**HOW TO RUN THIS PROGRAM:**

Steps - FROM THE PROJECT HOME DIRECTORY

1. Compile the token ring.
    a. ant compile;
    b. ant jar;
2. Compile the bridge.
    a. Enter the bridge/ directory.
        i. cd bridge/
    b. make;
3. Start the bridge.
    a. Go back to the root directory.
        i. cd ../
    b. ant -Dprop1="ring1.conf" run;
    c. ant -Dprop1="ring2.conf" run;
    d. The port numbers for the bridge to connect to will be displayed, the bridge will ask for them when it starts executing.
    e. NOTE: Each of these commands will need to be run in a new terminal window.
4. Start the bridge.
    a. Return to the bridge directory
        i. cd bridge/
    b. ./Bridge bridge_log.txt
    c. Enter the port numbers it prompts for.

| Feature | Status/Description |
|---|---|
| Multiple Executables | ***Complete*** |
| Configuration file loading and parsing | ***Complete*** |
| Construct all the networks specified in configuration file | ***Complete*** |
| Join the bridge into all the token sub networks | ***Complete*** |
| Successful frame buffering in bridge | ***Complete*** |
| Bridge successfully tranports non-local traffic across sub-network | ***Partial.***<br><br>**The bridge hangs shortly after execution.** |
| All nodes and networks shut down after all the data has been transmitted | ***Missing*** |

| | |
|---|---|
| Bridge creates log file | ***Complete*** |
| Program loads input file | ***Complete*** |
| Program parse input file | ***Complete*** |
| Program generates tokens (one for each ring) | ***Complete*** |
| Nodes generate proper frames | ***Complete*** |
| Nodes run as autonomous thread | ***Complete*** |
| All nodes terminate after all data in the network has been transmitted | ***Missing*** |
| Program can be executed via command line | ***Complete*** |
| Randomly accept a frame | ***Complete*** |
| Extract all fields from a frame | ***Complete*** |
| Noder switch between listen and transmit state | ***Complete*** |
| Pass the tokens and frames to the next neighbor | ***Partial.***<br><br>**Bridge hangs shortly after beginning execution, but nodes pass their frames perfectly.** |
| Print to the output file with correct format | ***Complete*** |
| Documention is complete | ***Complete*** |
| Well documented code | ***Complete*** |
| **Extra Credit** | |
| Bridge and token are implemented in different languages. | ***Partial.***<br><br>**The token rings are implemented in Java, and the bridge in C++. But the bridge executes poorly.** |
| Dynamic bridge learning for network topology | ***Partial.*** |

| | |
|---|---|
| | **It's implemented, but the bridge has so many problems it's impossible to tell if it is effective, and/or how buggy it is.**<br><br>**As frames are received by the bridge it notes the source address and source ring, then adds that address to the list of known nodes the ring is originated from. When it transmits frames, the destination address is checked against the list of mapped source addresses. If there is a match, it transmits appropriately, else it floods.**<br><br>**The log file shows this works well until the entire bridge hangs.** |

Known Bugs:

The Bridge hangs shortly after execution. For some reason it hangs when recv()ing the first or second token sent to it by the token rings. I'm unsure why this occurs, or how to fix it. As a result, the entire system only passes ~10 frames for a 10 node internet.

It is necessary to Ctrl-C out of both token rings and the bridge in order halt execution.

*Log File Events:*

Sub-ring connections
Frame reception
Token reception
Errors
Dynamic Mapping events
Transmission information

*Bridge Buffer Synchronization Method:* Two std::queue's were used to synchronization token ring's traffic. Frames are pushed onto the appropriate queue as frames are received; they are popped off as tokens are received and transmit states are entered.

*Method for communicating socket numbers to bridge:* As the rings are started, they display the port number for the bridge to connect to. As the bridge initializes, it request's these numbers through console prompts. The bridge then connects to the monitor node of each sub-ring and passes it's own port information. The monitor then creates a special frame

containing the bridge's port number, which is passed around the token ring, where the last node in the ring receives, parses, and uses it to connect to the bridge.

File Descriptions:
the proj_files/ directory contains the python scripts, and is home to ALL the output-file*, input-file*, and sorted-output-file* files. The generation and grading scripts need to be run from this directory.

The TokenRing/ directory contains the token ring *.java implementation files.

The bridge/ direction contains the bridge *.c and *.cpp implementation files, as well as the LOG FILE (bridge_log.txt).

The src/ and build/ directories contain the files used by Ant in the build process. These shouldn't be touched.

Misc
Everything must be done in a Unix environment.