Stateful Inspection Firewall Session Table Processing

Xin Li, ZhenZhou Ji, and MingZeng Hu

School of Computer Science and Technology
Harbin Institute of Technology
92 West Da Zhi St. Harbin, China
{lixin, jzz, mzhu}@pact518.hit.edu.cn

Abstract

Stateful Inspection is a key technology to network devices such as routers and firewalls. Existing session table architectures of Stateful Inspection devices store all session information in a single entry, which causes high time cost of session table timeout processing. In this paper we present a new architecture which divides a session entry into two parts, and designs different data structures for each other. The new architecture can improve the performance of session table greatly. A new PATRICIA algorithm is proposed to organize session table, which is proved to be an optimal 2-ary trie for fixed-length match. An ASIC is implemented for the architecture and corresponding algorithms. Both theoretical and experimental results show that the new architecture has better performance than existing architectures, and can work well in Gigabit Ethernet network.

Keyword: PATRICIA, stateful inspection, session table, timeout processing

I. Introduction

Stateful Inspection refers to an extension of packet-by-packet filtering process that tracks individual flows, enabling policy checks that extend across series of packets[1]. Many firewalls have implemented Stateful Inspection technology, such as Cisico PIX[2], 3COM Secure Gateway [3], Netsreen Firewall[1] and Checkpoint FW-1[4]. Stateful Inspection requires a session table whose entries typically record source and destination IP addresses and port numbers. For each arriving packet, the session table is looked up for a match. A session entry in the format <src-addr, src-port, dst-addr, dst-port, ip-p, state, time> is created when the first packet appears from a flow previously not tracked. Subsequent packets of an established session are checked against the session table rather than against the Rule Base.

The performance of Stateful Inspection firewall mainly depends on the performance of processing session table. Eexisted session table architectures have bad timeout processing performance.

This paper proposed a new architecture for session table, which improves firewall's performance greatly. An ASIC is implemented for the proposed architecture and corresponding algorithms, which can improve firewall's performance further.This document is a template for papers submitted to International Journal of Information Technology. If your paper is intended to this journal, please observe this format. Do not change the fonts or line spacing to squeeze more text into a limited number of pages.

II. Session Table Processing

Generally, the <src-addr, src-port, dst-addr, dst-port, ip-p> is used to identify a unique session, which is called SID in this paper. For each arriving packets, session table is looked up for a match. If packets belong to an existed session, session state and session time will be updated. If a session entry has overtime, it will be deleted to minimize security holes.

A. Session Table Operations

Generally, there are 4 kinds of session table operation: match session table with packets' SID, update an entry's state and time, insert a new entry and delete an overtime entry. Because an update operation is always after a match operation, we call them a match-and-update operation. Required time for inserting an entry (T_{ins}), deleting an entry (T_{del}), and match-and-update(T_{mau}) are three key parameters to the performance of Stateful Inspection firewalls. Because memory access is most time-consuming, we use the number of memory accesses to measure T_{ins}, T_{del}, T_{mau} in this paper.

B. Processing Methods of Session Table

Now all existed firewalls put both SID and <state, time> in a single entry, as Table 1. Because the number of entries may up to 1 million, and each entry is wider than 128 bits, over 128Mb memory space is required for session table. Generally, DDR SDRAM is used to store session table.

Table 1. General format of session table entry

src-addr	src-port	dst-addr	dst-port	ip-p	state	time

C. Full Match Algorithm in IBM NP4GS

Generally, PATRICIA trie is typically used for session table. PATRICIA trie is time-saving for match-and-update and insertion, but traversing PATRICIA trie to process timeout is time-consuming. In order to improve traversing performance, leaves should be linked to a rope. For example, Figure 1 illustrates the data structure in IBM NP4GS3[5] for fixed-length match. All leaves in the PATRICIA trie are linked as a rope in inserted order.

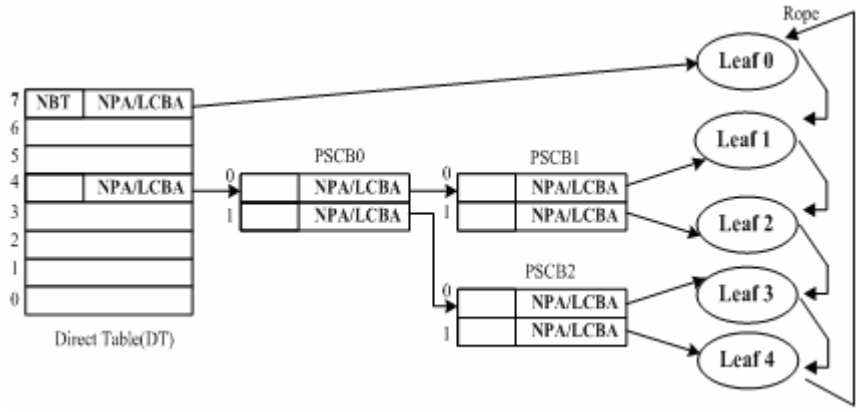


Figure 1. Data Structrue of IBM NP4GS3's FM

T_{search}, T_{insert} and T_{delete} are required time of searching, inserting and deleting an entry in PATRICIA trie respectively. Following functions has considered the time cost of applying and reclaiming memory block. The R_{ot} is the ratio of the number of overtime entries to all entries.

$T_{mau(1)} = T_{search} + T_{(DDR)}$
$T_{ins(1)} = T_{insert} + 2T_{(DDR)}$
$T_{del(1)} = T_{delete} + T_{(DDR)} / R_{ot}$

R_{ot} generally is very small, then timeout processing is time-consuming. When processing many overtime entries once, the time cost of timeout processing is intolerable.

D. *Doubly linked list structure*

In order to improve timeout processing performance, the rope should be linked in time order. If the rope is linked as a doubly linked list in time order, we need not read many entries when processing timeout. When an entry does not timeout, all subsequent entries do not timeout too. Figure 2 shows the data structure.

This architecture can improve the performance of timeout processing effectively, but it causes the performance of match-and-update down heavily. The main reason is that it needs too many memory accesses when updating session table. It needs 7 memory accesses at least, and $T_{(DDR)}$ needs many cycles.

$T_{mau(2)} = T_{search} + 7T_{(DDR)}$
$T_{ins(2)} = T_{insert} + 2T_{(DDR)}$
$T_{del(2)} = T_{delete} + T_{(DDR)}$

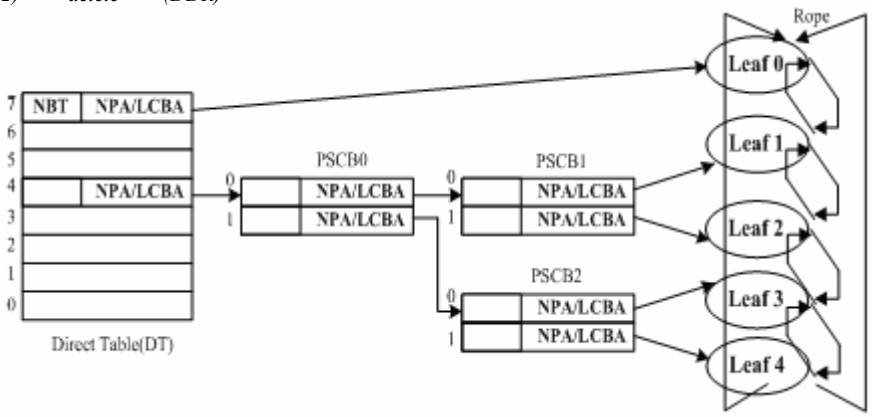


Figure 2. PATRICIA based doubly linked list

E. *Proposed architecture*

Higher speed memory than DDR can improve the performance of match-and-update. In this paper, we proposed a new architecture for session table. We use 2 kinds of memory to store session table, which can improve the performance of session table greatly. Because SID is wide and session table generally have too many entries, we uses DDR SDRAM to store SID*s*. SID*s* are organized in PATRICIA trie.

In order to decrease $T_{del(1)}$, we use a doubly linked list to organize <state, time>. Figure 3 shows its data structure. ZBT SRAM is used to store the doubly linked list. We use *DS1_addr* and *DS2_addr* to relate the two data structures, as shown in Table 2 and Table 3.

Following functions measure the performance parameters of the proposed architecture.

$T_{mau(3)} = T_{search} + 7T_{(ZBT)}$
$T_{ins(3)} = T_{insert} + 3T_{(ZBT)}$
$T_{del(3)} = T_{delete} + 2T_{(ZBT)}$

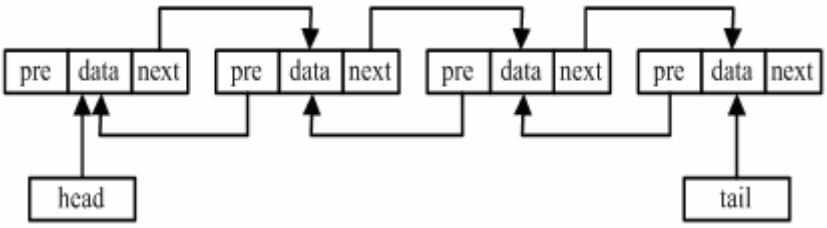


Figure 3. Doubly linked lists for DS2

Table 2. Leaf entry of PATRICIA(DS1)

SID	*DS2-addr*

Table 3. Entry of doubly linked list(DS2)

pre	*state*	*time*	*DS1-addr*	*next*

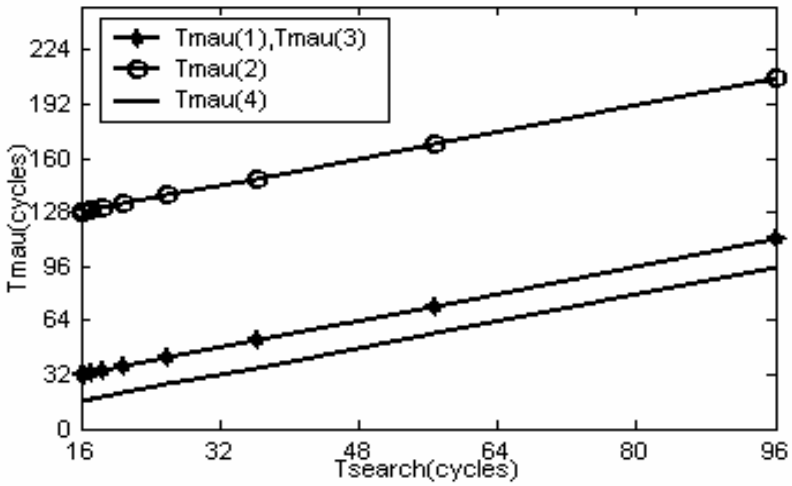
F. *Pipeline the proposed architecture*

T_{search}, T_{insert} and T_{delete} only involve DDR SDRAM. Because $T_{(DDR)}$ needs about 16 cycles for 32 bits memory data bus and 256 bits data, T_{search}, T_{insert} and T_{delete} at least need 32 cycles. $T_{(ZBT)}$ only involves ZBT SRAM, and $T_{(ZBT)}$ only need 2 cycles for 36 bits memory data bus and 72 bits data. So, $7T_{(ZBT)}<T_{search}$, $3T_{(ZBT)}<T_{insert}$ and $2T_{(ZBT)}<T_{delete}$. If we pipeline DS1 processing and DS2 processing, the performance of session table will only be determined by T_{search}, T_{insert} and T_{delete}. The following functions show the performance parameters of the pipelined architecture.

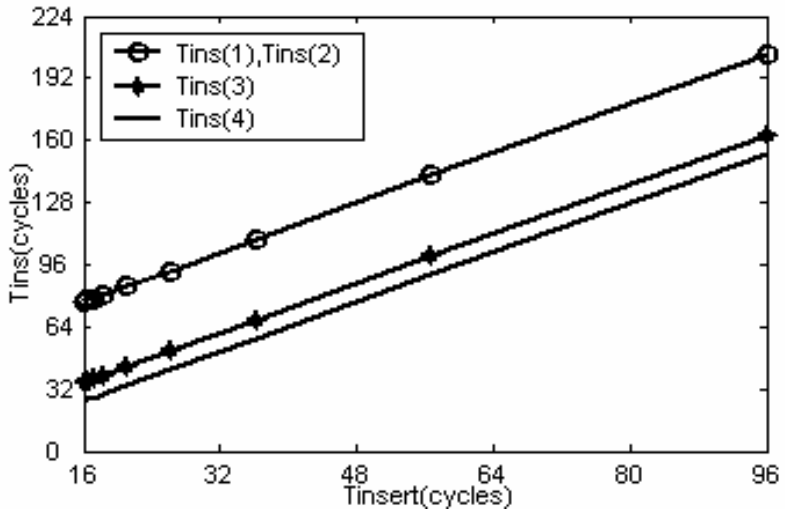
$T_{mau(4)}=T_{search}$
$T_{ins(4)}=T_{insert}$
$T_{del(4)}=T_{delete}$

G. *Performance analysis*

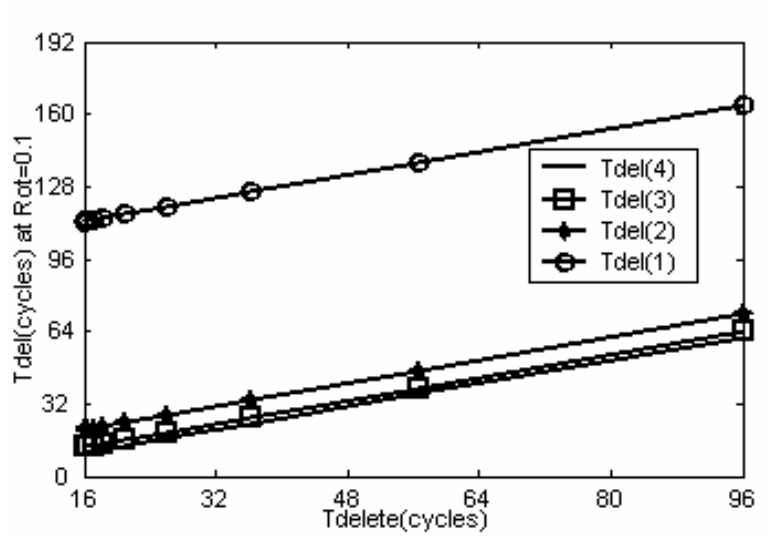
In our design, $T_{(DDR)}$ equals 16 cycles and $T_{(ZBT)}$ equals to 2 cycles, Figure 4 shows the performance comparison of above architectures. $T_{del(4)}$ is much less than $T_{del(1)}$, and $T_{mau(4)}$ is much less than $T_{mau(2)}$. So, the pipelined architecture is the best architecture of the above four architectures.



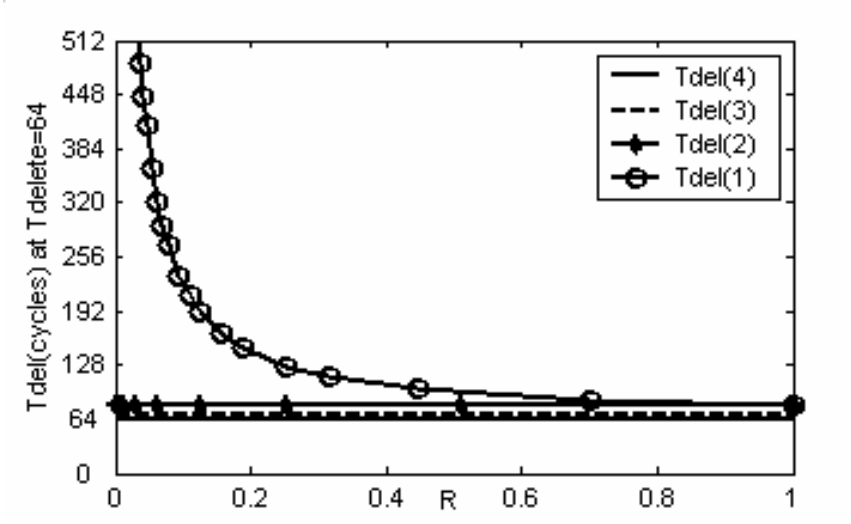
(a) Match-and-update performances



(b) Insertion performances



(c) Deletion performances at R_{ot}=0.1



(d) Deletion performances when T_{delete}=64

Figure 4. Performances comparison

III. A New PATRICIA for Fixed-length Match

From above analysis, we know that T_{search}, T_{insert} and T_{delete} determine session table's performance. So to minimize T_{search}, T_{insert} and T_{delete} is the most important thing.

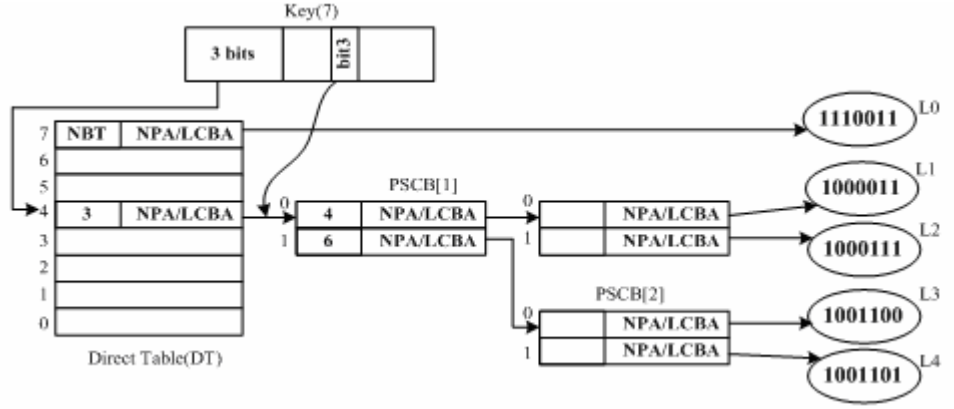
Because trie search needn't compare the whole key, trie is very useful for extremely long key. Because SID is about 128 bits long, trie is more suitable for session table than tree. Because PATRICIA trie compresses all nodes which have one-way branch, it is the lowest trie[6]. If PATRICIA trie is L indepth, PATRICIA search and deletion only need L memory accesses, and PATRICIA insertion need L+1~2L memory accesses which is 1.5L on average. The performances of PATRICIA search and deletion are optimal, but their insertion performance is not optimal.

By analyzing PATRICIA trie, we found that for fixed-length match the performance of PATRICIA insertion can be improved to optimum. The insertion operation can only need L memory accesses. The new PATRICIA algorithm is called PAT-FM in this paper.

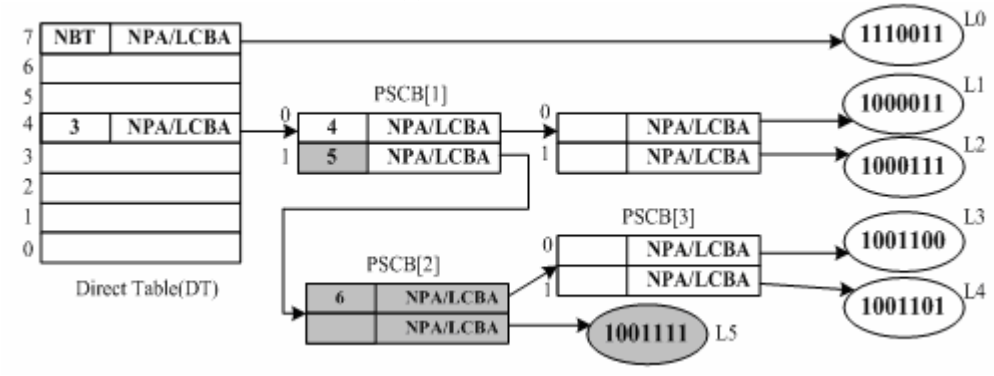
A. PATRICIA insertion

Algorithm 1. PATRICIA insertion[5] .
(1) Search PATRICIA trie, read out the leaf node.
(2) Compare leaf with key, calculate the *DISP* value.
(3) If not equal, search PATRICIA trie again.
(4) Insert the key between two neighbor nodes where the *DISP* value is greater than the *NBT* value of one and smaller than the *NBT* value of the other.

Figure 5 shows PATRICIA insertion process. Along a search path from left to right, that is to say, from a root to a leaf, *NBT* values are increased.



(a) PATRICIA trie



(b) PATRICIA insertion

Figure 5. IBM NP4GS3's PATRICIA for FM[5]

B. *PAT-FM insertion*

By analyzing PATRICIA, we find that prefix match need ordered *NBT* values, but fixed-length match needn't ordered *NBT* values, and disordered *NBT* values can improve the insertion performance of fixed-length match.

Algorithm 2. PAT-FM insertion algorithm

(1) Search PATRICIA trie, read out the leaf node.

(2) Compare the leaf with the key, calculate *DISP*.

(3) Insert the new node at the tail of search path.

Figure 6 shows the insertion process of PAT-FM.

After get a leaf and the *DISP* value, insert the new node directly at the tail of the search path, which needn't search PATRICIA trie again. This insert algorithm results that *NBT* values in the search path is disordered.

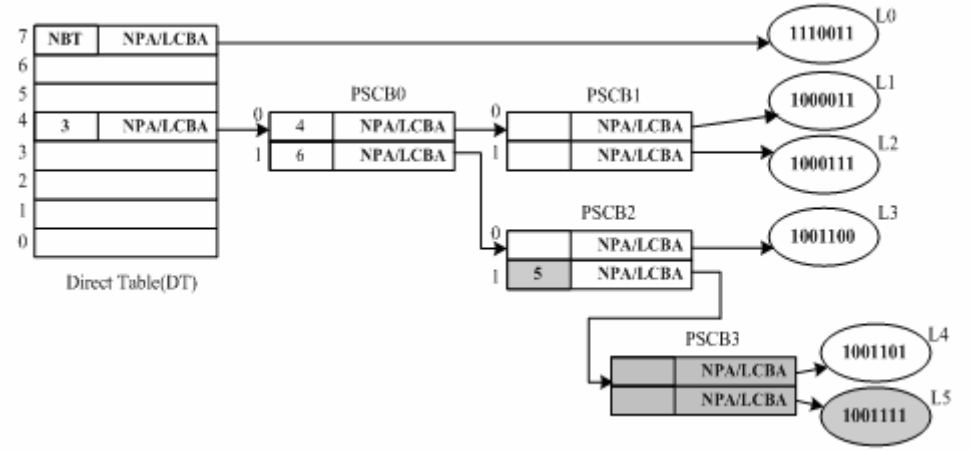


Figure 6. PAT-FM insertion

C. *Performance analysis*

Considering PATRICIA trie and PAT-FM trie which both their average depth are *L*, their time parameters are shown in Table 2, their time complicities are measured with the number of memory accesses.

Table 4. The numbers of memory accesses

Algorithms	Search	Insert	Delete
PATRICIA	*L*	*L*+1~ 2*L*	*L*
PAT-FM	*L*	*L*	*L*

From Table 4, we know that PAT-FM insertion is more time-consuming than PATRICIA. PAT-FM's average insertion time only is 2/3 of PATRICIA. If R_{in} is the ratio of insertion operations to all operations, we can get the following formulas. Figure 7 shows improved performance of PAT-FM over PATRICIA.

$$T_{PATRICIA} = 1.5LR_{in} + L(1 - R_{in}) = L(1 + 0.5R_{in})$$

$$T_{PAT-FM} = LR_{in} + L(1 - R_{in}) = L$$

$$P_{enhance} = \frac{T_{PACTICIA} - T_{PAT-FM}}{T_{PACTICIA}} = \frac{R_{in}}{2 + R_{in}}$$

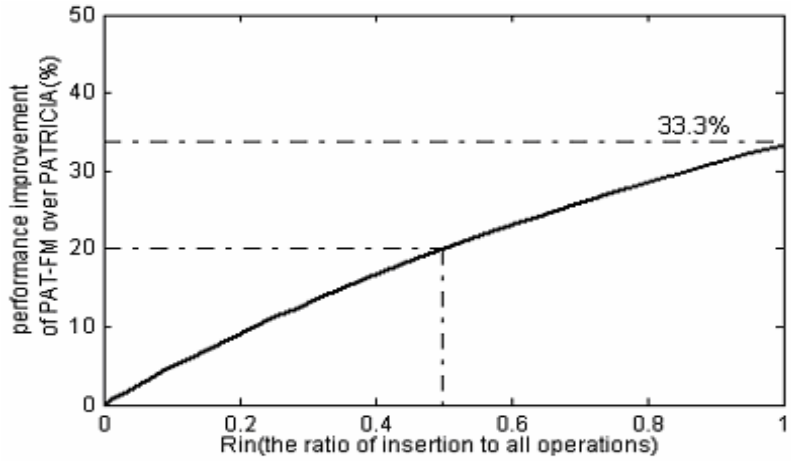


Figure 7. Performance improvement

The main difference between PAT-FM and PATRICIA trie is that *NBT* values are ordered in PATRICIA trie, but disordered in PAT-FM trie. PATRICIA trie is only determined by inserted data set. But PAT-FM trie is determined by both inserted data set and insertion order.

For a dedicated data set which is very balanced in PATRICIA trie may be very unbalanced in PAT-FM trie in some insertion order. But for an unbalanced PATRICIA trie, PAT-FM trie may be very balanced in some insertion order. For random data set, there is no way to determine which trie is more balanced.

For applications whose insertion and deletion is not frequent, we can get more balanced PAT-FM than PATRICIA by pre-calculation. But for applications whose insertion and deletion is frequent, we cannot improve the balance of PAT-FM by pre-calculation. For session table application which session entries are generated and inserted randomly, we cannot compare which trie is more balanced. In fact, for random insertion, balances of PATRICIA trie and PAT-FM trie are close.

Theorem 1. PAT-FM is an optimal 2-ary trie. Proof:

Step 1. From the definition of PATRICIA[6], we know PAT-FM is a kind of PATRICIA.

Step 2. PATRICIA trie is the shallowest trie [6].

Step 3. From the definition of trie[7][8], we know that trie insertion, search and deletion must walk down from a root to a leaf. So for a trie whose average depth is *L*, the best performance of trie insertion, search and deletion is *L*.

Step 4. From Table 4, if the depth of PAT-FM trie is *L*, PAT-FM insertion, search and deletion need *L* memory accesses. So, PAT-FM is the optimal 2-ary trie for fixed-length match.

IV. Specialized Hardware Design

For high speed network, ASIC is often used to improve performance of network device. We implement an ASIC for the proposed session table architecture and proposed algorithms.

We use two methods to reduce the depth of PAT-FM trie further. One is to hash SID, the other is to make use of multi-ary PAT-FM trie. For fixed-length match, we can hash keys to improve performance. To reduce the depth of trie further, we implement a 4-ary PAT-FM trie.

We use a Xilinx FPGA(XC2V3000) to implement the whole Stateful Inspection firewall which supports 3 Gigabit Ethernet ports and a PCI interface. PAT-FM trie is stored in DDR SDRAM(two piece of MT46V32M4), doubly linked list is stored in ZBT SRAM(two piece of IDT75602).

We use random 128 bits data to test the performance of session table insertion, search and deletion. The experimental results are shown in Table 5. Performances of session entries insertion, search and deletion are close.

Furthermore, we mainly test the performance of lookup session table for a match. Figure 8 shows the experimental result, the ASIC can do 2.28 million lookups even the number of session entries up to 1 million. If all packets are smallest packet (64 bytes), the application-specific hardware can process 1.4 Gbps' traffic. And if the average size of packets is 128 bytes, the traffic can up to 2.8 Gbps.

Table 5. Operation numbers per second

Insertion	Search	Deletion
2,464,274	2,796,528	2,471,232

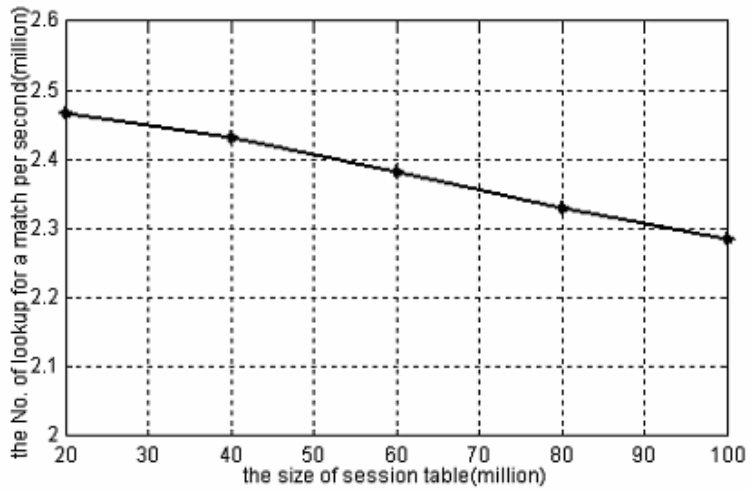


Figure 8. Lookup performance of session table

V. Conclusions

Existed session table architectures have bad timeout processing performance. Dividing session entry to two separate parts and designing different data structures for each other can improve the performance of session table. By pipelining operations of the two parts, the performance of session table is only determined by trie's performance. The new session table architecture can improve its processing performance greatly.

The new fixed-length match algorithm, PAT-FM, improves PATRICIA insertion performance. PAT-FM has following advantages:

(1) It is a dedicated algorithm for fixed-length match.

(2) It is more suited for extremely long-key match than tree.

(3) Can improve PATRICIA insertion performance.

(4) It's an optimal 2-ary trie algorithm.

(5) It is simpler than PATRICIA, which results that it is easier to be implemented in hardware.

Both theoretical and experimental results show that the new session table architecture can work well in Gigabit Ethernet network.

References

[1] Stateful-inspection firewall: The Netscreen way, In *http://www.netscreen.com/products /firewallwpaper.html.*
[2] David W. Chapman Jr., *Cisco Secure PIX Firewalls*, Cisco press. 2001
[3] 3COM Office connect cable/DSL secure gateway data sheet, In *http://www.3com.com /other/pdfs/products/en_US/*
[4] Marcus Goncalves**,** Steven Brown, *Check Point Firewall-1 Administration Guide*, McGraw-Hill, November 2001
[5] IBM Co. *IBM NP4GS3 DATAsheet*. May 2001
[6] M. Okuno, K. Ando, J. Aoe, "An efficient compression method for Patricia tries", *In Proc. of IEEE International Conference on Computational Cybernetics and Simulation*, Volume: 1 , 12-15 Oct. 1997, pp.415 - 420
[7] J. Aoe, "Computer Algorithms-Key Search Strategies", *IEEE Comput. Society Press*, 1991.
[8] J. Aoe, "A Fast Digital Search Algorithm by Using a Double-Array Structure", *IEEE Trans. Software Eng*, vol. 15, no. 9, 1989, pp. 1,066-1,077

Li Xin was born in China, 1976. He received the B. Sc and M. Sc degrees in Material Science from Harbin Institute of Technology, China, in 1999 and 2001 respectively. Since 2001, he has been a Ph. D. degree candidate in computer Science and Engineering from Harbin Institute of Technology, Harbin, China. His current research interests include network firewall, packet classification and VLSI design.

Ji Zhenzhou was born in China, 1965. He received Ph.D degree in Computer Architecture in 2000 from Harbin Institute of Technology, China. He is a professor of Harbin Institute of Technology. His main research interests are computer architecture, information security and high performance computing.

Hu Mingzeng was born in China, 1935. He is a professor of Harbin Institute of Technology. His main research interests are computer architecture, information security and parallel computing.