CSE 489/589 Programming Assignment 1

Li Sun, Swetank Kumar Saha
{lsun3, swetankk}@buffalo.edu

CSE 489/589: Modern Networking Concepts

- **Instructor: Dimitrios Koutsonikolas**
  - Office: 311 Davis Hall
  - Email: dimitrio [at] buffalo.edu
  - Office Hours
    - Tuesday 5:00-6:00 PM
    - Thursday 5:00-6:00 PM
    - By Appointment
- Course website: http://www.cse.buffalo.edu/faculty/dimitrio/courses/cse4589_s16/index.html
- Piazza: https://piazza.com/buffalo/spring2016/cse4589/home

TA office hours

- **Li Sun**
  Office: 302 Davis Hall/300 Davis Hall Student Lounge
  Office Hours: Monday 5:00 - 6:00 PM
  Email: lsun3 [at] buffalo.edu

- **Swetank Kumar Saha**
  Office: 302 Davis Hall/300 Davis Hall Student Lounge
  Office Hours: Thursday 4:00-5:00 PM
  Email: swetankk [at] buffalo.edu

PA1 Deadline

- **Due Date : 02/26/2016 @ 23:59:59 EST**

- **Start early!**

Outline

- Introduction to Socket Programming
  - Protocol Stack
  - TCP Overview
  - About Sockets
  - TCP Socket Overview

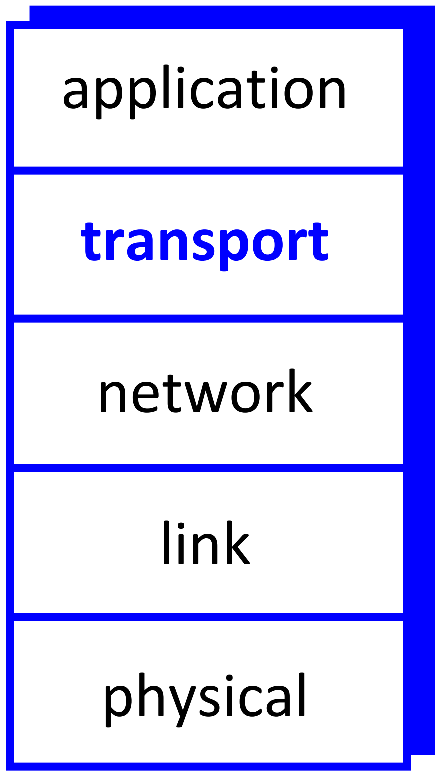
- Introduction to Programming Assignment 1
  - Project Objective, Description, and Requirements
  - PA1 Template
  - Tips and useful links

Part 1:

Introduction to Socket Programming

Protocol Stack

- *Application layer*: Supporting network applications
  - HTTP, SMTP
- **Transport layer**: Process-process data transfer
  - **TCP**, UDP
- *Network layer*: Routing of datagrams from source to destination
  - IP, routing protocols
- *Link layer*: Data transfer between neighboring network elements
  - Ethernet, MAC addresses
- *Physical layer*: Bits “on the wire”



TCP Overview

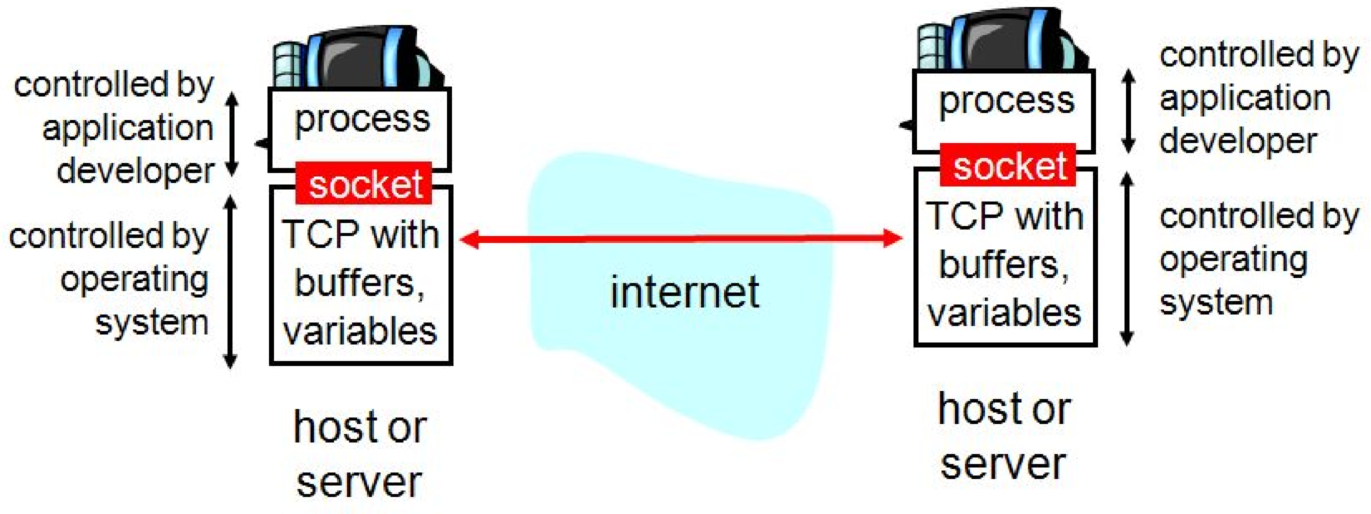
- Establish connection
  - 3-way handshake

- Data transmission
  - Reliable (retransmission with timer)
  - In-order delivery (reorder packets if necessary)
  - Support flow control (fast sender vs. slow receiver)
  - Full-duplex (data transferred both ways)

- Close connection

Sockets

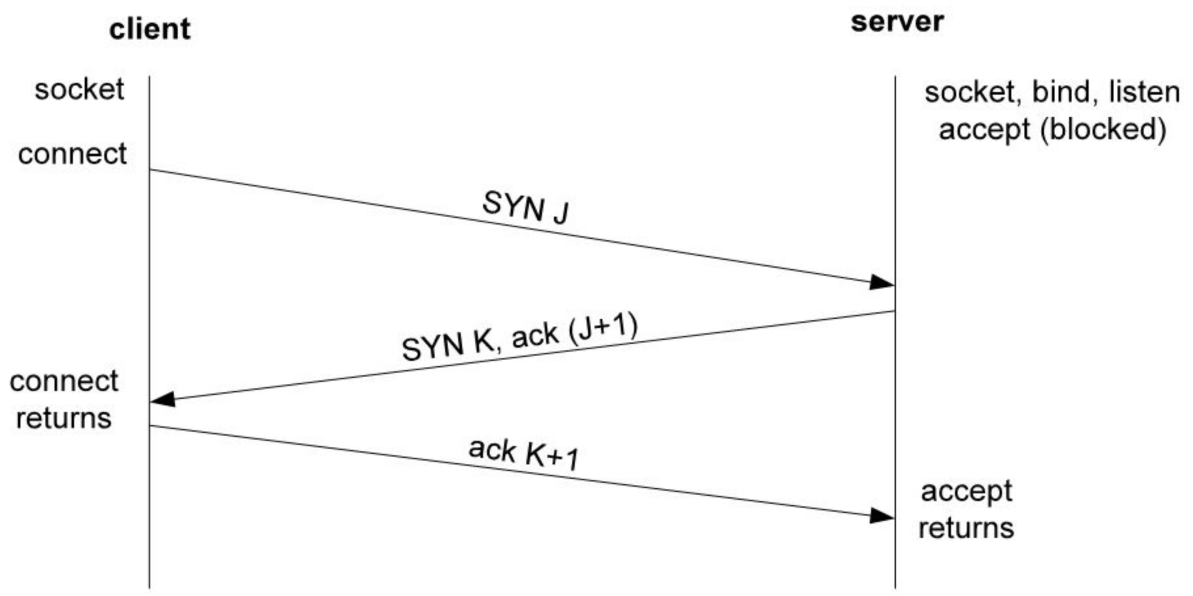
- A door between application process and end-end-transport protocol (TCP or UDP)


controlled by application developer
controlled by operating system
process
socket
TCP with buffers, variables
host or server
internet
process
socket
TCP with buffers, variables
host or server
controlled by application developer
controlled by operating system

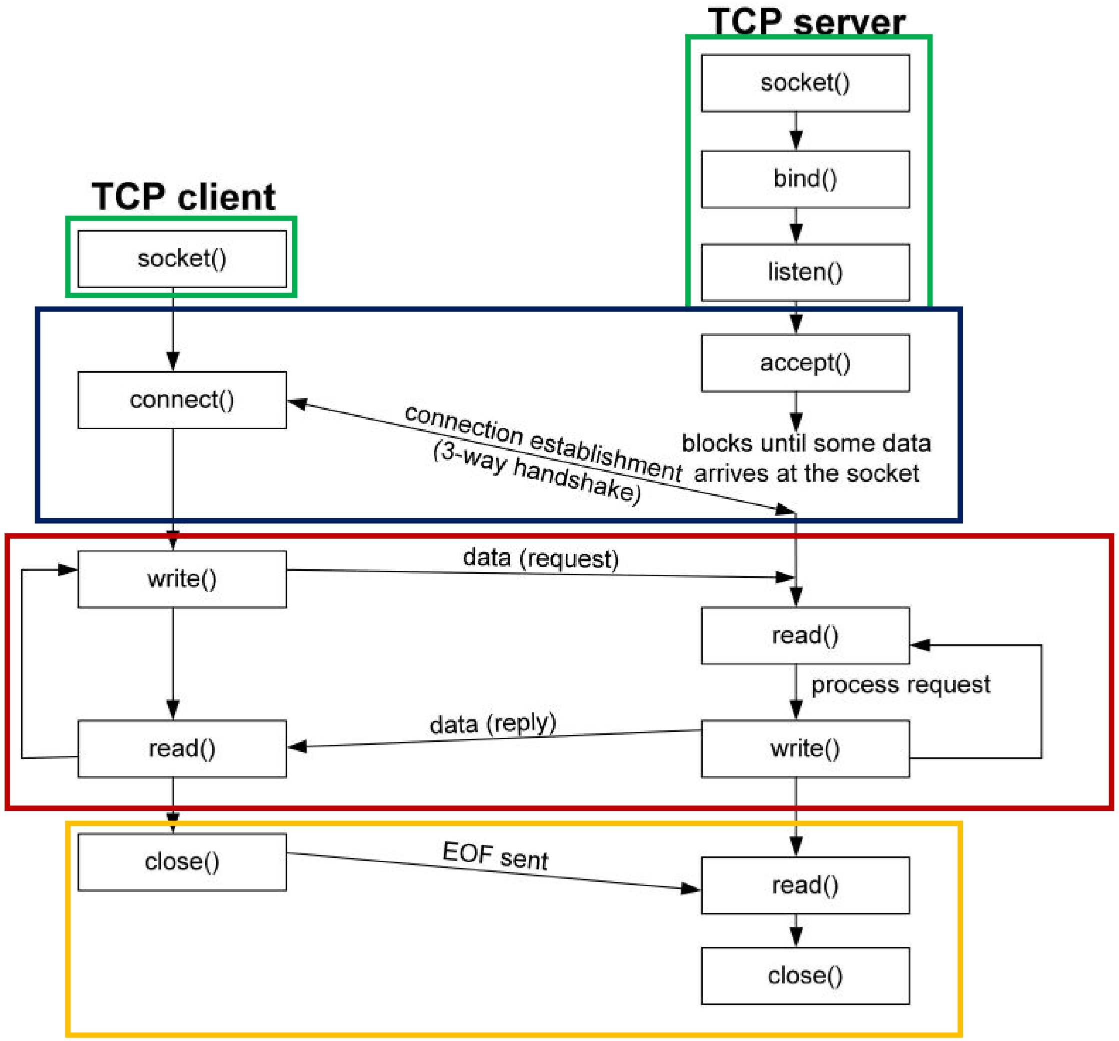
IP Address + Port Number

TCP Socket

- TCP Connection Establishment
  - Server gets ready (socket, bind, listen)
  - Client gets ready (socket)
  - Client requests connection (connect)



TCP Socket

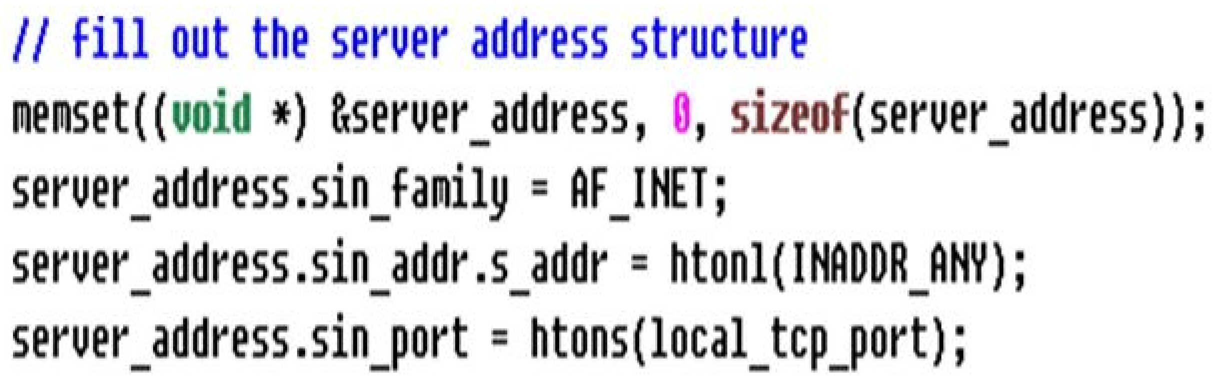

TCP server
socket()
bind()
listen()
accept()
blocks until some data arrives at the socket
TCP client
socket()
connect()
connection establishment (3-way handshake)
write()
data (request)
read()
process request
data (reply)
read()
write()
close()
EOF sent
read()
close()

TCP Socket

- Socket structure
    - You need to fill all the values listed in the structure

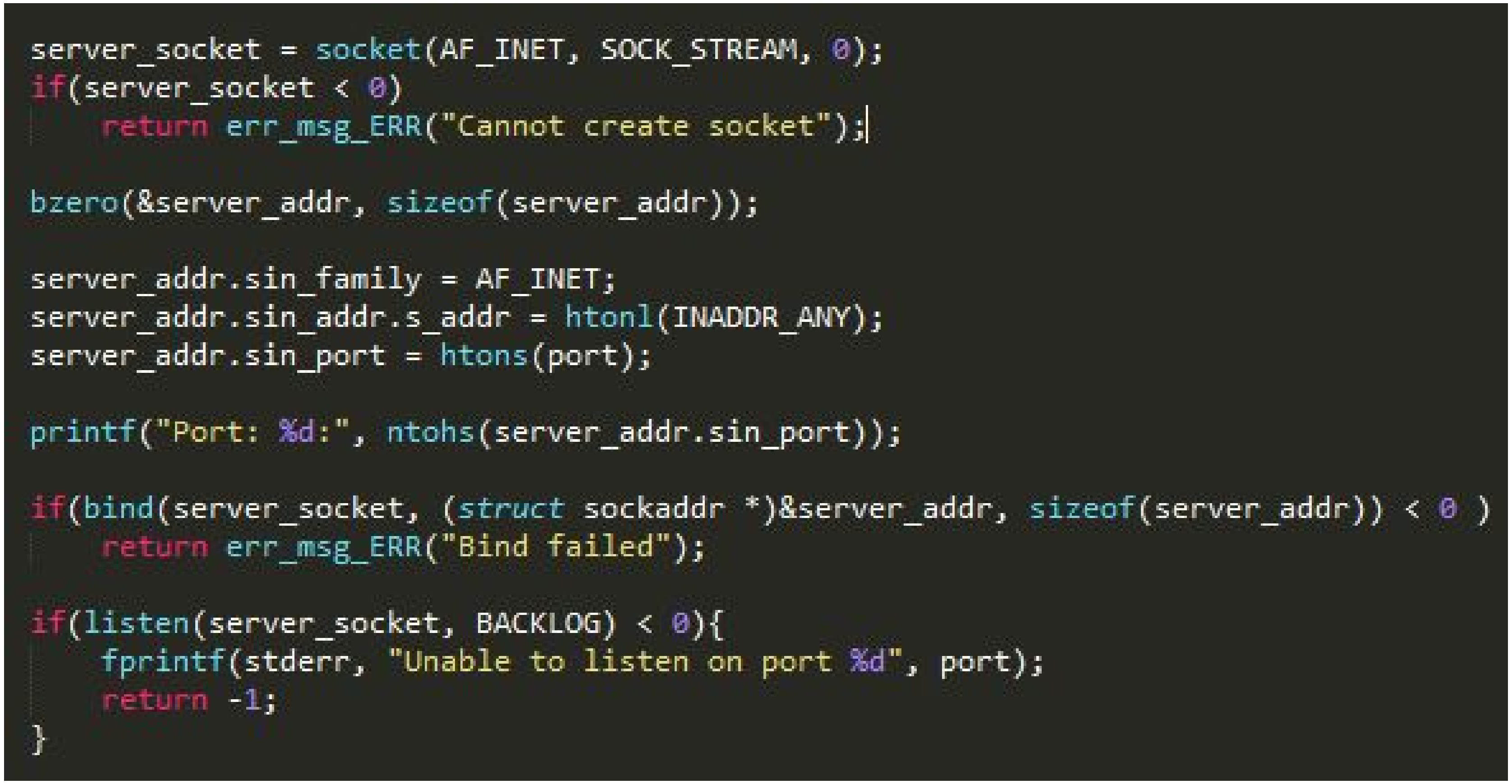
```
// IPv4 only
struct sockaddr_in {
     short int sin_family;          // Address family, AF_INET
     unsigned short int sin_port; // Port number
     struct in_addr sin_addr;       // Internet address
     unsigned char sin_zero[8];     // sizeof(sockaddr) and all 0s
};
```

TCP Socket



```
// fill out the server address structure
memset((void *) &server_address, 0, sizeof(server_address));
server_address.sin_family = AF_INET;
server_address.sin_addr.s_addr = htonl(INADDR_ANY);
server_address.sin_port = htons(local_tcp_port);
```

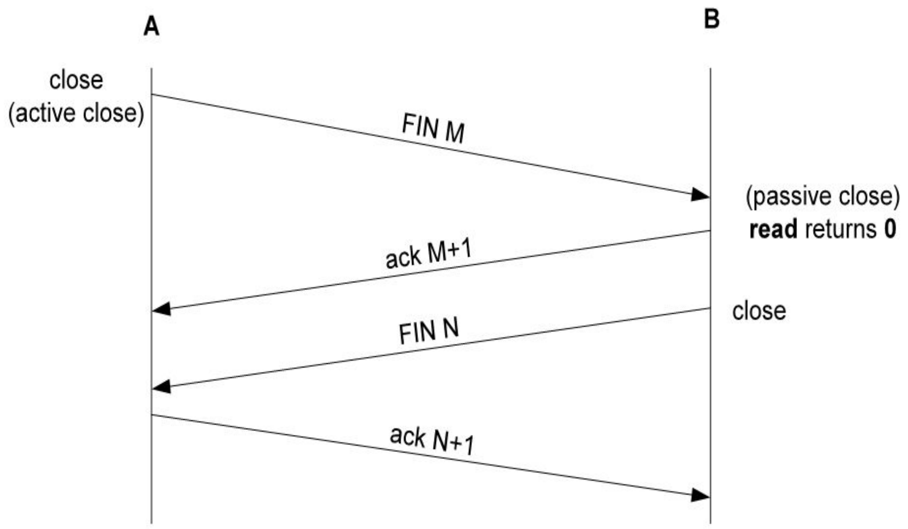
TCP Socket



```c
server_socket = socket(AF_INET, SOCK_STREAM, 0);
if(server_socket < 0)
    return err_msg_ERR("Cannot create socket");

bzero(&server_addr, sizeof(server_addr));

server_addr.sin_family = AF_INET;
server_addr.sin_addr.s_addr = htonl(INADDR_ANY);
server_addr.sin_port = htons(port);

printf("Port: %d:", ntohs(server_addr.sin_port));

if(bind(server_socket, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0 )
    return err_msg_ERR("Bind failed");

if(listen(server_socket, BACKLOG) < 0){
    fprintf(stderr, "Unable to listen on port %d", port);
    return -1;
}
```

TCP Socket

- TCP Connection Termination
  - A performs active close, sends FIN
  - B performs passive close & acknowledges
  - B closes its socket and sends FIN
  - A acknowledges the FIN



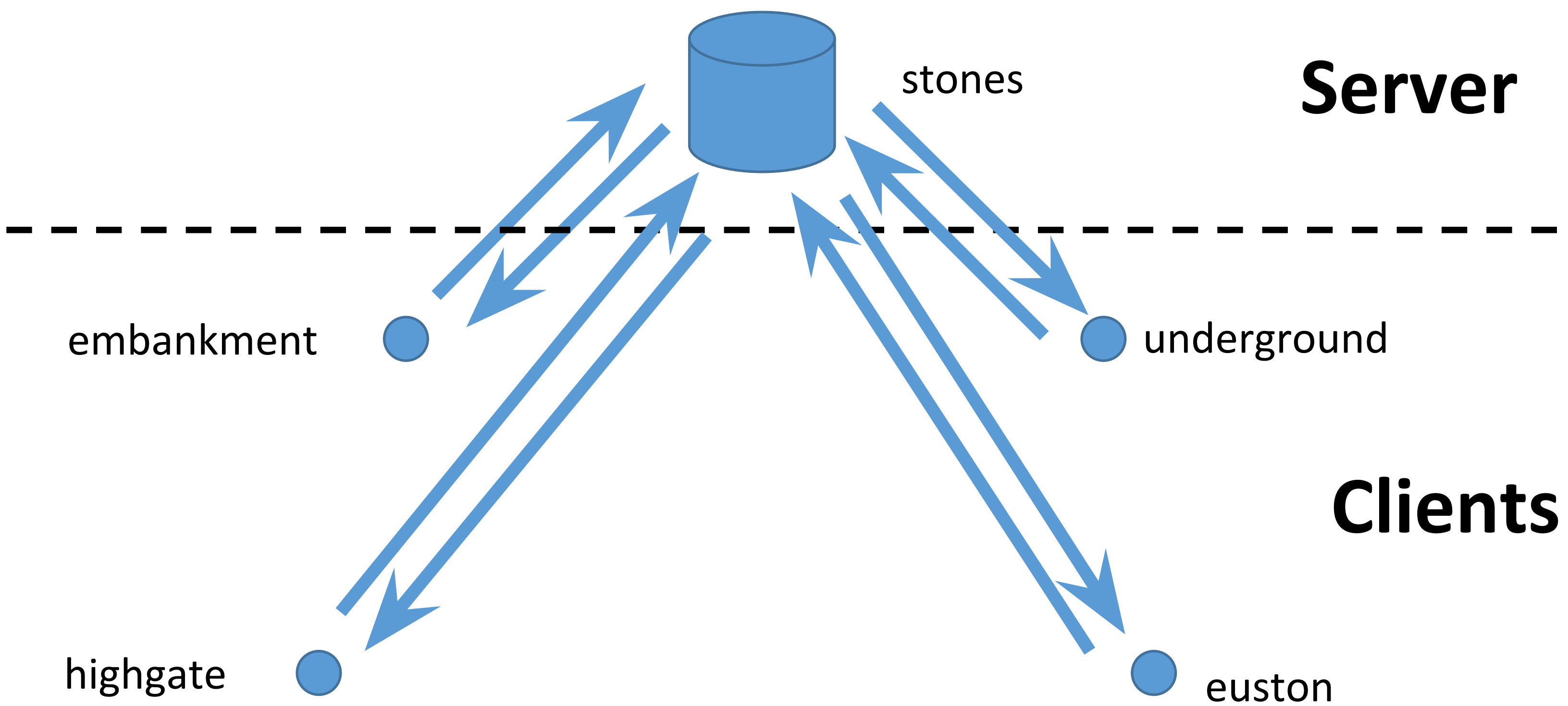
Part 2:

Introduction to Programming Assignment 1

Project Objective

- Develop a text chat application for message exchange among remote hosts:
  - One Server
  - Multiple (at most 4) Clients
  - Communication using TCP sockets

Project Description

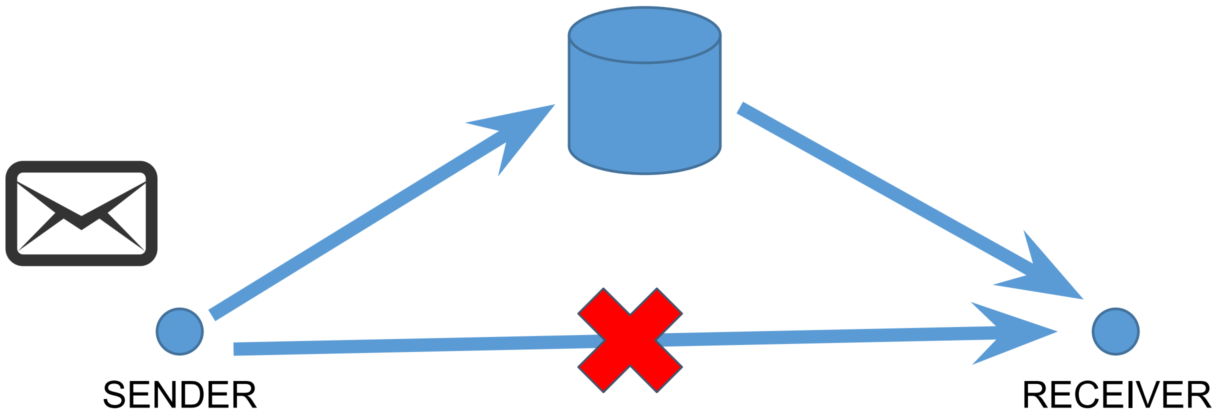


Project Description: Clients

- When launched
  - Login to the Server
  - Identify yourselves
  - Obtain list of other logged-in clients
- Clients send messages
  - Unicast
  - Broadcast
- Clients communicate with each other **ONLY** **through the server**

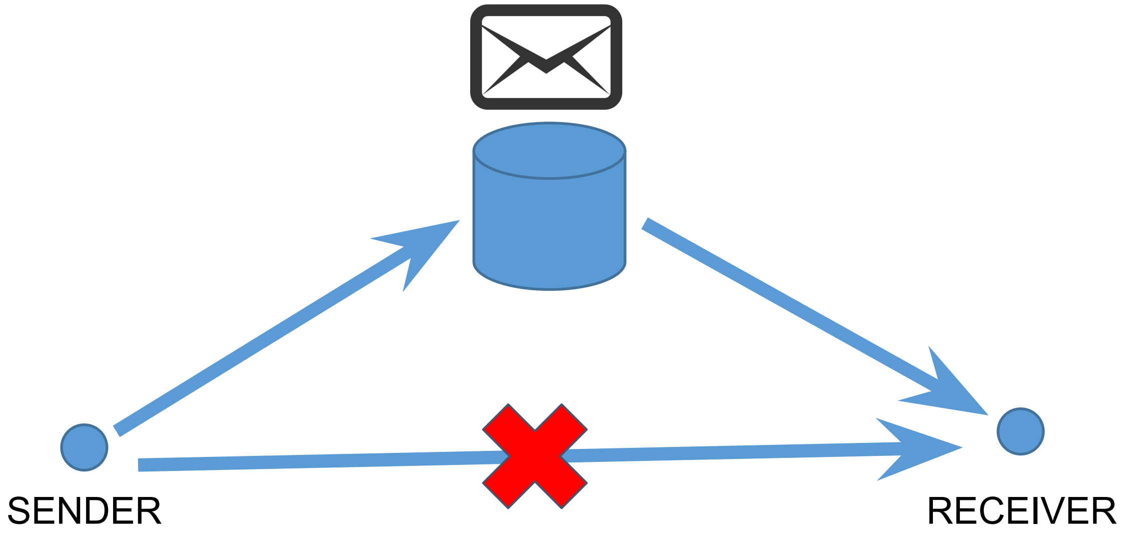
Project Description: Server

- Facilitate exchange of messages between clients
    - Relays all messages
    - Maintains list of logged-in clients
    - Stores/Buffers messages for offline clients



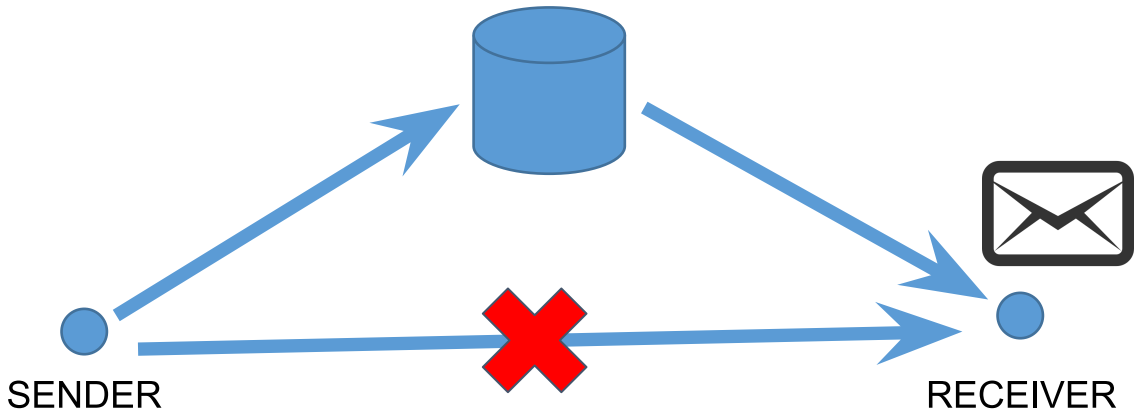
Project Description: Server

- Facilitate exchange of messages between clients
  - Relays all messages
  - Maintains list of logged-in clients
  - Stores/Buffers messages for offline clients



Project Description: Server

- Facilitate exchange of messages between clients
  - Relays all messages
  - Maintains list of logged-in clients
  - Stores/Buffers messages for offline clients



Application Functions

- Your application will have two major functions/interfaces
  - Network operations for the chat application
  - *NIX-like user command prompt (shell) to accept user commands

- Use select() system call
  - No multi-threading or fork-exec

Commands/Events

- Assignment description lists the commands/events you need to accept/handle
- Mandatory/Required output for each command/event
- Follow command output syntax EXACTLY
    - Use the supplied format strings
- All printing using the *cse4589_print_and_log(char* format, ...)* function
- Extra output will make the automated grader fail test cases

Project Requirements

- Make/Run on CSE Servers (stones, underground, embankment, ….), make sure it compiles/runs on these servers.

- Only one program is running on each server, but takes different arguments:
  - ./chat_app s 4322
  - ./chat_app c 4322

- C or C++
  - No external libraries for socket programming
  - No external binaries/utilities

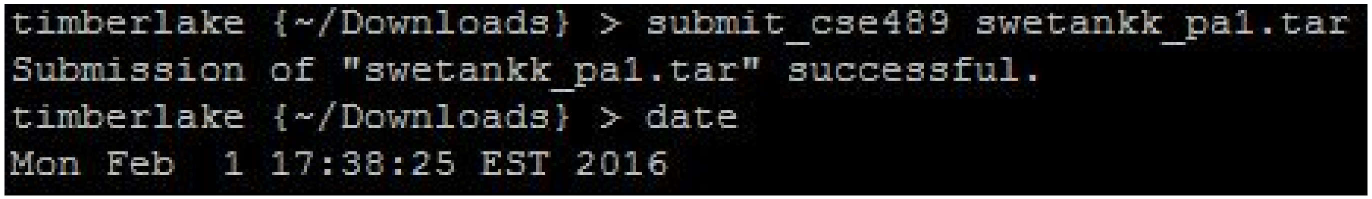
- Use the PA1 template (https://goo.gl/4TBUbw) **[MANDATORY]**

Packaging & Submission

- Use the supplied script in the template to create a package (.tar) from your code.
    - assignment1_package.sh
- Do NOT package manually
- This ONLY packages; **Does NOT SUBMIT**

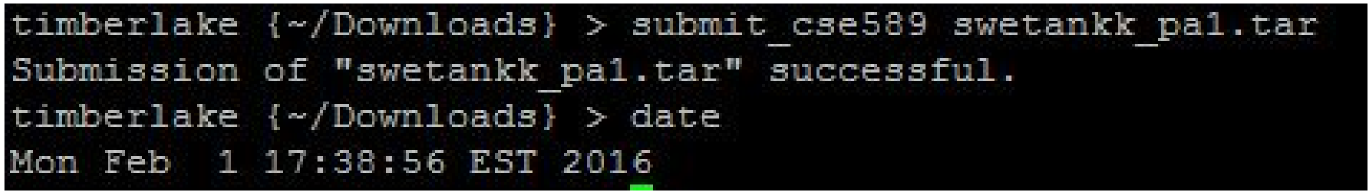
Project submission

- **Use the submit scripts, available on CSE servers.**
  - For CSE 489



```
timberlake {~/Downloads} > submit_cse489 swetankk_pa1.tar
Submission of "swetankk_pa1.tar" successful.
timberlake {~/Downloads} > date
Mon Feb  1 17:38:25 EST 2016
```

  - For CSE 589



```
timberlake {~/Downloads} > submit_cse589 swetankk_pa1.tar
Submission of "swetankk_pa1.tar" successful.
timberlake {~/Downloads} > date
Mon Feb  1 17:38:56 EST 2016
```

Project Grading

- Automated test cases
- Relies on exact output format/syntax
- Use the verifier provided with the template to avoid mistakes
    - Only checks basic syntax/format
    - NO correctness checking

Comment your code

- At the start of the program
  - Author name
  - Short description of your whole program
- Describe the variables/data structures
- At the start of each method/function
  - Purpose of the function
  - Return value
- References for code snippets (like beejs or online links)

Sample

```
// Class or file level comments //
/*
 * proj1.cpp : Single file to handle the file sharing application
 * Starts off as a server or a client on a given port
 * Created for CSE 589 Spring 2014 Programming Assignment 1
 * @author John Doe
 * @created 29 January 2014
 */

//Method level comments //
/*
 * Method to process the input line and split it into arguments.
 * @arg line The user input line
 * @return arguments parsed using whitespace delimiter added into a vector
 * <MENTION ANY REFERENCES HERE: e.g. Copied from stackoverflow thread: http://stackoverflow....
 */


 //code level comments //
 ...
 } else {
   // If valid: close socket, clear from list(s), clear from master fd list too
   //conn id - 1 is the index into clnlist
   //i dont know what to do if i close the server connection?
```

Template Demo