CS118 Discussion Week 2

Taqi

Outline

- Any Questions for Course Project 1?
- Socket Programming: Non-blocking mode
- Lecture Review: Application Layer

- Much of the other related stuff we will only discuss during the discussion.

Serve Multiple TCP Connections Simultaneously

- Problem: accept ( ) works under **blocking mode** by default
    - Unless a new connection request arrives, accept ( ) will not return
- Perquisite: listen ( ) allows multiple TCP connection
- Three approaches
    - fork ( ): each connection is served by a new process
        - Easy to write, but expensive and hard to share data between processes
    - POSIX pthread: each connection is served by a new thread
        - Hard to maintain
    - **Non-block mode**: use select ( )

What is select ( )?

- A monitor for multiple sockets (or file descriptors)
    - Given a set of sockets, if any of them were ready to receive/send, select ( ) would exit
- **int select (int numfds, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, struct timeval *timeout);**
    - **numfds**: the highest file descriptor plus one
    - **readfds**, **writefds**, **exceptfds**: set of sockets
    - **timeout**: timer for select to exit without any changes
    - return when some sockets are ready, or timeout
    - return value: the number of sockets that are ready

What is fd_set?

- A set of sockets (or file descriptors) that will be monitored by select ( )
- Macros of set operation
  - FD_SET(int fd, fd_set *set); //add fd to the set
  - FD_CLR(int fd, fd_set *set); // remove fd from the set
  - FD_ISSET(int fd, fd_set *set); // return if fd is in the set
  - FD_ZERO(fd_set *set); // clear all entries in the set

How to use select ( )?

- Assume the server has created a socket ***sock***, which is bound with server's IP address and port number

```
fd_set active_fd_set;     //set for holding sockets

int new_sock;         //socket representing client

/* Initialize the set of active sockets */

FD_ZERO (&active_fd_set);

FD_SET (sock, &active_fd_set);   /*put sock to the set, s.t. we can monitor whether a new connection request arrives. This means you have to do accept(), etc.*/
```

How to use select ( )?

```
while (1){

/* Block until some sockets are active. Let N is #sockets+1 in active_fd_set*/

      if(select (sock+1, &active_fd_set, NULL, NULL, NULL)<0) {exit(-1);} /*errors*/

/* Now some sockets are active */

    if(FD_ISSET(sock, &active_fd_set)) //new connection request

    {

          new_sock = accept (sock, (struct sockaddr*)&client_addr, sizeof(client_addr);

          FD_SET (new_sock, &active_fd_set);

    }
```

How to use select ( )?

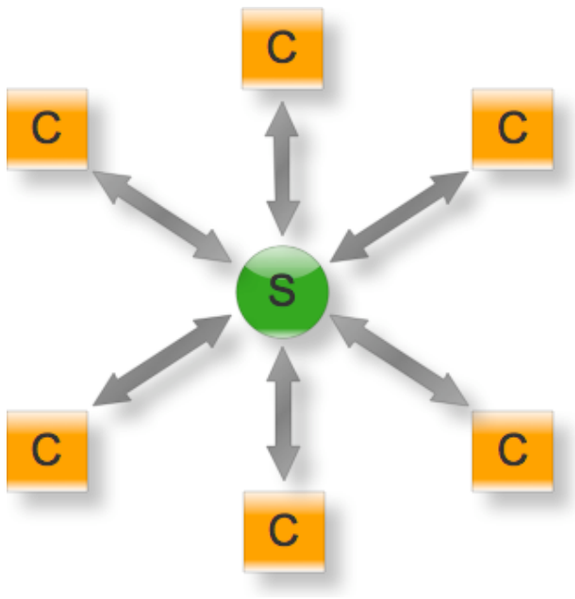
```
/* Decide whether client has sent data */

if (FD_ISSET(new_sock, &active_fd_set))

{
        /*receive and process data here*/

}

}//end of while
```

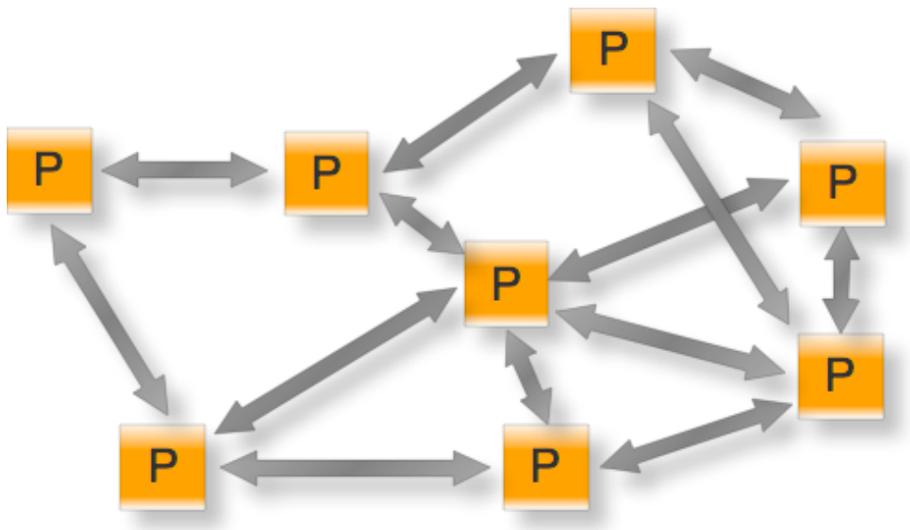
How to use select ( )?

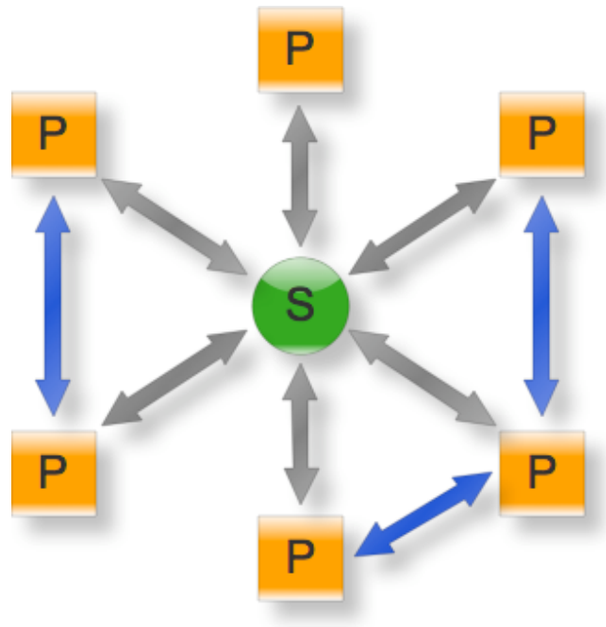
```
int socket_fd, result;
fd_set readset;
...
/* Socket has been created and connected to the other party */
...

/* Call select() */
do {
   FD_ZERO(&readset);
   FD_SET(socket_fd, &readset);
   result = select(socket_fd + 1, &readset, NULL, NULL, NULL);
} while (result == -1 && errno == EINTR);

if (result > 0) {
   if (FD_ISSET(socket_fd, &readset)) {
      /* The socket_fd has data available to be read */
      result = recv(socket_fd, some_buffer, some_length, 0);
      if (result == 0) {
         /* This means the other side closed the socket */
         close(socket_fd);
      }
```

Application Layer

- Application Architectures
  - Client-server model: Web (TCP), FTP (TCP), E-mail (TCP), DNS (TCP), RTP
  - Peer-to-Peer (P2P): BitTorrent (TCP)
  - Hybrid model: Skype (TCP&UDP), GTalk (TCP&UDP)







Application Layer

- HTTP: a stateless protocol on top of TCP
    - HTTP is based on pull model
    - Persistent HTTP V.S. Non-persistent HTTP
        - How many TCP connections do we need to get one HTML-file with 5 embedded images? How many RTTs shall we need?
- Method Types: GET, HEAD, POST, PUT, DELETE, Conditional GET
- What if we want stateful service (e.g. online banking)? **Cookie**
- Web Caches (proxy server)

Application Layer

- FTP: separate control from data transmission
- SMTP: protocol for email exchange between email servers
    - SMTP is based on push model
    - Mail access protocol: POP, IMAP, HTTP between client and server
- P2P: no always-on server, peers are intermittently connected
    - BitTorrent: tracker and torrent. Files are divided into multiple chunks.

Application Layer

- DNS: convert name (IP address) to IP address (name)
    - Scalability via distribution and caching
    - Who will use Iterative/recursive query?
    - Why is DNS resolver needed?

Exercise

- Assume the cache is empty initially
- Host A queries www.ucla.edu, how many queries should resolver issue?
- After A's DNS query, host B queries www.mit.edu, how many queries should resolver issue?