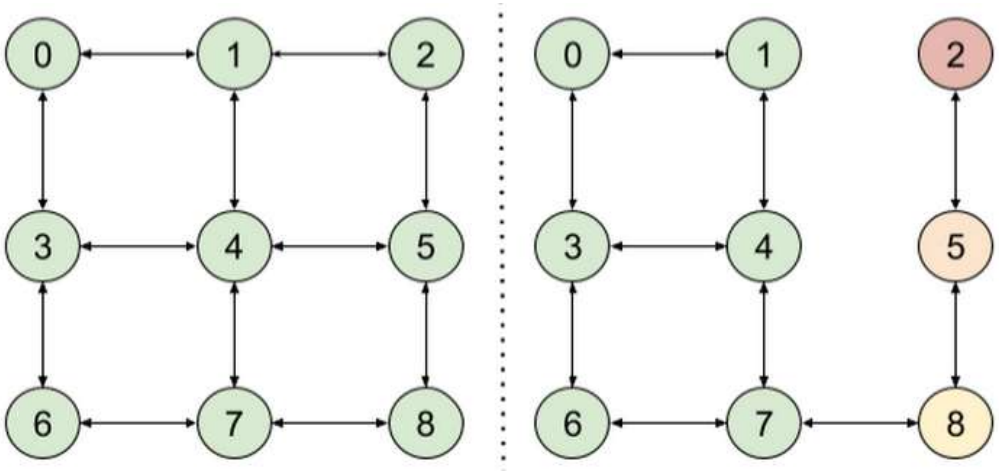
EECS 578 - Checkpoint 3

1. Project Title: Efficient execution of MapReduce applications on irregular NoC topology

2. Team name: HAMM

3. Problem to be addressed:

Hard faults in a network on chip (NoC) interconnect can cause links to break, creating an irregular network topology and potential bottlenecks. Since MapReduce applications create considerable network traffic they are sensitive to the network topology. Nodes communicating on congested links will experience communication delays causing them to finish later than well connected nodes. For example, in figure 1, each node in the regular topology (left) completes the mapping phase of MapReduce at approximately the same time. Conversely, nodes 2, 5, and 8 in the irregular topology (right), might finish much later than other nodes when given the same number of jobs. The overall execution time depends on the node that finishes last. As a result, the total execution time will be prolonged by the slow node.



Regular Topology Irregular Topology

Figure 1. Regular versus Irregular topology

4. Progress:

We gathered results by setting up a 4X4 mesh network on gem5 and running various Phoenix++ MapReduce workloads on it. As some of the workloads (kmeans and PCA) were not properly functioning, we had to fix them in order to run. We measured the characteristics of the regular, baseline network as well as those of the irregular topology shown below. The parameters we selected are:

- Average link utilization - This gives the average number of flits that are present in a network link per cycle. This gives an estimate of the activity on the network links.
- Average network latency - This gives an estimate of the average latency a flit sees in the network.

Our baseline network is a 16-core regular topology that distributes workloads to all nodes equally. To implement our load-balancing algorithm, we broke 5 links of the mesh network in a manner that there are minimal functional links for the network to remain connected (see figure below). We modified the task assignment during the mapping phase by assigning well-connected nodes a larger chunk of data than the others.

We plan to optimize our algorithm by examining the shuffling operation and modifying the hash function.

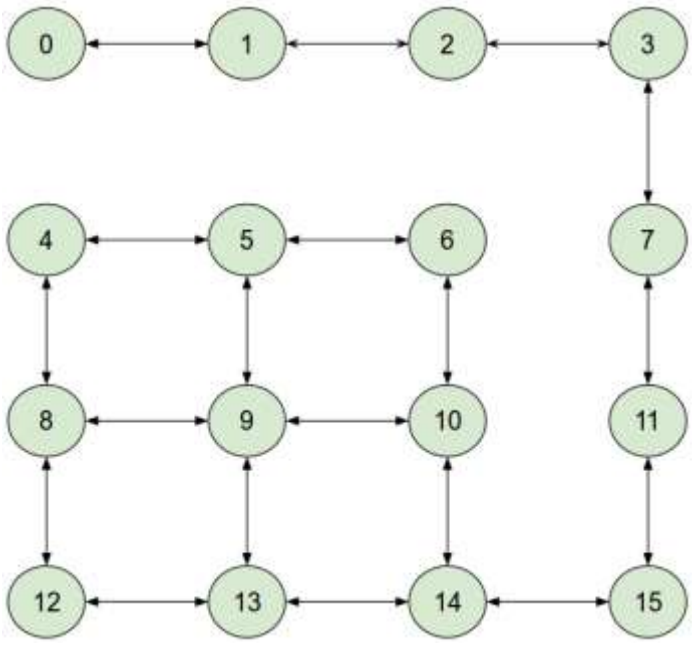


Figure 2. Faulty network

5. Issues:

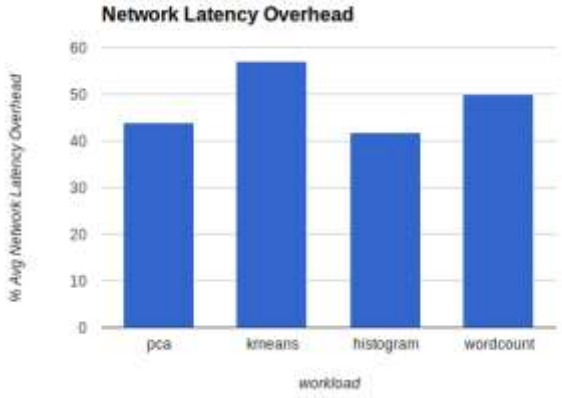
1. The ruby memory system has many coherence bugs; it frequently panics from accessing an unmapped addresses or detects a deadlock. The only way we have been able to

avoid this was by reducing the number of active threads, but we are still looking for a better solution.

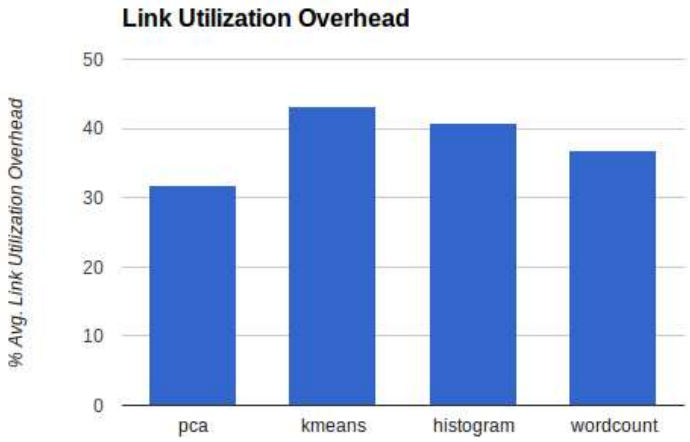
2. The baseline data we gathered shows that the increased network latency from running on a faulty mesh has little impact on the actual runtime of the MapReduce workloads. Additionally, we had a strange result where wordcount ran faster on a network with a faulty mesh.

6. Current Results:

An average network latency overhead of more than 40% was observed in the irregular topology as compared to the fully-connected regular network.



As Expected, the average link utilization in the faulty mesh was significantly higher when compared to a fault-free mesh (>30% increase), showing the potential for congestion in the network.



Furthermore, we evaluated the execution time of each workload in the regular mesh topology and irregular topology. We found out that the the percentage of execution overhead is minimal. pca has the highest overhead, whereas kmeans and histogram show small overhead over our baseline. The execution time of wordcount is even better than the non faulty mesh network, which is not expected. We are still analyzing the variation of execution time slowdown across workloads.