Indirection: rather than reference an entity directly, reference it ("indirectly) via another entity, which in turn can or will access the original entity.

"Every problem in computer science can be solved by adding another level of indirection"
-- Butler Lampson
Multicast: one sender to many receivers

**Multicast**: act of sending datagram to multiple receivers with single “transmit” operation

- analogy: one teacher to many students

**Question**: how to achieve multicast

Network multicast

- router actively participate in multicast, making copies of packets as needed and forwarding towards multicast receivers

Multicast routers (red) duplicate and forward multicast datagrams
Internet multicast service model

multicast group concept: use of indirection

- hosts addresses IP datagram to multicast group
- routers forward multicast datagrams to hosts that have “joined” that multicast group
Multicast via Indirection: why?
Mobility and indirection

How do you contact a mobile friend?

Consider friend frequently changing addresses, how do you find her?

- search all phone books?
- call her parents?
- expect her to let you know where he/she is?

I wonder where Alice moved to?
Mobility and indirection:

- mobile node moves from network to network
- correspondents want to send packets to mobile node
- two approaches:
  - indirect routing: communication from correspondent to mobile goes through home agent, then forwarded to remote
  - direct routing: correspondent gets foreign address of mobile, sends directly to mobile
Mobility: vocabulary

**home network:** permanent “home” of mobile (e.g., 128.119.40/24)

**home agent:** entity that performs mobility functions on behalf of mobile, when mobile is remote

**Permanent address:** address in home network, can always be used to reach mobile e.g., 128.119.40.186
Mobility: more vocabulary

Permanent address: remains constant (e.g., 128.119.40.186)

Care-of-address: address in visited network. (e.g., 79.129.13.2)

visited network: network in which mobile currently resides (e.g., 79.129.13/24)

foreign agent: entity in visited network that performs mobility functions on behalf of mobile.

correspondent: wants to communicate with mobile

wide area network
Mobility: registration

End result:
- foreign agent knows about mobile
- home agent knows location of mobile
Mobility via Indirect Routing

1. Correspondent addresses packets using home address of mobile
2. Home agent intercepts packets, forwards to foreign agent
3. Foreign agent receives packets, forwards to mobile
4. Mobile replies directly to correspondent

- Home network
- Wide area network
- Visited network
Indirect Routing: comments

- Mobile uses two addresses:
  - **permanent address**: used by correspondent (hence mobile location is transparent to correspondent)
  - **care-of-address**: used by home agent to forward datagrams to mobile
- Foreign agent functions may be done by mobile itself
- **Triangle routing**: correspondent-home-network-mobile
  - Inefficient when correspondent, mobile are in same network
Indirect Routing: moving between networks

- Suppose mobile user moves to another network
  - Registers with new foreign agent
  - New foreign agent registers with home agent
  - Home agent updates care-of-address for mobile
  - Packets continue to be forwarded to mobile (but with new care-of-address)

- Mobility, changing foreign networks transparent: ongoing connections can be maintained!
Mobility via Direct Routing

correspondent forwards to foreign agent

foreign agent receives packets, forwards to mobile

mobile replies directly to correspondent

visited network

wide area network

home network

correspondent requests, receives foreign address of mobile
Mobility via Direct Routing: comments

- overcomes triangle routing problem
- non-transparent to correspondent: correspondent must get care-of-address from home agent
  - what happens if mobile changes networks?
Mobile IP

- RFC 3220
- has many features we’ve seen:
  - home agents, foreign agents, foreign-agent registration, care-of-addresses, encapsulation (packet-within-a-packet)
- three components to standard:
  - agent discovery
  - registration with home agent
  - indirect routing of datagrams
Mobility via indirection: why indirection?

- transparency to correspondent
- “mostly” transparent to mobile (except that mobile must register with foreign agent)
  - transparent to routers, rest of infrastructure
  - potential concerns if egress filtering is in place in foreign networks (since source IP address of mobile is its home address): spoofing?
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

![Diagram showing the process of sending and receiving packets using identifiers and triggers.](image-url)
Service model

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

- best-effort service model (like IP)
- triggers periodically refreshed by end-hosts
- reliability, congestion control, flow-control implemented at end hosts, and trigger-storing overlay nodes

Note: use of soft-state 😊. Why?
Discussion

- trigger similar to routing table entry
- essentially application layer publish-subscribe infrastructure
- application-level overlay infrastructure
- unlike IP, end hosts control triggers, i.e., end hosts responsible for setting and maintaining “routing tables”
- provides 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
Mobility

- receiver updates its trigger as it moves from one subnet to another
  - mobility transparent to sender
  - location privacy
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
Anycast (cont’d)

- route to any one in set of receivers
- receivers i in anycast group inserts same ID, with anycast qualifications

Sender

```plaintext
send(ID|a, data)
```

```
ID|s₁  | R1
ID|s₂  | R2
ID|s₃  | R3
```

Receiver (R1)

Receiver (R2)

Receiver (R3)
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

Sender (MPEG)

S_MPEG/JPEG

send(ID, data)

Receiver R (JPEG)

send((ID_MPEG/JPEG, R), data)
Heterogeneous multicast

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

```
Receiver R1
(MPEG)

ID: (ID_MPEG/JPEG, R1)

Sender
(MPEG)

ID: ID_MPEG/JPEG

S_MPEG/JPEG

send(ID, data)

Receiver R2
(MPEG)

ID: R2

send(R, data)

send((ID_MPEG/JPEG, R1), data)

send(R1, data)
```
Discussion of I3

- how would receiver signal ACK to sender? What is needed?
- does many-to-one fit well in this paradigm?
- security, snooping, information gathering: what are the issues?
Indirection: summary

We’ve seen indirection used for
- multicast
- mobility
- ...

- need infrastructure to store/fetch triggers
Implementing insert/retrieve: distributed hash table (DHT)

- hash table
  - data structure that maps “keys” to “values”
  - essential building block in software systems, distributed Hash Table (DHT)
  - similar, but spread across the Internet

- interface: just as in i**3
  - insert(key, value)
  - lookup(key)

- DHT: overlay (on top of internet) of nodes
  - each DHT node supports single operation:
  - given key as input; route messages toward node holding key
DHT in action
DHT in action

**Operation:** take key as input; route messages to node holding key
DHT in action: insert()

Operation: take key as input; route messages to node holding key
DHT in action: insert()

**Operation:** take key as input; route messages to node holding key
DHT in action: insert()

Operation: take key as input; route messages to node holding key
DHT in action: lookup()

Operation: take key as input; route messages to node holding key
DHT Design Goals

- an “overlay” network with:
  - flexible mapping of keys to physical nodes
  - small network diameter
  - small degree (fanout)
  - local routing decisions

- a “storage” or “memory” mechanism with
  - best-effort persistence (via soft state)
Example DHT: Chord

- m-bit identifier space: $2^m$ ids
- identifiers ordered on identifier ring (Chord ring) modulo $2^m$

- each key, $k$, maps to node on ring
  - # key values may be > # nodes
  - key $k$ maps to node on ring with next largest id: successor ($k$)

Circular 7-bit ID space: 128 keys
Basic lookup

“Where is key 80?”

“N90 has K80”

Lookup:
- node $i$ forwards to node $i+1$, until node holding $k$ is reached
- memory; $O(1)$, each node need only know one neighbor
- routing time: $O(N)$
Accelerating lookups

- Lookups accelerated by maintaining additional routing information
- Each node maintains a routing table with (at most) m entries (where N = 2^m) called **finger table**
- i\textsuperscript{th} entry in table at node n contains identity of first node, s, that succeeds n by at least 2^{i-1} on chord
- s = successor(n + 2^{i-1}) (all arithmetic mod 2)
Accelerating lookups

- Lookups accelerated by maintaining additional routing information
- Each node maintains a routing table with (at most) $m$ entries (where $N = 2^m$) called **finger table**
- $i^{th}$ entry in table at node $n$ contains identity of first node, $s$, that succeeds $n$ by at least $2^{i-1}$ on chord
- $s = \text{successor}(n + 2^{i-1})$ (all arithmetic mod 2)
Routing in Chord

example: routing node 1 to node 0:
Routing in Chord

**Example:** routing node 1 to node 0:
Routing in Chord

example: routing node 1 to node 0:
Routing in Chord

Example: routing node 1 to node 0:
Routing in Chord

example: routing node 1 to node 0:
Routing in Chord

- what happened?
  - constructed the binary number 15!
  - routing from \( x \) to \( y \) is like computing \( y - x \mod n \) by summing powers of 2

Diameter: \( \log n \)
Degree: \( \log n \)
Chord and DHTs

- **churn**: nodes join, leave system
  - stored keys must be moved to (on departure) or moved from (join) neighboring nodes
  - finger tables must be updated
  - too much churn: too much overhead
- middleware to store/retrieve values
Deconstructing DHTs

- DHT composed of
  - logical, underlying interconnection network
  - an “emulation scheme”
    - works on a “non-round” # of nodes
    - without global knowledge of network size
  - self-monitoring components
    - track and react to churn
6. Multiplexing

Multiplexing: *sharing* resource(s) among users of the resource.
Many dimensions of multiplexing

Other dimensions?
- time granularity
- 

- resource unavailability
- block, drop
- queue
- packet
- burst
- call
- user granularity
- on demand
- statistical reservations
- guaranteed
- shared among class
- per user
Packet-level multiplexing

Other dimensions?
- time granularity

- On demand
- Guaranteed
- Statistical reservations
- Per user
- Shared among class
- Queue
- Block, drop
- Resource unavailability
Scheduling and Policing Packets

- **scheduling**: choose next packet to send on link
- **FIFO (first in first out) scheduling**: send in order of arrival to queue
  - real-world example: stop sign
  - **discard policy**:
    - Tail drop: drop arriving packet
    - RED

![Diagram of packet scheduling and policing](image)
Scheduling Policies: more

**Strict Priority scheduling** transmit highest priority queued packet

- multiple classes, with different priorities
  - class may depend on marking or other header info, e.g. IP source/dest, port numbers, etc..
  - real world example: reservations versus walk-ins
Scheduling Policies: still more round robin scheduling:

- multiple classes
- cyclically scan class queues, serving one from each class (if available)
- real world example: 4-way stop (distributed scheduling)
Scheduling Policies: still more

Weighted Fair Queuing:
- generalized Round Robin
- each class gets weighted amount of service in each cycle
- real-world example: policeperson at intersection, giving priority to Rt 9 traffic
Policing Mechanisms

**Goal:** limit traffic to not exceed declared parameters

Three commonly-used criteria:

- **(Long term) Average Rate:** how many pkts can be sent per unit time (in the long run)
  - crucial question: what is interval length: 100 packets per sec or 6000 packets per min have same average!
- **Peak Rate:** e.g., 6000 pkts per min. (ppm) avg.; 15000 ppm peak rate
- **(Max.) Burst Size:** max. number of pkts sent consecutively (with no intervening idle)
Policing Mechanisms

**Token Bucket:** limit input to specified Burst Size and Average Rate.

- bucket can hold
- tokens generated at rate \( r \) token/sec unless bucket full
- over interval of length \( t \): number of packets admitted less than or equal to \( (r \cdot t + b) \)
Policing Mechanisms (more)

- token bucket, WFQ combine to provide guaranteed upper bound on delay, i.e., QoS guarantee!

\[
D_{\text{max}} = \frac{b}{R}
\]
Call-level multiplexing: routing and call admission in the telephone network

- Resource(s) unavailability
- Block, drop
- Queue
- Packet
- Burst
- Call
- User
- Granularity

- On demand
- Statistical reservations
- Guaranteed
- Shared among class
- Per user
Telephone network structure

3 level hierarchy:
- Wide area core
- Local exchange carrier (LEC), e.g. Verizon
- Central Office (CO) (e.g. Northampton, 169 Verizon COs in MA)
Telephone network topology

- **core**: one logical hop from one node to any other
  - multiple long distance providers -> multiple cores

- **LEC**
  - CO's fully meshed: 1 hop to any other CO
  - 1 or 2 connections to each core
Telephone network: routing

Routing

- If source, destination in same CO, connect them
- If source, destination in same LEC, take one hop path
- Otherwise, send call to core
  - Choose 1 or 2 hop path through core to dest. LEC
  - At most N-1 paths thru core to dest. LEC (routing decision is order of trying paths)
  - If no free path: block (busy)
Characteristics of telephone network routing

- reliable switches: seldom need to route around failure
- at most three “AS” hops: LEC->core->LEC
- highly connected: all paths one or two-hop paths
  - the routing problem: in which order to try 2-hop paths?
- telephone traffic is predictable: use past traffic to predetermined order in which routes attempted:
  - day-of-week, time, special occasion (e.g., mother’s day)
Dynamic, non-hierarchical routing (DNHR)

- 24-hour day divided into ~10 periods.
- State-independent routing: routing order downloaded into switch from central site
  - Try one-hop path to destination
  - If busy, try two-hop paths in specified order

Metastability:
- As load increases, more 2-hop paths taken
- 1 two-hop call can block 2 one-hop calls
- Metastability: degradation to where nearly all calls carried are two-hop calls
Avoiding metastability: trunk reservation

- **trunk reservation**: reserve some capacity on each link for 1-hop calls, purposefully block 2-hop calls (even if two 1-hop links are free)

**Q**: when to purposefully block 2-hop call?
- **con**: carried call generates revenue
- **pro**: carried call may block future 1-hop calls

**A**: shadow pricing: block 2-hop call if expected revenue lost due to blocked future 1-hop calls exceeds revenue gained by carrying the 2-hop call
Advanced telephony routing: TSMR

Trunk Status Map Routing (TSMR): switch measures current load, reports to central site
- central site computes new routing order
- routing order changed more frequently than DNHR
Advanced telephony routing: RTNR

Real-Time Network Routing (RTNR):
- switch dynamically determines order to try 2-hop routes
  - source knows loading of its own outgoing links
  - source asks dest. for loading of dest’s incoming links
- source can determine lightly loaded source-dest 2-hop paths

1. source knows SA, SF lightly loaded
2. Dest tells source CD, FD, ED lightly loaded
3. source chooses SFD
Burst-level multiplexing

- we’ve seen packet, call as unit of multiplexing
- **burst**: yet another unit of multiplexing

**TASI**: time assigned speech interpolation (AT&T mid 50’s)
- detect silence periods in voice conversation
- only send traffic during periods of speech activity ("talkspurts")
Multiplexing: what have we learned?

- fully-connected topology and lack of let-me-do-everything-possible-to-optimize-performance brings simplicity in routing
- predictable traffic (and measuring traffic in the first place, unlike the internet) makes allocating resources easier in telephone net
- lots of mechanism
  - particularly for packet-level sharing
  - mechanisms: for implementing policy