Lecture 5 Overview

- Brief intro to overlay networking
- Active networking
- Wetherall ‘98 discussion
Internet “Ossification”

- Internet protocols evolve over long timescales
  - Witness the paper we just read from 1974
  - IPv6 was proposed almost 20 years ago now! (RFC2460)

- Yet new ideas appear all the time
  - As of July 2015, there are 7,567 RFCs
  - SIGCOMM/NSDI/etc. are full of new protocols/architectures

- Key challenge: how to deploy new things STAT?
  - In particular changes to the networking layer
  - Would naively require changing every router on the Internet!
Obvious solution: Overlays

- Build the service at a layer above IP
  - Create an “overlay” of nodes connected by IP
  - Functionality implemented at IP “end hosts” and forwarded on
  - IP provides “tunnels” between nodes of overlay
- Exactly the same approach originally used by IP
  - IP treated the phone network as a series of tunnels
- Used to introduce a variety of services in the ‘90s
  - M-Bone, X-Bone, A-Bone, etc., etc.
- Major downside is performance
  - Each packet needs to be handled by an end host
  - Likely traverses an inefficient route
An Alternative Approach

- Make the routers extensible: Active Networking
  - Provide a mechanism to implement services at an Internet router
  - Removes the need to route indirectly
  - But still requires additional end-host-like processing

- Same idea, two different layers
  - Active nodes are implemented at the (extended) network layer
  - Overlay nodes operate at the application layer

- Performance/deployment tradeoffs
  - Anybody can deploy an overlay network
  - But Active networks could be much more efficient
Active Nodes

- Execute protocols in restricted environment
  - Limits access to sensitive/shared resources
- Primitives for application-defined protocol processing
- Enforce limits on resource consumption
  - Active Nodes responsible for network integrity and errors
  - TTL fields decreased as resources are consumed
  - Capsules with 0 TTL’s discarded
- Code propagation
  - Capsules identify protocol
  - Protocol uniquely define code path, Active Node retrieves it
  - MD-5 signature for safety
Active Networking Benefits

- Active Reliable Multicast
  - Problem: NACK implosion
  - Duplicate NACK suppression, cache multicast data, upstream NACK generation

- Online Auctions
  - Process failed bids closer to users, reducing server load

- Mixing Sensor Data
  - Combine multiple input signals to reduce transmitted messages and end host computing burden
Caching Fast Changing Data

- Service that provides rapidly changing information
  - Military information system, airline flight status, stock quotes

- Web Caching?
  - Yesterday’s proxy caches could not cache dynamically generated data – Akamai to the rescue
  - Wrong granularity: pages as opposed to objects

- Active Networks can be customized to provide:
  - Application-specific cache coherence
  - Application-specific object granularity
AN Caching Protocol

- Quotes cached at Active Nodes on client-to-server path
- Subsequent requests intercepted to consult local cache
- Caches automatically lie on the path between client/server
  - Do not redirect requests to caches in wrong direction
- Application-specific cache coherence
  - Different clients have different requirements for “freshness”
- (Potential) Benefits:
  - Decrease client latency
  - Decrease the traffic at routers
  - Decrease server load
ANTS

- Java toolkit for writing and executing active protocol code
- Goals
  - Simultaneous use of multiple network protocols
  - Deploy multiple protocols with no central control (orig)
  - Dynamic deployment of new protocols (orig)
- Migration path from non-active to active world
  - Benefits from a small number of active nodes
  - Rather than make hop by hop routing decisions, make active node to active node routing decisions
  - Minimum number of nodes necessary for success?
  - Small number of nodes successful/meltdown under scale?
Code Propagation

1. capsule
2. request
3. response
4. code group

previous node
loadng node
Performance Discussion

- 1999 PC-based routers able to forward 200,000 pkt/sec
  - Reaching 1 Gbps at typical packet sizes
- Modern commodity routers forward ~60M pkt/sec/port @ 40G

- Active Networks (1999 numbers) limited by Java, user-level implementation, safety, general-purpose routines
  - 1700 capsules/sec, 16 Mbps on Sun Ultra 1
  - Latencies range from 500-700 us
  - Related effort (PAN) forwards 100 Mbps w/kernel support
Active Node Throughput

C Rela y
JAVA Relay
ANTS

Throughput (capsules/sec)

Capable Payload Size (bytes)
Active Node Latency

![Graph showing latency vs capsule payload size for different systems: C Relay, JAVA Relay, ANTS.](image)
## Overheads

<table>
<thead>
<tr>
<th>Operation</th>
<th>IP?</th>
<th>Time (μs)</th>
<th>(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Packet Receive</td>
<td>no</td>
<td>180</td>
<td>29</td>
</tr>
<tr>
<td>2. Header Processing</td>
<td>yes</td>
<td>30</td>
<td>5</td>
</tr>
<tr>
<td>3. Type Demultiplex</td>
<td>no</td>
<td>20</td>
<td>3</td>
</tr>
<tr>
<td>4. Capsule Decode</td>
<td>no</td>
<td>110</td>
<td>18</td>
</tr>
<tr>
<td>5. Capsule Evaluate</td>
<td>no</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>6. Route Lookup</td>
<td>yes</td>
<td>30</td>
<td>5</td>
</tr>
<tr>
<td>7. Capsule Encode</td>
<td>no</td>
<td>90</td>
<td>14</td>
</tr>
<tr>
<td>8. Header Processing</td>
<td>yes</td>
<td>40</td>
<td>7</td>
</tr>
<tr>
<td>9. Packet Transmit</td>
<td>no</td>
<td>80</td>
<td>13</td>
</tr>
<tr>
<td>Other</td>
<td>n/a</td>
<td>25</td>
<td>4</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>n/a</td>
<td><strong>615</strong></td>
<td><strong>100</strong></td>
</tr>
</tbody>
</table>
Rethinking Performance

- Traditional networking metrics:
  - Bandwidth, latency on a packet level
- What really matters is end-to-end performance
  - Application throughput
  - Client-perceived latency
- Active Networks may slow routing down
  - But improve end-to-end application performance
  - Use application-specific notions of throughput/latency
Resource Allocation Issues

- Difficulties with allocating resources in active nets:
  - Single capsule consumes too much resources at active node
  - Capsule and other capsules it creates consume unbounded resources across wide area
  - End application introduces large number of capsules

- How to address these problems?
Resource Allocation Issues

- Difficulties with allocating resources in active nets:
  - Single capsule consumes too much resources at active node
    Java technology allows per-capsule resource consumption limits
  - Capsule and other capsules it creates consume unbounded resources across wide area
    Difficult problem
    Hard to know \textit{a priori} how much resources a capsule needs
  - End application introduces large number of capsules
    Not well-addressed in either Internet or Active Networks
    The model is that all users cooperate to provide fair access
Resource Containers

- OS abstraction (Banga, et al OSDI 99)
  - The unit of protection (process) should not equal unit of allocation
  - Ex. user A w/9 processes, user B w/1 process
- Allow processes to be bound to resource containers
  - E.g., clients pass resource containers to servers
    Server computation performed in context of individual client containers
Distributed Resource Containers?

- Allocation mechanism for arbitrary wide-area computation
  - Local decisions are affected by remote characteristics
  - E.g., ensure that protocol A does not consume more than 0.1% of aggregate global resources at any time
  - E.g., ensure that user A does not consume more than fixed amount of resource in aggregate across wide area
  - Must use approximate information
  - Problem of multicast to varying audience size?
Who Can Introduce Services?

- Originally, goal was to allow anyone to introduce and test a new service
  - However, issues with wide-area resource allocation makes it important to verify the “correctness” of capsule code
  - Current model requires approval from central authority (such as IETF)
  - Makes deploying protocols slower than original vision, but still much faster than current Internet
Protection Issues

- Need to protect against
  - Corruption of runtime environment by service code
  - Corrupted/spoofed capsule code
  - Soft state cached at Active Nodes for one protocol manipulated by another service

- How does Active Networks provide protection for above?
Protection Issues

- Need to protect against
  - Node runtime corruption by service code
    - Java
  - Corrupted/spoofed capsule code
    - MD-5 signature
  - Soft state cached at Active Nodes for one protocol manipulated by another service
    - Restricted ANTS API
    - Guarded access to state among separate services
    - Hierarchical service model allows multiple service types to cooperate
Security and Resource Allocation

- Multicast program that spawns two packets at each node

*All Possible Network Programs*

- Node-Safe Programs
- Network-Safe Programs
How should we prevent this?
- TTLs are a weak solution; not related to topology
- Fairness mechanisms mitigate, but not enough
- ANTS falls back on certification of programs …
Active Networks Discussion

- Introduce programmability for
  - Rapid introduction of new protocols
  - Increased end-to-end performance

- Rethink network performance in terms of app performance

- Issues:
  - Speed, Resource allocation, Safety/Security

- Active Networks can make explicit “transparent” network caching, network address translation, etc.
For Next Class…

● Read and review ONIX

● Submit project groups by EOD

● First take-home quiz over the papers going out shortly