Technology-Driven, Highly-Scalable Dragonfly Topology

- Kim, John, et al. "Technology-driven, highly-scalable dragonfly topology." 2008 International Symposium on Computer Architecture. IEEE, 2008.

Motivation:

Interconnection networks are a critical component of modern computer systems and significantly impact the overall performance and cost of the system.

Evolving technology and increasing pin-bandwidth motivate the use of high-radix routers to reduce the diameter, latency, and cost of interconnection networks.

require longer cables

Cables = cost

→ minimize global cables to realize efficient network.

Contribution:

introduce the dragonfly topology

group of high-radix routers as a virtual router
increase the effective radix of the network

reduce global channels: each minimally routed packet traverses at most one global channel

Increase physical length of the global channels (favor optical signaling tech)

dragonfly reduces cost (≥ 16K nodes):
20% compared to a flattened butterfly
52% compared to a folded Clos network

High-radix networks reduce the diameter of the network
but require longer cables compared to low-radix networks.

Global cables are between cabinet.
Cabinet >> backplanes >> circuit board >> routers

optical

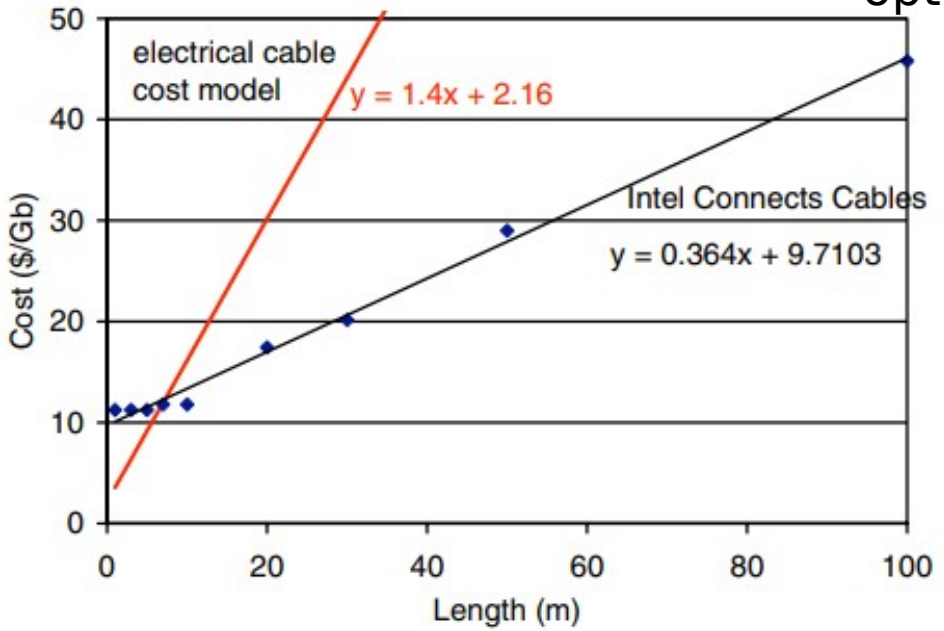


Figure 2. Cost model comparison of active optical cables [12] and electrical cables with repeaters [14].

Contribution:

introduce two new variants of global adaptive routing that enable load-balanced routing in the dragonfly.

Each router make an adaptive routing decision
based on the state of a global channel.

Introduce
1. the use of selective virtual-channel discrimination:
(eliminates bandwidth degradation)

2. the use of credit round-trip latency to both sense and signal channel congestion: (eliminates latency degradation)

The combination of 1 and 2 gives ideal throughput and latency.

Dragonfly Topology:
The dragonfly is a hierarchical network with three levels: router, group, and system

each router has connections to **p** terminals, **a − 1** local channels — to other routers in the same group — and **h** global channels — to routers in other groups.

A group consists of **a** routers
Each group has **ap** connections to terminals and **ah** connections to global channels
all of the routers in a group collectively act as a virtual router with radix **k' = a(p + h)**

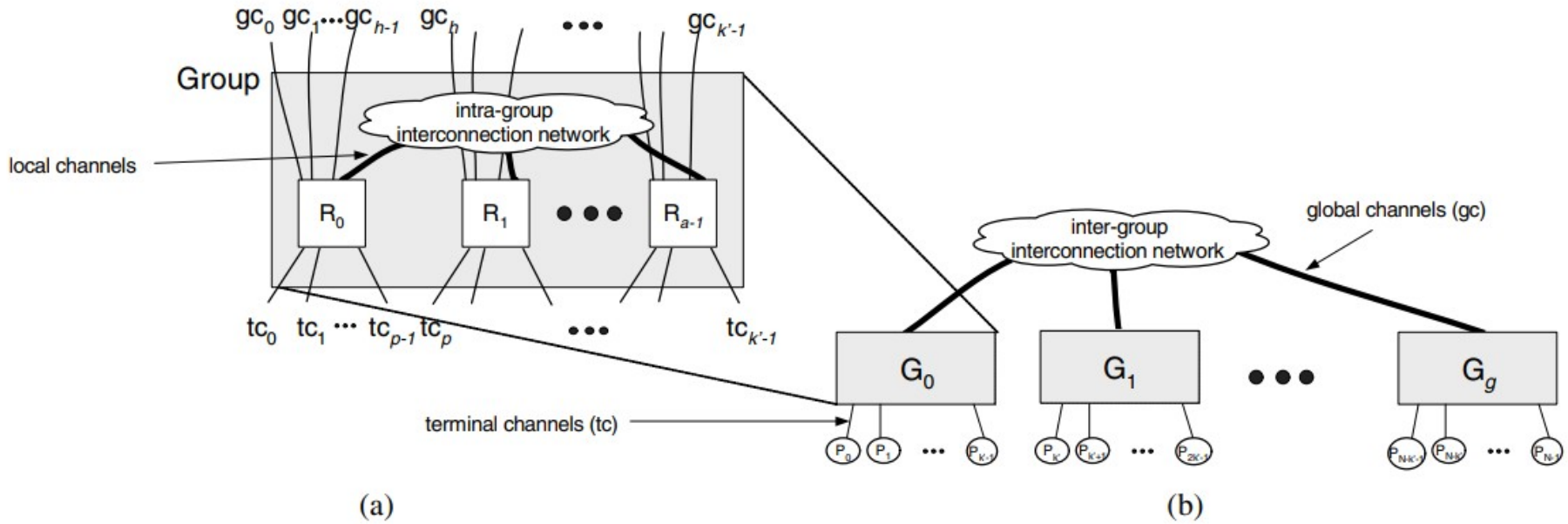


Figure 3. (a) Block diagram of a group (virtual router) and (b) high-level block diagram of a dragonfly topology composed of multiple groups. gc_i corresponds to global channels for inter-group connections and tc_i corresponds to channels connected to the terminals (or processors).

Dragonfly Topology:

Figure 4 shows good scalability with increasing router radix (k).

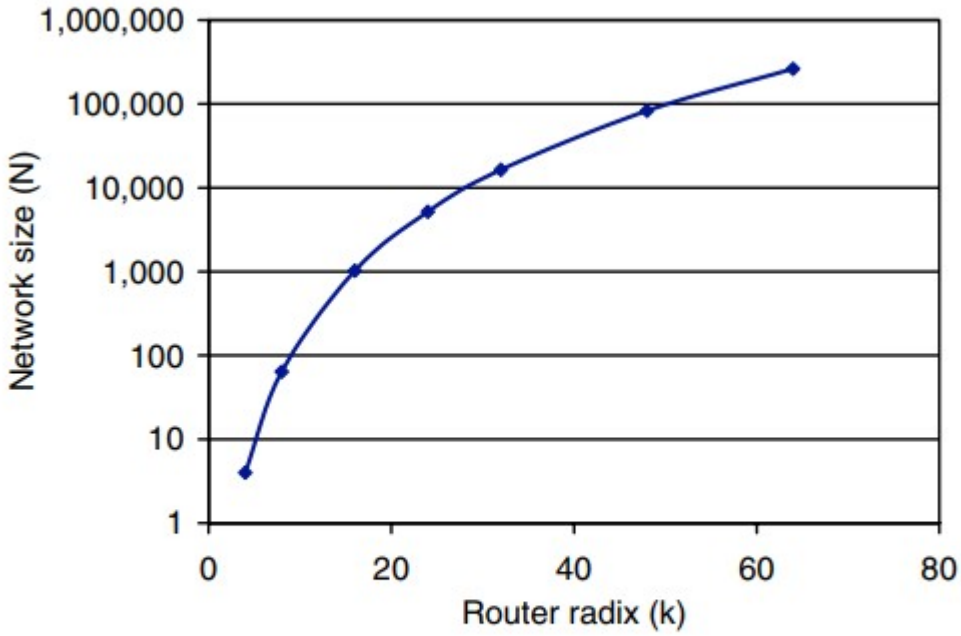


Figure 4. Scalability of the dragonfly topology as the router radix (k) is increased.

Dragonfly Variations:

Fig5 shows normal Dragonfly with 72 network terminals.

Fig6(a): has same effective radix, provide more bandwidth to local routers → Exploit packaging locality.

Fig6(b): increases the number of dimensions within the group → Increased group radix.

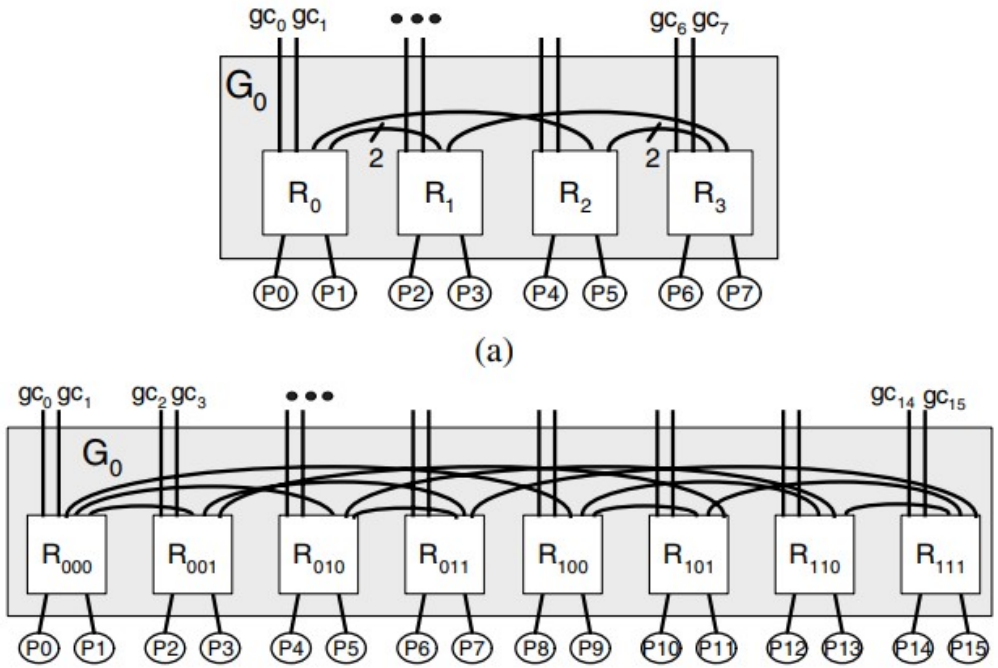


Figure 6. Alternative organization of a group in dragonfly. (a) The same group radix is maintained as in Figure 5 but packaging locality is exploited by providing more bandwidth to the neighboring routers. (b) Increasing group radix by increasing the number of dimensions within the group. The routers within the group are connected by a 3-D flattened butterfly. With $p = 2$, the resulting 3-D flattened butterfly is equivalent to a simple 3D cube.

V.S

.

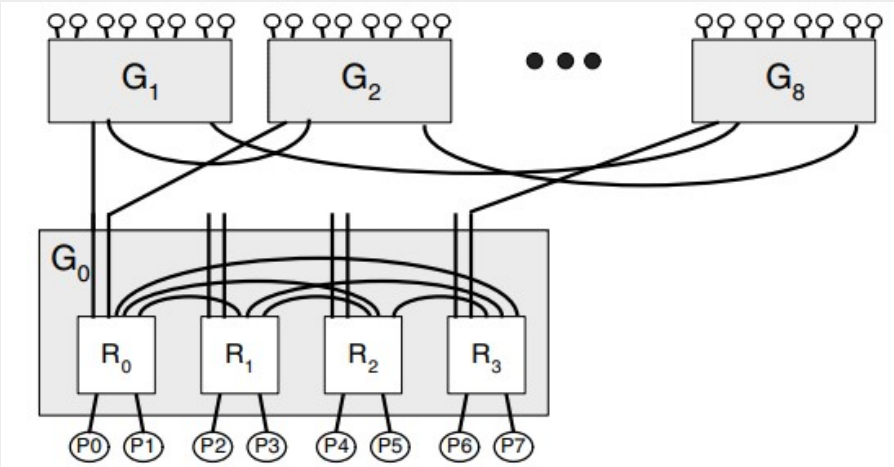


Figure 5. An example block diagram of a dragonfly topology with $N = 72$.

Routing:

global adaptive routing using local information leads to limited throughput and very high latency at intermediate loads.

→ not suit dragonfly

→ propose new mechanisms to ideal implementation of global adaptive routing.

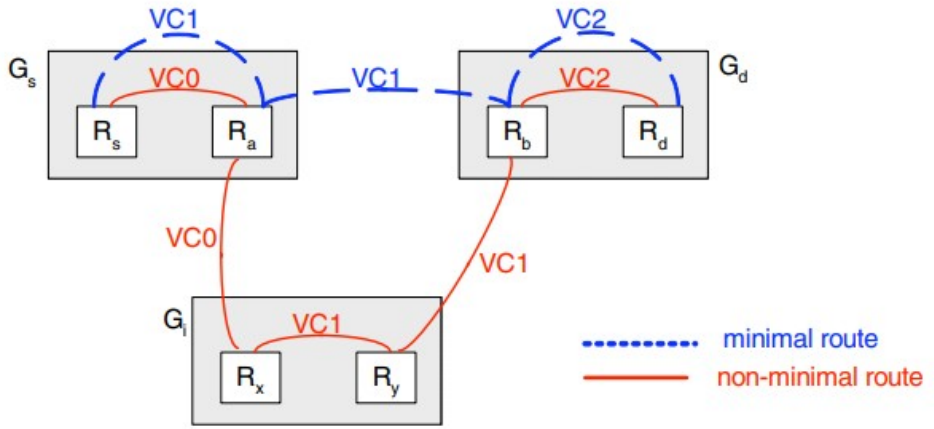


Figure 7. Virtual channel assignment to prevent routing deadlock in a dragonfly topology with both minimal and nonminimal routing.

Applying Valiant's algorithm to groups suffices to balance load on both the global and local Channels. (Randomized non-minimal routing)

Minimal routing:

Step 1 : If $G_s \neq G_d$ and R_s does not have a connection to G_d, route within G_s from R_s to R_a, a router that has a global channel to G_d.

Step 2 : If $G_s \neq G_d$, traverse the global channel from R_a to reach router R_b in G_d.

Step 3 : If $R_b \neq R_d$, route within G_d from R_b to R_d.

Step 1 : If $G_s \neq G_i$ and R_s does not have a connection to G_i, route within G_s from R_s to R_a, a router that has a global channel to G_i.

Step 2 : If $G_s \neq G_i$ traverse the global channel from R_a to reach router R_x in G_i.

Step 3 : If $G_i \neq G_d$ and R_x does not have a connection to G_d, route within G_i from R_x to R_y, a router that has a global channel to G_d.

Step 4 : If $G_i \neq G_d$, traverse the global channel from R_y to router R_b in G_d.

Step 5 : If $R_b \neq R_d$, route within G_d from R_b to R_d.

Evaluation:

MIN: minimal routing
VAL: randomized non-minimal routing
UGAL: Universal Globally-Adaptive load-balanced
UGAL-L: UGAL with using local queue information at current router node
UGAL-G: UGAL with using queue information for all global channels.

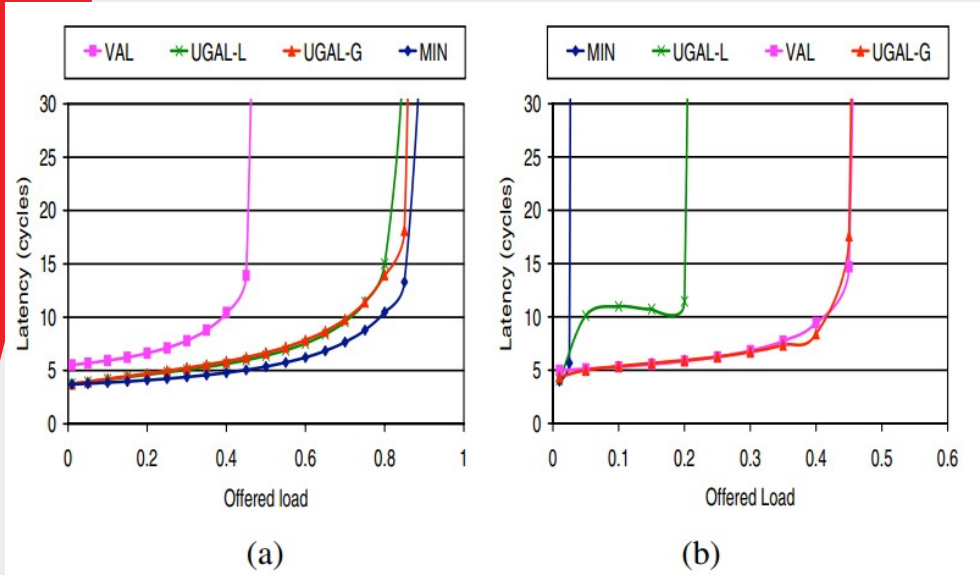


Figure 8. Routing algorithm comparison on the dragonfly for (a) uniform random traffic and (b) adversarial traffic pattern.

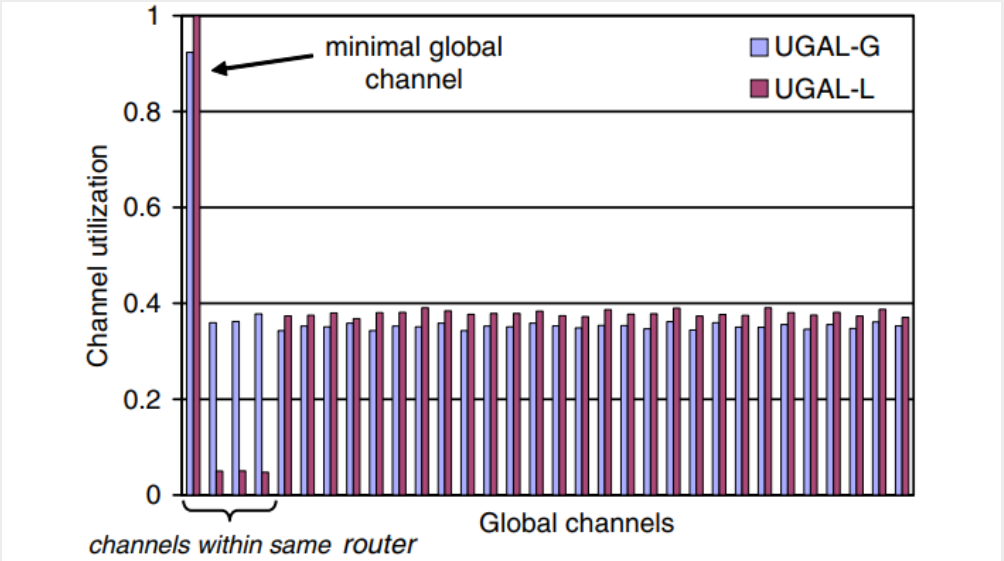


Figure 9. Global channel utilization for the dragonfly topology with UGAL-L and UGAL-G routing using the adversarial traffic pattern. The data is collected from an offered load of 0.2, just prior to the saturation of UGAL-L.

This minimal routing works well for load-balanced traffic, but results in very poor performance on adversarial traffic patterns.

Found problem 1: limited throughput

The throughput issue with UGAL-L is due to a single local channel handling both minimal and non-minimal traffic.

Solution:

modify the UGAL algorithm to separate the queue occupancy into minimal and nonminimal components by using individual virtual channels (UGAL-L_VC)

```
if   (q_m_vc H_m ≤ q_nm_vc H_nm )
     route minimally;
else
     route nonminimally;
```

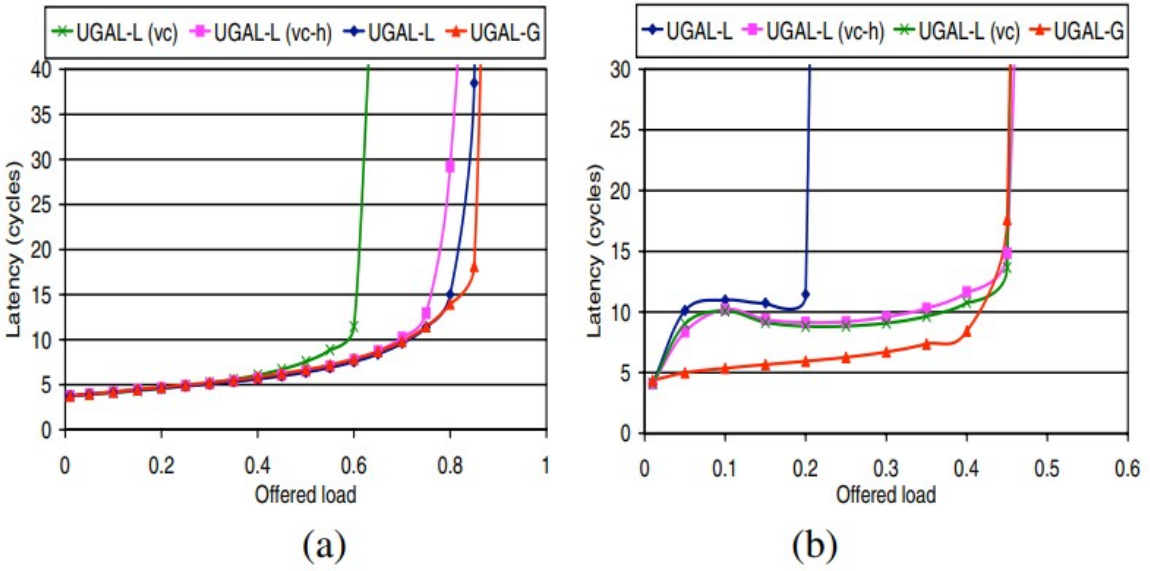


Figure 10. Evaluation of alternative UGAL-L implementation for (a) uniform random traffic and (b) worst-case traffic.

Problem 1.1:
for load-balanced traffic when most traffic should be sent minimally, individual VCs do not provide an accurate representation of the channel congestion – resulting in throughput Degradation

Solution:
modify the UGAL algorithm to separate the queue occupancy into minimal and non-minimal components only when the minimal and nonminimal paths start with the same output port.
→ Hybrid modified UGAL routing algorithm (UGAL-L_VC_H)

```
if  (q_m H_m ≤ q_nm H_nm && Out_m ≠ Out_nm )  ||
   (q_m_vc H_m ≤ q_nm_vc H_nm && Out_m = Out_nm )
     route minimally;
else
     route nonminimally;
```

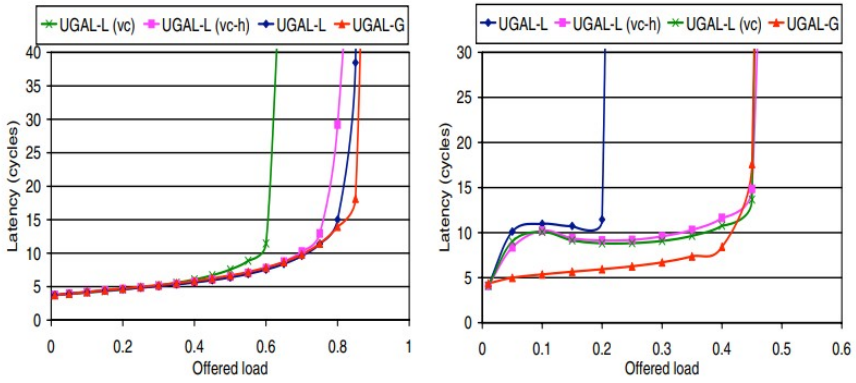


Figure 10. Evaluation of alternative UGAL-L implementation for (a) uniform random traffic and (b) worst-case traffic.

Problem 2: Higher intermediate latency

The high intermediate latency of UGAL-L is due to minimally-routed packets having to fill the channel buffers between the source and the point of congestion before congestion is sensed

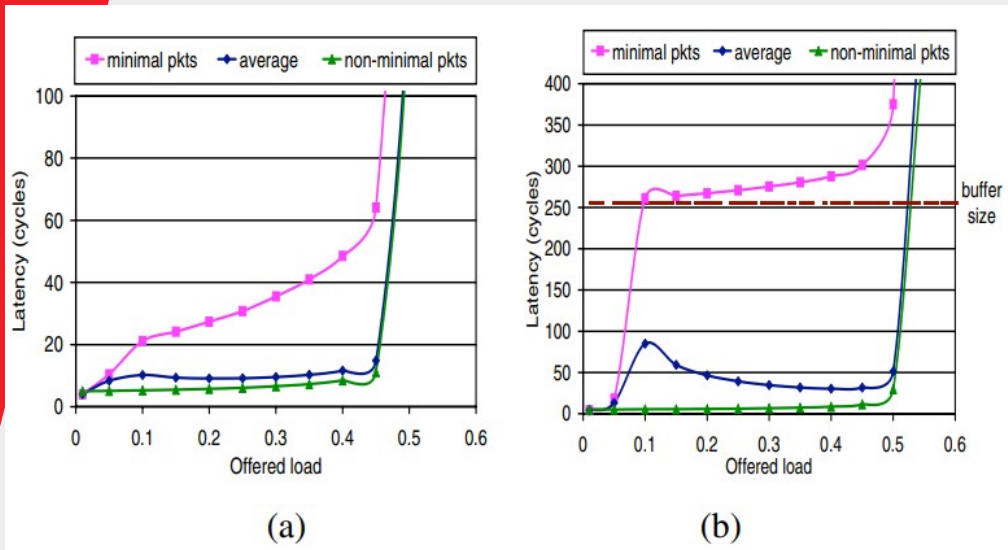


Figure 11. Latency vs. offered load for the dragonfly topology with UGAL-L routing and adversarial traffic pattern with the input buffers of depth (a) 16 and (b) 256.

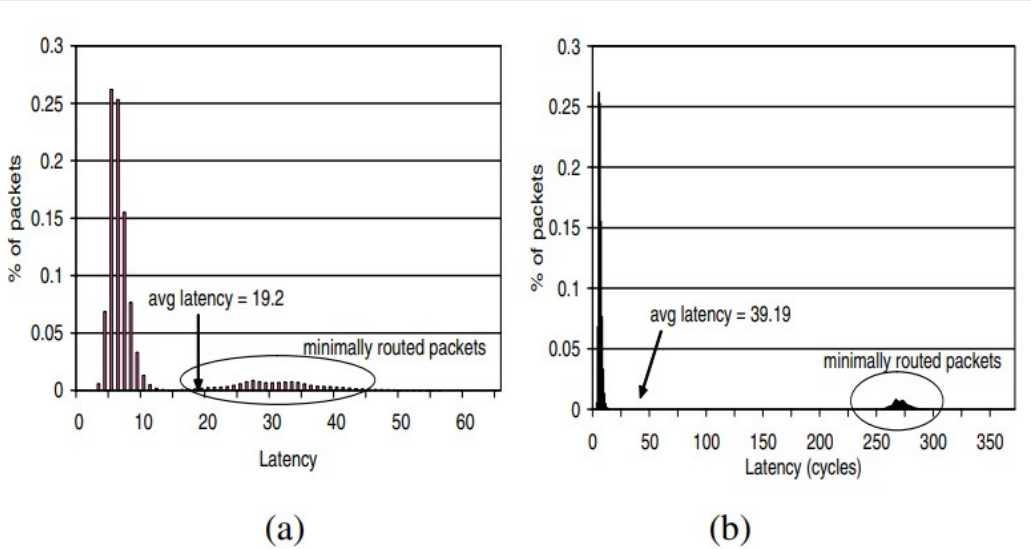


Figure 12. Histogram distribution of average packet latency at an offered load of 0.25 in the dragonfly topology with UGAL-L routing adversarial traffic pattern and the input buffers of depth (a) 16 and (b) 256.

Problem 2's cause

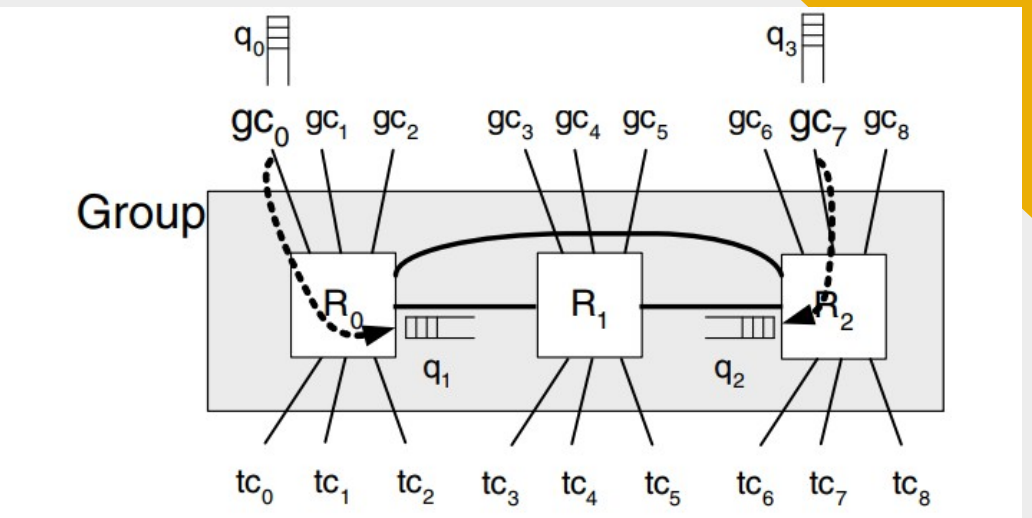


Figure 13. A block diagram of a dragonfly topology to illustrate indirect adaptive routing.

Q1 reflects the state of q0 and q2 reflects the state of q3.

q0 needs to be completely full in order for q1 to reflect the congestion. Thus, using local information requires sacrificing some packets to properly determine the congestion – resulting in packets being sent minimally having much higher latency.

As the load increases, although minimally routed packets continue to increase in latency, more packets are sent non-minimally and results in a decrease in average latency until saturation.

Solution:
using credit round-trip latency to sense congestion faster and reduce latency.

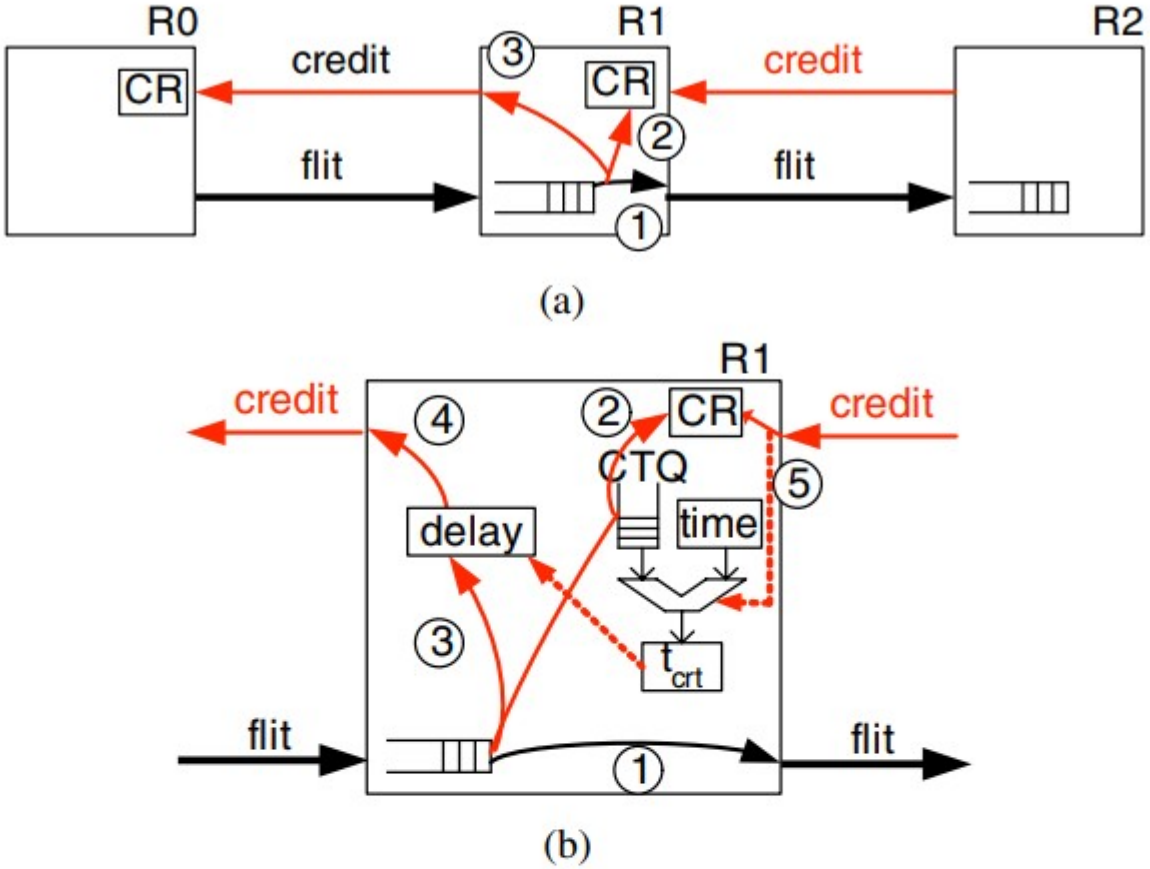
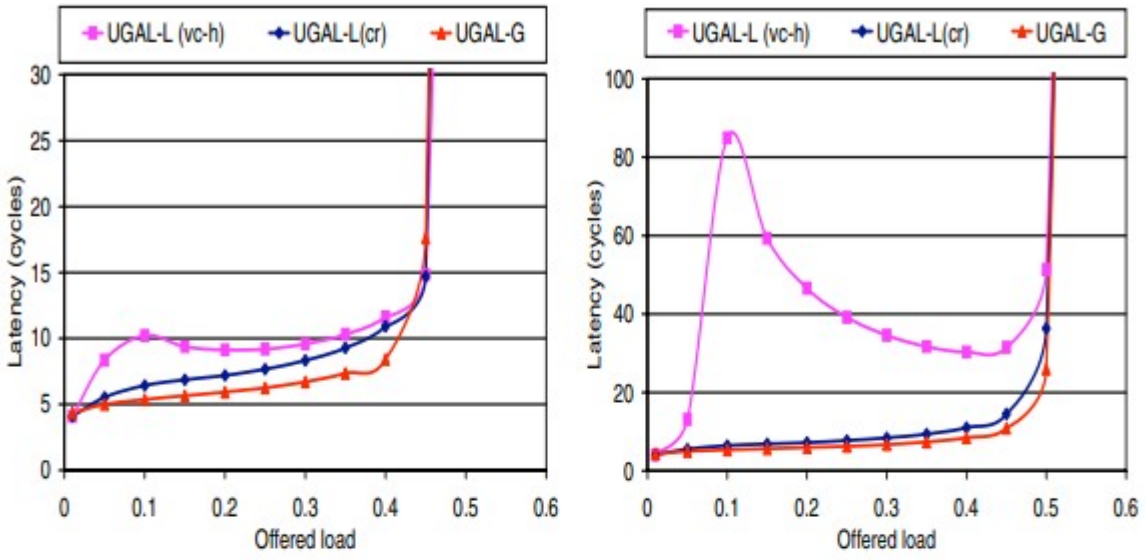


Figure 17. (a) Conventional credit flow control. As packets are sent downstream ①, the output credit count is decremented ② and credits are sent back upstream ③. (b) Modification to the flow control to use credit round trip latency to estimate congestion. In addition to the output credit count being decremented ②, the time stamp is pushed into the credit time queue (CTQ). Before sending the credit back upstream ④, the credit is delayed ③. When downstream credits are received ⑤, credit count (CR) is updated as well as the credit round trip latency (t_{crt}).

Comparison between UGAL-L_VC_H and UGAL-L_CR
Shows the solution is effective.



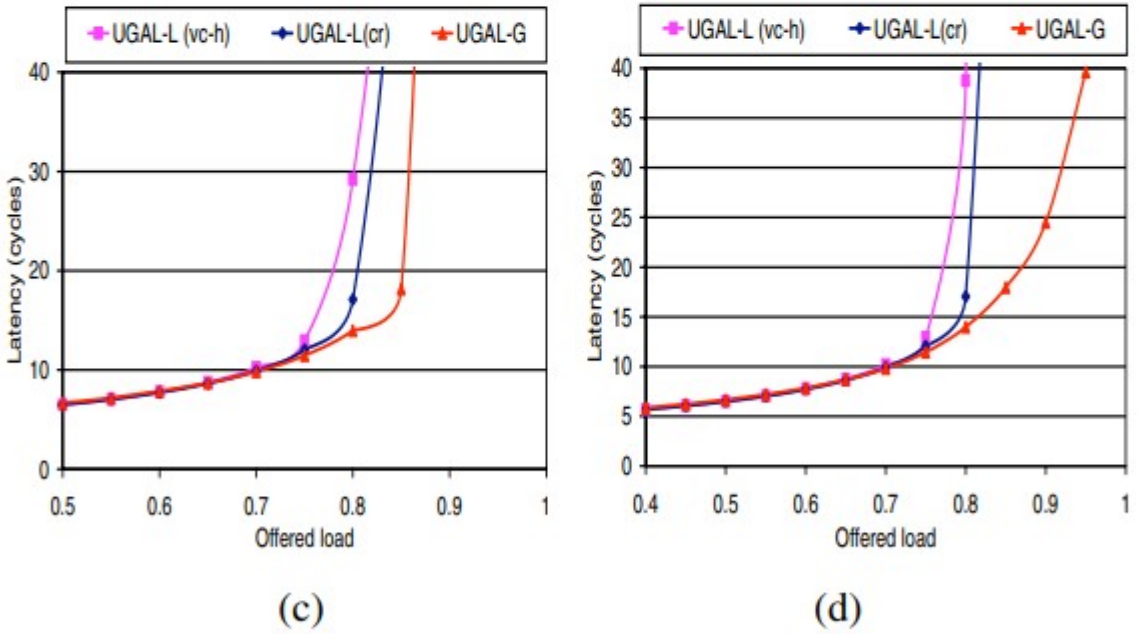


Figure 16. Performance comparison of UGAL-L$_{CR}$ with (a,b) WC traffic and (b,d) UR traffic. The buffer sizes are 16 for (a,c) and 256 for (b,d).

Cost Comparison:

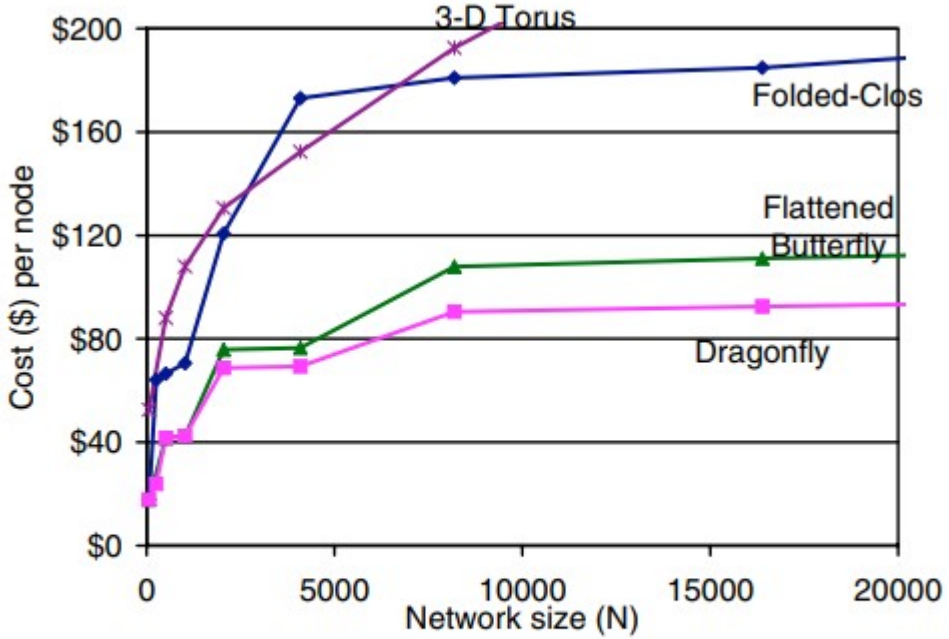


Figure 19. Cost comparison of the dragonfly topology to alternative topologies.

That’s all,

Thank you!

Cite:

- Kim, John, et al. "Technology-driven, highly-scalable dragonfly topology." 2008 International Symposium on Computer Architecture. IEEE, 2008.