Internet Indirection Infrastructure (i3)

Ion Stoica, Daniel Adkins, Shelley Zhuang, Scott Shenker, Sonesh Surana
UC Berkeley
SIGCOMM 2002

Motivations

• Today’s Internet is built around a unicast point-to-point communication abstraction:
  – Send packet “p” from host “A” to host “B”

• Point-to-point communication
  – Implicitly assumes there is one sender and one receiver, and that they are placed at fixed and well-known locations
  – Example: a host identified by the IP address 128.111.xxx.xxx is located in UCSB
Motivations

• This abstraction allows Internet to be highly scalable and efficient, but…
• … not appropriate for applications that require other communications primitives:
  – Multicast
  – Anycast
  – Mobility
  – …
• Key Observation: Virtually all previous proposals use indirection
  – Physical indirection point → mobile IP
  – Logical indirection point → IP multicast

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
Solution

Multicast | Anycast | Mobility | Service Composition

An indirection layer based on overlay network (decouples sending and receiving)

DHT

IP Layer

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 $s_i$: encode application semantics, e.g., location
Using i3

- Service Composition
  - Server initiated
  - Receiver initiated
- Large Scale Multicast

Service Composition: Sender Initiated

- Use a stack of IDs to encode sequence of operations to be performed on data path
Service Composition: Receiver Initiated

- Receiver can also specify the operations to be performed on data

Large Scale Multicast

- Can create a multicast tree for scalability
Implementation Overview

- ID space is partitioned across infrastructure nodes
  - Each node responsible for a region of ID space
- Each trigger \((id, R)\) is stored at the node responsible for \(id\)
- Use Chord to route triggers and packets to nodes responsible for their IDs
  - \(O(\log N)\) hops

Properties

- Robustness, Efficiency, Scalability, Stability
  - Robustness: refresh triggers, trigger replication, back-up triggers
  - Efficiency: Routing optimizations
- Incremental deployment possible
- Legacy applications can be supported by proxy which inserts triggers on behalf of client
Example

- ID space [0..63] partitioned across five i3 nodes
- Each host knows one i3 node
- R inserts trigger (37, R); S sends packet (37, data)
Optimization: Path Length

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

Optimization: Location-aware Triggers

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

Private Triggers:
- S can insert a trigger \([1,S]\) that is stored at server 3
- R can choose a trigger \([30,R]\) that is stored at server 35
Security

• *i3* end-points also store routing information
  – New opportunities for malicious users

• **Goal:** make *i3* not worse than today’s Internet

---

Some Attacks

- **Eavesdropping**
- **Loop**
- **Confluence**
- **Dead-End**
Solutions

- **Eavesdropping**: Use private triggers, periodically change them, multiple private triggers

- **DoS Attacks**: Challenges, Fair Queueing for resource allocation, loop detection

- **Dead-End**: Use push-back

Experimental Results

- Simulation over two topologies
  - Power-law random graph topology
  - Transit-stub topology

- \( Latency\ Stretch = \frac{\text{i3 latency}}{\text{IP latency}} \)

- First Packet Latency, End-to-end Latency
First Packet Latency

Heuristics
- Closest Finger Replica: Store successors of each finger
- Closest Finger Set: Use base $b < 2$ to find fingers, but consider only $\log_2 N$ closest fingers when routing

End-to-end Latency

90th percentile latency stretch vs. no of i3 servers for Transit-stub topology

90th percentile latency stretch vs. no of samples (16384 i3 servers)
i3 latency = (sender to i3 server)+(i3 server to receiver)
Conclusions

- Indirection – key technique to implement basic communication abstractions
  - Multicast, Anycast, Mobility, …
- This research
  - Advocates for building an efficient Indirection Layer on top of IP
  - Explores the implications of changing the communication abstraction

Status

- [http://i3.cs.berkeley.edu/](http://i3.cs.berkeley.edu/)
- *i3* is available as a service on Planetlab
- Support for legacy applications in Linux and Windows XP
- Applications
  - Mobility
  - Transparent access to machines behind NATs
  - Secure and transparent access to services behind firewalls