UDP Socket Programming

Christian Grothoff

Berner Fachhochschule

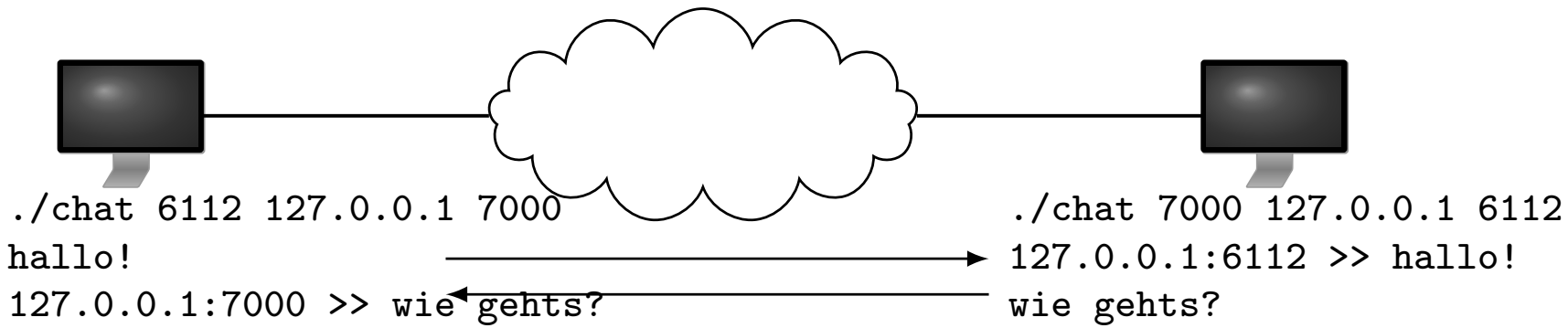
May 4, 2018

Today: UDP Socket Programming and `select()`

Learning objectives:

- More socket APIs: `recvfrom()`, `SO_BROADCAST`, `select()`
- Practice UDP

Programming objective: implement a *group chat* application.



Creating a socket

```
#include <sys/types.h>
#include <sys/socket.h>

int socket (int domain, int type, int protocol);
```

Use `AF_INET` or `AF_INET6` for *domain*.
Today, we will discuss the *type* being `SOCK_DGRAM`.
We need to set *protocol* to `IPPROTO_UDP` or 0.

Configuring the socket

Tell the kernel that we do want to enable broadcasts:

```
const int one = 1;
setsockopt (sock,
            SOL_SOCKET,
            SO_BROADCAST,
            (char *) &one,
             sizeof (one));
```

Bind to a port

```
struct sockaddr_in local;
local.sin_family      = AF_INET;
local.sin_port        = htons (LOCALPORT);
local.sin_addr.s_addr = INADDR_ANY;

bind (sock,
      (struct sockaddr *) &local,
      sizeof(local));
```

Waiting for data

How can we tell when data is available?

- Call `read()` on the socket
  - `read()` blocks until data is ready
  - Can only watch a single socket per process/thread,
  - Cannot even react to keyboard input
- Use traditional event loop like `select()`
  - Put all file descriptors to monitor into a set
  - Pass `select()` the set
  - Once something happens, `select()` returns a set with those FDs that are ready
- Use non-portable edge-triggered event loop like “epoll”

A select() loop

```
while(1) {
  fd_set rfds;

  FD_ZERO (&rfds);
  FD_SET (STDIN_FILENO, &rfds);
  FD_SET (sock, &rfds);
  maxfd = MAX(sock, STDIN_FILENO);
  select (maxfd+1, &rfd, NULL, NULL, NULL);
  if (FD_ISSET (sock, &rfds)) { ... };
  if (FD_ISSET (STDIN_FILENO, &rfds)) { ... };
}
```

Receiving data

```
char buf[65536];
struct sockaddr_storage from;
socklen_t slen = sizeof (from);

recvfrom (sock,
          buf,
          sizeof (buf),
          0,
          (struct sockaddr *) &from,
          &slen);
```

Transmitting data

```
struct sockaddr_in dest;

sendto (sock,
        msg,
        strlen (msg) + 1,
        0,
        (const struct sockaddr *) &dest,
        sizeof (dest));
```

UDP group chat

- Start with a 1:1 chat
- Pass IP address and port via command-line
- Use broadcast for group chats
- Optional: ensure transmission is in UTF-8

UDP multicast: sending

```
struct in_addr li;
struct sockaddr_in sa;
sa.sin_family = AF_INET;
sa.sin_addr.s_addr = inet_addr("226.42.62.42");
sa.sin_port = htons(PORT);
li.s_addr = inet_addr("192.168.0.52");
/* Specify interface to use for multicast */
setsockopt (sd, IPPROTO_IP, IP_MULTICAST_IF,
            (char *)&li, sizeof(li));
/* Transmit to multicast address */
sendto (sock, data, datalen, 0,
        (struct sockaddr*)&sa, sizeof(sa));
```

UDP multicast: receiving

```
struct ip_mreq group;
int reuse = 1;
group.imr_multiaddr.s_addr
   = inet_addr("226.42.62.42");
group.imr_interface.s_addr
   = inet_addr("192.168.0.52");
/* allow multiple applications to bind */
setsockopt (sd, SOL_SOCKET, SO_REUSEADDR,
            (char *)&reuse, sizeof(reuse));
bind (sd, ...); /* can use IP=0 */
/* join group */
setsockopt (sd, IPPROTO_IP, IP_ADD_MEMBERSHIP,
            (char *)&group, sizeof(group));
/* receive */
read (sd, databuf, datalen);
```