Lab 2: Topology Comparison [50 pts]

Goal:
In this lab, you will compare a Mesh (called Mesh_XY in Garnet), Flattened-Butterfly, and a hierarchical ring topology for network performance. The focus of this lab is on design space exploration – you will run a suite of simulations for of these three topologies and plot the results.

Step 0:
Update your gem5 copy
hg pull –u

Now build the simulator. This only needs to be done ONCE (the first time you pull).
./my_scripts/build_Garnet_standalone.sh

Sample Run Command:
./build/Garnet_standalone/gem5.debug
configs/example/garnet_synth_traffic.py \
--network=garnet2.0 \
--num-cpus=16 \ 
--num-dirs=16 \ 
--topology=Mesh_XY \ 
--mesh-rows=4 \ 
--sim-cycles=50000 \ 
--inj-vnet=0 \ 
--router-latency=2 \ 
--injectionrate=0.02 \ 
--synthetic=uniform_random \ 
--link-width-bits=32

The highlighted parameters are what you will be sweeping through in this Lab.
- All experiments will be with a 16 router system.
- Unless otherwise mentioned, all your simulations should be for 50000 cycles.

Traffic Description:
All packets are 64-bits wide.
The number of flits in every packet = (packet_size / link width).
If you change the link widths, the number of flits per packet will go up – this is handled internally within the code and you do not need to worry about it.
The command for changing link-width when you run garnet from the command line is (for e.g.)
--link-width-bits=32

You will run Uniform Random (--synthetic=uniform_random), Tornado (--synthetic=tornado) and Neighbor (--synthetic=neighbor) traffic for all the designs.

The details of each traffic pattern can be seen in src/cpu/testers/garnet_synthetic_traffic/GarnetSyntheticTraffic.cc
How to run Traffic Simulations

Start at a (packet) injection rate of 0.02, and keep incrementing in intervals of 0.02 till the network saturates (i.e., the latency becomes > 100 cycles). In other words, you do not need to run it till a fixed injection rate (like 0.5 in Lab 1) but till the injection rate at which that network saturates. **This is because you will cut off the y-axis off at 100 cycles.**

Network Stats:

`.my_scripts/extract_network_stats.sh` generates `network_stats.txt`. You will be working with `average_packet_latency` and `packets_received` as the stats for this lab.

How to Plot Results

For each (configuration, traffic pattern) pair, you need to plot the *average packet latency vs. injection rate* for all three topologies on the same graph. In other words, each graph in your report will have 3 lines: Mesh, Flattened Butterfly and Hierarchical ring. Note: average packet latency is in cycles. **Make sure to label the axes, and add clear legends to specify which line corresponds to which topology.**

**Step 1: Flattened Butterfly Topology [10 pts]**

**Step 1.1**
Read the Flattened Butterfly paper (Kim et al., “Flattened Butterfly Topology for On-Chip Networks”, MICRO 2007) and implement it in garnet.
- You need to focus just on the topology - don’t worry about the routing and flow-control aspects discussed in the paper.
- You **do not need to implement the concentration factor** (4 nodes connected to one Router) used in the paper. You can assume garnet’s default **one traffic injector per router**.

**Step 1.2**
Create a `FlattenedButterfly.py` file in `$gem5/configs/topologies`

It is a python file. But you do not need to be a python expert to write this.

**Tips: Take a look at Mesh_XY.py for reference.**
- Mesh_XY.py has some print commands to print all the links that are created every time a simulation is run – this will be useful for debugging.
- All links are uni-directional – i.e., **you need to add links in both directions**.
- You will notice a link weight of “1” on the x-links and “2” on the y-links. This is for deadlock avoidance which we will talk about later. Please use the same allocation in the topologies you implement.
- Reuse the `mesh-rows` parameter that Mesh_XY.py uses to specify the number of rows in the Flattened Butterfly topology.
- The router ids used in Mesh_XY code follow the following numbering scheme (0 to 15):
Step 1.3
You can run this topology by specifying `--topology=FlattenedButterfly`
Test your topology using the run command. You can also use the debugging tips on the garnet GT website: [http://tusharkrishna.ece.gatech.edu/teaching/garnet_gt/](http://tusharkrishna.ece.gatech.edu/teaching/garnet_gt/) to make sure the latency and hop values make sense with this topology.

Step 2: Hierarchical Ring Topology [10 pts]

Step 2.1
Implement the following simple hierarchical ring topology. It is built with 4 base rings (shown with red links), and one additional ring to rule them all connecting these rings (shown with blue links). Name this as `HierarchicalRing.py`

Step 2.2
Use the same Tips as Step 1.2 for FlattenedButterfly.
You can assume that this topology will only be called with 16 routers and add links accordingly; you do not have to make it generic.

Step 2.3
Run and test this topology.
Step 3: Performance Simulations and Plots [10 pts]

Configuration A: Equal Link Widths.

Suppose there are no wire constraints. Assume that all three topologies have the same link width: 32b.

**Step 3.1:** For each topology – Mesh_XY, FlattenedButterfly, and HierarchicalRing, plot the average packet latency vs injection rate across all three traffic patterns. [Look at “How to Plot Results” above].

**Step 3.2:** Add these three graphs into a document called Report. Label each graph clearly

Configuration B: Equal Bisection Bandwidth.

Suppose that all three topologies have the same wire area. Assume that the Mesh has 32b links. Scale the link widths in HierarchicalRing and FlattenedButterfly accordingly.

**Step 3.3:** For each topology – Mesh_XY, FlattenedButterfly, and HierarchicalRing, plot the average packet latency vs injection rate across all three traffic patterns. [Look at “How to Plot Results” above].

**Step 3.4:** Add these three graphs into your Report. Label each graph clearly

Step 4: Analysis Questions [20 pts]

Complete Lab2-Questions.docx.

What to Submit:
Create a tarball called Lab2.tar.gz with the following files:
FlattenedButterfly.py
HierarchicalRing.py

Report.doc/pdf
Lab2-Questions.doc/pdf