Socket Options

- Various attributes that are used to determine the behavior of sockets.
- Setting options tells the OS/Protocol Stack the behavior we want.
- Support for generic options (apply to all sockets) and protocol specific options.

Getting option values

Gets the current value of a socket option:

```c
int getsockopt( int sockfd,
    int level,
    int optname,
    void *opval,
    socklen_t *optlen);
```

`level` specifies whether the option is a general option or a protocol specific option (what level of code should interpret the option) see P. 179 (Stevens).
Setting option values

Sets the current value of a socket option:

```c
int setsockopt(int sockfd,
               int level,
               int optname,
               const void *opval,
               socklen_t optlen);
```

- `level` specifies whether the option is a general option or a protocol specific option (what level of code should interpret the option).

General Options

- Protocol independent options.
- Handled by the generic socket system code.
- Some options are supported only by specific types of sockets (SOCK_DGRAM, SOCK_STREAM).
- This was just an overview
  - There are many details associated with the options described.
  - There are many options that haven’t been described.

Summary of Socket Options

- **Socket Level**
  - SO_SNDBUF, SO_RCVBUF, SO_KEEPALIVE, SO_BROADCAST, SO_REUSEADDR, SO_REUSEPORT
- **IP Level**
  - IP_TTL, IP_MULTICAST_IF, IP_MULTICAST_TTL, IP_MULTICAST_LOOP, IP_ADD_MEMBERSHIP, IP_DROP_MEMBERSHIP
- **TCP Level**
  - TCP_KEEPALIVE, TCP_MAXSEG, TCP_NODELAY
- See Stevens’ book P. 179
TCP socket options

- **TCP_KEEPALIVE**: set the idle time used when SO_KEEPALIVE is enabled.
- **TCP_MAXSEG**: set the maximum segment size sent by a TCP socket.
- **TCP_NODELAY**: can disable TCP’s Nagle algorithm that delays sending small packets if there is unACK’d data pending.

Multicasting Concept

- **Unicast**: a packet is received by a *single* interface
- **Broadcast**: a packet is received by *all* interfaces
- **Multicast**: a packet is received by multiple interfaces (receivers) using a *single* local “transmit” operation

<table>
<thead>
<tr>
<th>Unicast</th>
<th>Broadcast</th>
</tr>
</thead>
<tbody>
<tr>
<td>Send to one</td>
<td>Send to all</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Multicast</th>
</tr>
</thead>
<tbody>
<tr>
<td>Send to some</td>
</tr>
</tbody>
</table>

IP Multicast and the Mbone

- **IP Multicast group addressing**:
  - Group level (process): IP mcast address + port
  - IP level: multicast address (224.x.x.x)
  - Ethernet level: IP mcast address mapping
IP Multicast and the Mbone

- **Server-Oriented Multicast**
  - source sets up one-to-many multicast group
  - discourages dynamic groups
- **Receiver-Oriented Multicast (Deering, 1991)**
  - senders need not be members (called open group)
  - Unlimited group size
  - no topological restrictions on membership
  - membership dynamic and autonomous
  - host groups may be transient or permanent (e.g., 224.0.0.1)

IP Multicast

- **host-group model**
- **network-level; same packet format, different address**
- **routers do all of the work**
- **special IP addresses:**
  - 224.0.0.0 - 239.255.255.255
- **28 bits -> 268 million groups (plus scope)**
- **ttl value limits distribution**

### IP Multicast

<table>
<thead>
<tr>
<th>Class</th>
<th>NetID</th>
<th>HostID</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>110</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>1110</td>
<td><strong>Multicast Address</strong></td>
</tr>
</tbody>
</table>

8 bits  8 bits  8 bits  8 bits
IP Multicast in WAN

- IGMP and Multicast Routing Protocols
- Prune branches where no members and branches not on shortest paths

Join grp

Join grp

Mbone

- Mbone = multicast backbone
- virtual network overlaying Internet
- needed until mcast capable routers deployed
- IP in IP encapsulation

Mbone Protocols

- IP
- UDP: best effort
- RTP: real-time transport
- RSVP: resource reservation protocol
- SDP/SAP: session description, announcement protocols
Session Protocols

- Session Description Protocol (SDP)
  - used to describe the contents of mcast sessions
    - name, purpose, start time, duration
    - media (type, transport protocol, format)
    - how to receive media

- Session Announcement Protocol (SAP)
  - mcast protocol for SDP
  - periodic transmission to known mcast address
  - frequency depends on other announcements and scope

- Session Directory (SDR)
  - used to allocate multicast addresses to sessions
    - random allocation currently used
    - advertises multicast sessions uses SDP

Mbone Applications

- freeware
  - vic (video), vat (audio), wb (whiteboard), NTE (Text Editor)
- commercial
  - IP/TV - teleconferencing (Precept)

IP Multicast Data Structure

```
struct ip_mreq {
    struct in_addr imr_muliaddr /*class D mcast address */
    struct in_addr imr_interface /* addr of local interface */
}
// IP_ADD_MEMBERSHIP strcut ip_mreq
// Join mcast group
// IP_DROP_MEMBERSHIP strcut ip_mreq
// Leave mcast group
// IP_MULTICAST_IF strcut in_addr
// Interface for mcast out
// IP_MULTICAST_TTL u_char
// Set TTL for mcast out
// IP_MULTICAST_LOOP u_char
// On/off loopback
```
Multicast Client/Server Algorithm & Examples

- **Algorithm**
  - UDP client/server algorithm
  - Use `setsockopt()` to join before bind or send

- **Multicast Examples**
  - Simple SDR Example P. 505
  - Send/Receive Multicast Example P. 510
  - NTP Example P. 512
  - SUN Example
    - See [www.depaul.edu/~ealshaer/courses/tdc561](http://www.depaul.edu/~ealshaer/courses/tdc561)