Performance Prediction Methods for Address Lookup Algorithms of IP Routers

Ryo Kawabe[1], Shingo Ata[2], and Masayuki Murata[3]

[1] Graduate School of Engineering Science, Osaka University
1–3 Machikaneyama, Toyonaka, Osaka 567–8531, Japan
r-kawabe@ics.es.osaka-u.ac.jp
[2] Graduate School of Engineering, Osaka City University
3–3–138 Sugimoto, Sumiyoshi-ku, Osaka 558–8585, Japan
ata@info.eng.osaka-cu.ac.jp
[3] Cybermedia Center, Osaka University
1–30 Machikaneyama, Toyonaka, Osaka 567–0043, Japan
murata@cmc.osaka-u.ac.jp

Abstract. Many address lookup methods for an IP router have been recently proposed to improve a packet forwarding capability, but their performance prediction is very limited because of lack of considering actual traffic characteristics. It is necessary to consider actual traffic to predict more realistic performances on routers, specially in case of layers 3 and 4 switches whose performances are more influenced by flow characteristics. In this paper, we propose new methods for predicting the router's performance based on the statistical analysis of the Internet traffic. We also present an example of its application to the existing table lookup algorithms, and show that simulation results based on our method can provide accurate performance prediction.

1 Introduction

A rapid growth of the Internet traffic with the spread of multimedia applications such as streaming media leads to an explosive demand of high-speed packet transmission technologies. For this purpose, it is necessary to improve the packet forwarding capability at IP routers as well as to increase the link capacity. One main task of routers is to determine the output interface for forwarding the arriving packet according to the packet header information, e.g., the destination address. Other information such as the source address, source/destination port numbers, and protocol number may also be used for policy routing and/or the layer 4 switching. IP routers perform the following two steps for every arriving packet.

1. Look up the next-hop of the packet from the routing table.
2. Forward the packet to the outgoing interface determined by step 1.

Step 1 heavily influences the performance of routers because the longest prefix matching [1] in Step 1 is complicated after CIDR (Classless Inter-Domain Routing) [2] is introduced. Therefore, an address lookup method easily becomes a performance bottleneck at high-speed routers.

To overcome the performance bottleneck of address lookup, many approaches have been proposed in recent years. See, e.g., [3, 4]. However, performance studies on their proposed methods are very limited. Roughly speaking, there are two metrics for the performance evaluation of the address lookup algorithms; the worst-case and the average-case (or actual-case). The worst-case performance is easy to derive from the complexities of lookup algorithms, and by using them, one can easily compare the proposed algorithm with other existing algorithms. The worst-case performance is also a useful index to describe its basic capability. Most of papers on the longest prefix matching thus use the worst-case performance because of above reasons.

However, the actual performance is heavily affected by the time–dependent behavior of destination addresses of arriving packets. Furthermore, the worst-case performance is not always the best metric. It is likely that the design choice according to the worst-case performance leads to very expensive algorithms. A closer look at the performance behavior of the target method may allow a more elegant solution. For example, we may be able to have a much cheaper solution at the sacrifice of limited performance degradation (e.g., introducing a small packet loss probability). For that purpose, we need a realistic address generation method for evaluating the performance of IP routers while the past researches only considered the simulation technique using the random address generation [3]. Or, a small amount of traced data is used [4]. Since the amount of the traced data is limited, the actual performance behavior is likely to be missed. Furthermore, because traced data is very few in the public domain, the simulation result is lack of generality.

In this paper, we propose performance prediction methods of address lookup algorithms with consideration of characteristics of the actual traffic. We first analyze the behavior of the address generation pattern based on the traced data in Section 2. It is then used to generate the pseudo traffic using the mathematical model. Results of simulation experiments are presented in Section 3 to compare our proposed method with the trace–driven simulation method and we show that it is possible to predict more realistic performance using our method. In Section 4, we conclude our paper with future research works.

2 Statistical Address Generation Methods

In this section, we propose two performance prediction methods of address lookup algorithms of IP routers. The one is for generating the destination address of IP packets, i.e., for layer 3 routers. The other is for generating the flow, where the flow duration (the number of packets contained in the flow) is obtained by the statistical analysis of traced-data. Then, the destination address of the flow is generated by the statistical model. See Figure 1 for an outline of our methods.

We will first introduce the model of the destination addresses in Subsection 2.1. The address generation method is then described in Subsection 2.2. It is applied to the traffic generation method, which can be used for simulating the table lookup algorithms.

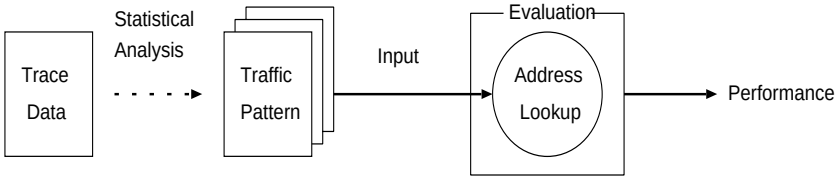


Fig. 1. Outline of Performance Prediction

2.1 Modeling Destination Addresses

We use the approach of the address generation algorithm proposed in [5], where the addresses are generated by a LRU (Least Recently Used) stack and ISGF (Inverse Stack Growth Function).

For briefly summarizing ISGF, we introduce t_i, denoting the arrival time of ith packet (or flow). We define $f(t, T)$ as the expected value of the number of distinct addresses of packets arrived during a period $(t - T, t)$. Assume that $f(t, T)$ is independent from time t, i.e., $f(t_i, T) = f(t_j, T)$ for all i, j. Then, $f(t, T)$ can be denoted by $f(T)$. This assumption is called a time–translation invariance, and makes it for us to obtain the same value $f(T)$ whenever trace data are gathered. ISGF is a power low function given by

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

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

While ISGF was originally used for predicting the cache hit ratio of computer architectures, the authors in [5] have found that the number of distinct destination addresses also satisfies ISGF. Namely, when we collect T packets from the traced data, the number of distinct destination addresses can be estimated as T^{α}. By using ISGF, it is possible to derive the probability that the destination address of the arriving packet has already appeared. However, applying only ISGF is insufficient in some situations.

Let us consider the sequence of packet arrivals for the TCP flow. Because the TCP flow is divided into packets, the probability that the next packet of the TCP flow arrives at the router tends to be decreased as the time passes. This tendency cannot be modeled by ISGF solely. Therefore, an LRU (Least Recently Used) stack model is applied to the probability structure as follows.

The probability a_i is defined as

$$\begin{aligned} a_i = &\{f(g(i-1)+1) - (i-1)\} \\ &-\{f(g(i)+1) - i\}, \end{aligned} \tag{2}$$

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

$$\begin{aligned} a_i = &\{((i-1)^{\frac{1}{\alpha}} + 1)^{\alpha} - (i-1)\} \\ &-\{(i^{\frac{1}{\alpha}} + 1)^{\alpha} - i\}. \end{aligned} \tag{3}$$

2.2 Address Generation Method

In this subsection, we explain the procedure of destination address generation based on ISGF. We prepare the set of distinct destination addresses of trace data with their numbers of references from the traced data. Then, if N_p packets include N_a distinct destination addresses, α is calculated from Eq.(1). Since Eq.(1) holds when N_p is large, we need to calculate α with the adequately large number of packets. We use a least–square method to calculate α.

Moreover, we define a list of destination addresses as $A(i)$ $(i = 1, 2, \ldots, N_a)$. The following procedure generates time–dependent destination addresses using the above quantities; N_p, N_a, α and $A(i)$.

1. LRU stack size S is set to N_a.
2. Store $A(1), A(2), \ldots, A(N_a)$ sequentially in the LRU stack.
3. Choose a random number p $(0 \leq p < 1)$.
4. Calculate the minimum k if $p \leq \sum_{i=1}^{k} a_i$.
5. Assign the address of a new access as k-th element of the LRU stack.
6. Move k-th element to the top of the LRU stack.
7. Return to Step 3.

The above procedure gives a series of destination addresses, which can be embedded in the simulation program for packet generation. More specifically, either of the following procedures is performed as a packet generator in the simulation.

- Address Generation per Packet (AGP)
  1. A packet is generated according to, e.g., a Poisson process. A heavy-tailed distribution may also be applied to determine the interarrival times of packets.
  2. The destination address of the generated packet is decided by the address generation method described above.
- Address Generation per Flow (AGF)
  1. Generates a flow, and determines the number of packets in the flow.
  2. The destination address of the flow is decided by the address generation method mentioned above.
  3. Determines the interarrival times of packets during which the flow is active. All the packets in the flow use the same destination address.

We note here that in AGF, the number of packets within the flow may be determined from the statistical analysis of the actual data. For example, in [6], the entire flow duration can be well approximated by the log-normal distribution while the tail–part has a heavy-tailed distribution. Thus, the combination of the two distributions can be used for the flow duration and is used in our experiments. To fit the combination of the two distribution function accurately, we estimate parameters using the maximum-likelihood-estimator (MLE) method [7]. Once the flow is accurately characterized, the packet interarrival times may be modeled by a Poisson distribution [6].

On the other hand, AGP cannot model the heavy-tailed distribution for representing the number of flows because the probability that the generated

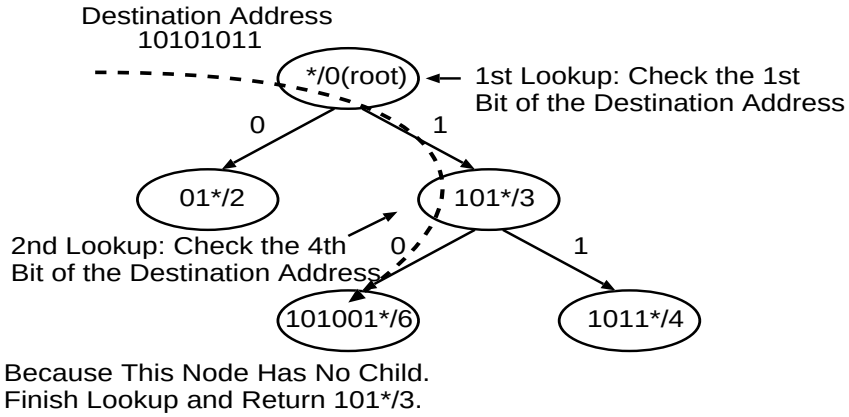


Fig. 2. An Example of the Patricia Tree Search

address would appear again in the future depends only on the position of the LRU stack, not the number of references. However, AGP has some advantages. It requires lower processing overhead and the smaller number of parameters than AGF.

3 Performance Evaluation

In this section, we evaluate the performance of existing address lookup algorithms by using our proposed methods. We use the Patricia Tree search [8] and the pointer cache method proposed in [3].

3.1 Target Algorithms

We briefly summarize the Patricia Tree and the pointer cache method in this subsection. At legacy routers, the longest prefix matching has been performed by the binary tree. The Patricia Tree search [8] is a popular one, which eliminates the node having only one child node. See Figure 2 for an example of the Patricia Tree search. In this case, the nodes for the fourth and fifth bit of the destination address are removed because the routing table has only the entries beginning with 101001 when the fourth bit of the destination address is 0. The Patricia Tree search is simple and hence easy to implement, but relatively slow because the number of removable nodes becomes decreased when the number of entries is increased. In the worst case, the Patricia Tree still requires up to 32 or 128 memory references in IPv4 or IPv6, respectively.

To resolve the problem, several algorithms have been proposed by using the CAM (Content Addressable Memory), which can decrease the number of memory accesses (see [9] and references therein). The pointer cache method [3] is one

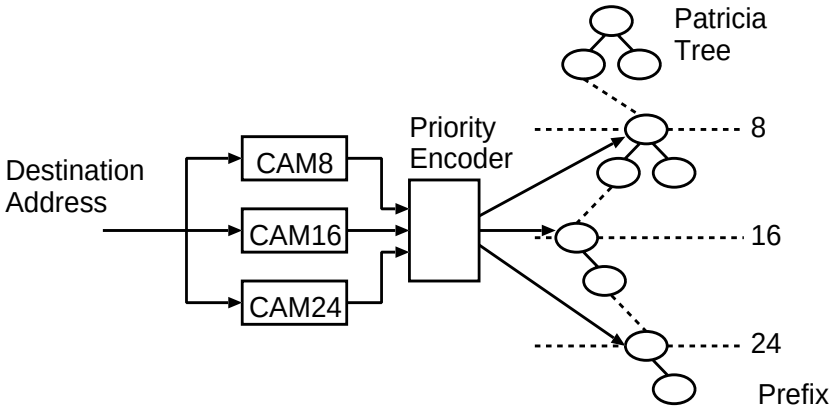


Fig. 3. Composition of Pointer Cache Method

of such CAM-based algorithms, combining CAM and the Patricia Tree. Figure 3 shows its composition. The pointer cache method first divides the Patricia Tree into some parts by borders with predefined values as a prefix length of the Patricia Tree. Then the set of pointers to the root node of each part is stored into CAM. The address lookup for the arriving packet is performed by the following procedure;

1. Search the CAM for the pointer to the objective part which includes the longest prefix matched entry for the destination address.
2. Look up the address from the part of Patricia Tree.

Because CAM is used like a cache memory for the Patricia Tree, the performance of the pointer cache method depends on entries stored in CAM, which must be strongly affected by the traffic characteristics such as the distribution of addresses of arriving packets. This is why we choose the pointer cache method as an example.

3.2 Traffic Models

Before evaluating the performance, we need to determine the parameters N_a, N_d, and $A(i)$ from the traced data. We use one million packet headers gathered by the traffic monitor (OC3MON [10]) at the gateway of Osaka University. Traced data set includes 6,966 distinct destination addresses. Its BGP information (4,669 entries) are used to construct the Patricia Tree and CAM entries in the pointer cache method. The data is collected on January 24th, 2001. The values α of ISGF are 0.64 in AGP and 0.77 in AGF, respectively. For comparison purpose, we also consider the following three kinds of address generation methods.

Actual: A raw sequence of packet headers obtained by OC3MON is used.

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

Trace-Random: Randomly pick up the destination address of the packet from the traced data.

3.3 Performance Metrics

In our experiments, we generate ten million packets according to the Poisson process for the input to address lookup algorithms. Their destination addresses are generated by five traffic patern mentioned above. "Trace-Random" traffic case, AGP and AGF uses one million packet headers of the traced data. The buffer size of the router is set to be 3,000 packets.

We define an address lookup delay S as the time required for a longest prefix matching of an arrival packet. In the Patricia Tree search, S is given by

$$S = t_s \times d_r, \tag{4}$$

where t_s is the number of times of lookup in Patricia Tree and d_r is the delay for read / comparison operations in RAM. Under the present circumstance, since a read requires 5 nsec and a comparison requires 10 nsec in RAM, the value of d_r can be assumed to be 15 nsec. In the case of Figure 2, since two lookups are required to determine the longest prefix entry of the packet, the value t_s is 2 and then the value S is estimated as 30 nsec. An address lookup by using the pointer cache method consists of a search for CAM and a lookup in the Patricia Tree. Since an access to CAM is performed independently of an access to the Patricia Tree, it can be improved by performing in parallel on the pipeline. An arriving packet at the router first searches the longest prefix entry among all entries stored in CAM. After the search has finished, the lookup of the packet begins if no packets are being looked up in the Patricia Tree. Otherwise, it is kept waiting for finishing the lookup in the Patricia Tree and a search of a newly arriving packet for CAM does not start. In the present situation, the search for CAM requires 15 nsec [11–13] and we used this value in our experiments.

We use the maximum throughput as the performance metric, which is the reciprocal of the minimum average of packet interarrival time if no packet is lost during the simulation. We also obtain the behavior of the time–dependent queue length (the number of packets queued in the buffer).

3.4 Simulation Results

Patricia Tree Table 1 compares simulation results of the Patricia Tree search among five traffic patterns (AGP, AGF and three packet generation methods described above). In Table 1, the second column is the maximum throughput and the third column is the relative error ratio to the maximum throughput of the "Actual" case. The table shows that the throughput of "Random" case is 1.8 times larger than the one of the "Actual" case. It is due to the characteristics of the destination address distribution. In the Patricia Tree search, the packet

Table 1. Maximum Throughput (Patricia Tree Search)

Input Traffic	Maximum Throughput	Error Rate
Actual	4.63 mpps	—
Random	8.33 mpps	79.9%
Trace-Random	4.67 mpps	0.86%
AGP	4.63 mpps	0.00%
AGF	4.52 mpps	2.38%

lookup delay is decided by the prefix length of the matched entry in the routing table. If the number of packets whose destination addresses match the longer-prefix entries of the routing table, the average lookup delay becomes larger. Furthermore, the prefix length of the entry shows the size of the organization belonging to it. 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`, packets tend to access the destination address space which includes more `www` servers. From the above reason, the "Actual" traffic case tend to match the longer prefix entries, and the address lookup delay becomes larger. On the other hand, the "Random" pattern generates uniformly–distributed addresses; the number of accesses of the entry is inversely proportional to the prefix length. Namely, in "Random" case, the entry with the short prefix length is preferably accessed. Therefore, "Random" case overestimates the performance of Patricia Tree significantly, when compared with "Actual" case. In contrast to "Random" case, results of "Trace-Random" case, AGF, and AGP provide good estimations with low errors. Among three methods, AGP provides the best prediction with respect to the throughput. Note that since the result of "Random" case is far from other traffic patterns, we will not show the result of "Random" case in the below.

Pointer Cache Method To see the applicability of our proposed methods to other algorithms, we next examine the pointer cache method [3]. Table 2 compares the maximum throughput. We can observe that the results are almost same as the case of the Patricia Tree search, and our proposed methods can provide good prediction.

We also show the behavior of the time-dependent queue length in Figure 4. From Figure 4(a), it is observed that the behavior of "Actual" case has two properties. Its fluctuation is low (below 50 packets). However, the queue length sometimes increases significantly. It is caused by the characteristic of packet arrivals. Due to the window flow control of TCP, traffic of the TCP connection contains a burst of packets. Of cource, the continuos arrival of packet belonging to the same flow is another reason. Then, a significant increase (called as *spike* below) appears when the packets with the long-prefix-matched address arrive bursty. On the other hand, other three methods show different results. In "Trace-Random" case (Figure 4(b)), any spike does not appear during the simulation.

Table 2. Maximum Throughput (Pointer Cache Method)

Input Traffic	Maximum Throughput	Error Rate
Actual	35.2 mpps	—
Random	58.8 mpps	67.0%
Trace-Random	36.5 mpps	3.69%
AGP	35.0 mpps	0.57%
AGF	35.0 mpps	0.57%

The behavior of AGP (Figure 4(c)) shows many spikes, but its whole fluctuation is not so low, compared with "Actual" case. On the other hand, the result of AGF (Figure 4(d)) shows some spike and its whole fluctuation is low, and its behavior of the time–dependent queue length is similar to the one of the "Actual" case. However, the overall queue length of AGF is longer than the one of "Actual" case. This is because our proposed method cannot generate a new address (i.e., it does not appear in the traced data) that will arrive, for "Trace-Random" traffic, AGP and AGF are based on 6,966 distinct addresses gathered from one million traced packet headers, while "Actual" traffic uses ten million traced packet headers.

This is the open issue of our generation methods that does not consider the new address generation. One solution may be to set the index for each entry of the routing table based on the number of accesses, and to generate the new address according to the index of the entry. A proposal of new address generation algorithm is one of our future research topics.

Recall that AGF explicitly models the flow characteristics (i.e., the number of packets in the flow, interarrival times of flows, and the number of active flows). Thus, AGF can be applied to flow classification algorithms (e.g., [14]) as well as the address lookup algorithms. Its accuracy should be investigated as a future research topic.

4 Conclusion

In this paper, we have proposed performance evaluation methods of the address lookup algorithms based on the statistical model obtained from the real traffic. We have found out that through a packet driven simulation, we can evaluate them accurately with our proposed method, AGF.

As a future topic, we are now developing the analysis approach to know the performance of address lookup algorithms more quickly. Also, we need to validate the applicability of AGF to the packet classification algorithm.

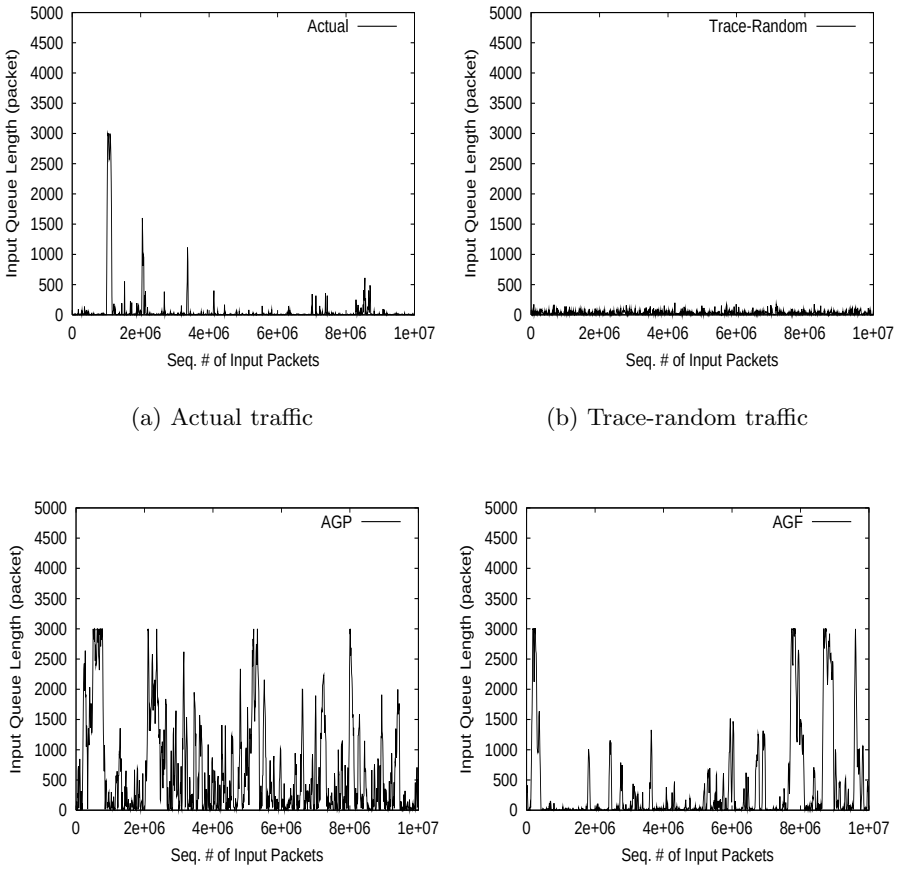
Acknowledgements

We would like to thank Dr. Naoaki Yamanaka and Dr. Kohei Shiomoto of NTT Network Innovation Laboratories, and Mr. Masanori Uga of NTT Network Service Systems Laboratories for their comments and suggestions.

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. 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.
6. 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.
7. 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.
8. D. E. Knuth., *The Art of Computer Programming*, vol. 3. Addison-Wesley, 1973.
9. A. McAuley and P. Francis, "Fast routing table lookup using CAMs," in *Proceedings of IEEE INFOCOM'93*, vol. 3, pp. 1382–1391, Mar. 1993.
10. 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.
11. Lara Network, `http://www.laratech.com/`.
12. NetLogic Microsystems, `http://www.netlogicmicro.com/` .
13. Siber Core, `http://www.sibercore.com/` .
14. P. Warkhede, S. Suri, and G. Varghese, "Fast packet classification for two-dimensional conflict-free filters," in *Proceedings of IEEE INFOCOM 2001*, Apr. 2001.



(a) Actual traffic

(b) Trace-random traffic

(c) AGP

(d) AGF

Fig. 4. Transition of Input Queue Length (Pointer Cache Method)