

# CS 268: Lecture 25

## Internet Indirection Infrastructure

Ion Stoica  
Computer Science Division  
Department of Electrical Engineering and Computer Sciences  
University of California, Berkeley  
Berkeley, CA 94720-1776

### Motivations

- 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
  - ...

## Why?

- 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

3

## IP Solutions

- 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

4

## Application Level Solutions

- Implement the required functionality at the application level, e.g.,
  - Application level multicast (e.g., Narada, Overcast, Scattercast...)
  - Application level mobility
- Disadvantages:
  - Efficiency hard to achieve
  - Redundancy: each application implements the same functionality over and over again
  - No synergy: each application implements usually only one service; services hard to combine

5

## Key Observation

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

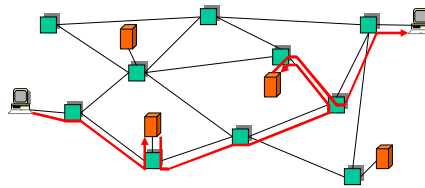
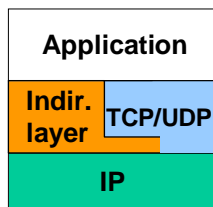
**“Any problem in computer science can be solved by adding a layer of indirection”**

6

## Our Solution

Build an efficient indirection layer  
on top of IP

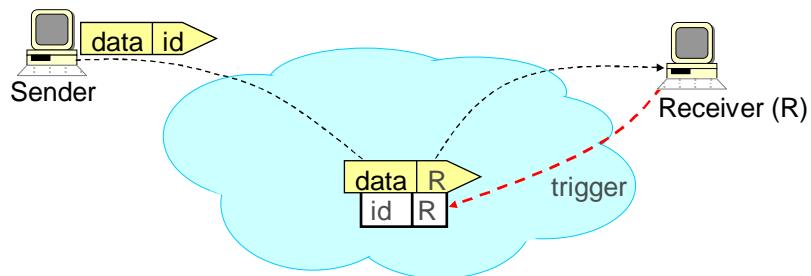
- Use an overlay network to implement this layer
  - Incrementally deployable; don't need to change IP



7

## 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



8

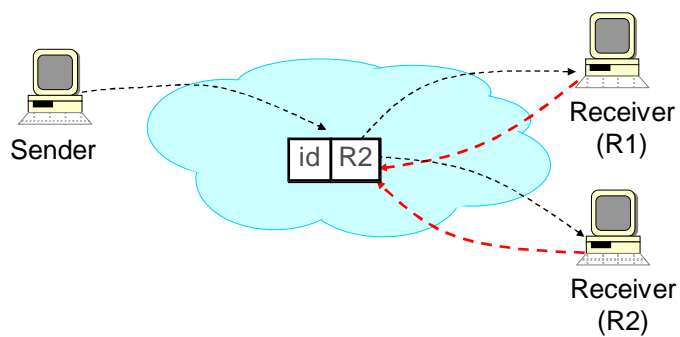
## Service Model

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

9

## Mobility

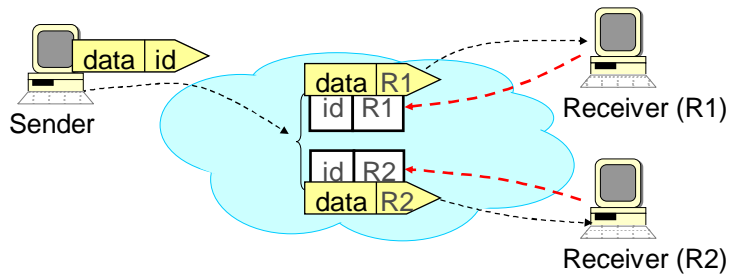
- Host just needs to update its trigger as it moves from one subnet to another



10

## Multicast

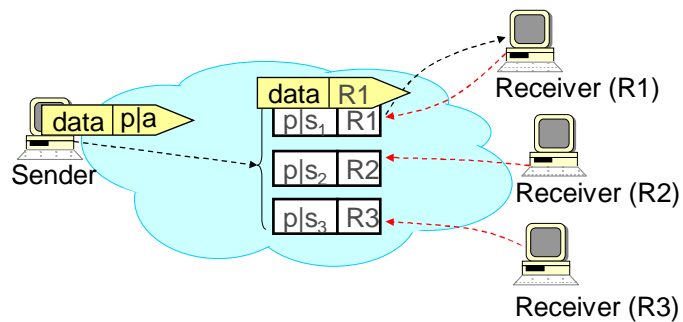
- Receivers insert triggers with same identifier
- Can dynamically switch between multicast and unicast



11

## Anycast

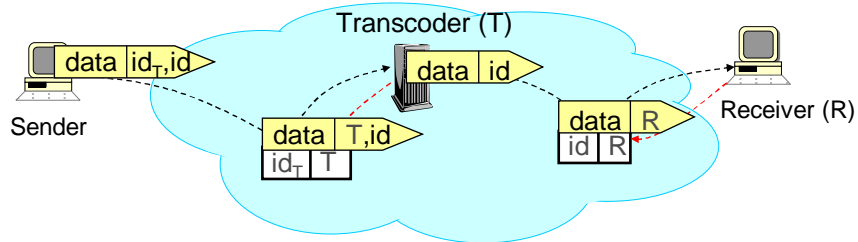
- Use longest prefix matching instead of exact matching
  - Prefix  $p$ : anycast group identifier
  - Suffix  $s_i$ : encode application semantics, e.g., location



12

## Service Composition: Sender Initiated

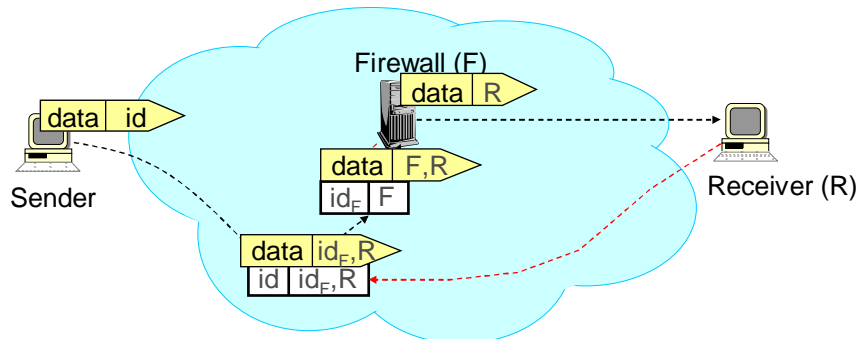
- Use a stack of IDs to encode sequence of operations to be performed on data path
- Advantages
  - Don't need to configure path
  - Load balancing and robustness easy to achieve



13

## Service Composition: Receiver Initiated

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



14

## Outline

- Implementation
  - Examples
  - Security
  - Applications

15

## Quick Implementation Overview

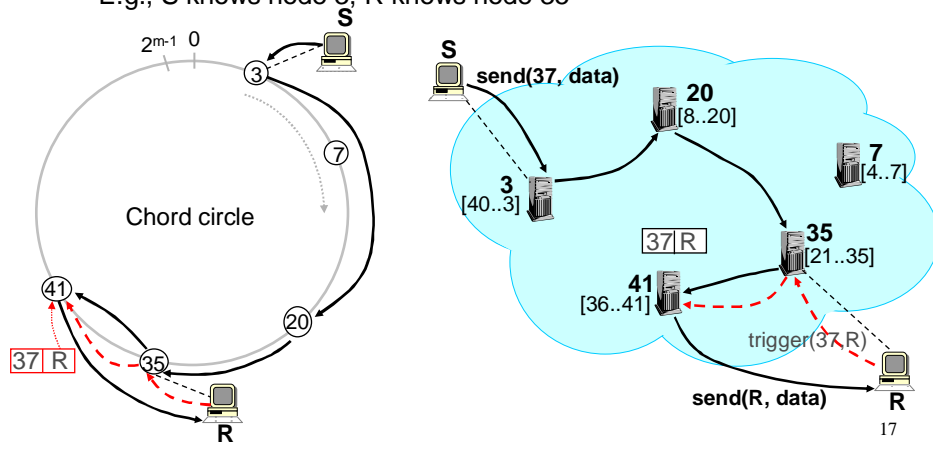
- i3 is implemented on top of Chord
  - But can easily use CAN, Pastry, Tapestry, etc
- Each trigger  $t = (id, R)$  is stored on the node responsible for  $id$
- Use Chord recursive routing to find best matching trigger for packet  $p = (id, data)$

16



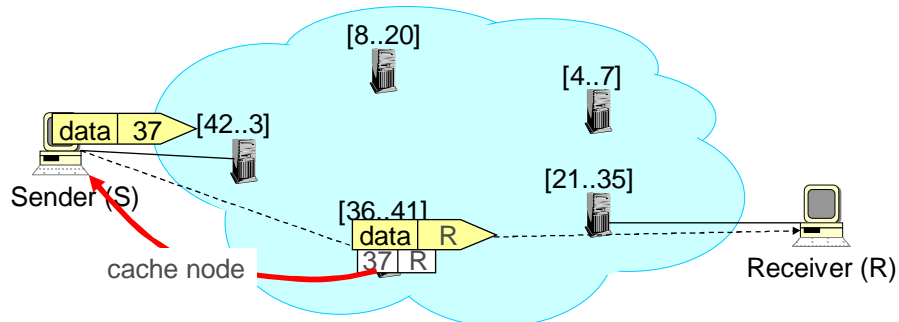
## Routing Example

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



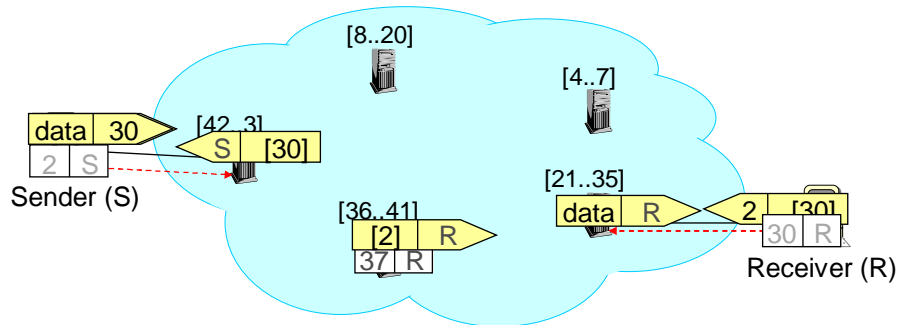
## Optimization #1: Path Length

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



## 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



19

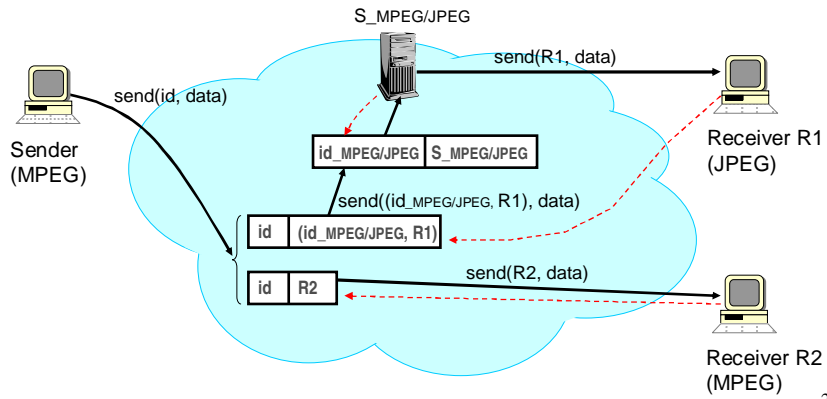
## Outline

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

20

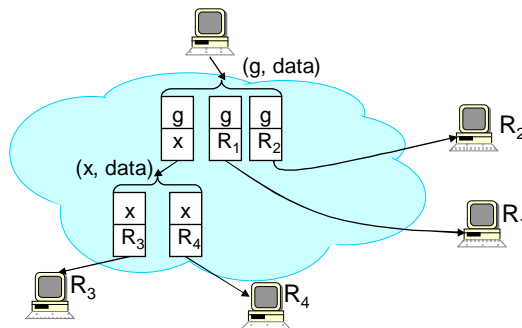
## Example 1: Heterogeneous Multicast

- 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 trigger of bounded degree



## Example 2: Scalable Multicast (Discussion)

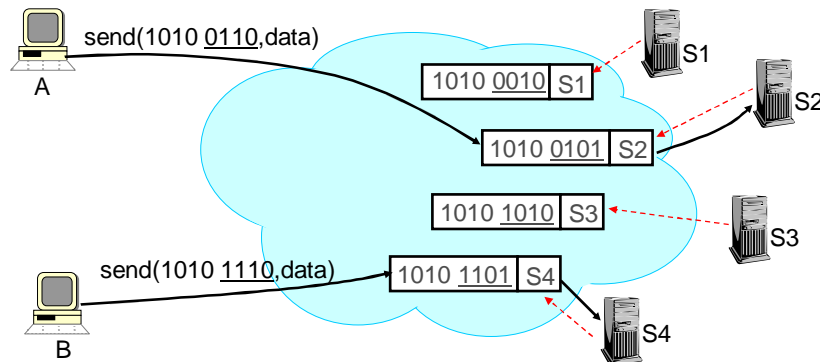
Unlike IP multicast, i3

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

23

## Example 3: Load Balancing

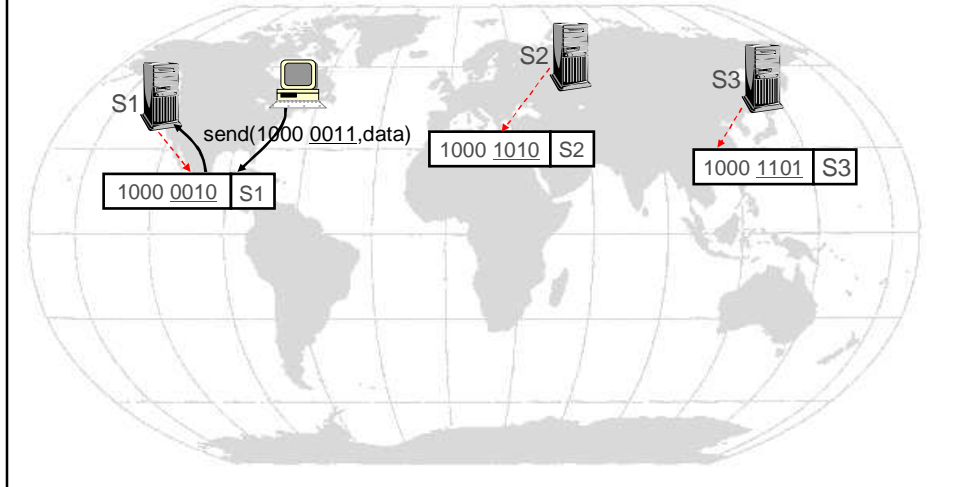
- Servers insert triggers with IDs that have random suffixes
- Clients send packets with IDs that have random suffixes



24

## Example 4: Proximity

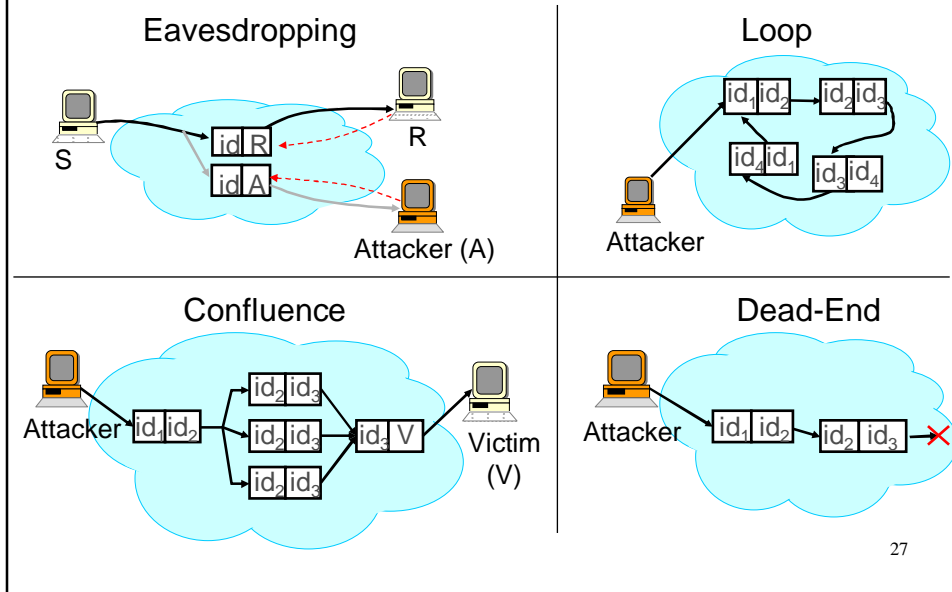
- Suffixes of trigger and packet IDs encode the server and client locations



## Outline

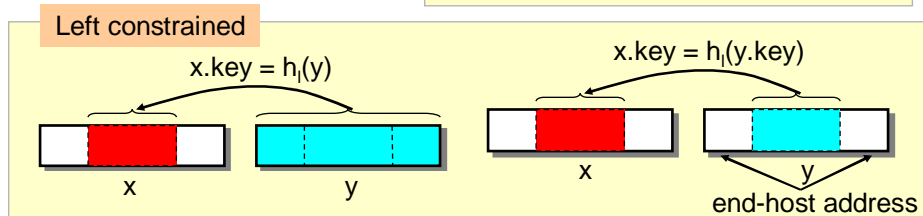
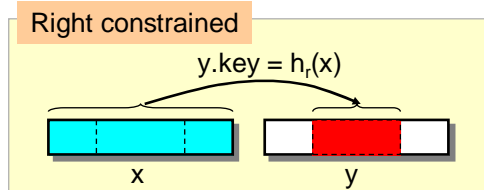
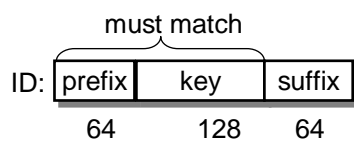
- Implementation
- Examples
- Security
- Applications

## Some Attacks



## Constrained Triggers

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



## Attacks & Defenses

<b>Attack</b> <b>Defense</b>	Trigger constraints	Pushback	Trigger challenges	Public ID constraints
Eavesdropping & Impersonation	✓			✓
Loops & Confluences	✓			
Dead-ends		✓		
Reflection & Malicious trigger-removal	✓		✓	
Confluences on i3 public nodes				✓

29

## Outline

- Implementation
- Examples
- Security
  - Applications
    - Protection against DoS attacks
      - Routing as a service
      - Service composition platform

30

## In a Nutshell

- Problem scenario: attacker floods the incoming link of the victim
- Solution: stop attacking traffic before it arrives at the incoming link
  - Today: call the ISP to stop the traffic, and hope for the best!
- Our approach: give end-host control on what packets to receive
  - Enable end-hosts to stop the attacks in the network

31

## Why End-Hosts (and not Network)?

- End-hosts can better react to an attack
  - Aware of semantics of traffic they receive
  - Know what traffic they want to protect
- End-hosts may be in a better position to detect an attack
  - Flash-crowd vs. DoS

32



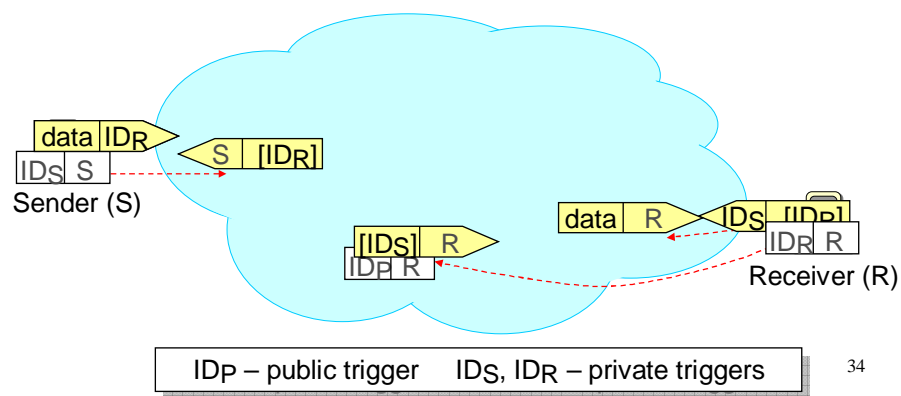
## Some Useful Defenses

1. White-listing: avoid receiving packets on arbitrary ports
2. Traffic isolation:
  - Contain the traffic of an application under attack
  - Protect the traffic of established connections
3. Throttling new connections: control the rate at which new connections are opened (per sender)

33

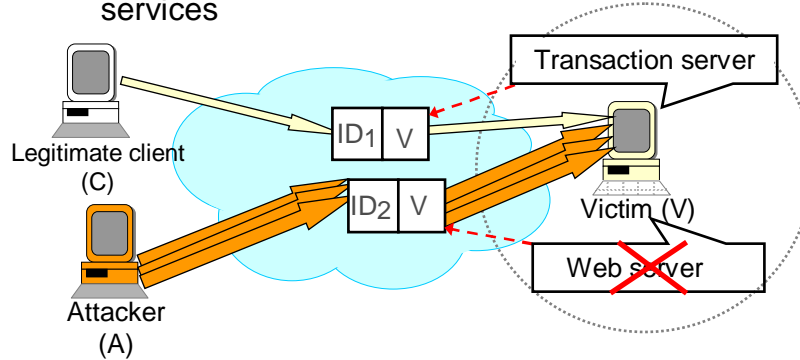
## 1. White-listing

- Packets not addressed to open ports are dropped in the network
  - Create a public trigger for each port in the white list
  - Allocate a private trigger for each new connection



## 2. Traffic Isolation

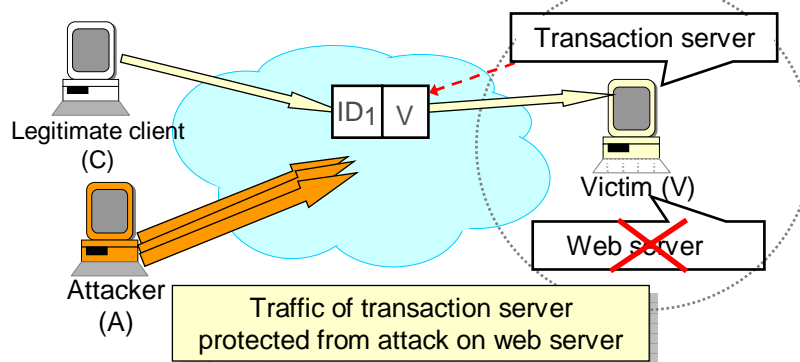
- Drop triggers being flooded without affecting other triggers
  - Protect ongoing connections from new connection requests
  - Protect a service from an attack on another services



35

## 2. Traffic Isolation

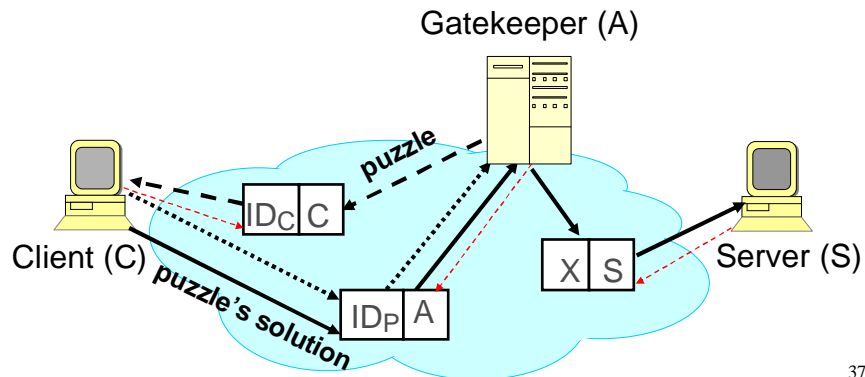
- Drop triggers being flooded without affecting other triggers
  - Protect ongoing connections from new connection requests
  - Protect a service from an attack on another services



36

### 3. Throttling New Connections

- Redirect new connection requests to a gatekeeper
  - Gatekeeper has more resources than victim
  - Can be provided as a 3<sup>rd</sup> party service



37

### Outline

- Implementation
- Examples
- Security
- Architecture Optimizations
- Applications
  - Protection against DoS attacks
  - Routing as a service
  - Service composition platform

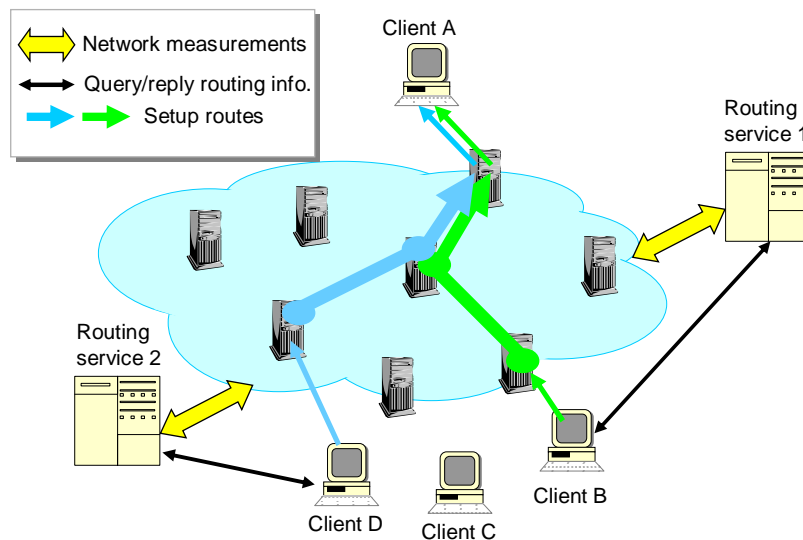
38

## 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

39

## Routing as a Service (cont'd)



40

## Outline

- Implementation
- Examples
- Security
- Architecture Optimizations
- Applications
  - Protection against DoS attacks
  - Routing as a service
  - Service composition platform

41

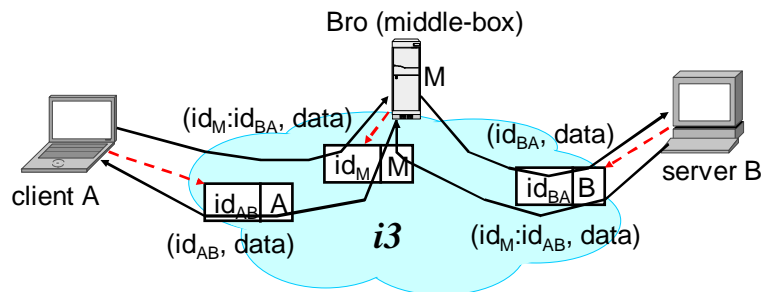
## Service Composition Platform

- 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

42

## Example

- Use Bro system to provide intrusion detection for end-hosts that desire so



43

## Design Principles

- 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

44

## Design Principles (cont'd)

	Host	Infrastructure
Internet & Infrastructure overlays		Data plane Control plane
p2p & End-host overlays	Data plane Control plane	
i3	Control plane	Data plane

45

## 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
  - Explore the implications of changing the communication abstraction; already done in other fields
    - Direct addressable vs. associative memories
    - Point-to-point communication vs. Tuple space (in Distributed systems)

46