JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

PCAM-BASED WIRE-SPEED MULTIDIMENSIONAL PACKET CLASSIFICATION

Firoz Shaikh, Graduate Student,
Computer Science and Engineering Department,
Jadavpur University, Kolkata-700032
Email: firoz983656@gmail.com

Swapan K. Ray, SMIEE
Professor,
Computer Science and Engineering Department
Jadavpur University, Kolkata-700032
Email: skray@ieee.org

ABSTRACT: Packet classification is the process of categorizing packets according to a given Rule Database. Building large high-speed multi-field packet classifiers is an acknowledged challenge to researchers. Although the rules specify ranges, IP prefixes and wildcards, current classifiers inefficiently search a range as multiple prefixes. The present paper deals with a part of the total packet classification problem. Specifically, it proposes a generalized hardware-specific wire-speed direct search technique for ranges and wildcards but not IP prefixes. The proposed hardware is basically a pipelined content-addressable or associative memory (PCAM or PAM), based on a pipelined content-to-address memory (PCTAM). Prior to performing the high-speed pipelined associative search, some preprocessing of the Rule Database is done to build an associative search table. Since it uses only commercial RAM arrays with a few simple processing elements, the PAM is likely to offer a better alternative, in respect of cost, capacity and power consumption, to ternary CAM (TCAM) which is often used in high-speed packet classifiers. The proposed range-based search engine has a constant search time of about two memory access times, independent of the size of the Rule Database and the width of the field values and, thus, may prove attractive for use as a component in large high-speed multidimensional packet classifiers.

INTRODUCTION

IP routers in the global Internet are increasingly being required to offer sophisticated capabilities like providing different qualities of service (QOS) to different applications for fulfilling the various security, accounting and management needs of Internet Service Providers (ISP). In order to meet these objectives, routers need to categorize the arriving IP packets into different "flows" using the packet headers. Simple examples of flows may be "all packets belonging to a particular TCP connection", "all voice-over-IP packets coming from ISP5", etc. This activity of categorizing packets into different flows is called "packet classification" and is one of the most important and, at the same time, difficult packet processing functions performed by the contemporary IP routers deployed in the global Internet [1], [2], [3]. A flow is defined by a "rule" that specifies some criteria to be met by some selected fields in the header of a packet arriving at a router which, if satisfied, will enable the router to classify the packet as belonging to the flow. Each rule has an "action" associated with it which is applied on all the packets that belong to the flow defined by this rule. Such actions may include discarding the packet, sending the packet to a particular

queue, forwarding the packet via a special high-speed network (e.g., ATM), and so on. It should be noted that flows may not be mutually exclusive, i.e., Rule X may include rule Y also or they overlap arbitrarily.

A collection of rules is called a "packet classifier" (PC) and TABLE 1 [1] shows a real-life classifier with 4 rules in 4 dimensions or fields included in the header of an IP packet. The most commonly used fields for packet classification, a 5-tuple, are the source and destination IP addresses, source and destination port numbers and the transport layer protocol. A PC dealing with K fields (K>2) is generally called a multi-dimensional PC. A packet may match multiple rules and, in such a case, the action of the PC corresponds to the rules having the highest priority. When no explicit priorities are specified against the rules in the rule database of a PC, the convention is to treat the rules higher in the list as having higher priority. It should be emphasized that a PC may have a large rule database (tens of thousands of rules) and the multidimensional packet classification may have to be carried out at a very high throughput (tens of millions of packets per second, depending on the line speed). Finally, it needs to be mentioned that packet classification is also known as" layer 4 switching" because routing decisions can be based on headers available at layer 4 or higher in the OSI Architecture [4].

The work reported in this paper is a part of the total process of packet classification using the independent field search approach which is broadly overviewed in Sections II and IV. The present Paper describes a high-speed pipelined hardware technique, using a Pipelined CAM (PCAM) [9], [10] for searching any ranged-based field in the packet header, i.e. fields other than the two prefixes. The search is associative so that the search output is a pointer to the intermediate result which is the rule set associated with the field value in the packet header that was searched against the corresponding rule components of all the rules in the Rule Database.

Following this section, we present in Section II, a review of the previous work in Packet Classification that have been published in the literature. In Section III is presented a brief overview of the PCTAM and PCAM. After an initial detailing of the decomposition or independent field search approach, we describe, in Section IV, the procedure for the

Rule	Network layer destination (address/mask)	Network layer source (address/mask)	Transport layer destination	Transport layer protocol	Action
R1	152.163.190.69/255.255.255	152.163.80.11/255.255.255.255	*	*	Deny
R2	152.168.3.0/255.255.255.0	152.163.200.157/255.255.255.255	eq www	udp	Deny
R3	152.163.198.4/255.255.255.255	152.163.160.0./255.255.255.255	gt 1023	tcp	Permit
R4	0.0.0.0/0.0.0.0	0.0.0.0/0.0.0.0	*	*	Permit

TABLE 1. A real-life classifier in four dimensions

creation of the Associative Search Table (AST) which will be associatively searched at a high speed by the PCAM. The mechanism of carrying out the high-speed search of the AST is described in section V. Finally, some concluding remarks are made in section VI. A glossary of the various acronyms used in the paper has been provided in the APPENDIX for the convenience of the readers.

PREVIOUS WORK

Many different approaches towards packet classification have been reported in the literature during the last ten years or more. A naïve approach, which works for very small Rule Database (e.g., the number of rules N<100) and relatively low-speed links, tries to sequentially match all the relevant fields in the header of the arriving packet against the corresponding rule-components of each rule in the Rule Database. For moderately large Rule Databases (N hundreds to few thousands) and fast links, Ternary Content Addressable Memories (TCAMs) [5] are often employed for parallel search of the Rule Database by storing the rules in the descending order of priority. Though it achieves a high-speed search, TCAM-based solutions have limitations in respect of cost, capacity and power dissipation and, hence, are not much scalable for building large multidimensional PCs. There are some decision-trees based classification algorithms like Grid-of-tries [4] and Fast Inverted Segment (FIS) trees [6] which are specialized for the widely used case of two IP address fields (source and destination) and thus cannot work for multidimensional packet classification. However, though it also employs a decision-tree approach, the algorithm Hierarchical Intelligent Cuttings (HiCuts) [7] can perform multidimensional classification and takes O (MN) memory where M>>1.

Unfortunately, none of the above approaches except, possibly, the HiCuts, is at all suitable for performing the general packet classification at "wire sped" (throughput>100 million packet per second) on a large Rule Database (N>=100,000 rules) and covering a large number of fields (K>=10) in the IP packet header. Additionally, the algorithm must not either require "impractically large" amount of memory (e.g.; exponentially increasing with N) or be too slow to adapt to, say, tens of updates in the Rule Database per sec. Fortunately, however, the "decomposition" or "independent search" approach proposed in [2], [4] appears to have provided a broad framework towards a possible solution to this generalized packet classification problem which is acknowledged to be hard. The basic idea is to decompose a K-field search problem into K single-field search problems to generate K intermediate results and then suitably combine these intermediate results, possibly through an appropriate encoding, to obtain the final result. The big advantage of this approach is that the K single – field searches can be performed independently, in parallel and by using any specialized high-speed optimized technique (some such techniques now exist for IP prefix fields). However, the real challenge lies in the second part of the procedure, i.e., in combining or aggregating the K intermediate results most efficiently to select the highest priority rule. In the following paragraph, we present a brief review of several works on packet classification which have adopted this decomposition approach. Since our work is also based on this approach (however, its scope is limited to only a part of the total procedure), we present some additional details of this generalized approach in Section IV,

Lakshman and Stiliadis [2] have provided a solution which is called Lucent Bit Vector or simply Bit Vector (BV) algorithm because it uses bit vectors to represent the sets of rules forming the intermediate results. However, both the time and memory complexity of the BV scheme is O (N). Since the bit vectors in the BV algorithm are sparse, Baboescu & Varghese [3] have suggested an improved encoding technique called Aggregate BV (ABV) to make the time complexity logarithmic. The Cross-producting solution [4] does not perform the aggregation of intermediate results in real time as in [2] but pre-computes (during the preprocessing phase) all the possible results. Though this approach results in a constant time classification (throughput performance is excellent), the memory requirement becomes O (N^2). Similar performance as cross-productioning is obtained by the heuristic approach called Recursive Flow Classification (RFC) proposed in [7]. Another decomposition technique called parallel packet classification (P^2C) [8] introduced a novel technique of encoding and aggregation of the intermediate results. It achieves a design trade-off between lookup complexity, memory requirement and update speed where the last one is especially impressive. Moreover, the P^2C reduces the TCAM storage requirement significantly by using some amount of SRAM. However, the algorithm is complex.

A BRIEF OVERVIEW OF THE PCAM

Since packet classification requires high speed search of a large rule database, the traditionally popular hardware device CAM and, specifically, its augmented version Ternary CAM (TCAM) which is a special type of associative memory, has been employed for this purpose. However, though operational simplicity and high speed are the main attractive features of the TCAM, it has important disadvantages like high cost, large power dissipation and somewhat limited capacity [1]. The reason that is common to all these three factors is that while the static RAM (SRAM) needs only 4-6 transistors per bit, the TCAM requires 11-15 transistors per bit. As a result, use of TCAMs in building very large PCs having hundreds of thousands of rules is not envisaged. As an alternative to the TCAM which has a parallel architecture, design of a Pipelined CAM (PCAM) was recently described [9]. Since, barring a few simple digital circuits, PCAM is built almost entirely with RAM array, it can overcome all the aforesaid disadvantages of TCAMs. In this section, we present a brief overview of the PCAM as an aid to understand its use in the proposed high-speed search engine for searching of different range-based field values. The latter is an important component in the design of a PC [see Section V]. More details on the design and implementation of a PCAM are available in [9], [10].

In order to explain the basic principle of operation of the PCAM, we need to present the novel concept of a Content-To-Address Memory (CTAM) in the general context of the functioning of a memory unit. A $N(=2^n)\times W$ memory unit may be viewed as a bidirectional one-to-one association between a set of addresses **{**A_i}, A_i€{0,1,2,…..,N-1} and the set of their respective stored contents {X_i}, X_i €{0,1,2,……,2^W-1}, where the A_is and the X_is are all represented in binary. This bidirectional association can be viewed as a pair of complementary mapping functions shown in equation 1.

$X_i = f(A_i)$……...(1a) ; $A_i = f^{-1}(X_i)$…….(1b);

The traditional memory unit (RAM/ROM), which offers a read access based on the forward mapping function of equation 1(a), actually functions as an "Address-To-Content Memory" (ATCM). In contrast a memory unit that offers a read access based on the inverse mapping function of equation **1(b)** may be termed a "Content–To-Address Memory" (CTAM). Two important characteristics of a CTAM are obvious:

1.The CTAM will suffer from duplicate content problem since the same content may be stored at more than one address.

2. Both the data bus and the address bus in the CTAM will be bidirectional.

Though no physical devices for building a CTAM exist, a Binary Search Processor (BSP), which runs the well-known binary search algorithm on ordered data stored in an adjunct RAM, was design to act as a N-word CTAM. Through $n=\log_2 N$ or less number of trials involving read accesses made in a (2^n-1)-word ordered-data RAM, the BSP can determine the presence or absence of a given search data or search key index (SKI) in the adjunct RAM and can also output the address in binary of the RAM location where the SKI resides. Thus, a BSP in conjunction with an adjunct RAM containing the ordered data to be searched effectively functions as a CTAM. By pipelining the operation of a modular BSP having n identical simple processing stages (processing elements PEs) and letting each PE search its dedicated local copy of the ordered RAM, a synchronous pipelined BSP (PBSP) or pipelined CTAM (PCTAM) has been designed. Figure 1(a) shows the block diagram of an individual stage in the PCTAM and Figure 1(b) shows the block diagram representation of the n-stage PCTAM itself. The r-th stage in the PCTAM**,** r=n-1, n-2,…..,0 receives from its predecessor the trial address (TA_r+1) and the result of comparison of SKI with (TA) in the form of "Greater than "(G),"Equal to"(E) and "Less than" (L) bits and generates the corresponding output values for its successor stage. A few basic points relevant to the design and performance of the PCTAM are as follows:

1. Each local copy of the ordered-RAM is of the size $(2^n \times w)$, where $N<=2^n-1$. In case $N<2^n-1$, then the highest possible value of the SKI, i.e., 2^W-1, is stored in all the higher addresses beyond N. Location 0 is not used to store a data element since it is not accessed in the binary search algorithm.

2. Although, in the PCTAM, the ordered-data RAM needs to be replicated n times, the fast dwindling cost of RAM together with the extreme simplicity of the n PEs make the PCTAM an attractive hardware element for building large and high- throughput search processors.

3. Addition of two more stages (these are only marginally different from the other n-stages) to the PCTAM can take care of the duplicate content problem [9].

4. The PCTAM can achieve a throughout rate higher than $\mathbf{1/2t_a}$, where $\mathbf{t_a}$ is the RAM access time in seconds.

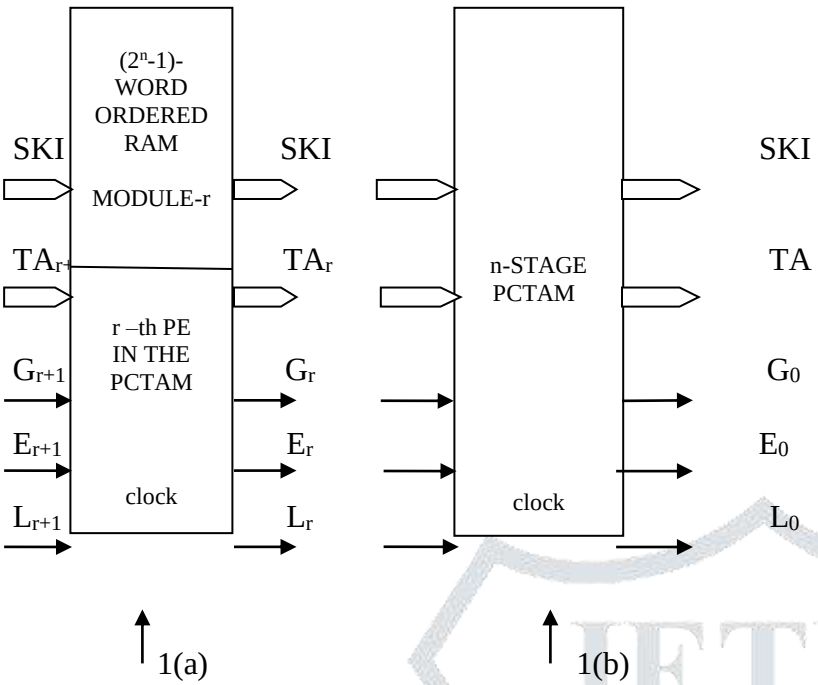

Figure1: Block diagram representation of (a) the r-th stage of a PCTAM and (b) a n-stage PCTAM

Rest of this section will be devoted to explain how PCAM has been realized from the PCTAM with a little augmentation. A CAM or AM is a memory array where a word is accessed by specifying a part of its content, rather than its address. Functionally, each W-bit word in an N×W CAM array has two broad or major fields, namely a W_s (<W)-bit "search key" field (SKF) and a W_d (<=W)-bit data field (DF). We assume that, for each of the N words in the CAM, the SKF is stored in an N×W SKF Storage RAM (SKFSR) and the DF is stored in a separate N×W_d DF Storage RAM (DFSR). The following two steps can now build a PCAM

1. Order the contents of the SKFSR, ascendingly or descendingly and build a n-stage PCTAM for searching the ordered content of the SKFSR.
2. Deploy a N-word "reordering" RAM called "Address Mapping RAM" (AMR) and write in its location j, the word i, for all j and all i, if the word originally occupying location i in the SKFSR has moved to location j after the contents of the SKFSR were ordered..
3. Augment the n-stage PCTAM with two more stages namely, the AMR followed by the DFSR.

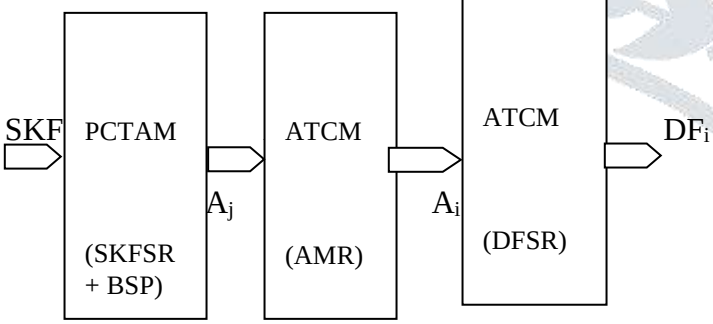

Figure 2. Three block representation of a PCAM

Figure 2 shows block diagram representation of the n+2 stage PCAM, where only the first stage is a PCTAM but the next two stages are simple ATCMs or RAMs. Using static RAM (SRAM), the PCAM is capable of searching an arbitrarily large SKFSR for any large stream of SKFs (i.e, SKIs) and reading out its associated information from the DFSR at a constant high throughput rate on the order of 50-100 million associative searches per second. How the PCAM has been employed for the application discussed in this paper will be shown in Section V.

ASSOCIATIVE SEARCH TABLE CREATION

As stated earlier in Section II, much of the previous research in packet classification has established [2], [7], [4] ,[8], [3] that the complex problem of multidimensional packet classification on a large number of fields can be solved

using a "divide-and-conquer" approach, broadly employing the four step procedure outlined below. The first step involves an elaborate preprocessing whereas the remaining three steps may to be carried out in real time.

1. Create a search table, which we shall call the Associative Search Table (AST) for each of the K fields $\{F_k\}$, k=1,2,…..,K, by including as index in AST_k, the k-th AST, all the rule components (ranges, prefixes, wildcards or their derivatives) corresponding to $\{F_k\}$, as specified in each of the N rules comprising the Rule Database of the PC. With the i-th index in AST_k, i=1, 2, 3,…., I_k, associate a set of rules taken from the Rule Database (we shall call this rule set the Associated Rule Set ARS_{ik}), such that if the field value (FV_k) in the field F_k of the header of a packet arriving at the PC matches the i-th index in AST_k, then it will match the k-th component in all the rules included in ARS_{ik}. Each rule in ARS_{ik} will have the "potential" for also matching all the remaining K-1 FVs in the packet header, though, of course, only a few (but at least one) will ultimately match. Stated in another way, if the search of AST_k against the FV_k of an arrived packet generates ARS_{ik}, then the packet has the "potential" for being classified by one, more all of the rules comprising the ARS_{ik}

2. For each packet arriving at the router, carry out an independent search, in parallel of each of the K ASTs**,** namely, AST_1 through AST_K, for finding match with the corresponding K SVs $\{SV_k\}$, in the header of the arrived packet. For the matching index (entry) in each of the K ASTs, read out its ARS_{ik} (i is different in different ASTs).

3. Perform a set of intersection operation between all the K ARSs generated in step 2, to obtain a small set of rules (the set may even have only one rule), to be called the Intersection Rule Set (IRS), for the arrived packet. Each rule in the IRS "actually" matches all the K fields in the header of the arrived packet and, thus, can classify the packet correctly.

4. Out of all the rules in the IRS, select the one which has the highest priority in the Rule Database (Rule are usually arranged in the rule database in descending order of priority) for classifying the packet and, accordingly, apply the action specified against this rule in the Rule Database to the arrived packet.

In the present paper, we deal only with the first two steps in the total process of the packet classification and propose a PCAM-based "wire-speed" search technique for range-based fields. The technique is applicable to those of the K fields in the packet header where the FVs are specified in the form of ranges expressed as operator/number pairs (including the operators <, >, <=,>= and =) or wildcards (*) but not as IP prefixes. Thus FVs for fields like source and destination ports, transport layer protocols, type of service (TOS), transport layer protocol, flags, etc; can be searched and their associated IRSs can be generated with a high throughput by the proposed range-matching scheme and its hardware implementation. In the remaining part of this section, we describe the general preprocessing scheme which explains how the AST for each of the K range-based fields may be created from the given Rule Database of a PC. Thereafter, we show how an AST is searched by the PCAM, described in the previous section, to obtain the ARS_k corresponding to any FV_k. We shall make the realistic assumption that the Rule Database is updated much more slowly than it is searched, so that, in case of an update in the Rule Database, the affected AST(s) may even be created afresh.

PREPROCESSING SCHEME TOWARD AST CREATION

In order to explain the various steps involved in the preprocessing scheme leading to the creation of the 2-column AST, comprising of a fixed-length index column and variable-length ARS column. We shall consider a small hypothetical Implied Range-Based Rule Table (IRBRT) shown in TABLE 2(a) which may be looked upon, for the present purpose, as the given Rule Database (minus the "action" column) of a one –dimensional PC. It contains one number each of all the seven possible types of range (operator/number) specifications including wildcards and hence can act as representative of any range-oriented field data in the packet header that may be specified in the Rule Database of a multidimensional PC. The various steps in the preprocessing scheme are described blew.

Step1. We first create an Explicit Range-Based Rule Table (ERBRT), shown in TABLE 2(b), by converting each implicit range in the given IRBRT to an explicit range represented by two End-point FVs (EFV), namely, the Low EFV and the High EFV. For simplicity, we have assumed a domain of 8-bit positive numbers for representing the EFVs.

Step2. From the N-entry ERBRT, we next generate a 2N-entry Unordered End-point FV Table (UEFVT), shown in TABLE 2(c), by associating a 2-tuple (Rule #, Extremity Type) with each EFV. The Extremity Type (ET) is either Low (L) or High (H) and specifies the type of the EFV in a given range. We shall henceforth denote these 2-tuples as Rule-ET (RET) pairs or as R-ET-P elements.

Step3. In order to be able to perform a binary search, we next arrange the 2N entries in the UEFVT in an ascending order of their EFVs to build the Ordered EFVT (OEFVT) and then merge together all entries in the OEFVT which have the same EFV. TABLE 2(d) and TABLE 2(e) shows the OEFVT and the Merged OEFVT (MOEFVT), respectively. Note that merging not only avoids duplicate EFVs in the EFV column but also addresses the reality that, because the N ranges specified in the N rules in the IRBRT may overlap in an arbitrary manner, a particular EFV (as also any FV, in general) may occur in multiple ranges and hence may correspond to multiple RETs. It should also be noted that whereas the OEFVT has 2N (in the present case 2N=14) entries, the MOEFT has a considerably reduced number of entries, say N' (N'=9 in the present case).

TABLE2. The various interim tables leading to the creation of the AST.
(a) Implied Range-Based Rule Table (IRBRT);
(b) Explicit Range-Based Rule Table (ERBRT);
(c) Unordered End-point FV Table (UEFVT);
(d) Ordered End-point FV Table (OEFVT);
(e) Merged Ordered End-point FVT (MOEFVT)

(a)

Rule#	Implicit Range
1	<185
2	=188
3	>=140
4	137-177
5	<=171
6	>202
7	*

(b)

Rule#	Explicit Range
1	0-185
2	188-188
3	140-255
4	137-177
5	0-171
6	202-255
7	0-255

(c)

End-point Value	R-ET-P
0	1, L
185	1 ,H
188	2, L
188	2, H
140	3, L
255	3, H
137	4, L
177	4, H
0	5, L
171	5, H
202	6, L
255	6, H
0	7, L
255	7, H

(d)

End-point Value	R-ET-P
0	1,L
0	5,L
0	7,L
137	4,L
140	3,L

(e)

End-point Value	R-ET-P
0	1,L; 5,L; 7,L
137	4,L
140	3,L
171	5,H
177	4,H

171	5,H	185	1,H
177	4,H	188	2,L; 2,H
185	1,H	202	6,L
188	2,L	255	3,H; 6,H; 7,H
188	2,H		
202	6,L		
255	6,H		
255	3,H		
255	7,H		

Step4. In this final step of preprocessing, we create a two- column AST for the k-th field which will have an index column (a search table) and, associated with it, an ARS column where the entry ARS_{ik}, i=1,2,3,…..,I_k and k=1,2,3,…,K (k=1=K) in our present discussion is associated with the index value Augmented EFV_{ik} or $AEFV_{ik}$ (AEFV is different from EFV, as will be explained shortly) under the index column. Towards the goal of building this AST, we first make a fundamental observation regarding ranges that if the relevant component of a rule R_j specifies the range (EFV_{jL}-EFV_{jH}), then the rule R_j should be associated with any FV that not only equals EFV_{jL} or EFV_{jH} but also lies in between, i.e., lies within the non-inclusive range EFV_{jL}<FV<EFV_{jH} .This implies that the FV in the header of an arriving packet should be compared with each EFV not only for equality/inequality but also for the type of the inequality, i.e., the comparison should provide the comparison result (CR) as less than (L), equal to (E) or greater than (G). In this context, it may be noted that, after the search of every SKI, the PCTAM in the PCAM (see Figures 1 and 2 in Section III) generates, as its output, (i) the Trial Address (TA) as well as (ii) the CR (L, E or G) of the comparison between the SKI and (TA), the content of TA.

It may be observed, that, if the EFVs are ascendingly ordered in the PCTAM, then implication of the CR is as follows:

If L=1, then (TA-1) <SKI< (TA)

If E=1, then (TA) = SKI

If G=1, then (TA+1) > SKI> (TA)

From the above discussion, the desired 2-column AST can now be obtained from the MOEFVT of TABLE 2 (e) by augmenting it as follows.

1. Convert each entry in the EFV column of TABLE 2 (e) to three Augmented EFV (AEFV) entries under the index column of the AST by tagging the TA, where the EFV is stored, and the 3 CR bits L, E, and G, respectively, after each EFV. Thus a N'-entry MOEFVT will be converted to a 3N'-entry AST.

2. Build the ARS column to be associated with the above index column by employing the following two-pass algorithm.

Pass1: Build an Interim ARS (IARS) column by choosing the R-ET-P elements in each IARS corresponding to each AEFV in the index column as follows (see TABLE 3).

AEFV Choice of R-ET-P elements in the IARS

EFV-L: Include all R-ET-P elements present in the IARS of the previous entry (it is associated with an EFV-G).

EFV-E: Include (a) all"L"-suffixed R-ET-P elements present in the IARS of the previous entry and (b) all R-ET-P elements that are associated with this EFV in the MOEFVT [TABLE 2(e)], after deleting all the "H" ETs.

EFV-G: Include only the L-suffixed R-ET-P elements present in the IARS of the previous entry.

Pass 2: Delete the ET"H" in all R-ET-P elements in the entire IARS column to obtain the ARSs and hence the desired AST.

TABLE 3. The Associated Search Table (AST). [The column IARS is not a part of the AST- it only meets an interim requirement in preprocessing].

AEFV			IARS	ARS
EFV	TA	CR		
0	1	L	NULL	NULL
		E	1L, 5L, 7L	1, 5, 7
		G	1L, 5L, 7L	1, 5, 7
137	2	L	1L, 5L, 7L	1, 5, 7
		E	1L, 4L, 5L, 7L	1, 4, 5, 7
		G	1L, 4L, 5L, 7L	1, 4, 5, 7
140	3	L	1L, 4L, 5L, 7L	1, 4, 5, 7
		E	1L, 3L, 4L, 5L, 7L	1, 3, 4, 5, 7
		G	1L, 3L, 4L, 5L, 7L	1, 3, 4, 5, 7
171	4	L	1L, 3L, 4L, 5L, 7L	1, 3, 4, 5, 7
		E	1L, 3L, 4L, 5, 7L	1, 3, 4, 5, 7
		G	1L, 3L, 4L, 7L	1, 3, 4, 7
177	5	L	1L, 3L, 4L, 7L	1, 3, 4, 7
		E	1L, 3L, 4, 7L	1, 3, 4, 7
		G	1L, 3L, 7L	1, 3, 7
185	6	L	1L, 3L, 7L	1, 3, 7
		E	1, 3L, 7L	1, 3, 7
		G	3L, 7L	3, 7
188	7	L	3L, 7L	3, 7
		E	2, 3L, 7L	2, 3, 7
		G	3L, 7L	3, 7
202	8	L	3L, 7L	3, 7
		E	3L, 6L, 7L	3, 6, 7
		G	3L, 6L, 7L	3, 6, 7
255	9	L	3L, 6L, 7L	3, 6, 7
		E	3, 6, 7	3, 6, 7
		G	NULL	NULL

The complete AST is shown in TABLE 3 where the IARS column has also been included, though it will play no part in the searching process. Note that in both the IARS and the ARS column in TABLE 3, the two border entries corresponding to the first "L" and the last "G" is NULL for obvious reasons. In the following section, we describe how the ARS corresponding to an FV, found in the header of a packet arriving at the router, is read out at wire-speed.

HIGH SPEED SEARCH OF THE AST

As a part of the total process of packet classification, scope of the present research requires that when an IP packet arrives at a router, the router should extract the field value FV_k in the k-th field of the packet header and match it with the k-th component (specified as either ranges or wildcards but not as IP prefixes) of all rules in the Rule Database. Then the router should read out the numbers (ID's) of all the "matching" rules, i.e., the rules where the k-th component match, i,e, include FV_k. This section is intended to describe how this is done using the PCTAM (or PAM), in conjunction with the AST shown in TABLE 3. The search is carried out in the following steps:

Step 1. High-speed search by PCTAM

The N(=9) ordered unique numbers in the EFV column of the MOEFVT, reproduced under the index column in the AST, are stored in consecutive locations in the local memory in each stage of an n-stage PCTAM , where n=^$\log_2$ N^ but N<=2^n-1. If N<2^n-1, then the highest possible value of EFV(=255) is stored in the remaining (2^n-1-N) high-end locations, i.e., locations 10 through 15. When the FV_k is fed as a SKI into the PCTAM, the latter, after performing pipelined binary search generates, as its output, the TA, where the matching EFV is stored along with the 3 CR bits L,E or G which indicate SKI <(TA), SKI=(TA) and SKI>(TA), respectively . For example, for three successive SKIs 100, 175 and 200, the PCTAM output will be (TA=1, G=1), (TA=5, L=1) and (TA=7, G=1), respectively. In this context of high speed pipelined binary search carried out in a PCTAM, it should be pointed out that each stage in the PCTAM has a delay of 2 t_a sec (t_a is the memory access time), so that the n-stage PCTAM has a latency of 2nt_a seconds but a high throughput of (1/2t_a) searches/sec.

STEP 2: Associative readout by PCAM

This step suitably augments the PCTAM to a PCAM or PAM. The PCAM directly provides, as its output, the ARS which is associated with the SKI, i.e., the FV_k that is fed to the PCTAM. However, it should be noted in this context that, in a large PC, not only the size of the ordered N-word RAM but also the sizes of the individual ARSs may be large. Hence the ARSs should be stored together sequentially, in a separate large RAM, to be called the ARS STORAGE RAM (ARSSRAM) .The i-th ARS ARS_i will be stored in a variable-sized block of memory with the beginning address A_i. Thus, so far as the PCAM is concerned, it will only store the pointers A_i', i=1, 2, 3,……. 3N', which will contain the addresses {A_i}. These pointers will be stored in a RAM which we would call the ARS POINTER RAM (ARSPRAM). Below we provide a simple scheme for mapping the PCTAM addresses output available in the form of the 2-tuple (TA, CR) to the pointer addresses {A_i}, to be stored in the ASRPRAM.

We note from the index column of the AST (see TABLE 3), that, for every TA, there are 3 ARSs corresponding to the 3 bits in the CR. So, a simple technique for obtaining the desired mapping will be to sequentially allot every 4 consecutive locations in the ARSPRAM for each successive TA. Out of these 4, the first three will store pointers to the ARSSRAM and the 4-th one will remain unused. The 3 CR bits will be encoded into three 2-bit numbers, namely, L(00), E(01) and G(10) and these 2-bit numbers will be concatenated (suffixed) with the TA to generate the (n+2)-bit Augmented TA(ATA) which will address the ASRPRAM to read out the {A_i}. Figure 3 shows the proposed 3-block structure of the PCAM. Interestingly, this PCAM structure may be viewed as a "FV-TO-ARS CONVERTER" for range-based fields in a multidimensional PC.

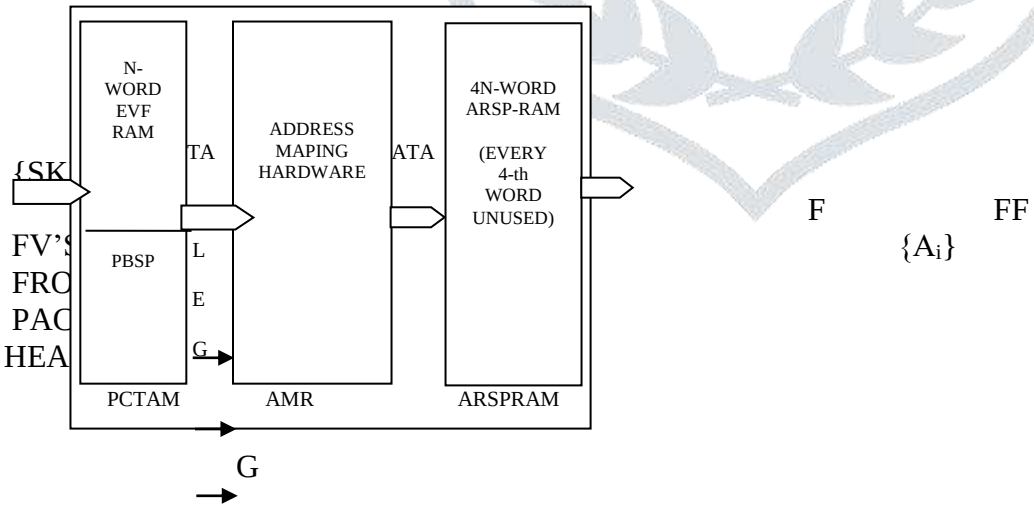

Figure 3. Three-block structure of the PCAM

CONCLUDING REMARKS

A scheme of wire-speed search of range-based fields for potential use in multidimensional packet classifiers having very large rule databases has been described. Ranges are allowed to be specified in any form including equality and wildcard. This implies that the field in the packet classifier may correspond to any field in the header of an IP

packet, except the source and destination network prefixes. Moreover, the specified ranges in the field are allowed to overlap arbitrarily. The range matching has been done directly as ranges and not by converting ranges into multiple prefixes and then performing IP prefix match, as is commonly done.

A method of preprocessing the specified ranges, i.e. the specified range-based rule components of all rules in the given rule database of a packet classifier, is shown for the creation of an associated search table (AST) for the concerned field. The AST is searched, in an associative manner, against the field value of the concerned field in the header of each arriving packet. This search yields a pointer to an associated rule set (ARS) which is stored in an ARS Storage RAM along with all other ARSs of the same field, and which contains all rules whose corresponding rule component matches the field value in the arrived packet. The associative binary search of the AST is carried out extremely fast, in less than two RAM access times, by a Pipelined Content-Addressable Memory (PCAM). The PCAM is designed around a pipelined Content-To-Address Memory (PCTAM) and a brief overview of both the PCTAM and the PCAM has been provided in this paper.

It is expected that the PCAM-based range matching subsystem described in this paper will be superior in respect of capacity, power dissipation, and design flexibility and, in all probability, even cost, compared to its any possible implementation on the commonly used Ternary CAM (TCAM). That is, the PCAM-based range-based field search engine described in this paper is likely to be more scalable and hence more suitable for building large packet classifiers than a similar search engine which may possibly be built around TCAM. This expectation stems from the fact that the PCAM hardware is predominantly RAM with only some simple processing elements, and cost of RAM has been dwindling phenomenally fast. As a matter of fact, only $\log_2 N$ processing elements each having its own N-word RAM, are needed to build an N-rule field-searching engine. Further work towards the ultimate goal of designing a large classifier is in progress.

REFERENCES

1. P. Gupta and N. Mckeown, “Algorithm for packet classification, “ IEEE Network, March/April 2001, pp. 24-32.

2. T. V. Lakshman and D. Stiliadis, “High speed policy-based packet forwarding using efficient multi-dimensional range matching,” in Proc. ACM SIGCOMM, Sep. 1998, pp. 191-202.

3. F. Baboescu and G. Varghese, “Scalable Packet Classification, “IEEE/ACM Transactions on Networking, Vol-13, No. 1, Feb 2005.

4. V. Srinivasan, G. Varghese, S. Suri, and M. Waldvogel, “Fast scalable level four switching,” in Proc. ACM SIGCOMM, Sep. 1998, pp. 203-214

5. Siber Core Technologies Inc., “Siber CAM Ultra-18 M SCT 1842,“ Product Brief, 2002.

6. A. Feldman and S. Muthukrishnan, “Tradeoffs for packet classification,”in Proc. IEEE INFOCOM, vol. 3, Mar. 2000, pp. 1193–1202.

7. P. Gupta and N. McKeown, “Packet classification on multiple fields,” in Proc. ACM SIGCOMM, Sep. 1999, pp. 147-160.

8. J. V. Lunteren and T. Engbersen, “Fast and scalable packet classification,” IEEE J. on Selected Areas in Communication, Vol.-21, No.4, MAY 2003, PP. 560-571.

9. S. K. Ray, “ Large - Capacity High – Throughput Low – Cost Pipelined CAM Using Pipelined CTAM, “ IEEE Transactions on Computers, Vol-55, No. 5, May 2006, pp. 575-587.

10. S. K. Ray, S. Dutta and A. K. Saha, "A low-cost pipelined multi-lingual E-dictionary using a pipelined CTAM, Proc. Int. Conf. on Computing: Theory and Applications, held at ISI, Kolkata during March 5-7, pp. 158-163, Published by IEEE Computer Science Press.

APPENDIX
A GLOSSARY OF ACRONYMS USED

AEFV : Augmented End-point Field Value
AM : Associative Memory
ARS : Associated Rule Set
ARSPRAM : ARS Pointer RAM
ARSSRAM : ARS Storage RAM
ATA : Augmented Trial Address
ATCM : Address-To-Content Memory
BSP : Binary Search Processor
CAM : Content Addressable Memory
CR : Comparison Result
CTAM : Content-To-Address Memory
DF : Data Field
DFSR : DF Storage RAM
EFV : End-point Field Value
ERBRT : Explicit Range Based Rule Table
FV : Field Value
H : High
IARS : Interim ARS
IRBRT : Implied Range Based Rule Table
IRS : Input Rule Set
ISP : Internet Service Provider
L : Low
MOEFVT : Merged Ordered EFV Table
OEFVT : Ordered EFV Table
PAM : Pipelined Associative Memory
PBSP : Pipelined BSP
PC : Packet Classifier
PCAM : Pipelined CAM
PCTAM : Pipelined CTAM
R-ET-P : Rule Extremity Type Pair
RAM : Random Access Memory
SKF : Search Key Field
SKI : Search Key Index
SKFSR : SKF Storage RAM
SRAM : Static RAM
TA : Trial Address
TCAM : Ternary CAM
UEFVT : Unordered End-point FV Table