Signaling
Common network/protocol functions

Goals 2nd part of lecture:
- Focus on **principles and concepts**: Identify, study *common* architectural components, protocol mechanisms
- **Synthesis**: big picture
- **Depth**: important topics not covered in introductory courses

Overview:
- Where do we need **signaling**?
  - Signaling = set up a connection, establish state
  - TCP, Telephone network, Internet...
- When do we need **state**? And how to handle state?
  - TCP window size, cookies...
- **Randomization**?
  - Access to shared media (random backoffs)
- **Indirection**?
  - Peer-to-peer systems, ...
  - ... or mobility (mobile number stays the same, IP address also?!)
- Service location
- Network **virtualization**
  - Abstraction, decoupling of services from physical constraints, ...
Signaling

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

- When necessary? 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 all about state! (see next lecture!)
  - Hard state = state stays, no periodical maintenance (e.g., signaling to set up state)
  - Soft state = state expires / timers (e.g., signaling to refresh state)
  - What is better on Internet-scale?
  - ... soft
Signaling Examples?

- **Signaling in the Internet?**
  - Not much in early Internet („stateless“ IP-layer)!
  - **TCP** handshake (to set up connection)
  - **RSVP** (to reserve resources: Resource Reservation Protocol, e.g. for guaranteed QoS by bandwidth reservation)
  - **SIP** (Session Initiation Protocol for IP Telephony)

- **But also in telephone networks of course…**
  - **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

But there are new requirements! Which ones?

- Transport protocol needs state and variable initialization (e.g., for **reliable transfer**)
- Want to have **location-independent services** (e.g., SIP number, or mobile IP...)
- For **QoS services**, I want to reserve resources! (requires state too)
Signaling in the Internet

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

- **New requirement 1:** Transport protocol needs state and variable initialization

- **TCP:** Resource Reservation Protocol
  [RFCs 793, 1122, 1323, 2018, 2581]

- **State in TCP?**
  - Window sizes, SEQ#, ACK#, etc.
Example: TCP Connection Management

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

- Initialize TCP variables / state:
  - Seq. #s
  - Buffers, flow control info (e.g. RcvWindow)
  - MSS and other options

- Client: connection initiator, server: contacted by client

- Signaling:
  - Three-way handshake
    - Simultaneous open
  - TCP Half-Close (four-way handshake)
  - Connection aborts via RSTs
Recall: TCP Connection Set-up

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

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

**Step 3:** Client system receives SYNACK
Recall: Closing TCP Connection

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.
Recall: Close TCP Connection (2)

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

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

**Note:** With small modification, can handle simultaneous FINs.
Recall: TCP State Machine

TCP client lifecycle

Client initiates set-up and termination!

- **CLOSED**: client application initiates a TCP connection
  - send SYN

- **SYN_SENT**: receive SYN & ACK send ACK

- **ESTABLISHED**: client application initiates close connection
  - send FIN

- **FIN_WAIT_1**: receive ACK send nothing

- **FIN_WAIT_2**: receive FIN send ACK

- **TIME_WAIT**: wait 30 seconds
Recall: TCP State Machine

TCP server lifecycle

- CLOSED
  - receive ACK
  - send nothing

- LISTEN
  - receive SYN
  - send SYN & ACK

- SYN_RCVD
  - receive ACK
  - send nothing

- ESTABLISHED
  - receive FIN
  - send ACK

- CLOSE_WAIT
  - send FIN

- LAST_ACK
  - receive nothing
TCP State Machine
Telephone Network

- When created?
  - 1876
- Circuit or packet switching?
  - Circuit-switching but there is also packet-switched telephony, e.g.,
  - IP telephony...
- Currently a global infrastructure

![Diagram of a telephone network with nodes labeled central office, long haul network, PBX, and subscriber access lines. Connections show the flow of calls and billing.]
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. Advantages?
  - PBX: „Nebenstellenanlagen“
  - Allows users to share fixed number of external lines to central office ("~NAT")
  - Saves cost of line per user to central office
Long-haul Network

- Toll switches provide long-distance connectivity over long distance trunks
  - Provided e.g., by Telekom (unlike AT&T, not split up in regulation)
- ≈500 toll switches in US
  - Not so long ago: one switch has size of an entire house (compare to CISCO switches today)...
- Toll switch runs 100,000+ phone calls
How is voice transmitted?

Two ways:

- **Analog voice transmission (resources?)**
  - 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
  - Wider frequency spectrum / 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 time-division-multiplex (TDM))

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 Time Division Multiplex. (Even earlier: FDM…)

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
- Question: what are advantages and disadvantages of FDM and TDM? When is what better?
FDM vs TDM?

- More convenient for a **party**?
  - FDM more casual & relaxed... 😊

- **What is more simple/flexible?**
  - TDM non-contiguous => simpler?
  - TDM allows easier addition of additional users / channels?

- **Interference?**
  - TDM’s pulsating power?

- **Synchronization overhead?**

- **Dynamics? What if only someone has to say something?**
  - Both are too rigid! It would be nice if resource is shared flexibly among the participants that *currently* have to say something...
  - Such mechanisms exist! (e.g., CDM)
  - See also lecture on resource allocation!
Digital Multiplexing: Aggregation Hierarchy

- 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 Mbps</td>
</tr>
</tbody>
</table>

- Nowadays more flexible, some levels in aggregation hierarchy can be skipped…
How are subscribers found?

- **Addresses** and routing!
- Each subscriber has address = *telephone number*: How organized?
  - Hierarchical addresses
  - **Example**: Antonio’s Pizza in downtown Amherst

<table>
<thead>
<tr>
<th>country code</th>
<th>area code</th>
<th>number of local exchange</th>
<th>subscriber number</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>413</td>
<td>253</td>
<td>0808</td>
</tr>
</tbody>
</table>

- Telephone address used for *setting up route* from caller to callee
- Numbers typically change when I move, unlike mobile phone numbers!
  - What if I move? => *indirection concept* needed, see later!
- How did telephone routing work in the old times?
  - Operators called each other multi-hop!
Telephone Network: Services

- Point-to-point POTS ("plain old telephone system") calls
- Special telephone numbers
  - 800 (888) number service: Free call for customer
  - 0180 number service: Fixed fee call (decide how costs are shared)
  - 900 number service: bill caller more (e.g., horoscope services, etc.)
  - Numbers for life (e.g., keep when switching providers)
- Caller ID (transmit caller’s name to callee)
- Calling card/third part charging
- Call routing (to end user):
  - E.g., respecified by time-of-day, congestion, etc.!
- “Follow me” service
  - provide list of possible numbers, call will be directed to all of them!
- Incoming/outgoing call restrictions
- Support for cellular roaming:
  - “Home” number routed to current cell location

All these services require different signalling!
Example Telephone Network: AIN

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

- Service layer on top of switching: advantage?
  - Allows telephone companies to differentiate themselves by value-added services
  - Since 90s

- Looks like Internet philosophy:
  - 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, flexibility and innovation
Telephone network: Circuit-switched voice trunks (data plane)

Network consists of two parts:
subscriber access lines (twisted pair)...

... and voice “trunk” lines (carrying multiple calls)

This voice trunk network is called the data plane (transports the “data”). What about signaling?
Besides data plane, there is a control plane! This is called “out of band signalling”: not on “same paths” as communication! Why? (Can be done in parallel, with different technology, ...)

Packet-switched SS7 protocol

Circuit-switched telephone network
SS7: Telephone Signaling Network

Protocol for signalling here: SS7!
SS7: Telephone Network Signaling

- **Out-of-band signaling**: telephony signaling carried over *separate* network from telephone calls (data)
  - Allows for signaling between any switches (not just directly-connected)
  - Advantages of out-of-band signaling?
    - Allows for signaling **during** call (not just before/after)
    - Allows for higher-than-voice-**data-rate** signaling
    - Security: In-band tone signaling helps phone phreaks (e.g., get free calls by “toneing” ctrl info); out of band signaling more **secure**

- Signaling System 7 (SS7) network: *Packet-switched*
  - calls themselves circuit-switched (why? see exercises…)
- Lots of **redundancy** (for reliability) in signaling network links, elements
- SS7 can find new paths very quickly, features **high availability**
In-band or Out-of-band?

- In-band == data and control is done „serially“ in „the same protocol“ (advantage? disadvantage?)
  - 😊 easier, same errors happen, fewer connections (maybe lower delay)
  - 😞 interruption needed for control, no authentication on control level, worse QoS
  - along same physical or logical connection (out-of-band may use different ports for connection, but of course may still go over same physical link)

- What is HTTP?
  - In-band...: cannot do „other things“ during transfer

- What is FTP?
  - Seperate channels, can do control during transfer, so...
  - ... out-of-band
  - Makes sense for very long transfers of large files!
**SS7: Telephone Network**

- Signaling *between telephone network elements*:

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

  **Signaling control point (SCP):**
  - "Services" go here
  - E.g., database function

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

Note: "SS7" stands for signaling system, not "switching"
Example: Signaling a POTS Call?

What needs to be signalled?

- Is callee found? („number not available“)
- Is it ringing? Is it occupied?
- Callee picks up the phone!
- Etc.
Example: Signaling a POTS Call?

1. Caller goes offhook, dials callee. SSP A determines to route call via SSP B. Assigns so-called idle trunk A-B (for circuit).

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

3. STP W forwards IAM to STP X.

4. STP X forwards IAM SSP B.

SSP = service switching point
STP = service transfer point
Example: Signaling a POTS Call (2)

5. B determines it serves callee, creates **address completion message** (ACM[A,B,trunk]), “the ACK”, **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**)

![Diagram](image-url)
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 ("service number")
- Translation to physical phone number needed, e.g., 1-800-CALL_ATT translates to 162-962-1943 (see DNS, indirection!)

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 DB lookup, sends reply to A
Example: Signaling a 800 Call (2)

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

1. Now A begins signaling to set up call **to number associated with 800 number**
Example: SS7 Protocol Stack

Complex stack reminiscent of Internet (IP the future?): OSI hierarchy, application layer, etc.

TCAP: Application layer protocols: allows for provider differentiation: 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?
  - Logics not “on the edge” in SS7: 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 less centralized: hierarchical
  - Anyone can run a names server, but not a 800 server
  - ...

- **Q**: Where is state stored? POTS call vs 800 call?
  - 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?
  - … no way, too expensive, typically only 8k Byte per TCP connection, whereas average telephone call lasts 3 minutes!

- What is expensive in your call today?
  - Lion’s share actually administration (sending bills, etc.), not infrastructure!
Signaling in the Internet

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

- **New requirement 2**: Application layer protocol, that enables users to be reachable independent of the device and its location (**location-independent service**): requires signaling!

- **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 or IP addresses.
  - You can reach the callee, no matter where the callee **roams**, no matter what IP device the callee is currently using.
  - => need signalling!
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 ("Kürzel", e.g. email address style or home phone) identifier to current IP address
  - Can put it on website!

- Call management?
  - Quite flexible!
  - Add new media streams during call
  - Change encoding during call
  - Invite others
  - Transfer and hold calls

SIP not only for calls, but also video, text, instant messaging... Alternative, e.g.: H.323!
Setting up a call to a **known** IP address

- Alice’s SIP invite message indicates her port number & IP address, and preferred encoding for reception (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

Like HTTP...

Everything is ACKed

**Flexible pulse code modules**...

**In-band or out of band?** Different ports for data => out of band!
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

Payload: Alice wants audio and has IP 167...
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 are necessary: INVITE sent to SIP proxy (returns maybe voice mail box or current location, depending on caller!)

• Indirection via **SIP Registrar**
  - ~ authoritative name server

• Alice sends and receives SIP messages using the SIP default port number 506

• 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 since: why?
  - 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
  - „Authorative name server“
- SIP proxy server
  - „Local DNS server“
SIP registrar: Register IP address

Whenever Bob turns on SIP client, client sends SIP REGISTER message to Bob’s registrar server (similar function needed by Instant Messaging)

Register Message: binds IP address to nickname

```
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: Lookup of Bob

Alice sends her 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 DNS lookup + 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 3: reserve resources along end-to-end path (end system, routers) for...
  - e.g., 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
Internet multicast service model

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

- How are multicast groups addressed?
- Class D Internet addresses reserved for multicast:

\[ 1110 \text{ Multicast Group ID} \]

- 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

- Routing typically along spanning trees (also as no TTL on network layer!)

How to find a group spanning multiple providers!?
Joining a mcast group:

Two-step process

- **Local easy:** Host informs **local mcast router** of desire to join group: **IGMP** (*Internet Group Management Protocol*)

- **Wide area:** Local router interacts with other routers to receive mcast datagram flow (there is also a **BGP counterpart...**)

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 because router periodically checks ("soft-state approach")
- **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
- **Host:** Host Membership Report msg to indicate group membership (not group specific query!)
  - Randomized delay before responding...
  - ... to avoid storm
  - *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

More efficient...
Multicast Today

- Out of fashion? (Inconsistency problems?)
- Mostly ISP-local groups only
- Trend towards application layer multicast like...?
  - Content Distribution Networks (e.g., YouTube)
  - Peer-to-Peer Networks (may yield inefficient distribution)
  - Advantage?
  - Routers do not need to support multicast!
Resource Reservation Protocol (RSVP)

1. Signaling for Intserv architecture which guarantees individual Quality-of-Service (QoS)
2. For example: max delay in queues
3. Allows for bandwidth reservation along multicast trees (unicast = special case)
4. RSVP is soft-state, so…
5. … state will expire if not refreshed
6. Implies that signaling can be unreliable (but frequent enough…)
RSVP Design Goals

1. Accommodate heterogeneous receivers (different bandwidth along paths)
   - receiver-orientation: receiver reserves resources, allows for different qualities for example

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

7. Router in Internet are responsible to ensure the flow quality…

8. Pure signalling, no path semantic (no routing)
RSVP does not...

- Specify how resources are to be reserved
  - Rather: only mechanism for communicating needs

- Determine routes packets will take
  - That’s the job of routing protocols
  - Signaling decoupled from routing
  - Separation of control (signaling) and data (forwarding) planes
  - In-band or out-of-band?
RSVP: Overview of Operation

- Senders, receiver 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 (but no resource reservation)
  - Path teardown: delete sender’s path state from routers

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

- Network-to-end-system signaling
  - Path error
  - Reservation error
Senders’ Path Messages

- **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 (how to reach)
  - **Refresh time:** time until this info times out (soft-state!)

- **Path message:** communicates sender info, and reverse-path-to-sender routing info
  - Receiver reservation flow in opposite direction (upstream forwarding)

- **No path semantic:** What if route changes before state is updated?
  - Packets forwarded without reservations, „best effort“
Reservation: Example

Senders

Routers (need to support RSVP!)

Receivers

upstream
Reservation: Example

reserve bandwidth for this source: reservation forwarded upstream
Reservation: Example

Other receiver wants same bandwidth to same sender: what to reserve?

Only needs to add reservation to lower part!
Reservation: Example

What if first receiver also wants reserve data from 2nd sender?

Additional reservation if different data...
Example 2: Simple audio conference

- H1, H2, H3, H4, H5 both senders and receivers
- All for same multicast group m1
- No filtering: packets from any sender forwarded
- Audio rate: $b$
- Only one multicast routing tree possible
  - No path pinning necessary
Example 2: Building up path state

- H1, ..., H5 all send path messages on *m1*:
  
  (address=*m1*, Tspec=*b*, filter-spec=no-filter, refresh=100)

- Suppose sender H1 sends first path message

Routers store where msg for *m1* arrives (upstream info), and forward to other ports.
Example 2: Building up path state

- Next, H5 sends path message, creating more state in routers.

Why L1 in and out port? All senders are also receivers, need to receive data from other senders…
(spanning tree, don’t return directly, …)
Example 2: Building up path state

- H2, H3, H5 send path msgs, completing path soft state tables

No path semantics, simple forwarding tables for this group m1!
Receivers’ Reservation Messages

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

- Reservations flow upstream from receiver-to-senders, reserving resources, creating additional, receiver-related state at routers
Example 1: *Receiver Reservation*

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
  - Cannot receive all senders simultaneously over this connection!
- Reservation is of type “no filter” – any sender can use reserved bandwidth
Example 1: *Receiver Reservation*

- H1 reservation msg flows uptree to sources
- Routers and hosts reserve bandwidth b needed on downstream links towards H1 (along rooted spanning tree: downstream unique)

```
Example 1: *Receiver Reservation*

- H1 reservation msg flows uptree to sources
- Routers and hosts reserve bandwidth b needed on downstream links towards H1 (along rooted spanning tree: downstream unique)

```

```
```

```
```
Example 1: *Receiver Reservation*

- 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 (enough for same content, same group, see earlier example!)
Example 1: Receiver Reservation – Issues

What if multiple senders (e.g., H3, H4, 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 (see policing in next lectures...)

```
+-------------------+-------------------+
| H2               | H1               |
| L2               | L1               |
| b                | b                |
|                  |                  |
| R1               |                  |
| L6               |                  |
| b                |                  |
|                  |                  |
| L1(b)            |                  |
|                  |                  |
| L2(b)            |                  |
|                  |                  |
| L6               |                  |
| b                |                  |
|                  |                  |
| R2               |                  |
| L6               |                  |
| b                |                  |
|                  |                  |
| L7               |                  |
|                  |                  |
| R3               |                  |
| L4               |                  |
| b                |                  |
|                  |                  |
| L7               |                  |
| b                |                  |

m1: in L1 L2 L6
    out L1(b) L2(b) L6

m1: in L5 L6 L7
    out L5 L6(b) L7

m1: in L3 L4 L7
    out L3 L4 L7(b)
```
RSVP: Example 2

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

- **H1, H4 are only senders**
  - The two senders send *path messages* as before, indicating filtered reservation
  - Routers store for each outgoing link set of upstream senders (e.g., L2 has two upstream senders: to sender H1 via “router“ H1 and to sender H4 via router R2)

<table>
<thead>
<tr>
<th>In</th>
<th>L1, L6</th>
</tr>
</thead>
</table>
| Out | L2(H1-via-H1 ; H4-via-R2)  
     | L6(H1-via-H1) (only upstream sender)  
     | L1(H4-via-R2) (only upstream sender) |

<table>
<thead>
<tr>
<th>In</th>
<th>L4, L7</th>
</tr>
</thead>
</table>
| Out | L3(H4-via-H4 ; H1-via-R2)  
     | L4(H1-via-R2)  
     | L7(H4-via-H4) |

<table>
<thead>
<tr>
<th>In</th>
<th>L6, L7</th>
</tr>
</thead>
</table>
| Out | L6(H4-via-R3)  
     | L7(H1-via-R1) |
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>
<th>L2: only for sender H4!</th>
</tr>
</thead>
<tbody>
<tr>
<td>out</td>
<td>L2(H1-via-H1 ;H4-via-R2 ($b$))</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>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 requires “refresh”*

- 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>
Soft State: *Implicitly removed*

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

<table>
<thead>
<tr>
<th>in</th>
<th>L1,</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(H1-via-R2), L4(H1-via-R2), L7()</td>
</tr>
</tbody>
</table>

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

- Use of soft-state and refresh messages?
- Recover from an earlier lost refresh message
  - How often refreshing needed?
  - 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 automatically
- 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 (=> automatic join/leave)
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?

- **State**: Which is soft-state, which hard-state?
- RSVP: soft, TCP+calls: typically hard
- Which need **signaling**?
- All of them... as all have state
- Which make **reservation**?
- SS7, RSVP, but not TCP, etc.
- **State where**? In end system or in-network?
- RSVP: in network, TCP: in end systems, etc.
- A routing protocol **without signaling**?!?!?
- E.g., flooding, hot potato, etc. => no state!
- **Band**: In-band vs out-of-band?
- SS7 out-of-band, RSVP in-band, etc.
- Etc.