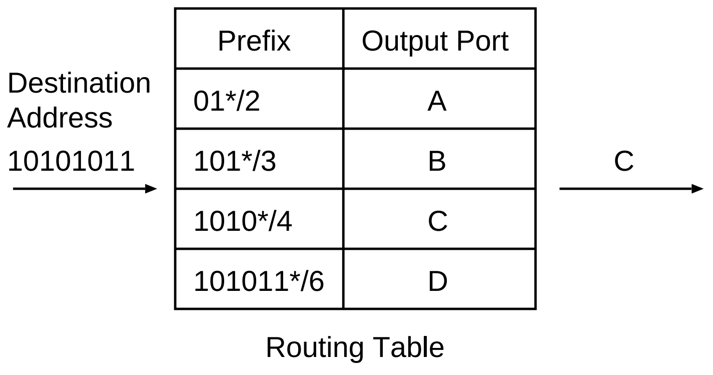


Figure 1: An Example of Longest Prefix Matching

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 address lookup algorithms thus use worst-case performance.

However, actual performance is greatly affected by the sequence of the destination addresses of arriving packets. Therefore, it is not always the best metric for the customers. An algorithm designed to maximize worst-case performance may be very expensive. A closer look at the performance of the target address 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 [9]. 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, packet loss ratio, and the packet-processing delay, to find a more appropriate design choice.

In this thesis, we propose a method for predicting router performance based on statistical analysis of the Internet traffic. Section 2 describes its detail and Section 3 presents an example of its application to an address lookup algorithm and show that, based on simulation results, our method can accurately predict performance. In Section 4, we also investigate the analytic method by using the queueing system for quick (but rough) performance prediction.

2 Performance Prediction Method for Address Lookup Algorithms

2.1 Overview

We will first give an overview of our proposed method for predicting the performance of address lookup algorithms. Problems of existing methods are also discussed. As shown in Figure 2, our method consists 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 of the address lookup algorithm 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 entries tend to include more organizations. Because the most popular traffic in the current Internet is `http` [10], 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 [11] 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. Therefore, we have to consider the characteristics of the sequence of packet arrivals when we generate traffic

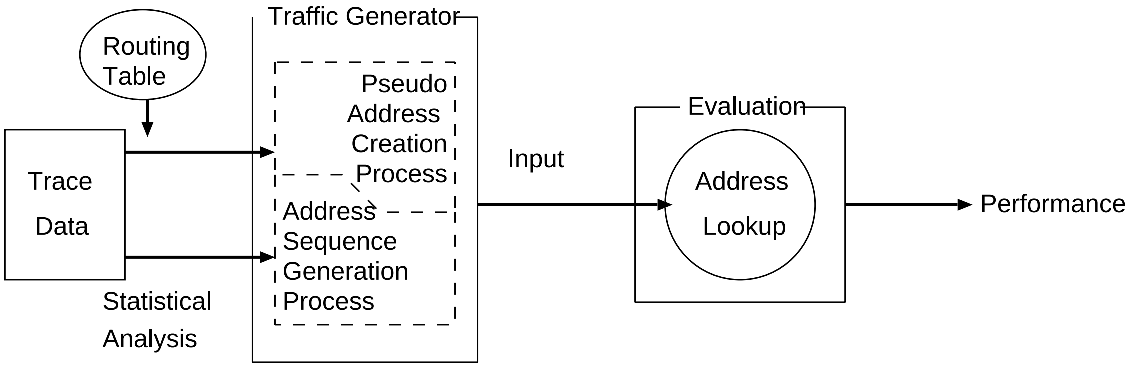


Figure 2: Overview of Our Proposed Performance Prediction Method

patterns. To model the tendency of packet arrivals, Aida and Abe [12] described the approach in which address sequences are generated using a least recently used (LRU) stack model [13] on the basis of the notion of an inverse stack growth function (ISGF). Although it can practically grasp the tendency of the packet arrivals, it does not consider the flow characteristics of the Internet traffic. Accordingly, we improve the approach and propose a method of address sequence generation considering the flow characteristics to evaluate the actual performance of address lookup algorithms. We generate traffic patterns by creating addresses in the first process and arranging them in the second process.

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 implement address lookup algorithms on a trace-driven simulator and evaluate their performance using our traffic generation method.

We discuss these three processes in more detail in the following subsections. After describing our target address 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 model based on the notion of ISGF. The third process in our performance prediction method is described with an instance of application in Section 3.

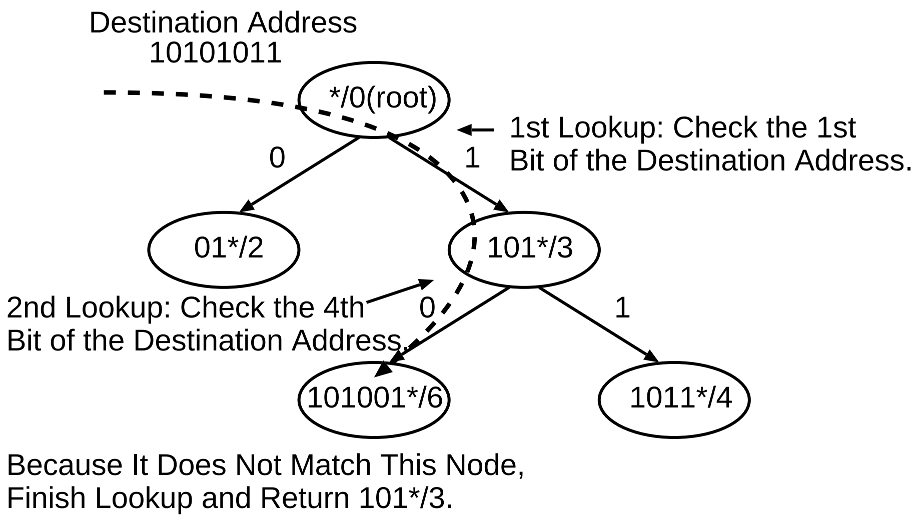

Destination Address
10101011
*/0(root)
1st Lookup: Check the 1st Bit of the Destination Address.
0
1
01*/2
101*/3
2nd Lookup: Check the 4th Bit of the Destination Address.
0
1
101001*/6
1011*/4
Because It Does Not Match This Node, Finish Lookup and Return 101*/3.

Figure 3: Example Patricia Tree Search

2.2 Target Address Lookup Algorithm

Because our pseudo address creation method uses algorithm–dependent parameters, we refer to the address lookup algorithm to be evaluated by our method as the *target algorithm*. As example of the target address lookup algorithm is a Patricia Tree Search, which is well known to be simple and easily implemented for address lookups. We will use the Patricia Tree for explaining our proposed method in the following subsections.

Legacy routers use a binary tree for longest prefix matching. A Patricia Tree Search [14] eliminates nodes having only one child node. In the example Patricia Tree Search shown in Figure 3, 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 [15], 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.

2.3 Pseudo Address Creation Process

In this subsection, we describe our pseudo address creation process. We first investigate the characteristics of address lookup delays and then explain our method 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

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

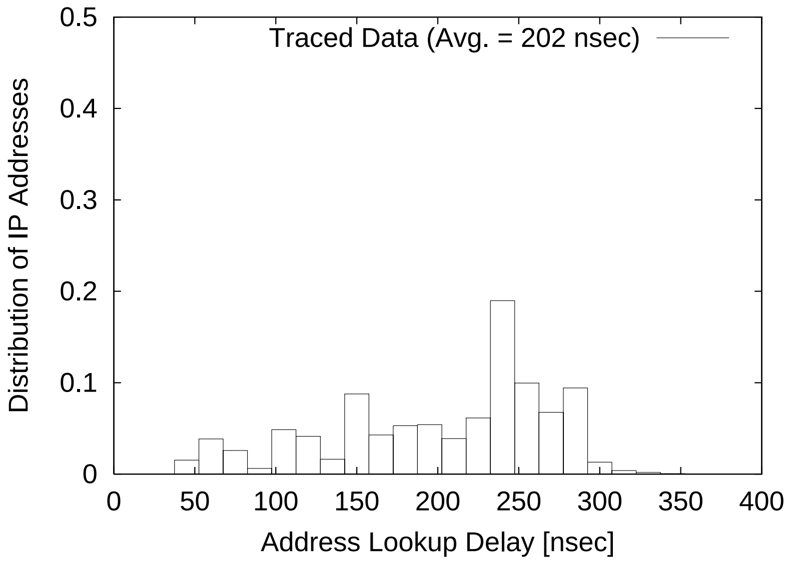
where d_r is the delay due to read/comparison operations in RAM. If a read requires 5 nsec and a comparison requires 10 nsec, d_r is 15 nsec. In the example shown in Figure 3, since two lookups are required to determine the longest prefix entry ($101*$) of the packet, D_i is 2 and S_i is 30 nsec.

To obtain the distribution of S_i, we need an actual routing table. In the example of this thesis, 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 [16]). 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 $(1 \leq i \leq N_p)$ and finally we obtain the distribution of S_i. Figure 4(a) shows the example of the distribution of S_i (referred to as $F(S)$ throughout this thesis) derived from 10,000,000 packets obtained on January 24th, 2001. There is a large difference between the actual distribution and the distribution derived from randomly generated addresses as shown in Figure 4. It is clear that it leads to the gap between performances evaluated through trace-driven simulation using their addresses, and a simulation technique using random address generation cannot predict the realistic performance of address lookup algorithms.

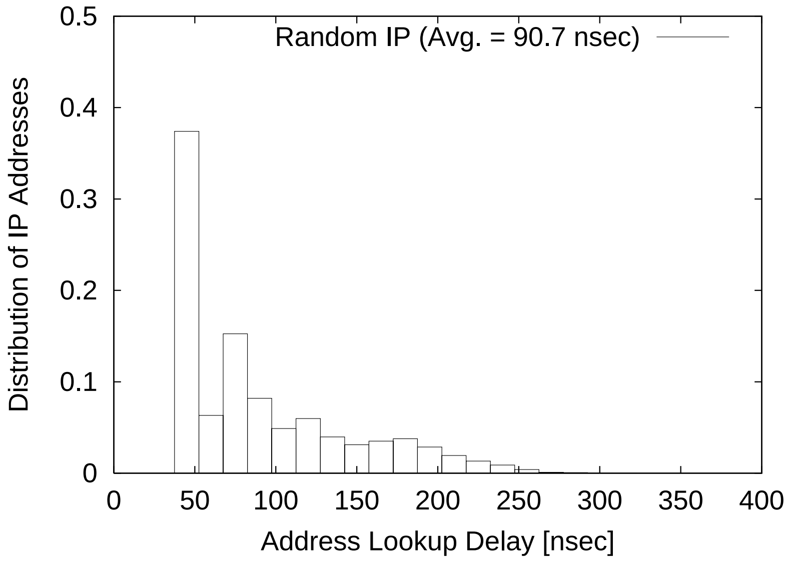
In our proposed method, addresses are created according to $F(S)$ obtained by traced data. The pseudo address creation process works as follows.

1. Obtain the distribution $F(S)$ from traced data.

2. Choose a random number p $(0 \leq p < 1)$.



(a) Traced Data



(b) Randomly Generated Addresses

Figure 4: Distribution of Address Lookup Delays

3. Determine the minimum S_i satisfying $p \leq \sum_{S=1}^{S_i} F(S)$.

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.

2.4 Address Sequence Generation Process

As described in Subsection 2.1, another important factor that must affect the performance of address lookup is the addresses of arriving packets. In this subsection, we describe our address sequence generation method.

2.4.1 Modeling Packet Arrival Sequences

We use the least recently used (LRU) stack model on the basis of the notion of the inverse stack growth function (ISGF) to model the tendency of packet arrivals and generate address sequences [12].

To briefly summarize ISGF, we introduce t_i, denoting the arrival time of the i-th packet (or flow). Inverse stack growth function (ISGF), $f(t, T)$, is denoted as the expected number of distinct destination 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. It is revealed that ISGF satisfies 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 [12]. That is, if T packets

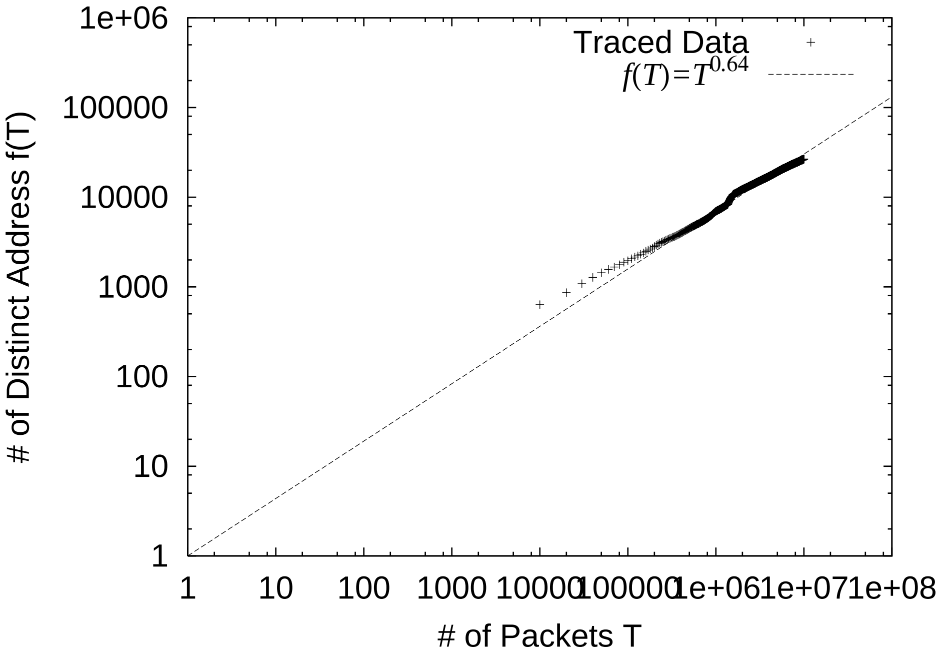


Figure 5: An Example of ISGF

are collected from traced data, the number of distinct destination addresses can be estimated as T^{α}. Eq. (2) is confirmed by an example result of ISGF presented in Figure 5. 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. Arrived distinct addresses are stored in the LRU stack in descending order of arrival time, i.e., the most 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., the address of the top element on the LRU stack will be arrived again with the probability a_1.

By using ISGF, a_i is given by

$$a_i = \{f(g(i-1)+1)-(i-1)\} - \{f(g(i)+1)-i\}, \tag{3}$$

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

$$a_i = \{((i-1)^{\frac{1}{\alpha}} + 1)^{\alpha} - (i-1)\} - \{(i^{\frac{1}{\alpha}} + 1)^{\alpha} - i\}. \tag{4}$$

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, we consider two IP addresses. The one is located at bottom of the LRU stack, the other is next to the former one. If the size of the LRU stack is very large, the probabilities of each address to be accessed are almost same and very small. Therefore, the LRU stack does not need to have them since these two addresses remain nearly unaffected by ISGF. Thus, the extremely large size of the LRU stack is not meaningful, and only results in the time and memory consumption in algorithm execution. In principle, the LRU stack model assumes an infinite size of the stack. However, in actual, we have to set the LRU stack size to the finite value, and further, we want to set it to the appropriate size to reduce the processing overhead. We consider the address generation method taking account of the finite stack depth suitable to our problem. We will discuss about the appropriate size of the LRU stack for the accurate performance prediction in Section 3.

2.4.2 Proposed Traffic Generation Methods

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: α and $F(S)$. We refer to this procedure as *Address Sequence Generation Procedure*. We denote m as the number of elements on the LRU stack and M as the maximum number of the stack size.

Address Sequence Generation Procedure:

1. Set $m = 0$.

2. Choose a random number p $(0 \leq p < 1)$.
3. Obtain the minimum j satisfying $p \leq \sum_{i=1}^{j} a_i$.
4. If $j \leq m$
   - (a) Output the j-th element of LRU stack.
   - (b) Shift the k-th $(1 < k < j)$ element to the ($k+1$)-th entry.
   - (c) Move the output address to the top of LRU stack.

   If $j > m$
   - (a) Generate an address by the *pseudo address creation process* described in Subsection 2.3.
   - (b) Output the new address.
   - (c) If $m = M$
     - Remove the bottom element of LRU stack.

     otherwise,
     - $m \leftarrow m + 1$.
   - (d) Shift the k-th $(1 < k < m)$ element to the ($k+1$)-th entry.
   - (e) Insert the output address at the 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.

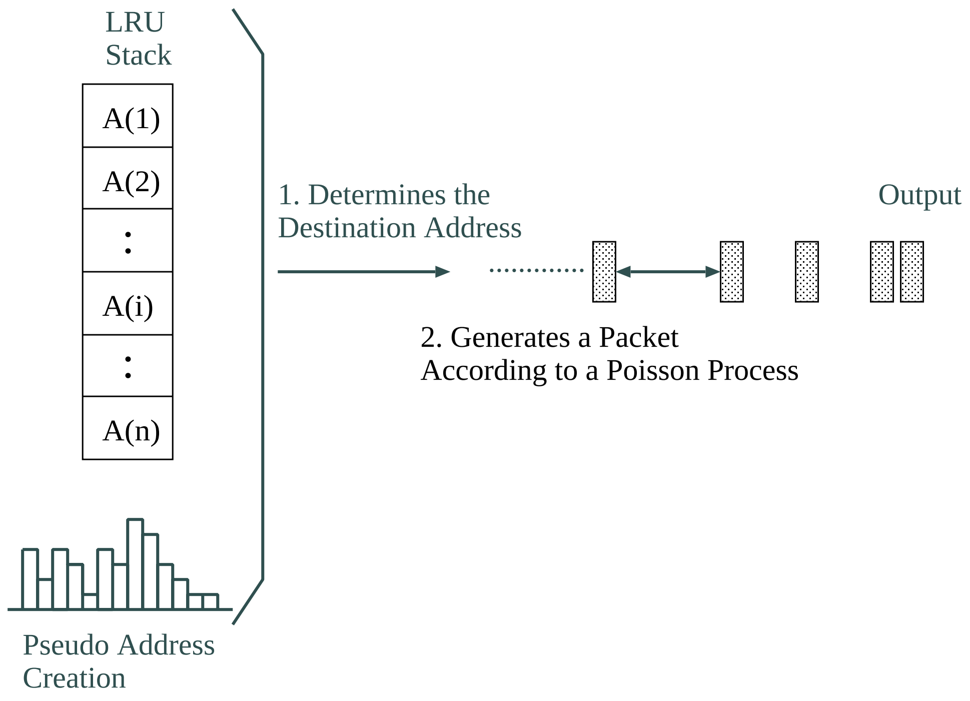

LRU Stack
A(1)
A(2)
:
A(i)
:
A(n)
Pseudo Address Creation
1. Determines the Destination Address
Output
2. Generates a Packet According to a Poisson Process

Figure 6: Traffic Generation with AGP

Our packet generation process is summarized as follows. We first construct the data structure of the target address 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 generate a traffic pattern by using method by applying the above address generation procedure. To put it concretely, a traffic pattern is generated per packet according to the following procedure.

1. Determines the destination address of packet to be generated by using the address sequence generation procedure described above.

2. Generates a packet according to, for example, a Poisson process.

We call it as Address Generation per Packet (AGP). The overview of AGP is illustrated in Figure 6. AGP is a simple procedure that assigns an address that LRU stack outputs to each packet.

Note that AGP does not consider the flow characteristics of actual traffic. For example, the probability that the generated address will appear again in the future only depends on its location in the LRU stack, not on the number of references in AGP. Thus, the flow distribution (the number of packets in the same flow) cannot be modeled well, and we may miss its effect on the performance. Actually, the flow distribution is important in performance prediction as well as demonstrated in Section 3. Our new traffic generation method, called Address Generation per Flow (AGF), takes into account flow statistics as follows (see also Figure 7).

1. Generates a flow according to, e.g., a Poisson process.
2. Determines the number of packets in the flow according to a given flow distribution.
3. Determines the destination address of generated flow by using the address sequence generation procedure described above.
4. Determines interarrival times of packets in the flow. (All packets in the flow have same destination address.)

In AGP, the ISGF function $f(T)$ was introduced to model the expected number of distinct destination addresses in T packets. In AGF, on the other hand, the ISGF function is not used for packets, but for flows. Figure 8 shows an example result of ISGF on destination addresses of flows. In this example, we treat a sequence of packets which have the same (source IP, destination IP, source port, destination port, protocol) values in their headers as one flow. As can be observed in the figure, the distinct destination addresses of flows also satisfy the ISGF function.

In AGF, the number of packets in a flow is also necessary in Step 2. We follow the results on statistical analysis of flows in [17]. The distribution for the number of packets within the flow can be approximated by the log-normal distribution, except the tail part that is modeled by the Pareto distribution. Results of statistical modeling in our experiments are shown in Figures 9 and 10. We can confirm the same result, and hence use the distribution of numbers of packets within the

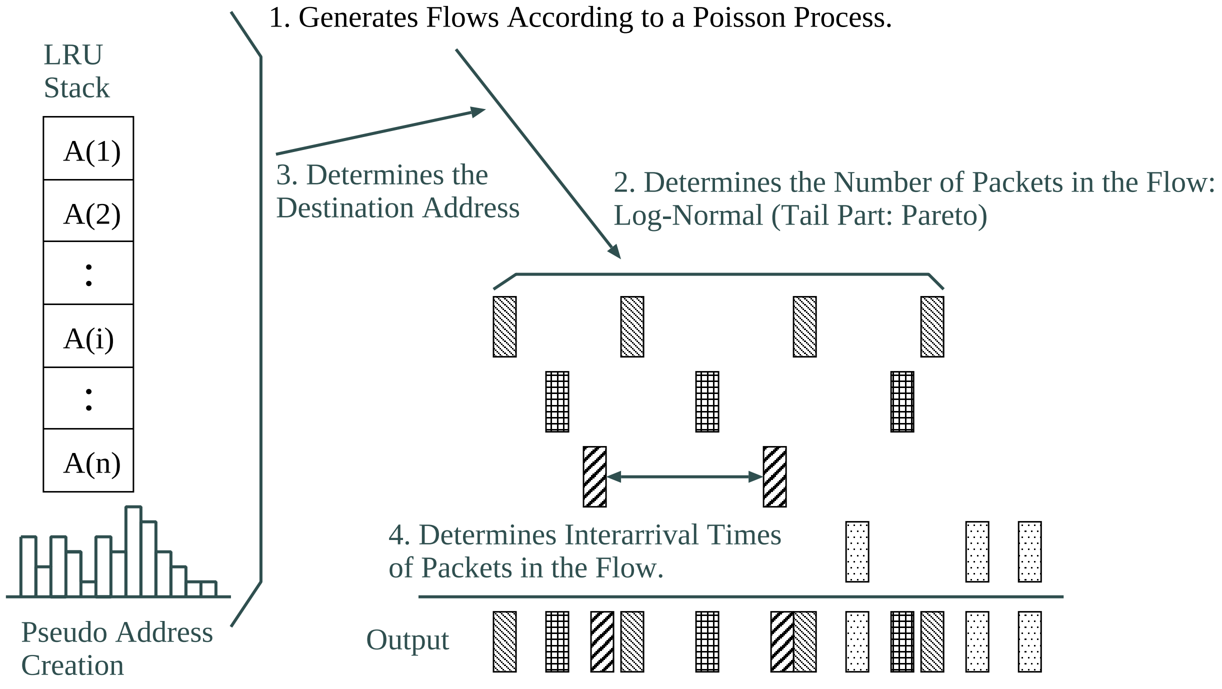

1. Generates Flows According to a Poisson Process.
LRU Stack
A(1)
A(2)
:
A(i)
:
A(n)
Pseudo Address Creation
3. Determines the Destination Address
2. Determines the Number of Packets in the Flow: Log-Normal (Tail Part: Pareto)
4. Determines Interarrival Times of Packets in the Flow.
Output

Figure 7: Traffic Generation with AGF

flow as the combination of log-normal and Pareto functions in evaluating accuracies of our method presented in Section 3. Note that more specifically, we will apply the log-normal function 0% to 95% region of the distribution, and the Pareto function is used for above 95% region. Parameters of log-normal and Pareto functions are determined by an MLE (Maximum Likelihood Estimator) method [18]. Furthermore, we will generate flows according to a Poisson process, and assume interarrival times between packets in the same flow are exponentially distributed [17].

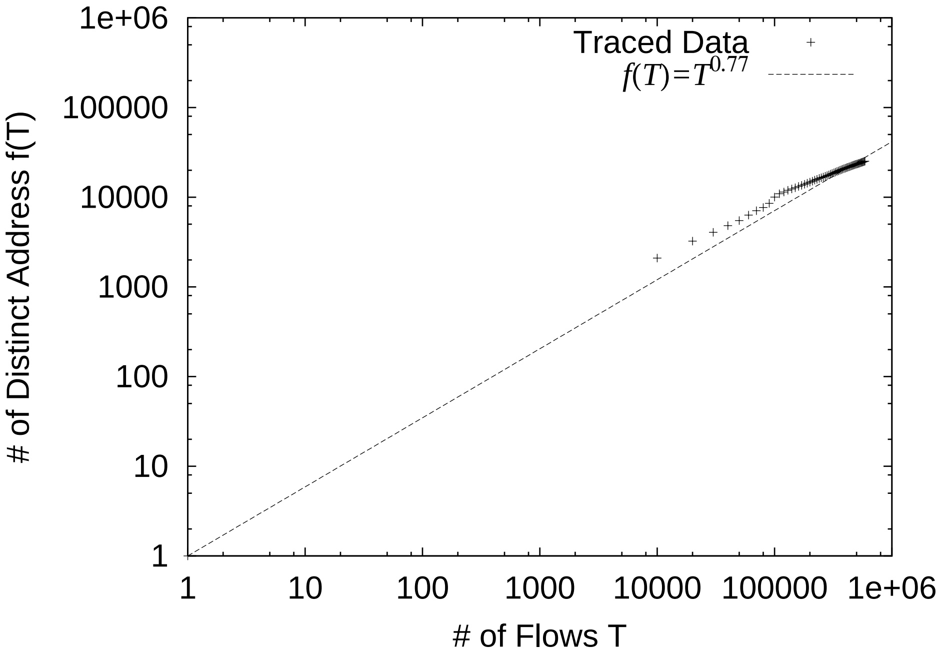


Figure 8: An Example of ISGF on Destination Addresses of Flows

3 Accuracies of Performance Prediction in Proposed Method

To assess the accuracy of our prediction method, we applied our proposed method to evaluate the performance of existing address lookup algorithms. We use a Patricia Tree search as an example of applications.

3.1 Input Traffic Patterns

We first determined the parameters (α and $F(S)$) from the traced data. We use ten million packet headers gathered by the traffic monitor (OC3MON [16]) at the gateway of Osaka University. The data was collected on January 24th, 2001. The ISGF α values were 0.64 for AGP and 0.77 for AGF. The distribution $F(S)$ was shown in Figure 4(a). We run simulations using both AGP and AGF, and for comparison purpose, the following three traffic generation procedures:

Actual: A raw sequence of traced packet headers.

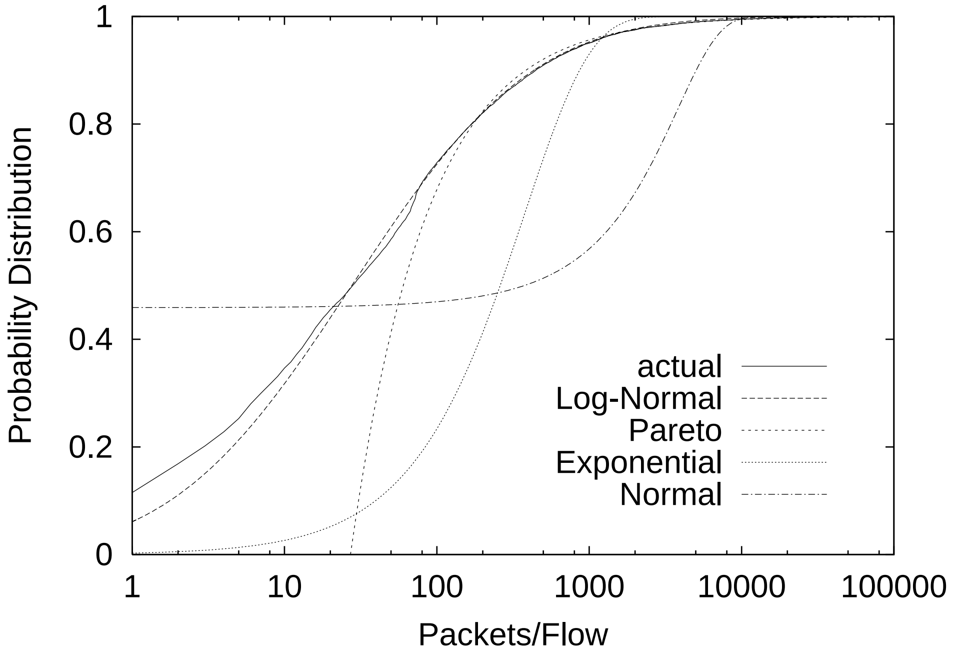


Figure 9: Distribution for Number of Packets in Flows

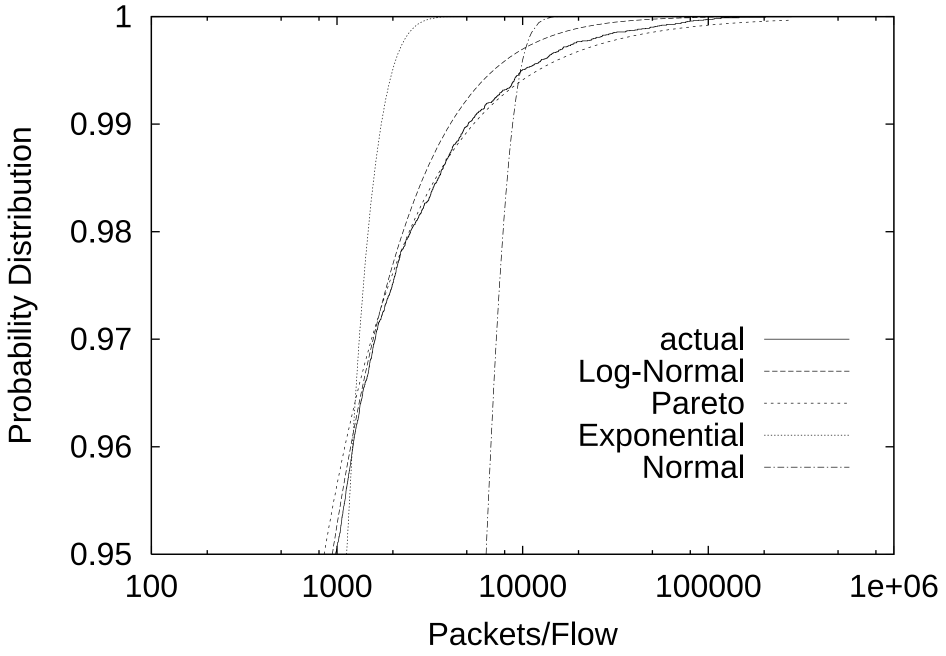


Figure 10: Distribution for Number of Packets in Flows (Tail Part)

Random: A randomly chosen 32-bit value is used as the destination address of each packet.

Trace-Random: A sequence of addresses generated by the *pseudo address creation process* described in Subsection 2.3. Different from AGP or AGF, Trace-Random creates all packets by the pseudo address creation process only. The tendency of packets sequences is not considered.

We suppose that trace-driven simulation using “Actual” traffic provides the actual performance. To investigate the accuracy of other four above-mentioned traffic patterns, we compare results obtained from simulation using them with “Actual” traffic case in Subsection 3.3.

3.2 Performance Metrics

In each simulation experiment, we generated ten million packets as an input of address lookup algorithms. Their destination addresses are generated by five traffic patterns, “Actual”, “Random”, “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 when 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

$$\rho = \lambda E[S], \qquad (5)$$

where λ is the packet arrival rate at the router and $E[S]$ is the average value of address lookup delays for traced packets. The size M of the LRU stack should be determined when packets are generated using the LRU stack in AGP and AGF, which will be examined in Subsection 3.3.1.

Through simulation, we compare the following performance metrics.

- Maximum Throughput

  The maximum throughput is defined as the highest input rate that the router processes the packet without any loss during the simulation. To increase the input rate, the average packet interarrival time is simply decreased in “Actual”, “Random”, “Trace-Random”, and AGP. In AGF, on the other hand, increasing the flow arrival rate can also be considered in addition to decreasing the packet interarrival time. However, increasing the flow arrival rate leads to an increase in the number of *active flows*. A different numbers of active flows may also affect the router’s performance. For example, the case with a small number of active flows tends to repeat packets with the same destination addresses rather than a larger number case, because more distinct addresses are mixed if the number of active flows is large. From this reason, we fix the flow arrival rate (which can be determined from the traced data), and decrease the average interarrival time as well as other traffic generation methods.

- Average Packet Processing Delay

  An average packet processing delay 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.

- Packet Loss Ratio

  As described in the previous subsection, we set the finite buffer size K, and regard the packet arrived when the queue length is K as a lost packet. We count the number of packet losses during the simulation, and calculate the packet loss ratio from the number of packet losses divided by the simulation time.

- Time-dependent Queue Length Behavior

  To investigate the detailed behavior at the router, we also observed the time-dependent transition of queue length in our simulation.

Table 1: Maximum Throughput

Input Traffic	Maximum Throughput	Error Rate
Actual	4.57 mpps	—
Random	9.35 mpps	105%
Trace-Random	4.85 mpps	6.13%
AGP	4.85 mpps	6.13%
AGF	4.76 mpps	4.16%

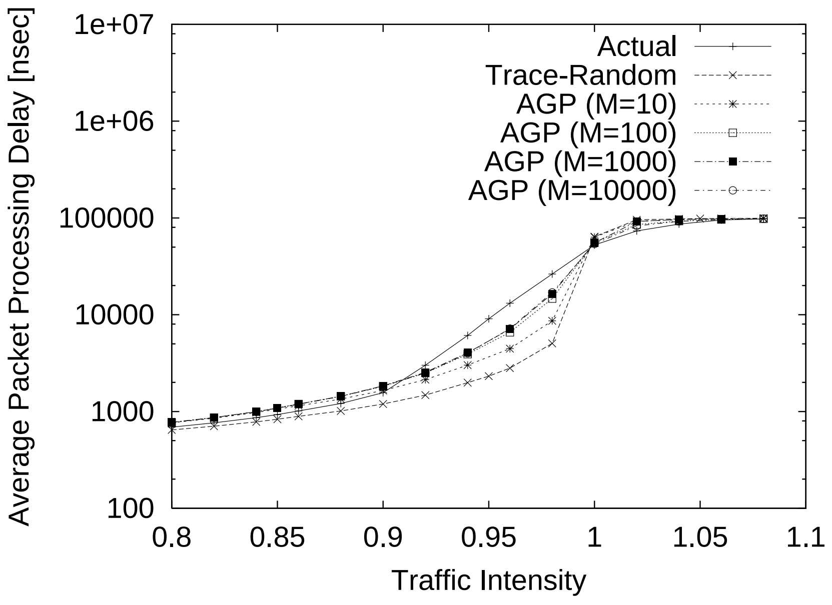
3.3 Simulation Results

3.3.1 Effect on the LRU Stack Size

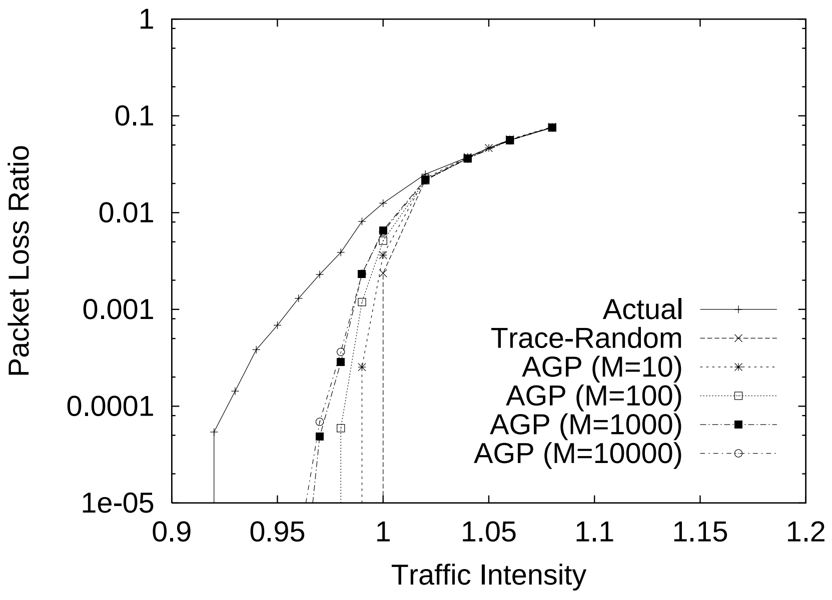
Figure 11 shows the average packet processing delay and the packet loss probability according to the traffic intensity. Values of both average packet processing delay and packet loss ratio become almost same in cases of $M = 1,000$ and $M = 10,000$. For light overhead of computation, a small value of M is preferred. We set the size M to 1,000 in the following experiments.

3.3.2 Comparisons of Maximum Throughput

Table 1 compares simulation results for the Patricia Tree search among five traffic patterns ("Actual", "Random", "Trace-Random", AGP and AGF). The buffer size of the router K is set to be 3,000 packets. In Table 1, the second column shows the maximum throughput and the third column does the relative error ratio to the maximum throughput of the "Actual" case. It can be seen in the table that the throughput of "Random" case is about twice larger than the one of the "Actual" case. The difference between them is caused by the gap between distributions of their address lookup delays shown in Figure 4. In contrast to the "Random" case, results of the "Trace-Random" case, AGP and AGF provide good estimations with low errors. Among three methods, AGF provides the best prediction with respect to the maximum throughput. Note that since the



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



(b) Packet Loss Ratio ($K = 500$)

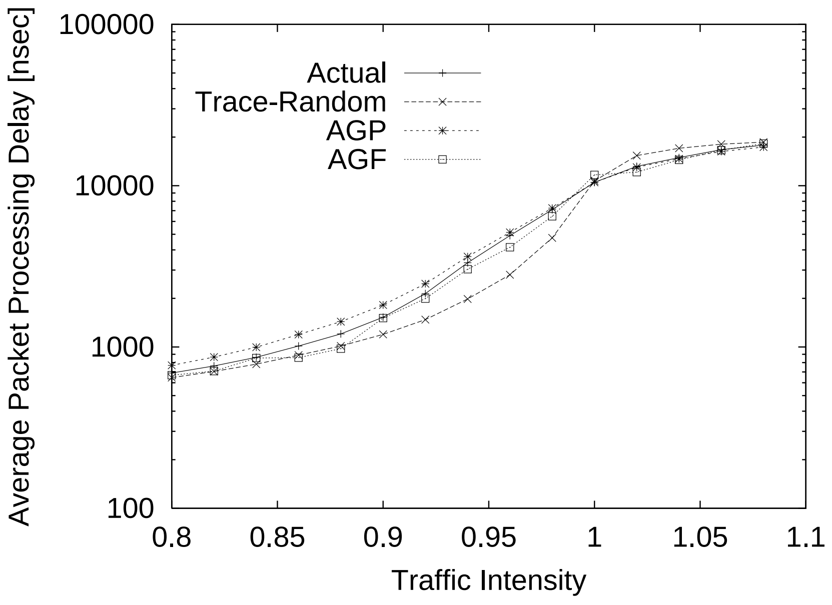
Figure 11: Transition of LRU Stack Size M

result of the “Random” case is far from other traffic patterns, we will not show the result of the “Random” case in the below.

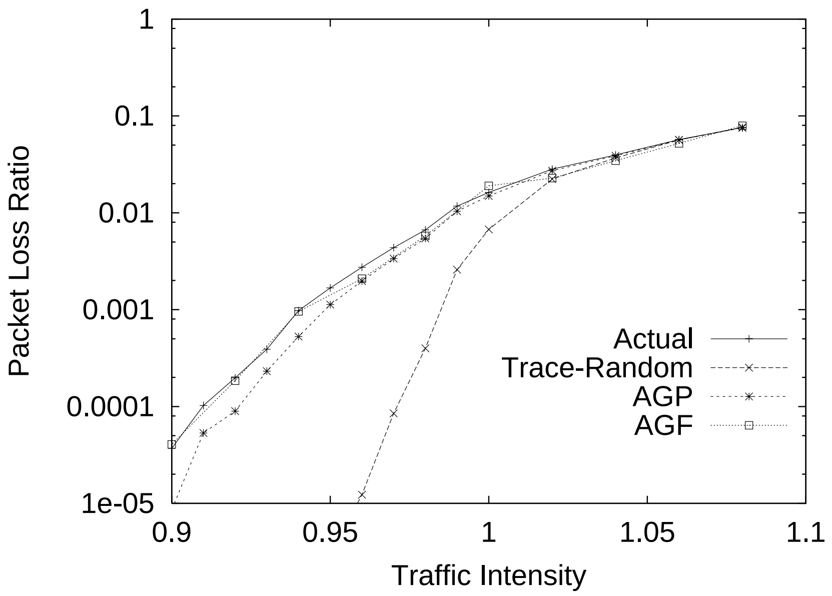
3.3.3 Comparisons of Average Processing Delays and Loss Ratios

Figures 12, 13 and 14 compare the average packet processing delay and packet loss ratio dependent on the traffic intensity among “Actual”, “Trace-Random”, AGP and AGF. In these figures, we varied the traffic intensity from 0.80 to 1.08, and three values of the queue length K are examined ($K = 100$, $K = 500$ and $K = 3000$). From these figures, we can observe that AGP and AGF provide better estimation than “Trace-Random”. While in case of lower traffic intensity ($\rho = 0.82$), the results in “Trace-Random” is often closer to the “Actual” line than the ones in AGP and AGF. However, results in the region of low intensity is not important to predict the performance. A more notable point is that “Trace-Random” underestimates the average packet processing delay and packet loss ratio in the region of $\rho = 0.96$. AGP and AGF also underestimate the performance, however, their differences are limited.

We then focus on comparisons of AGP and AGF. When the buffer size K is 100 packets, their results are almost same as shown in Figure 12. As K is set to be large, a gap between both results becomes spread, which are shown in Figure 13 and 14. We can explain the reason as follows. In the actual traffic case, the number of packets contained in the flow is heavy-tailed (exactly, we used a combination of log-normal and Pareto distributions), and many flows are activated simultaneously. On the contrary, in AGP, once the address is determined, it tends to arrive continuously for a certain period. However, in AGP, all flows have the same statistics since their generation probabilities of addresses depend not on the characteristics of the flow durations but only on the position of the LRU stack. This is why we proposed the AGF taking account of the flow characteristics (i.e., the number of packets in the flow, interarrival times of flows). Moreover, the advantage of AGF is that it can be directly applied to the layer 4 switches where some scheduling is performed on a per flow basis, but its evaluation is necessary, which is left to be a future research topic.

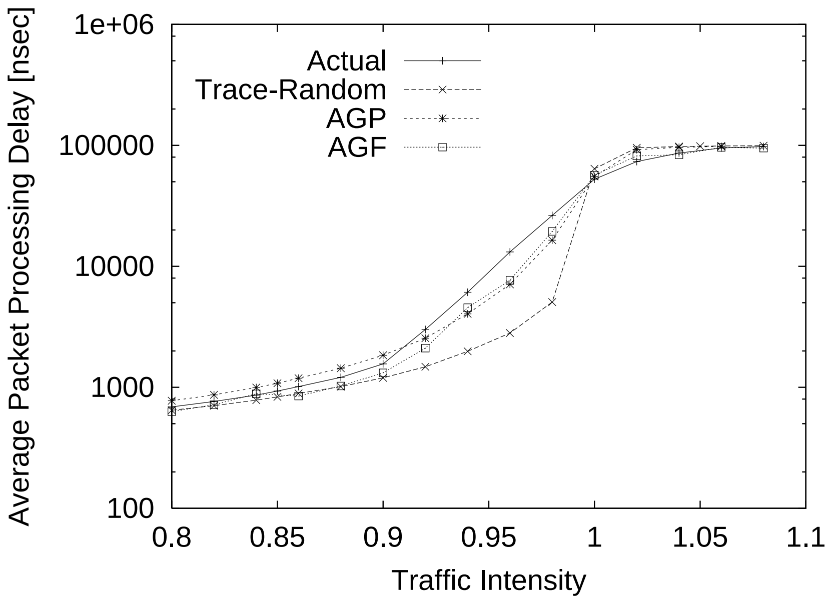


(a) Average Packet Processing Delay

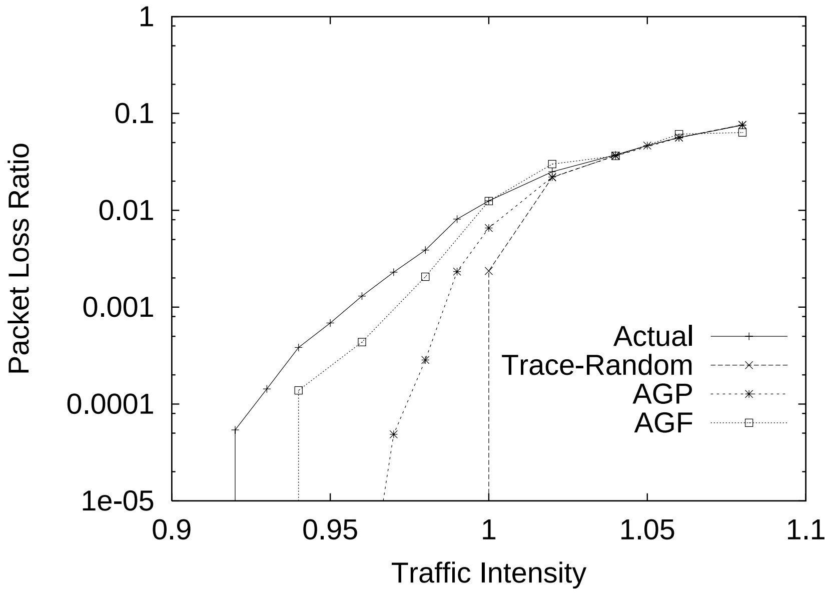


(b) Packet Loss Ratio

Figure 12: Performance Prediction Result ($K = 100$)

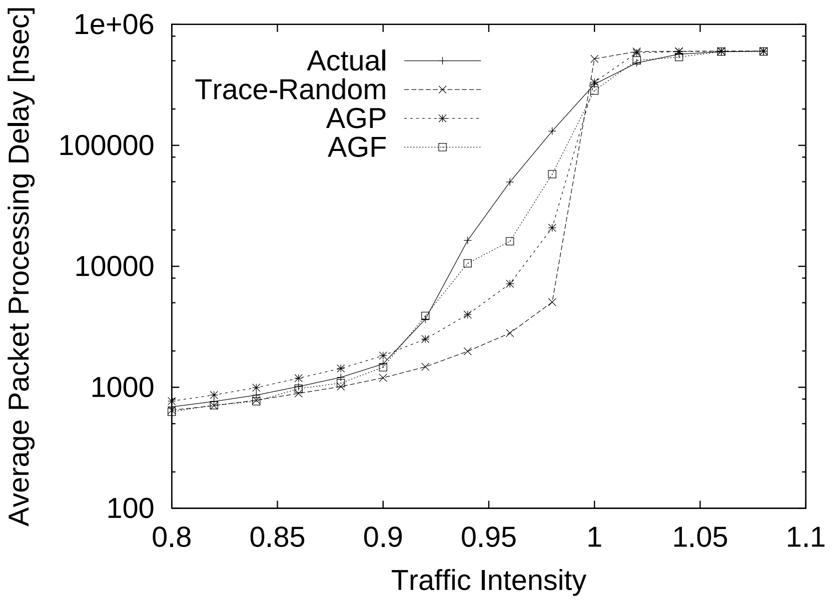


(a) Average Packet Processing Delay

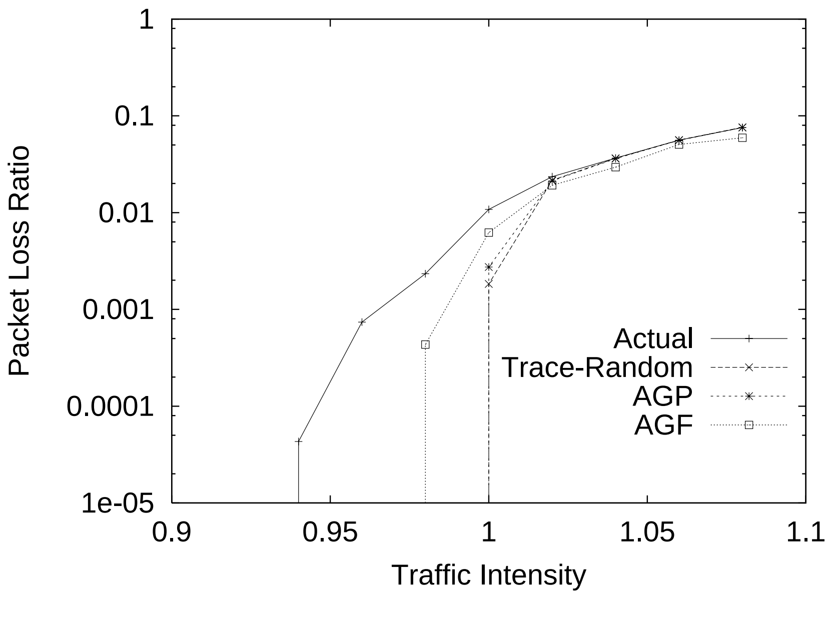


(b) Packet Loss Ratio

Figure 13: Performance Prediction Result ($K = 500$)



(a) Average Packet Processing Delay



(b) Packet Loss Ratio

Figure 14: Performance Prediction Result ($K = 3000$)

3.3.4 Comparisons of Behaviors of Time-Dependent Queue Length

Observing the behavior of the time-dependent queue length helps understanding above the observations. From Figures 15 through 18, the buffer size K is set to 500 packets and the traffic intensity ρ to 0.94. From Figure 15, it can be observed that the queue length in the “Actual” case sometimes increases significantly. It is caused by the characteristics 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 *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 16), any spike does not appear during the simulation. AGP (Figure 17), which considers the characteristic that the same address tends to arrive bursty, shows many spikes, but those are frequently observed and their amplitudes are not so high, compared with the “Actual” case. This is the reason why AGP underestimates the packet loss probability in Figures 13 and 14. On the contrary, the behavior of AGF is similar to that of “Actual case”. It is due to the consideration on the flow characteristics. This tendency becomes more apparent in the case when the buffer size K is 3000 and the traffic intensity ρ is 0.96.

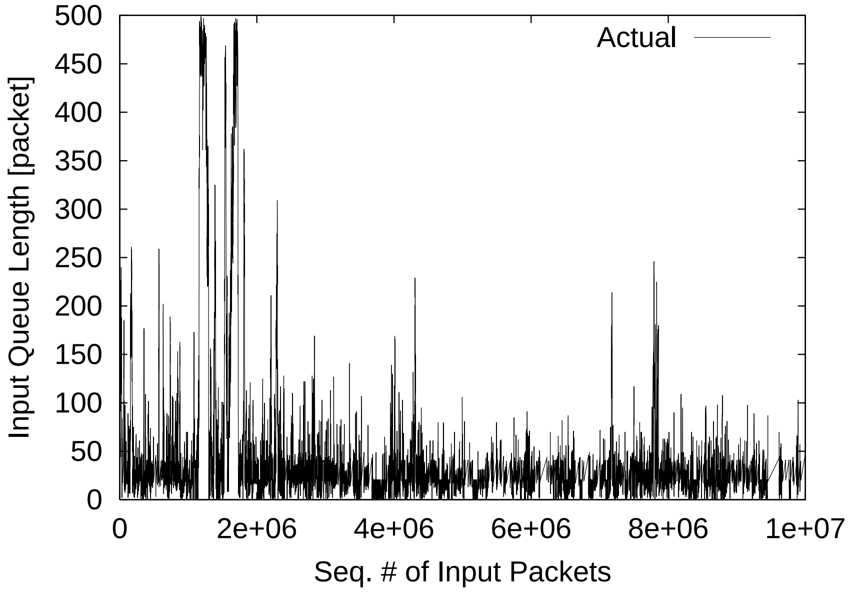


Figure 15: Time-dependent Behavior of Input Queue Length in Actual Traffic ($K = 500$, $\rho = 0.94$)

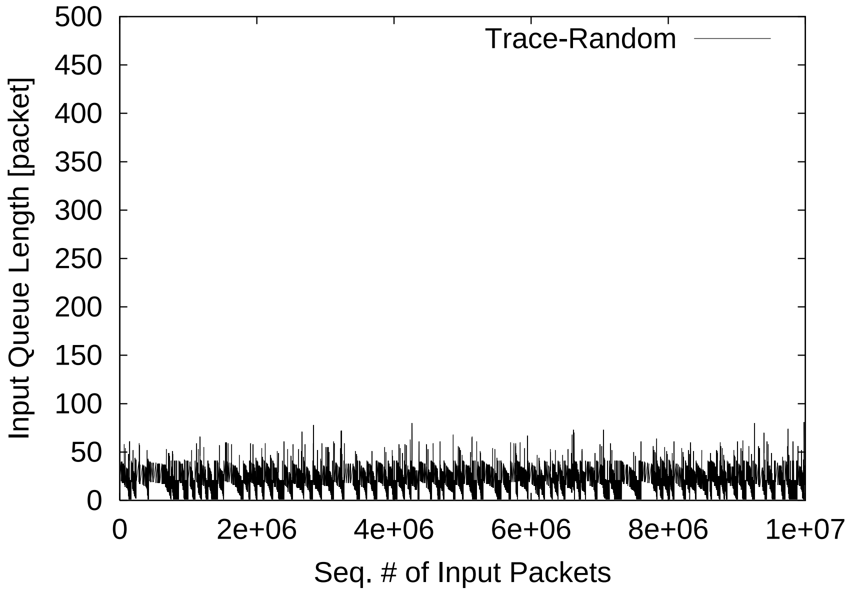


Figure 16: Time-dependent Behavior of Input Queue Length in Trace-Random Traffic ($K = 500$, $\rho = 0.94$)

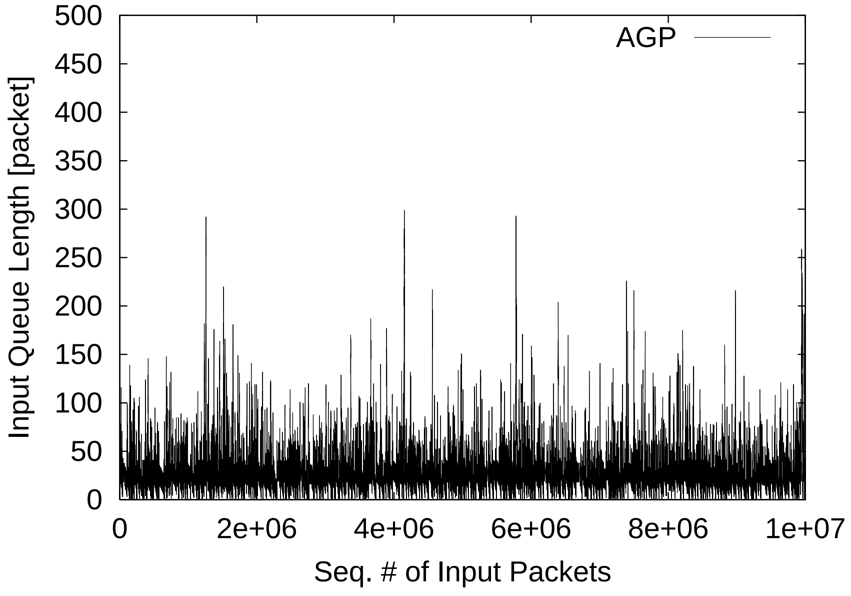


Figure 17: Time-dependent Behavior of Input Queue Length in AGP ($K = 500$, $\rho = 0.94$)

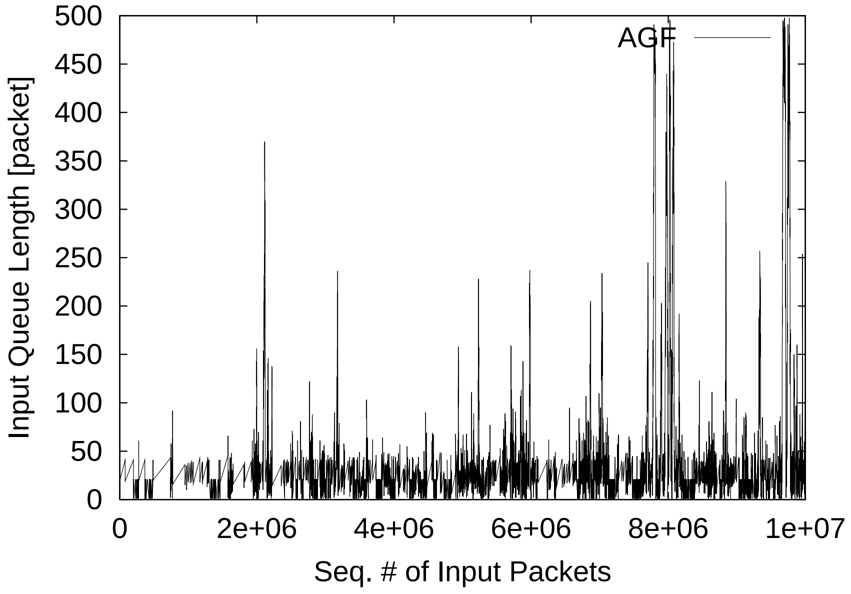


Figure 18: Time-dependent Behavior of Input Queue Length in AGF ($K = 500$, $\rho = 0.94$)

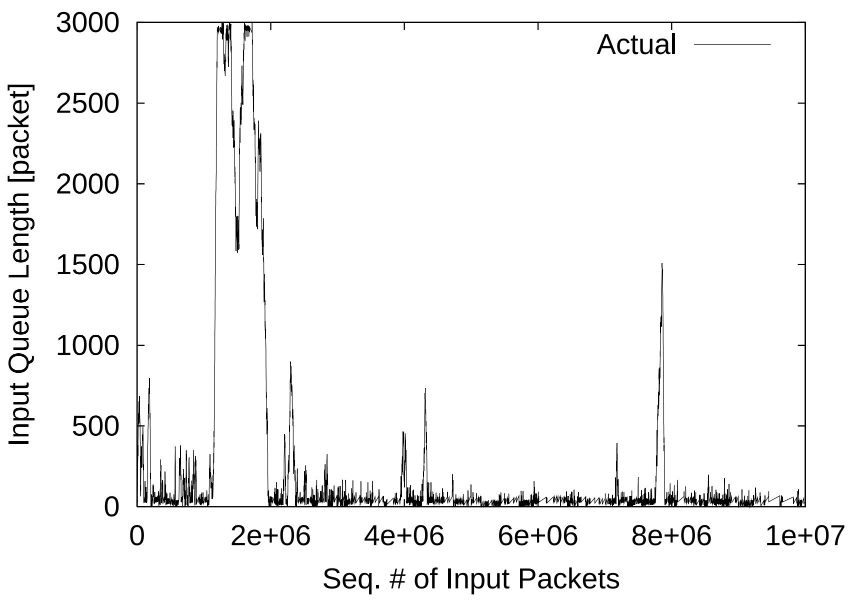


Figure 19: Time-dependent Behavior of Input Queue Length in Actual Traffic ($K = 3000$, $\rho = 0.96$)

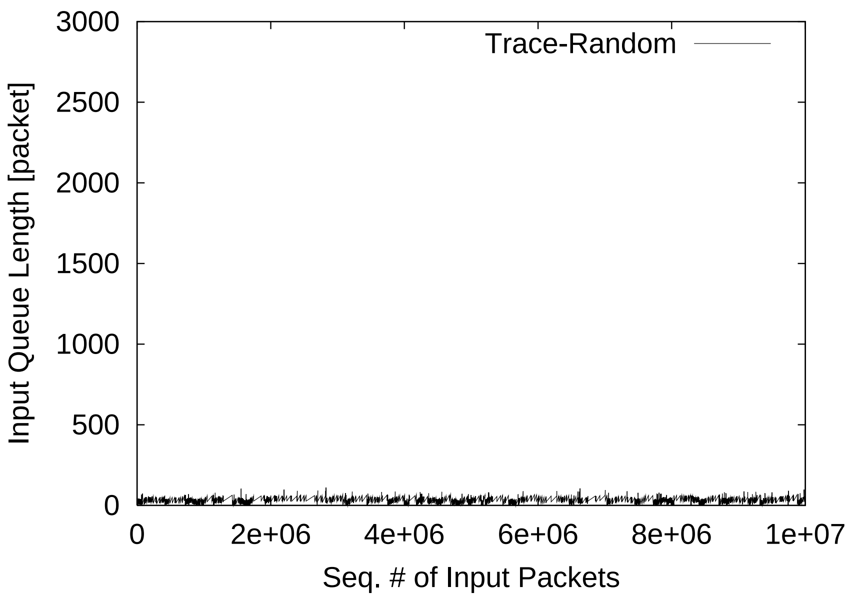


Figure 20: Time-dependent Behavior of Input Queue Length in Trace-Random Traffic ($K = 3000$, $\rho = 0.96$)

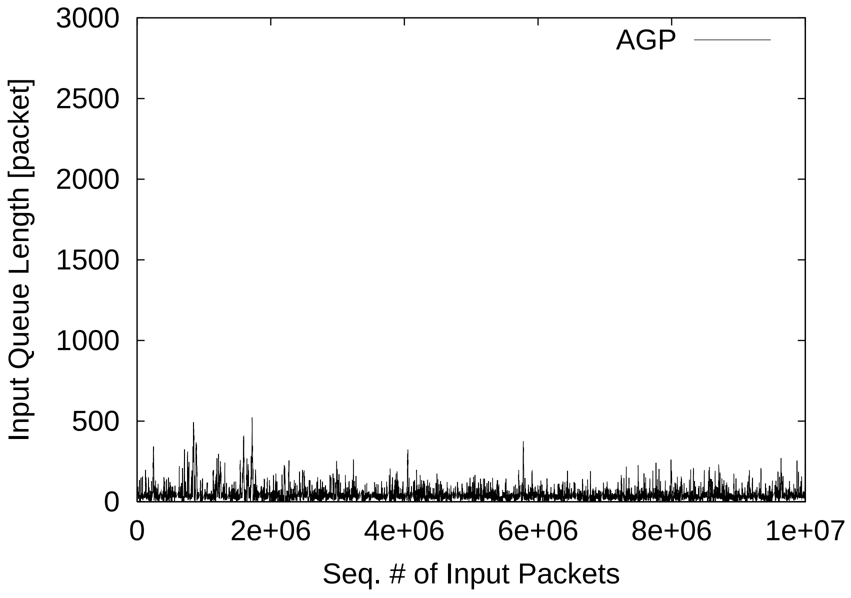


Figure 21: Time-dependent Behavior of Input Queue Length in AGP ($K = 3000$, $\rho = 0.96$)

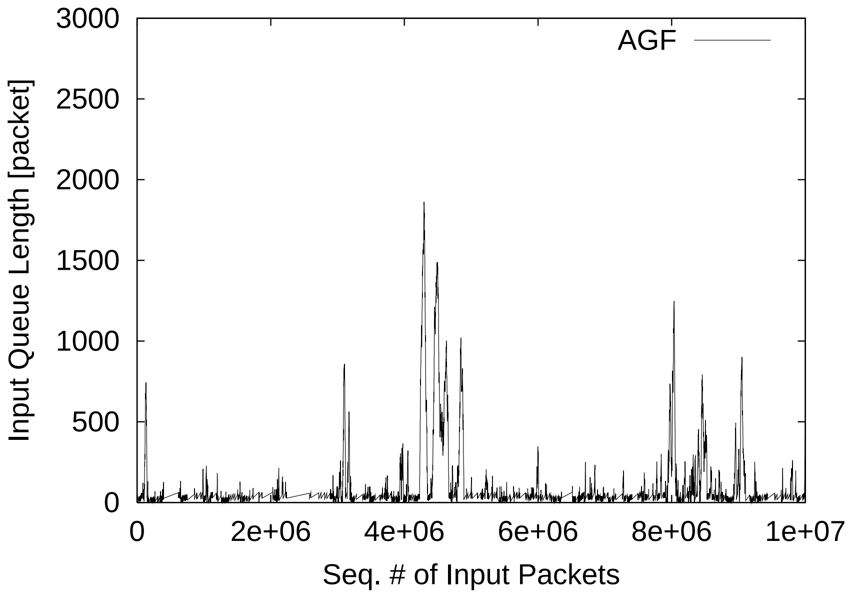


Figure 22: Time-dependent Behavior of Input Queue Length in AGF ($K = 3000$, $\rho = 0.96$)

4 Approximate Mathematical Approach for Quick Performance Prediction

We have shown the performance prediction method of address lookup algorithms through a simulation technique using proposed traffic generation method. Of cource, it requires much CPU processing. If the algorithms can be evaluated by a mathematical analysis, we can know the performance of address lookup algorithms quickly in various network conditions, which is a subject of this section.

4.1 Approximate Analysis Method

In this subsection, we describe the analysis method by using the queueing system [19]. We assume that arrival packets are stored into the buffer and processed by the First-Come-First-Served policy. We also assume that the packet interarrival time can be modeled by a Poisson process. Then, we can apply a $M/G/1$ queueing model, whose service time is given by the address lookup delay distribution. It can be determined from the address lookup algorithm and traced data as having been described in Subsection 2.3. This modeling approach is identical to the simulation method using Trace-Random traffic, which was a reasonable solution for obtaining the average performance such as throughput values as having been shown in Table 1.

We denote λ as the packet arrival rate at the router. Furthermore, S is denoted as the address lookup delay. An average of S (denoted by $E[S]$) is determined according to the probability density of addresses and the corresponding table lookup delays determined by the memory access times. We define the traffic intensity ρ as

$$\rho \quad = \quad \lambda E[S]. \tag{6}$$

The traffic intensity ρ is the quantity that governs the stability of the system, and the maximum throughput can be determined once we obtain the address distribution.

Let us introduce r as the packet processing delay, which is defined as the time duration from when a packet arrives at the router to when the packet is forwarded to the output link. By applying the $M/G/1$ queueing model, the average packet processing delay is derived by

$$E[r] \quad = \quad E[S] + \frac{\lambda E[S^2]}{2(1-\rho)}, \tag{7}$$

where $E[S^2]$ is the second moment about the origin of S.

4.2 Accuracy of Mathematical Approach

We now investigate the accuracy of the average packet processing delay analysis described in the previous subsection. In the previous section, we have shown that AGF provides the good performance predictions when comparing with the "Actual" traffic case. We thus investigate the accuracy of the analysis by comparing with the AGF and "Actual" cases. The delay analysis requires the distribution of the address lookup delay S derived from traced data. In following numerical results, we use the distribution shown in Figure 4(a). For comparison purpose, we also use the distribution derived from randomly generated addresses in Figure 4(b).

Figure 23 compares the simulation and analysis results for the average packet processing delay dependent on the packet arrival rate in the case of the Patricia Tree Search. As shown in this figure, the analysis result using the distribution of address lookup delays given by traced data is closer to simulation results than the analysis result using by random address generation. When the traffic load is not high, we can observe the good agreements between results of simulation and analysis using traced data. However, when the traffic load becomes high, the $M/G/1$ analysis using traced data underestimates the packet processing delay. In the $M/G/1$ queueing model, the address lookup delay for each packet is chosen from the distribution of lookup delays and time–dependent correlation for packet addresses is not considered. Recall that we observe the spikes in Figure 15 in simulation. A more accurate analysis result may be obtained by taking account of those factors. Otherwise, the simulation using AGF is a reasonable approach to predict the performance under

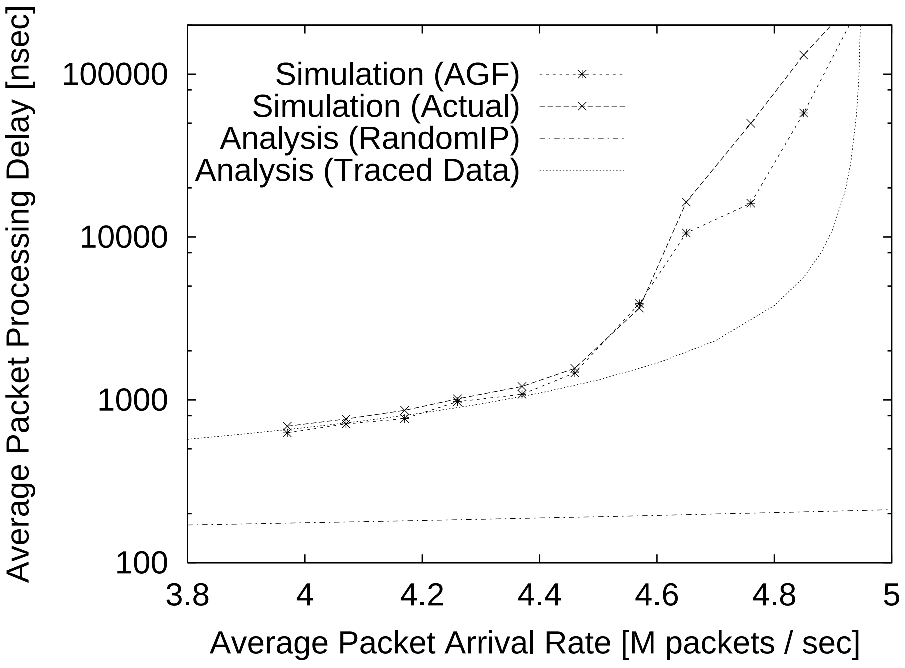


Figure 23: Comparison between Simulation and Analysis Results (Average Packet Processing Delay)

the heavy–loaded condition.

5 Concluding Remarks

In this thesis, we have proposed performance prediction methods of the address lookup algorithms based on the statistical model obtained from the real traffic. Our method consists of the folloing three steps. The first step creates addresses according to the distribution of address lookup delays for destination addresses collected from traced data. The second step generates address sequences by modeling the tendency of packet arrivals using the LRU stack, which is based on ISGF model. We have applied our address generation method using LRU stack to two traffic generation methods. The one is AGP, a simple procedure that assigns an address that LRU stack outputs to each packet. It means that AGP does not consider the flow characteristics of the actual traffic. To overcome this weakness, we have next proposed AGF, applying flow statistics. The third step predicts the performance of address lookup algorithms through simulations using our traffic generation methods. We have used a Patricia Tree Search as the target algorithm. We have investigated the accuracy of performance prediction, and through numerical results, we have shown that AGF can provide good prediction of the actual performance of the address lookup algorithm.

Furthermore, we have discussed the accuracy of the average packet processing delay analysis using the $M/G/1$ queueing model for quick (but rough) performance prediction. We have shown that when the traffic load is not high, we can observe a good agreement between results of simulation using traced data and analysis, but as it is high, the analysis method underestimates the packet processing delay. A more accurate analysis may be able to provide better results, our conclusion in this thesis is that the simulation using AGF is a reasonable approach to predict the performance.

As future research works, an improvement of AGF to avoid underestimating the performance should be investigated.

Acknowledgements

My sincere gratitude goes to my supervisor Professor Masayuki Murata of Osaka University, who has guided and supported me continuously to give me the chance to fulfill my hope of taking a Master's degree in engineering science at Osaka University. This thesis owes much to his thoughtful and helpful advice.

My Special thanks are due to Research Assistant Shingo Ata of Osaka City University, for his helpful suggestions and valuable hints. He has opened the way for the accomplishment of this thesis.

I greatly express my appreciation to Professor Hideo Miyahara, Assistant Professor Naoki Wakamiya, Research Assistant Hiroyuki Ohsaki, Research Assistant Go Hasegawa, and Research Assistant Shin'ichi Arakawa of Osaka University, to Dr. Naoaki Yamanaka and Dr. Kohei Shiomoto of NTT Network Innovation Laboratories, and also to Mr. Masanori Uga of NTT Network Service Systems Laboratories, for their critical comments.

I am grateful to Research Assistant Toyokazu Akiyama of Osaka University for his assist in monitoring the traffic at the gateway of Osaka University.

I am thankful to academic colleagues in our research group, especially Mr. Kazumine Matoba and Mr. Kouhei Fujimoto who have given me much useful advice and encouraged me. Furthermore, I express my gratitude to a number of friends and colleagues in the Department of Informatics and Mathematical Science of Osaka University for their generous help, enlightening and valuable suggestions.

Finally, I wish to thank my parents for their financial and spiritual support.

References

[1] Y. Rekhter and T. Li, “An architecture for IP address with CIDR.” RFC1518, Sept. 1993.

[2] 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.

[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] V. Srinivasan and G. Varghese, “Faster IP lookups using controlled prefix expansion,” in *proceedings of ACM SIGMETRICS ’98*, pp. 1–10, June 1998.

[6] B. W. Lampson, V. Srinivasan, and G. Varghese, “IP lookups using multiway and multicolumn search,” *IEEE/ACM Transactions on Networking*, vol. 7, no. 3, pp. 324–334, 1999.

[7] M. Degermark, A. Brodnik, S. Carlsson, and S. Pink, “Small forwarding tables for fast routing lookups,” in *proceedings of ACM SIGCOMM ’97*, pp. 3–14, Sept. 1997.

[8] F. Ergun, S. Mittra, S. C. Şahinalp, J. Sharp, and R. K. Sinha, “A dynamic lookup scheme for bursty access patterns,” in *proceedings of IEEE INFOCOM 2001*, Apr. 2001.

[9] G. Narlikar and F. Zane, “Performance modeling for fast IP lookups,” in *Proceedings of ACM SIGMETRICS 2001*, pp. 1–12, June 2001.

[10] T. Berners-Lee, R. Fielding, and H. Frystyk, “Hypertext transfer protocol – HTTP/1.0.” RFC1945, May 1996.

[11] J. Postel, “Transmission control protocol specification.” RFC793, Sept. 1981.

[12] 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.

[13] J. E. Shemer and G. Shippey, "Statistical analysis of paged and segmented computer system," *IEEE Trans. Electronic Computers*, vol. EC-15, no. 6, 1966.

[14] D. E. Knuth., *The Art of Computer Programming*, vol. 3. Addison-Wesley, 1973.

[15] S. Deering and R. Hinden, "Internet protocol, version 6 (IPv6) specification." RFC2460, Dec. 1998.

[16] 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.

[17] 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.

[18] 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.

[19] S. S. Lavenberg, ed., *Computer Performance Modeling Handbook*. Academic Press, 1983.

[20] A. McAuley and P. Francis, "Fast routing table lookup using CAMs," in *Proceedings of IEEE INFOCOM '93*, vol. 3, pp. 1382–1391, Mar. 1993.

[21] NLANR, available at `http://www.nlanr.net/`.

[22] P. Gupta, S. Lin, and N. McKeown, "Routing lookups in hardware at memory access speeds," in *Proceedings of IEEE INFOCOM '98*, pp. 1240–1247, Apr. 1998.

[23] R. P.Draves, C. King, S. Venkatachary, and B. D.Zill, “Constructing optimal IP routing tables,” in *Proceedings of IEEE INFOCOM ’99*, vol. 1, pp. 88–97, Mar. 1999.