Proxy & Networking

15-213: Introduction to Computer Systems – Recitation H
April 11, 2011
Today

- News
- Proxy
- Networking
News

- Proxylab out last Thursday
- Exam in two weeks
Proxy
What is a Proxy?

- Intermediary between client and server
Your Proxy

- Threaded
  - Client requests must use different threads

- Caching
  - Content is cached locally and served from there

- IPv4
  - Only 32-bit addresses: 192.168.99.211

- HTTP
  - Only serve HTTP/1.0 GET requests
Your Proxy

- Both client and server

Client socket address: 128.2.194.242:51213

Server socket address: 208.216.181.15:80

Client host address: 128.2.194.242

Server host address: 208.216.181.15
Recommended Progress

- Implement a sequential proxy
- Upgrade to a threaded proxy
- Make into a caching proxy
  - Worry about race conditions and proper locking of data structures! (next recitation)
Testing

- No autograder
- Test simple pages at the beginning and more complicated ones as your proxy improves
- Not all pages will work!
- Useful: netcat
  - Client: `nc <host> <port>`
  - Server `nc -l <port>`
Networking
Server

- Berkeley Sockets Interface
- For making a server:
  - socket
  - bind
  - listen
  - accept
  - close
socket(domain, type, protocol)

```c
int sock_fd = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
```

- **Through which all communication goes through**
- **Domain:** Protocol family to use
  - PF_INET is IPv4
- **Type:** Type of protocol to use
  - SOCK_STREAM stream with in-order delivery
- **Protocol:** Specific protocol to use
  - IPPROTO_TCP means TCP
bind(sock_fd, my_addr, addrlen)

struct sockaddr_in sockaddr;
memset(&sockaddr, 0, sizeof(sockaddr));
sockaddr.sin_family = AF_INET;
sockaddr.sin_addr.s_addr = INADDR_ANY;
sockaddr.sin_port = htons(listenPort)
err = bind(sock_fd, (struct sockaddr *) sockaddr, sizeof(sockaddr));

- **sock_fd**: file descriptor of socket
- **my_addr**: address to bind to (where we are registering this server)
  - Information about what the socket will be used for: internet, listening any address, port
- **addrlen**: size of addr struct
listen(sock_fd, backlog)

err = listen(sock_fd, MAX_WAITING_CONNECTIONS);

- **sock_fd**: Socket to listen on
- **backlog**: maximum number of waiting connections
accept(sock_fd, addr, addrlen)

```c
struct sockaddr_in client_addr;
socklen_t my_addr_len = sizeof(client_addr);
client_fd = accept(listener_fd, &client_addr, &my_addr_len);
```

- **sock_fd**: Socket to listen on
- **addr**: pointer to sockaddr struct to hold client information
  - Find out who connected to you
- **addrlen**: length of the addr struct
- **Return**: File descriptor!
Client

- For connecting to a server:
  - socket
  - connect
  - close
connect(sock_fd, addr, addrlen)

```c
struct sockaddr_in remote_addr;
/* initialize remote_addr */
err = connect(listener_fd, &remote_addr, sizeof(remote_addr));
```

- **sock_fd**: Socket to connect with
- **addr**: pointer to sockaddr struct that holds information about who to connect to
- **addrlen**: length of the addr struct
close(sock_fd)

err = close(sock_fd);

- sock_fd: Socket to close
Sending/Receiving Data

- `send/recv`
- `write/read`

- `func(fd, buffer, buffer_len)`
  - `fd`: File to write to
  - `buffer`: What to write
  - `buffer_len`: How many bytes to write

- `send/recv` have extra flags parameter
Questions?