Interconnection Networks
ECE 8823 A / CS 8803 – ICN
Spring 2017
Lab 2: Topology Comparison

Goal:
In this lab, you will compare a Mesh, Flattened-Butterfly, and a simple-hierarchical 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.

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 \
--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 default link-width is 64-bits => by default only one flit is injected per packet. *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=64`

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

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.

Step 0:
Update your gem5 copy
```
hg pull –u
```

Now build the simulator. This only needs to be done ONCE.
```
./my_scripts/build_Garnet_standalone.sh
```

Step 1: Flattened Butterfly Topology

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.
The paper is on the course website: http://tusharkrishna.ece.gatech.edu/teaching/icn_s17/

- 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.

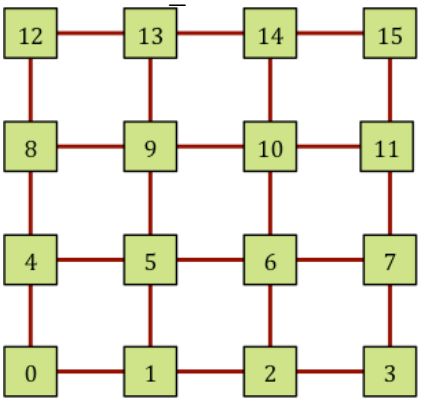
Step 1.2

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 everytime a simulation is run – this will be useful for debugging.
- All links are bi-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.
- 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):



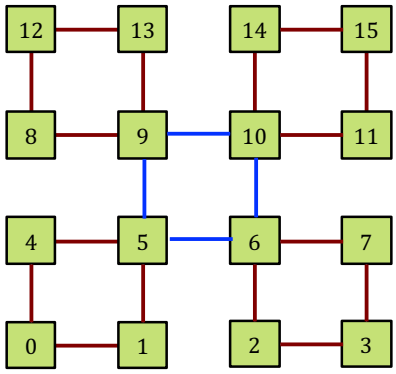
Step 1.3

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/ to make sure the latency and hop values make sense with this topology.

Step 2: Hierarchical Ring Topology

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

Configurations

For each topology – Mesh_XY, FlattenedButterfly, and HierarchicalRing – you will run 2 **configurations**:

(i) equal link-widths: **32-bit links** in all topologies
(ii) equal bisection-bandwidth (in bits): **32-bit links in Mesh**. Link widths in HierarchicalRing and FlattenedButterfly scaled accordingly to equalize the bisection bandwidth.

Traffic Simulations

Step 3.1

Run Uniform Random traffic through the 16-node Mesh, HierarchicalRing, and Flattened Butterfly. Start at an 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.

In excel / google spreadsheet, add 4 columns:
Packet Injection Rate, Mesh Latency, Flattened Butterfly Latency, Hierarchical Ring Latency.
The latency is the average packet latency (cycles).

Plot the latency vs. injection rate for all three topologies on the *same* graph.
Make sure to add clear legends to specify which line corresponds to which topology.

Step 3.2
Repeat the above steps with tornado and neighbor traffic.
Add data for each configuration x traffic in a separate "sheet" in excel / google spreadsheet.

Step IV: Report

You will have 6 plots in total the three 3 patterns x 2 configurations.
Add all 6 plots into a **report**.
For each plot, write down which topology has the lowest low-load latency, and which has the highest throughput.

What to Submit:

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

Results.xlsx : Excel / Google spreadsheet with 6 sheets, one for each (config x traffic)
Report.doc/pdf