Fault-tolerant & Adaptive Stochastic Routing Algorithm

for Network-on-Chip

Team CoheVer:

Zixin Wang, Rong Xu, Yang Jiao, Tan Bie

Idea & solution to be investigated by the project

There are some options available for the implementation of stochastic routing algorithm, including probabilistic flood, directed flood, random walk, etc. Firstly, we need to analyze the algorithm essence and choose one of them that can be suitably adapted into the NoC system. In order to accommodate adaptive features, the basic stochastic routing algorithm cannot be too complex, otherwise large area or power overhead will emerge. Thus, at current stage, we prefer to utilize random walk based stochastic algorithm due to its advantage in low complexity and relatively small area overhead.

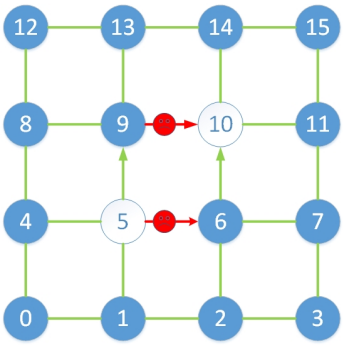
Even though stochastic routing algorithm gains advantage in fault-tolerance, its inherent drawback in performance (due to its oblivious routing strategy) restricts its popularity in NoC system. So we plan to add some adaptive features to our implementation of stochastic routing algorithm for high performance. Specifically, some modules that are responsible for collecting and analyzing real-time data of NoC will be added, by comparing probability of each route, the optimal route can be determined. Of course, adaptive features cannot do harm to fault-tolerance, some evaluations will be conducted not only for the performance improvement, but for its reliability.

In all, we plan to develop a fault-tolerant & adaptive stochastic routing algorithm for NoC in our project.

Progress so far:

1. A basic random-walk routing algorithm has been implemented in C++ and added into the 'routefunc.cpp' in Booksim.
   - For the routing algorithm we designed, dimension-order will be used if there is no fault in the network to achieve a smaller latency. While a packet enters a router with faulty links, the router will change to use random-walk routing and randomly chose other ports to avoid faulty links.

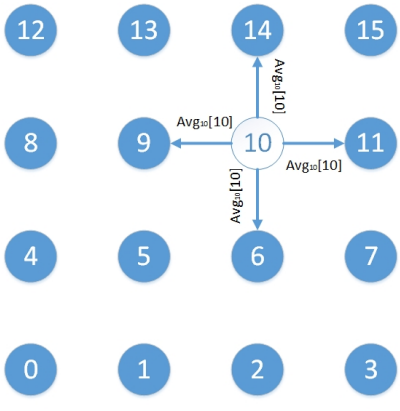
2. Inserting fault tables into the Booksim to simulate a network with permanent link faults. So we could verify our routing algorithm in a faulty environment.



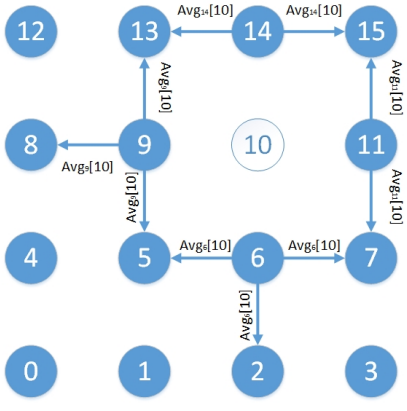
Here we use a 2D vector to construct a faulty network and import to Booksim

3. Based on the simulation results, we found that the average latency of this random-walk routing was too long and the performance wasn't satisfying. After the discussion with GSI, we decided to modified our baseline design.

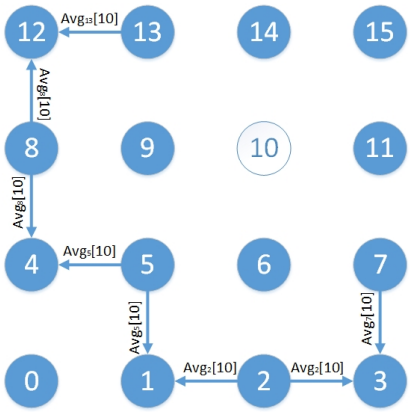
4. An adaptive feature has been added:
   - we implemented a global congestion table to store the estimated delay from each node to every possible destination in the network. The global congestion table is destination-based and upgraded periodically.
   - Then by using this table, we could generate port-selection ratio based on destinations for each router, which indicate the corresponding path delays. And all packets destined for the same destination are distributed in the same ratio to the downstream routers so that balancing the network congestion and achieving a smaller overall transmission latency.
   - The delay measurement and propagation is shown below.

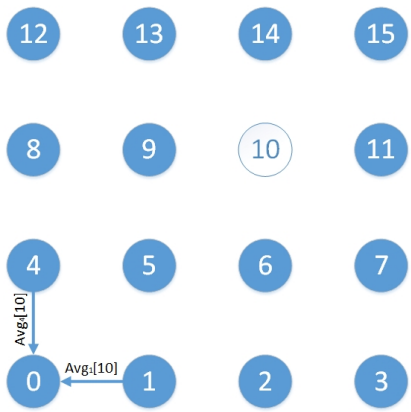


Time 1



Time 2





- And here is a part of the pseudo-code we implemented.

```
foreach i = 1:gNodes; //Destination
    foreach j = 1:2(gK-1);
        foreach k = 1:gNodes; //Source
            if(abs(node[i].x-node[k].x)+abs(node[i].y-node[k].y) == j)
                update_average[k];
                update_w[k];
            endif
        end
    end
end
```

Issues/showstoppers:

1. Right now, our global congestion table will update immediately, which don't stand for the real situation in the hardware execution and we are studying how to update the estimated delay information by clock cycles inside Booksim because the timing mechanism of Booksim is still unclear.
2. Due to the limitation of Booksim, a head flit is unable to send separately in order to generate probability information, which bring some difficulties for a stochastic design. Thus a more efficient fault-tolerant routing feature are still under implementing and we are also still discussing a proper method to combine it with the current adaptive feature.
3. Deadlock recovery mechanism are still under studying.

Further Works:

1. Implement an efficient fault-tolerant feature and combine with the current adaptive routing algorithm
2. Verify the routing algorithm with/without link fault in the network to evaluate the performance in latency & fault-tolerance.
3. Further analysis of the proposed design such as power consumptions.