Friday 5/22/2020: VII.2: I made some small edits of things I hadn't noticed yesterday, and then I re-exported my video.

Thursday 5/21/2020: VII.2: I recorded my screen while I went through the install and did a small demo of some of the features of the product. I actually spent a while on that, since I was trying to capture one specific window for the installation and I couldn't figure out how to do it (until I did). I recorded my voiceovers for each section. I edited the parts together with VSDC and added some simple transitions. I took screenshots of title/section slides that I made to mimic my presentation's theme. Finally, after one last round of editing, I exported it.

Wednesday 5/20/2020: VII.2: No work done; I had two APs today.

Tuesday 5/19/2020: VII.2: I made a plan for the video and looked into applications to record my screen with. I also looked into editing software for videos (to add voiceover). I settled on using OBS Studio for recording and a tool called VSDC for editing.

Monday 5/18/2020: VII.2: No work done; studying for APs.

Friday 5/15/2020: VII.2: I tested my whole project on another computer in order to make sure the installer and the code works on another environment, and found some issues. First, I apparently forgot to change the uninstaller, which does not actually remove all of the files. Next, I found out that the network scanning feature only sometimes works. I did more testing on this and found out the following: with an infinite timeout, Java always gives the correct answer (whether or not a port on a computer is accessible) but it's really slow when the port is closed; with a low timeout of 100ms, you have to run about 50 tests to actually see if the port is open; and with a timeout of 500ms, you still have to run at least 30 tests. This led me to conclude that there is some variance either in Java or on the network related to traffic that changes how long it takes for a TCP connection to occur. Unfortunately, this is not something I can control, and the alternatives of running a lot of tests and having large timeouts take up too much time, so I can't actually fix this. Instead, I made sure the title of each client GUI will show your local IP to make it easier for manual connections to occur. In addition, you couldn't stop the server without killing java from the Task Manager, so I made another window that killed the server when you press a button. After that, I had to re-convert it into modular, use jlink to make a new custom JRE for it, and reapply my NSIS script.

Thursday 5/14/2020: VII.1: I finished up working on my finishing touches section and took screenshots of the installer doing its thing. I reviewed my presentation and made some small changes in wording and such.

Wednesday 5/13/2020: VII.1: I continued describing what I learned for the finishing touches section, and then I wrote about the issues I had. I also realized that I had forgotten to get pictures for the GUI section, so I ran my project and took some screen shots.

Tuesday 5/12/2020: VII.1: I finished the “Hitches in the Road” section for my GUI supersection. I wrote my initial goals for my finishing touches and described what I learned.

Monday 5/11/2020: VII.1: I continued working on the “What I learned” section for my GUI supersection, and I started describing the problems I had when writing the GUIs and how I fixed them.

Friday 5/8/2020: VII.1: I started working on the GUI section today, describing my initial goals for the GUIs. I also worked on the various things I learned with regards to the GUIs, such as both Swing and JavaFX topics.

Thursday 5/7/2020: VII.1: I finished the Low-Level section today by concluding my list of problems and workarounds. I also had to screenshot the run of one of my old test programs to show the progress of Mimic after this stage was finished.

Wednesday 5/6/2020: VII.1: I continued work on the Low-Level section of the PowerPoint, adding more to what I learned and working on the section for problems I had.

Tuesday 5/5/2020: VII.1: I continued work on the Low-Level section of the PowerPoint, detailing the topics I researched. I also added an “Introduction” section to share what I originally intended the overall application to do and be.

Monday 5/4/2020: VII.1: Decided on general format for presentation and what sections I was going to use. I then reviewed my journal entries to remind myself of struggles, what I learned, etc. I started working on the content for the first section of the PowerPoint, which will be Low-Level.

Friday 5/1/2020: VI.1: I wrote a small C++ script to silently start Mimic using the smaller JRE, and tested that. It worked, yay. I then read up on NSIS (a system for making your own installers using scripts), and reviewed a couple of example installers I had made in the past for other projects. I wrote the installers and uninstallers for Mimic and tested them. Seems like they work!

Thursday 4/30/2020: VI.1: I resolved all of the circular require statements that had been created and removed the main mimic package and module, as it wasn’t really doing anything. I set the class to start for the application to the Main GUI (section V), and read up on jlink commands. I used jlink to create a smaller, customized JRE for my project. This took a while, as I mostly had to do trial and error.

Wednesday 4/29/2020: VI.1: I researched Java 9 modules, and found out that they’re essentially just packages of packages, with some access control and encapsulation thrown in. Each module needs its own module information file that tells Java what other packages it needs and what packages it lets other packages use. I first downloaded Java 9 and Apache NetBeans and then created a new modular project. I just turned each of my previous packages into a module:

so the gui package is now encapsulated by the gui module, lowlevel package is now in a lowlevel module, and so on. I then wrote all of the require statements for the modules.

Tuesday 4/28/2020: VI.1: Since I'm ahead, I figured that I would go ahead and look into this step (making it run on machines without Java/converting it to an executable). I didn't really find any satisfactory ways of converting a jar to an executable, so I looked into distributing a smaller JRE with the program instead of forcing users to install Java. I found one tool, provided by Java itself: jlink. It appears to generate a smaller JRE that only contains the libraries used in an application, but it's for Java 9 (I was using Java 8) and needs something called modules, so I'll look into that tomorrow.

Monday 4/27/2020: V.1, V.2: I wrote the GUI that should be launched when the application is run. Since it's essentially just two buttons without any fancy layout, it was pretty simple to implement. The event handlers for each button press were also pretty much just: call other gui and then hide this gui, so it didn't take me very long to write.

Friday 4/24/2020: IV.1, IV.2, IV.3: I made the Server gui, which was pretty simple, as I copied the basic JavaFX code to set up the window from my other GUIs. After that, I just had to research file chooser dialogs in JavaFX a little to fulfill IV.1, and then it was off to the races! It appears that I severely overestimated the time required for this section.

Thursday 4/23/2020: III.6: I implemented the additional details. The first one was coloring each user's name a semi-random color, which required me to change the TextArea I had previously been using to a TextFlow, which supports rich text. I made a few edits in the function that formats the messages and voila! Next, I added a timestamp and some spacing to each message (spacing to accomodate larger usernames and make sure messages all line up correctly). This was achieved by some edits in the same format function.

Wednesday 4/22/2020: III.6: Today I realized that I had forgotten to uncomment and convert the focus listener (so that the place where you write your messages grabs keyboard focus whenever the window is the active one), so I did some research on the necessary behavior in JavaFX and implemented it. I also looked at some images of IRC clients (which Mimic is loosely based on) to get some ideas for additional details to put in.

Tuesday 4/21/2020: III.6: I finished converting the creation of components, so that the behavior now closesly matches what it was previously with Java Swing. However, the channels list was originally too large, so I had to approximate a size for it (due to some stupid JavaFX stuff). After I fixed that, I tested the application out and it works pretty much like the Swing version.

Monday 4/20/2020: III.6: I started on converting the ClientWindow class to use JavaFX. This time, I think a BorderPane is going to look great. First I converted all of the components into the JavaFX forms. Next I re-wrote all of the GUI event listeners to conform with JavaFX. Finally, I

started working on converting the creation of components and their specified behaviors into JavaFX.

Friday 4/17/2020: III.6: I worked on the layout of the auxiliary client GUI, one pane at a time. I ended up using a BorderPane and multiple containers (BorderPane has one left, right, top, bottom, center element; so to put multiple things in the center you set the center to a container with more stuff in it). I think it looks decent, but not exactly what I had hoped for.

Thursday 4/16/2020: III.6: So it turns out that the pane isn't centered in the window, due to some Windows things and offsets. However, you can get the offset between the pane and the window, so I thought I could now center things in the pane precisely! Bad news: the offset is calculated after the window is shown, so you can't pre-calculate the locations for all of the buttons and components. If you just wait until after the window is shown, you get a brief look at the window without the components on it before the components are actually drawn onto the pane + window. That's obivously undesirable, so I'm going to have to make concessions about exact placement and just use a stupid JavaFX layout.

Wednesday 4/15/2020: III.6: I continued research into JavaFX layouts and similar things. I found that you can have pixel perfect control over where everything goes by having unmanaged components on a plane Pane, without any special layout. I started working out where things should go and worked on implementing it for the auxiliary client GUI (one step at a time, of course). However, I noticed some inconsistencies. Things I put at the center don't end up at the exact center, and the window is a little smaller than it should be. I'll investigate more tomorrow.

Tuesday 4/14/2020: newly added III.6: I continued research into JavaFX's workflow and found it to be significantly different from Java Swing. However, since I have time, I'll convert all of my GUIs anyway. I noticed that there was no equivalent Spring Layout to precisely define where all components will go, so I might have more trouble with that.

Monday 4/13/2020: N/A

Friday 4/10/2020: N/A

Thursday 4/9/2020: III.5.b: I continued research on optimization, and found nothing really useful. However, I did find out that Java Swing, what I'm using for the GUIs, is an old technology, and Java now uses JavaFX, so I started looking into that, since I'll probably end up converting.

Wednesday 4/8/2020: III.5.b: I worked on fixing the miscellaneous errors I had found yesterday (I had to debug a lot) and continued testing the integration of the GUI and client. I found no more errors, thankfully. I researched some optimization for Java GUIs.

Tuesday 4/7/2020: III.5.b: I finished up what I consider to probably be the first round of integration and I started testing the client GUI. Some of it works, so that's good. I got a lot of

NullPointerExceptions and found several errors in relation to the changing channels integration, so I started working on fixing those.

Monday 4/6/2020: III.5.a, III.5.b: I worked some more on the layout until I was tired of it; I think I'll just leave it the way it is now. I continued integrating the GUI with the client by way of Java event handlers and the provided methods from the client.

Friday 4/3/2020: III.5.a: I continued messing with the layout of all of my components on the screen and I think I have something that looks decent; it's not the best, but it looks ok. I then started integrating the GUI components into the lowlevel client using event handlers and such, but it's taking awhile. I did more research on Java event handling.

Thursday 4/2/2020: III.5.a: I started working on the layout of the main chat environment, and just the layout. I wasn't certain that the SpringLayout I had previously been using was going to fit my needs for this application, so I did a bit more research and found the GridBagLayout, which I think should work nicely. However, each component needs to have a set of constraints that take awhile to work through. There are so many different combinations of constraints for the different components, and each constraint can make the layout completely different. So I'm forging ahead.

Wednesday 4/1/2020: III.1, III.3, III.4: I connected the client GUI to the lowlevel client, so that we have a client instance in the GUI, for us to make use of the various methods I've exposed. I changed the 'server flags' concept and made it redundant: now the client GUI just needs an IP and a port to connect, so the server GUI will pass 127.0.0.1 and the port it's on, and the auxiliary client GUI will pass the provided IP and port. I researched how to make message boxes and prompt for data in Java, and then made a message box to prompt for a username and some error message boxes for an invalid username or being unable to connect to the server provided.

Tuesday 3/31/2020: III.2, III.3: I made a basic JFrame for the client GUI and linked the auxiliary GUI to this basic GUI so that the auxiliary GUI will pass the selected/entered port and IP along and start the client GUI. This also closes the auxiliary GUI, and I added some error handling in relation to server connectivity and the validity of the IP and port provided.

Monday 3/30/2020: III.2: I worked a while today. I changed the layout to a SpringLayout and made the main panel a JTabbedPane and added the tabs for both manual entry and network scanning entry. I finished the layouts of both manual and network scanning, and both took a while to tweak until I liked their looks. I interfaced the GUI with the actual network scanning functionality and made sure to inform the user of any errors. The network scanning GUI now asks for a port, runs the scan after a button is hit, and then updates the same page to now offer a list of servers to choose from or puts out an error box.

Friday 3/27/2020: III.2: I started working on the auxiliary client GUI, which offers access to either manual entry of IPs or the network scanning function to discover other hosts. I had to brush up on my Java GUI stuff, and I started working on creating the basic layout of the components and buttons on the screen. This GUI stuff is really touchy and is taking a lot longer than I thought it might.

Thursday 3/26/2020: II.4: I made some alternative methods to make the scanning process faster. First, I just did the same thing as yesterday, but with a parallel stream, which took the scan time down to 3s, which is acceptable. I tried to make it even faster by manually splitting the generated IPs into several sections and having each section run at the same time using Threads, with each section also using a parallel stream. That method was also 3s, so no real benefit there unfortunately.

Wednesday 3/25/2020: II.4: I worked on creating a method to iterate over all of the generated IPs and first test if they were reachable or not. This was accomplished using InetAddress.isReachable, and took far too long, so I looked into some possible optimizations by changing loop types and so on, but nothing really helped. I moved on to just trying to connect to the specified port on each machine, and that took a lot less time for some reason (only 26s). I'll try to make it faster tomorrow.

Tuesday 3/24/2020: II.4: Tested my implementation and realized that it didn't work because I was being an idiot. I thought about it and planned an alternate version out. I implemented that with recursion, and the Network class can now accurately generate all of the possible IPs on a network. It takes a while for larger networks though, so I might have to look into optimization later.

Monday 3/23/2020: II.3, II.4: Realized that "Parse messages from the server" is essentially a function of the GUI, not the client, since there's no lowlevel stuff happening (just how/what to display). Started work on network scanning functionality and wrote Network class to represent a home network with an IP prefix and a netmask.

Friday 3/13/2020: N/A

Thursday 3/12/2020: II.2, II.3: I finished up exposing necessary functions to the GUI with no problems. Phew. I started working on the actual logic behind the methods determining what to display on the GUI/driver.

Wednesday 3/11/2020: II.2: Continued research on networking: network interfaces, different types of InetAddresses, netmasks, etc.. Finished up the methods necessary to expose to the GUI, and I started working on a small driver program to test them. So far, so good.

Tuesday 3/10/2020: II.2: Continued research on networking. Started specific methods to expose to a GUI/driver class, like setUsername and initiate. No problems so far, as it's pretty simple.

Monday 3/9/2020: I.6, II.2: Remembered to add an option to be able to specify the port of the server through the config file. I then had to modify the Client and Server to make sure they could run with a specified port instead of just the default port. I then continued researching Java's networking options for the client.

Friday 3/6/2020: I.6: Continued work on config files. Finished implementing silent channels and completed muted users, so that the config file can specify users who can't send any messages, but can still read them. I encountered no problems today (thankfully), so everything worked the first time. I think I'm done with the config file stuff, but I might add more later if I have time.

Thursday 3/5/2020: I.6: Continued work on config files. Fixed the multiline default message error, it had to do with escaping \n and the BufferedReader only being able to do readLine. Added ability to disable the default message popup through the config file. Continued working on implementing silent channels.

Wednesday 3/4/2020: N/A

Tuesday 3/3/2020: I.6: Continued work on config files. I worked on making a way to specify and show a default message to each user when they connect to the server, but I'm getting some weird errors for multiline messages. I started working on a way to make silent channels where no one can talk (so like for the welcome channel, where the client program puts new user notifications and stuff).

Monday 3/2/2020: I.6: Researched java.util.concurrent.Flow interfaces as another probable solution. This was the Observer pattern/Publisher-Subscriber model, and I was considering using it until I fully perused the documentation, and I discovered it wasn't suitable for my use. I then considered looking at Java swing classes to look at how they did event listeners, and I discovered they essentially had a main loop that checked. I considered refactoring my code into that form too, but it's the same number of threads and essentially the same code, too, so I've finally decided to stick with what I have so far.

Friday 2/28/2020: N/A

Thursday 2/27/2020: I.6: Continued research on event listener patterns. I looked at the source code for some GUI components to try and understand how they accomplished event listeners, as I still want a more elegant way of solving the below problem. I wrapped up the implementation of reading from a config file. I'll see if I can add to it tomorrow.

Wednesday 2/26/2020: I.6: Researched some "Observer" patterns and learned more about general Thread mechanics in Java. This was to try to find a more elegant and efficient version of my current listeners (which are essentially just while true loops checking a condition in another thread), in order to both solve the problem from yesterday and improve the quality of my code.

Tuesday 2/25/2020: I.6: I made sure the default channel can't be forbidden to anyone, so when a channel is forbidden, the client is just changed to the default channel. Fixed an error of timing: the buffered reader listener would pick up on the messages sent to change channels and but in, grabbing the lock from the changing channels function and just wait for a message that would never come. I just added another if statement to make sure the buffer was actually ready to be read. I'll probably change that to a more elegant synchronized block tomorrow.

Monday 2/24/2020: I.6: Worked on reading from a config file, specifically with forbidding specific channels from specific usernames. I had a bit of trouble with regex checking the validity of the config string, but I fixed that. Currently working on what to do if the user is forbidden from a channel, as there's no immediate recourse defined.

Friday 2/21/2020: I.6: Worked on reading from a config file, specifically with allowing users to define forbidden usernames clients aren't allowed to use. This includes regex, so a user can forbid usernames more generally.

Thursday 2/20/2020: I.6: Started research on and worked on reading from a config file. I thought about what I wanted each config file to have and provided a default in the code, so that if someone doesn't know what they want, then they can just use my defaults.

Wednesday 2/19/2020: I.3: I finally tracked down that pesky error from Friday. It turns out it wasn't anything to do with spamming. Instead, when the message was whitespace only, trim() trimmed off the entire end, leading to a StringIndexOutOfBoundsException when I tried to access the message string past the end (since a space I had expected was trimmed off). I replaced the trims in the combined listener and the client buffer listener with my custom trim that only trims off newlines.

Tuesday 2/18/2020: N/A

Monday 2/17/2020: N/A

Friday 2/14/2020: I.3, I.5: I think I finished up the management thread. I also continued testing the server/client code, and I found several bugs. I fixed the two bugs related to changing channels (incorrectly setting the index when the channel changed, so that the newly changed client gets old messages and one other minor one). Haven't been able to fix the other bug yet (exception when one client spams a lot of messages).

Thursday 2/13/2020: I.3, I.5: Worked on the management thread and tested. Fixed a bug that would not skip messages it didn't need to read (so the client was getting stuck and not reading past that). Researched GUI some more, specifically with requesting focus.

Wednesday 2/12/2020: I.3, I.5: Worked on the management thread some more. Combined the previous two listeners (buffered reader and message) into a single thread to try to minimize the amount of threads necessary on the server machine.

Tuesday 2/11/2020: I.3, I.4, I.5: Revised the changing channels procedure; it was previously sending a duplicate of the default channel. Worked on the management thread some more and looked into possible ways to reduce the number of threads, cause it's 3 threads / client + 1 management thread + 1 GUI for the local client currently.

Monday 2/10/2020: I.3, I.5: started working on another thread to manage the unread messages left in the messages construct. Once a message is read by all threads in the specific channel, it should be deleted. Still working on actual logic.

Friday 2/7/2020: I.3, I.5: Did more extensive testing and found a bug in the message listener: it was reading multiple messages over and over again, so I worked on debugging that and fixing it. Continued research on GUI.

Thursday 2/6/2020: I.3, I.5: Finished up the message listener and started more testing. So far, everything works the way it's supposed to (surprisingly). Started researching GUI stuff to be able to more effectively test the client-server code.

Wednesday 2/5/2020: None, was at the club fair.

Tuesday 2/4/2020: I.3, I.5: Worked on the message listener and did more research on threads and reading from a config file.

Monday 2/3/2020: I.3, I.5: Finished buffered reader listener and started working on the message listener for server threads to send out any new messages from other clients.

Friday 1/31/2020: I.3: Worked on buffered reader listener and fixed a NullPointerException by making sure to remove a reference to a non-existing object upon deconstruction of a server thread.

Thursday 1/30/2020: I.3: Worked on buffered reader listener and worked on the message listener for the server. Made sure they were both interrupted in addition to being controlled by a running flag.

Wednesday 1/29/2020: I.3: Worked on buffered reader listener for the server and made a similar class for the simple client (so that they have the same methods of accessing the data passed).

Tuesday 1/28/2020: I.3: Worked on buffered reader listener some more and did some intermittent testing, fixing NullPointers (had to do with errors) and IndexOutOfBoundsExceptions (had forgotten to add to some lists intially).

Monday 1/27/2020: I.3: Worked on buffered reader listener and started a message listener to disseminate messages (will write when I get around to it).

Friday 1/24/2020: I.3, I.4: revised the server/client protocol. Worked on the buffered reader listener and researched locks/synchronization between threads.

Thursday 1/23/2020: I.3, I.4: finished deciding on protocol between server and client. Continued working on the ServerThread class and started implementing a listener on the server's buffered reader to handle messages from the client.

Wednesday 1/22/2020: I.3, I.4: tested basic server + thread, so it does work with multiple clients. Continued working on the thread class, and worked on the format for communication between client and server.

Tuesday, 1/21/2020: I.2, I.3: wrote loop to handle connections from multiple clients with threads, started working on basic thread for the server, researched event listeners for later.

Friday, 1/17/2020: I.1, II.1: completely finished simple server and tested it with a simple client I wrote (to be improved upon much later). I found out that PrintWriter could not be used with sockets, so I changed it to DataOutputStream, which fixed the issue I had been getting (no sent data).

Thursday, 1/16/2020: I.1, continued research, finished up most of the simple server, and worked on doing error handling for the various exceptions thrown in a server. No roadbumps yet.

Wednesday, 1/15/2020: I.1, did research on simple multi-threaded servers in Java, and started implementing the simple server.