Meeting summary from Monday, 11am-12pm EST for omnet simulation project

Topics covered

- readability of graphs and explaining statistics
- understanding the graphs for the simple o-o-o-o topology case

Overview of simple topology setup:

- nodes: [0] - [1] - [2] - [3] with transactions being sent between node 0 and node 3
- propagation delay on all channels is 30 milliseconds in both directions
- initial balance between nodes 0 and 1 is (10000000,10000000), between node 2 and node 3 is (10000000,10000000), between node 1 and node 2 is (50,50)
- graphs collected:
    - *constant rate* in both directions (transactions/sec): **100, 200, 300, 400, 500**
    - *poisson process* arrival in both directions (transactions/sec): **100, 200, 300**
    - *unbalanced constant rate* in both directions (transactions/sec): **50 and 100, 100 and 200, 300 and 150**
    - *constant rate* in both directions with exponentially distributed transaction size with avg size of 1 (transactions/sec): **200,300**

Statistics collected

- **completionTime (seconds)** - signal emitted when a the sending node has received an acknowledgement that the transUnit has been delivered, computed by calculating: currentTime - timeSent for the transaction
    - without any queuing, completionTime is 180 milliseconds for a transaction
- **total num in queue (number of transactions)** - total number in queue (this statistic is recorded at increments of *statRate = 0.5* in time, where statRate is the frequency at which the nodes report their state)
- **num processed in statRate time** - should change name to "number of processing events that take place", is incremented every time a transaction is removed from incoming or outgoing queue, the counter is reset at every statRate increment

Graph Analysis:

<u>Constant Rate of 200 transactions/second in both directions</u>

- both node 0 and node 3 (overlapping in the graph) have data:
    - num processed in statRate time has constant value 200: send 200 transactions every second and receive 200 transactions every second, which adds up to 400 transactions every second, which is 200 transactions every 0.5 seconds which is what we see on the graph.
        - Note: Since the statRate is 0.5 seconds and we have two data points of 200 at 1.5 and 2.0, that means that a total of 400 units was sent between time t=1.0 and t=2.0
- both node 1 and node 2 (overlapping) have data:
    - num processed in statRate time: since the statistic is incrementing every time we remove a job from incoming or outgoing queue, intermediate (non-endpoint nodes) increment the counter twice by putting a single transaction on the incoming and outgoing queue, so the value of 400 events per 0.5 seconds (which is 800 transactions/second = 2*(200 transaction/second from node 0 + 200 transactions/second from node 3)
    - this is double the values of node 0 and node 3
- completion time: constant at 180 milliseconds is expected, as this is 6*30 milliseconds (channel delay)
- total num in queue: all nodes have 0 transactions in queue

Constant Rate of 400 transactions/second in both directions:

- both node 1 and node 2:
    - total num in queue (overlapping): we see linear queue buildup, as the rate of job arrivals is a constant amount above what the system can support
- both node 0 and node 3:
    - num processed in stat time (overlapping): we see that the number of transactions processed is at a pretty consistent value between 300 and 400, so smaller than the job arrival rate of 400 per second. That value should be the max number of transactions that can be supported in the system.

Exponential rate of 100 transactions per second in both directions

- num processed in statRate time for nodes 0 and 3 are both around 100 as expected
- num processed in statRate time for nodes 1 and 2 are around 200 as expected (look at constant rate analysis for explanation for doubled amount)
- **Question:** even at 100 transactions/sec, we see queue buildup at time 8,9.5 seconds in node 1, however there is no build up at all for node 2. Why? This pattern of only node 1 having a queue build up is even more prevalent in graphs of exponential rate of 200 and 300 transactions per second.

Constant rate of 100 transactions/second in one direction, and 50 transactions/second in the other

- We see an initial spike in num processed in statRate in second 0 to 1 as the middle channel capacity is 100, and the in-balance has not yet locked up all the funds on the side sending more quickly
- The num processed in statRate stays constant at 100 transaction events per second for nodes 0 and node 3, meaning 50 transactions are being processed at each second. This makes sense as the rate 50 trans/sec is the bottleneck for the process.
- The queue builds up for node 1 linearly, there is no queue build up for node 2

Constant rate of 200 transactions/second with an exponentially distributed transaction size with an average of 1

- We see that this results in the same graph as a constant rate with constant size, so the effects of size is secondary to the importance of arrival rate

Next Steps:

- Figure out the reason for the queuing behavior for exponentially sampled arrival times. We plan to do this by:
    - making the statRate value smaller
    - run multiple trials for each graph
    - run graphs with the average transaction rate at 400 and 500 transactions/sec, more than the system can handle
- Simulate the hotnets (figure 4a) graph
- Implement the waterfilling algorithm
- Easy fixes:
    - rename "num processed in statRate" to "num processing events in statRate"
    - change completion time units to be measured in milliseconds
    - change sizes of legend symbols to be able to recognize overlapping lines
    - add axis
        - x: time in seconds
        - y: number of transaction units, milliseconds