Multicast

Srinidhi Varadarajan
Topics

- Multipoint communications
- IP Multicast
  - Addressing
  - IGMP
- API support for multicast
  - IP multicast API
- Multicast application examples
  - IP multicast API: sender, recvr
Multipoint Communications

- Multipoint communications support communications between more than two hosts
  - One-to-many
  - Many-to-many
- Unlike broadcast, allows a proper subset of hosts to participate
- Example standards
  - IP Multicast (RFC 1112, standard)
  - ST-II (RFC 1819, experimental)
  - T.120 (Data conferencing)
  - ATM point-to-multipoint
Logical Multipoint Communications (1)

- Two basic *logical* organizations
  - Rooted: hierarchy (perhaps just two levels) that structures communications
  - Non-rooted: peer-to-peer (no distinguished nodes)

- Different structure could apply to control and data “planes”
  - Control plane determines how multipoint session is created
  - Data plane determines how data is transferred between hosts in the multipoint session
Logical Multipoint Communications (2)

Rooted

Non-Rooted
Non-rooted logical structure does not necessarily imply an implementation using multiple point-to-point connections.
Control Plane

- The control plane manages creation of a multipoint session

- Rooted control plane
  - One member of the session is the root, c_root
  - Other members are the leafs, c_leafs
  - Normally c_root establishes a session
    - Root connects to one or more c_leafs
    - c_leafs join c_root after session established

- Non-rooted control plane
  - All members are the same (c_leafs)
  - Each leaf adds itself to the session
Data Plane

- The data plane is concerned with data transfer

- Rooted data plane
  - Special root member, \(d\_root\)
  - Other members are leafs, \(d\_leafs\)
  - Data transferred between \(d\_leafs\) and \(d\_roots\)
    - \(d\_leaf\) to \(d\_root\)
    - \(d\_root\) to \(d\_leaf\)
  - There is no direct communication between \(d\_leafs\)

- Non-rooted data plane
  - No special members, all are \(d\_leafs\)
  - Every \(d\_leafs\) communicate with all \(d\_leafs\)
Forms of Multipoint Communications

- **Server-based -- rooted multipoint communications with server as $d\_root$**
  - Passive or inactive
    - Relay
    - Reflector
  - Active
    - Bridge or multipoint control unit (MCU)

- **Strictly peer-to-peer multipoint -- non-rooted**
  - Multicast
Passive Multipoint Server

- Server provides a relay or reflector service
  - Provides no processing of the data
- Minimum requirement is for transport-level semantics, so can operate at the transport or application level
Active Multipoint Server

- Server receives inputs from hosts and does application-level processing
  - Select receivers for “chat room” applications
  - Select video source for videoconferencing MCUs
- Server uses application-level semantics

Bridge/MCU

Voice activated video
- Current speaker shown
- Last speaker shown for current speaker
Multipoint Servers

- Transport mechanism can be general since only point-to-point communications must be supported between end hosts (clients) and the reflector (server)
  - Reliable or unreliable
  - Connection-oriented or connectionless
  - Stream or datagram

![Diagram of Multipoint Servers](image)
Multicast Communication (1)

- Communication is peer-to-peer
  - No infrastructure for inherently broadcast network
  - Requires router knowledge in routed networks
- Multicasting provided at network protocol level, e.g. IP multicast

![Diagram showing multicast communication]

- **Host that is a group member**
- **Host that is not a group member**
- **Multicast-enabled router**
Multicast Communication (2)

- Transport mechanism and network layer must support multicast
- Internet multicast limited to UDP
  - Unreliable: No acknowledgements or other error recovery schemes (perhaps at application level)
  - Connectionless: No connection setup (although there is routing information provided to multicast enabled routers)
  - Datagram: Message-based multicast
IP Multicast

- IP supports multicasting
  - Uses only UDP, not TCP (other experimental transport protocols support multicast)
  - Special IP addresses (Class D) identify multicast groups
  - Internet Group Management Protocol (IGMP) to provide group routing information
  - Multicast-enabled routers selectively forward multicast datagrams
  - IP TTL field limits extent of multicast

- Requires underlying network and adapter to support broadcast or, preferably, multicast
  - Ethernet supports multicast
### Multicast Addresses

- **Multicast addresses**
  - **Class D**: 224.0.0.0 — 239.255.255.255
  - “Well-known” and dynamic assignment within this group

<table>
<thead>
<tr>
<th>Class</th>
<th>Netid</th>
<th>Hostid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class A</td>
<td>0</td>
<td>hostid</td>
</tr>
<tr>
<td>Class B</td>
<td>10</td>
<td>netid</td>
</tr>
<tr>
<td>Class C</td>
<td>110</td>
<td>netid</td>
</tr>
<tr>
<td>Class D</td>
<td>1110</td>
<td></td>
</tr>
<tr>
<td>Class E</td>
<td>1111</td>
<td>reserved for future use</td>
</tr>
</tbody>
</table>
Multicast Address Assignment

- 224.0.0.0 — 224.0.0.255 reserved for routing, topology discovery, maintenance protocols
  - Not forwarded by routers
- 224.0.0.0 — 232.255.255.255 assigned (RFC 1700, ftp://ftp.isi.edu/in-notes/iana/assignments/multicast-addresses)
- 239.000.000.000 — 239.255.255.255 are "administratively scoped (RFC 2365)
  - 239.192.000.000 — 239.251.255.255 organization-local scope
  - 239.255.000.000 — 239.255.255.255 site-local scope
Multicast Versus Unicast Addressing

- **IP unicast address**
  - Statically bound to a *single* local network interface on a *single* IP network

- **IP host group (multicast) address**
  - Dynamically bound to a *set* of local network interfaces on a *set* of IP networks
  - Host group address not bound to a set of IP unicast addresses
Multicast Interference and Security

- Host cannot assume that …
  - Datagrams sent to any host group address will reach only the intended hosts, or
  - Datagrams received as a member of a transient host group are intended for the recipient

- Misdeliveries must be detected by the application

- If content is sensitive, then datagrams should …
  - Have their data encrypted, or
  - Be routed according to administrative controls that limit extent of transmission
Multicast Routing (1)

- Multicast routers do not maintain a list of individual members of each host group.
- Multicast routers do associate zero or more host group addresses with each interface.
Multicast Routing (2)

- Multicast router maintains table of multicast groups that are active on its networks
- Datagrams forwarded only to those networks with group members
IGMP (1)

- IGMP (RFC 1112, RFC 2236) provides information to routers so that it can build its multicast routing table
  - Hosts (service providers, not applications) send reports of all groups with at least one joined process
  - Routers send queries for reports
- IGMP message is carried by IP

**IP datagram**

- 20 bytes
- 8 bytes
### IGMP (2)

- **IGMP message format**
  - 4-bit IGMP version (=1,2,3)
  - 4-bit IGMP type
    - 1: Query sent by a router
    - 2: Report sent by a host
  - 32-bit group address (Class D IP address)
  - 16-bit checksum

<table>
<thead>
<tr>
<th>0</th>
<th>3</th>
<th>4</th>
<th>7</th>
<th>8</th>
<th>15</th>
<th>16</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>vers</td>
<td>type</td>
<td>not used</td>
<td>checksum</td>
<td>group address</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
IGMP (3)

- Joining a group
  - Host sends group report when the first process joins a given group
  - Application requests join, service provider (end-host) sends report

- Maintaining table at the router
  - Multicast router periodically queries for group information
  - Host (service provider) replies with an IGMP report for each group
  - Host does not notify router when the last process leaves a group -- this is discovered through the lack of a report for a query
MBONE: Internet Multicast Backbone

- The MBone is a virtual network on top of the Internet
  - Routers that support IP multicast
  - IP tunnels between such routers and/or subnets

Multicast router
- Route via tunnels
- Encapsulates IP in IP
- May do multiple sends
API Requirements

- The application program interface must explicitly support multicast
  - IP service interface extended to provide two new operations (RFC 1112)
    - JoinHostGroup (group-address, interface)
    - LeaveHostGroup (group-address, interface)

- JoinHostGroup binds a host group address to an interface
- LeaveHostGroup removes the binding
- These are conceptual, not the actual API calls (as we’ll see)
IP Multicast API

- Data is sent and received using a standard datagram socket
  - `sendto()` to send — or `send()` with prior `connect()`
  - `recvfrom()` to receive
- Host group address treated like standard IP address for `sendto()`, `recvfrom()`, and `connect()` calls
- Port numbers play standard role
- New socket options — set using `setsockopt()` — enable multicast
  - Protocol level is IP (IPPROTO_IP)
Add Membership Socket Option (1)

- **Option:** IP_ADD_MEMBERSHIP
- **Parameter:** Multicast address structure
- **Operation**
  - Supports “JoinHostGroup” of RFC 1112 — allows a host’s interface to join a multicast group
    - Required to receive multicast datagrams
    - Not required to send multicast datagrams
  - Each interface can be in multiple groups
  - Multiple interfaces can be in the same group
  - Causes host to send IGMP report if this is a new group address for this host
  - Tells network adapter multicast group address
Add Membership Socket Option (2)

- Example call to setsockopt():

```c
setsockopt(
    sock,            // socket
    IPPROTO_IP,      // level
    IP_ADD_MEMBERSHIP, // option
    (char *) &mreq,  // argument
    sizeof(mreq)    // argument size
);
```
Multicast Address Structure

- Multicast address structure specifies the multicast group address and the interface
  - Interface specified as an IP address
  - INADDR_ANY specifies use of the default multicast interface

```c
struct ip_mreq {
    struct in_addr imr_multiaddr; // group
    struct in_addr imr_interface; // interface
}
```

```c
char group[] = “234.5.6.7”;
mreq.imr_multiaddr.s_addr = inet_addr(group);
mreq.imr_interface.s_addr = INADDR_ANY;
```
Reusing Port Numbers

- What if you want to have multiple sockets on the same host listen to the same multicast group?
  - Need to bind the same port number to all sockets
  - This will cause an error when bind is called for the second and later sockets ... unless socket has been set to reuse address

- Set SO_REUSEADDR socket option

```
OptValue = 1;
setsockopt(sock, SOL_SOCKET, SO_REUSEADDR, (char *) &OptValue, sizeof(OptValue));
```
Drop Membership Socket Option (1)

- **Option:** IP_DROP_MEMBERSHIP
- **Parameter:** Multicast address structure
- **Operation**
  - Supports “LeaveHostGroup” of RFC 1112 — allows host to leave a multicast group
  - Host’s TCP/IP implementation maintains a counter for each group address
    - Incremented for IP_ADD_MEMBERSHIP
    - Decremented for IP_DROP_MEMBERSHIP
  - If count reaches zero
    - Tells adapter to drop multicast address
    - Won’t report group address for IGMP query
Drop Membership Socket Option (2)

- Need to set group address and interface in ip_mreq structure (same values as used with IP_ADD_MEMBERSHIP)
- Example call to setsockopt():

```c
setsockopt(
    sock,        // socket
    IPPROTO_IP,  // level
    IP_DROP_MEMBERSHIP, // option
    (char *) &mreq,  // argument
    sizeof(mreq)  // argument size
);
```
Receiving Multicast Data

- Create a standard SOCK_DGRAM socket
- Set SOL_REUSEADDR option for socket
- Bind address to socket
  - Specify port
- Set IP_ADD_MEMBERSHIP option for socket
  - Specify host group address
- After these steps complete successfully, receive multicast data for specified group address and port using recvfrom()
- Drop group membership when finished using IP_DROP_MEMBERSHIP option
Sending Multicast Data

- Use standard SOCK_DGRAM socket
- Sending alone does not require group membership
- To send multicast datagrams:
  - Use sendto() to send to appropriate group address and port number, or
  - Use connect() to set group address and port and then use send()
- Concerns (controlled with socket options)
  - Interface used to send: IP_MULTICAST_IF
  - Extent of multicast: IP_MULTICAST_TTL
  - Receiving own data: IP_MULTICAST_LOOP
Multicast Interface Socket Option (1)

- Option: IP_MULTICAST_IF
- Parameter: Interface (struct in_addr)
- Operation
  - Overrides the default for the interface is used to send multicast datagrams
  - Relevant only for hosts with multiple interfaces
  - Interface specified in IP_ADD_MEMBERSHIP option will take precedence
Multicast Interface Socket Option (2)

Example:

```c
struct in_addr if_addr;

setsockopt(
    sock, IPPROTO_IP, IP_MULTICAST_IF, (char *) &if_addr, sizeof(if_addr));
```
Time To Live Socket Option (1)

- Option: IP_MULTICAST_TTL
- Parameter: TTL value (int)
- Operation
  - Controls the time-to-live (TTL) value that IP will use for multicast datagrams
  - Default TTL is 1 — multicast datagrams will not leave the local network
  - To send multicast datagrams beyond the local network …
    - TTL must be greater than 1, and
    - Intermediate routers must support multicast
  - Group address 224.0.0.0 — 224.0.0.255 not routed, regardless of TTL value
Time To Live Socket Option (2)

- Example to set multicast TTL to 0
  - TTL = 0 will confine multicast traffic to local host

```c
int ttl = 0;

setsockopt(
    sock,            // socket
    IPPROTO_IP,      // level
    IP_MULTICAST_TTL, // option
    (char *) &ttl,   // argument
    sizeof(ttl)      // argument size
);
```
Multicast Loop Socket Option (1)

- Option: IP_MULTICAST_LOOP
- Parameter: Boolean (TRUE to enable)
- Operation
  - If enabled (default), socket will receive a copy of multicast datagrams that were sent on that socket
  - Even if disabled, host with two interfaces may receive a copy on the other interface(s)
Multicast Loop Socket Option (2)

- Example:

```c
BOOL opt = FALSE;

setsockopt(
    sock,       // socket
    IPPROTO_IP, // level
    IP_MULTICAST_LOOP, // option
    (char *) &opt, // argument
    sizeof(opt)   // argument size
);
```
You should now be able to …

- Describe and distinguish between different forms of multipoint communications
- Describe the operation of IP multicast
- Describe the addressing scheme for IP multicast
- Describe the basic operation of IGMP
- Associate application actions with IGMP and IP multicast operation
- Analyze and design multicast applications using both the IP multicast API