New Routing Ideas

- Border Gateway Protocol (BGP) cont.
- Overlay networks
- Active networks
- Assigned reading
  - [S+99] The End-to-End Effects of Internet Path Selection
  - [W99] Active network vision and reality: lessons from a capsule-based system

Outline

- Multi-Homing
- BGP Misconfiguration
- Overlay Routing
- Active Networks

Multi-homing

- With multi-homing, a single network has more than one connection to the Internet.
- Improves reliability and performance:
  - Can accommodate link failure
  - Bandwidth is sum of links to Internet
- Challenges
  - Getting policy right (MED, etc.)
  - Addressing
Multi-homing to Multiple Providers

- Major issues:
  - Addressing
  - Aggregation
- Customer address space:
  - Delegated by ISP1
  - Delegated by ISP2
  - Delegated by ISP1 and ISP2
  - Obtained independently

Address Space from one ISP

- Customer uses address space from ISP1
- ISP1 advertises /16 aggregate
- Customer advertises /24 route to ISP2
- ISP2 relays route to ISP1 and ISP3
- ISP2-3 use /24 route
- ISP1 routes directly
- Problems with traffic load?

Address Space from Both ISPs

- ISP1 and ISP2 continue to announce aggregates
- Load sharing depends on traffic to two prefixes
- Lack of reliability: if ISP1 link goes down, part of customer becomes inaccessible.
- Customer may announce prefixes to both ISPs, but still problems with longest match as in case 1.

Pitfalls

- ISP1 aggregates to a /19 at border router to reduce internal tables.
- ISP1 still announces /16.
- ISP1 hears /24 from ISP2.
- ISP1 routes packets for customer to ISP2!
- Workaround: ISP1 must inject /24 into I-BGP.
### Address Space Obtained Independently
- Offers the most control, but at the cost of aggregation.
- Still need to control paths
- Some ISP's ignore advertisements with long prefixes

### Outline
- Multi-Homing
- BGP Misconfiguration
- Overlay Routing
- Active Networks

### Origin Misconfiguration
- Can only tell additions not omissions

<table>
<thead>
<tr>
<th>Old Route</th>
<th>New Route</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Self deaggregation</strong></td>
<td></td>
</tr>
<tr>
<td>a.b.0/16 X Y Z</td>
<td>a.b.c.0/24 X Y Z</td>
</tr>
<tr>
<td><strong>Related origin</strong></td>
<td></td>
</tr>
<tr>
<td>a.b.0/16 X Y Z</td>
<td>a.b.0/16 X Y</td>
</tr>
<tr>
<td></td>
<td>a.b.0/16 X Y Z O</td>
</tr>
<tr>
<td></td>
<td>a.b.c.0/24 X Y</td>
</tr>
<tr>
<td></td>
<td>a.b.c.0/24 X Y Z O</td>
</tr>
<tr>
<td><strong>Foreign origin</strong></td>
<td></td>
</tr>
<tr>
<td>a.b.0/16 X Y Z</td>
<td>a.b.0/16 X Y O</td>
</tr>
<tr>
<td></td>
<td>a.b.c.0/24 X Y O</td>
</tr>
<tr>
<td></td>
<td>e.f.g.h.i X Y O</td>
</tr>
</tbody>
</table>

### Export Misconfiguration
- Need desired policy inferred from announcements

<table>
<thead>
<tr>
<th>Export</th>
<th>Policy Violation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provider → AS → Provider</td>
<td>Route exported to provider was imported from a provider</td>
</tr>
<tr>
<td>Provider → AS → Peer</td>
<td>Route exported to peer was imported from a provider</td>
</tr>
<tr>
<td>Peer → AS → Provider</td>
<td>Route exported to provider was imported from a peer</td>
</tr>
<tr>
<td>Peer → AS → Peer</td>
<td>Route exported to peer was imported from a peer</td>
</tr>
</tbody>
</table>
Interesting Observations - Origin Misconfig

- Results
  - 72% of new routes may be misconfigured
  - 11-13% of incidents (4% of prefixes) affect connectivity
    - But only identify addition of origins
    - Should really evaluate % of connectivity problems come from misconfig
    - Mostly from self-deaggregation
    - Problems from foreign origin

- Causes
  - Router bugs (initialization)
  - Reliance on upstream filtering
  - Old config files
  - Links to IGP
  - Hijacks

Interesting Observations - Export Misconfig

- Primarily causes extra traffic
- Causes
  - Export policies based on prefix based configuration
    - Export client’s prefix from one provider to another

Outline

- Multi-Homing
- BGP Misconfiguration
- Overlay Routing
- Active Networks

Overlay Routing

- Basic idea:
  - Treat multiple hops through IP network as one hop in “virtual” overlay network
  - Run routing protocol on overlay nodes
- Why?
  - For performance – can run more clever protocol on overlay
  - For functionality – can provide new features such as multicast, active processing, IPv6
**Overlay for Features**

- How do we add new features to the network?
  - Does every router need to support new feature?
  - Choices
    - Reprogram all routers → active networks
    - Support new feature within an overlay
  - Basic technique: tunnel packets
- Tunnels
  - IP-in-IP encapsulation
  - Poor interaction with firewalls, multi-path routers, etc.

**Examples**

- IP V6 & IP Multicast
  - Tunnels between routers supporting feature
- Mobile IP
  - Home agent tunnels packets to mobile host’s location
- QOS
  - Needs some support from intermediate routers → maybe not?

**Overlay for Performance [S+99]**

- Why would IP routing not give good performance?
  - Policy routing – limits selection/advertisement of routes
  - Early exit/hot-potato routing – local not global incentives
  - Lack of performance based metrics – AS hop count is the wide area metric
- How bad is it really?
  - Look at performance gain an overlay provides

**Quantifying Performance Loss**

- Measure round trip time (RTT) and loss rate between pairs of hosts
  - ICMP rate limiting
  - Alternate path characteristics
    - 30-55% of hosts had lower latency
    - 10% of alternate routes have 50% lower latency
    - 75-85% have lower loss rates
Bandwidth Estimation

- RTT & loss for multi-hop path
  - RTT by addition
  - Loss either worst or combine of hops – why?
    - Large number of flows ➔ combination of probabilities
    - Small number of flows ➔ worst hop
- Bandwidth calculation
  - TCP bandwidth is based primarily on loss and RTT
- 70-80% paths have better bandwidth
- 10-20% of paths have 3x improvement

Possible Sources of Alternate Paths

- A few really good or bad AS’s
  - No, benefit of top ten hosts not great
- Better congestion or better propagation delay?
  - How to measure?
    - Propagation = 10th percentile of delays
      - Both contribute to improvement of performance
  - What about policies/economics?

Overlay Challenges

- “Routers” no longer have complete knowledge about link they are responsible for
- How do you build efficient overlay?
  - Probably don’t want all $N^2$ links – which links to create?
  - Without direct knowledge of underlying topology how to know what’s nearby and what is efficient?

Future of Overlay

- Application specific overlays
  - Why should overlay nodes only do routing?
- Caching
  - Intercept requests and create responses
- Transcoding
  - Changing content of packets to match available bandwidth
- Peer-to-peer applications
Outline

• Multi-Homing
• BGP Misconfiguration
• Overlay Routing
• Active Networks

Why Active Networks?

• Traditional networks route packets looking only at destination
  • Also, maybe source fields (e.g. multicast)
• Problem
  • Rate of deployment of new protocols and applications is too slow
• Solution
  • Allow computation in routers to support new protocol deployment

Active Networks

• Nodes (routers) receive packets:
  • Perform computation based on their internal state and control information carried in packet
  • Forward zero or more packets to end points depending on result of the computation
• Users and apps can control behavior of the routers
• End result: network services richer than those by the simple IP service model

Why not IP?

• Applications that do more than IP forwarding
  • Firewalls
  • Web proxies and caches
  • Transcoding services
  • Nomadic routers (mobile IP)
  • Transport gateways (snoop)
  • Reliable multicast (lightweight multicast, PGM)
  • Online auctions
  • Sensor data mixing and fusion
• Active networks makes such applications easy to develop and deploy
Variations on Active Networks

- Programmable routers
  - More flexible than current configuration mechanism
  - For use by administrators or privileged users
- Active control
  - Forwarding code remains the same
  - Useful for management/signaling/measurement of traffic
- “Active networks”
  - Computation occurring at the network (IP) layer of the protocol stack → capsule based approach
  - Programming can be done by any user
  - Source of most active debate

Case Study: MIT ANTS System

- Conventional Networks:
  - All routers perform same computation
- Active Networks:
  - Routers have same runtime system
  - Tradeoffs between functionality, performance and security

System Components

- Capsules
- Active Nodes:
  - Execute capsules of protocol and maintain protocol state
  - Provide capsule execution API and safety using OS/language techniques
- Code Distribution Mechanism
  - Ensure capsule processing routines automatically/dynamically transfer to node as needed

Capsules

- Each user/flow programs router to handle its own packets
  - Code sent along with packets
  - Code sent by reference
- Protocol:
  - Capsules that share the same processing code
  - May share state in the network
  - Capsule ID (i.e. name) is MD5 of code
Capsules are forwarded past normal IP routers.

- When node receives capsule uses “type” to determine code to run
- What if no such code at node?
  - Requests code from “previous address” node
  - Likely to have code since it was recently used

Code is transferred from previous node
- Size limited to 16KB
- Code is signed by trusted authority (e.g. IETF) to guarantee reasonable global resource use

Research Questions
- Execution environments
  - What can capsule code access/do?
- Safety, security & resource sharing
  - How isolate capsules from other flows, resources?
- Performance
  - Will active code slow the network?
- Applications
  - What type of applications/protocols does this enable?
Functions Provided to Capsule

- Environment Access
  - Querying node address, time, routing tables
- Capsule Manipulation
  - Access header and payload
- Control Operations
  - Create, forward and suppress capsules
  - How to control creation of new capsules?
- Storage
  - Soft-state cache of app-defined objects

Safety, Resource Mgt, Support

- Safety:
  - Provided by mobile code technology (e.g. Java)
- Resource Management:
  - Node OS monitors capsule resource consumption
- Support:
  - If node doesn’t have capsule code, retrieve from somewhere on path

Applications/Protocols

- Limitations
  - Expressible → limited by execution environment
  - Compact → less than 16KB
  - Fast → aborted if slower than forwarding rate
  - Incremental → not all nodes will be active
- Proof by example
  - Host mobility, multicast, path MTU, Web cache routing, etc.

Discussion

- Active nodes present lots of applications with a desirable architecture
- Key questions
  - Is all this necessary at the forwarding level of the network?
  - Is ease of deploying new apps/services and protocols a reality?
Next Lecture: TCP Reliability

- TCP reliability
- Assigned reading
  - [FF96] Simulation-based Comparisons of Tahoe, Reno, and SACK TCP