15-418 Project Checkpoint

Ashwin Adulla, aadulla

Title: Modeling Interconnection Networks

https://github.com/aadulla/15418_Interconnection_Network_Simulator/

Overall Progress:

I have implemented most of the underlying architecture of the simulator and have been able to run a few large simulations to stress test the system. The simulator makes use of the OpenMP API to support multithreading in multiple stages (i.e. setup, simulation, data logging). Following the simulation, the results are logged to a series of .txt files which can be visualized as plots by a built-in python interface to matplotlib. As of now, I have only implemented the following configurable interconnection network options:

- Topology
    - Mesh Network
- Flow Control Algorithms
    - Packet Store Forward
- Routing Algorithms
    - Dimension-Ordered X-Y
    - Dimension-Ordered Y-X

Simulator Workflow:

1. **User Configures Simulator:**
   Prior to launching, the user must create a test directory with a 'config.txt' file that contains the parameters for the simulation. Following the simulation, the results will be logged into this test directory as .txt files. An example of a 'config.txt' file is shown to the right. The user must configure all the provided parameters with specific key words that are stated in 'options.txt' which will be located the main directory of the project (I have not made it yet). To launch the simulation, the user must then provide the path to his test directory with a command line option.

```
/* Network Parameters */
Network Type: Mesh
Number of Processors: 100
Number of Routers: 100

/* Router Parameters */
Router Buffer Capacity: 10
Number of Virtual Channels: 1

/* Message Parameters */
Packet Width: 1
Number of Data Flits Per Packet: 1

/* Routing Algorithm */
Routing Algorithm: Mesh XY

/* Flow Control Algorithm */
Flow Control Algorithm: Packet Store Forward

/* Simulator Parameters */
Number of Messages: 10000
Lower Message Size: 20
Upper Message Size: 50
Message Size Distribution: Random
Message Node Distribution: Random
```

2. **Network Creation:** When the user launches the simulator, his ‘config.txt’ is first parsed for all the necessary parameters and the network is subsequently initialized with those parameters. To parallelize the network creation, OpenMP is used so that multiple processors and routers can be initialized and connected at the same time as there is no sequential dependency between these connections. The simulator also uses OpenMP to create the list of messages that will be transmitted across the network.

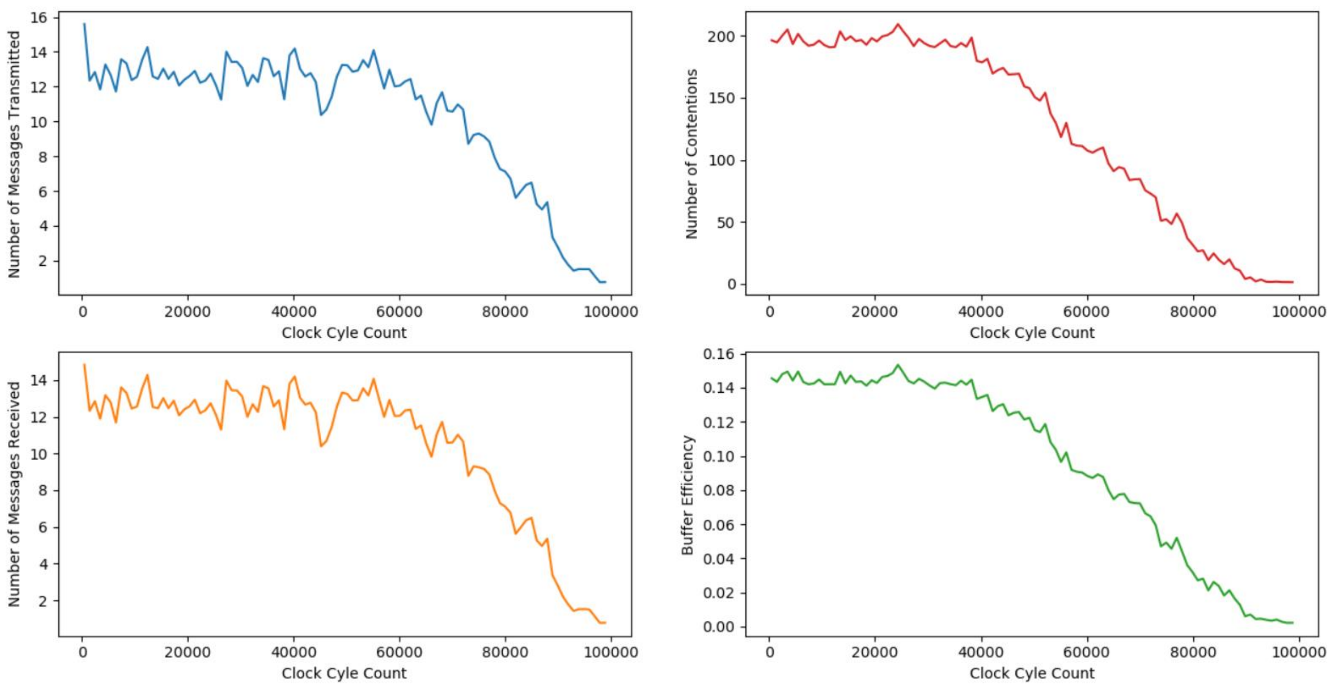
3. **Simulation Run:** Once the network and the message list has been created, the simulator launches the simulation. The simulation keeps running until all the messages in the message list have been transmitted by the appropriate processors and then received by the appropriate processors. At the end of the simulation, a message transmission information table will be printed out which shows the statistics for each message that was transmitted. Note that the ‘TX Time’ and ‘RX Time’ columns correspond to the clock tick at which the message was either transmitted or received, respectively. An example table is shown below for a simulation with 15 messages and a 3x3 mesh:

Message ID	TX Processor ID	TX Time	RX Processor ID	RX Time
0	2	0	5	86
1	4	0	1	40
2	3	0	6	40
3	3	39	0	83
4	7	0	8	50
5	4	38	5	89
6	0	0	7	42
7	3	81	1	127
8	8	0	0	43
9	7	47	3	91
10	1	0	8	91
11	1	76	7	160
12	2	77	7	164
13	1	154	0	199
14	7	87	1	163

Following the simulation, various metrics that measure the performance of the network architecture are also displayed for the user as shown to the right.

```
Average Latency in Clock Cycles: 77.360001
Average Distance in Channels: 4.920000
Average Throughput in Messages/Clock Cycles: 0.104603
Average Speed in Distance/Latency: 0.063599
Total Simulation Time in Clock Cycles: 239
Total Simulation Time in Secs: 0.020329
```

4. **Visualizing Results:** During the simulation, various metrics such as the number of flits transmitted/received by processors, number of failed transmissions due to contention, and buffer efficiency are logged each clock cycle. After the simulation is finished, these metrics are logged to .txt files in the user specified test directory. The user can then visualize these results with ‘data_visualizer.py’ (still need to work a lot more on this interface). An example of the results from a simulation run with 10000 messages on a 10x10 mesh is shown below:



<u>Reassessment of Goals/Deliverables:</u> Given the current time constraints and my progress so far in the project, I have decided to slightly restrict my goals and deliverables that were stated in my proposal.

- In my proposal, I stated that a goal would be to have my simulator could “support multiple topologies (3+), flow control (3+), and routing algorithms (3+)”. However, I will be modifying this goal to instead focus only on a) mesh topology, b) packet store forward and wormhole flow control, and c) dimension-ordered x-y/y-x and adaptive A* routing.

- In my proposal, I stated that a goal would be to do a “complete analysis of 10+ popular/widely-used network designs”. However, given the previous point that I will be restricting myself to mesh networks only, I will only be performing an analysis of the mesh network topology and its associated flow control/routing algorithms.

- In my proposal, I stated that a “nice to have” would be to create a GUI for the user to define the simulation parameters rather than in a config file. However, this is unnecessary work that does not add anything to the simulator's overall significance so I will not be pursuing this.

Revised Goals/Deliverables:

A. Goals I plan to achieve
   a. A fully functioning network simulator that can support a) mesh topology, b) packet store forward and wormhole flow control, and c) dimension-ordered x-y/y-x and adaptive A* routing.
   b. A complete analysis of different mesh-topology based interconnection networks with regards to the following performance metrics: latency, throughput, contention, and buffer efficiency.

B. Goals I hope to achieve
   a. Add more topologies, flow control algorithms, and routing algorithms
   b. A visual heatmap to highlight certain areas of contention/dense traffic in the network as it is being simulated

Final Presentation: I plan to perform a demo of my simulator as well as present several charts and graphs analyzing the performance of various interconnection network architectures I have already simulated. From these charts and graphs, I hope to draw some conclusions regarding what certain aspects of an interconnection network architecture are better suited for certain traffic conditions.

Issues/Concerns: My main concern is being able to extract meaningful information from the visualization of my simulation results. From a quick look at my visualizations, I think that I should change my approach to injecting messages into the network. As of now, each processor just draws from a message queue to inject messages into the network. There is now notion of timed injection/interval injection. Moving forward, I think that I should add in this feature where the processor will not continuously attempt to inject a message but will rather sample its message queue at periodic intervals and then insert. With this approach, I am hoping that my visualizations will show more meaningful information where I am able to correlate certain peaks and valleys in the plots to certain periods of injection. This will also allow me to better understand how certain network architecture react to sudden bursts of traffic rather than a continuous stream of it which will mimic more real-life interconnection networks.