An Internet Indirection Infrastructure

Motivation:

- Today’s Internet is built around point-to-point communication abstraction:
  - Send packet “p” from host “A” to host “B”
  - One sender, one receiver, at fixed and well-known locations

- … not appropriate for applications that require other communications primitives:
  - Multicast (one to many)
  - Mobility (one to anywhere)
  - Anycast (one to any)

- We’ve seen indirection used to provide these services
  - Idea: make indirection a “first-class object”

Internet Indirection Infrastructure (i3)

- Change communication abstraction: instead of point-to-point, exchange packets by name
  - Each packet has an identifier ID
  - To receive packet with identifier ID, receiver R stores trigger (ID, R) in network
  - Triggers stored in network overlay nodes
Service Model

- API
  - `sendPacket(p);`
  - `insertTrigger(t);`
  - `removeTrigger(t); // optional`

- Best-effort service model (like IP)
- Triggers periodically refreshed by end-hosts
  - Q: what is this approach called?
- Reliability, congestion control, flow-control implemented at end hosts, and trigger-storing overlay nodes

Discussion

- Trigger is similar to routing table entry
- Application-level overlay infrastructure
  - Essentially: application layer *publish-subscribe infrastructure*
- Unlike IP, end hosts control triggers, i.e., end hosts responsible for setting and maintaining “routing tables”
- Provide support for
  - Mobility
  - Multicast
  - Anycast
  - Composable services
Mobility

- Receiver updates its trigger as it moves from one subnet to another
  - Mobility transparent to sender
  - Location privacy

```
Sender
send(ID, data)
```

```
Receiver (R1)
```

```
Receiver updates its trigger as it moves from one subnet to another
```

```
ID R1
```

```
Sender
send(ID, data)
```

```
Receiver (R1)
```

```
ID R2
```

```
Receiver (R2)
```

```
(new) trigger
```

```
Sender
send(ID, data)
```

```
Receiver (R2)
```

```
ID R2
```

```
trigger
```
**Multicast**

- Unifies multicast and unicast abstractions
  - Multicast: receivers insert triggers with same ID
- Application naturally moves between multicast and unicast, as needed
  - “Impossible” in current IP model!

![Multicast Diagram]

**Anycast**

- Route to any one in set of receivers
- Receiver $i$ in anycast group inserts same ID, with anycast qualifications
- Route to receiver with best match between $a$ and $s_i$

![Anycast Diagram]
Composable Services

- Use stack of IDs to encode successive operations to be performed on data (e.g., transcoding)
- Don’t need to configure path between services

Composable Services (cont’d)

- Both receivers and senders can specify operations to be performed on data
Heterogeneous Multicast

- Both receivers and senders can specify operations to be performed on data

Indirection: Summary

- We’ve seen indirection used in many ways:
  - Multicast
  - Mobility
  - Internet indirection

- Uses of indirection:
  - Sender does not need to know receiver id
    - Do not want sender to know intermediary identities
  - Elegant
  - Transparency of indirection is important
  - Performance: is it more efficient?
  - Security: important issue for I3