Reproducing GOAL Algorithm on 3D Torus Network

Ziren Wang*
wang-zr22@mails.tsinghua.edu.cn
IIIS, Tsinghua
Haidian District, Beijing

Chuan Liu
liuchuan22@mails.tsinghua.edu.cn
IIIS, Tsinghua
Haidian District, Beijing

ABSTRACT

We reproduce the current SOTA algorithm for torus networks, GOAL - Globally Oblivious Adaptive Locally [Singh et al. 2003]. GOAL is a load balanced adaptive routing algorithm that provides high throughput on adversarial traffic patterns. GOAL also preserves locality to provide up to 4.6x the throughput of fully randomized routing on local traffic. GOAL achieves global load balance by randomly choosing the direction to route in each dimension and therefore achieves local load balance by routing adaptively.

CCS CONCEPTS

• **Networks → Packet scheduling**.

1 INTRODUCTION

Torus or *k-ary n-cube* is a network topology that is widely used in parallel computing. Torus network has high path diversity, therefore offering many alternative paths between a message source and destination. This property makes torus network a popular candidate for designing routing algorithms [Jerger et al. 2022].

A good routing algorithm must strike a balance between the need of providing low local latency and maintaining high throughput under specific adversarial traffic patterns. It is known that fully randomized routing can achieve global load balance, but it suffers from high expected latency. On the other hand, shortest path routing algorithms can provide low latency for local traffic, but they are fragile to some traffic patterns that load some links very heavily.

Previous work has attempted to address the issue of providing high worst-case performance while preserving locality. Valiant's randomized algorithm [Valiant 1982] gives good performance on worst-case traffic, but at the expense of completely destroying locality and hence giving very poor performance on local traffic and greatly increasing latency. Minimal adaptive routing [Berman et al. 1992] employs randomization to misroute from a shared queue of packets in each node when the network becomes congested. However, the misrouting decision is very localized and does not address the global load imbalance caused by adversarial traffic patterns.

The GOAL algorithm exceeds minimal routing by more than 40% on adversarial traffic patterns, while exploits locality giving 4.6x the throughput of Valiant on local traffic and more than 30% lower zero-load latency than Valiant on uniform traffic. In the remaining parts of the report we will introduce how we reproduce the GOAL algorithm and evaluate its performance on 3D torus network under *gem5* framework. We compare the results with three baselines, the *default TABLE* algorithm, the *Dor routing* algorithm and the *minimal adaptive* algorithm.

*Both authors contributed equally to this project.

2 PRELIMINARIES

In this part, We will introduce some definitions and basic concepts that are used in the routing algorithms.

2.1 Torus Network

A k-ary n-cube is a n dimensional torus network with k nodes per dimension giving a total of $N = k^n$ nodes. Each link is unidirectional, hence there are two links between adjacent nodes - one for each direction. A packet may be divided into fixed size units called flits. Each link has unit capacity, i.e. it can forward one flit per cycle.

In all the algorithm except, we employ virtual channels for flow control. VCs are mechanism that allows multiple independent flits to share the same physical communication link without interfering with each other.

2.2 Routing Algorithms

A routing algorithm R maps a source-destination pair to a path through the network from the source to the destination.

Oblivious algorithms select the path using only the identity of the source and destination nodes. *Adaptive* algorithms may also base routing decisions on the state of the network. Both oblivious and adaptive algorithms may use randomization to select among alternative paths. *Minimal* algorithms only route packets along a shortest path from source to destination while *non-minimal* ones may route packets along longer paths.

3 GOAL

3.1 Routing Algorithm

GOAL algorithm decides the routing direction by obliviously selecting a quadrant in which a packet will be routed in a manner that balances load among the quadrants. The algorithm is described in Algorithm 1.

Algorithm 1 Selecting quadrant

Input: Source node s, destination node d
Output: direction vector r
$x_s, y_s, z_s \leftarrow$ coordinates of s
$x_d, y_d, z_d \leftarrow$ coordinates of d
$\delta_x = \min(x_d - x_s, x_s - x_d)$
$p_x^+ = 1 - \delta_x / k$
$r_x \sim \text{Bernoulli}(p_x^+)$
r_y and r_z are updated similarly

Then the package will be adaptively routed within the selected quadrant from source to destination. We introduce the term *productive dimension* for the package to find its path while avoiding crowded hops. The algorithm is described in Algorithm 2.

Algorithm 2 Adaptive routing within quadrant

Input: Source node s, destination node d, direction vector r
Output: path from s to d
$x_d, y_d, z_d \leftarrow$ coordinates of d
$x, y, z \leftarrow x_s, y_s, z_s$
$path \leftarrow \emptyset$
while $(x, y, z) \neq (x_d, y_d, z_d)$ **do**
  calculate productive dimensions, i.e. dimensions that are not equal to the destination
  choose the hop that has shorter queue length in the productive dimensions
  $path \leftarrow path +$ next hop
end while
return $path$

3.2 Flow Control

Our implementation of GOAL employs 4 virtual channels per physical channel to achieve deadlock freedom in the network. We group the 4 VCs into 3 pairs, $*_0$, $*_1$, and *non-*$*$.

The *non-*$*$ channels are fully adaptive and can be used at any time, while $*$ can only be used when the packets are traversing the most significant productive dimension. $*$ channels may have deadlocks regarding to adversarial traffic, therefore we further divide the $*$ channels into two sub-channels, $*_0$ and $*_1$. Packets will move through $*_1$ when crossing a wrap-around edge otherwise they will move through $*_0$. With these constraints it can be seen that the channel dependency graph for the $*$-channels associated with GOAL is acyclic.

4 IMPLEMENTATION DETAILS

For 3D torus topology, we implement a general torus network, you can specify the number of nodes in each dimension because the implementation is an iteration method.

First RoutingAlgorithm and FlowControl, we implement our routing algorithms at *RoutingUnit* in *gem5*. The *RoutingUnit* is responsible for selecting the next hop for a packet. We implement the GOAL algorithm in the *RoutingUnit* and we also implement the *Dor routing* and the *minimal adaptive* algorithm for comparison. For the GOAL Routing algorithm, we implement the *count-free-vc()* in *OutputUnit* to count the number of free VCs in the next possible outports.

The specific flow control of GOAL is below: we separate the VCs into 3 groups, $*_0$, $*_1$, and *non-*$*$. For a flit, we'll maintain a *is-torus-dims-checkpoint* vector to record whether the flit has crossed a wrap-around edge in each dimension, which is updated when the outport of the flit is determined.

Consider there is a certain outport of the current flit, it will first check the *non-*$*$ channel, if there is a free VC, it will use it. Otherwise, it will check the $*$ channel, which one it choose is based on whether it had crossed a wrap-around edge. If it had, it will use the $*_1$ channel, otherwise it will use the $*_0$ channel.

Especially, For GOAL, we change the time of using *outportCompute()* function from *InputUnit.wakeup()* to *SwitchAllocator.arbitrate-inports()*, this movement can make sure that the flit will choose the outport based on the current state of the network and the deadlock freedom of the network.

5 EXPERIMENTS SETTING

For almost all experiments, we use a 4-ary 3-cube torus network with 64 nodes. The network is configured with 4 virtual channels per physical channel and the sim-cycles is 20000.

When we evaluate the performance of torus topology, we use the *Ring* and *Mesh* topology as the baselines. The routing algorithm is set to be the *default TABLE* algorithm.

In the evaluation of the GOAL algorithm, we carefully select three baselines, the *default TABLE* algorithm, the *Dor routing* algorithm and the *minimal adaptive* algorithm and we briefly introduce them below.

- *default TABLE* is a minimal routing algorithm that is implemented in *gem5*. It does not include any flow control.
- *Dor routing* is a minimal routing algorithm that routes packets along the shortest path from dimension X to Z. We expand 2 VCs for each physical channel, which one is chosen is based on the wrap-around edge acrossing.
- *minimal adaptive* first decides the closest quadrant to the destination and then adaptively routes the packet within the quadrant. We expand 3 VCs for each physical channel.

Notice that we do not include the *Valiant's* algorithm in our evaluation as it has been proved that Valiant's provides optimal worst-case throughput but performs identically on every admissible destination matrix on k-ary n-cubes (see **Claim 2** in the original paper [Singh et al. 2003]).

The traffic patterns we used are *uniform random*, *torus tornado*, *torus transpose*, *shuffle*, *bit reverse* and *bit rotation*.

6 RESULTS AND ANALYSIS

For the evaluation of the Torus topology, we use the *Ring* and *Mesh* topology as the baselines.

Figure 1 shows the reception rates of the three topologies on uniform random traffic. We can see that the ring topology performs the worst, which gets deadlock when the injection rate goes to 0.18. The mesh topology performs better than the ring topology, but it also gets conjestion when the injection rate goes to 0.66. The torus topology performs the best, which still has a healthy throughput even the injection rate is high. The reason is that the complexity of these three topologies increase progressively, the torus topology has more alternative paths between a message source and destination, which can provide many alternative paths for the message to avoid the congestion.

In the figure 2, the hops of these three topologies are shown. The decreasing trend of the hops for ring topology is because the deadlock happens, which causes the packets to be dropped. The hops of the mesh topology and the torus topology are stable, and the torus topology has a lower hops than the mesh topology, which is because the torus topology has more alternative paths between a message source and destination.

Figure 3 shows the saturation throughput of the four algorithms on a 4-ary 3-cube torus network on each traffic pattern. We can see that the Dor routing algorithm performs the worst except the *torus tornado* traffic patterns. According to the original paper, on the

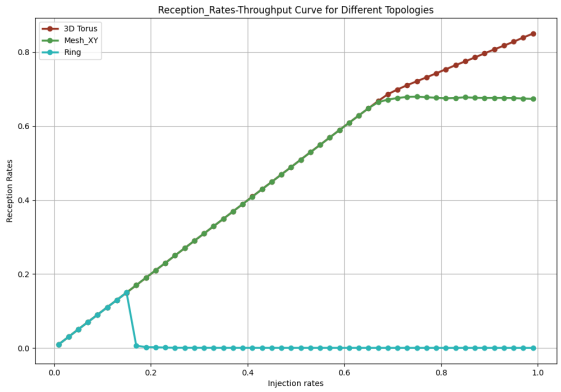

Figure 1: Throughput of different topologies for 64 nodes on uniform random traffic.

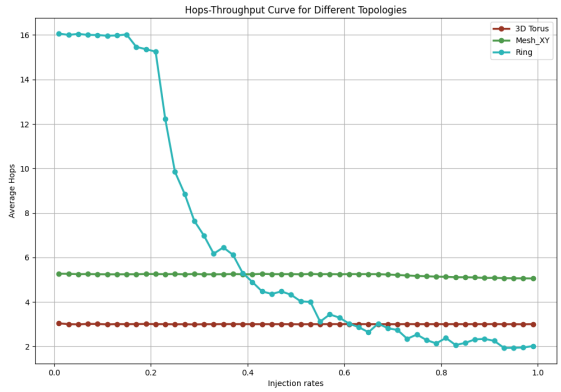

Figure 2: Average Hops of different topologies for 64 nodes on uniform random traffic.

benign traffic patterns, such as *uniform random* and *nearest neighbor* which is not reproducing here, the minimal adaptive algorithm and the GOAL algorithm does not perform very well. But for the adversarial traffic patterns, such as *torus transpose* and *shuffle*, the GOAL algorithm and MIN AD algorithm perform better than other baselines. It may be because for the adversarial traffic patterns, the shortest paths are not the best paths, the balance of the load becomes more important, which is the advantage of the GOAL algorithm.

Figure 4 and Figure 5 show the latency of the four algorithms on *bit rotation* and *torus tornado* traffic patterns respectively. In the beginning, the minimal algorithms perform the best, while the GOAL algorithm performs the worst because when in low load, these minimal algorithms goes the shortest path, while the GOAL algorithm might go a longer path. But when the injection rates are increasing, the GOAL algorithm performs more stable than other algorithms. The reason is that the GOAL algorithm can balance

Figure 3: Comparison of saturation throughput of four algorithms on an 4-ary 3-cube torus network.

the load among the quadrants and reduce the congestion in the network. Actually, this is a common situation, the other figures will attached in the appendix.

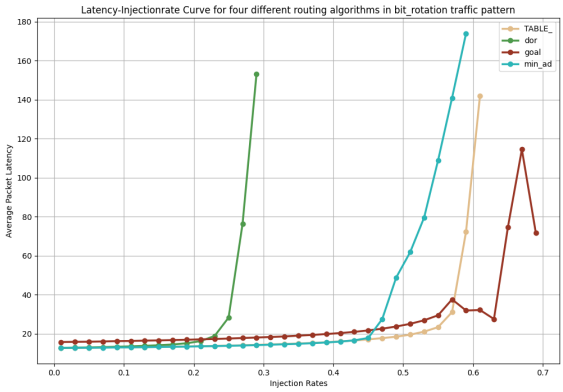

Figure 4: Average Latency of different algorithms on bit rotation traffic.

Figure 6 shows the throughput of the four algorithms on *torus transpose* traffic patterns. We can observe that the Dor routing algorithm gets conjestion first, and then the default TABLE algorithm gets some conjestion or deadlock when the injection rate goes to near 1. The minimal adaptive algorithm and the GOAL algorithm perform similarly, and the minimal adaptive algorithm performs a little better than the GOAL algorithm. The possible reason is that the minimal adaptive algorithm can still handle the conjestion when the injection rate is 1. The advantage of the GOAL algorithm is the stability of throughput but the low load performance is not as good as the minimal adaptive algorithm.

7 CONCLUSION

In this report, we reproduce the GOAL algorithm on 3D torus network and evaluate its performance on *gem5* framework.

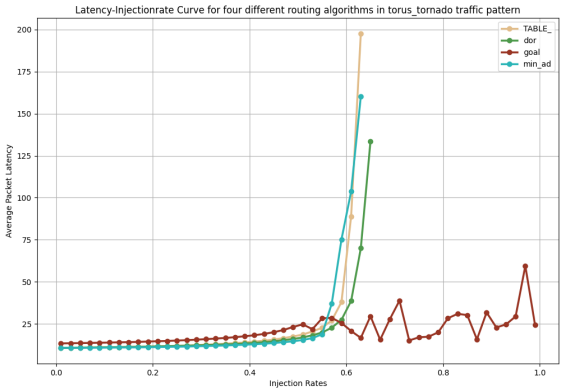

Figure 5: Average Latency of different algorithms on torus tornado traffic.

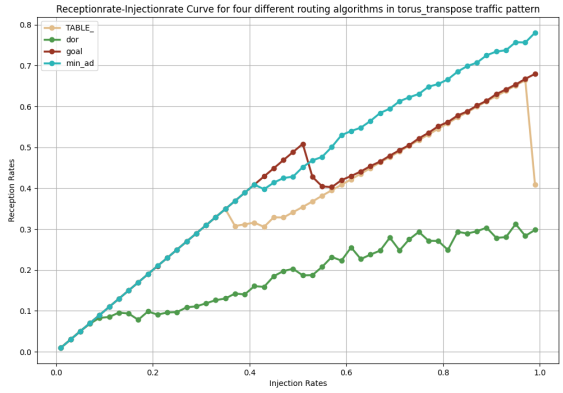

Figure 6: Reception Rates of different algorithms on torus transpose traffic.

The torus topology has a better performance than the ring and mesh topology because of its high path diversity, specifically, it has lower hops, higher and more stable throughput.

The GOAL algorithm can balance the load among the quadrants selection and reduce the congestion in the network, which makes it perform more stable than other algorithms when the injection rates are increasing. But correspondingly, this algorithm sacrifices the low load performance.

In this experiments setting, a part of the experiments results don't shows the significant differences among GOAL and other algorithms, but the GOAL algorithm still has its advantages in some traffic patterns theoretically.

In the future, we can adjust more tough experiments setting to catch out the advantages of the GOAL algorithm.

8 DIVISION OF LABOR

- Ziren Wang: Implementation of the Torus topology, GOAL routing algorithm, FlowControl and the evaluation of the performance of the Torus topology and GOAL. Chapter 4- of report. PPT presentation.
- Chuan Liu: Implementation of Dimension Order Routing (Dor) baseline. the evaluation of the performance of the Torus topology and GOAL. Chapter 1-3 of report. PPT producing and PPT presentation.

REFERENCES

Pablo E Berman, Luis Gravano, Gustavo D Pifarre, and Jorge LC Sanz. 1992. Adaptive deadlock-and livelock-free routing with all minimal paths in torus networks. In *Proceedings of the fourth annual ACM symposium on Parallel algorithms and architectures*. 3–12.

Natalie Enright Jerger, Tushar Krishna, and Li-Shiuan Peh. 2022. *On-chip networks*. Springer Nature.

Arjun Singh, William J Dally, Amit K Gupta, and Brian Towles. 2003. GOAL: a load-balanced adaptive routing algorithm for torus networks. *ACM SIGARCH Computer Architecture News* 31, 2 (2003), 194–205.

Leslie G. Valiant. 1982. A scheme for fast parallel communication. *SIAM journal on computing* 11, 2 (1982), 350–361.