Networking Team Design

Server Side

Jeff, Georgi & Eric

Table of Contents

Flow Diagrams 4

Game Logic Thread 4

Listen Process 7

Receive Process 8

Send Process 9

Pseudo-Code 9

Functions used on Game Logics main thread 10

startServer() 11

stopServer() 11

Session.disconnect() 11

Session.send() 11

virtual ServerNetworkEntity.onUpdate(data) = 0 11

Listen Process 12

listenProcess() 12

listenSignalHandler() 12

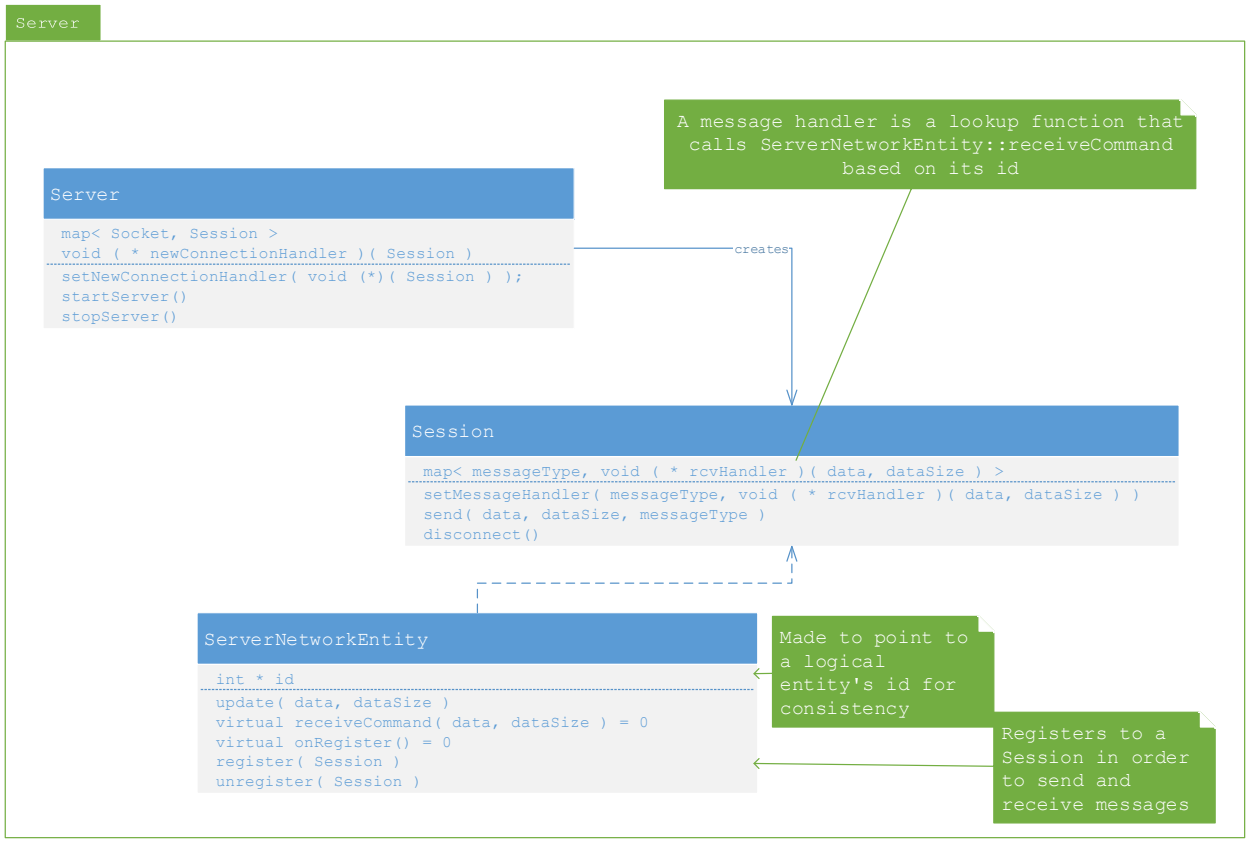
Receive Process 12

receiveProcess() 12

Send Threads 13

sendMessage() 13

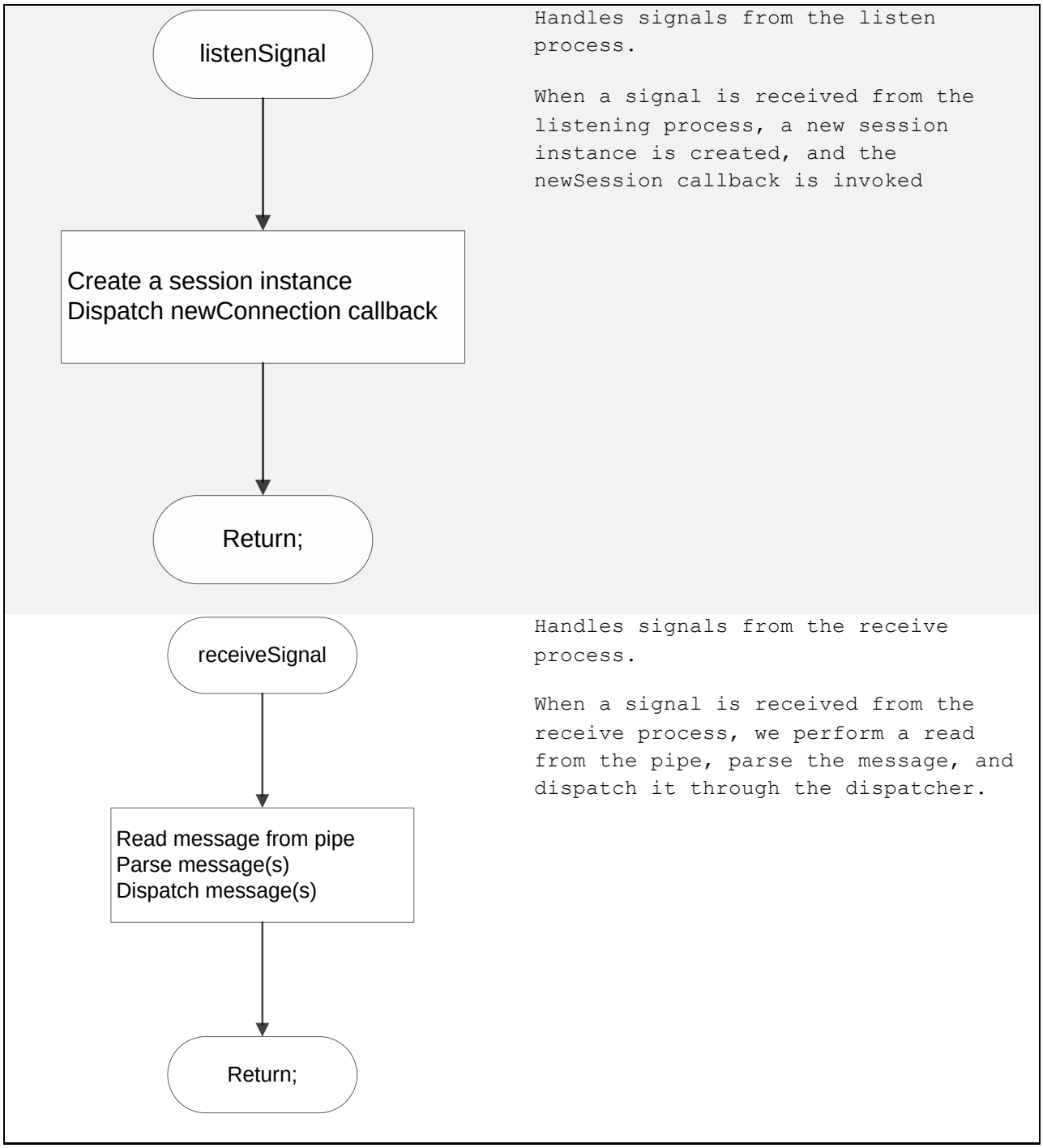
Class Diagram

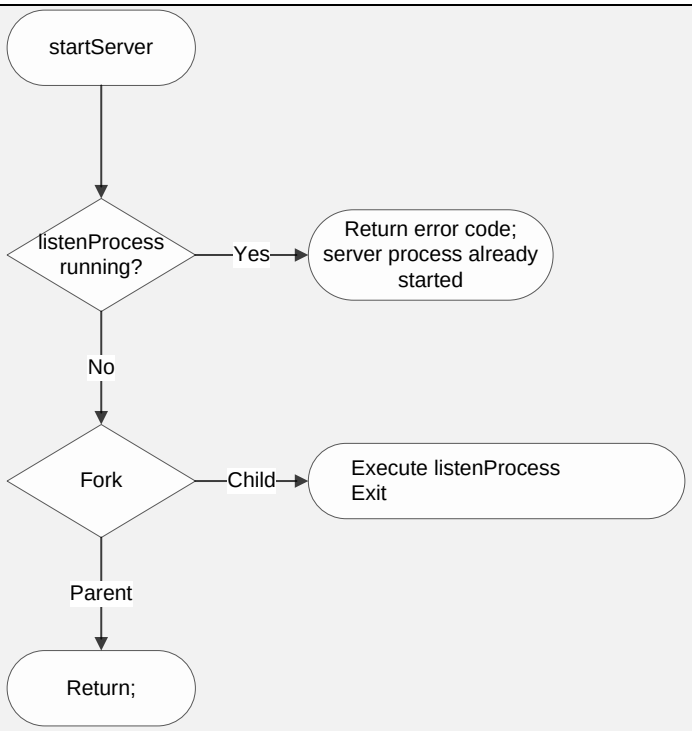


Flow Diagrams

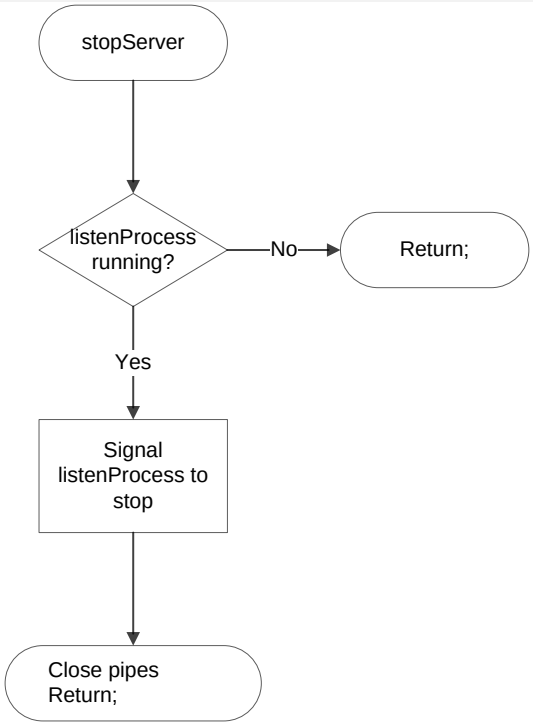
Game Logic Thread

The networking API exposed to the game logic people are all non-blocking functions.

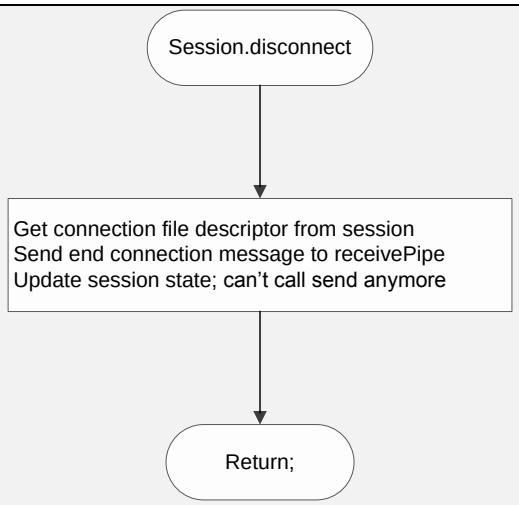




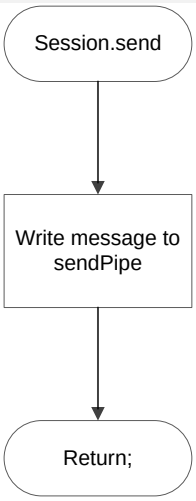
The start server function. It starts the server on a separate process if it isn't already running.



Stops the server process if it is running.



Ends the calling session, and sends an "end connection" message to the receive process.



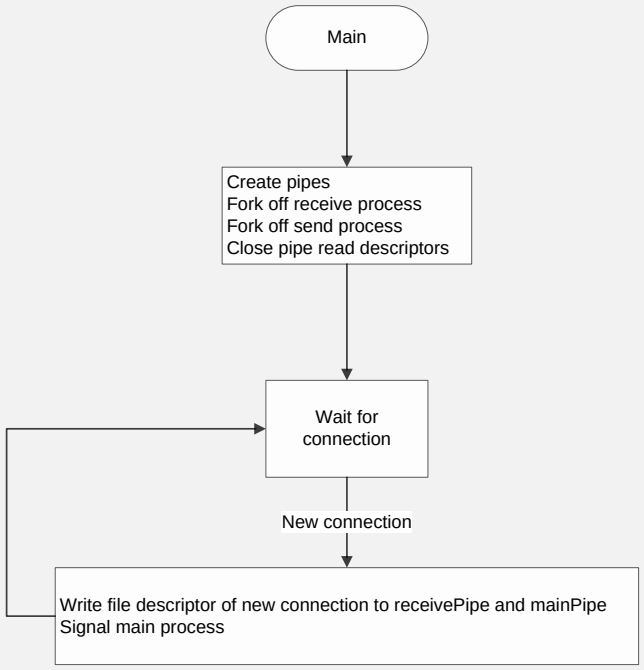
Sends the specified data over the network.

When this function is called, it writes the data to send into a message, and sends the message to the send process.

Listen Process

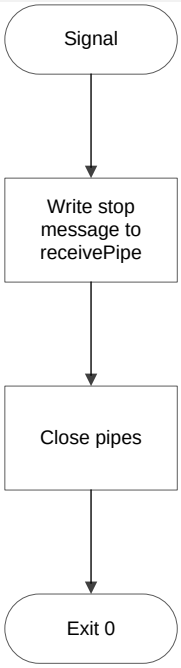
The listen process is responsible for listening for new connection requests, and it informs the main thread (game logic), and receive thread of the new connections whenever a new connection is made.

This process is stopped with a signal.



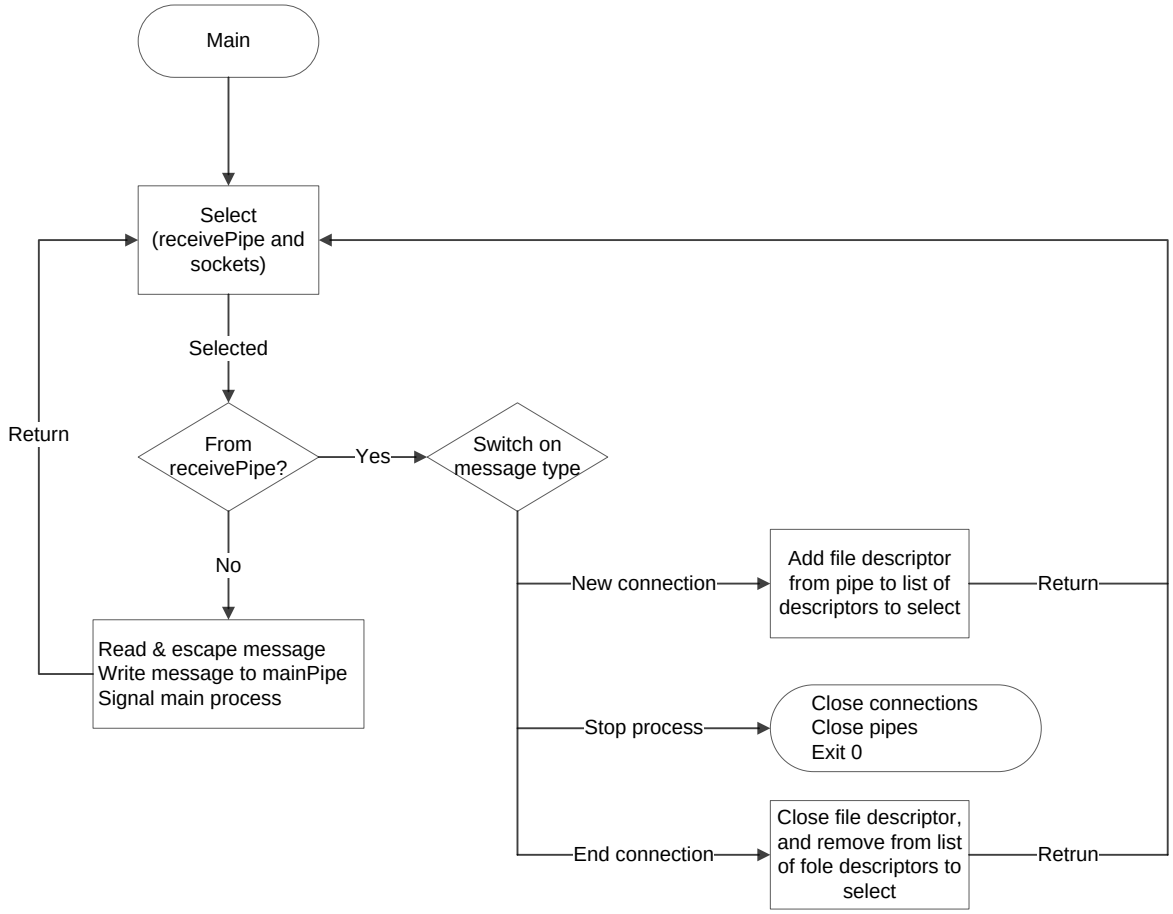
This is the main function of the listening process.

It creates the other processes, then continuously waits for new connections



When the listening process receives a signal, it sends messages and closes pipes as necessary to stop the receive and send processes, then exits itself.

Receive Process



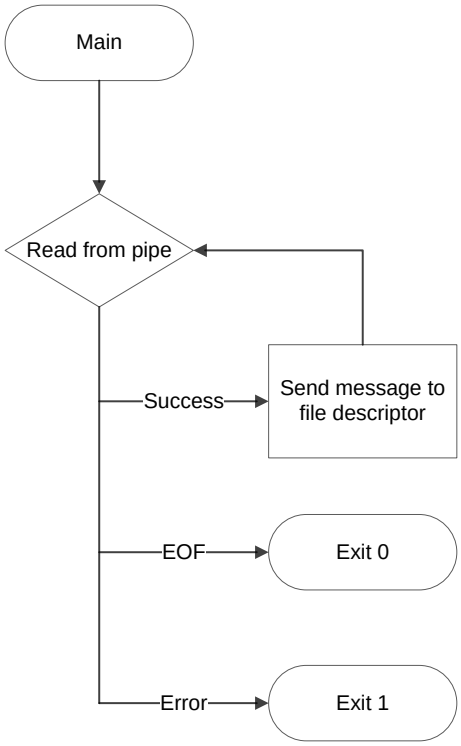
The receive process polls an array of file descriptors. The array contains a pipe for controlling the process, and the rest of the file descriptors are sockets of connected clients.

If the poll function returns a file descriptor of a socket, it adds a header to it, forwards to the main process and finally, signals the main process to notify it of the message in the pipe.

If the poll function returns the file descriptor of the control pipe, it will do one of 3 things depending on the type of control message it is:

- The message is an "add connection" message. This kind of message is sent to this process when a new client has connected, and we want to add the file descriptor of the socket to the array of sockets to poll.
- The message is an "end connection" message. This message is sent to this process when a connection terminates. The corresponding socket file descriptor is removed from the array of file descriptors to poll.
- The message is a "Stop process" message; this process closes all the file descriptors, and ends.

Send Process



This is the main function of the send process.

This function reads from a pipe in a loop:

- When the pipe closes, and returns EOF, then this process exits.
- When the pipe returns an error, then the process exits in error.
- When the pipe returns data, it parses the header, and sends the message out through the specified socket file descriptor, then it loops back to read more data from the pipe.

Pseudo-Code

The following pseudo-code is for the network portion of the server side application. It will be required to handle connections with up to 12 different clients, each having their own session object on the server.

Each session will have multiple network entities. These entities will be mirrored on the client side, and whenever there is a change on one side, that change will be sent across the network to the other side, and the appropriate onUpdate function will be invoked.

There network portion of the server will 4 distinct processes (or threads) of execution as follows:

1. Some functions will execute on the main Game Logic thread
2. There will be a process dedicated to listening for new connections
3. There will be a process dedicated to receiving messages from clients
4. There will be a thread pool to send data to clients

Functions used on Game Logics main thread

This function is invoked when a new connection is detected, it creates a new session instance with its own socket.

listenSignal()

Create a new session instance, and pass it the socket
Add the socket, and session to our map of sessions and sockets
Server.newConnection(Session)
Return;

This function takes a received TCP message and passes it to the correct session based on the receiving socket.

receiveSignal()

retrieve Session based on socket
extract destination id from message
retrieve ServerNetworkEntity based on id
call ServerNetworkEntity::receiveCommand()

Start server is called to initiate the network on the server

startServer()

```
if already started, return
fork, send child to listenProcess()
create thread pool for send
```

Stops the networking component of the server, cleans up pipes etc

stopServer()

```
if !LISTENING
    return
send signal to listen process
close pipes
return
```

Disconnects this session object from the network

Session.disconnect()

```
get socket file descriptor
write end connection message to receivePipe with the socket file
descriptor
invalidate the session...set state to closed; can't send to this
session anymore
```

Assigns a thread to send a message across the network

Session.send()

```
assign thread to sendMessage()
return
```

Send an update to the other side

ServerNetworkEntity.update(data)

```
iterate through session set and invoke their send functions
```

Meant to be overwritten by user to handle an incoming update

virtual ServerNetworkEntity.onUpdate(data) = 0

Method used to register a session so the entity can send and receive to and from the session.

ServerNetworkEntity.register(session, data)

```
add session to set of sessions
Send register message
```

send data

Meant to be overwritten by user. Called when the associated entity on the other side calls the register method.
virtual ServerNetworkEntity.onRegister(data) = 0

Unregisters the session from the entity so it will no longer be able to send or receive updates.
ServerNetworkEntity.unregister(session, data)
remove session to set of sessions
send data

Meant to be overwritten by the user. Called when the associated entity on the other side calls the unregister method.
virtual ServerNetworkEntity.onUnregister(data) = 0

Listen Process

This function is the entry point into the listen process, it is called after a fork.

listenProcess()

```
create receivePipe and mainPipe
fork, send child to receiveProcess()
fork, send child to sendProcess()
close unused pipe read descriptors
create, bind and set listen on listen socket
register listen signal handler callback
while(1)
    when new connection is accepted
           write new socket descriptor to receivePipe
           write new socket descriptor to mainPipe
           signal main process
```

This function is called when the TERM signal is sent to the listen process
listenSignalHandler()

```
write stop message to receivePipe
close receivePipe and mainPipe
close listenSocket
exit
```

Receive Process

This function is the entry point into the receive process, it is called after a fork.

receiveProcess()

```
for(;;)
    monitor receivePipe and sockets for messages
        if message from receivePipe
            if new connection
                update socket list and number
            if stop process message
                close all receive sockets
                close pipes
                exit
            if terminate connection message
                close socket
                remove socket from list
                decrease socket count
                break
        else // from a socket
            read and escape message
            write message to mainPipe
            signal main process
```

Send Threads

sends messages via tcp

sendMessage()

```
    write message to appropriate socket
    if write fails
        return ERROR
```