Algorithms for Advanced Packet Classification with TCAMs
(sigcomm 2005)

Karthik Lakshminarayanan
UC Berkeley

Joint work with
Anand Rangarajan and Srinivasan Venkatachary
(Cypress Semiconductors Inc.)

Modified by Yaxuan Qi, for NSLab Seminar,
Tsinghua Univ. Beijing China
May 5, 2009
Packet Processing Environment

Rule: acl-id src-addr src-port dst-addr dst-port proto
(e.g. acl1231 128.32.0.0/8 0-1023 32.12.1.1/16 1024 tcp)

- Packet matches a set of rules based on the header
- Examples: routers, intrusion detection systems
Ternary Content Addressable Memory

- Memory device with **fixed width** arrays
- Each bit is 0, 1 or x (don’t care)
- Search is performed against all entries in *parallel* and the *first result* is returned

<table>
<thead>
<tr>
<th>TCAM</th>
<th>Width = W bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>row1</td>
<td>00100x1x001110x0x</td>
</tr>
<tr>
<td>row2</td>
<td>01110xxx001100xxx</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>rown</td>
<td>1111101x1101000xx</td>
</tr>
</tbody>
</table>

**Search key**

011101xx001100x10

**Output**

row2

**Width** = W bits
TCAM: Benefits and Disadvantages

- **Benefits:**
  - Deterministic Search Throughput—O(1) search

- **Disadvantages:**
  - Cost
  - Power consumption

- **Current TCAM usage:**
  - 6 million TCAM devices deployed (by 2005)
  - Used in multi-gigabit systems that have O(10,000) rules
  - TCAMs can support a table of size 128K (18Mbits/144bits) ternary entries and 133 million (133M/15M=88Gbps 64B packets) searches per second for 144-bit keys
Range Representation Problem

• Representing prefixes in ternary is trivial
  – IP address prefixes present in rules

• Representing arbitrary ranges is not easy though
  – port fields might contain ranges
    • e.g. sPort [1024, 65536], dPort [6110, 6112]
  – intrusion detection may check packet length field
    • e.g. packet size [1, 254]

• Problem Statement
  – given a range R, find the minimum number of ternary entries to represent R
Why is efficient range representation an important problem?

<table>
<thead>
<tr>
<th>Statistic</th>
<th>1998 database</th>
<th>2004 database</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total number of rules</td>
<td>41190</td>
<td>215183</td>
</tr>
<tr>
<td>With single range field</td>
<td>4236</td>
<td>54352</td>
</tr>
<tr>
<td>(10.3%)</td>
<td>(25.3%)</td>
<td></td>
</tr>
<tr>
<td>With single non-“$\geq 1024$” range field</td>
<td>553</td>
<td>25311</td>
</tr>
<tr>
<td>(1.3%)</td>
<td>(11.8%)</td>
<td></td>
</tr>
<tr>
<td>With two range fields</td>
<td>0</td>
<td>3225</td>
</tr>
<tr>
<td>(0%)</td>
<td>(1.5%)</td>
<td></td>
</tr>
<tr>
<td>Unique ranges in first field</td>
<td>62</td>
<td>270</td>
</tr>
<tr>
<td>Unique ranges in second field</td>
<td>0</td>
<td>37</td>
</tr>
</tbody>
</table>

Number of unique ranges have increased over time
Prefix expansion of ranges:
- express ranges as a union of prefixes
- have a separate TCAM entry for each prefix
- expansion: the number of entries a rules expands to

• Example: the range [3,12] over a 4-bit field would expand to:
  - 0011 (3), 01xx (4-7), 10xx (8-11) and 1100 (12)

• Worst-case expansion for a single W-bit field is $2W-2$
  - example: [1,14] would expand to 0001, 001x, 01xx, 10xx, 110x, 1110
  - 16-bit port field expands to 30 entries
  - F W-bit fields is thus $(2W-2)^F$
Earlier Approaches – II

Database-dependent encoding:

– observation: TCAM array has some unused bits
– use these additional bits to encode commonly occurring ranges in the database

• TCAMs with IP ACLs have ~ 36 extra bits
  – 144-bit wide TCAMs
  – 104-bits + 4-bits for IP ACL rules
Earlier Approaches – II

Database-dependent encoding:
– observation: TCAM array has some unused bits
– use these additional bits to encode commonly occurring ranges in the database

• Example:

<table>
<thead>
<tr>
<th>Address</th>
<th>Port</th>
<th>…</th>
<th>Set extra bit to 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.123.0.0/16</td>
<td>20-24</td>
<td>…</td>
<td></td>
</tr>
<tr>
<td>32.12.13.0/24</td>
<td>1024-</td>
<td>…</td>
<td>Set extra bit to x</td>
</tr>
<tr>
<td>128.0.0.0/8</td>
<td>20-24</td>
<td>…</td>
<td>Set extra bit to 1</td>
</tr>
</tbody>
</table>

If search key falls in 20-24, set extra bit to 1, else set it to 0
Earlier Approaches – II

Database-dependent encoding:
- observation: TCAM array has some unused bits
- use these additional bits to encode commonly occurring ranges in the database

- Disadvantages:
  - extra bits is limited
  - number of unique ranges is increasing
  - incremental update is hard
  - ...
  - all due to: database dependency
Database-Independent Range Pre-Encoding

- **Key insight:** use additional bits in a *database independent* way
  - wider representation of ranges
  - reduce expansion in the worst-case
Database-Independent Range Pre-Encoding

- **Fence encoding** \((W \text{ bits})\):
  - total of \(2^W - 1\) bits
  - encoding of \(i\) has \(i\) ones preceded by \(2^W - i - 1\) zeros
  - e.g. \(W=3\), \(f(0) = 0000000\), \(f([1, 3]) = 0000xx1\)
- **With** \(2^W - 1\) bits, fence encoding achieves an expansion of 1

<table>
<thead>
<tr>
<th>Range</th>
<th>Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>(= i)</td>
<td>(0^{2^W - i - 1}1^i)</td>
</tr>
<tr>
<td>(\geq i)</td>
<td>(x^{2^W - i - 1}1^i)</td>
</tr>
<tr>
<td>(&lt; i)</td>
<td>(0^{2^W - i}x^{i - 1})</td>
</tr>
<tr>
<td>([i, j])</td>
<td>(0^{2^W - 1 - j}x^{j - i}1^i)</td>
</tr>
</tbody>
</table>

**Theorem:** For achieving a worst-case row expansion of 1 for a \(W\)-bit range, \(2^W - 1\) bits are necessary
DIRPE: Using the Available Extra Bits

- Two extremes:
  - no extra bits $\Rightarrow$ worst case expansion is $2W-2$
  - $2^W-W-1$ extra bits $\Rightarrow$ worst case expansion is 1

- Is there something in between?
  - appropriate worst-case based on number of extra bits available
Database-Independent Range Pre-Encoding

• Procedure:
  – split W-bit field into multiple chunks
  – encode each chunk using fence encoding
  – “combine” the chunks to form ternary entries

\[
\begin{array}{c|c|c}
\hline
k_0 \text{ bits} & k_1 \text{ bits} & k_2 \text{ bits} \\
\hline
\end{array}
\]

W bits

Combining chunks: analogous to multi-bit tries
Unibit view of DIRPE (Prefix expansion)

- \( W=3 \) divided into 3 one-bit chunks
- \( R=[1,6] \)—prefixes = \{001,01x,10x,110\}
- Each level can contribute to at most 2 prefixes (but the top level)
Multi-bit view of DIRPE

Width of each encoded chunk = $2^3-1 = 7$ bits

- 9-bit field ($W=9$)
- 3 chunks, 3 bits wide
- Range = $[11, 54] = [013, 066]$  

Worst case expansion  
$= 2W/k - 1$

Number of extra bits needed  
$= (2^{k-1})W/k - W$
## Comparison of Expansion

<table>
<thead>
<tr>
<th>Extra bits</th>
<th>DIRPE</th>
<th>Region-based Range Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>8</td>
<td>15</td>
<td>30</td>
</tr>
<tr>
<td>18</td>
<td>11</td>
<td>16</td>
</tr>
<tr>
<td>27</td>
<td>9</td>
<td>14</td>
</tr>
<tr>
<td>44</td>
<td>7</td>
<td>12</td>
</tr>
</tbody>
</table>

**Worst-case expansion**

<table>
<thead>
<tr>
<th>Extra bits</th>
<th>DIRPE</th>
<th>Region-based Range Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2.69</td>
<td>2.69</td>
</tr>
<tr>
<td>8</td>
<td>2.08</td>
<td>2.33</td>
</tr>
<tr>
<td><strong>18</strong></td>
<td><strong>1.79</strong></td>
<td><strong>2.17</strong></td>
</tr>
<tr>
<td>36</td>
<td>1.57</td>
<td>1.58</td>
</tr>
</tbody>
</table>

**Real-life expansion**

DIRPE + DB-dependent $\Rightarrow$ Net expansion was 1.12
<table>
<thead>
<tr>
<th>Metric</th>
<th>Prefix Expansion</th>
<th>Region-based Encoding (with $r$ regions)</th>
<th>DIRPE (with $k$-bit chunks)</th>
<th>DIRPE + Region-based Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extra bits</td>
<td>0</td>
<td>$F(\log_2 r + \frac{2n-1}{r})$</td>
<td>$F(\frac{W(2^{k-1})}{k} - W)$</td>
<td>$F(\frac{(2^{k-1}) \log_2 r}{k} + \frac{2n-1}{r})$</td>
</tr>
<tr>
<td>Worst-case capacity degradation</td>
<td>$(2W-2)^F$</td>
<td>$(2\log_2 r)^F$</td>
<td>$(\frac{2W}{k} - 1)^F$</td>
<td>$(\frac{2\log_2 r}{k})^F$</td>
</tr>
<tr>
<td>Cost of an incremental update</td>
<td>$O(W^F)$</td>
<td>$O(N)$</td>
<td>$O((\frac{W}{k})^F)$</td>
<td>$O(N)$</td>
</tr>
<tr>
<td>Overhead on the packet processor</td>
<td>None</td>
<td>Pre-computed table of size: $O(\frac{(\log_2 r+ \frac{2n-1}{r}) F.2^w}{r}\text{ or }O(nF)\text{ comparators of width }W\text{ bits}$</td>
<td>$O(\frac{W.2^k}{k})\text{ logic gates}$</td>
<td>Both pieces of logic from previous two columns</td>
</tr>
</tbody>
</table>
DIRPE: Summary

↑ Database independent
↑ Scales well for large databases
↑ Good incremental update properties

↓ Additional bits needed
↓ Small logic needed for modifying search key
Related Work I

- **Range-to-prefix conversion**
  - Represent a range by a set of prefixes, each of which can be stored by a single TCAM entry. *(V. Srinivasan, G. Varghese, S. Suri, and M. Waldvogel, “Fast and scalable layer four switching,” in ACM SIGCOMM, Sep. 1998, pp. 191–202.)*
  - The worst-case expansion ratio is $2W-2$, in a single dimension.
  - A single rule can generate up to 900 prefixes (only for the two port fields).
  - Prefix expansion may increase the number of required TCAM entries by a factor of more than 6.

- **Direct hardware solution**
  - Extended TCAMs, implements range matching directly in hardware. *(E. Spitznagel, D. Taylor, and J. Turner, “Packet classification using extended TCAMs,” in ICNP, 2003.)*
  - Reducing power consumption by over 90% relative to standard TCAM
  - Will *not* be accomplished in the near future
Related Work II

- Database-dependent range encoding algorithms
  - Encoding is a function of the distribution of ranges in the database
  - Basic idea: a single extra bit is assigned to each selected range \( r \) in order to avoid the need to represent \( r \) by prefix expansion
    - the number of unique ranges in today's databases is ~300
    - we have ~30 extra bits…
  - Region Partition: split a range into multiple sub-ranges. Each such sub-range is encoded by two numbers: the region number into which it falls, and the sub-range number within that region. (H. Liu, “Efficient mapping of range classifier into ternary-cam,” in Hot Interconnects, 2002.)
  - Layered Interval Coding (LIC): a more efficient representations based on the observation that, sets of disjoint ranges may be encoded much more efficiently than sets of overlapping ranges. (Anat Bremler-Barr, David Hay, Danny Hendler, Beer-Sheva and Boris Farber, “Layered interval codes for tcam-based classification, INFOCOM 2009.”)
Related Work III

• Database-independent range encoding algorithms
  – Encoding of a specific range does not change across different databases.
  – Grey coding: based on the observation that small ranges, which occur frequently in real-world databases, are encoded more efficiently. (A. Bremler-Barr and D. Hendler, “Space-efficient team-based classification using gray coding,” in IEEE INFOCOM, 2007, pp. 1388–1396.)
Thanks!

Q & A