Reading for this week

- Sorry I forgot to put these up before I go – but no critiques required

- IP lookup and IPnG
  - Survey and Taxonomy of IP Lookup Algorithms (no critique, but please take a look)
  - IPNL paper
  - The case for IPv6

- I also put up the multi-cast reading set, with one critique required
Two Classes Ago

- Discussion of Router design

- Some important concepts:
  - Data plane vs. control plane
    - Adding functionality to the data plane is difficult
    - Implications on services such as multicast, QoS and classification?
  - Router organization; modern routers VoQ/IQ, OQ and switched fabric
Keshav and Sharma Paper

Issues and Trends in Router Design
Components of a Router

- Input Queueing vs. Output Queueing
- Switch fabric (busses, crossbars, shared memory, switched network)

Figure 1. Architecture of a router.
Fast Path (Normal Case)

- Receive packet from input link
- Lookup packet destination (IP lookup)
- Validate checksum, update TTL/checksum
- Buffer packet in input queue
- Send packet to output interface through switching fabric (multicast?)
- Buffer packet in output queue
- Send it
Router Types

• Backbone routers: routers at the core of the Internet
  – Desire speed (to scale with traffic and routing tables)
  – ... and reliability
  – at any cost

• Enterprise routers (e.g., campus or company network)
  – Primary goal is connectivity of a large number of endpoints cheaply
  – Low cost per port for large number of ports

• Access routers: links small business or homes to ISP
  – Increasing load with broadband technologies
  – Many protocols supported by the router
Backbone Router Traffic

Figure 2. a) Packet size distribution in a backbone router (data courtesy MCI, 25 June 1997, OC-3 backbone link); b) number of active flows in a backbone router.
Advances and Trends

- High speed router lookup
  - We will discuss using the IP-lookup paper
  - But other approaches as well (e.g., MPLS)

- Advances in switching fabrics (cell-based vs. packet-based)

- Advances in Input and Output queueing (discussed last time)

- Router Operating Systems
  - In the past, routers viewed as hardware devices
  - Router OS
    * Provides a carefully controlled API to allow customization of functionality and policy
Open Problems

- Flow Identification
- Resource Reservation
- Ease of Configuration
- Stability of large systems
- Accountability
One development since: Network Processors

- Would like flexible routers and edge devices
- But also fast
- Options:
  - Custom hardware (ASIC): fast but inflexible
  - General purpose processors: flexible but slow
  - Programmable Hardware (FPGA)...maybe?
  - Network Processors
Network Processors

- What if we design a programmable processor optimized to network operations
- Use parallelism and pipelining at the packet level
- Support for lookup, security, classification ...
- Can be used as building block for routers as well as other networking devices
- Very hot area for a while
  - Over 50 companies offering products, including Intel (big player here) and IBM
Li, Alderson, Willinger and Doyle – SIGCOMM 2004

A First-Principles Approach to Understanding the Internet’s Router-Level Topology
Overview

• The aim of this paper is to provide insight into the router level structure of the Internet
  – Have researchers and developers work against more realistic assumptions
  – Proves that existing topology generators are significantly flawed
    * Unrepresentative networks
    * Poor performance networks
• Our interest is in terms of the getting a closer look at the structure of the Internet
Power Law Structure of the Internet

• It has long been known that the Internet exhibits “power-law” distribution
  – Heavy tail distribution
  – Most routers have a small degree; a small number have a very large degree

• Assumption has been that the core is heavily connected, with high connectivity nodes connected to each other
  – This paper uses first-principles (examination of technology trends) to show that the above assumption is not correct
Technology Constraints

Figure 1: Technology constraint for Cisco 12416 Gigabit Switch Router (GSR): degree vs. bandwidth as of June 2002. Each point on the plot corresponds to a different combination of line cards and interfaces for the same router. This router has 15 available line card slots. When the router is configured to have less than 15 connections, throughput per degree is limited by the line-card maximum speed (10 Gbps) and the total bandwidth increases with the number of connections, while bandwidth per degree remains the same (dash-dot lines). When the number of connections is greater than 15, the total router bandwidth and bandwidth per degree decrease as the total number of connections increases (solid lines), up to a maximum of 120 possible connections for this router (dotted line). These three lines collectively define the feasible region for configuring this router.
Figure 2: Aggregate picture of router technology constraints. In addition to the Cisco 12000 GSR Series, the constraints on the somewhat older Cisco 7000 Series is also shown. The shared access technology for broadband cable provides service comparable to DSL when the total number of users is about 100, but can only provide service equivalent to dialup when the number of users is about 2000. Included also is the Linksys 4-port router, which is a popular LAN technology supporting up to 5 100MB Ethernet connections. Observe that the limits of this less expensive technology are well within the interior of the feasible region for core network routers.
Economic Picture

Figure 3: Aggregate picture of end user connection bandwidths for the Internet. Most users of the current Internet have relatively slow (56Kbps) connections, while only a relative few have high speed (10Gbps) connections.
Heuristically Optimal Networks

- Mesh like structure at the core of the network
  - Low connectivity routers (to stay in the feasible region)
  - Heavily aggregated traffic

- Tree like structure near the edge
  - High connectivity routers to end users (economics)
  - Aggregate traffic towards mesh
Abilene Network

- Internet 2 provider (not quite a tier 1)
- Core routers located in 11 cities
- Carries 1% of traffic in the US
- Sparsely connected mesh providing minimal amount of redundancy
- Degree 5 for LA router to 12 for NY router
- Peering connections to other Internet 2 sites; no transit
CENIC and Abilene Networks

Figure 4: CENIC and Abilene networks. (Left): CENIC backbone. The CENIC backbone is comprised of two backbone networks in parallel—a high performance (HPR) network supporting the University of California system and other universities, and the digital California (DC) network supporting K-12 educational initiatives and local governments. Connectivity within each POP is provided by Layer-2 technologies, and connectivity to the network edge is not shown. (Right): Abilene network. Each node represents a router, and each link represents a physical connection between Abilene and another network. End user networks are represented in white, while peer networks (other backbones and exchange points) are represented in gray. Each router has only a few high bandwidth connections, however each physical connection can support many virtual connections that give the appearance of greater connectivity to higher levels of the Internet protocol stack. ESnet and GEANT are other backbone networks.
CENIC Network

- Typical of regional ISP
- Can see the same effect with loosely connected mesh and high speed links at the core
- The rest of the paper uses the above observations to question the assumptions made by topology generators
- Different ways of generating networks that have power-law distribution
  - The show that approaches such as Preferential Attachment and PLRG are not consistent with the heuristically optimal structure
Figure 6: Five networks having the same node degree distribution. (a) Common node degree distribution (degree versus rank on log-log scale); (b) Network resulting from preferential attachment; (c) Network resulting from the GRG method; (d) Heuristically optimal topology; (e) Abilene-inspired topology; (f) Sub-optimally designed topology.
Lookup

- Packet header examined to get destination address
- Longest match lookup applied to figure out output link
  - Routing tables are getting big
  - Lookup is only one of the bottlenecks in the router, but an important one
- configure switching fabric to transfer from input to output link
- Must be able to do millions of lookups per second to match link speed
Longest Prefix Match

- Consider the following table entries:

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>128.226.*</td>
<td>A</td>
</tr>
<tr>
<td>128.226.123.*</td>
<td>B</td>
</tr>
<tr>
<td>128.*</td>
<td>C</td>
</tr>
<tr>
<td>128.200.*</td>
<td>A</td>
</tr>
<tr>
<td>128.210.*</td>
<td>A</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

- Multiple matches possible for a destination

- Many memory lookups needed to find best match – bad
One Possibility – Change/Alter the Protocol

- Use VC, single lookup of circuit number

- Tag switching: add a pointer in packet header as an index into next hop’s forwarding table
  - Pointers exchanged by routing protocol

- Both require significant changes to architecture
Basic Tries

- Organize addresses as a trie data structure
- How is prefix match accomplished?
  - Walk the trie, keeping the best match found so far
Discussion

• What is the criteria to evaluate a lookup data structure/algorithm?
  – Lookup speed
  – Space efficiency – why?

• Additional criteria
  – Data structure construction/update time
  – Control plane function, not too important

• How do basic tries perform according to each of these criteria?
Improvement I – Path compression

- Linear portions of the trie which have a single alternative – compress
- Only keep nodes where alternatives are possible
- Provide skip count to indicate how many bits have been compressed
- Must keep info to allow check of the compressed path; backtracking needed
Prefix Expansion

• Idea: reduce the different possible prefix lengths by expanding some prefixes

• How is this useful? We’ll see in a second

• Expansion rule:
  – Expand: X.* — A into (X0.* — A and X1.* — A)
  – Capture: if a node for an expanded prefix exists, discard it
Improvement: disjoint prefix

- Idea: push prefix nodes to the leaves
  - Prefixes do not overlap anymore; prevents backtracking
  - Can be used to reduce the number of length alternatives of prefixes
  - Any other advantages? Any disadvantages?
- Requires expanding some prefixes
Basic Alternatives

- Tries provide space efficient tables, but require many lookups (even with compression)
  - Good space efficiency, but bad lookup speed
- Full expansion: expand all prefixes to maximum length
  - Good lookup speed (one table lookup), but terrible space efficiency
- Optimized schemes try to find balance between these two alternatives
  - Many schemes out there, many are in the survey paper we read
- Hashing: try to replace some lookups with hash computation
Multibit Tries

- Instead of examining the address one bit at a time, look at a number of bits at the same time ("stride")
  - Requires expanding some prefixes to match the "stride" of the search
  - What if the stride is 1? What if the stride is 32?
  - How do we determine the stride?
Improvement: Disjoint Prefix Multi-bit

Figure 14. A disjoint-prefix multibit trie.
Stanford Hardware Scheme

- Observation: most prefixes have length of 24 or less
  - Use full expansion to length 24 of all prefixes
    * Requires a table of size $2^{24}$ entries
  - Have a spill-over table for prefixes that are bigger than 24
  - Fully expand these to 32-bits

- Anything new conceptually/algorithmically?
Stanford Scheme

Figure 15. The hardware scheme of [8].
Level Compressed Tries

- Problem: how to decide what stride to use to balance storage space and overhead?
- LC: start with a binary trie
  - Find the deepest full sub-trie starting from the root
  - Replace the sub-trie with a multi-bit trie matching the depth
  - Apply recursively to the subtries rooted the leaves of the compressed subtrie
- What do you think?
  - What do we gain? What do we lose?
  - At the mercy of the statistical properties of our prefixes?
Improvement – Partially full LC

- Idea: what if sub-trie is 99% full? LC will not compress it
  - Can expand prefixes to fill sub-trie

- Define a fill factor (percentage of the subtrie that has to be full) up to which you will compress the level

- Basic LC is this scheme with fill factor = 1
Alternative – Multibit Trie Compression

• If we expand a big subtrie, we end up with a large number of consecutive trie leaves matching the same entry

• Try to compress the resulting data structure

• One approach: fully expand the table, then compress subtries that are identical
  – Does not result in a small enough table

• Lulea scheme is better, but only high-level details discussed
Example

Figure 17. A full expansion parallel compression scheme.
Discussion of Multibit Tries

• Would like to control the depth of the tree (lookup speed)
• But would like to reduce the storage requirements (cache capacity/hit rate)
• Big decision is to determine the stride size
  – LC, let the structure of the tree determine stride
  – Unfortunately, this reduces space, but not necessarily lookup time
  – Srinivasan et al use dynamic programming to derive optimal stride size to minimize storage needs given a worst-case bound on number of lookups
• Updates are really expensive (especially so for the ones that adapt the stride size)
Alternative – Search on Prefix Length

- Consider this approach: separate prefixes into different tables based on length
  - Use hashing to search each table, starting with the one for longest prefix
  - Stop when you find a match

- Problem: need to search the tables sequentially – many lookups in the worst case
Improvement – Binary Search on Prefix Length

- Try to eliminate half of the candidate lengths with each lookup

- When we look in a table, if there is a match, then we only need to look in tables with longer length

- Problem: if there is no match, what does that mean?
  - Use markers to direct the search
  - Markers can mislead, requiring backtracking
Yet Another Alternative – Prefix Range Search

- Expanded tries have a lot of redundancy
  - A prefix 1100* for an 8-bit trie represents the range 11000000 to 11001111
  - Instead of storing every entry, store range

- Problem: ranges are overlapping
Binary Range Search

Figure 19. Binary range search.
Range Search Tree

![Diagram of a Range Search Tree](image)

*Figure 20. A basic range search tree.*
Evaluation of Schemes

- Paper presents complexity analysis as well as experimental evaluation of the schemes

Figure 22. Lookup time distributions of several lookup mechanisms.