Packet filter implementation on the Intel network Processor IXP1200

Avdhoot A. Bane
(02D07016)
Dual degree, Electrical Engineering

Why network processors?

- **Multiprocessing hence higher speed**

  General purpose processors are not parallel, hence not fast for high packet traffic rate network.

- **Programmability.**

  Hardware routers are fast; but not programmable.

Multifield packet classification

- Many fields:

  source IP address, destination IP address, source port number, destination port number, protocol
- Different layers:

  transport layer header,

  network layer header,

  link layer header. Application level layer is also used in some cases.

What is DoS attack ?

- an attack on a computer system or network that causes a loss of service to users, typically the loss of network connectivity and services by consuming the bandwidth of the victim network or overloading the computational resources of the victim system..[

Most malicious ways of attack

- Distributed DoS attack

  Many attackers simultaneously send requests to the target. Hence,

  1) difficult to detect

  2) combined power of all attackers increases traffic drastically

- Spoofed source IP address DoS attack

  Due to spoofed source IP exact attacking source cannot be detected.

  Blocking prevents legitimate user and not attacking source

Intel IXP1200 platform

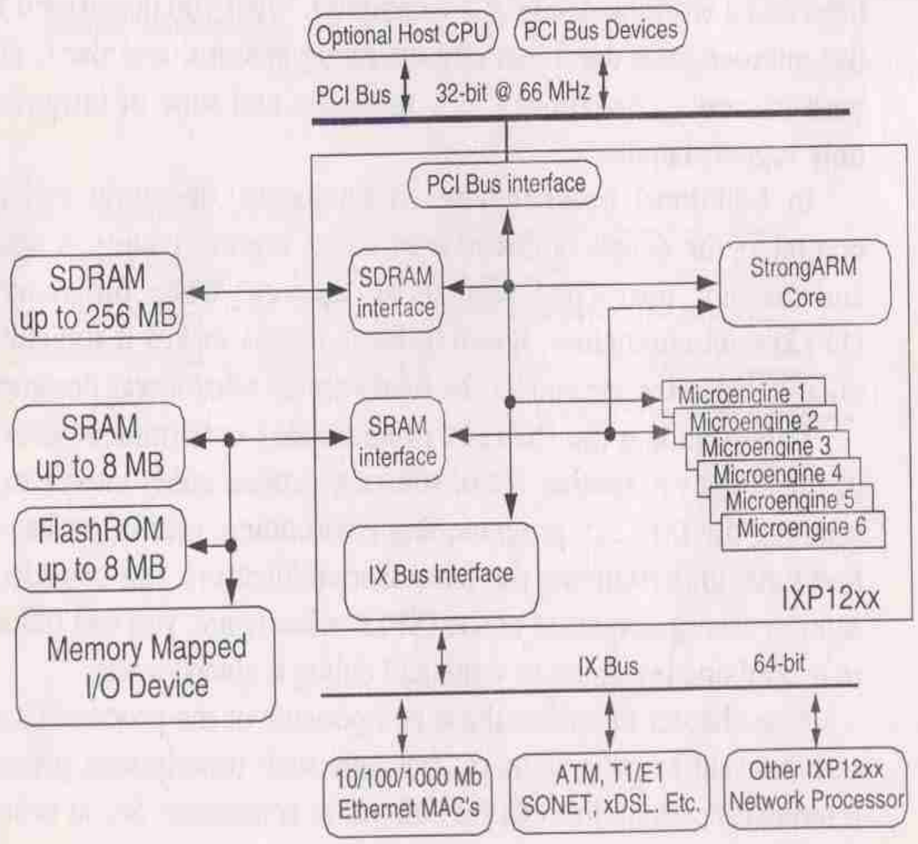


Figure 2.1 The Functional Units of the IXP12xx

Intel IXP 1200 architecture

- StrongARM core – full 32 bit RISC processor with integrated caches .It can be used for management functions, running protocols, exception handling and other tasks
- Microengines -compact and efficient RISC processor engines. They are used for high speed packet inspection, data manipulation and data transfer.
- Scratchpad-small, low latency memory interface to all of the microengines. It is inside chip
- SRAM- off chip, medium latency memory
- SDRAM- can move data to and from the FBI unit without the data going through the microengines. It has high latency

Performance Metrics for classification algorithms

- search speed
- Low storage requirements
- Fast updates – Adding and deleting rules, changing the priority of rule should be easy.

  ‘Preprocessing’ should not be required
- Scalability in dimension
- Flexibility in specifying rules : general rules, prefix, range, operators, wild cards

Bitmap intersection algorithm

- the set of rules, *S*, that match a packet is the intersection of *d* sets, *Si*, where *Si* is the set of rules that match the packet in the *i*th dimension alone & d is the dimension.
- The *d* sets are intersected (a simple bitwise AND operation) to give the set of matching rules, from which the best matching rule is found .

Comparison with other algorithms

Algorithm	Worst-case time complexity	Worst-case storage complexity
Linear search	N	N
Ternary CAM	1	N
Hierarchical tries	W^d	NdW
Set-pruning tries	dW	N^d
Grid-of-tries	W^{d-1}	NdW
Cross-producting	dW	N^d
FIS-tree	$(l + 1)W$	$l \times N^{1+1/l}$
RFC	d	N^d
Bitmap-intersection	$dW + N/memwidth$	dN^2
HiCuts	d	N^d
Tuple space search	N	N

Comparison with other algorithms

- hierarchical tries, Grid of tries – Worst case time complexity proportional to power of d
- Set pruning tries, cross-producting, HiCuts, RFC- Worst case storage complexity proportional to some power of d.
- Tuple space search- does not support wildcards & works only for ranges
- Linear search -poor scaling property
- FIS tree - works only for 2 dimensional classification.
- Bitmap intersection fits our requirements more than other algorithms.
- worst time complexity can further be reduced by parallel processing. ABV algorithm explained next improves its performance still further.

Parallel design mapping

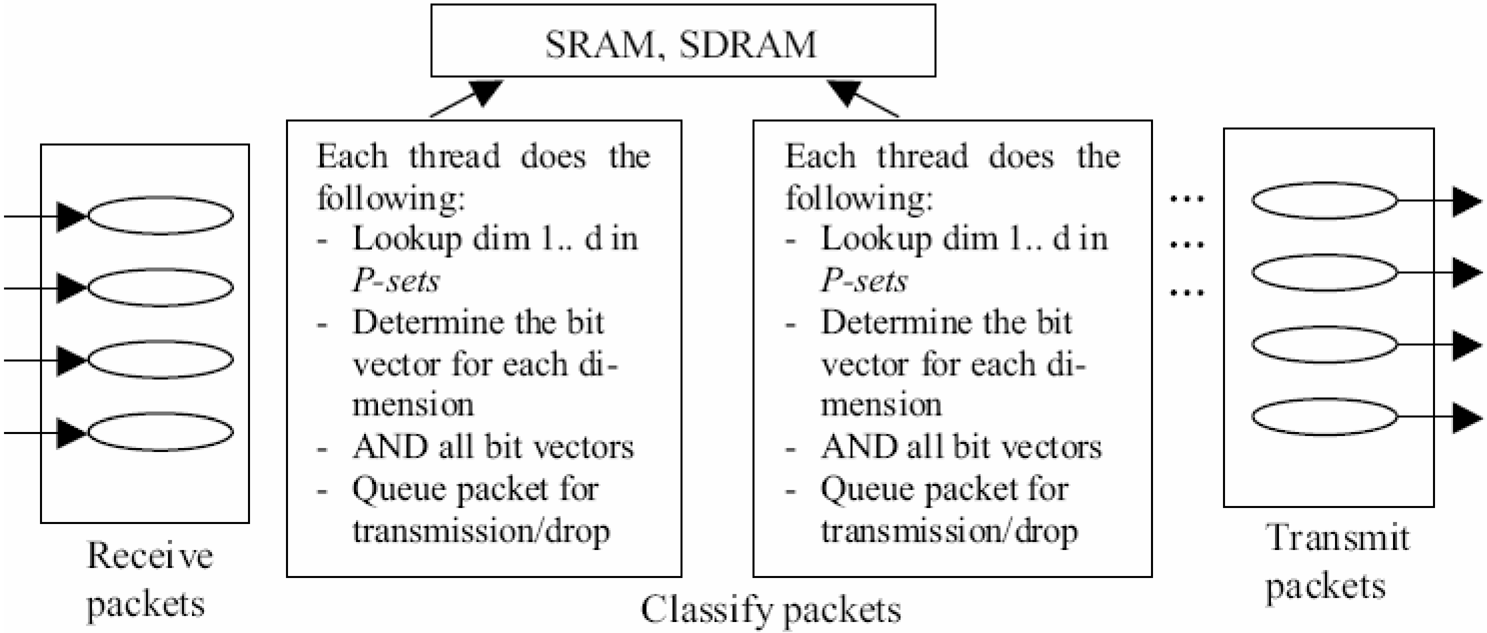


Figure 3: ***Parallel*** **design mapping of the Bit Vector algorithm**

Pipelined design mapping

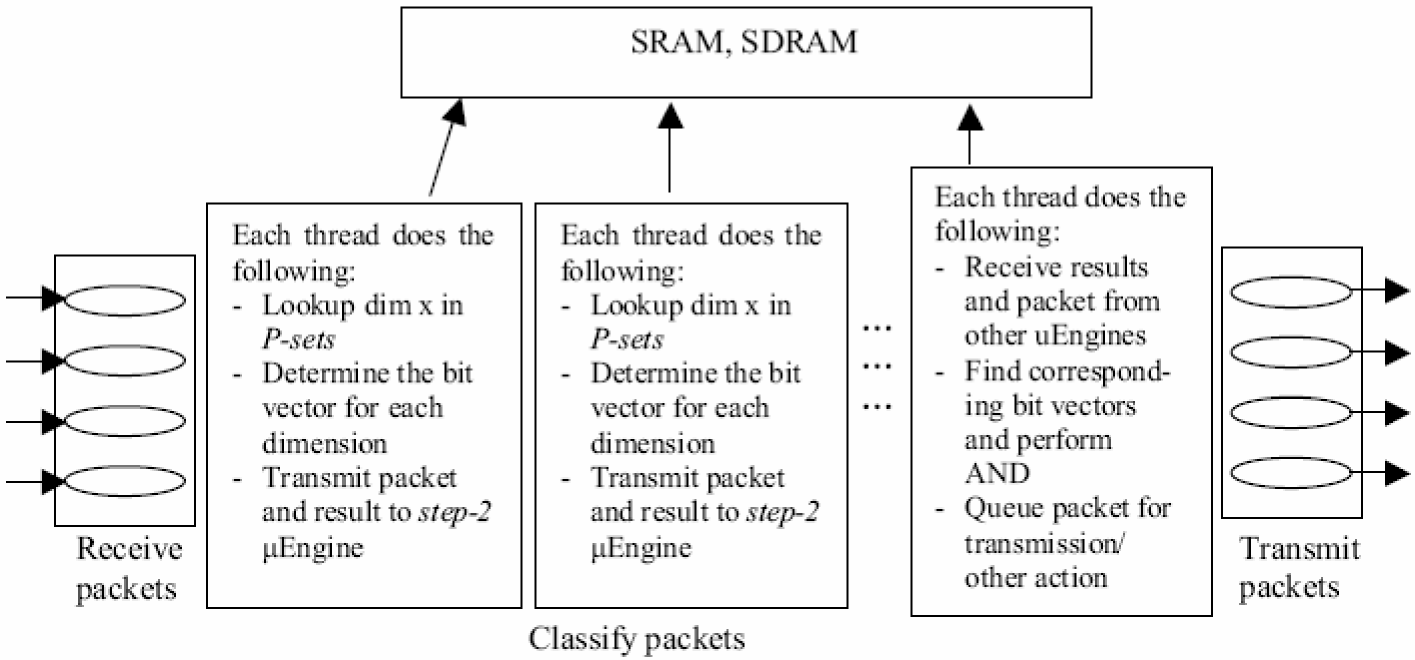
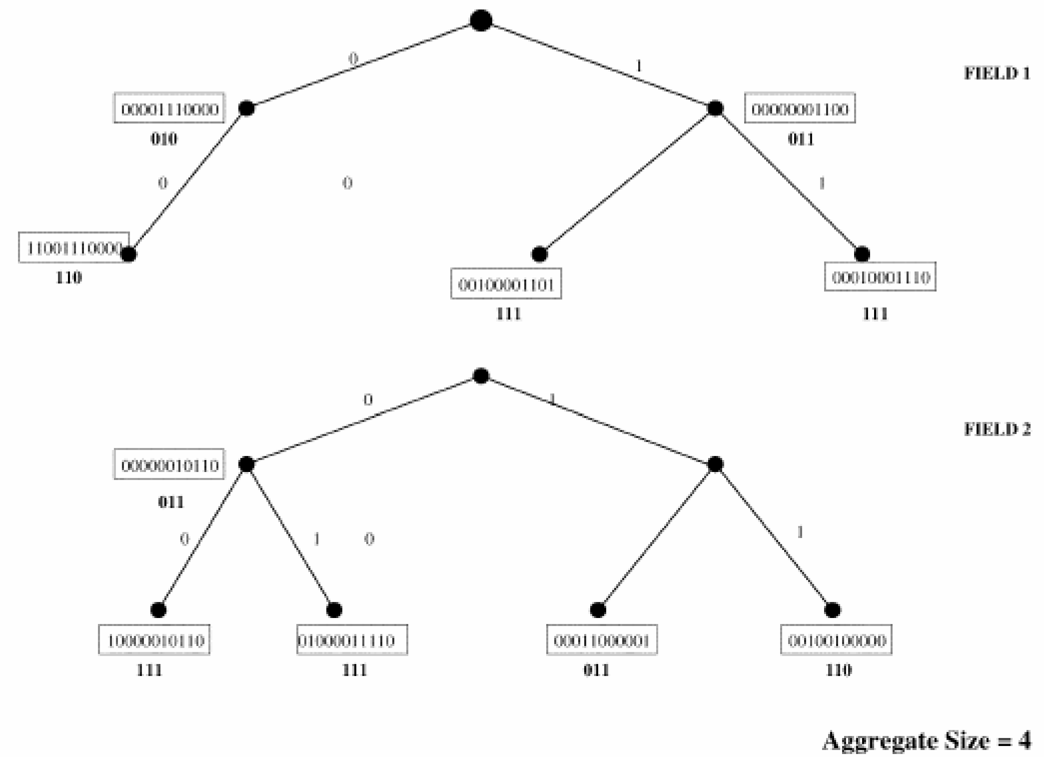


Figure 4: *Pipelined* design mapping of the Bit Vector algorithm

Reasons for choosing parallel design mapping

- the packet processing speed is reduced by 25% in *pipelined* than in parallel.
- In *parallel*, the SDRAM access to read the packet header for classification occurs only once, by a single hardware thread
- three hardware threads on different microengines (1, 2 and 3) to access the packet header in SDRAM for that packet, thus increasing the memory access time

Aggregate Bit Vector Algorithm (ABV)



False matches

$Rule$	$Field_1$	$Field_2$
F_1	X	A_1
F_2	A_1	Y
F_3	X	A_2
F_4	A_2	Y
F_5	X	A_3
F_6	A_3	Y
F_7	X	A_3
$\ldots$	$\ldots$	$\ldots$
$\ldots$	$\ldots$	$\ldots$
F_{60}	A_{30}	Y
F_{61}	X	Y

Why rearrangement of rules can help?

$Rule$	$Field_1$	$Field_2$
F_1	X	A_1
F_2	X	A_2
F_3	X	A_3
$\ldots$	$\ldots$	$\ldots$
F_{30}	X	A_{30}
F_{31}	X	Y
F_{32}	A_1	Y
F_{33}	A_2	Y
$\ldots$	$\ldots$	$\ldots$
F_{60}	A_{29}	Y
F_{61}	A_{30}	Y

Rule Sorting Algorithm

- Arrange-Ent( )
- 1 if (there are no more fields) or (first == last) then return;
- 2 for (each valid size of prefixes) then
- 3 Group together all the elements with the same size;
- 4 Sort the previously created groups.
- 5 Create subgroups made up of elements having the same prefixes on the field col
- 6 for (each subgroup with more than two elements) then
- 7 Arrange-Ent (S.first, S.last, col+1);

Aggregated search

Get Packet $P(H_1, \ldots, H_k)$;
for $i \leftarrow 1$ **to** k **do**
 $N_i \leftarrow LPMNode(Trie_i, H_i)$;
$Aggregate \leftarrow 11\ldots1$;
for $i \leftarrow 1$ **to** k **do**
 $Aggregate \leftarrow Aggregate \bigcap N_i.aggregate$;
$BestRule \leftarrow Null$;
for $i \leftarrow 0$ **to** $sizeof(Aggregate) - 1$ **do**
 if $(Aggregate[i] == 1)$
   for $j \leftarrow 0$ **to** $A - 1$ **do**
    if $(\bigcap_{l=1}^{k} N_l.bitVect[i \times A + j] == 1)$
     if $(R_{i \times A+j}.cost < BestRule.cost)$
      $BestRule = R_{i \times A+j}$;
 return BestRule;

Window based packet filtering (WBPF)

- Assumption:

  1) malicious flows maintains a high flow rate without altering the flow identification

  2) legitimate flows such as TCP traffic are bursty in nature, and there are periods of high and low activities.

Main idea (WBPF)

- The WBPF is turned on only if the aggregated arriving flow rate is larger than the available bandwidth of the output link and the queue experiences packet drops
- When the WBPF is on, a new incoming packet is dropped if a packet from the same flow has arrived in the previous consecutive window since it should be from a high rate flow
- The time-window size is dynamically calculated based on several parameters observed during the current window period.

Estimating flow rate

- New rate=

[(old rate x k)+size of packet just arrived] / [inter arrival time + k]

Where k is some constant

WBPF pseudocode

Algorithm 1 ALGORITHM WBPF

// If no packet has arrived during the window period $s(i)$ while $F = 1$,
// then set $F = 0$ at the end of window i.
// We assume that initially $F = 0$.

when a new packet p **arrives**
begin
    estimate the aggregate arrival rate A;
    if $F = 1$ **then**
        if $(A \leq B$ **or** $Q\_D(\delta) = 0)$ **then**
            set $F = 0$;
        endif
    else **if** $(A > B$ **and** $Q\_D(\delta) = 1)$ **then**
            set $F = 1$;
            set $s(0) = packet\_size_{ave}(\delta)/(\alpha B)$;
        endif
    endif

WBPF pseudocode

if $F = 0$ **then**
    // Do nothing at the filter.
else *// Adopt WBPF, and let the current window be i.*
  if *(not end of current window)* **then**
    add p to $L(i)$;
    if $p \in L(i-1)$ **then**
      drop p;
    endif
  else **if** $Q\_D(i) = 0$ **then**
      if $u(i) < 1$ **then**
        $s(i+1) = s(i) - (1 - u(i)) * s(i)$;
      else $s(i+1) = s(i)$;
      endif
    else $s(i+1) = s(i) + \beta * s(i) * packet\_size_{ave}(\delta) * R_{Q\_D}(i)$;
    endif
  endif
endif
end.

Scope for future work

- **Aggregate Bit Vector Algorithm (ABV)**

  sorting algorithm for rearrangement can be made more efficient by choosing the order in which fields should be sorted out.

- **Window based packet filtering algorithm (WBPF)**

  detection & blocking of single source IP

References:

- [1] "DoS attack"
- http://en.wikipedia.org/wiki/Denial-of-service_attack
- [2] Erik Johnson and Aaron Kunze,"*IXP1200 programming, the microengine coding*
- *guide for the Intel IXP1200 network processor family*", Intel press
- [3] Pankaj Gupta and Nick Mckeown, "*Algorithms for packet classification*", IEEE
- network, Volume 15, Issue 2, March-April 2001
- [4] Deepa Srinivasan and Wu-chang Feng, "*Performance analysis of multi-dimensional*
- *packet classification on programmable network processors",* Proceedings of the
- 29th Annual IEEE International Conference on Local Computer Networks (LCN'04)
- [5] Florin Baboescu and George Varghese, "*Scalable Packet Classification",* IEEE/ACM,
- transactions on networking vol.13, no. 1, February 2005
- [6] Yen-Hung Hu, Hongsik Choi, Hyeong-Ah Choi, "*Packet Filtering for Congestion*
- *Control under DoS Attacks",* Proceedings of the Second IEEE International
- Information Assurance Workshop (IWIA'04)
- [7] Networking basics available on
- http://en.wikipedia.org/wiki/