Internet Routing
Seminar in Distributed Computing
with papers chosen by Prof. Bernhard Plattner

Erich Schär
Challenges for Internet Routing

- Internet initially designed as a small network for researchers

- More and more users
  - Cheap computers
  - Mobile devices
Increasing Internet Traffic

- 160 TB per second today
  - Growth rates per year 50% to 100%

- Drivers
  - Video applications like youtube
  - P2P filesharing
  - Webradio
  - Voice applications like Skype / TeamSpeak

- More traffic from more endpoints
  - Throughput hard to handle efficiently
Packet Scheduling

- Idea has been around for a long time

Possible applications:
- Give high priority to important packages and delay less important packages
IPv6

- IPv4 address space limited

- IPv6 also offers nice features
  - Protocol optimized for routers
  - IPsec
  - Multicast

- Proposed in 1995

- Still not widely used in practice
Overview

1. Scalable High Speed IP Routing Lookups
2. A Software Architecture for Routers
Scalable High Speed IP Routing Lookups

Marcel Waldvogel, George Varghese, Jon Turner, Bernhard Plattner

In ACM SIGCOMM Proceedings, 1997
Motivation

Three limiting Factors:

- Link Speed
- Router Data Throughput
- Packet Forwarding Rates
Refresher: Routing

User → Router 1 → Router 2 → Server

1 0

1 0 1 0
Refresher: Longest Prefix Matching

<table>
<thead>
<tr>
<th>Routing Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 0 1</td>
</tr>
<tr>
<td>1 1 0 1 0 1 0 1</td>
</tr>
<tr>
<td>1 1 1 0 1</td>
</tr>
<tr>
<td>0 1 0 1 0 0</td>
</tr>
</tbody>
</table>

| 1 0 0 1 1 1 1 1 0 |
Increasing Routing Table Size

- 140 000 to 160 000 entries in the routing table today

- Increasing
Existing Approaches

- Trie based schemes
- Hardware solutions
- Protocol based solutions
- Caching
Trie

Example trie for longest prefixes:
- 0 0 0 0
- 0 1
- 1 0

Example:
- Search longest prefix for IP 01001111

Runtime:
- Worst case runtime linear in length of address size O(w)
Algorithm: Hashing

- Needs a *precise* Key

<table>
<thead>
<tr>
<th>Hash Table for Length 1</th>
<th>Hash Table for Length 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 0</td>
</tr>
<tr>
<td>0</td>
<td>1 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hash Table for Length 3</th>
<th>1 0 0 1 1 1 1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 1</td>
<td></td>
</tr>
<tr>
<td>1 1 0</td>
<td></td>
</tr>
<tr>
<td>0 0 0</td>
<td></td>
</tr>
<tr>
<td>0 0 1</td>
<td></td>
</tr>
</tbody>
</table>
Algorithm: Hashing

- Search every hash table in constant time
- Worst case: have to search every hash table
  \[ \rightarrow O(w) \]

- Can we do better?
Binary Search

\[1\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ ?\]
Binary Search

1 1 1 1 0 0 1 0

?
Binary Search

<table>
<thead>
<tr>
<th>Increasing Prefix Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>

**Binary Search**

- Length 1: 1
- Length 2: 11
- Length 3: 111

**Hash Tables with Marker**

- Length 1: 0
- Length 2: 00
- Length 3: ???
Runtime Analysis

- Every hash look-up in constant time
- At most $O(\log(w))$ hash look-ups
  - At most 7 hash computation for a 128 bit IPv6 Address
Optimizations

- Adopt to actual data to improve average case behavior
- Some prefix lengths looked up more often
- Some prefix lengths never looked up
- Optimizations do not improve worst case behavior
Idea: Remove “empty” Hash Tables from the Search Tree

- Remove not used hash tables from the search tree
- Rebalance the tree
- Average case gets worse, since frequently looked up hash tables are further down the tree
Changing the Tree

- Reorder the tree to have frequently accessed hash tables on top
- Improve average search time
- Worst case still $O(\log(w))$
Changing the Tree

- Even better average case
- Worst case $O(w)$
Mutating Binary Search

- Observation: For a given matching prefix of length 16, the longest matching prefix usually has only very few possible lengths.

- E.g. if we do a look up for 10111100.01011101.00011101.10001110 we find a matching prefix 10111100.01011101

- We know from statistics, that the longest matching prefix for any IP address starting with the prefix 10111100.01011101 either has length 24 or 26.
Mutating Binary Search

- Use this information to adopt search based on the matching 16 bit prefix.

- Store information on where to continue search in markers.
Mutating Binary Search

### Mutating Search Trees

<table>
<thead>
<tr>
<th>Tree 1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Hash Tables

<table>
<thead>
<tr>
<th>E: ..., Tree 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>F: ...111, Tree 3</td>
</tr>
<tr>
<td>H: ...101, Tree 4</td>
</tr>
<tr>
<td>J: ...1010, End</td>
</tr>
<tr>
<td>G: ...11100, End</td>
</tr>
</tbody>
</table>

**Structure of Hash Table Entry:**

- Name of this node
- Prefix ("..." is the 16 bit prefix corresponding to E)
- Tree to use from now on
Mutating Binary Search

- Typically for a given prefix of length 16 only very few possible length of the longest matching prefix.

<table>
<thead>
<tr>
<th>Distinct Lengths</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4977</td>
</tr>
<tr>
<td>2</td>
<td>608</td>
</tr>
<tr>
<td>3</td>
<td>365</td>
</tr>
<tr>
<td>4</td>
<td>249</td>
</tr>
<tr>
<td>5</td>
<td>165</td>
</tr>
<tr>
<td>6</td>
<td>118</td>
</tr>
<tr>
<td>7</td>
<td>78</td>
</tr>
<tr>
<td>8</td>
<td>46</td>
</tr>
<tr>
<td>9</td>
<td>35</td>
</tr>
<tr>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>11</td>
<td>9</td>
</tr>
<tr>
<td>12</td>
<td>3</td>
</tr>
</tbody>
</table>
Results

- Algorithm for longest prefix matching
- $O(\log(w))$
- Evaluation (on a 200MHz Pentium Pro)
  - 80 ns for IPv4 lookup
  - 200 ns for IPv6 lookup
- Router throughput no longer limited by the speed of the lookup
Router Plugins
A Software Architecture for Next Generation Routers

Dan Decasper, Zubin Dittia, Guru Parulkar, and Bernhard Plattner

In ACM SIGCOMM Proceedings, 1998
Today's Routers

Routers today:
- Monolithic operating systems
- Not easily upgradable and extensible
Design Goals

- Modularity
- Extensibility
  - New Protocols (IPv6)
  - Security
- Flexibility
- Performance
Plugins

- Modules
  - Packet scheduler
  - Longest prefix matching

- Add and configure at runtime

- Run in kernel space
Gates

- Possible entry points for plugins in the IP core

Examples:
- IPV6 option processing
- IP security
- Packet scheduling
- Best-matching prefix algorithm
Flow / Filter

- Filter used to bind packet flow to a plugin instance

- Six-tuple
  - Source address
  - Destination address
  - Protocol
  - Source port
  - Destination port
  - Incoming interface

- Wildcards allowed
Association Identification Unit

- Binding of plugin instances and filters
- Performance critical
- Directed acyclic graph (DAG)

<table>
<thead>
<tr>
<th>#</th>
<th>Source Address</th>
<th>Destination Address</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>129.*</td>
<td>192.94.233.10</td>
<td>TCP</td>
</tr>
<tr>
<td>2</td>
<td>128.252.153.1</td>
<td>128.252.153.7</td>
<td>UDP</td>
</tr>
<tr>
<td>3</td>
<td>128.252.153.1</td>
<td>128.252.153.7</td>
<td>TCP</td>
</tr>
<tr>
<td>4</td>
<td>128.252.153.*</td>
<td>*</td>
<td>UDP</td>
</tr>
</tbody>
</table>
Example
Results

- Implemented in NetBSD
- Efficient (only 8% slower than best effort)
- Extensible
- Flexible
Conclusions

- Short overview over some important routing problems
  - Increasing number of users
  - Large traffic
  - Inflexible routers

- Algorithm for faster routing
  - A fast algorithm with good asymptotic runtime
  - Various optimizations to typical lookups

- Powerful software routing architecture
  - Can easily be adapted to new problems with routing
  - Very efficient
Questions?