In this lab, you will study how efficient management of buffers helps improve network throughput. Given 16 buffers per port, you need to compare the performance of one 16-flit deep FIFO with wormhole flow control versus 16 1-flit deep VCs.

**Useful Definitions:**
- **Reception Rate** (packets/node/cycle) is \( \frac{\text{total_packets_received}}{\text{num-cpus} \times \text{sim-cycles}} \)
- **Peak Throughput.** Maximum reception rate provided by the network.

**Getting Started:**
[clone a fresh copy of gem5]
```
hg clone /nethome/tkrishna3/teaching/simulators/gem5/repo/gem5
```

Now build the simulator.

**THIS NEEDS TO BE DONE EVERYTIME YOU CHANGE THE C++ CODE.**
```
./my_scripts/build_Garnet_standalone.sh
```

On a fresh login to the machines, don’t forget to source the environment file before running
```
source <path_to_gem5>/my_scripts/set_env.sh
```

**Part A: Impact of Number of VCs [15 points]**

**Run Command:**
```
./build/Garnet_standalone/gem5.opt configs/example/garnet_synth_traffic.py \ 
--network=garnet2.0 \ 
--num-cpus=64 \ 
--num-dirs=64 \ 
--topology=Mesh \ 
--mesh-rows=8 \ 
--sim-cycles=10000 \ 
--synthetic=uniform_random \ 
--vcs-per-vnet=16 \ 
--inj-vnet=0 \ 
--routing-algorithm=xy \ 
--garnet-deadlock-threshold = 20000 \ 
--injectionrate=0.02
```

Sweep the injection rate in increments of 0.02 and run the simulation with `--vcs-per-vnet` equal to 1, 2, 4, 8, and 16.

- **Graph 1:** Plot all the latency vs injection rate curves in ONE graph (clearly labeled and y-axis clipped at 50 like Lab 1). Add the graph to your report. [5 points]
- **Graph 2:** Plot the reception rate vs injection rate curves in ONE graph. Add the graph to your report. [4 points]
**Question 1:** What is the theoretical peak throughput for the configuration you ran above? What is the actual observed peak throughput with VC =1? [3 points]

**Question 2:** Looking at Graph 2, you will see a sudden increase in reception rate beyond some number of VCs. Why do you observe this phenomenon? Explain briefly. [3 points]

**Part B: Wormhole Flow Control [25 points]**

Implement Wormhole flow-control within Garnet. It will get enabled when you run Garnet with the option `--wormhole`. This will run Garnet with one VC per port and set the depth of the VC to the value provided within `vcs-per-vnet`. Recall that by default, each VC can only hold one packet. Your wormhole implementation needs to allow it to hold up to 16 packets. *You are only injecting single-flit packets in this lab, so packet => HEAD_TAIL flit.*

**Example Run Command:**

```
./build/Garnet_standalone/gem5.opt configs/example/garnet_synth_traffic.py \
  --network=garnet2.0 \
  --num-cpus=64 \
  --num-dirs=64 \
  --topology=Mesh \
  --mesh-rows=8 \
  --sim-cycles=10000 \
  --synthetic=uniform_random \
  --vcs-per-vnet=16 \
  --inj-vnet=0 \
  --routing-algorithm=xy \
  --garnet-deadlock-threshold = 20000 \
  --injectionrate=0.02 \
  --wormhole
```

You can use any existing datastructures or add new ones.

**Hints:**

- the `decrement_credit` and `increment_credit` functions called by SwitchAllocator.cc will be useful to look at and leverage for your implementation.
- You are only injecting single-flit (HEAD_TAIL_) packets, so only focus on the if (`t_flit->get_type() == HEAD_TAIL_`) parts of the code
Part C: Wormhole vs VCs analysis [10 points]

Graph 1: Plot the latency vs injection rate curve for the following three configurations on the SAME graph for uniform random traffic and add it to the report [3 points]

- VC = 1, Depth = 1 (directly from Part 1)
- VC = 16, Depth = 1 (directly from Part 1)
- VC = 1, Depth = 16 (i.e., --wormhole)

Repeat the same for Tornado traffic. [3 points]

Question 3: For uniform random traffic, how does the performance of wormhole compare to the (VC=16, Depth=1) configuration.? Why? [2 points]

Question 4: For tornado traffic, how does the performance of wormhole compare to the (VC=16, Depth=1) configuration.? Why? [2 points]

What to Submit:
Report.doc/pdf
garnet2.0.tar.gz with your wormhole implementation