Design Principles

Goals:
- Identify, study common architectural components, protocol mechanisms, approaches do we find in network architectures?
- **Synthesis:** big picture

7 design principles:
- Separation of data, control
- Hard state versus soft state
- Randomization
- Indirection
- Network virtualization / overlays
- Resource sharing
- Design for scale
Signaling
Common network/protocol functions

- **Goals:**
  - Identify, study common architectural components, protocol mechanisms
  - *Synthesis:* big picture
  - *Depth:* important topics not covered in introductory courses

- **Overview:**
  - Signaling: TCP, Telephone network, Internet
    - Protocols
  - State handling
  - Randomization
  - Indirection
  - Service location
  - Network virtualization
Signaling

**Signaling:** Exchange of messages among network entities to enable (provide service) to connection/call

- Before, during, after connection/call
  - Call setup and teardown
  - Call maintenance
  - Measurement, billing
- Between
  - End-user <-> network
  - End-user <-> end-user
  - Network element <-> network element
Signaling is about state!

- “... exchange information between network components required to provide and maintain service”

  - Hard state: no periodic maintenance/explicit teardown
  - Soft state: expires/timers
  - What is better?
  - More in the next lecture
Signaling examples

- Internet
  - TCP handshake (connection setup/teardown)
  - RSVP (Resource Reservation Protocol, e.g., for QoS)
  - SIP (Session Initiation Protocol for Internet telephony)

- Telephone network
  - SS7 (Signaling System no. 7)
Signaling in the Internet

connectionless (stateless) forwarding by IP routers + best effort service = no network signaling protocols in initial IP design

- **New requirement:** Transport protocol needs state and variable initialization
- **TCP** Transport Control Protocol [RFCs 793, 1122, 1323, 2018, 2581]
TCP Connection Management

Recall: TCP sender, receiver establish “connection” before exchanging data segments

- Initialize TCP variables:
  - Seq. #s
  - Buffers, flow control info (e.g. RcvWindow)
  - MSS and other options
- **Client:** connection initiator, **server:** contacted by client

- Three-way handshake
  - Simultaneous open
- TCP Half-Close (four-way handshake)
- Connection aborts via RSTs
TCP Connection Management (2)

Three way handshake:

**Step 1:** Client end system sends TCP SYN control segment to server
- Specifies initial seq #
- Specifies initial window #

**Step 2:** Server end system receives SYN, replies with SYNACK control segment
- ACKs received SYN
- Allocates buffers
- Specifies server → receiver initial seq. #
- Specifies initial window #

**Step 3:** Client system receives SYNACK
TCP Connection Management (3)

**Closing a connection:**

Client closes socket:
```java
clientSocket.close();
```

**Step 1:** Client end system sends TCP FIN control segment to server

**Step 2:** Server receives FIN, replies with ACK. Closes connection, sends FIN.
TCP Connection Management (4)

**Step 3:** Client receives FIN, replies with ACK.

- Enters “time wait” – will respond with ACK to received FINs

**Step 4:** Server, receives ACK. Connection closed.

**Note:** With small modification, can handle simultaneous FINs.
TCP Connection Management (5)

TCP client lifecycle

- CLOSED: client application initiates a TCP connection
- TIME_WAIT: wait 30 seconds
- FIN_WAIT_2: receive FIN, send ACK
- SYN_SENT: send SYN
- ESTABLISHED: receive SYN & ACK, send ACK
- FIN_WAIT_1: receive ACK, send nothing
- FIN_WAIT_2: send FIN
- client application initiates close connection
TCP Connection Management (6)

TCP server lifecycle

- CLOSED: server application creates a listen socket
- LISTEN: receive SYN, send SYN & ACK
- SYN_RCVD: receive ACK, send nothing
- ESTABLISHED: receive FIN, send ACK
- CLOSE_WAIT: send FIN
- LAST_ACK: receive ACK, send nothing
TCP state machine
Telephone network

- Created 1876
- A global Infrastructure

- central office
- long haul network
- central office
- voice “trunk” lines
- central office
- subscriber access lines (twisted pair)
- Toll switch (Backbone Switch)
- PBX
Central office and local loop

- Each phone user (**subscriber**) has direct connection to switch in central office (**local loop**)
- Local loop has length 1 – 10 km
- Switches in central office called (**local**) **exchanges**
- Company providing local telephone service called **local exchange carrier** or **LEC** (e.g., Bell Atlantic)
**Private Branch Exchange (PBX)**

- **PBX (Private Branch Exchange)** telephone system within enterprise that switches calls on local lines; allows users to share fixed number of external lines to central office.
- Saves per-user cost (lines per user to central office).
Long-haul network

- Toll switches provide long-distance connectivity over long distance trunks
- ~500 toll switches in US
- Toll switch runs 100,000+ phone calls
How is voice transmitted?

Two ways:

- **Analog voice transmission**: voice channel allocated bandwidth of 3.5 kHz
- **Digital voice transmission**: analog voice stream converted to digital stream
  - Standard scheme: 8000 8-bit samples == 64Kbps
  - Wide band: e.g., 16,000 8-bit samples
The digital phone network

Until 1960s:
- Analog telephone network
- Frequency-division multiplexing

Today:
- Local loop analog
- ISDN (Integrated services Digital Network) all digital circuit switching technology. Available since the early-1990s (in Europe) or mid-1990s (US). No wide deployment in US
- Rest of network digital (based on TDM)
- Voice over IP (VoIP)

When do we get all digital network?
- All ISDN (No wide deployment in US)
- Another all digital – but not circuit-switched – telephony solution is IP telephony
First telephone switch digitizes voice call (8000 8-bit samples per second)
  - Switching method is TDM.

Switch multiplexes calls, interleaving samples in time. Call receives one 8-bit slot every 125 μs
All digital network

- Telephone at subscriber digitizes voice, sends one 8-bit sample every 125 µs
Digital Multiplexing

- Digital Signaling (DS) transmission hierarchy used in US for multiplexing digital voice channels

<table>
<thead>
<tr>
<th></th>
<th>Number of voice circuits</th>
<th>Bandwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>DS0</td>
<td>1</td>
<td>64 kbps</td>
</tr>
<tr>
<td>DS1</td>
<td>24</td>
<td>1.544 Mbps</td>
</tr>
<tr>
<td>DS2</td>
<td>96</td>
<td>6.312 Mbps</td>
</tr>
<tr>
<td>DS3</td>
<td>672</td>
<td>44.736</td>
</tr>
</tbody>
</table>
Addressing and routing

- Each subscriber has address (telephone number)
  - Hierarchical addresses
  - Example: Antonio’s Pizza in downtown Amherst

  Telephone address used for setting up route from caller to callee
Telephone network: Services

- Point-to-point POTS ("plain old telephone service") calls
- Special telephone numbers
  - 800 (888) number service: Free call to customer
  - 0180 number service: Fixed fee call for customer
  - 900 number service: Bill caller
  - Numbers for life (invariant under provider change)
- Caller ID (identify the caller at the receiver)
- Calling card/third part charging
- Call routing (to end user):
  - Prespecified by time-of-day
- “Follow me” service
- Incoming/outgoing call restrictions
- Support for cellular roaming:
  - “Home” number routed to current cell location

All of this requires signaling!
Telephone network: AIN

**AIN**: Advanced Intelligent (phone) Network: Migration from service-in-the-switch to service logic external to (on top of) switching systems

- Looks like Internet architecture:
  - E.g., DNS is at application layer; RIP, OSPF, BGP above IP
- AIN advantages:
  - Introduce new services rapidly
  - Open interfaces
    - Vendor customization
    - Vendor independence of services
Telephone network: Circuit-switched voice trunks (data plane)

subscriber access lines (twisted pair)

voice “trunk” lines (carrying multiple calls)
Telephone: Data and control plane
SS7: Telephone signaling network
SS7: Telephone network signaling

- **Out-of-band signaling**: telephony signaling carried over *separate* media from telephone calls (data)
  - Enables signaling between any switches (not just directly-connected)
  - Enables signaling during call (not just before/after)
  - Enables higher-than-voice-data-rate signaling
  - Security: *In-band* tone signaling helps phone phreaks; *Out of band* signaling more secure

- Signaling System 7 (SS7) network: *Packet-switched*
  - Calls itself circuit-switched

- Lots of redundancy (for reliability) in signaling network links, elements
SS7: Telephone network

- Signaling *between telephone network elements*:

  - **Signaling switching point (SSP):**
    - Attach directly to end user
    - Endpoints of SS7 network

  - **Signaling transfer point (STP):**
    - Packet-switches of SS7 network
    - Send/receive/route signaling messages

  - **Signaling control point (SCP):**
    - “Services” go here
    - E.g., database function
Example: Signaling a POTS call

1. Caller goes offhook, dials callee. SSP A determines to route call via SSP B. Assigns idle trunk A-B

2. SSP A formulates Initial Address Message (IAM), forwards to STP W

3. STP W forwards IAM to STP X

4. STP X forwards IAM to SSP B
Example: Signaling a POTS call (2)

5. B determines it serves callee, creates address completion message (ACM[A,B,trunk]), rings callee phone, sends ringing sound on trunk to A

6. ACM routed to Z to Y to A

7. SSP A receives ACM, connects subscriber line to allocated A-B trunk (caller hears ringing)
Example: Signaling a POTS call (3)

8. Callee goes off hook, B creates, sends answer message to A (ANM[A,B,trunk])

9. ANM routed to A

10. SSP A receives ANM, checks caller is connected in both directions to trunk. *Call is connected!*
Example: Signaling a 800 call

800 number: Logical phone number
- Translation to physical phone number needed, e.g., 1-800-CALL_ATT translates to 1-162-962-1943

1. Caller dials 800 number, A recognizes 800 number, formulates translation query, send to STP W
2. STP W forwards request to M
3. M performs lookup, sends reply to A
Example: Signaling a 800 call (2)

800 number: Logical phone number
- Translation to physical phone number needed

4. A begins signaling to set up call to number associated with 800 number
Example: SS7 protocol stack

TCAP: Application layer protocols: 800 service, calling card, call return, cellular roaming

SCCP: Demultiplexing to multiple upper layer applications

SS7-specific network, link, physical layer protocols

- move to IP (RFC 2719)?
Signaling: Discussion

- 800 logical-number-to-physical number translations: Looks like DNS
- Q: Differences?
  - In DNS end system generates request; DNS is transparent to IP network- network layer in phone net does 800-service location translation for you [phone net has more “smarts in net”]
  - DNS is more decentralized: Hierarchical
  - Anyone can run a name server, but not a 800 server
  - ...
- Q: Where is state stored?
  - In POTS call: SSP stores circuit allocation, start/stop time
  - In 800 call: STPs know where to go for 800 service; in Internet, DNS location transparent to IP routers (knowledge of where to go to DNS service is in end-systems, not in router – intelligence at the edge)
- Q: Internet versus SS7 / telephone network for accessing services
  - Adding new services more difficult than in Internet ...
  - Billing per TCP connection?
  - Administration (sending bills etc.) takes the biggest share in call billing.
Signaling in the Internet

connectionless (stateless) forwarding by IP routers + best effort service = no network signaling protocols in initial IP design

- **New requirement**: Application layer protocol, that enables users to be reachable independent of the device and his location

- **SIP**: Session Initiation Protocol [RFC 3261]
  - IETF protocol
  - All telephone calls and video conference calls take place over the Internet
  - People are identified by names or e-mail addresses, rather than by phone numbers.
  - You can reach the callee, no matter where the callee roams, no matter what IP device the callee is currently using.
SIP Services

- **Setting up a call**
  - Provides mechanisms for caller to let callee know she wants to establish a call
  - Provides mechanisms so that caller and callee can agree on media type and encoding
  - Provides mechanisms to end call

- **Determine current IP address of callee**
  - Maps mnemonic identifier to current IP address

- **Call management**
  - Add new media streams during call
  - Change encoding during call
  - Invite others
  - Transfer and hold calls
Setting up a call to a known IP address

- Alice’s SIP invite message indicates her port number & IP address. Indicates encoding that Alice prefers to receive (PCM ulaw)
- Bob’s 200 OK message indicates his port number, IP address & preferred encoding (GSM)
- SIP messages can be sent over TCP or UDP; here sent over RTP/UDP
- Default SIP port number is 5060
Setting up a call (more)

- Codec negotiation
  - Suppose Bob doesn’t have PCM ulaw encoder
  - Bob will instead reply with 606 Not Acceptable Reply and list encoders he can use
  - Alice can then send a new INVITE message, advertising an appropriate encoder

- Rejecting a call
  - Bob can reject with replies “busy,” “gone,” “payment required,” “forbidden”
  - Media can be sent over RTP or some other protocol
Example of SIP message

INVITE sip:bob@domain.com SIP/2.0
Via: SIP/2.0/UDP 167.180.112.24
From: sip:alice@hereway.com
To: sip:bob@domain.com
Call-ID: a2e3a@pigeon.hereway.com
Content-Type: application/sdp
Content-Length: 885

c=IN IP4 167.180.112.24
m=audio 38060 RTP/AVP 0

Notes:
- HTTP message syntax
- SDP = session description protocol
- Call-ID is unique for every call.

- Here we don’t know: Bob’s IP address.
- Intermediate SIP servers is necessary
- Alice sends and receives SIP messages using the SIP default port number 5060
- Alice specifies in Via: Header that SIP client sends and receives SIP messages over UDP
Name translation and user location

- Caller wants to call callee, but only has callee’s name or e-mail address.
- Need to get IP address of callee’s current host:
  - User moves around
  - DHCP protocol
  - User has different IP devices (PC, PDA, car device)
- Result can be based on:
  - Time of day (work, home)
  - Caller (don’t want boss to call you at home)
  - Status of callee (calls sent to voicemail when callee is already talking to someone)

Service provided by SIP servers:
- SIP registrar server
- SIP proxy server
SIP registrar

- When Bob starts SIP client, client sends SIP REGISTER message to Bob’s registrar server (similar function needed by Instant Messaging)

Register Message:

REGISTER sip:domain.com SIP/2.0
Via: SIP/2.0/UDP 193.64.210.89
From: sip:bob@domain.com
To: sip:bob@domain.com
Expires: 3600
SIP proxy

- Alice sends’ invite message to her proxy server
  - Contains address sip:bob@domain.com
- Proxy responsible for routing SIP messages to callee
  - Possibly through multiple proxies.
- Callee sends response back through the same set of proxies.
- Proxy returns SIP response message to Alice
  - Contains Bob’s IP address
- Note: proxy is analogous to local DNS server
Example
Caller jim@umass.edu with places a call to keith@upenn.edu

(1) Jim sends INVITE message to umass SIP proxy. (2) Proxy forwards request to upenn registrar server. (3) Upenn server returns redirect response, indicating that it should try keith@eurecom.fr

(4) Umass proxy sends INVITE to eurecom registrar. (5) eurecom registrar forwards INVITE to 197.87.54.21, which is running keith’s SIP client. (6-8) SIP response sent back (9) media sent directly between clients.

Note: Also a SIP ack message, which is not shown.
Signaling in the Internet

connectionless (stateless) forwarding by IP routers + best effort service = no network signaling protocols in initial IP design

- **New requirement:** reserve resources along end-to-end path (end system, routers) for QoS for multimedia applications
- **RSVP:** Resource Reservation Protocol [RFC 2205]
  - “… allow users to communicate requirements to network in robust and efficient way.” i.e., signaling!
- Earlier Internet Signaling protocol: ST-II [RFC 1819]
- Designed with **multicast** in mind
Multicast group concept:

- Hosts send IP datagram pkts to multicast group
- Hosts that have “joined” that multicast group will receive pkts sent to that group
- Routers forward multicast datagrams to hosts
Multicast groups

- Class D Internet addresses reserved for multicast:
  
  | 1110 | Multicast Group ID |
  
- Host group semantics:
  - Anyone can “join” (receive) multicast group
  - Anyone can send to multicast group
  - No network-layer identification to hosts of members

- **Needs:** Infrastructure to deliver mcast-addressed datagrams to all hosts that have joined that multicast group
Joining a multicast group:

Two-step process

- **Local**: Host informs local multicast router of desire to join group: IGMP (Internet Group Management Protocol)
- **Wide area**: Local router interacts with other routers to receive multicast datagram flow
  - Many protocols (e.g., DVMRP, MOSPF, PIM)
IGMP: Internet Group Management Protocol

- **Host**: sends IGMP report when application joins mcast group
  - IP_ADD_MEMBERSHIP socket option
  - Host need not explicitly “unjoin” group when leaving
- **Router**: sends IGMP query at regular intervals
  - Host belonging to a mcast group must reply to query
IGMP

**IGMP version 1**
- **Router:** Host Membership Query msg broadcast on LAN to all hosts (for all groups)
- **Host:** Host Membership Report msg to indicate group membership
  - Randomized delay before responding
  - Implicit leave via no reply to Query
- RFC 1112

**IGMP v2:** Additions include
- **Group-specific query**
- **Leave Group msg**
  - Last host replying to Query can send explicit Leave Group msg
  - Router performs group-specific query to see if any hosts left in group
- RFC 2236

**IGMP v3:** Internet draft
RSVP design goals

1. Accommodate heterogeneous receivers (different bandwidth along paths)
2. Accommodate different applications with different resource requirements
3. Make multicast a first class service, with adaptation to multicast group membership
4. Leverage existing multicast/unicast routing, with adaptation to changes in underlying unicast, multicast routes
5. Control protocol overhead to grow (at worst) linear in # receivers
6. Modular design for heterogeneous underlying technologies
RSVP does not ...

- Specify how resources are to be reserved
  - Rather: a mechanism for communicating needs
- Determine routes packets will take
  - That’s the job of routing protocols
  - Signaling decoupled from routing
- Interact with forwarding of packets
  - Separation of control (signaling) and data (forwarding) planes
RSVP: Overview of operation

- Senders and receivers join a multicast group
  - Done outside of RSVP
  - Senders need not join group

- Sender-to-network signaling
  - *Path message*: make sender presence known to routers
  - Path teardown: delete sender’s path state from routers

- Receiver-to-network signaling
  - *Reservation message*: reserve resources from sender(s) to receiver (specified by the receiver)
  - Reservation teardown: remove receiver reservations

- Network-to-end-system signaling
  - Path error
  - Reservation error
Path msgs: RSVP sender-to-network signaling

- **Path message** contents:
  - **Address**: unicast destination, or multicast group
  - **Flowspec**: bandwidth requirements spec.
  - **Filter flag**: if yes, record identities of upstream senders (to allow packets filtering by source)
  - **Previous hop**: upstream router/host ID
  - **Refresh time**: time until this info times out

- **Path message**: communicates sender info, and reverse-path-to-sender routing info
  - **Also**: upstream forwarding of receiver reservations (later)
RSVP: Simple audio conference

- H1, H2, H3, H4, H5 both senders and receivers
- Multicast group m1
- No filtering: packets from any sender forwarded
- Audio rate: $b$
- Only one multicast routing tree possible
RSVP: Building up path state

- H1, ..., H5 all send path messages on $m1$:
  (address=$m1$, Tspec=$b$, filter-spec=no-filter, refresh=100)
- Suppose H1 sends first path message
RSVP: Building up path state

- Next, H5 sends path message, creating more state in routers
RSVP: Building up path state

- H2, H3, H5 send path msgs, completing path state tables
Reservation msgs: *Receiver-to-network* signaling

- Reservation message contents:
  - *Desired bandwidth:*
  - *Filter type:*
    - *No filter:* any packets address to multicast group can use reservation
    - *Fixed filter:* only packets from specific set of senders can use reservation
    - *Dynamic filter:* set of senders that can use reservation changes over time
  - *Filter spec* (*data for the filter, e.g., sender names*)

- Reservations flow upstream from receiver-to-senders, reserving resources, creating additional, *receiver-related* state at routers
RSVP: *Receiver* reservation – example 1

- H1 wants to receive audio from all other senders
- H1 reservation msg flows uptree to sources
- H1 only reserves enough bandwidth for 1 audio stream
- Reservation is of type “no filter” – any sender can use reserved bandwidth
RSVP: *Receiver* reservation – example 1

- H1 reservation msgs flows uptree to sources
- Routers, hosts reserve bandwidth $b$ needed on downstream links towards H1

![Diagram showing network topology with reservations and bandwidth allocation](image-url)
RSVP: *Receiver* reservation – example 1

- Next, H2 makes no-filter reservation for bandwidth $b$
- H2 forwards to R1, R1 forwards to H1 and R2
- R2 takes no action, since $b$ already reserved on L6

```
<table>
<thead>
<tr>
<th>m1</th>
<th>L1</th>
<th>L2</th>
<th>L6</th>
</tr>
</thead>
<tbody>
<tr>
<td>in</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>out</td>
<td>L1(b)</td>
<td>L2(b)</td>
<td>L6</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>m1</th>
<th>L3</th>
<th>L4</th>
<th>L7</th>
</tr>
</thead>
<tbody>
<tr>
<td>in</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>out</td>
<td>L3</td>
<td>L4</td>
<td>L7(b)</td>
</tr>
</tbody>
</table>
```
RSVP: Receiver reservation – issues

What if multiple senders (e.g., H3-H5) over link (e.g., L6)?

- Arbitrary interleaving of packets
- L6 flow policed by leaky bucket: if H3+H4+H5 sending rate exceeds b, packet loss will occur

```
<table>
<thead>
<tr>
<th>in</th>
<th>L1</th>
<th>L2</th>
<th>L6</th>
</tr>
</thead>
<tbody>
<tr>
<td>out</td>
<td>L1(b)</td>
<td>L2(b)</td>
<td>L6</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>in</th>
<th>L5</th>
<th>L6</th>
<th>L7</th>
</tr>
</thead>
<tbody>
<tr>
<td>out</td>
<td>L5</td>
<td>L6(b)</td>
<td>L7</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>in</th>
<th>L3</th>
<th>L4</th>
<th>L7</th>
</tr>
</thead>
<tbody>
<tr>
<td>out</td>
<td>L3</td>
<td>L4</td>
<td>L7(b)</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>in</th>
<th>L1</th>
<th>L2</th>
<th>L6</th>
</tr>
</thead>
<tbody>
<tr>
<td>out</td>
<td>L1</td>
<td>L2</td>
<td>L6</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>in</th>
<th>L5</th>
<th>L6</th>
<th>L7</th>
</tr>
</thead>
<tbody>
<tr>
<td>out</td>
<td>L5</td>
<td>L6</td>
<td>L7</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>in</th>
<th>L3</th>
<th>L4</th>
<th>L7</th>
</tr>
</thead>
<tbody>
<tr>
<td>out</td>
<td>L3</td>
<td>L4</td>
<td>L7</td>
</tr>
</tbody>
</table>
```

m1:

```
<table>
<thead>
<tr>
<th>in</th>
<th>L5</th>
<th>L6</th>
<th>L7</th>
</tr>
</thead>
<tbody>
<tr>
<td>out</td>
<td>L5</td>
<td>L6</td>
<td>L7</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>in</th>
<th>L3</th>
<th>L4</th>
<th>L7</th>
</tr>
</thead>
<tbody>
<tr>
<td>out</td>
<td>L3</td>
<td>L4</td>
<td>L7</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>in</th>
<th>L1</th>
<th>L2</th>
<th>L6</th>
</tr>
</thead>
<tbody>
<tr>
<td>out</td>
<td>L1</td>
<td>L2</td>
<td>L6</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>in</th>
<th>L5</th>
<th>L6</th>
<th>L7</th>
</tr>
</thead>
<tbody>
<tr>
<td>out</td>
<td>L5</td>
<td>L6</td>
<td>L7</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>in</th>
<th>L3</th>
<th>L4</th>
<th>L7</th>
</tr>
</thead>
<tbody>
<tr>
<td>out</td>
<td>L3</td>
<td>L4</td>
<td>L7</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>in</th>
<th>L1</th>
<th>L2</th>
<th>L6</th>
</tr>
</thead>
<tbody>
<tr>
<td>out</td>
<td>L1</td>
<td>L2</td>
<td>L6</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>in</th>
<th>L5</th>
<th>L6</th>
<th>L7</th>
</tr>
</thead>
<tbody>
<tr>
<td>out</td>
<td>L5</td>
<td>L6</td>
<td>L7</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>in</th>
<th>L3</th>
<th>L4</th>
<th>L7</th>
</tr>
</thead>
<tbody>
<tr>
<td>out</td>
<td>L3</td>
<td>L4</td>
<td>L7</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>in</th>
<th>L1</th>
<th>L2</th>
<th>L6</th>
</tr>
</thead>
<tbody>
<tr>
<td>out</td>
<td>L1</td>
<td>L2</td>
<td>L6</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>in</th>
<th>L5</th>
<th>L6</th>
<th>L7</th>
</tr>
</thead>
<tbody>
<tr>
<td>out</td>
<td>L5</td>
<td>L6</td>
<td>L7</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>in</th>
<th>L3</th>
<th>L4</th>
<th>L7</th>
</tr>
</thead>
<tbody>
<tr>
<td>out</td>
<td>L3</td>
<td>L4</td>
<td>L7</td>
</tr>
</tbody>
</table>
```

m1:
RSVP: Example 2

- H1, H4 are only senders
  - Send *path messages* as before, indicating filtered reservation
  - Routers store upstream senders for each upstream link
- H2 will want to receive from H4 (only)
RSVP: Example 2

- H1, H4 are only senders
  - Send *path messages* as before, indicating filtered reservation (with upstream routers)

<table>
<thead>
<tr>
<th></th>
<th>in</th>
<th>L1, L6</th>
</tr>
</thead>
<tbody>
<tr>
<td>out</td>
<td>L2(H1-via-H1) ; H4-via-R2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>L6(H1-via-H1)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>L1(H4-via-R2)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>in</th>
<th>L4, L7</th>
</tr>
</thead>
<tbody>
<tr>
<td>out</td>
<td>L3(H4-via-H4) ; H1-via-R2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>L4(H1-via-R2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>L7(H4-via-H4)</td>
<td></td>
</tr>
</tbody>
</table>
RSVP: Example 2

- Receiver H2 sends reservation message for source H4 at bandwidth $b$
  - Propagated upstream towards H4, reserving $b$

<table>
<thead>
<tr>
<th>in</th>
<th>L1, L6</th>
</tr>
</thead>
<tbody>
<tr>
<td>out</td>
<td>L2(H1-via-H1 ; H4-via-R2), L6(H1-via-H1), L1(H4-via-R2)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>in</th>
<th>L4, L7</th>
</tr>
</thead>
<tbody>
<tr>
<td>out</td>
<td>L3(H4-via-H4 ; H1-via-R2), L4(H1-via-R2), L7(H4-via-H4)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>in</th>
<th>L6, L7</th>
</tr>
</thead>
<tbody>
<tr>
<td>out</td>
<td>L6(H4-via-R3), L7(H1-via-R1)</td>
</tr>
</tbody>
</table>
RSVP: *Soft-state*

- Senders periodically resend path msgs to refresh (maintain) state
- Receivers periodically resend resv msgs to refresh (maintain) state
- Path and resv msgs have TTL field, specifying refresh interval

```
<table>
<thead>
<tr>
<th>in</th>
<th>L1, L6</th>
</tr>
</thead>
<tbody>
<tr>
<td>out</td>
<td>L2(H1-via-H1 ; H4-via-R2 (b))</td>
</tr>
<tr>
<td></td>
<td>L6(H1-via-H1 )</td>
</tr>
<tr>
<td></td>
<td>L1(H4-via-R2 )</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>in</th>
<th>L4, L7</th>
</tr>
</thead>
<tbody>
<tr>
<td>out</td>
<td>L3(H4-via-H4 ; H1-via-R2 )</td>
</tr>
<tr>
<td></td>
<td>L4(H1-via-R2 )</td>
</tr>
<tr>
<td></td>
<td>L7(H4-via-H4 (b))</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>in</th>
<th>L6, L7</th>
</tr>
</thead>
<tbody>
<tr>
<td>out</td>
<td>L6(H4-via-R3 (b))</td>
</tr>
<tr>
<td></td>
<td>L7(H1-via-R1 )</td>
</tr>
</tbody>
</table>
```
RSVP: *Soft-state*

- Suppose H4 (sender) leaves without performing teardown
- Eventually state in routers will timeout and disappear!

<table>
<thead>
<tr>
<th>in</th>
<th>L1, L6, L7</th>
</tr>
</thead>
<tbody>
<tr>
<td>out</td>
<td>L2(H1-via-H1); L6(H1-via-H1); L1( )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>in</th>
<th>L7</th>
</tr>
</thead>
<tbody>
<tr>
<td>out</td>
<td>L3( ); L4(H1-via-R2); L7( )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>in</th>
<th>L6, L7</th>
</tr>
</thead>
<tbody>
<tr>
<td>out</td>
<td>L6(I); L7(H1-via-R1)</td>
</tr>
</tbody>
</table>
Use cases for reservation/path refresh

- Recover from an earlier lost refresh message
  - Expected time until refresh received must be shorter than timeout interval! (short timer interval desired)
- Handle receiver/sender that goes away without teardown
  - Sender/receiver state will timeout and disappear
- Reservation refreshes will cause new reservations to be made to a receiver from a sender who has joined since receivers last reservation refresh
  - E.g., in previous example, H1 is only receiver, H3 only sender
  - Path/reservation messages complete, data flows
  - H4 joins as sender, nothing happens until H3 refreshes reservation, causing R3 to forward reservation to H4, which allocates bandwidth
RSVP: reflections

- Multicast as a “first class” service
- Receiver-oriented reservations
- Use of soft-state
Signaling Discussion:
SS7 vs. TCP vs. SIP vs. RSVP

- Similarities
- Differences