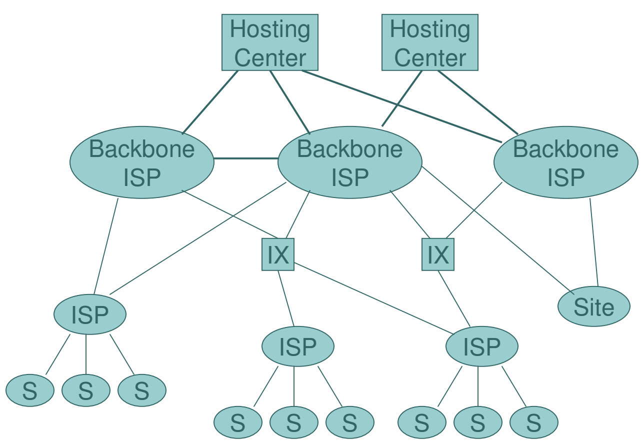
CS514: Intermediate Course in Computer Systems

Lecture 32: April 9, 2003

"Internet Indirection Infrastructure (i3 and Secure-i3)"

Remember this?

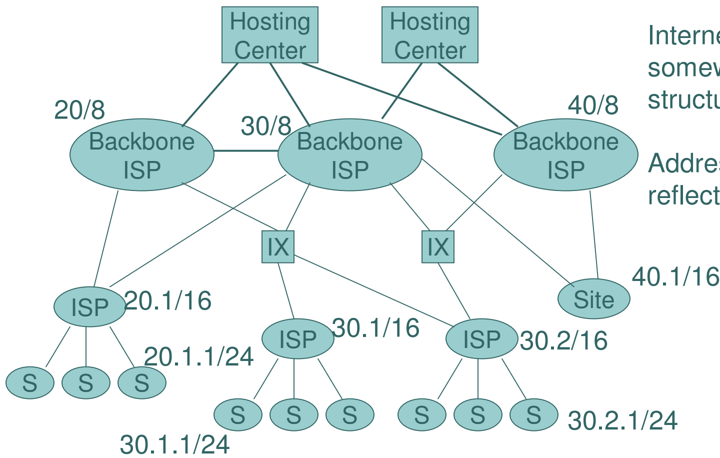
CS514



Internet has a somewhat hierarchical structure.

IP address assignment

CS514



Internet has a somewhat hierarchical structure.

Address assignment reflects this hierarchy

IP address are “overloaded”

CS514

- Two roles:
  - Acts as a host identifier
    - Used to determine which host sends or receives a packet
  - Acts as a hierarchical address
    - Used to route packet to a host across the Internet

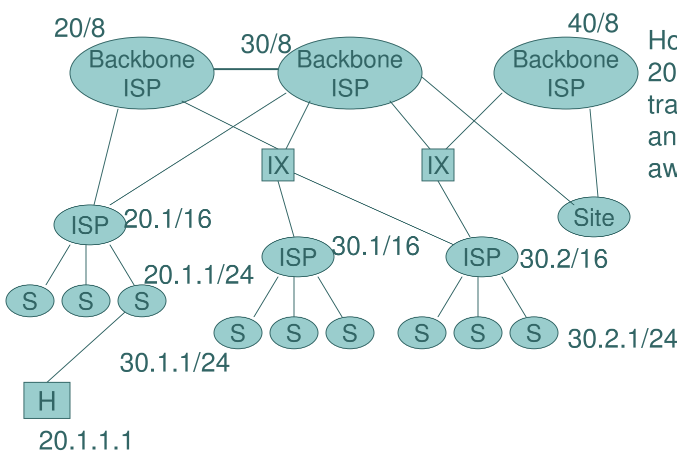
Overloading is important for security

CS514

- "Reverse routability"
- A host can spoof its identify in transmitted packets
  - Spoofed source address
- But return packets won't go back to it
  - The routing infrastructure prevents it
- Therefore, a host cannot easily pretend to be another host

But overloading limits mobility

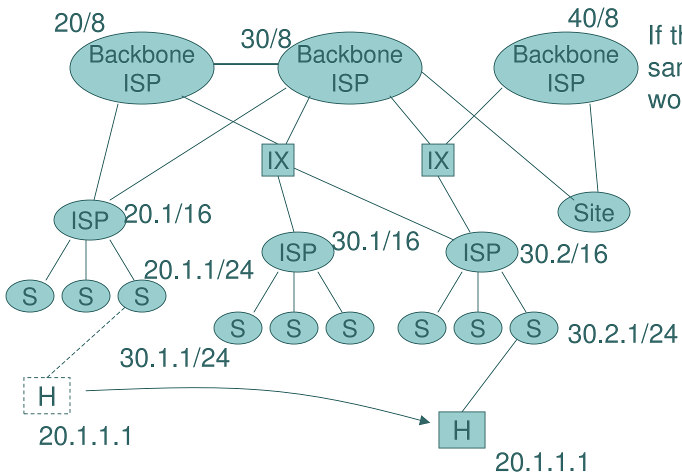
CS514



Host with address 20.1.1.1 cannot transparently move to another location far away

But overloading limits mobility

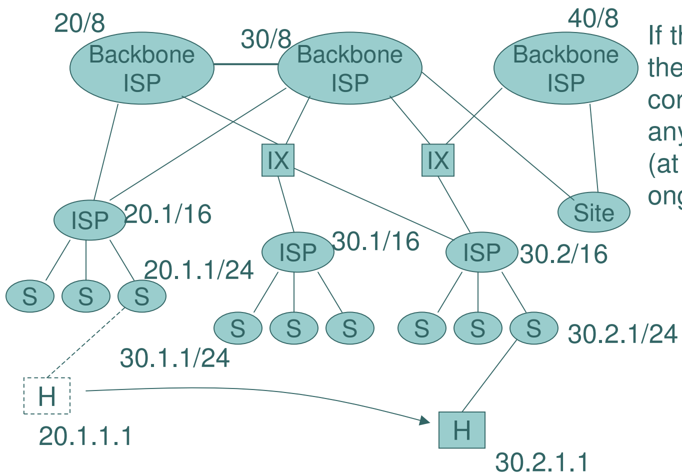
CS514



If the address stays the same, routing won't work anymore

But overloading limits mobility

CS514



If the address changes, the host won't be correctly identified anymore
(at least not for ongoing connections)

IP multicast/unicast addresses are not overloaded

CS514

- They are pure identifiers, not hierarchical addresses
- But as a result, multi/anycast don't scale well
  - Routers must keep per-group state
- Multi/anycast also have security issues
  - In the absence of higher-level security mechanisms, any host can join a group

Overlay multicast

CS514

- Poor IP multicast scaling has led to the use of overlay multicast
- IP hosts form multicast tree
  - Tunnel over IP
- Typically application specific
  - Streaming (Real Networks, etc.)

Question for today

CS514

- Is there another *simple model* for an *infrastructure service* that has scalable unicast, multicast, and anycast services?
- Internet Indirection Service (i3) is an interesting answer
  - Ion Stoica (Berkeley) et. al.

i3: a DHT application

CS514

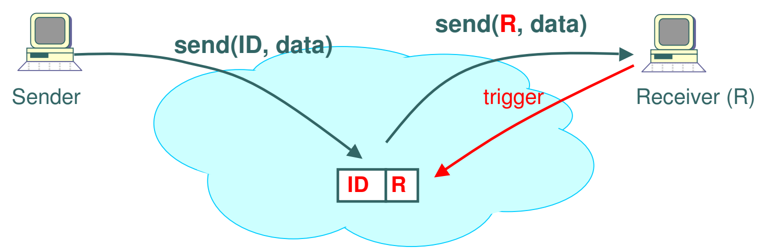
- Hosts use flat identifiers
- Hosts can make them up anytime, as many as they want
- A DHT is used to map identifier to IP address
  - Unicast, anycast, or multicast
  - Also composable services
- But this DHT built from high-end, stable infrastructure boxes
  - Not "P2P"
- Best explained by example…

Drawings and some slides care of Ion Stoica

i3 model

CS514

- Receiver maintains a mapping between ID and R (address) in the DHT
  - Using soft-state "trigger"
- Sender sends to ID, DHT forwards to R



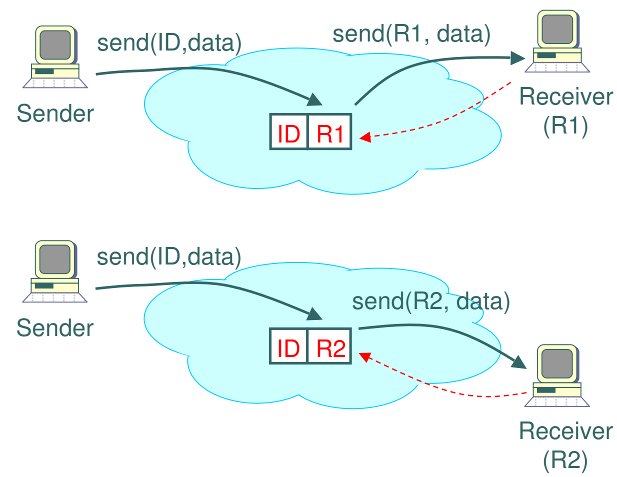
Service Model

CS514

- API
  - sendPacket(*p*);
  - insertTrigger(*t*);
  - removeTrigger(*t*); ***// optional***
- Best-effort service model (like IP)
- Triggers are periodically refreshed by end-hosts
- Reliability, congestion control, and flow-control implemented at end-hosts

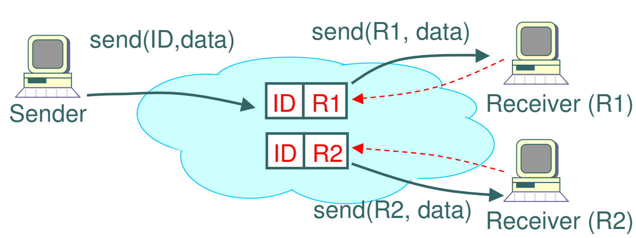
Mobility (invisible to sender)

CS514



Multicast

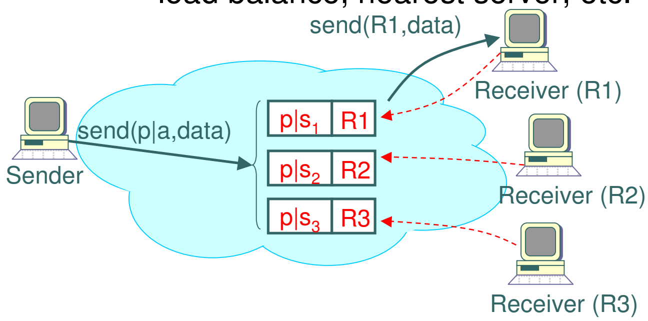
CS514



Anycast

CS514

- i3 matching rule is actually longest prefix match
  - This allows various forms of anycast
    - load balance, nearest server, etc.



IDs are stackable

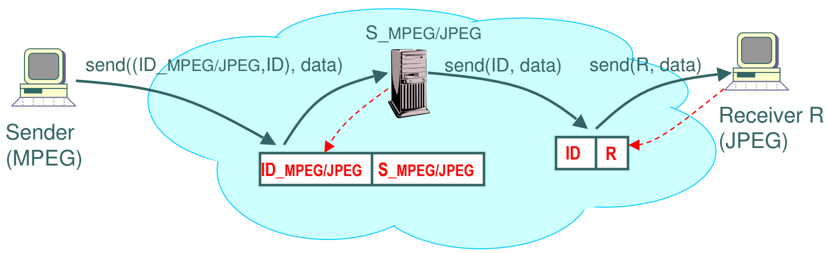
CS514

- Instead of *send(ID, data)*
  - Can have *send(ID-stack data)*
- Instead of *trigger(ID, R)*
  - Can have *trigger(ID, ID-stack)*
- Behavior at i3 node:
  - Pop stack until match found

Service composition

CS514

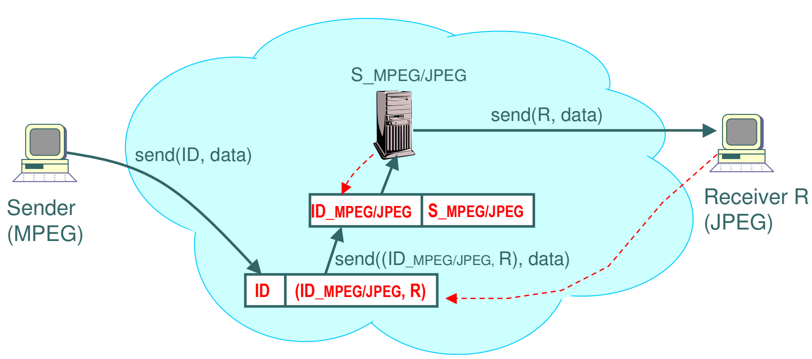
- Sender specified



Service composition

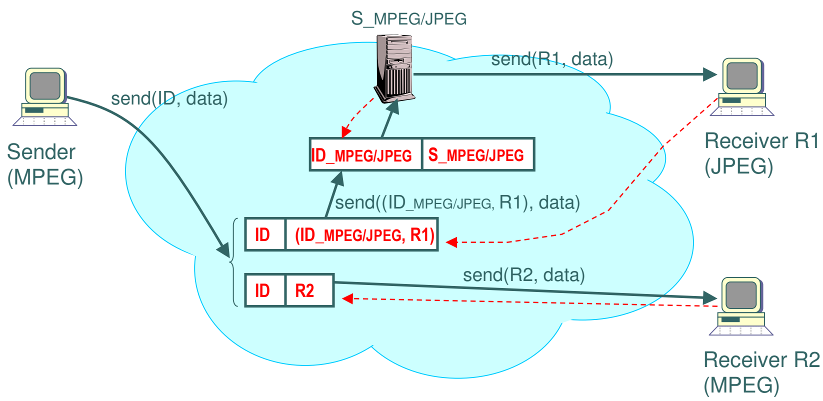
CS514

- Receiver specified



Even heterogeneous service multicast

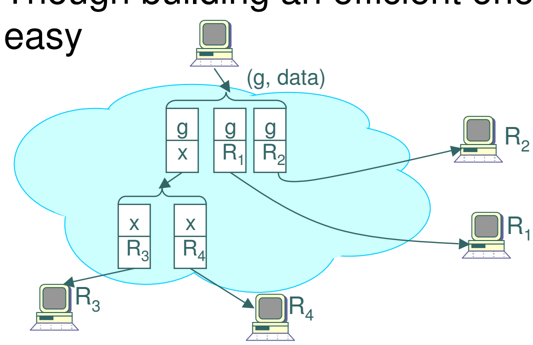
CS514



Trigger can map to another ID (as well to an address)

CS514

- Use to scale multicast, for instance
- End hosts can build this tree . . .
  - Though building an efficient one not easy



i3

CS514

- Clever

i3

CS514

- Clever
- Very Clever

i3

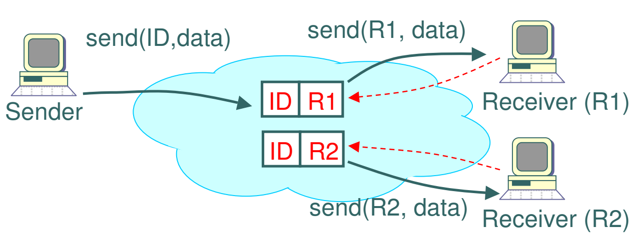
CS514

- Clever
- Very Clever
- But not without issues…

Eavesdropping problem

CS514

- R2 wishes to eavesdrop on R1's communications
  - Uses generalized multicast trigger



New work solves this

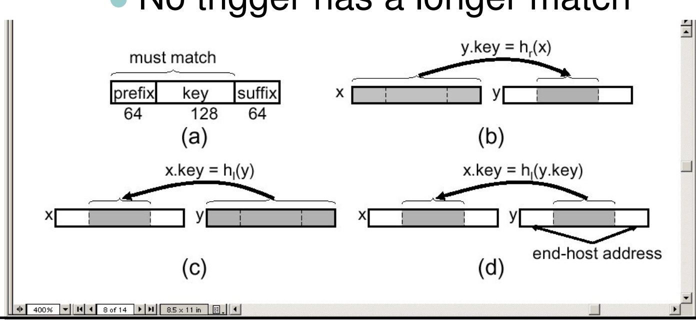
CS514

- Initial paper (SIGCOMM '02) suggested the use of e2e public key encryption
  - To securely negotiate a second pair of "private" IDs that the eavesdropper cannot guess
- A subsequent unpublished paper solves this an other problems
  - An i3 redesign called Secure-i3
- As well as develops a whole DoS defense around i3

Secure-i3 identifier

CS514

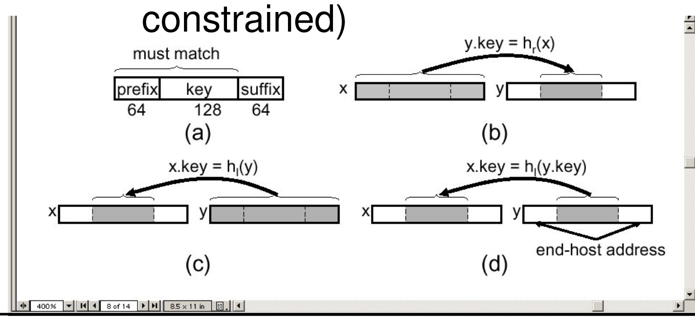
- ID composed of three parts
- Packet matches key iff:
  - Prefix and key match
  - No trigger has a longer match



Constraints on secure-i3 identifier

CS514

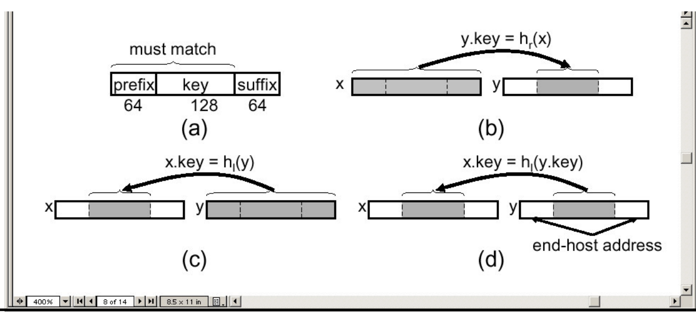
- Trigger(x,y) must satisfy one of three types of one-way hashes
  - (b) right constrained, (c) left constrained, (d) end-host (left constrained)



Constraints on secure-i3 identifier

CS514

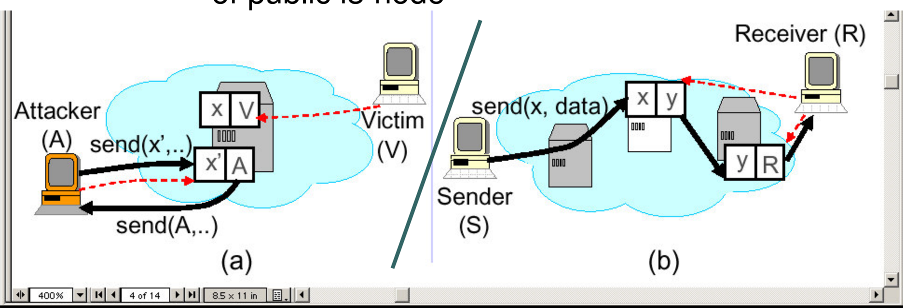
- Left constrained prevents eaves-dropping or impersonation
- Right constrained prevents a loop of triggers



Secure-i3 has defenses against DoS attacks

CS514

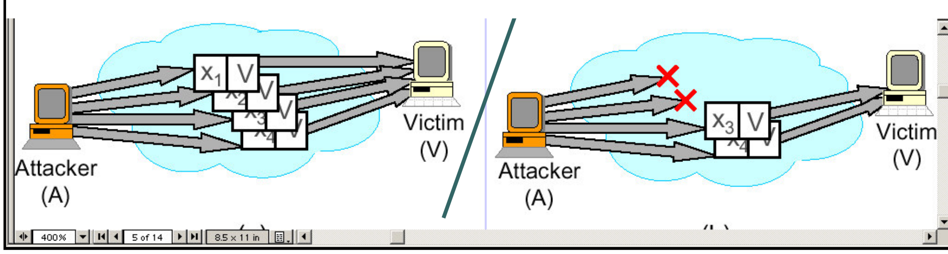
- Partition ID space into public and private
- Only allow public IDs to point to other IDs, not IP addresses
  - Makes it hard for attacker to learn IP address of public i3 node



Secure-i3 has defenses against DoS attacks

CS514

- Use multiple public IDs
  - Senders choose randomly among them
- Switch to private IDs to communicate
- Attacker has to attack all public IDs
- Remove some triggers to lessen attack



Secure-i3 has defenses against DoS attacks

CS514

- Other defenses as well
  - Slow down the attack
    - cryptographic puzzle
  - Evade the attack
    - Choose a different trigger
  - Multicast access control
    - Different IDs for senders and receivers
- Note that none of the DoS stuff works if the sender is not an i3 client!
  - IPv6-like deployment issues in this regard

i3 trust and service level agreement (SLA) issues

CS514

- i3 runs over DHT
  - In theory, any i3 node anywhere may be your i3 node
- But, user wants i3 node in user's ISP
  - Perhaps assign blocks of IDs to ISPs?
  - ISP's i3 nodes have DHT IDs in its block
  - ISP's users assign own IDs from block
- i3 paper doesn't talk about this
- Also, what is the relationship between ISPs, for instance for DHT security?

Who would be motivated to deploy i3?

CS514

- Elegant architecture, but what critical problem does it solve?
  - IP mobility not that important (yet), reasonably handled by existing standards
  - Anycast in a sense is handled by DNS
  - Multicast "style" very application dependent
    - App processing at each node, security, reliability, user tracking, acceptable latency, etc…
- Not sure I buy the argument that it is better to have a single (one-size-fits-all) solution to all problems

What about service composition?

CS514

- This is something without a clean parallel in the current architecture
- How is service composition done today?
  - Transparent to end host:
    - Service in physical path, transparent to end host
      - Firewall, HTTP-style authentication
    - Smart DNS directs user to service
      - Akamai web caches
  - End host configured to go through service
    - Web proxy, WAP (Wireless Application Protocol) gateway

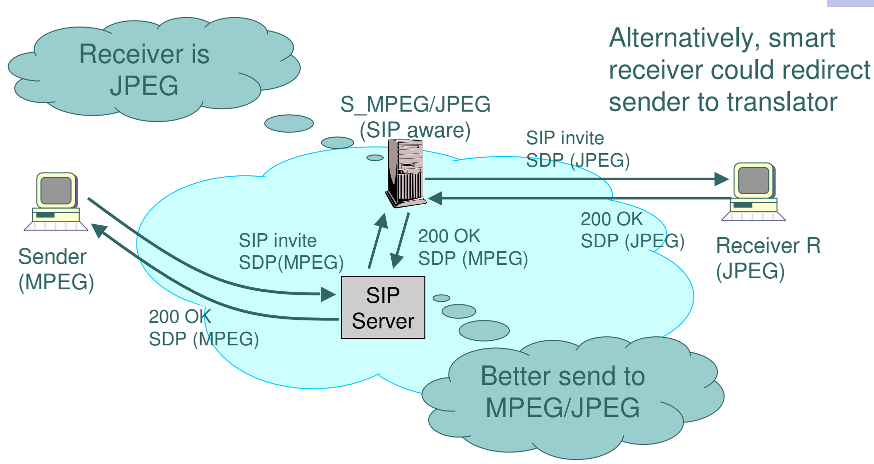
But more generally…

CS514

- SIP provides a basis for service composition
  - For instance, to route call through VoIP/circuit gateway
- SIP routing can be service aware
  - Service is encoded in the SDP (Session Description Protocol) part of the message
  - SDP is richer service description language than i3 IDs
- SIP also has a redirect feature (URI level)
  - I wonder if i3 could benefit from a redirect feature . . .

MPEG/JPEG example in SIP (one of many)

CS514



i3 versus SIP for service composition

CS514

- My intuition is that SIP is better
- Richer service description (SDP)
- Separate control from data
  - Data can take direct path, not go through service point, though this has pros and cons both ways
- Can be controlled by source, destination, or the middle
  - Probably i3 could be controlled in the middle, though they don't give an example
- Note, SIP could be used for anycast too

i3 Summary

CS514

- Very very interesting idea
- i3 creates an infinite number of new addresses, and allows hosts to create them at will and control routing to them!
- If we had i3 15 years go, we may not have DNS or SIP, would not have STUN, or IPv6 today
- But we do have those things, so hard to imagine that i3 will take off (doesn't fill a void)