UNIX Network Programming

An Overview of IP Multicasting

Douglas C. Schmidt

Introduction

- Standard RPC mechanisms offer point-to-point semantics
- Many applications require more flexible communication semantics
  - *e.g.*, 1-to-n or n-to-m communication
- Group communication is an abstraction that supports more flexible communication
  - May be supported in hardware and/or software

Design Issues for Group Communication

- *Addressing*
- *Reliability*
- *Ordering*
- *Delivery semantics*
- *Response semantics*
- *Group Structure*

Addressing

- Four methods of addressing:
  1. *Sender explicitly enumerates addresses*
  2. *Single group address*
  3. *Source addressing*
  4. *Functional addressing*
- The latter two mechanisms are difficult to implement in hardware

Reliability

- Reliability deals with recovery from communication failures
  - *e.g.*, buffer overflows and bit errors
- Much more difficult to implement reliability for group communication efficiently
- Thus, many group communication mechanisms are unreliable

Ordering

- Four ordering semantics:

1. *No ordering*
   - Easy to implement, but hard to use
2. *FIFO ordering*
   - Message from one member are delivered in order sent
3. *Causal ordering*
   - All messages that are related are ordered
4. *Total ordering*
   - Difficult to implement, but easy to use

Delivery Semantics

- Defines when a message is considered delivered successfully to a group
- Three choices

1. *k-delivery*
   - Succeeds when k participants have received message
2. *Quorum delivery*
   - Succeeds when a majority have received message
3. *Atomic delivery*
   - All must receive or none receive

Response Semantics

- Defines what a sender expects from the receivers
- Four choices

1. *No responses*
2. *A single response*
3. *Many responses*
4. *All responses*

- This is related to the reliability dimension...

Group Structure

- Two choices
  1. *Closed groups*
     - Only group members can send to the group
  2. *Open groups*
     - Non-members can also send to the group
- Two more choices
  1. *Static groups*
     - Members must join for the duration of the group
  2. *Dynamic groups*
     - Members can join and leave as they wish
- Note that support for *overlapping* groups is also available

IP Multicasting

- IP multicasting implements a simple model of group communication
- IP multicasting is used to send an IP or UDP datagram to a finite number of hosts using a single IP address
- IP multicasting is becoming available on modern operating systems
  - *e.g.*, Solaris and Windows NT

Advantages of IP Multicasting over Broadcasting

- Provides the ability to transmit information to a group of hosts via a single multicast address
- Multicasting is more efficient than broadcasting
  - It doesn't disturb hosts that are not participating in the communication
  - *i.e.*, it reduces extraneous packet examination in lower-level protocols...

IP Multicast Model

- *Addressing*
  - The IP address used for multicasting identifies a "multicast group"
    - ▹ A group may consist of 0 or more hosts
- *Reliability, Delivery, and Reponse Semantics*
  - IP multicasting is unreliable, delivery is on a best effort basis
- *Ordering*
  - IP multicasting provides no ordering guarantees

IP Multicast Model (cont'd)

- *Group Structure*
  - A single host may belong to many groups
    - ▷ A host may join or leave a group at any time during the lifetime of the group
  - A host need not be a member of a group to send a message to the group
  - Groups may overlap

IP Multicast Groups

- Two types of multicast groups

1. *Permanent*
   - Can have zero or more hosts with well-known IP address (*e.g.*, 224.0.0.1)
2. *Transient*
   - All multicast groups except permanent ones

- Note that groups are defined by the IP address
  - Note on membership to the group

IP Multicast Addressing

- Accomplished through the use of IP class D addresses
  - Four high order bits are set to 1110
  - Remaining 28 bits identify specific multicast groups
    - ▷ Ranges from 224.0.0.0 to 239.255.255.255
  - Address 224.0.0.0 is permanently unassigned
  - Address 224.0.0.1 is a permanent group to address all multicast hosts on a directly connected subnetwork

Mapping Multicast Addresses to Ethernet Addresses

- Multicast IP packets ultimately resolve down to Ethernet destinations
  - To create a unique Ethernet address the low order 23 bits of the IP address are mapped to the Ethernet address
  - Note the potential for conflict since there are 28 significant bits in a Class D IP address, but only the lower 23 are used...

Host Participation

- A host can be in one of three states when participating on a multicasting network:

  1. *Participate fully by belonging to a multicast group and sending/receiving multicast datagrams*
  2. *Be configured to send, but not receive, multicast datagrams*
  3. *Not participate in mulitcasting at all*

IP Multicast Administration (IGMP)

- The Internet Group Management Protocol (IGMP) handles administrative tasks related to hosts and gateways
  - IGMP resides upon the IP layer
  - Similar in spirit to ICMP, *i.e.*, sent in IP datagrams
- IGMP keeps hosts/gateways informed about status and configuration of multicast groups
  - Accomplished by continuously querying hosts and waiting for responses to be sent by one of the hosts in the group

Programming IP Multicast via Sockets

- Use `setsockopt` to join/leave a multicast group:

```
struct ip_mreq mcast_addr;

// Join group
setsockopt (sd, IPPROTO_IP, IP_ADD_MEMBERSHIP,
        (char *) &mcast_addr, sizeof mcast_addr);

// Leave group
setsockopt (sd, IPPROTO_IP, IP_DROP_MEMBERSHIP,
        (char *) &mcast_addr, sizeof mcast_addr);
```

- See the **ip**(7) manual page for more info on options

Example Header File

- Shared by both sender and receiver

```
// Multicast group address.
#define MCAST_ADDR "224.9.9.2"

// Port number.
#define UDP_PORT 3112

// Name of the interface.
#define INTERFACE "le0"
```

Example Receiver

- Server program

```
int main (void) {
  struct sockaddr_in local;
  struct ifreq interface_addr;
  struct inaddr in_addr;
  struct ip_mreq mcast_addr;
  struct sockaddr_in sa;
  int    sd = socket (PF_INET, SOCK_DGRAM, 0);

  // Determine interface address.
  strcpy (interface_addr.ifr_name, INTERFACE);
  ioctl (sd, SIOCGIADDR, &interface_addr);

  mcast_addr.imr_multiaddr.s_addr = inet_addr (MCAST_ADDR);
  sa = &interface_addr.ifr_addr;
  mcast_addr.imr_interface.s_addr = sa->sin_addr.s_addr;

  // Join group
  setsockopt (sd, IPPROTO_IP, IP_ADD_MEMBERSHIP,
              (char *) &mcast_addr, sizeof mcast_addr);

  // Initialize local port number.
  memset ((void *) &local, 0, sizeof local);
  local.sin_family = AF_INET; local.sin_port = htons (UDP_PORT);
  local.sin_addr.s_addr = htonl (INADDR_ANY);

  bind (sd, &local, sizeof local);
  // Go into loop on readfrom() and print what is received...
```


Example Sender

- Multicast data to receiver(s)

```
int main (int argc, char *argv[]) {
  struct sockaddr_in sa;
  struct hostent *hp;
  int sd = socket (AF_INET, SOCK_DGRAM, 0);
  sa.sin_family = AF_INET;
  sa.sin_port = htons (UDP_PORT);
  sa.sin_addr.s_addr = inet_addr (MCAST_ADDR);
  hp = gethostbyname (argv[1]);
  memcpy (&sa.sin_addr, hp->h_addr, hp->h_length);

  // Multicast data to receiver(s) via sendto();
```


UNIX Utilities

- `netstat`
  - View multicast groups via `netstat -g`
  - View multicast statistics via `netstat -s`
- `snoop`
  - View live packets as they arrive via `snoop multicast`
- See manual pages for options...

Summary

- Group communication is increasingly important
- IP multicast provides very low-level mechanisms to support group communication
- Building more sophisticated semantics on top of IP multicasting remains a research issue