Internet Indirection Infrastructure (i3)

Ronghui Gu
ronghui.gu@yale.edu
Outline

- Introduction
- i3 Overview
- Using i3
- Implementation and optimization
- Experimental Results
- Conclusion
Introduction

MOTIVATION

- **Unicast** point-to-point communication
  - One sender, One receiver
  - Fixed location, which is well-known
  - Host A sends packet p to host B, identified by IP

- Highly scalable and efficient

- Not appropriate for:
  - Multicast
  - Anycast
  - Mobility
Why not appropriate?

- IP layer: lose scalability, requires consensus
- Application layer: in a disjointed fashion

Indirection

- More general abstraction
- Decouples the sending hosts from the receiving host
- Send packet p to a “rendezvous”
- IP layer will send p to the receiver(s)
Build an efficient *indirection* layer on top of IP

- Use an *overlay* network
- Incrementally deployable
  - IP layer remains the same
  - Application layer is not aware of its existence
Rendezvous-based communication (simple)
- Packet is associated with an identifier id (256B)
- Receiver R maintains the trigger (id, R)
- Triggers with the same id are stored on the same server

Best-effort service model
Host only needs to update the trigger
The group member register triggers with same id

- Packet matches id will be sent to all the members

- No difference between unicast or multicast
**Overview**

- **Extended version 1:** Use longest prefix matching
  - The length of matching prefix is at least \( k \) \((125)\)
  - Id's that have \( k \)-bit prefix are stored on same server

- **Multicast group shared k-bit prefix**
  - Members have different \((m-k)\) bits suffix
  - When multicasting, send packet with id, which has a k-bit prefix match with all the members

\[\begin{array}{c}
p|s_1 & R_1 \\
p|s_2 & R_2 \\
p|s_3 & R_3 \\
\end{array}\]
• Anycast: deliver packet to one receiver in a group
• Send packet to member with longest prefix

Overview

ANYCAST(2)

Sender

Receiver (R1)

Receiver (R2)

Receiver (R3)

data

p|a

p|s1 R1

p|s2 R2

p|s3 R3

trigger

trigger

trigger
i3 Overview

STACK

● Extended version 2
  ● Replace identifier with identifier stack
  ● Identifier stack is a list of identifiers
    ● $id_{stack} = (id_1, id_2, id_3, \ldots, id_n)$
    ● $id_i$: identifier or address
  ● Packet $p = (id_{stack}, data)$, trigger $t = (id, id_{stack})$

● Match the head of $p$'s stack with $t$
  ● Pop $p$'s stack
  ● Prepend $t$'s stack to $p$'s stack

(id1,id2) data $\oplus$ id1 (x,y) $\Rightarrow$ (x,y,id2) data
● Data is required to be processed before reach
● Use p’s $id_{stack}$ to encode the seq of operations
● Use ability of sender

```
send(((ID_B/J, ID), data))
send((S_B/J, ID), data)
send(ID, data')
send(R, data')
```
HETEROGENOUS MULTICAST

- Receiver could process the data before received
- Use t’s $id_{stack}$ to encode the seq of operations
- Use ability of receiver

```
send(ID, data)
send((ID_B/J, R), data)
send(R, data)
```
• One server sends packets to all members
• Not scale to large multicast group
• Create multicast tree for scalability
Implementation and Optimization

IMPLEMENTATION OVERVIEW

- **Properties**
  - Robustness, Scalability, Efficiency, Stability

- **Chord lookup protocol**
  - Route triggers and packets
  - $N$ nodes: $O(\log N)$ hops
Implementation and Optimization

IMPLEMENTATION OVERVIEW

<table>
<thead>
<tr>
<th>i</th>
<th>finger</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>i</th>
<th>finger</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
</tr>
</tbody>
</table>

Sender (S)

Receiver (R)

[3, 5]

[6, 7]
Implementation and Optimization

Public and private triggers

- Public trigger: long lived, contact
- Private trigger: short lived, inform through public one
- Increase efficiency and security

Robustness

- Refresh triggers
- Back-up triggers
- Replicate triggers (successor of node)
Implementation and Optimization

OPTIMIZATION(2)

● Routing efficiency
  ● Cache i3 server’s IP address
  ● Triangle routing problem:
    ● Choose location of private triggers

● Avoiding hot-spots
  ● Copy triggers to the predecessor
Implementation and Optimization

SECURITY(1)

• **New opportunities for malicious users**
  - IP: end-points can only send and receive packets
  - i3 end-points should maintain routing information

• **Goal**
  - Not worse than today’s Internet
Implementation and Optimization

SECURITY(2)

Eavesdropping

Hijacking

Confluence

Loop

Attacker

Victim (V)

Attacker
Implementation and Optimization

SECURITY(3)

- Eavesdropping
  - Use private triggers, periodically change them
  - **Multiple** private triggers

- Trigger hijacking
  - Add a level of *indirection*

- DoS Attacks
  - Send challenges when a trigger is inserted
  - Limited triggers, limited packets
  - Loop detection
Experimental Results

PACKET LANCENCY

- Latency stretch = (i3 latency) / (IP latency)
- First packet latency
  - Slow: need to find the trigger
- Improvement
  - Closest finger replica: store r succs of finger
  - Closest finger set:
    - Use base b<2 to find finger
    - Consider closest $\log_2 N$ when routing
  - $\log_b N = r \cdot \log_2 N$
Experimental Results

FIRST PACKET LATENCY

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

END-TO-END PACKET LATENCY

90th percentile latency stretch vs. no of samples (16384 i3 servers)
Conclusion

WHAT THE PAPER HAS DONE

- Main idea: *indirection*
- More general abstraction in one overlay
  - Multicast
  - Anycast
  - Mobility
- Based on *Chord*
THANKS FOR YOUR TIME

QUESTIONS?