

15213 Recitation Section C

Shimin Chen
Nov. 25, 2002

Outline

- Extending echo server
- HTTP
- Broken pipe error
- Feedback and evaluation

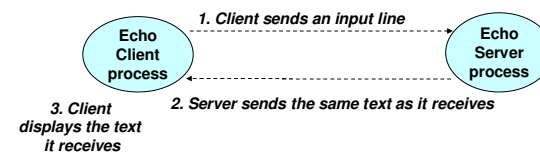
Important Dates

- Lab 7 *Proxy*: due on Thursday, Dec 5
- Final Exam: Tuesday, Dec 17

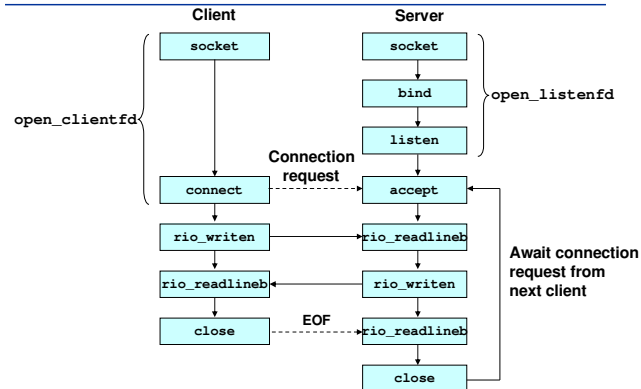
Extending Echo Server

- We will recap the echo client and server taught in the last lecture
- Then we will extend the echo server to build an echo proxy

An Echo Client-Server Transaction



Review of the Sockets Interface



15213 Recitation C

5

Shimin Chen

Echo Client Main Routine

```

#include "csapp.h"

/* usage: ./echoclient host port */
int main(int argc, char **argv)
{
    int clientfd, port;
    char *host, buf[MAXLINE];
    rio_t rio;

    host = argv[1];
    port = atoi(argv[2]);

    clientfd = Open_clientfd(host, port);
    Rio_readinitb(&rio, clientfd);

    while (Fgets(buf, MAXLINE, stdin) != NULL) {
        Rio_writen(clientfd, buf, strlen(buf));
        Rio_readlineb(&rio, buf, MAXLINE);
        Fputs(buf, stdout);
    }

    Close(clientfd);
    exit(0);
}
    
```

Echo Client: open_clientfd

```

int open_clientfd(char *hostname, int port)
{
    int clientfd;
    struct hostent *hp;
    struct sockaddr_in serveraddr;

    if ((clientfd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
        return -1; /* check errno for cause of error */

    /* Fill in the server's IP address and port */
    if ((hp = gethostbyname(hostname)) == NULL)
        return -2; /* check h_errno for cause of error */
    bzero((char *) &serveraddr, sizeof(serveraddr));
    serveraddr.sin_family = AF_INET;
    bcopy((char *) hp->h_addr,
          (char *) &serveraddr.sin_addr.s_addr, hp->h_length);
    serveraddr.sin_port = htons(port);

    /* Establish a connection with the server */
    if (connect(clientfd, (SA *) &serveraddr, sizeof(serveraddr)) < 0)
        return -1;
    return clientfd;
}
    
```

This function opens a connection from the client to the server at hostname:port

Echo Server: Main Routine

```

int main(int argc, char **argv) {
    int listenfd, connfd, port, clientlen;
    struct sockaddr_in clientaddr;
    struct hostent *hp;
    char *haddrp;

    port = atoi(argv[1]); /* the server listens on a port passed
                           on the command line */
    listenfd = open_listenfd(port);

    while (1) {
        clientlen = sizeof(clientaddr);
        connfd = Accept(listenfd, (SA *) &clientaddr, &clientlen);
        hp = Gethostbyaddr((const char *) &clientaddr.sin_addr.s_addr,
                          sizeof(clientaddr.sin_addr.s_addr), AF_INET);
        haddrp = inet_ntoa(clientaddr.sin_addr);
        printf("server connected to %s (%s)\n", hp->h_name, haddrp);
        echo(connfd);
        Close(connfd);
    }
}
    
```

Echo Server: open_listenfd

```
int open_listenfd(int port)
{
    int listenfd, optval=1;
    struct sockaddr_in serveraddr;

    /* Create a socket descriptor */
    if ((listenfd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
        return -1;

    /* Eliminates "Address already in use" error from bind. */
    if (setsockopt(listenfd, SOL_SOCKET, SO_REUSEADDR,
                  (const void *)&optval , sizeof(int)) < 0)
        return -1;

    ... (more)
}
```

Echo Server: open_listenfd (cont)

```
...

/* Listenfd will be an endpoint for all requests to port
on any IP address for this host */
bzero((char *) &serveraddr, sizeof(serveraddr));
serveraddr.sin_family = AF_INET;
serveraddr.sin_addr.s_addr = htonl(INADDR_ANY);
serveraddr.sin_port = htons((unsigned short)port);
if (bind(listenfd, (SA *)&serveraddr, sizeof(serveraddr)) < 0)
    return -1;

/* Make it a listening socket ready to accept
connection requests */
if (listen(listenfd, LISTENQ) < 0)
    return -1;

return listenfd;
}
```

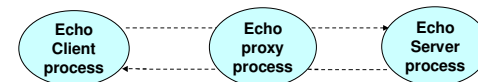
Echo Server: echo

```
void echo(int connfd)
{
    size_t n;
    char buf[MAXLINE];
    rio_t rio;

    Rio_readinitb(&rio, connfd);
    while((n = Rio_readlineb(&rio, buf, MAXLINE)) != 0) {
        printf("server received %d bytes\n", n);
        Rio_writen(connfd, buf, n);
    }
}
```

Proxy

- A *proxy* is an intermediary between a *client* and an *origin server*.
 - To the client, the proxy acts like a server.
 - To the server, the proxy acts like a client.



Changing "echo" procedure

```
void echo_forward(int connfd, char *server, int server_port)
{
    int forwardfd;
    rio_t rio_conn, rio_forward;
    ssize_t n;
    char buf[MAXLINE];

    /* connect to the server */
    forwardfd = Open_clientfd(server, server_port);
    Rio_readinith(&rio_forward, forwardfd);

    Rio_readinith(&rio_conn, connfd);
    while (1) {
        if ((n = Rio_readlineb(&rio_conn, buf, MAXLINE)) == 0)
            break;
        Rio_writen(forwardfd, buf, n);

        if ((n = Rio_readlineb(&rio_forward, buf, MAXLINE)) == 0)
            break;
        Rio_writen(connfd, buf, n);
    }
    Close(forwardfd);
}
```

15213 Recitation C

13

Shimin Chen

Changing main

```
int main(int argc, char **argv)
{
    int listenfd, connfd, port, clientlen, server_port;
    struct sockaddr_in clientaddr;
    struct hostent *hp;
    char *haddrp, *server;

    port = atoi(argv[1]);
    server = argv[2];
    server_port = atoi(argv[3]);

    listenfd = Open_listenfd(port);
    while (1) {
        clientlen = sizeof(clientaddr);
        connfd = Accept(listenfd, (SA *)&clientaddr, &clientlen);
        . . .

        echo_forward(connfd, server, server_port);
        Close(connfd);
    }
    exit(0);
}
```

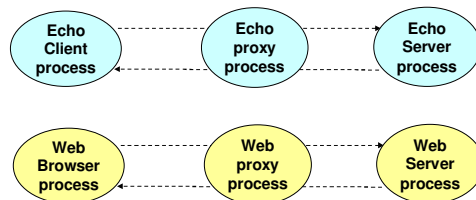
15213 Recitation C

14

Shimin Chen

L7 Proxy

- Different request and response (HTTP)
- Concurrency



15213 Recitation C

15

Shimin Chen

HTTP Request and Response

- Let's use telnet to examine a simple HTTP request and response
- More details in Tuesday's lecture

15213 Recitation C

16

Shimin Chen

HTTP Request and Response

```
[chensm@bass ~]$ telnet www.cmu.edu 80
Trying 128.2.11.43...
Connected to WEB3.andrew.cmu.edu.
Escape character is '^]'.
```

```
GET / HTTP/1.1
host: www.cmu.edu
```

```
HTTP/1.1 200 OK
Date: Sun, 24 Nov 2002 21:52:53 GMT
Server: Apache/1.3.26 (Unix) PHP/4.2.2 mod_pubcookie/a5/1.76-009
      mod_ssl/2.8.10 OpenSSL/0.9.6d
Last-Modified: Thu, 21 Nov 2002 15:28:47 GMT
ETag: "d3226-38b1-3ddcfbaf"
Accept-Ranges: bytes
Content-Length: 14513
Content-Type: text/html
```

```
<HTML>
  <HEAD>
    <TITLE> Carnegie Mellon University </TITLE>
  </HEAD>
  . . . . .
```

HTTP Request

Request line: <method> <uri> <version>

```
GET / HTTP/1.1
host: www.cmu.edu
```

Request headers: <header name>: <header data>

'\r\n' to mark the end of the request

HTTP Response

Response line: <version> <status code> <status msg>

Response headers: <header name>: <header data>

```
HTTP/1.1 200 OK
Date: Sun, 24 Nov 2002 21:52:53 GMT
Server: Apache/1.3.26 (Unix) PHP/4.2.2 mod_pubcookie/a5/1.76-009
      mod_ssl/2.8.10 OpenSSL/0.9.6d
Last-Modified: Thu, 21 Nov 2002 15:28:47 GMT
ETag: "d3226-38b1-3ddcfbaf"
Accept-Ranges: bytes
Content-Length: 14513
Content-Type: text/html
```

```
<HTML>
  <HEAD>
    <TITLE> Carnegie Mellon University </TITLE>
  </HEAD>
  . . . . .
```

Broken Pipe Error

- When reading or writing to a socket
- If the socket has already been closed at the other end
 - e.g. click “stop” on web browser
- SIGPIPE signal and EPIPE errno
- Don't want to terminate the web server

Details

- Write to a broken pipe
 - For the first write, return -1 and set EPIPE
 - For subsequent writes,
 - Send SIGPIPE signal, which terminates process
 - If the signal is blocked or handled, return -1 and set EPIPE.
- Example of broken pipe with echo server

How to deal with broken pipe?

- Block SIGPIPE signal
- Ignore EPIPE error in Rio wrappers of csapp.c
- Example of how we handle broken pipe in echo server
 - In server main(), we block SIGPIPE signal
 - In csapp.c, we ignore EPIPE error in Rio wrappers
 - In echo(), deal with cases when return value is -1

Feedback and Evaluation

Thank you!