Administrivia

- Second Midterm 11/19
- Project help session Friday at 10 am, Enginet Room J-15
- Last time: started lookup
- Today
  - Finish Lookup
  - IPnG: Redesign (e.g., IPv6) or Retrofit (e.g., IPNL)
Basic Tries

- Organize addresses as a trie data structure
- Improvements: Path compression; Expansion for disjoint prefix; Multi-bit tries (LC tries, partially-full, disjoint prefix); expand and compress
- Stanford Hardware Scheme
- Tradeoff space?
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 tables, 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

![Diagram of Binary Range Search](image)

*Figure 19. Binary range search.*
Range Search Tree

Figure 20. A basic range search tree.
Paper presents complexity analysis as well as experimental evaluation of the schemes
IP Next Generation

• IPv4 has been in use since the 1970s.
  – Developed for a different time and place
  – Attempted to be scalable, but could not imagine the scale of success of the internet

• Address Depletion primary driver
  – IP address space is 32-bits, or 4 billion
    * 6 billion people on the planet
    * Less than 10 years worth of addresses left at current growth rate (2010 expected crisis time)
    * What happens when we run out of addresses?
IP Next Generation

- Need became apparent, many people started developing proposals for next generation IP
  - Address depletion not the only driver; would like to build in some other features into IP (e.g., multicast)

- Meanwhile, NAT emerged, CIDR started getting used

- Big Tussle: Retrofit (fix/extend) IPv4, or replace it with something completely new

- We read two papers, one from each camp
IPv6 Timeline

- 1990: Something is going to break (addresses are running out)
- 1992: CIDR adopted, proposals solicited for IPng (IP next generation)
  - Several proposals received, including some NAT based
- 1994: IPv6 draft is selected
  - What happened to IPv5??
- Meanwhile, IPv6 is being developed; address space is running out; NAT is prevalent
- Peer to Peer, always-on applications, personal networks
IPv6 – why

- Address depletion is the main driver; but
  - Since we are taking the painful decision to upgrade IP, might as well fix everything that is wrong with it
  - Address routing table growth (approx. 100,000 entries in backbone)
  - Easier to configure/use
    * Makes administration easier; supports stateless configuration; good for mobility
  - Simplify packet processing
  - Multicast
  - Security
  - Quality of Service
  - Real time traffic support
  - Flow identification
  - IP billing
  - ...

- A lot of what is “optional” in IPv4 is required in IPv6
## IPv6 Header

<table>
<thead>
<tr>
<th>Field</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>4</td>
</tr>
<tr>
<td>TrafficClass</td>
<td>4</td>
</tr>
<tr>
<td>FlowLabel</td>
<td>4</td>
</tr>
<tr>
<td>PayloadLen</td>
<td>16</td>
</tr>
<tr>
<td>NextHeader</td>
<td>8</td>
</tr>
<tr>
<td>HopLimit</td>
<td>8</td>
</tr>
<tr>
<td>SourceAddress</td>
<td></td>
</tr>
<tr>
<td>DestinationAddress</td>
<td></td>
</tr>
</tbody>
</table>

### Next header/data

<table>
<thead>
<tr>
<th>Field</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>NextHeader</td>
<td>16</td>
</tr>
<tr>
<td>Reserved</td>
<td>16</td>
</tr>
<tr>
<td>Offset</td>
<td>16</td>
</tr>
<tr>
<td>RES</td>
<td>16</td>
</tr>
<tr>
<td>M</td>
<td>16</td>
</tr>
</tbody>
</table>

### Points:

- IPv4 – complex header
- IPv6: simple header; how to implement fragmentation, authentication, etc...?
- Extension headers: headers are chained; the next header field holds the protocol id
Currently Defined Headers

- Hop-by-hop options
  - Options examined by every hop
  - Example: jumbogram (32-bit payload)

- Routing header
  - Similar to source routing

- Fragmentation header

- Authentication header

- Encapsulations security

- Destination options header
IPv6 Routing

- IPv4 backbone has big routing table size; headache for backbone operators
- IPv6 addressing specification restricts the number of routing table entries by using architecture-enforced “routing aggregation”
- Hierarchical routing
  - Geographically
  - Provider based (change of provider = change of address)
- How does NAT compare in terms of addressing? in terms of changing provider?
IPv6 Routing (cont’d)

- Aggregatable Global address
  - Top level – 13-bits (3 bits to indicate aggregatable unicast addresses)
  - Next level – 48-bits; Site level – 16-bits

- CIDR from the beginning, so the divisions are not too important

- Only 8192 entries in the top most level
Other Routing Features

• Supports source routing (hmmm)

• Multicast
  – Scope identifier to limit how far a multicast propagates

• Anycast
  – Like Multicast, several nodes have the same address
  – Like Unicast, messages sent go to only one of them (which one?)
  – Why?
Autoconfiguration

• Network configuration is difficult
  – At a minimum need to set IP number, subnet mask and nameserver
  – DHCP helps but requires the presence of a DHCP server
  – Would like true plug-and-play/“stateless” configuration

• Obtaining a locally unique address
  – Put your physical (ethernet) address in low order 48-bits
  – Attach yourself to the link-local address space (beginning with 1111111010)

• How to obtain a globally unique address?
  – Rely on subnet router to broadcast its prefix address periodically, with at least 48-bits that can be filled in by the unique address identifier
Transition Plan

- One of the basic requirements is to make the transition from IPv4 to IPv6 easy – why?
  - RFC 1933

- In the early stages (today)
  - Internet is IPv4
  - Most nodes are IPv4
  - Some IPv6 nodes

- IPv6 nodes use their equivalent IPv4 compatible address

- IPv6 enabled nodes run both IPv4 and IPv6 stacks

- Use tunneling when crossing IPv4 network

- Similar to Mbone, 6bone spans 50 countries
Transition Plan

- Late stages – IPv6 most everywhere
- Some IPv4 nodes
- Not enough IPv4 addresses for all nodes
  - Must rely on translation
- IPv4 relegated to an option within the IPv6 stack
- Not there yet; it remains to be seen what will happen as we approach address depletion
## IPv4 vs. IPv6 Functions

<table>
<thead>
<tr>
<th>IP Service</th>
<th>IPv4</th>
<th>IPv6</th>
</tr>
</thead>
<tbody>
<tr>
<td>QoS</td>
<td>IP Precedence driven</td>
<td>Individual Flow Identification; classes of traffic</td>
</tr>
<tr>
<td>Security</td>
<td>IPSec available</td>
<td>IPsec mandatory</td>
</tr>
<tr>
<td>Auto config</td>
<td>DHCP for hosts</td>
<td>Serverless or DHCP</td>
</tr>
<tr>
<td></td>
<td>future site renumbering</td>
<td>site level numbering</td>
</tr>
<tr>
<td>Scalability</td>
<td>Hierarchical routing</td>
<td>Hierarchical routing</td>
</tr>
<tr>
<td>Mobility</td>
<td>Mobile IP</td>
<td>Mobile IP</td>
</tr>
<tr>
<td>Multicast</td>
<td>Multicast BGP</td>
<td>Scope identifier, multicast</td>
</tr>
<tr>
<td>Addressing</td>
<td>32-bits</td>
<td>128-bits</td>
</tr>
</tbody>
</table>
IPv6

- Department of Defense
  - All machines are IPv6 enabled as of October 2004
  - All must move to IPv6 by 2008
  - Check this article: http://dc.internet.com/news/article.php/3287191

- Is that enough to tip the scales in favor of IPv6?
Commercial World

• This is the big question

• There has to be something in it for them
  – Do they need the IP addresses? Yes but not badly, thanks to NAT
  – Does IPv6 really deliver on its promise? Can they get features they can charge customers for? This is not clear
  – Are there alternatives? Yes! IPNL/other solutions of this flavor
Network Address Translation – Recall

- Use any addresses internally on your private network

- NAT enabled gateway replaces internal address with its own address on outgoing connections
  - Uses port numbers to keep track of which internal host to route incoming packets to
NAT is Evil?

- Unidirectional communication
- inefficient;
- breaks the structure of the internet/end to end argument
  - protocols that rely on IP address will break
- single point of failure;
- Possible address conflicts;
- Hard to account usage
- But maybe not: will revisit in a little bit
Can we retro-fit instead of re-engineer?

Given a limited or faulty design, is it better to tear it down and restart or to try and fix it?
  – Should you fix a clunker or buy a new car?

NAT already used to delay the most critical problem with IPv4 – address depletion
  – We discussed why NAT is considered a hack, but are there legitimate benefits to it?

Only a quick discussion of this paper; need to move on to multicast
Why NAT may not Evil

• Isolates site address from global address space
  – More secure
  – External changes don’t affect site and vice versa
    * Changing provider becomes easy (only NAT box changes)
    * Internal IP reassignment easier
  – What if two NAT’d networks merged?
• Is global addressability really needed?
  – Dominant applications are client/server based
    – only server needs to be addressable
  – Is this true with new applications such as peer to peer?
• Delays (eliminates?) address depletion, without having to re-engineer the net
  – Is the cost of maintaining NAT really less than switching an network over to IPv6?
Motivation: Can a “suitable” Internet architecture be developed using an extension to NAT?

“suitable”

- All hosts have long-lived, globally routable addresses (if they so choose) that serve to also identify the host.
- Routers are stateless – no per connection state
- A network’s address prefix is assigned independently of where the network attaches to the Internet
- Packets cannot be easily hijacked by rogue or misconfigured hosts

what is an extension vs. a brand new protocol

- Isolate changes to hosts and NAT boxes, no changes to routers
General Idea

- Create another layer on top of IP (IPNL) that is routed using “NAT boxes” (IPNL routers)
  - Uses FQDN as a unique end to end identifier
  - Extends the available IP address space by defining an private IP address space
  - Original IP address becomes “high order” part of the full address
- A Realm is behind one (or more?) frontdoors
  - Internal IP routers route based on the private address only
- To an IPNL router, the internal IP “realm” becomes like a switched non broadcast link layer, with IP being the address family
Topology

- IP Host
- Global IP and DNS
- IPNL Host
- Frontdoor
- Global IP and DNS
- Internal nl-router
- Extend Edges of Infrastructure

- Global IP
- Realm
- Local IP
- Extend IP Address

- TCP/UDP
  - IP
  - Link
- TCP/UDP
  - IPNL
  - IP
  - Link

- Add New Protocol Layer
More specifically

• Multiple Realms (zones) behind a front door (16-bit realm identifier)
  – Internal NL routers dynamically maintain information to reach realms behind the same front door
  – DNS zone associated with every realm
• Routing possible by FQDN or by IPNL address
Routing by FQDN

- Realm associated with one or more DNS zones (exclusive)
- DNS must be aware of visiting hosts too
- An internal router keeps explicit paths to zones behind the same nl-router
Routing

- If source and destination don't share the same front door
  - Route by default to the front door
  - Front door routes through middle realm to destination front door (uses global DNS)

- Routing also possible by IPNL address (need address resolution, a little complicated)
Discussion

• Less drastic solutions than IPv6 possible to overcome the address depletion problems

• Does it address the shortcomings of NAT?

• Does it address the shortcomings of IPv4?
  – What about the other functionality that is allowed by IPv6?

• IPNL is an academic solution, but its representative of similar solutions that are getting traction (more in a little bit)

• Which solution will win?
  – Nobody really knows...
Discussion

• IPNL is an academic solution, but generally idea is being pushed using even less drastic changes
  – Provide peer-to-peer addressability on top of NAT

• One suggestion that is getting more traction than IPNL is...
  – NUTSS (NAT, URI, Tunnel, SIP, STUN)
  – Basically, uses SIP URI names as global names, uses STUN to associate name with current NAT box/port

• Another suggestion is Teredo (IPv6 over UDP over IPv4)
Getting through NAT

App: I'm steve
UDP: 1234
IP: 1.1.1.1

Application Server
App: I'm bob
UDP: 5678
IP: 2.1.1.1

1.1.1.1

steve

10.1.1.1

10.1.1.1

bob

2.1.1.1

10.1.1.2
Getting through NAT (2)
NUTSS

SIP INVITE message
To: bob@sip.bar.com
From: steve@sip.foo.com
SDP: 1.1.1.1:1234
Multicast

- Multicast is sending messages to a set of receivers potentially scattered all over the internet
  - Can you think of applications?

- Multicast approach: just unicast to the destinations, will this work?

- Multicast approach: broadcast to everyone, drop the packet if you don't need it

- What do we want in a multicast implementation?
Multicast – Issues

- Use Internet efficiently
  - Send messages only to intended receivers
  - Use best paths
  - Consolidate messages on common links
  - Scalable solutions

- General approach: recipients are anonymous
  - Identified by group addresses
  - Membership is dynamic

- What cooperation is needed from intermediate routers?

- Service Model
  - Reliable? QoS?