Token Ring Project

CprE 489

Ray Kinsella (Solo, 100%)

**Assumptions:**

Over the course of designing, implementing, and testing my solution to this project, I found that the project description was not clear on many of the implementation details and focused more on expected functionality. This is not a bad thing, and I personally prefer the realistic nature of allowing us to solve the problem however can. That being said, I did make some assumptions in certain areas of the project.

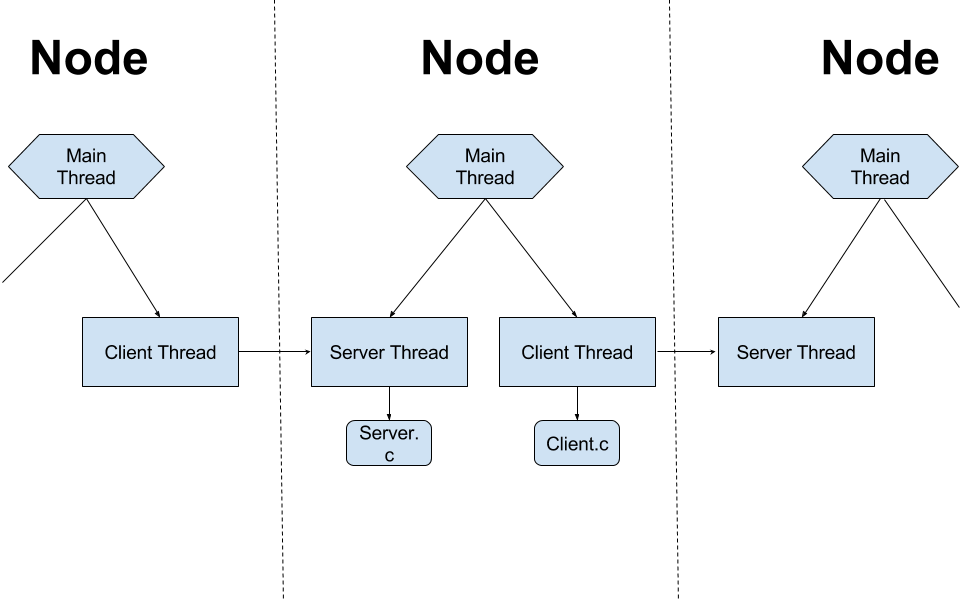
Starting out I was having trouble with the hostname field in both client and server, even after changing the hostname to my own system’s name. Instead of using hostname I added an integer port argument to both client and server, and set the IP address to localhost. While this does make the code less portable, it allows me to offer n-node token chains.

The other large assumption I made was about the behavior and use of the token. From what I have followed in class and researched on my own, tokens act as a currency for the ability to send a message. I was unable to follow what was meant in the following statement.

*A node which has a message to transmit must wait for a unique bit pattern, the token, which we assign the* ***DLE-ETX*** *code. It then changes it to the* ***DLE-STX*** *code and starts transmitting the data frame. At the end, it appends the token to the message.*

I took the above to mean that when a message is sent, the start DLE-STX bytes are modified, and the node sending the message spends a token to deliver its message. When the message arrives at the destination, that node receives a token as well as the message. If a node sends a message and the destination does not exist or the destination is itself, the system will notify you on what occurred, and the node gets its token back.

**Diagram:**

When creating the token chain, each node is executed individually by providing port numbers for the client and server threads as arguments. The first node is executed by proving the initial client port and a server of the port you would like the final node you create to attach to. For example in a 3 node chain, node 1 could have arguments 5000 and 5002, node 2 5001 5000, and node 3 would have to be 5002 5001 to properly connect the chain.

**Creative Component:**

As described above my solution to the project can create an n node chain. This is not necessarily an innovative idea, but it does supercede what the project requested of me. I feel that through this my code is very modular, and is an easily expandable service. Currently creating a n node chain via my code would take up n ports, and only run on the local machine. Viewing this as a production product, obviously it would need to run on multiple devices and connect IP addresses instead of ports.

In addition, not somthing I developed, but I implemented TMUX in my two bash scripts. TMUX is a terminal multiplexer that is configurable in bash, so my scripts init.sh and initMulti.sh create 3 and n instances of node respectively in their own pane of a multi pane window.

**Summary:**

This project was a good amount of challenge, and I feel that I had to apply a lot of the skills that this course has taught me to complete it. It would have been nice if the project description was more explicit, as I felt that I needed to make a lot of assumptions about what was meant in the description.