明新科技大學　校內專題研究計畫成果報告

快速及可擴充之多 TCAM 封包分類器於大量法則表格搜尋

Fast and Scalable Multi-TCAM Classification Engine for Wide Policy Table Lookup

計畫類別：☐任務型計畫　☐整合型計畫　■個人計畫

計畫編號：MUST-97 資工-01

執行期間：　97 年　1 月　1 日至　97 年　9 月　30 日

計畫主持人：陳奎伯

共同主持人：

計畫參與人員：

處理方式：公開於校網頁

執行單位：資訊工程系

中　華　民　國　97 年　10 月　9 日

快速及可擴充之多 TCAM 封包分類器於大量法則表格搜尋

摘要

由於網際網路傳輸資料量正以爆炸式的速度發展，下一代寬頻交換器被設計用以提供傳輸速度可以達到 2.5Gbps、10Gbps 甚至高達 40Gbps。對於傳送有線速度之深層封包檢查，並適合多種 QoS 需求目標，使得封包分類已成為下一代高速交換技術之發展瓶頸。三元內容可尋址記憶體(TCAM) 提供高速平行化之比較運算，並且適合開發出硬體式封包分類器。不過，超高密度之單一 TCAM 並不能夠做為解決方案，因為其並不具備可擴充及夠快速來填充大量封包分類法則。

為了提供可擴充與深層封包檢查，本研究將提出具有多層次 TCAM 之管線化及平行化系統架構，以取得對 IPv6 及多種應用之有線速度封包分類。為了達到管線化與平行化處理之目標，本研究將提出一種可以解決模糊情況案例，以及一種壓縮方法來降低 TCAMs 的空間要求。應用以上機制，本研究也將提出一種能夠以超高速度處理多種 QoS 需求目標之封包分類處理器。

關鍵詞：服務品質、封包分類、三元內容可尋址記憶體、下一代網際網路協定

Fast and Scalable Multi-TCAM Classification Engine for Wide Policy Table Lookup

Abstract

With the explosive growth of Internet traffic, the next generation switches are designed to provide forwarding speed up to 2.5Gbps, 10Gbps, or even 40Gbps. To meet the challenges that deliver wire-speed deep packet inspection for various QoS requirements makes classification become the bottleneck of next-generation high-speed switches. Ternary Content Addressable Memory (TCAM) provides high-speed parallel comparison operations and is suitable to implement hardware-based packet classifier. However, single-TCAM solution with ultra-high density may not be feasible as this solution is not scalable and fast enough to fulfill the wide-policy rule classification.

To provide scalability and deep packet inspection, our research plans to propose pipeline and parallel architectures with multiple TCAMs to obtain wire-speed classification for IPv6 and multi-layer applications. To maintain the pipeline and parallel processes, our research will propose an algorithm to resolve ambiguous cases and a compact method to reduce the space requirements of TCAMs. Applying above mechanisms, our research will also propose a packet classification engine capable of handling multiple QoS requirements at ultra-high speed.

Keywords：QoS, Packet Classification, Ternary CAM (TCAM), IPv6

Contents

Figure Lists

Table Lists

1. INTRODUCTION

In the past few years, with the explosive growth of Internet traffic, the challenges of next generation switches and routers are how to simultaneously provide multiple services and obtain wire-speed switching. Although network processor-based (NP-based) platforms capable of combining flexibility and high-speed switching, almost all network processors (NPUs) cannot offer enough computing power and require co-processor to assist to provide multiple services at 10 Gbps [1][2]. Furthermore, employing standard memory to perform software-based packet classification takes as many as a dozen of memory accesses, and the lookup procedure may cause extra delay and delay jitter that affects real-time services. Unfortunately, there are more and more Internet applications such as Voice over IP, Video Conference, Video on Demand, and etc requiring QoS support [3]. To furnish multi-gigabit multi-service switch platform, our research proposes a scalable and fast classification co-processor to assistance network processor to cope with wide policy table lookup at wire-speed.

To deal with various QoS requirements in multi-gigabit switches, the packet classifier should recognize multiple fields in packet headers within several nano-seconds (ns). Thus, the packet classification engine becomes the bottleneck of forwarding process within high-speed switches [4]. Generally, Layer-4 classification performs 5-tupple lookup [5] and requires less than 144-bit TCAM width. However, while dealing with IPv6 packets, a 5-tuple (source/destination IPv6 address, source/destination ports, and protocol) classifier demands 304-bit width. Because of the over-length header information, single commercial TCAM [6-9] are hardly applied to implement classification engine for complex policy rules. Considering the hardware limitations of TCAM, there are two limitations while employing commercial TCAMs. The first limitation is the width of TCAM is not so flexible and cannot be expanded with the policy rule. For example, TCAMs can only be configured as 72-, 144-, 288-bit width. Second, longer searching key should take more access times to input due to the limitation of data-bus width. For instance, 304-bit IPv6 classification requires 5 clock cycles for input, 1 clock cycle for lookup, and 1 clock cycle for outputting result; the whole lookup procedure totally takes 7 clock cycles. Thus, the delay of packet classification increases with the length of policy rule.

To provide fast and flexible solutions, our research proposes pipeline and parallel architectures to implement multi-TCAM-based classification co-processor for improving IPv6 and multi-layer (Layer 2-7) classification. In order to obtain the pipeline and parallel processes, the *trace-back* and *mis-match* problems should be eliminated. To handle these problems, our research also presents an algorithm to detect and resolve ambiguous cases. Applying above mechanisms, the classification engine can obtain 10Gbps wire-speed

classification for multiple services.

The rest structure of our research is as follows. First of all, we introduce the problem of packet classification and the useful hardware TCAM for routing lookup in Section 2. The proposed multi-TCAM solutions are introduced in Section 3. Section 4 describes the ambiguous cases for the multi-TCAM system and proposes the solutions. The design, implementation, and performance evaluation of multi-TCAM classification engine are presented in Section 5. Finally, we conclude the contributions in Section 6.

2. PACKET CLASSIFICATION WITH TCAM-BASED SOLUTIONS

In order to support end-to-end QoS, each node along the established QoS path should process and forward the packets according to the parameters that are set by signaling protocol. In other words, all nodes should be capable of classifying and forwarding the packets according to their QoS setup. Since the classification is the bottleneck of forwarding process in the switches and routers, this chapter discusses how to design a fast and scalable packet classifier to provide QoS in next generation mobile Internet.

There are two phenomena in the next generation mobile Internet. The former is the bandwidth is quickly growing up. Switches and routers should be capable of handling multi-gigabit traffic (up to OC-192/OC-768). That is switches and routers should be able to process dozen million packets per second. The later is there are more and more QoS required applications such as Voice over IP, Video Conference, Internet Games and so on. Since the classification is the bottleneck, more QoS requirement makes classifier more complex and critical to implement. To satisfy user's QoS requirement and obtain wire-speed switching, lookup process should catch up the speed of switching fabric.

Ternary Content Addressable/Associated Memory (TCAM) is s special type of fully associated memory. Each bit in a TCAM has three states- "0", "1", or don't care "*". TCAM can parallel search all entries of its database. Thus, TCAM performs lookup or classification procedure by using constant delay and suitable for designing and implementing high-speed switches and routers.

2.1. Routing Lookup Using TCAM

Generally, Layer-3 switches and routers using routing protocol such as RIP, OSPF, RIPng, BGP4+ to construct the routing table that indicated the best path to next hop. Routing table contains route entries consist of the prefix, output port, and network metrics. The prefix (140.114.78.*) consists of an IP (140.114.78.2) and a mask (255.255.255.0). The scope of this prefix is from 140.114.78.0 to 140.114.78.255. That is, if the destination of a coming packet falls in the scope of this prefix, the packet would be transfer to the output port recorded in this entry.

However, IP address of IPv4 is 32-bit, and then all possible entries are up to 4G (2^{32}). To search these entries to find a best result takes a lot of latency. Thus, recent researches are engaged in finding the better algorithm with less memory requirement and less lookup delay. TCAM capable of performing constant-delay lookup and longest prefix match (LPM) is a good hardware for routing lookup. TCAM consists of two major elements. The first element is the ternary CAM array. TCAM array performs parallel comparison and output the multi-matched index. The second element is priority encoder. Priority encoder can

select the index with highest priority. While DST address of IP address is input into TCAM as a key, TCAM performs parallel searching by TCAM array and output the best-matched index by priority encoder within constant latency.

2.2. Issues of Classification Using TCAM

Although TCAM classifies packets with in constant delay, there are still some issues using TCAM as a classifier.

- **Density**: Although the width and depth of a TCAM is configurable, the size is usually fixed. For a given density, the wider width of a TCAM, the less depth of TCAM entries. Hence, for a 2 Mega-bit 128-bit wide TCAM, at most 16K classification rules can be supported. As a TCAM row stores a (value, mask) pair, range specifications need to be split into mask specifications, further downloading into the number of usable TCAM entries. Therefore, the density of TCAM is a problem for IPv6 and higher-layer classification due to wider depth is required.

- **Power**: Power dissipated in one TCAM row increases proportionally to its width.

- **Updating**: Since the entries of routing table of rules should be sorted, TCAM updating is another problem. For example, if a prefix (140.114.*.*) has higher priority than prefix (140.114.78.*), then the prefix (140.114.78.*) is never matched. That is because the packet matched (140.114.78.*) is also matched (140.114.*.*), and (140.114.*.*) is selected by priority encoder. As a result, while inserting a routing entry or a rule, then TCAM should be sorted again. This procedure takes O(N) in worse case.

In summary, using TCAM has problems on density, power and updating procedure. Since single high density TCAM is heavy power consumption and less scalability, we proposed multiple-TCAM design to obtain lower power consumption, scalability and fast classification for IPv6 and higher layer applications.

2.3. TCAM Management

As the rule classification table is stored in the TCAM, the TCAM management is one of most important issues for TCAM-based designs. Before the rules are downloaded into the classification table, they are first pre-processed and sorted. For the pre-processing, the “range” defined in each rule is translated into a set of “prefixes” for longest prefix matching (LPM), and the “ambiguous rules” should be detected and resolved. The rules in the TCAM are stored in form of data and subnet mask. For example, an IPv4 address (140.114.78.*) is presented as data (140.114.78.0) and mask (255.255.255.0). Basically, the IP addresses and protocol fields can be directly put into TCAM. However, since the “port”

fields are often set as a range in the rule, the range should be translated into a set of prefixes for longest prefix matching employed in the classification.

The TCAM provides the LPM function that selecting the best-matched rule by the prefix length of rules [10-12]. For layer-3 routing based on the destination IP address, the classification is quite simple, only performing the LPM according to the destination IP address. However, for higher layer packet classifier, multiple-field examination is required. In this case, it is unsuitable to match the best-match rule only according to the prefix length. For instance, if two rules with the same total prefix length but not the same prefix length for every field, it is not suitable to use multiple-TCAM based classifier. This is also called the "ambiguous condition". Two rules are called ambiguous if any ambiguous condition between these two rules exists. Our research will discuss the ambiguous cases in multiple-TCAM environment and present the resolve solution.

Several algorithms have been proposed to improve the multi-field packet classification in recently years. Some of these papers presented the software designs [13-15] to find out the matched rule, and the others [16-18] designed hardware architectures, for example, using CAMs, to accelerate the searching procedure. However, in the worst case, these algorithms may suffer from the problem of backtracking in the searching procedure, which is undesired in the ultra high-speed lookup engines. For example, let us see the multi-level trie structure shown in Figure 1(a). When a packet matches node C in the DST-Tree but fails in the SRC-Tree, the algorithm should go back to node B in the DST-Tree and then searches the SRC-Tree again. This *"backtrack"* procedure may take a lot of time. To avoid the backtracking issue, some improving mechanisms are proposed by employing the additional links; this causes the data structure more complicate.

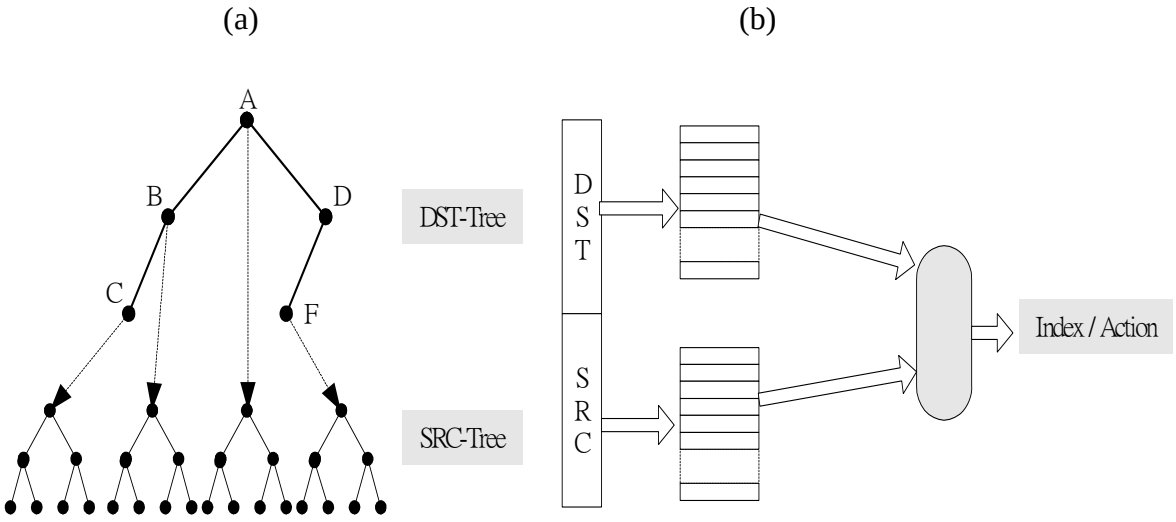

Figure 1. Typical packet classification algorithms.

Nevertheless, this situation may also occur for the hardware architecture. Figure 1(b) shows that if the destination address of a packet matches multiple entries in the DST table, then the searching procedure may need to check the related source information for each of the matched destination entries.

The reason why these algorithms need to perform back tracking in the searching procedure is due to in the hash tables or searching trees, the filter entries (rules) may be multiple matched. If this happens, then it is difficult to choose one correct result, which is an index for next level searching. An elementary idea to resolve this issue is that a rule is selected only when all parameters are matched simultaneously. Our research designs several logical circuits capable of choosing the accurate rule and exporting the searching result within a deterministic latency.

TCAM has been widely employed for the longest prefix matching lookup in Layer-3 or Layer-4 switches. TCAM always selects the best-matched rule according to the priority level, and therefore the rules need to be sorted before they are stored into the TCAM. We take the Layer-3 IP lookup as example, routes are sorted according to the prefix lengths, and for Layer-4 packet classification, rules are sorted according to the priority levels. For multi-TCAM design, each TCAM stores one field such as source IPv6 address or destination IPv6 address. The sorting in each TCAM is done according to the prefix length as described before. For example, we have two IPv6 classification rules, R_1 = (3ffe:3600:B::/48, 3ffe:3600::/32) and R_2 = (3ffe:3600::/32, 3ffe:3600:1::/48); where R_1 is with higher priority level. If R_1 is placed in front of R_2, then the packets with source/destination addresses within in the range (3ffe:3600:B::/48, 3ffe:3600:1::/48) always match with R_1. Nevertheless, those packets should match with R_2. The key matched 3ffe:3600:B::/48 should also match 3ffe:3600::/32. If 3ffe:3600::/32 is placed in front of 3ffe:3600:B::/48, then 3ffe:3600:B::/48 will never be selected. Consequently, for multi-TCAM system, it is interesting that the entries in each TCAM should be sorted according to the prefix lengths, instead of priority levels.

2.4. Translating Ranges to Prefixes

It is clear that an IP address can be stored in the TCAM by using the format of prefix/subnet mask. But typically, SRC port and DST port are often set as ranges. For example, in the typical rules for firewall shown in Figure 2, The SRC-port and DST-port for the first rule are set as ranges. Since it is possible that a rule is extended as several rules after the translation of a range into a set of prefixes, it is very crucial to have an efficient translating algorithm. This section presents a fast algorithm to translate a range into a set of minimum number of prefixes.

SRC-address	DST-address	Protocol	SRC-port	DST-port	ACTION
140.114.*.*	140.112.*.*	TCP	20-21	Gt1024	Pass
140.114.*.*	140.113.*.*	UDP	80	ANY	Pass
140.114.78.50	140.114.64.71	ANY	ANY	ANY	Block

Figure 2. An example of firewall rules with ranges.

For illustration, let us take a five-bit range for example. The range (2, 11) could be partitioned into (2,3), (4,7), and (8,11) which can be presented by three prefixes (0001*), (001**), and (010**), respectively as shown in Figure 3(b). Figure 3(a) shows another example with range (4,27) which can be translated into a set of four prefixes (001**, 01***, 10***, 110**). Let *L, M, N* represent the number of prefixes translated from the SRC-port, DST-port, and protocol fields respectively, of a rule. Then a rule can be translated into *L*M*N* rules, such that each field for each of these rules is presented by a prefix. For example, consider a rule with SRC-port = (4, 27) and DST-port = (2,11) as shown in Figure 3, then this rule can be translated into 4x3 = 12 rules as shown in Figure 3(c).

2.5. Range Translation

As shown in Figure 4, the policy rules are set through web-based interface. When a connection is established, the TCAM management program picks up a rule for this connection and downloads the rule into the TCAM. As the SRC and/or DST ports may be set as ranges in the rule, it is required to translate the rule into a set of rules with prefix format before downloading.

The task of translating a range (X,Y), say (4,15), is to convert the range into a set of binary prefixes. Let us again take five-bit prefixes for instance. Figure 4 shows the range translating procedure. Let the *Left* point and *Right* point stand for X (4) and Y (15), respectively. Then we can find the *Separate-point (SP)* by first performing the XOR operation on the binary presentations of X (00100) and Y (01111). Thus, Z = (X XOR Y) = (01011). Let the LSB of Z be numbered as 0-th bit and the MSB of Z be numbered as 4-th bit. Then start from the MSB of Z, let the position where bit transition happens be denoted as *k*-th bit. For this case, we have *k* = 3 (“0” → “1” happens). Let *W* be the number which all bits are “1” except the right most k bits are “0”. For this case, we have *W* = 11000. Then let SP = (Y AND W) = (01111 AND 11000) = (01000).

Range	Prefixes
(4, 7)	001**
(8, 15)	01***
(16, 23)	10***
(24, 27)	110**

(a)SRC Range (4, 27)

Range	Prefixes
(2, 3)	0001*
(4, 7)	001**
(8, 11)	010**

(b)DST Range (2, 11)

SRC Ranges	DST Ranges	SRC and DST Prefixes
(4, 7)	(2, 3)	001** : 0001*
(4, 7)	(4, 7)	001** : 001**
(4, 7)	(8, 11)	001** : 010**
(8, 15)	(2, 3)	01*** : 0001*
(8, 15)	(4, 7)	01*** : 001**
(8, 15)	(8, 15)	01*** : 010**
(16, 23)	(2, 3)	10*** : 0001*
(16, 23)	(4, 7)	10*** : 001**
(16, 23)	(8, 15)	10*** : 010**
(24, 27)	(2, 3)	110** : 0001*
(24, 27)	(4, 7)	110** : 001**
(24, 27)	(8, 15)	110** : 010**

(C) Translated SRC and DST prefixes

Figure 3. Example of five-bit prefixes stored in the TCAM.

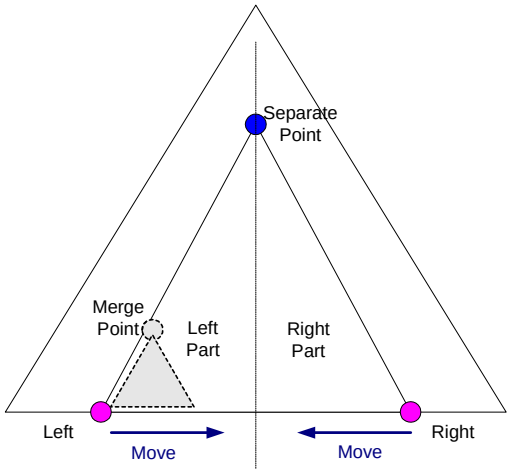

Figure 4. Diagram of range translating procedure.

The left part (4-7) and right part (8-15) of the SP then be processed individually. For the left part, let us start from the smallest element S_L of left part (4 in this case). Let w_L be the number of consecutive "0"s from the LSB of S_L. For this case, we have $w_L = 2$. Let W_L be the number which all bits are "0"s except the right most $m_L = min(w_L,k)$ bits are "1"s. In this case, we have $m_L = min(w_L,k) = (2,3) = 2$, and $W_L = 00011$. Let $M_L = (S_L \text{ OR } W_L)$ be the *merge* point of S_L. In this case, we have $M_L = (00100 \text{ OR } 00011) = 00111$. Then the elements located between S_L and M_L can be merged (presented) as a prefix $P(S_L)$ which is the same as S_L except the right most m_L bits are expressed as '*'. For this case, we have $P(S_L) = 001**$.

On the other hand, for the right part, let us start from the largest element L_R of right part (15 in this case). Let w_R be the number of consecutive "1" from the LSB of L_R. For this case, we have $w_R = 4$. Let W_R be the number which all bits are "1"s except the right most $m_R = min(w_R,k)$ bits are "0"s. In this case, we have $m_R = min(w_R,k) = (4,3) = 3$, and $W_R = 11000$. Let $M_R = (W_R \text{ AND } L_R)$ be the *merge* point of L_R. In this case, we have $M_R = (11000 \text{ AND } 01111) = 01000$. Then the elements located between M_R and L_R can be merged (presented) as a prefix $P(L_R)$ which is the same as L_R except the right most m_R bits are expressed as '*'. For this case, we have $P(L_R) = 01***$.

Let us illustrate this idea more clearly by an example shown in Figure 5, where a range (4,27) is translated into a set of four prefixes step by step. Initially, the SP is computed first. As (00100 XOR 11011) = 11111, we have $k = 4$, and $W = 10000$. SP = (11011 AND 10000) = 10000 (16). For the left part elements, let us start from the smallest element S_L (4). Then we have $w_L = 2$, $m_L = min(w_L,k) = (2,4) = 2$, and $W_L = 00011$. Let $M_L = (S_L \text{ OR } W_L) = (00100 \text{ OR } 00011) = 00111$. Then the elements located between S_L and M_L can be merged (presented) as a prefix $P(S_L) = 001**$ (the 1st generated prefix). Then let the next smallest un-processed left elements be S_L again (for this case, $S_L = 8 = 01000$). Then we have $w_L = 3$, $m_L = min(w_L,k) = (3,4) = 3$, and $W_L = 00111$. Let $M_L = (S_L \text{ OR } W_L) = (01000 \text{ OR } 00111) = 01111$. Then the elements located between S_L and M_L can be merged (presented) as a prefix $P(S_L) = 01***$ (the 2nd generated prefix).

For the right part elements, let us start from the largest element L_R of right part (27 = 11011 in this case). Then we have $w_R = 2$, $m_R = min(w_R,k) = (2,4) = 2$, and $W_R = 11100$. Let $M_R = (W_R \text{ AND } L_R) = (11100 \text{ AND } 11011) = 11000$. Then the elements located between M_R and L_R can be merged as a prefix $P(L_R) = 110**$ (the 3rd generated prefix). Then let the next largest un-processed right element be L_R again (for this case, $L_R = 23 = 10111$). Then we have $w_R = 3$, $m_R = min(w_R,k) = (3,4) = 3$, and $W_R = 11000$. Let $M_R = (W_R \text{ AND } L_R) = (11000 \text{ AND } 10111) = 10000$. Then the elements located between M_R and L_R can be presented as a prefix $P(L_R) = 10***$ (the 4th generated prefix). Finally, we can see that the

range (4,27) can be expressed by a set of four prefixes (001**, 01***, 10***, 110**).

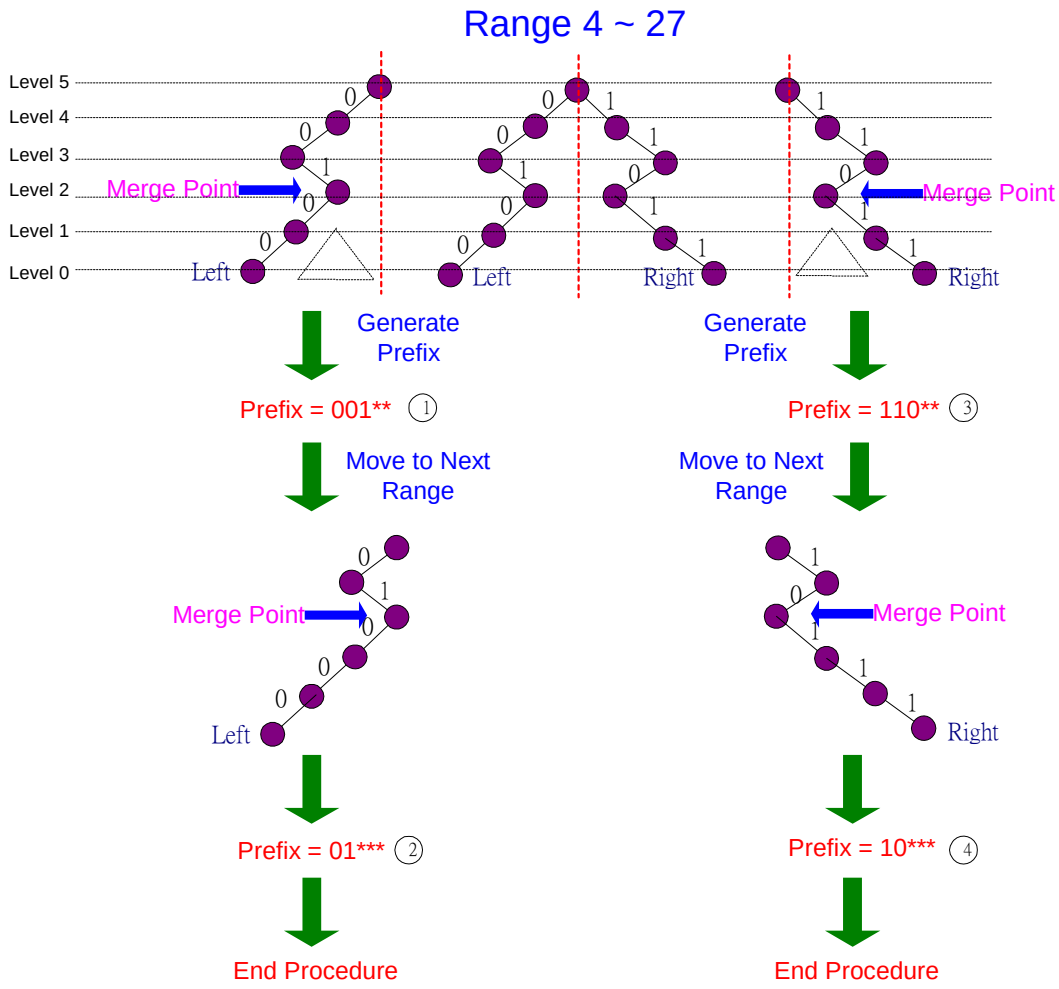

Figure 5. Translate range (4,27) into a set of four prefixes.

The proposed translation algorithm is presented as follows. First of all, the algorithm finds the SP of the input range (X, Y). Then check if the input range (X,Y) can be merged into a single prefix directly. If yes, then the algorithm returns the prefix and stops. Otherwise, the algorithm translates the left part elements and right part elements of the SP individually according to the procedure mentioned above.

2.6. Range Translation Algorithm

Algorithm Range2Prefix

Input: A range (X,Y)
Output: A set of translated prefixes for (X,Y)
Step 1. Find the Separate-Point (SP) of (X,Y).

Let Z = (X XOR Y). Start from the MSB of Z, let k be the first bit position where bit transition happens.

Step 2. Check if the (X,Y) can be merged into a single prefix directly. If yes, then return the prefix and Stop.

Step 3. //Translate left part elements

Let S_L be the smallest un-processed element of left part.

Let w_L be the number of consecutive "0" from the LSB of S_L.

$m_L = min(w_L,k)$;

Let W_L be the number which all bits are "0" except the right most m_L bits are "1".

Let $M_L = (S_L \ OR \ W_L)$ be the merge point of S_L.

Output a prefix $P(S_L)$ which is the same as S_L except the right most m_L bits are expressed as ''.*

If $M_L \neq SP-1$, then go to Step 3.

Step 4. //Translate right part elements

Let L_R be the largest un-processed element of right part.

Let w_R be the number of consecutive "1" from the LSB of L_R.

$m_R = min(w_R,k)$;

Let W_R be the number which all bits are "1" except the right most m_R bits are "0".

Let $M_R = (W_R \ AND \ L_R)$ be the merge point of L_R.

Output a prefix $P(L_R)$ which is the same as L_R except the right most m_R bits are expressed as ''.*

If $M_R \neq SP$, then go to Step 4.

Setp 5. Stop.

3. PROPOSED MULTI-TCAM SOLUTIONS

To cope with various QoS requirements and obtain wire-speed switching, our research proposes a classification co-processor performing constant-time classification for network processor-based platforms. Most control units break incoming packets into fixed-size segments; hence two classification scenarios exist in the network processor-based platforms. One scenario is NPU performs packet classification while receiving start-of-packet (SOP) signal, the other is NPU does not classify the packet until receive entire packet. The lookup operations of these two scenarios are significantly different. As the header information refers to several segments in the first scenario, NPU performs each lookup while NPU gets the header information. NPU can acquire the whole searching key and perform lookup procedure while receiving end-of-packet (EOP) signal. Therefore, our research proposes two architectures of classification co-processor by utilizing multiple TCAMs. In the rest of this session, our research first presents two multi-TCAM architectures for different classification scenarios. Then, our research describes the sorting scheme in the multi-TCAM systems. Finally, our research shows the ambiguous cases that may cause the *trace-back* and *mis-match* problems.

3.1. Pipeline and Parallel Multi-TCAM Architectures

Since the whole classification procedure includes input, search, and output steps, the lookup latency is the sum of the delays of all steps. In other words, searching longer policy table to classify packets takes more time due to the limitation of hardware bus. Take five-tuple IPv6 packet classification as an example; 304-bit searching key should be inputted 5 times on 72-bit data bus. To improve the throughput of packet classification, our research proposes pipeline architecture utilizing multiple TCAMs for the first scenario. This architecture shown in Figure 6 is able to output a result every four clock-cycles with pipeline process.

For the first scenario, FPGA controller is responsible for inputting searching keys, handling lookup procedures, and finally outputting the lookup results. TCAMs are used to store the policy table. Three TCAMs store IPv6 source addresses, IPv6 destination addresses, and information of protocols and ports respectively. In our design, the TCAM-3 also stores the action table. In the first searching step, TCAM-1 outputs the associated tag, and the FPGA controller combines the tag with second searching key (IPv6 destination address) to perform second search. After that, TCAM-2 outputs the second tag and FPGA controller combines the tag with third key (ports and protocol) to perform third search. Finally, TCAM-3 outputs the lookup result. This lookup procedure can be pipeline executed. Taking advantage of pipeline process, the proposed classification can output one result every four clock-cycles at full rate.

In the second scenario, the classifier can get the whole packet information. Thus, our research proposes the parallel architecture to employ multiple TCAMs and increase lookup speed. The architecture of proposed parallel multi-TCAM classification engine is shown in Figure 7. After NPU acquires the whole searching keys, NPU inputs the keys through FPGA controller. Then, all TCAMs execute the lookup procedure simultaneously and produce the associated indexes. Afterwards, the classifier combines all associated indexes and sends to the Binary CAM to perform hash operation. Eventually, FPGA controller replies the result to NPU. The proposed architecture can simultaneously search all fields and get a result after four clock-cycles.

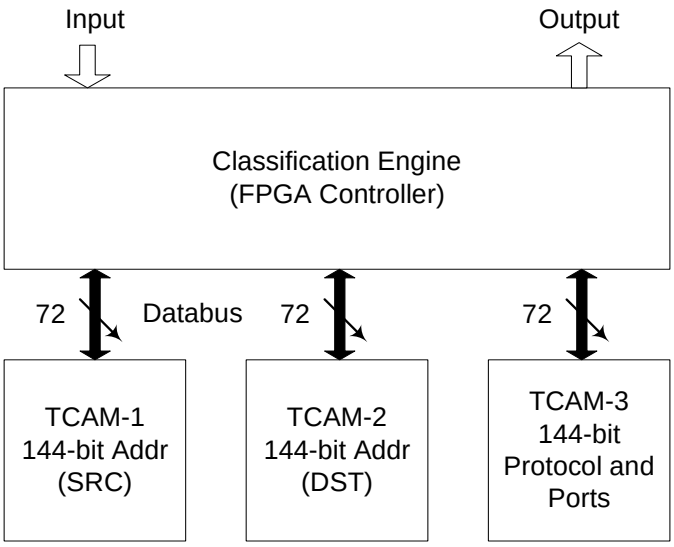

Figure 6. The pipeline architecture of proposed Classification Engine.

Since TCAM performs ternary (0, 1, and don't care) searching, multiple matches may exist in TCAMs. Although TCAM always selects the index with longest prefix length, not all fields of highest policy rule has longest prefix length. Therefore, selecting the accurate index is critical.

3.2. Sorting in Multi-TCAM Systems

Ternary CAM is adept at performing longest prefix match (LPM) to find out the best route for Layer-3 IP lookup. In other words, the data should be sorted before stored into TCAM, and TCAM selects the best-matched rule according to the priority. For Layer-3 lookup, routes are sorted by using their prefix lengths, and rules are sorted by their priorities for Layer-4 classification. Our research employs multiple TCAMs in pipeline and parallel architectures increasing the throughput of packet classification. Since each TCAM stores just one field such as source or destination IPv6 address, sorting in each TCAM is according to the prefix length. For example, two rules- $R_1$and R_2, R_1 is (3ffe:3600:B::/48, 3ffe:3600::/32) and R_2 is (3ffe:3600::/32, 3ffe:3600:1::/48); R_1 has higher priority than R_2.

If R_1 is put in front of R_2 along the priority, then the packets in the range (3ffe:3600:B::/48, 3ffe:3600:1::/48) always matches R_1. Nevertheless, the packets within this range should match R_2. The key matched 3ffe:3600:B::/48 should also match 3ffe:3600::/32. If 3ffe:3600::/32 is put in front of 3ffe:3600:B::/48, then 3ffe:3600:B::/48 is never selected. Accordingly, for multi-TCAM system, the entries in each TCAM should be sorted along the prefix length, instead of priority.

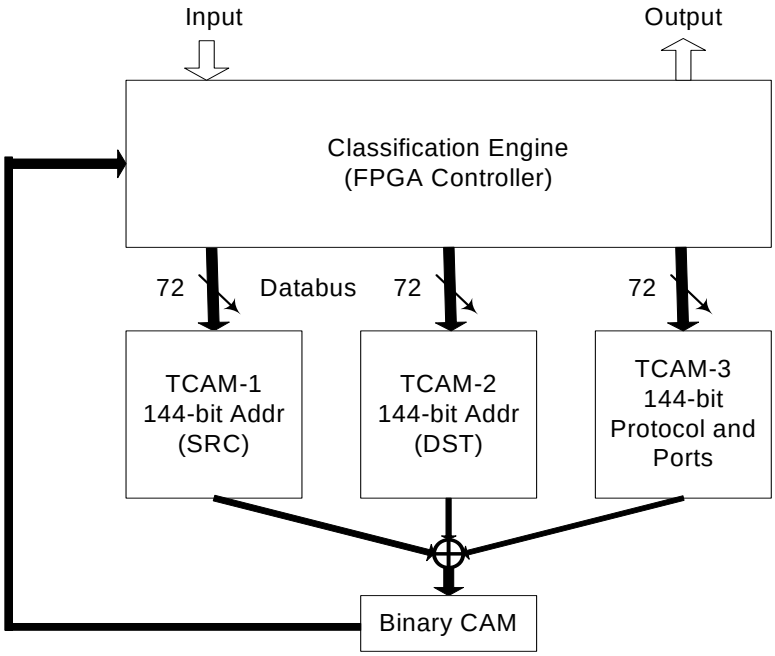

Figure 7. The parallel architecture of proposed Classification Engine.

3.3. Trace-back and Mis-match Problems

To furnish fast and scalable classification engines, the proposed scheme has to eliminate undesirable factors for smoothly executing pipeline and parallel classification. Take the pipeline architecture as an example, if any one TCAM selects incorrect index, the classification procedure would occur error. Then the procedure must go back to previous TCAM and search again. Thus, in this case, classification procedure might spend extra time to lookup and fail to obtain the index at constant-time. The problem that causes extra actions is called the *trace-back* problem. On the other hand, while executing the parallel classification, all TCAMs must simultaneously select an associated index for Binary CAM (BCAM) to execute the hash operation. However, if any TCAM selects incorrect index, then BCAM could not output the accurate result. We define this situation as a *mis-match* problem. Consequently, the factors of *trace-back* problem and *mis-match* problem in multi-TCAM system should be eliminated to ensure fast packet classification.

In our research, we define the *ambiguous cases* between two rules, and these two rules would cause *trace-back* and *mis-match* problems. The ambiguous cases between two

rules have been presented in previous articles [19-20]. These articles present that two rules contains ambiguous cases if they are overlapped, but the relations of policy rules in multi-TCAM systems are not the same. For simplicity, let us use two-field (source and destination address fields) rules as an example. Consider the relation of six rules shown in Figure 8. The rules R_1 and R_2 are completely *disjoined*, rules R_3 and R_4 contain an *overlapping* area, and R_6 is a *subset* of R_5. For instance, consider the rules R_5 and R_6 shown in Figure 8. Since both the source and destination fields of rule R_6 have longer prefix length than those of rule R_5, we say that R_6 is a subset of R_5. For this case, it is clear that R_6 should have a higher priority than R_5. Otherwise, rule R_6 will never have the chance to be matched. In this case, we can just set R_6 higher than R_5 in all TCAMs. On the other hand, the source field of R_2 is a subset of R_1. If lookup source field first, R_2 is always selected. If the key is matched R_1, this case causes the trace-back problem. Since Layer-4 packet classification lookups multiple fields, the priority between two rules in the classification table should be arranged more carefully.

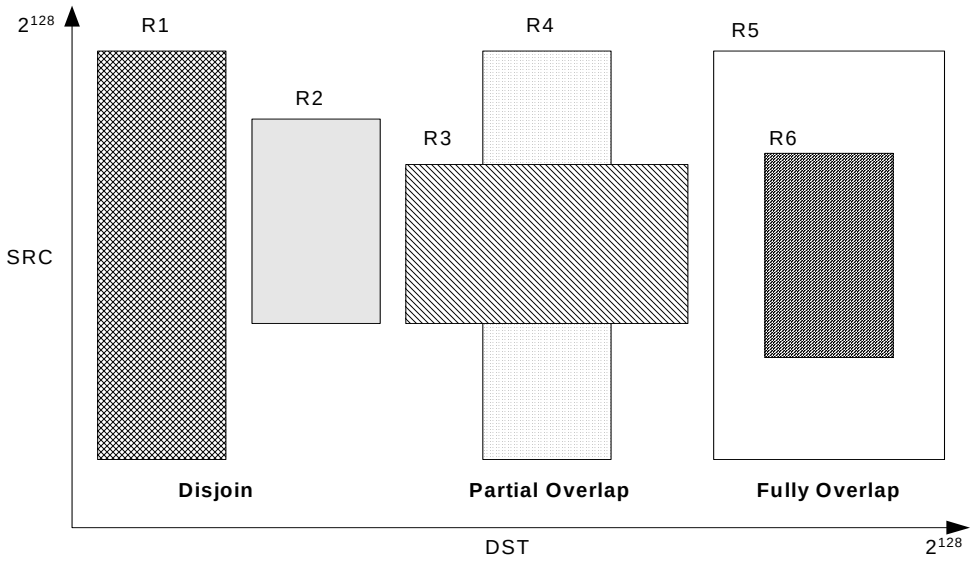

Figure 8. Relationships between six two-field rules.

To sum up, our research defines two ambiguous cases that cause two rules R_i and R_j ambiguous in multi-TCAM environments. The first one is when there exists one field of R_i is a subset of R_j and one field of R_j is a subset of R_i. The first ambiguous case is also called *conflict rules* in literature [19]. The second case is that there is one overlapped field and one disjoined field between R_i and R_j. In case the TCAM contains conflict rules, then it is difficult for TCAM to select the best-matched rule by only according to the prefix length. Searching each field sequentially may suffer second ambiguous case. Since all fields should be matched at the same time, to decide the best selection by LPM in one TCAM/field is inappropriate. In other words, if the TCAM selects a wrong entry for the

first field and lookups the second field, then this will cause a *trace-back* [21] search.

Obviously, TCAM-classification might not work well if exists ambiguous rules. The ambiguous cases in multi-TCAM environments are defined in the following section. The detection and resolution algorithms for this problem are also introduced as well.

3.4. Comparisons of Proposed Multi-TCAM Solutions

To conclude this section, we compare the proposed pipeline and parallel multi-TCAM solutions and depict their characteristics in Table 1. In the pipeline solution, the NPU performs classification while receiving start-of-packet (SOP) signal. In parallel solution, the NPU does not classify the packet until receives end-of-packet (EOP) signal. Thus the pipeline solution is suitable for applying on the classification algorithm that lookup each field at each step. On the other hand, the parallel solution is suitable for the algorithm that takes the whole header information to lookup. Looking into the hardware specification, both of these two solutions consist of a FPGA controller and three 144-bit-width TCAMs. Besides, parallel solution has an extra Binary CAM to perform hashing function and select the result. Since parallel solution has one extra BCAM, pipeline solution is cheaper than parallel solution. Considering the classification latency, the pipeline solution performs three-step lookup operations and each step takes 4 clock cycles to get a result. Thus, the whole lookup procedure takes total 12 clock cycles. By contrast, parallel solution takes 4 clock cycles to perform one lookup procedure. Nevertheless, the throughputs of these two solutions are the same.

Table 1. Comparison of characteristics with pipeline and parallel Multi-TCAMs.

Items	Pipeline Architecture	Parallel Architecture
Classification	When receiving SOP	When receiving EOP
# of FPGA	1	1
# of TCAM	3	3
# of Binary CAM	0	1
Classification Latency	12 clock cycles	4 clock cycles
Throughput	1 per 4 clock cycles	1 per 4 clock cycles
Problem	Trace-back	Mis-match

Both pipeline and parallel solutions can improve the throughput and the scalability for wide policy table lookup. However, the *trace-back* and *mis-match* problems may block the pipeline and parallel processes. Since the ambiguous cases in multi-TCAM systems causes these two problems, the proposed classification engine can obtain fast and scalable lookup as long as removing the ambiguous cases. Therefore, our research introduces the ambiguous cases and presents the solutions in the next session.

4. HANDLING AMBIGUOUS CASES

To provide pipeline and parallel processes for multi-TCAM systems, the *trace-back* and *mis-match* problems must be removed. Previous article [20] has proposed that if two rules contain any overlap region, then these two rules are ambiguous and may cause trace-back problem in multi-TCAM systems. By this definition, (R_5, R_6) and (R_3, R_4) in Figure 8 are ambiguous rules. However, since all entries in each TCAM are sorted by prefix length, R_5 and R_6 can be distinguished by prefix length. On the other hand, although R_1 and R_2 are disjoined in Figure 8, the source fields of these two rules are overlapped. R_1 and R_2 are ambiguous when lookup the source field first. Hence, our research presents two ambiguous cases that cause *trace-back* and *mis-match* problems in multi-TCAM systems. Finally, our research shows the solutions of detecting and solving these cases in the rest of this session.

4.1. Ambiguous Cases in Multi-TCAM Environments

All entries are represented as prefixes in TCAM (ranges can also be translated to prefixes), so the relation between the same field of two rules can be classified as subset, super-set, or disjoin. To clarify, let R_A^i and R_B^i be the *i*-th field of rules A and B, respectively. Then $R_A^i \subset R_B^i$ to denote that R_A^i is a subset of R_B^i, and $R_A^i \otimes R_B^i$ to denote that R_A^i and R_B^i are disjoined. The ambiguous cases in multi-TCAM systems are defined as follows, and for simplification, our research takes two five-bit fields for example:

Case I: Overlapping Rules

In the first case, rules A and B are overlapped if there exists $R_A^i \subset R_B^i$ and $R_B^j \subset R_A^j$, where $i \neq j$. Take Figure 8 as an example again and assume the priority of R_3 is higher than that of R_4. In single-TCAM environment, this is a simple case and what we should do is to put R_3 on a higher position than R_4 in the rule table. Nevertheless, in multi-TCAM environment, each field (a TCAM) should be sorted by prefix length. Taking two *conflicting rules*, (10***, *****) and (*****, 101**), for instance, these rules are overlapped in the area (10***, 101**).

Case II: Hidden Overlapped Rules

The second case is that rules A and B are *hidden overlapped* if there exists some field $R_A^i \subset R_B^i$ and for other fields $R_A^j \otimes R_B^j$, where $i \neq j$. For instance, the rules R_1 and R_2 in Figure 8 are *hidden overlapped.* As for the source field, R_2 is a subset of R_1. But for the destination field, they are *disjoined* in Figure 8. Actually, for single-TCAM systems, R_1 and R_2 are *completely disjoined*. However, in the multi-TCAM systems, each field is stored in one TCAM and the packet classification algorithm compares one field each step. In this

case, it is possible that while searching the source field in the first TCAM, and a wrong rule may be selected according to the longest prefix match algorithm. As a result, the packet classifiers should *trace-back* and retry several times. This *trace-back* scheme is undesirable since it would take various delays to select the best rule. The proposed method is to insert a new rule into the rule table to resolve this ambiguous problem and obtain constant-time lookup.

Take R_1=(10***,11***) and R_2=(101**,10***) for example, R_1 and R_2 cause a *hidden overlapped* case in a two-TCAM system. When a packet with (source, destination) = (10110, 11100) comes in, and the classifier takes the source field as the search key. The classifier selects R_2 with the longest prefix matching. However, when destination field (another TCAM) fails in searching, another entry in source field should be search again.

4.2. Detecting and Resolving Conflict rules

According to above mention, there are two kinds of conflicts (overlapped and hidden overlapped) among the rules in the multi-TCAM systems. This section proposes the ways to detect and resolve these ambiguous cases.

Solution for Case I:

In the first case, two rules A and B are ambiguous, if there exists $R_A^i \subset R_B^i$ and $R_B^j \subset R_A^j$, where $i \neq j$. The longest prefix match algorithm is unable to select the correct rule merely according to the prefix length.

The detecting procedure first distinguishes if all fields of two rules A and B are disjoined. Disjoin rules are undoubtedly not conflicted. If rule A and B are not disjoined but A is a subset of B or B is a subset of A, then A and B can be distinguished successfully by prefix length. The rest of the detecting procedure checks the subset relation of all fields of rule A and B.

The solution for this conflict case is to add a new rule C, in which each field of C is the longer prefix of rules A and B. This resolving algorithm should be recursively executed. After adjusting, the packets belongs to overlapping area will match rule C by LPM.

Solution for Case II:

Two rules A and B have ambiguous case 2, if there exists some field $R_A^i \subset R_B^i$ and for other fields $R_A^j \otimes R_B^j$, where $i \neq j$. The detecting procedure first distinguishes if two rules A and B are hidden overlapped. If rule A and B are hidden overlapped as for the source field, B is a subset of A, but for the destination field, they are disjoined. The detecting procedure of hidden overlapped rules only exists in the multi-TCAM systems as each field is stored

in one TCAM and the packet classification algorithm compares one field at each step.

Similar to case I, the solution for this conflict is to add a new rule C in the rule table. The source field of rule C is the longer prefix length of the source field of rules A and B; however, the destination field is referred to the shorter prefix length destination field of rule A and B. After inserting rule C, the packet will match this new rule. Though this scheme inserts additional rules, it keeps the advantage of constant searching time for all cases.

4.3. Lookup Operations Without Ambiguous Cases

After detecting and resolving the ambiguous cases, the pipeline and parallel multi-TCAM system will not encounter *trace-back* and *mis-match* problems. In other words, the proposed multi-TCAM classification engine can take advantage of the benefits of pipeline and parallel architecture and enhance the lookup speed. The following contents give two examples for pipeline and parallel architectures individually.

To simplify the demonstration, we utilize two fields to show the lookup operations and point out the difference between original and modified policy tables. Figure 9 shows the first example for pipeline architecture. There are two rules that R_1(140.114.78.*,140.114.*.*) and R_2(140.114.*.*,140.114.79.*) in the policy table, and R_1 has higher priority than R_2. For incoming packet (140.114.78.59,140.114.79.80), TCAM-1 selects longest prefix rules and outputs T1 Then, combine T1 and second key (140.114.79.80) will not match in TCAM-2 without handling ambiguous rules. In this case, the trace-back problem occurs and classifier cannot get the correct result. To avoid *trace-back* problem, new rule-R_3 detecting form ambiguous cases is created and inserted into policy table. The lookup procedure is as the following. First lookup in TCAM-1 also get the result (T1). Then, TCAM-2 outputs the correct result (A1) after eliminating ambiguous cases. In addition, the number of rule does not increase in TCAM-1 since the same data entries can be compacted with the same tag. Consequently, the pipeline architecture can be executed smoothly without the trace back problems.

For the parallel scenario shown in Figure 10, the policy rules are divided into two parts and store in TCAM-1 and TCAM-2. BCAM is a hash table and stores hash keys for policy rules. Classification engine inputs the searching keys into TCAMs concurrently and output the associated index. By combining the outputs from TCAM as a hash key, BCAM can lookup and output the correct result.

Originally, TCAM-1 and TCAM-2 output associated indexes (X1 and Y2) respectively. However, inputting the combination of X1 and Y2 will not match in the BCAM hashing. After the modification, only BCAM inserts a new hashing entry and

TCAMs make no change. Although TCAMs still output X1 and Y2, BCAM can extract the correct result after removing the *mis-match* problem.

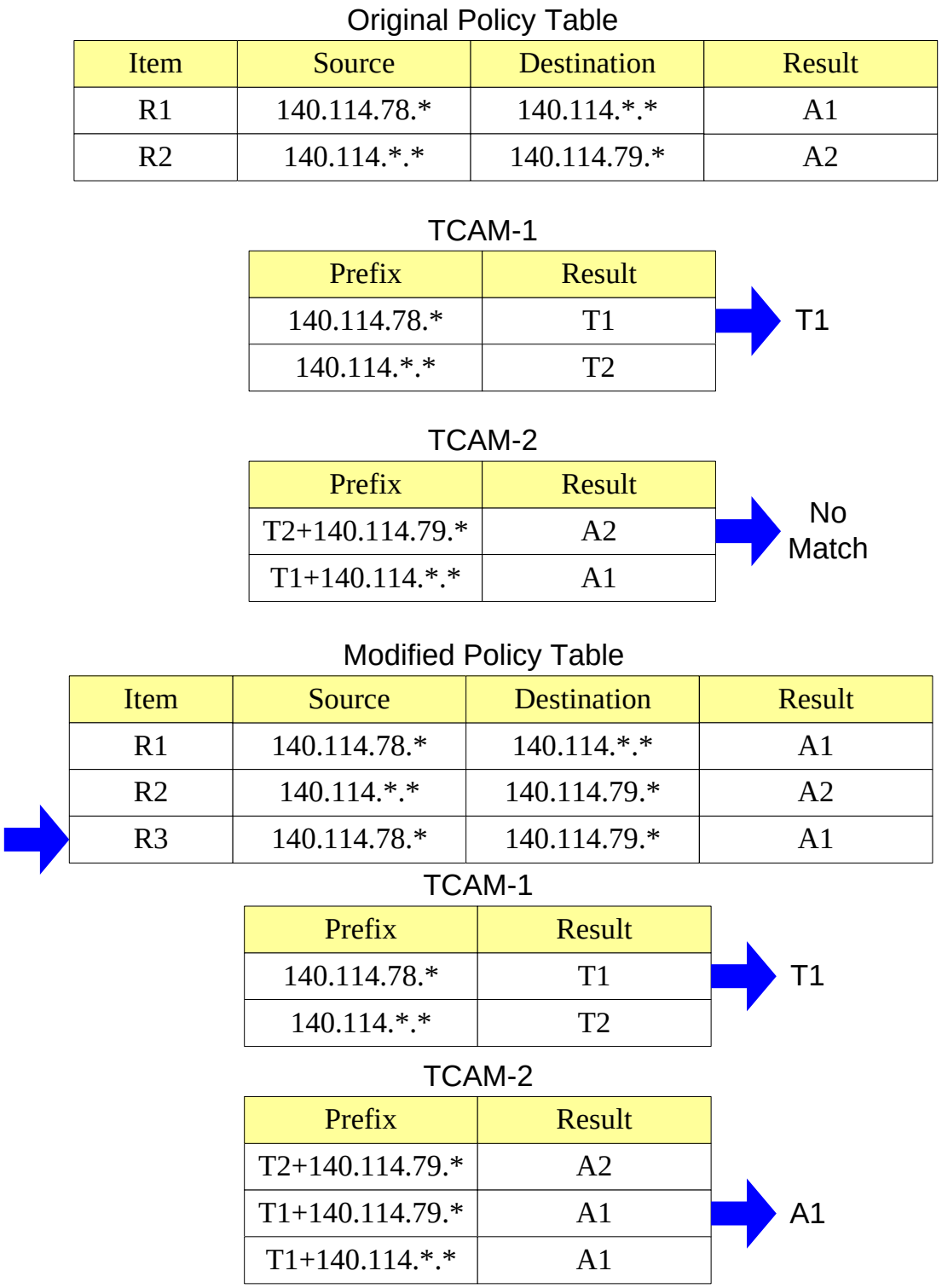

Figure 9. Lookup operations of the pipeline architecture.

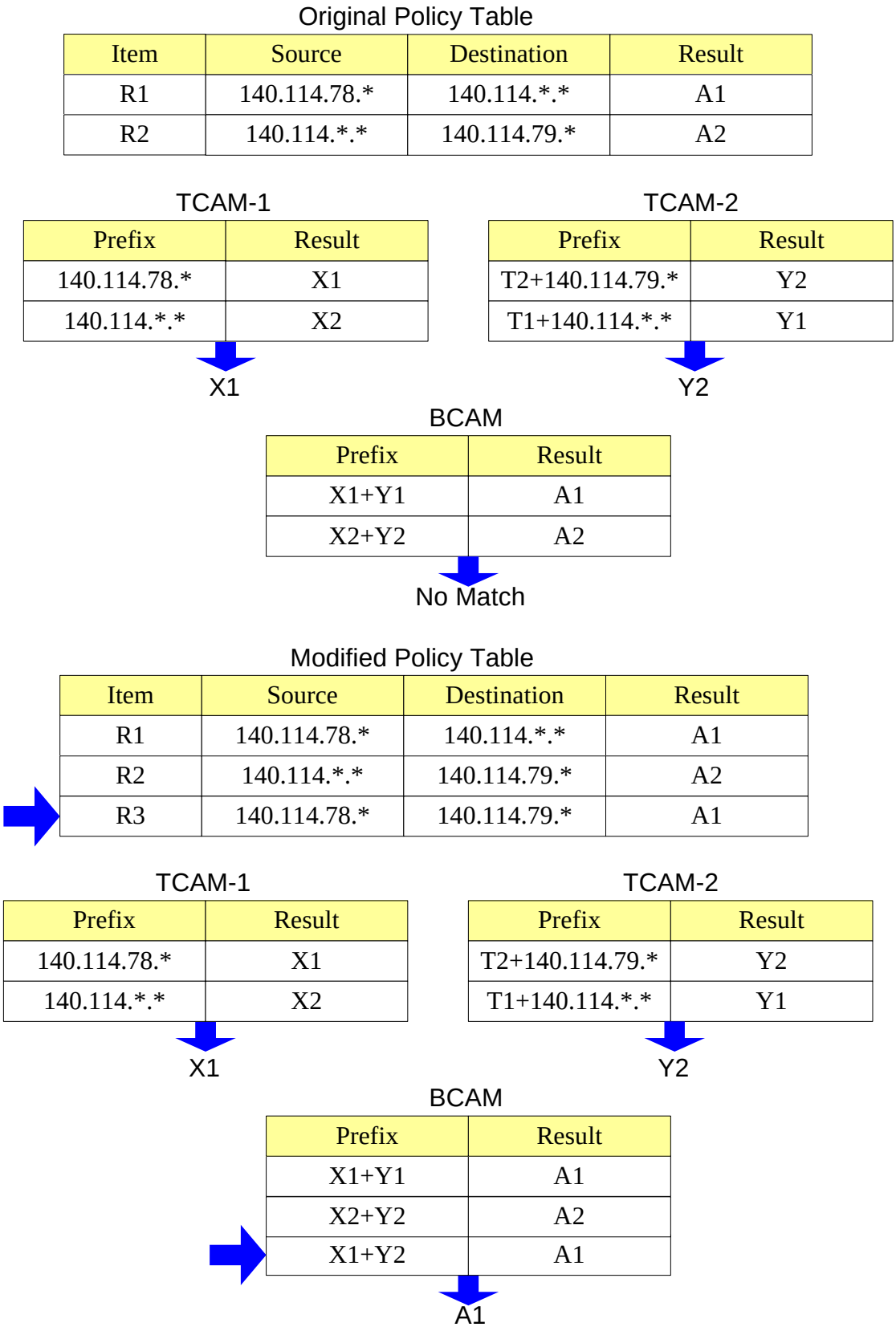

Figure 10. Lookup operations of the parallel architecture.

5. IMPLEMENTATION AND PERFORMANCE EVALUATION

Network processors [22-24] with flexibility, high performance, and rich set of APIs furnish good platforms to build up a QoS supported high-speed switches. However, network processors hardly provide enough computing power for more QoS requirement while scaling to 10 Gbps. Take Intel IXP1200 as an example, pure software-based packet processing requires roughly 700 instructions. Therefore, the classification not only consumes most computing power but also produces extra latency and delay jitter, which may affect real-time traffic. Our research proposes a classification co-processor with multiple TCAMs to share the NPU's loading and classify packets within constant latency.

Table 2 shows the comparison of single-TCAM systems [19], [21], [25-28] and multi-TCAM systems. Since commercial TCAM [6-9] has hardware limitation, single TCAM can only be configured to 72-, 144-, 288-bit widths and takes more time to input longer policy rules. On the other hand, multi-TCAM classification engine would be more scalable, fast and cost efficient than single-TCAM system. Furthermore, taking advantage of pipeline, outputting the 304-bit IPv6 packets classification result only requires 4 clock cycles for multiple TCAM systems. Obviously, due to smoothly executing pipeline procedure, multi-TCAM classifier is faster than single TCAM classifier. Since lookup procedure of single-TCAM system totally takes 7 clocks, the maximum speed of single-TCAM system can only reach 7Gbps with 100MHz TCAM and 64-byte packets. On the other hand, the pipeline and parallel solutions can output one result every 4 clocks. The maximum speed of proposed classification engine can achieve wire-speed deep packet inspection at 10Gbps.

Table 2. Comparison of single-TCAM and multi-TCAM classification engine.

Items	Single TCAM	Multiple TCAMs
Scalability	No	Yes
Cost	High	Low
Power Consumption	High	Low
Throughput	1 result per 7 clocks	1 result per 4 clocks
Max Speed	7Gbps	12Gbps

In our implementation, Verilog is used to develop the FPGA controller in Max Plus II software; the Altera FPGA (EPF10K200EBC600-1) controller and Network Search Engine (NSE3128, TCAM) of Netlogic Microelectronics Inc. [8] are employed. The simulation results in Max Plus II show that the FPGA controller can obtain 33-45Mhz, thus each clock cycle takes 22ns-33ns. Due to the width limitation of data-bus, the entire lookup procedure for 304-bit field totally takes 7 clock cycles. Applying proposed pipeline and parallel

architectures, each result is generated in every 132ns (4*33ns). Figure 11 shows the evaluation for pipeline architecture. This pipeline process furnishes 8-million packet classifications per second. In other words, this implementation has the ability to handle 4 gigabits per second in the evaluation systems when packet length is 64 bytes (minimum length for Ethernet packets). Similarly, parallel architecture can also reach the same speed. The proposed hardware classification engine can perform wire-speed lookup at 10 Gbps as long as replacing the FPGA controller with 100MHz chip.

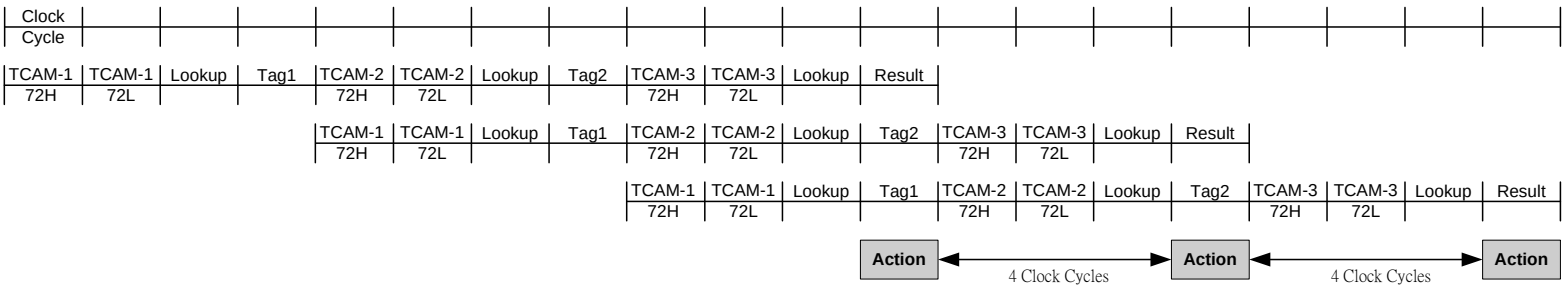

Figure 11. Evaluation of performance of the pipeline multi-TCAM architecture.

In order to increase the throughput of classification, our research proposes the pipeline and parallel hardware architectures. To perform pipeline lookup procedure, the *trace-back* and *mis-match* problems must be eliminated. Although the proposed algorithm solves the *trace-back* and *mis-match* problems, inserting rules may cause the rule table of TCAM to grow up in pipeline architecture. On the contrary, TCAMs make no changes in the parallel architecture. Therefore, our research presents a compact method capable of merging TCAM data entries for pipeline architecture.

Table 3. Comparison of number of original rules and TCAM rules.

Original rules	Max TCAM Rules	Compact Rules
100	100	95
1,000	1,002	992
10,000	10,263	10,006
50,000	50,914	50,045

We simulated the varied rule table sizes from 100 to 50,000 rules by using randomly generated rules, the compact result is shown in Table 3. Due to extra rules, the maximum number of TCAM entries is more than original rule number. For the pipeline lookup procedure, there are two cases that TCAM data entries can be compacted. The former case is $R_A^i \subset R_B^i$ for all *i*, and the action of these two rules are the same. In this case, R_A is a redundant rule and R_B can represent R_A and R_B. The later case is $R_A^i = R_B^i$ for any field, and then these two data entries can be merged even they are belong to different rules. Finally, the size of policy rules decreases and does not increase exponentially after compacting.

6. CONCLUSION

To provide wire-speed classification for multi-service switches at 10Gbps, our research proposes pipeline and parallel architectures employing multiple TCAMs to obtain wire-speed classification for IPv6 and multi-layer applications. To maintain pipeline and parallel process, our research proposes an algorithm to resolve ambiguous cases and a compact method to reduce the memory requirements of TCAMs.

By applying above mechanisms, the proposed classification engine can classify packets every four clock-cycles. In other words, this classification engine can obtain 12 Gbps with minimum length of Ethernet packets by employing 100 MHz TCAMs. In addition, the proposed classification engine is scalable for various QoS requirements by extending FPGA controller and number of TCAMs. Therefore, the proposed classification engine furnishes not only the much fast wide policy table lookup but also more scalability on ultra-high speed switch design.

7. REFERENCE

[1] Linley Gwennap, “Is it time for CAMs?”, EE Times, June 3, 2002 [online] Available WWW http://www.eet.com/semi/c/ip/OEG20020603S0011

[2] Linley Gwennap, “NPU software challenges”, EE Times, Mar 11, 2002 [online] Available WWW http://www.eetimes.com/story/OEG20020311S0054

[3] Fulu Li, Seddigh N., Nandy B. and Matute D., “An empirical study of today's Internet traffic for differentiated services IP QoS” ISCC 2000, Antibes-Juan les Pins, France, Jul. 2000, pp. 207 –213.

[4] Nen-Fu Huang and Shi-Ming Zhao, “A novel IP-routing lookup scheme and hardware architecture for multi-gigabit switching routers,” IEEE JSAC, June 1999, pp. 1093 –1104.

[5] Gupta P. and McKeown N., “Algorithms for packet classification,” IEEE Network, March-April 2001, pp. 24 –32.

[6] Kawasaki LSI’s Classification CAM products web page [online] Available WWW http://www.klsi.com/products/CAM.htm

[7] MUSIC Semiconductors’ Routing Co-processor and Ternary CAM [online] Available WWW http://www.music-ic.com/product/products.html

[8] NetLogic’s Network Search Engine (NSE) web page [online] Available WWW http://www.netlogicmicro.com/products/products.html

[9] SiberCore Technologies’ Ternary Content Addressable Memory (T-CAM) technology [online] Available WWW http://www.sibercore.com/products.htm

[10] Shah, D. and Gupta, P., “Fast updating algorithms for TCAM,” IEEE Micro, Jan.-Feb. 2001, pp. 36 –47.

[11] Shafai F., Schultz K.J., Gibson G.F.R., Bluschke A.G. and Somppi D.E., “Fully parallel 30-MHz, 2.5-Mb CAM,” Solid-State Circuits, Nov. 1998, pp. 1690 –1696.

[12] T. Hayashi and T. Miyazaki, “High-speed table lookup engine for IPv6 longest prefix match,” IEEE GLOBECOM '99, Brazil, December 1999, pp. 1576 –1581.

[13] Singh, K., “A configurable 5-D packet classification engine with 4Mpacket/s throughput for high-speed data networking”, ISSCC 2000, San Francisco, February 2000 , P82 -P83.

[14] Jiří Matoušek, “Range searching with efficient hierarchical cuttings”, Proceedings of the eighth annual symposium on Computational geometry, Berlin Germany, June 1992, P276–P285.

[15] Borg, N.; Svanberg, E.; Schelen, “Efficient multi-field packet classification for QoS purposes”, IWQoS '99, London, June 1999 ,P109–P118.

[16] V. Srinivasan, G. Varghese, S. Suri and M. Waldvogel , “Fast and scalable layer four switching”, Proceedings of the ACM SIGCOMM '98, Vancouver Canada, August 1998, P191–P202.

[17] Gupta P. and McKeown N., “Packet classification on multiple fields”, Proceedings of ACM SIGCOMM '99, Cambridge, August 1999, P147–P160.

[18] Jun Xu; Singhal, M.; Degroat, J., “A novel cache architecture to support layer-four packet classification at memory access speeds”, IEEE INFOCOM 2000, Israel, March 2000, P1445-P1454.

[19] Hari A., Suri S. and Parulkar G., “Detecting and resolving packet filter conflicts,” IEEE INFOCOM 2000, Israel, Mar. 2000, pp. 1203 -1212.

[20] M. Uga and K. Shiomoto, "A novel ultra high-speed multi-layer table lookup method using TCAM for differentiated services in the Internet" in Proc. IEEE HPSR 2001 pp.240-244.

[21] Nen-Fu Huang, Whai-En Chen, Jiau-Yu Luo and Jun-Min Chen, “Design of Multi-field Ipv6 Packet Classifiers Using Ternary CAMs,” IEEE GLOBECOM 2001, San Antonio, Texas, USA, 2001.

[22] Intel products web page [online] Available WWW http://www.intel.com/design/network/products/npfamily/ixp1200.htm

[23] MMC/AMCC products web page [online] Available WWW http://www.mmcnetworks.com/solutions/

[24] Vitesse products web page [online] Available WWW http://www.vitesse.com/products/index.cfm

[25] Kobayashi M., Murase T. and Kuriyama A., “A longest prefix match search engine for multi-gigabit IP processing,” IEEE ICC 2000, New Orleans, Louisiana, Mar. 2000, pp. 1360 –1364.

[26] Shafai F., Schultz K.J., Gibson G.F.R., Bluschke A.G. and Somppi D.E., “Fully parallel 30-MHz, 2.5-Mb CAM,” Solid-State Circuits, Nov. 1998, pp. 1690 –1696.

[27] Shah, D. and Gupta, P., “Fast updating algorithms for TCAM,” IEEE Micro, Jan.-Feb. 2001, pp. 36 –47.

[28] T. Hayashi and T. Miyazaki, “High-speed table lookup engine for IPv6 longest prefix match,” IEEE GLOBECOM '99, Brazil, December 1999, pp. 1576 –1581.

明新科技大學　97 年度　研究計畫執行成果自評表

計畫類別：		□任務導向計畫 □整合型計畫 ■個人計畫
所屬院(部)：		■工學院　□管理學院　□服務學院 □通識教育部
執 行 系 別：		資訊工程系(中心)
計畫主持人：		職 稱：講師
計畫名稱：		快速及可擴充之多 TCAM 封包分類器於大量法則表格搜尋
計畫編號：		MUST-97 資工-01
計畫執行時間：		97 年　1 月　1 日 至　97 年　9 月　30 日
計畫執行成效	教學方面	1. 對於改進教學成果方面之具體成效： 瞭解以傳送有線速度之深層封包檢查，並能在適合多種 QoS 之需求目標，已使得封包分類已成為下一代高速交換技術之發展瓶頸。 2. 對於提昇學生論文/專題研究能力之具體成效： 驗證封包分類問題為下一代高速交換技術之發展瓶頸，並經由三元內容可尋址記憶體(TCAM)所提供之高速平行化比較運算，開發出硬體式封包分類器。 3. 其他方面之具體成效：
	學術研究方面	1. 該計畫是否有衍生出其他計畫案 □是　■否 計畫名稱： 2. 該計畫是否有產生論文並發表 □已發表 ■預定投稿/審查中 □否 發表期刊(研討會)名稱： 發表期刊(研討會)日期：____年___月___日 3. 該計畫是否有要衍生產學合作案、專利、技術移轉 □是　■否 請說明衍生項目：
成果自評		計畫預期目標： 提出具有多層次 TCAM 之管線化及平行化系統架構，以取得對 IPv6 及多種應用之有線速度封包分類。 計畫執行結果： 提出可以解決兩大模糊問題之解決方案，及以壓縮方法來降低 TCAM 之空間要求。 預期目標達成率：100%
		其它具體成效： (若不敷使用請另加附頁繕寫)