CS4700/CS5700
Fundamentals of Computer Networking

Prof. Alan Mislove

Lecture 3: Crash course in socket programming

September 10th, 2009

Northeastern

Project 0

Goal: Familiarize you with socket programming in C

Implement a trivial/useless protocol

We have a server; you write the client

If you know C, it will take 5 minutes

If you don't, it will take 30

`client.c` stub code

We provide stub code

- Does all necessary incantations

You fill in the interesting bits

- Should be 5-10 lines

Rest of lecture: code walk

- Explain the (interesting) incantations

Includes

```
(1) #include <stdio.h>
(2) #include <string.h>   #include <unistd.h>
(3) #include <sys/types.h>
(4) #include <stdlib.h>
(5) #include <sys/socket.h>
(6) #include <netinet/in.h>
(7) #include <arpa/inet.h>
(8) #include <netdb.h>

(10)#include "common.h"
```

#defines

```
(11)#define pAssert(a,b) {
      if (!(a)) {
        printf("Line %d in File %s: %s\n",
                __LINE__, __FILE__, b);
        exit(1);
      }
    }
(12)#define pError(e,a) {
      if (e) {
        perror(a); exit(1);
      }
 }
```

main

```
(13)  struct sockaddr_in sin;
(14)  struct hostent *h;
(15)  int sd;
```

```
(17)  h = (argc > 2) ?
              gethostbyname(argv[1]) :
              gethostbyname(SERVER_HOSTNAME);

(19)  pAssert(h, "gethostbyname failed\n");
```

allocating sockets

```
(22)  // sd = socket(...);
```

Getting help on UNIX - man

```
prompt%  man 2 socket

SOCKET(2)                  BSD System Calls Manual                  SOCKET(2)

NAME
     socket -- create an endpoint for communication

SYNOPSIS
     #include <sys/socket.h>

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

DESCRIPTION
     Socket() creates an endpoint for communication and returns a descriptor.

     The domain parameter specifies a communications domain within which communication will take place; this
     selects the protocol family which should be used.  These families are defined in the include file
     <sys/socket.h>.  The currently understood formats are

          AF_UNIX         (UNIX internal protocols),
          AF_INET         (ARPA Internet protocols),
          AF_ISO          (ISO protocols),
```

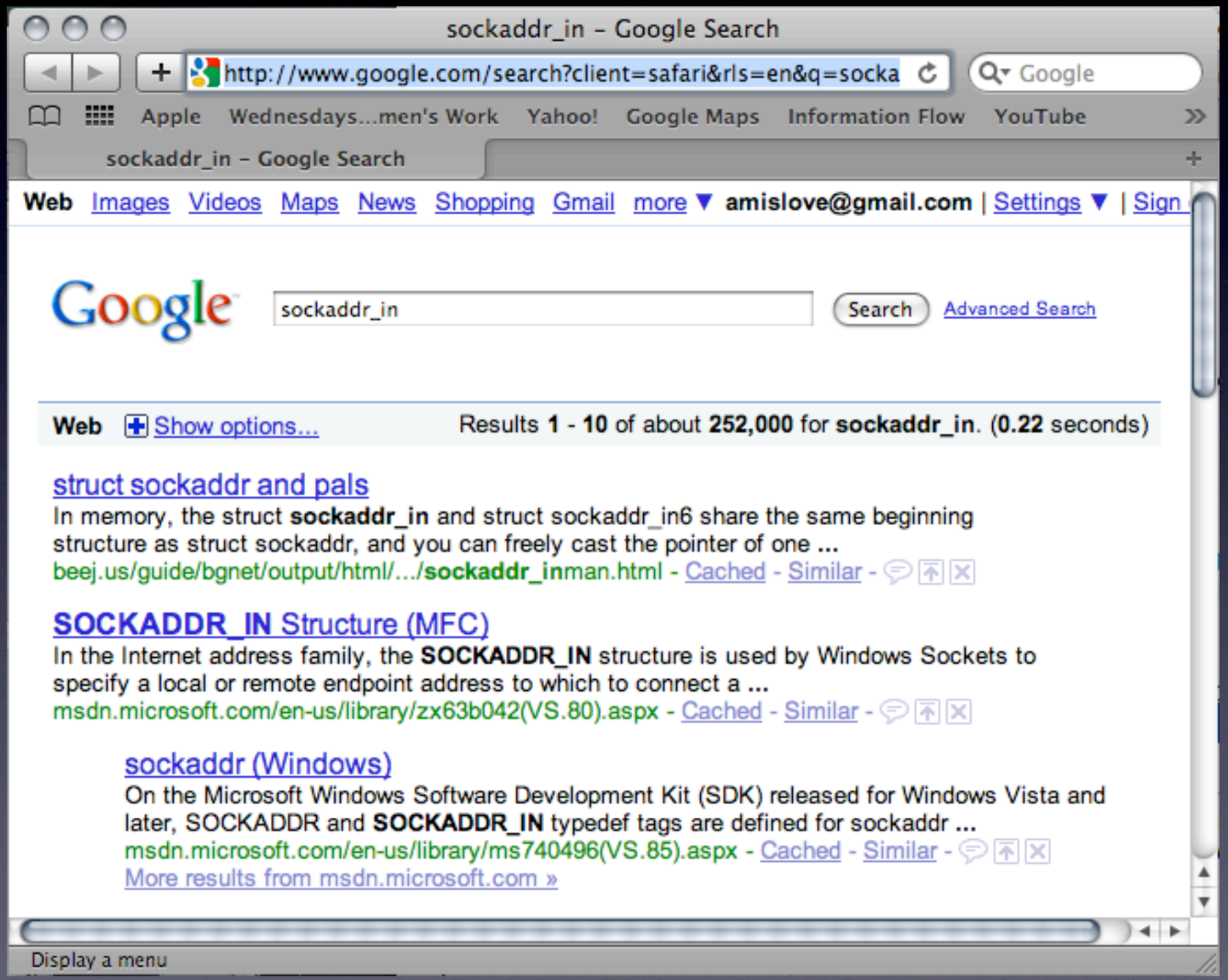
back to allocating sockets

```
(22)  // sd = socket(...);
(23)  pError((sd==-1), "socket");

(25)  /* Fill in '<struct sockaddr_in sin>' */
(26)  // bzero((char *) &sin, sizeof(sin));
(27)  // sin.sin_family =
(28)  // sin.sin_addr.s_addr =
(29)  // sin.sin_port =

(31)  /* Connect to the server */
```

Getting help on UNIX - Google



connect()ing

```
prompt%  man 2 connect

CONNECT(2)                  BSD System Calls Manual                 CONNECT(2)

NAME
     connect -- initiate a connection on a socket

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

     int
     connect(int socket, const struct sockaddr *address,
         socklen_t address_len);

DESCRIPTION
     The parameter socket is a socket.  If it is of type SOCK_DGRAM, this call
     specifies the peer with which the socket is to be associated; this
     address is that to which datagrams are to be sent, and the only address
     from which datagrams are to be received.  If the socket is of type
     SOCK_STREAM, this call attempts to make a connection to another socket.
     The other socket is specified by address, which is an address in the com-
     munications space of the socket.
```

reading and writing data

```
(32)  /* Format <client_msg> with HELLO string */
(33)  sprintf(client_msg, "%s HELLO %s\n", magic_str,
                argv[argc-1]);
```

```
(35)  // send client_msg to server

(37)  /* Wait for server's STATUS message, read it
         into server_msg1 */
(38)  token = strtok(server_msg1, delims);
(39)  pAssert(token && !strcmp(token, magic_str),
                "Invalid Magic String");
```

send()ing

```
prompt%  man 2 send

SEND(2)                    BSD System Calls Manual                    SEND(2)

NAME
     send, sendmsg, sendto -- send a message from a socket

SYNOPSIS
     #include <sys/socket.h>

     ssize_t
     send(int socket, const void *buffer, size_t length, int flags);

     ssize_t
     sendmsg(int socket, const struct msghdr *buffer, int flags);

     ssize_t
     sendto(int socket, const void *buffer, size_t length, int flags,
         const struct sockaddr *dest_addr, socklen_t dest_len);

DESCRIPTION
     Send(), sendto(), and sendmsg() are used to transmit a message to another
     socket.  Send() may be used only when the socket is in a connected state,
     while sendto() and sendmsg() may be used at any time.
```

recv()ing

```
prompt%  man 2 send

RECV(2)                    BSD System Calls Manual                    RECV(2)

NAME
     recv, recvfrom, recvmsg -- receive a message from a socket

LIBRARY
     Standard C Library (libc, -lc)

SYNOPSIS
     #include <sys/socket.h>

     ssize_t
     recv(int socket, void *buffer, size_t length, int flags);

     ssize_t
     recvfrom(int socket, void *restrict buffer, size_t length, int flags,
         struct sockaddr *restrict address, socklen_t *restrict address_len);

     ssize_t
     recvmsg(int socket, struct msghdr *message, int flags);

DESCRIPTION
```

writing servers

Client specifies who to talk to

- Called an active open

Servers are written differently

- Designed to wait until a client connects

```
(1) bind()
(2) listen()
(3) accept()
```