Sergey Naumets (snaumets@uw.edu 1025573)
Joseph Shieh (josephs2@uw.edu 1031718)

Tor61 Write Up

(3/12/14) EDIT: Our code now works for proj3.html and cnn.com!

We found the bug that was causing our browser to load duplicate data over and over again. Apparently in our listener we used "while (dataInputStream.available() >= 0)", so we keep looping through and reading. In our while loop, we used the buffer over and over again to do readFully() from the input stream. However, apparently even when there is no data, readFully() will still return us the current buffer even if it didn't read in new data. Therefore, once we fixed it so that it will zero out the buffer after every read, everything works happily. The only thing I would add is that, for some reason, adding relay end was giving us some problems, so I removed that from our code. We can, however, load proj3.html, cnn.com, and youtube.com correctly, so I think our implementation does reach Tier III.

(3/10/14) First of all, our implementation doesn't quite reach Tier III. It comes short of this tier simply because we weren't able to fetch complicated web pages correctly, and also because we decided to skip over the self-looping circuits until we were able to fetch some complicated webpages consistently. That being said, the version that we are turning in expects that there are at least three unique nodes that are fetched from the registration server other than the fetching node; this would allow us to make a circuit of length three. The nodes that are launched before there are three other nodes available simply don't create circuits and print a message to the terminal saying that. However, our implementation did allow us to fetch the simple.html page and yahoo.com (not sure why, because this seems like a complicated page), which shows that the foundation of the implementation is correct. More specifically, things like circuit creation, routing tables, message passing and forwarding, and the formats of the messages themselves should all be correct; this encompasses a majority of the necessary functionality for the Tor61 network to work. When we tried to fetch a relatively complicated page like the spec for this project (proj3.html), we often get inconsistent results. Sometimes the page will hang for longer periods of time and it will usually load the full html contents, but often with different CSS properties, and often with duplicate data. We didn't have a chance to find the cause of the duplicate data being sent back to the browser because we ran out of time. But it did seem like the duplicate data being sent back was the exact reason why we couldn't load complicated webpages, so if we were able to fix that error, we think that we would jump to a complete Tier III implementation, and maybe even Tier II.

The best way to see the functionality that we completed is first to start four separate instances (we did this with four separate terminals) of our Tor61Node. The nodes need to have the same group number, and different instance numbers, as well as different proxy port numbers. The first three nodes would fail to create circuits because there aren't yet four unique nodes that can be fetched from the registration server. Once the fourth node goes up, it will be able to fetch four unique nodes, so it will successfully create a circuit of length three. We output

enough information to the terminal to see that the circuit creation is correct, such as when a nodes receive open, create, and relay extend messages, and when they forward or respond to those messages. After the circuit is created, we can request a webpage (probably best to first try simple.html) by contacting the fourth node's proxy port.

Now we wanted to give some feedback as to why this project was so different and possibly more difficult than the CS projects we're used to. The first thing that comes to mind is just the amount of freedom that we had in this project. Everything from the programming language to the implementation details was up to us. The fact that students could choose to implement in whatever language they want, makes it much more difficult for TAs and the professor to help with issues that might be language specific. Another thing that comes to mind is that the first two projects provided a lot of freedom in the implementation, maybe a little too much again. This was less of a problem for project one because we just had to make the register and fetch methods take parameters instead of user input, but to integrate project two into the node was a little more time consuming and error prone. To be fair, we were given a warning to try to make it so that our code was modular, but it still required a good amount of time to just refactor it to allow it to be properly used as part of the Tor61Node. I imagine that project three would be easier as a whole if the specifications for the first two projects were more strict, and if we were to design the methods, classes, and threads in a way that they can be directly used as part of project three, but I understand that this may not be an easy task. It also seemed like project three just required a much larger workload in general, and that it could instead be split into smaller projects, like first being able to create a circuit of length three with unique nodes, and then maybe adding the abstraction of a stream to that, and then finally refactoring to allow for self-looping circuits if necessary, but that is just one idea. It would be completely fair and probably more helpful to have several more smaller, more structured, projects with smaller timetables, say a single week instead of two.

Another thing that comes to mind as to why this project was more difficult than many previous projects is that we haven't had too much experience with threads in previous courses. It seemed like in this class, we had to make many different threads that perform different tasks on many different occasions, and often times these threads needed to communicate between one another. Having little prior experience with this, along with having little information as to what exactly certain threads do, resulted in designs that we seemed to kind of just hack together to get the job done because things weren't always clear and straightforward, and because we didn't have a great understanding of how exactly these things would be used in the end product of a Tor network.

Going off of the first few points, this project also required a lot more "design" time than were used to. Even with a large amount of design time, it seemed very difficult to completely understand the entire project and the specification during the design phase, because there was a large amount of information to take in and make connections between. It seems like some of the details can't be completely understood until parts have already been implemented and tried until an error makes clear what some misinterpretation

was. And especially in project three, it seemed like even if the design was very well thought out, it's still possible to miss some small detail that could require a lot of refactoring and redesigning of other aspects.

To tie it all up, it just seems like this project had a very wide array of information that required both high and low levels of understanding, and it seems like in general, the less strict and less structured a project is, the harder it becomes for the TAs and professor to help students with issues.