CSE150 – Project 3
Networks and Distributed Systems

Group Thimbles

July 28, 2015

1 Networking syscalls

1.1 connect()

Port mapping Ports are mapped to sockets for both incoming and outgoing connections.

Pseudocode:

```
int connect(host, port){
        disable interrupts
        create a new socket s and assign it to a free port in (0,127)
        s.state = SYN_SENT
        send SYN packet
        block until SYN/ACK recv'd // timeout breaks this
        s.state = ESTABLISHED
        enable interrupts
        return s.fileDescriptor
}
```

1.2 accept()

Pseudocode:

```
int acccept(port){
        disable interrupts
        if there are connections waiting on port
                create a new socket s, assign it that port
        else return -1
        s.state = ESTABLISHED
        send SYN/ACK
        enable interrupts
        return s.fileDescriptor
}
```

1.3 write()

```
int write(fileDescriptor, buffer, count){
        ...
        <netcode>
        ...
}
```

1.4 read()

```
int read(fileDescriptor, buffer, count){
        ...
        // for a socket
        if (s.isOpen){
                read count bytes
                return bytes successfully read
        } else {
                if (socket isn't empty){
                        read count bytes
                        if (socket is empty)
                                delete socket
                        return bytes successfully read
                }
        }
}
```

2 Threads

2.1 Send thread

2.2 Receive thread

2.3 Timeout thread

This thread works like waitUntil, where it loops through the existing sockets and checks for any that have lived past their timeout value. If they have, it closes that socket.

3 Test cases

3.1 connect()

- Attempt to open a connection to a node that doesn't exist

  Check that connect() blocks

- Open a connection to an existing node

  Check that connect() returns

- Close an already-open connection

  Verify that socket is closed on both sides

- Open multiple connections to the same receiving port

  Check that they all send/receive data

- Open a connection, close it and re-open it

3.2 accept()

- Accept a waiting connection
- Accept multiple waiting connections on the same port
- Accept multiple waiting connections to different ports
- Return from accept() on a port that doesn't have a connection waiting

3.3 close()

- Close a connection that doesn't exist
- Close a connection that exists

  Check that it's actually closed

- Close a connection as the sender and as the receiver

  Check

- Close a connection twice in a row

  Check that errors don't occur

- Close a connection with a lot of data waiting

  Check that sender

3.4 read()

- Read from an open socket

  Check that it doesn't block

- Read from a closed socket with data remaining

  Check that it doesn't block

  Check that data is returned

  Return value?

- Read from an empty socket

  Check that it returns 0 w/o blocking

3.5 write()

- Write to an open socket
- Write to a closed socket

  Check that it returns -1 immediately

3.6 Connection

- Write to a connection and read from that connection on the receiving node

  Check that output matches input

  Check that window size never exceeds 16 packets (how?)
- Write to both sides of a connection simultaneously
- Read from both sides of a connection simultaneously
- Write/read to/from a connection with 10% loss rate

  Check that output matches input
- Write a large amount of data into an open connection and read it on the receiving node

  Check that output matches input

3.7 Chat client/server

- Broadcast a message to all clients

  Check that it works for small numbers and large numbers

  Check that output matches input for every client

  Check that sending a message doesn't interleave with receiving a message
- Connect a client, then drop the client
- Connect a client, drop the client, then reconnect that same client
- Connect multiple clients
- Drop multiple clients
- Reconnect multiple clients
- Broadcast multiple messages

  Check that the order of the messages is preserved for a given sender
- Quit the client

  Check that client exists gracefully (how?) after getting a single '.'

- Quit the server

  Check that the server exists upon receiving anything on stdin

  Check that all clients get a message about server closing