Today’s Internet is built around a unicast point-to-point communication abstraction:
- Send packet “p” from host “A” to host “B”
- This abstraction allows Internet to be highly scalable and efficient, but…
- … not appropriate for applications that require other communications primitives:
  - Multicast
  - Anycast
  - Mobility
  - …

Point-to-point communication implicitly assumes there is one sender and one receiver, and that they are placed at fixed and well-known locations
- E.g., a host identified by the IP address 128.32.xxx.xxx is located in Berkeley

Motivations
- Extend IP to support new communication primitives, e.g.,
  - Mobile IP
  - IP multicast
  - IP anycast
- Disadvantages:
  - Difficult to implement while maintaining Internet’s scalability (e.g., multicast)
  - Require community wide consensus – hard to achieve in practice

Why?

IP Solutions

Key Observation
- Virtually all previous proposals use indirection, e.g.,
  - Physical indirection point → mobile IP
  - Logical indirection point → IP multicast

"Any problem in computer science can be solved by adding a layer of indirection"
**i3’s Solution**

Build an efficient indirection layer on top of IP

- Use an overlay network to implement this layer
- Incrementally deployable; don’t need to change IP

**Internet Indirection Infrastructure (i3)**

- Each packet is associated an identifier `id`
- To receive a packet with identifier `id`, receiver R maintains a trigger `(id, R)` into the overlay network

**Service Model**

- **API**
  - sendPacket(p);
  - insertTrigger(t);
  - removeTrigger(t)  // optional
- Best-effort service model (like IP)
- Triggers periodically refreshed by end-hosts
- ID length: 256 bits

**Mobility**

- Host just needs to update its trigger as it moves from one subnet to another

**Multicast**

- Receivers insert triggers with same identifier
- Can dynamically switch between multicast and unicast

**Anycast**

- Use longest prefix matching instead of exact matching
  - Prefix p: anycast group identifier
  - Suffix s: encode application semantics, e.g., location
Service Composition: Sender Initiated

- Use a stack of IDs to encode sequence of operations to be performed on data path
- Advantages
  - Don’t need to configure path
  - Load balancing and robustness easy to achieve

Outline

- Implementation
- Examples
- Security
- Applications

Quick Implementation Overview

- i3 is implemented on top of Chord
  - But can easily use CAN, Pastry, Tapestry, etc
- Each trigger \( t = (id, R) \) is stored on the node responsible for \( id \)
- Use Chord recursive routing to find best matching trigger for packet \( p = (id, data) \)

Routing Example

- R inserts trigger \( t = (37, R) \); S sends packet \( p = (37, data) \)
- An end host needs to know only one i3 node to use i3
  - E.g., S knows node 3, R knows node 35

Optimization #1: Path Length

- Sender/receiver caches i3 node mapping a specific ID
- Subsequent packets are sent via one i3 node

\( \text{Chord circle} \)

\( \text{Sender} \)

\( \text{Receiver (R)} \)

\( \text{Firewall (F)} \)

\( \text{Transcoder (T)} \)

\( \text{Sender} \)

\( \text{Receiver (R)} \)

\( \text{Sender/receiver caches i3 node mapping a specific ID} \)

\( \text{Subsequent packets are sent via one i3 node} \)
Optimization #2: Triangular Routing

- Use well-known trigger for initial rendezvous
- Exchange a pair of (private) triggers well-located
- Use private triggers to send data traffic

Example 1: Heterogeneous Multicast

- Sender not aware of transformations

Example 2: Scalable Multicast

- i3 doesn’t provide direct support for scalable multicast
  - Triggers with same identifier are mapped onto the same i3 node
- Solution: have end-hosts build an hierarchy of triggers of bounded degree

Example 3: Load Balancing

- Servers insert triggers with IDs that have random suffixes
- Clients send packets with IDs that have random suffixes

Outline

- Implementation
  - Examples
    - Heterogeneous multicast
    - Scalable Multicast
    - Load balancing
    - Proximity
- Security
- Applications

Example 2: Scalable Multicast

Unlike IP multicast, i3
1. Implement only small scale replication → allow infrastructure to remain simple, robust, and scalable
2. Gives end-hosts control on routing → enable end-hosts to
   - Achieve scalability, and
   - Optimize tree construction to match their needs, e.g., delay, bandwidth
Example 4: Proximity

- Suffixes of trigger and packet IDs encode the server and client locations

Outline

- Implementation
- Examples
  - Security
- Applications

Some Attacks

Eavesdropping

Loop

Confluence

Dead-End

Defences

- Encryption:
  - A send to B
  - Encrypt private trigger id_x using public key of B (vice versa)
- Challenges
  - Nonce-based challenges to prevent unjustified insertion of triggers by third parties
  - Send random nonce to end-host. Remove if end-host does not respond.
- Push-back
  - When there is no more matching trigger for packet ID, i3 sends a push-back message to previous node
- Time-to-live (TTL)

More attacks...

- Malicious linking
  - Attacker can sign an end-host R to a high bandwidth traffic stream sent to id by inserting a trigger (id,R)
- Impersonation
  - Same as eavesdropping

Constrained Triggers

- $h_1(), h_0()$: well-known one-way hash functions
- Use $h_1(), h_0()$ to constrain trigger $(x, y)$

Encryption:

Use $h_1(), h_0()$ to constrain trigger $(x, y)$
Routing as a Service

Goal: develop network architectures that
- Allow end-hosts to pick their own routes
- Allow third-parties to easily add new routing protocols

Ideal model:
- Oracles that have complete knowledge about network
- Hosts query paths from oracles
- Path query can replace today’s DNS query
- Hosts forward packets along these paths

Routing as a Service (cont’d)

Routing as a Service

1) Give hosts control on routing
   - A trigger is like an entry in a routing table!
   - Flexibility, customization
   - End-hosts can
     - Source route
     - Set-up acyclic communication graphs
     - Route packets through desired service points
     - Stop flows in infrastructure
   - ... 

2) Implement data forwarding in infrastructure
   - Efficiency, scalability

Design Principles

<table>
<thead>
<tr>
<th>Internet &amp; Infrastructure overlays</th>
<th>Host</th>
<th>Infrastructure</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Data plane</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Control plane</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>p2p &amp; End-host overlays</th>
<th></th>
<th>Data plane</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Control plane</td>
</tr>
<tr>
<td>i3</td>
<td></td>
<td>Data plane</td>
</tr>
</tbody>
</table>
Outline
- Implementation
- Examples
- Security
- Architecture Optimizations
  > Applications
    - Routing as a service
    - Service composition platform
    - Support of legacy applications over overlays

Service Composition Platform
- Goal: allow third-parties and end-hosts to easily insert new functionality on data path
  - E.g., firewalls, NATs, caching, transcoding, spam filtering, intrusion detection, etc..
- Why i3?
  - Make middle-boxes part of the architecture
  - Allow end-hosts/third-parties to explicitly route through middle-boxes

Example
- Use Bro system to provide intrusion detection for end-hosts that desire so
- Spam filtering, etc.

Conclusions
- Indirection – key technique to implement basic communication abstractions
  - Multicast, Anycast, Mobility, ...
  - http://i3.cs.berkeley.edu
- Reminder:
  - Project proposal was due yesterday!
  - Volunteers for second presentations

See http://ocala.cs.berkeley.edu