Datacenter Network Design: Performance/Power Comparison of Large-Scale Network Configurations and a Way to Avoid it!

Christina Delimitrou
Electrical Engineering Department
Stanford University
Stanford, CA, USA, 94305
cdel@stanford.edu

Milad Mohammadi
Electrical Engineering Department
Stanford University
Stanford, CA, USA, 94305
milad@stanford.edu

Frank Austin Nothaft
Electrical Engineering Department
Stanford University
Stanford, CA, USA, 94305
fnothaft@stanford.edu

Laura Sharpless
Electrical Engineering Department
Stanford University
Stanford, CA, USA, 94305
lsharpless@stanford.edu

ABSTRACT

In this paper, we present a datacenter interconnction network topology that performs at the minimal cost-latency-power product point of all Dragonfly routers in our design space. With a 1.2x cost savings and a 4x power savings over other Dragonfly designs at a 1.2x performance hit, our topology and choice of routing function are effective optimal for a modern datacenter where design choices are centered around capital and operational expenses.

1. INTRODUCTION

As more and more computing is occuring in larger and larger clusters, the traffic that flows between the nodes in the cluster has grown, and the importance of the networks that connect these nodes has increased drastically. As discussed in [5], datacenters are prone to network collapses (TCP incast) when using a standard TCP/IP based networking system. This TCP incast problem causes network bandwidth within the datacenter to fall drastically, causing user visible application-level latency to decrease. Besides the TCP incast problem, a TCP/IP based networking solution is suboptimal in datacenters due to the (generally) short lived traffic flows that run inside the datacenter. Due to the TCP window's slow start, short lived flows frequently do not achieve peak bandwidth and are difficult to model. [6]

The aforementioned problems with the TCP/IP stack in datacenters has driven research into the design of special purpose interconnection networks for data centers, similar to those in a conventional supercomputer. Although the design of a custom interconnection network may be more expensive than the implementation of a normal network using COTS parts, a well designed interconnection network can provide substantially improved performance and lower the power consumed by the network. In this project, we explored the Dragonfly topology as explained in [7]. Although most current datacenters use a fat tree topology, the Dragonfly topology provides improved cost and increased fault tolerance due to link redundancy. This is especially significant, due to the tree saturation issues that are found in fat trees. [8]

As mentioned above, we chose a dragonfly based topology due to it's improved costs. As discussed in [7], the use of high radix routers in a dragonfly topology serves to decrease cost by minimizing global channels, which dominate the costs of operating a datacenter. While high radix routers had previously been frowned upon because they increased the length of the longest link in the datacenter, improvements in fiber optics technology have made long, high speed links feasible. [9] Besides the improvements in high speed links that have made high radix networks possible, datacenter topology research had been constrained to TCP/IP networking. Although TCP/IP is commonly used, it has large faults. Some of the seminal datacenter networking papers acknowledge that one of the significant reasons to choose a fat tree topology is that it allows for preventing loops from occuring with TCP/IP traffic. [3][11]

2. DESIGN OVERVIEW - TOPOLOGY

The state-of-the-art topology deployed in large-scale data centers today is the fat tree [2, 3, 4]. The reason behind this implementation is simplicity and good load balancing properties due to randomized routing. However, fat tree topologies are by no means the optimal for large-scale systems. In this work we implement a *dragonfly topology* to improve the network's latency by reducing the number of hops between

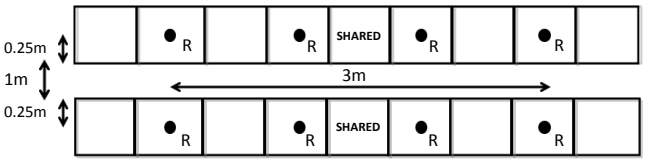

Figure 1: The topology of one group. Each row is comprised by 9 racks, and the whole group has 18 racks. Each square represents one rack, and each square with a dot, is a rack with a router. Routers are shared among neighboring racks, as shown in Figure 3.

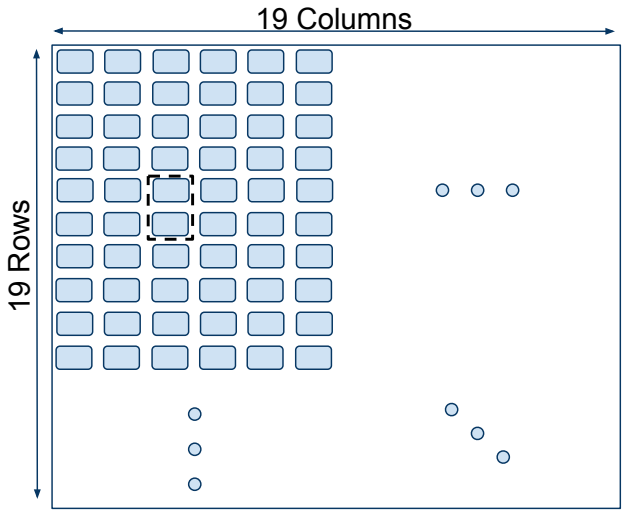

Figure 2: The floorplan of the datacenter, with each component representing 9 racks.

any two nodes, and the *Total Cost of Ownership* by reducing the number of optical cables used to connect different groups.

In this section, we present the design overview of our network, the layout of the data center as well as the configuration of the topology in terms of endpoints per router, routers per group and total number of groups in the system.

For our network configuration we have chosen to implement a topology that promotes energy and cost considerations to a first-order design constraint. We try to minimize the use of expensive optical cables, and maintain a low number of routers to reduce energy dissipation, and cost. Although this design will experience lower performance compared to networks that optimize strictly for throughput and latency, we have seen that quantifying the price of efficiency offers useful insight in provisioning trade-offs which are important challenges in current data centers. In Section 6, we perform a comparative study between different configurations in terms of Performance/Watt and Performance/$. In that study our configuration ranks among the top designs when the figure of merit is not strictly performance, but power and cost as well.

Based on our topology, each group is comprised of 18 racks, with 9 racks in each of two rows (Figure 3. Each rack has 32 nodes, and each router services *71 endpoints* (p=71). This means that in each group there are *8 routers*

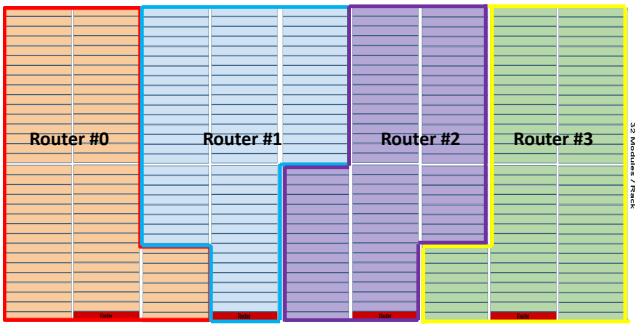

Figure 3: Router sharing across different racks. Nodes of the same color are serviced by the same router.

(a=8), and the *total number of groups is 177* (g=177). Finally, the radix of the router that we use is k=100, out of which 71 links are devoted to the intra-group electrical connections, 7 to the routers of the same group, and 22 to the inter-group optical links. Routers within a group are shared across racks in a way that strives to keep the distance from any node to its router as uniform as possible, in an effort to minimize the variability of latency across nodes of a group. Figure 3 shows this design, where nodes of different color are serviced by different routers. This scheme is symmetric every 8 racks, and although not currently adopted in data centers, could assist with network latency variability, which is a significant challenge for large-scale systems.

One disadvantage of this configuration is the oversubscription of the connections to the optical cables, i.e. if more than 22 endpoints simultaneously need to communicate to external groups, the optical links will become congested. Descreasing the size of the groups and as a result increasing the number of optical links is expected to solve this problem, although at the cost of a higher power consumption and TCO. All connections within a group are performed via electric links, and any router can communicate with any other router in the same group with a single electric wire, without repeaters, based on the Manhattan distance (i.e. no diagonal wires crossing the isles). One additional optimization that can be deployed for cost optimality is replacing some of the optical wires with electrical. Given the maximum length of electric wires without the use of repeaters (5m), neighboring groups of up to two groups away can be connected solely using electric wires, thus significantly reducing the dominant cost factor, which is the number of optical links in the system.

3. ROUTING

In determining the routing function that we wanted to implement, we considered Minimal (MIN), Uniform Globally Adaptive Load-Balanced routing (UGAL), Progressive Adaptive Routing (PAR), and Piggyback (PB). With an unbalanced offered load on different parts of the network due to hotspot traffic and varying sizes of packets, the routing function needed to be adaptive, which eliminated MIN. UGAL is a less adaptive routing scheme than PAR, so the limited extra cost of implementing PAR made it more attractive.Choosing between PAR and PB was determined by ease of implementation in the simulator: PAR was reason-

ably straightforward to implement while PB required significant changes to the routers. These led to PAR being selected as our routing function.

3.1 Progressive Adaptive Routing

PAR aims to route traffic along the shortest, least congested path to the destination. At every hop in the source group, the congestion along the minimal path is evaluated against congestion along a non-minimal path. The non-minimal path is determined by choosing an intermediate node (in line with Valiant's algorithm) to route to before routing to the destination. When the minimal path has greater congestion than the non-minimal path, the packet is diverted along the non-minimal path. From this decision point, the packet will route minimally to the intermediate node and then minimally to the destination node. Thus, at each router within the source group, the current route can be re-evaluated to determine the best overall route for the packet.

3.2 Implementation

At each router, when a head flit is being routed, its source group is compared against the current router's group. If they are equal and the packet is being routed minimally, then the packet originated from the current group and the router can decide whether to route the packet minimally or non-minimally.

Routing non-minimally requires an intermediate node to route through, as per Valiant's algorithm, so a random node in the network is chosen as the intermediate. The number of hops to reach the destination by routing through the intermediate node, along with the queue length for routing to the intermediate node at the router are compared to those for continuing to route minimally:

$$HopCount_{min} * QueueLength_{min} <= HopCount_{non-min} * QueueLength_{non-min} + Threshold \quad (1)$$

Given that the apparent congestion for routing through the intermediate node is less than the congestion encountered routing minimally, non-minimal routing begins. Once a packet starts routing non-minimally, it cannot switch back to minimal routing, so loops can't be created. But once a packet has an intermediate node chosen, it will be routed minimally to that node, and then routed minimally to the destination from there.

3.2.1 Virtual Channels

There are four virtual channels used in our implementation of PAR to avoid deadlock. These are labeled in Figure 4 for reference.

Within the source group, packets are routed minimally on VC0. When packets begin to be routed non-minimally, they switch to VC1. This prevents packets that are routed back along the same path from deadlocking the network. As packets are routed to the intermediate node, they use VC1, and when packets arrive at the intermediate node, they switch to use VC2. All packets being routed to their destination minimally use VC2. Thus, when minimally routed packets leave their source group, they route via VC2 on the optical cable. When packets reach the destination group, they swap to route on VC3, since they have crossed the network dateline. From then on, within the destination group, packets are routed minimally on VC3.

This scheme allows packets being routed minimally and non-minimally to be on separate virtual channels, eliminating the threat of deadlock.

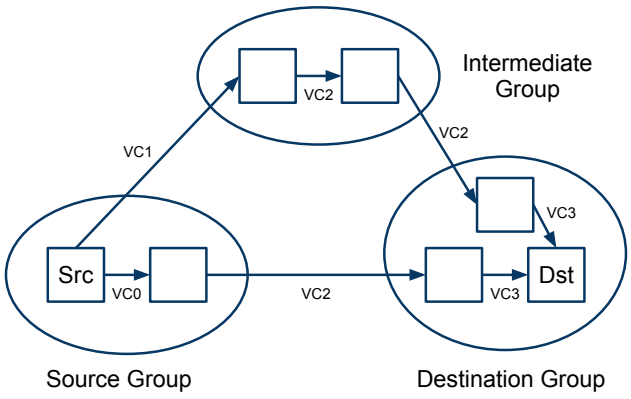

Figure 4: PAR routing using 4 virtual channels.

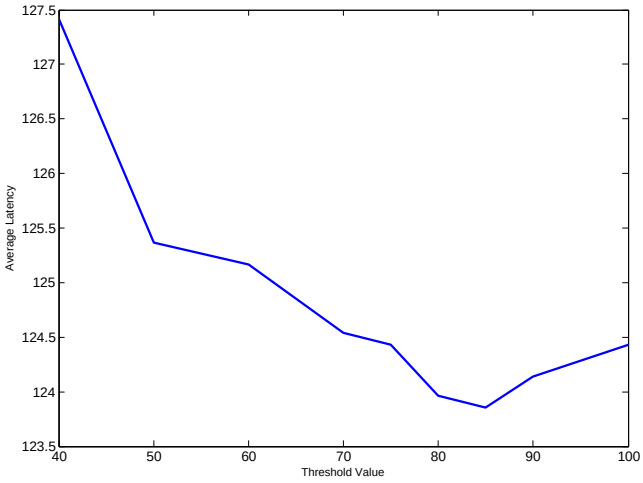

Figure 5: Average packet latency across various threshold values.

3.2.2 Threshold

Figure 11 shows the effect on average packet latency with varying threshold in all routers across the system. With very low thresholds, packet latency increases due to many packets being routed non-minimally when congestion isn't very bad. By increasing the threshold, packet latency drops until a threshold value of 85. Further increasing the threshold skews the decision towards minimal routing, at which point latency plateaus (beyond 100 in Figure 11).

The latency flattens out past 1 threshold of 100 due to our topology. Since we are bandwidth constrained between groups, queues for optical links become congested and packets are misrouted. But all of the optical link queues in the group are congested, so mis-routing packets increases the hop count without decreasing the latency. Packets will face similar congestion at the next router and won't have benefited from being misrouted. Thus, as the threshold increases, biasing the decision towards minimal routing, most pack-

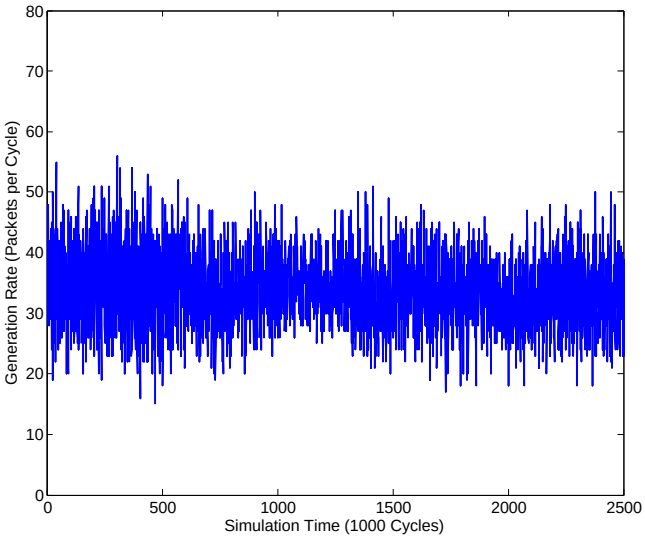

Figure 6: Number of packets generated in the network on a per-cycle basis.

ets are routed minimally unless congestion is very lopsided within a router.

The only downside of setting the threshold very high is that it doesn't allow for significant backpressure. With changing traffic patterns, the router will be slow to switch from minimal to non-minimal routing, increasing the latency of some packets. But the provided traffic patterns are reasonably well-behaved, so backpressure isn't a critical issue faced by the network.

4. TRAFFIC MANAGEMENT

The main focus of the implementation coding of the simulator was to correctly handle traffic management. This section explains our implementation approach to generating the desired mix of network traffic.

4.1 Packet Injection and Ejection

At every cycle, each node attempts to send a packet to the network. A node successfully sends its packet when it has fewer than four request packets in flight and when no outstanding replies are pending to be serviced. These condictions avoid congesting the network when older packets in the network have not yet been serviced, hence avoiding instability. Figure 6 confirms the stability of our network by showing the relatively constant rate of packet generation over time. Although the rate of generation fluctuates between cycles, the overall amount of traffic generated is bounded, ensuring that the network doesn't generate more traffic than it can process. The flow diagram on Figure 7 describes illustrates a simplistic version of our generation scheme.

The four traffic patterns outlined in the project disctiption (Uniform, HotSpot, Bad Dragon, and Uniform) are implementatecd using four traffic classes. The combination of these traffic patterns allowed us to model a realistic network traffic pattern for our experiments. We use a flit and packet size of 100B for all of the traffic classes, and number of packets generated by each traffic pattern depend on the actual message size.

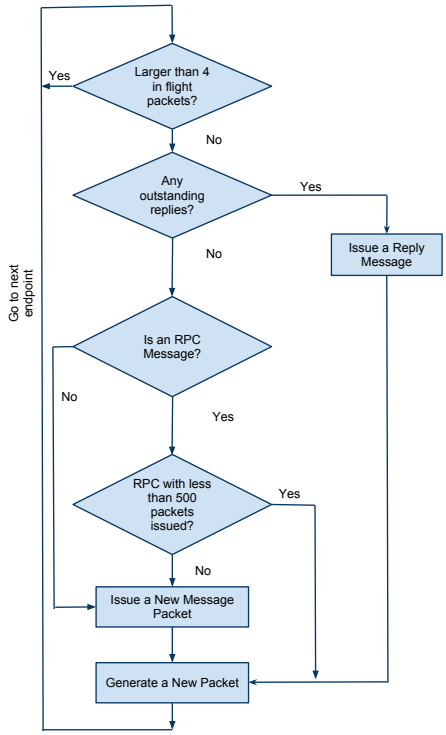

Figure 7: Traffic management flow control diagram (simplified) basis.

Sice RPC require 500 100B message packets to be sent, we implemented a feedback system that enables sending 100B packets over 500 cycles. If an RPC message is generated by a node, that node will stop issuing new message types until all the 500 packets are successfully injected into the network. Then, the node generates a new message with the probabilities outlined in the projet description. Packet destinatoins are chosen based on the traffic pattern determined by their message type. For instance, if a message type turns out to be adverserial RPC, it will assigned as a HotSpot message class and a HotSpot traffic generator determines the packet destination. Once the message type and the destination of the packet are determined, the packet is sent over the network.

4.2 HotSpot

Our implemetation of the HotSpot traffic generator assumes 10 fixed endpoints as the HotSpot nodes. Once a message type is set to be RPC Adverserial, its destination is automatically set to be one of the 10 HotSpot noeds that are present in the network. To model the entire network, we chose to have one of our HotSpot nodes in one of the groups that we are simulating and leave the other nine outside of the cloud. This way we ensure that one of the two simulated groups has hotspot traffic while the other one does not.

As you will see in the next section, including HotSpot nodes in the cloud creates extra complications that required us to think more carefully about how the cloud latency can

be calculated.

4.3 Slicing

To slice a netowrk topology, one should take into account the accuracy of the slice in representing the true characteristics of the network. We assumed the number of possisle nodes that can be simulated in a reasonable time is about 1000 (1% of the entire network size). We started off by taking a full group (with 8 routers and 568 endpoints) and 54 other groups, each with only router ndoes present for simulation (no endpoint nodes). While we believed this approach reflect a very accurate model of the network, we soon discovered the extreme complexity of implementing such a structure in code given the time we had. As a result, we came up with a simpler topology slice model that can represent the network within our 1000 node simulation budget; we chose to model two groups and a cloud node as it is shown on Figure 9. Below is the description of how we constructed the cloud network.

To construct the sliced topology, we need 1950 optical links in total, and 56 electrical links per group (112 in total) to connect the routers only. We also need a total of 1136 electrical links to connect the endpoints to their respective router. The algorithm below guarantees that the network is fully connected:

- Connecting Local Routers (Electrical) - Figure 8:
  - If my_router_ID is < destination_router_ID: connect my link_ID to the link_ID of the destintion router that satisfies dest_link_ID = my_router_ID.
  - If my_router_ID is > destination_router_ID: connect my link_ID to the link_ID of the destintion router that satisfies dest_link_ID = my_router_(ID-1).
- Connecting Global Link (Optical) - Figure 9: All optical links are connected to the cloud node while the link-ID=799 on router 7 is connected to link 1599 of router 15.

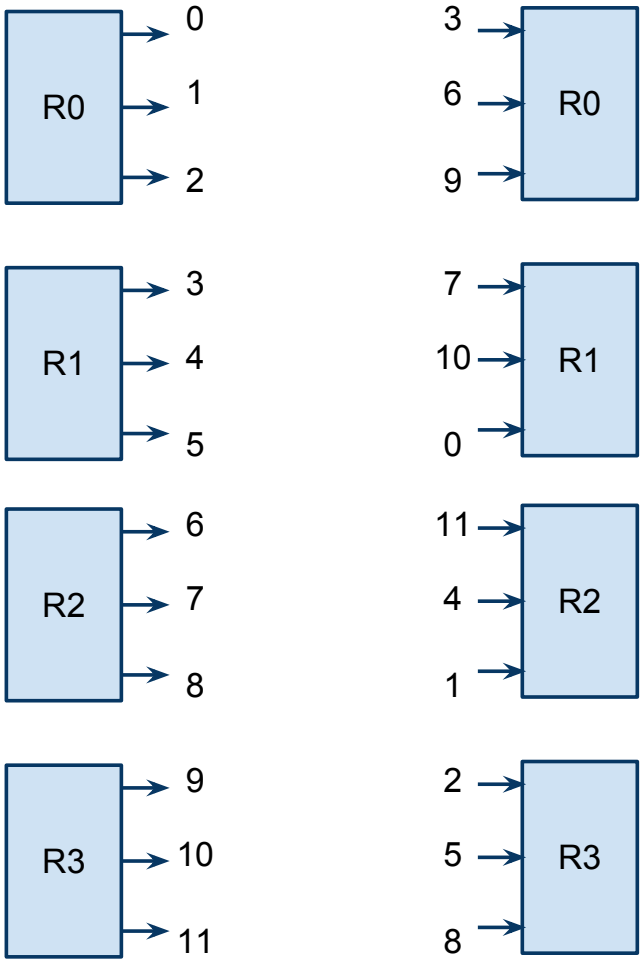

Figure 8: Intera-group router link connection example basis.

4.4 Cloud Abstraction

Ideally, to calculate the cloud response time latency we would measure the average traffic latency of a packet to be twice the latency of a packet going from one node on to an optical link (t_node-to-optical-link). This latency value includes an estimate of the routing, wiring, and queueing latecy. While this approach can give a reasonalble estimate of latency for UR traffic, we belive that it is not effective when multiple traffic patterns are present in the network. This estimate becomes even worse when one of the traffic patterns is HotSpot. Our solution for getting accurate latency results is to dynamically update the average latency of packets going from any node in the slice to any optical link (e.g. take the latency average every 20 cycles). This latency value takes into account the hotspot traffic latency (present in one of the two groups in the slice). Depending on the intensity of Hotspot traffic in the network, the cloud average response time varies. For this project the HotSpot traffic turned out to be not hot enought to make a large impact on the overall latency of the packets traversing the entire data-center.

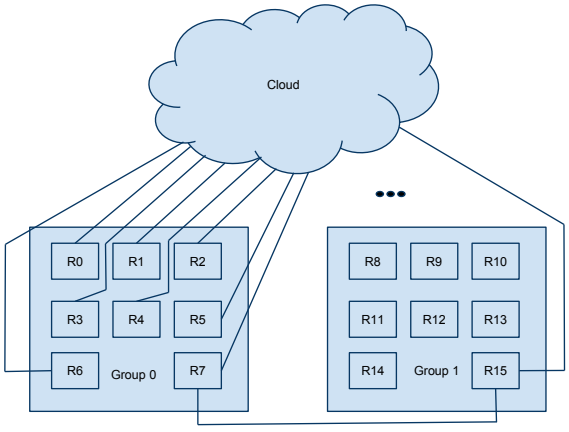

Figure 9: Inter-group router link connections basis.

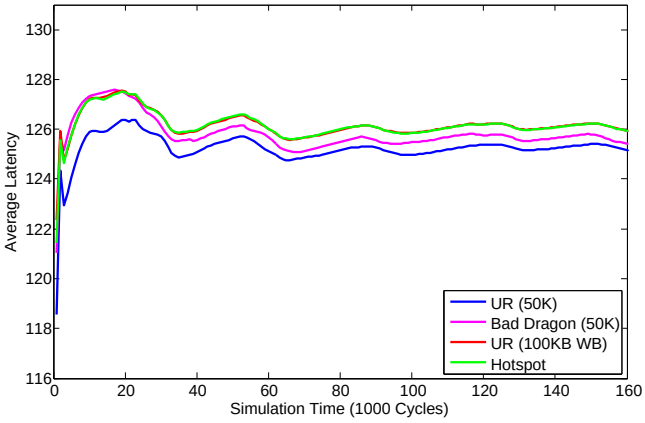

Figure 10: Trace of average latency for the four different classes of packets.

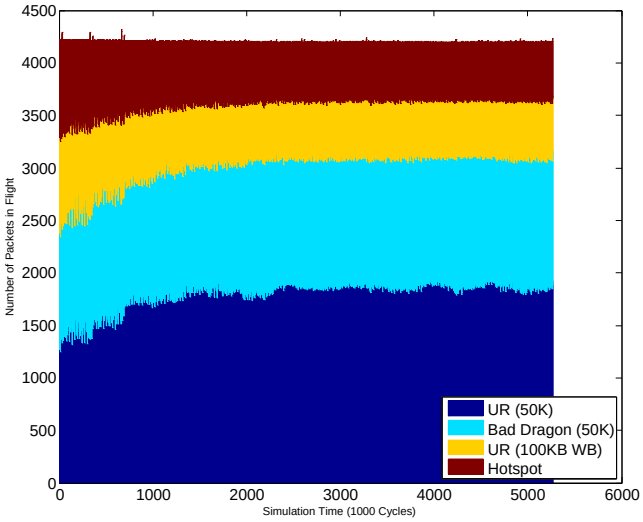

Figure 11: Total number of packets in flight over the simulation, broken down by packet class.

The second issue associated with the cloud network is related to deciding on how to send packets in and out of the cloud. Four different traffic scenarios are possible: sending packets to the slice (from the cloud), receiving packets from the slice and replying to them, sending packets non-minimally from a slice node to another slice node via the cloud node, and sending packets non-minimally from the cloud to the cloud via an intermediate slice node.

Transmitting packet:

- From the cloud to the slice: In this case, packets are sent to the cloud and their reply is returned according to the latency protocol described above. The probability of sending a packet to the slice is about 99% indicatin the need for accurate measurement of the reply message latencies from the cloud.

- From the slice to the cloud: As mentioned previously, the radix of the cloud is 350. This means that the traffic manager attempts to inject up to 350 packets from the cloud into network every cycle. Of all the injected packets from the cloud, only 1% of them are actually destined to the group in the slice that they are injected into.

- From cloud to cloud node via the slice: The remaining 99% of the packets injected by the cloud are non-minimally routed and their destination is the cloud. When a non-minimal packet is injected into a slice, the packet will take two optical hop and one local hop to return to the cloud.

- From slice to slice via the cloud: less than 1% of the time packets are routed non-minimally from a slice node to another slice node. The amount of time the cloud waits to reply to the sender is $2xt\_node2opt - link$. The amount of time the cloud waits to before it injects the packet to the destination group/router is $2xt\_node2optlink + t\_local - hop\&routing$. t_local-hop&routing is the routing latency + local link hop latency in the network. This value can also be meaured the same way t_node2optlink is measured.

5. COMPARATIVE DESIGN ANALYSIS: QUANTIFYING THE PRICE OF EFFICIENCY

5.1 Comparison of Performance/Watt and Performance/$ between Different DC Network Configurations

In Section 2 we presented the topology implemented in the current design. We discussed the advantages and disadvantages of this decision in terms of performance and efficiency and provided some reasoning as to why it was preferred over other configurations.

The scope of this Section is to perform a comparison between different configurations (e.g. topology, routing) in terms of Performance/Watt and Performance/$ and to provide insight in the trade-off between performance, efficiency. In order to do this, we perform an exhaustive simulation of all dragonfly network configurations that adhere to the design, cost and power constraints of the assignment and plot the ranking of each network design in terms of Latency over Power and Latency over Cost.

All simulations are run for 1 million cycles using virtual-cut through flow control, fully simulating around 1100 nodes in the Datacenter. These nodes are divided across a different number of groups, depending of the configuration that is simulated. The configuration of the network: the (a, p, g) parameters and the routing algorithm vary across different designs.

Additionally evaluating alternative flow control techniques, as well as a comparison between dragonfly and the previously state-of-the-art DC network, i.e. Fat Tree, is deferred to future work.

We calculate the power consumption and cost for the system for each configuration based on the provided guidelines. Figure shows how each configuration ranks in terms of Performance and Efficiency. The y-axis represents latency in ms and the x-axis power consumption in MW.

Figure 12 shows how the different configurations rank in

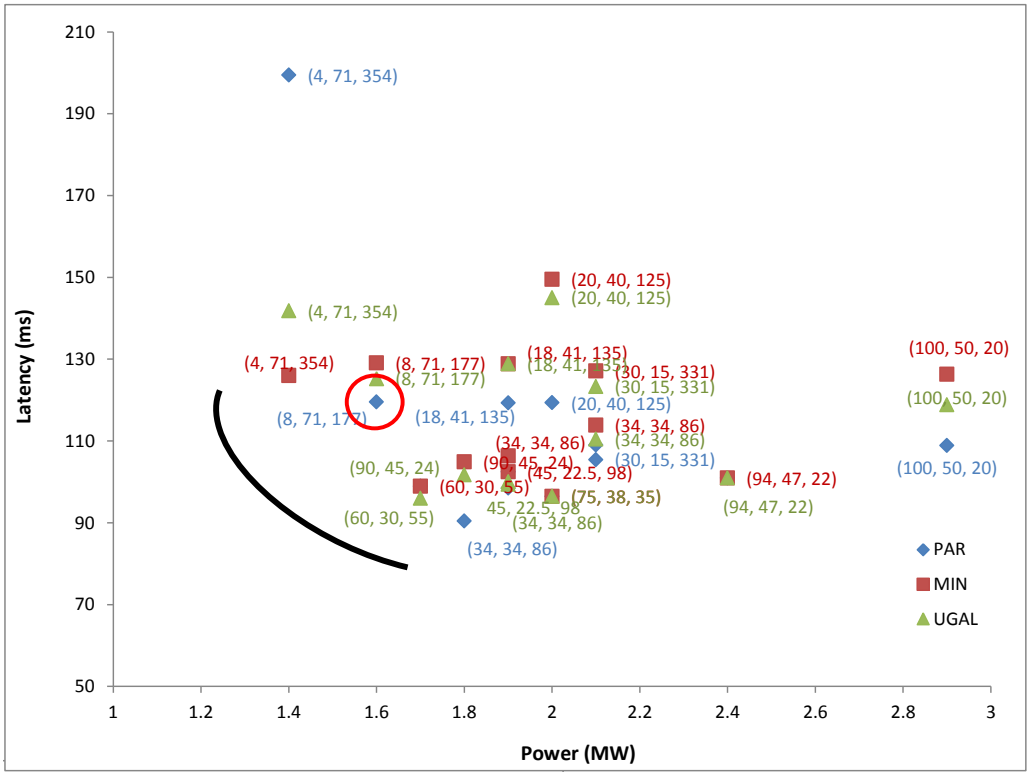

Figure 12: Performance over Watt comparison between different network configurations (topology and routing).

terms of latency over power consumption. The optimal points in the plot are the ones in the bottom left corner near the pareto-optimal curve, while the worst ones are the ones in the top right one. Our configuration ranks among the best for Latency/Watt (highlighted region), although more power hungry configurations (e.g. (a, p, g) = (30, 15, 331)) achieve lower latency. Also, although designs that use PAR tend to have lower latency than those with either MIN or UGAL routing functions, it is the configuration of the topology than mainly affects latency here (almost an over of magnitude difference between the weights of topology and routing in latency). There are some configurations that achieve lower latency per Watt than our network, and that brings up the issue of oversubscribing the optical links in the system. If we retune our design tohave slightly fewer endpoints per node, and more routers per group we expect a shift towards the bottom side of this graph. The best configuration for performance and power is (34, 34, 86) using PAR. We observe that, as expected designs with many routers per group tend to have lower latencies, but increased power, while networks with few routers per group experience lower performance.

We note here that the power shown in the graph is *calculated* and *not simulated*, therefore it does not account for any impact the routing decision might have on energy consumption, i.e. configurations that only differ in routing will have the same power budget.

Similarly for cost, in Figure 13 we show how latency ranks for different designs over the deployment cost of the network. Based on the provided guidelines, networks with high power consumption tend to have high cost as well, therefore the similarities between the two graphs are expected. There are however a few outliers, e.g. while cost increases monotonically as we increase the number of routers per group, power consumption experiences some unpredictability in its behavior.

Again, our design ranks highly on the performance-cost plane (highlighted region), with few designs achieving better performance, TCO trade-offs by reducing the number of endpoints per router, e.g. (34, 34, 86). Designs with high numbers of routers per group achieve significantly better performance - and reliability - but are severely hurt in the cost aspect.

5.2 Performance Predictive Scheme based on Linear Regression

In this Section we propose a *performance predictive scheme* that allows estimating the performance of a network configuration without actually deploying it, either in simulation or in actual implementation. Our scheme is based on simple linear regression and uses the data of the previous study as the training set to predict the performance of an unknown topology. For example, if we have simulated a set of networks (with different design parameters (a, p, g) and different routing algorithms, or flow-control techniques) we can extract the feature vector from these parameters and use the data as a training set to estimate the performance and/or efficiency and cost of a different configuration.

In our current evaluation, we collect statistics for latency over a long simulation period for a large number of different configurations, and use the configuration parameters, along with the routing and flow-control techniques, as the features that train the linear regression. The benefit of this scheme, apart from the insight it provides for different network configurations, is the ability to predict the performance (in either throughput or latency) of a new configuration without

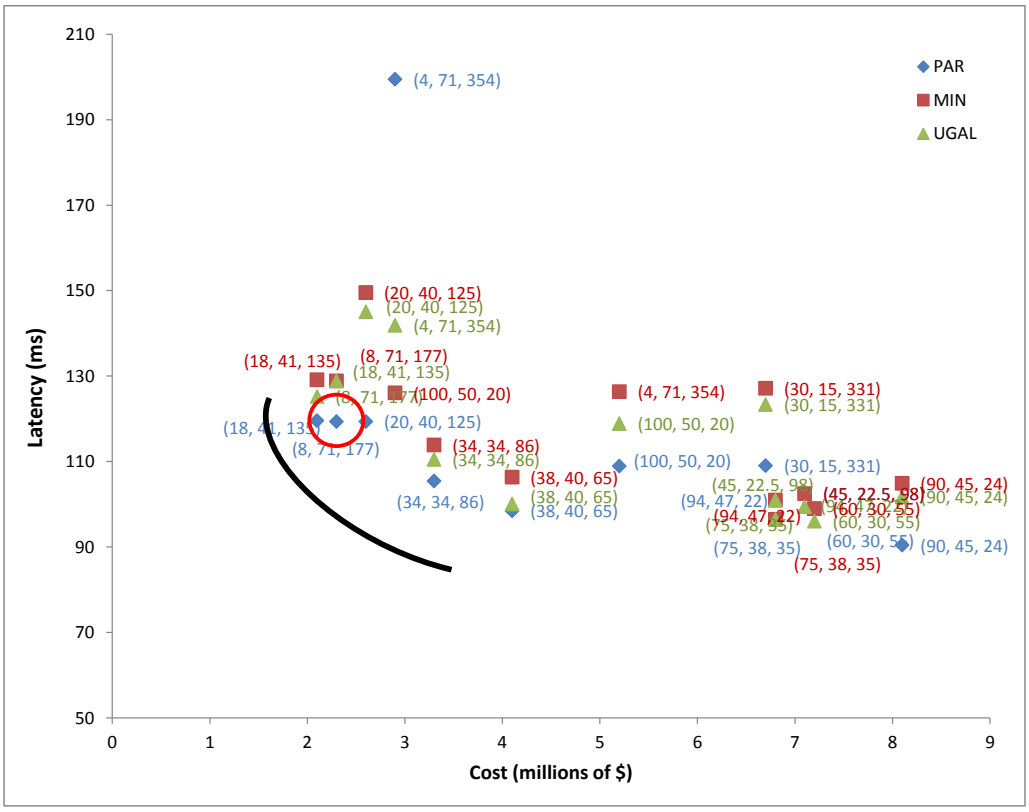

Figure 13: Performance over TCO comparison between different network configurations (topology and routing).

the overhead of **simulating** it, or worse **implementing** it. This greatly simplifies trimming down the available parameter space, which as shown in Section 6.1 is significantly large, while providing accurate estimations for the performance of the system.

We have used our scheme to predict the latency of new configurations based on the behavior of already simulated networks and achieve a **less than 5% deviation** between predicted and simulated performance in all cases. As our training set we use all previously simulated configurations, except for the one currently being evaluated, in order to ensure that training and testing set are decoupled. Figure 14 shows the comparison between predicted and simulated performance (latency) for three configurations that we have evaluated. In all cases the results from the predictive scheme resemble closely the simulated metrics, which is promising towards extending this method to a more complete and robust framework.

As part of future work, we plan to evaluate the sensitivity of our method in the size and divergence of the training set (e.g. number of different simulated configurations, number of samples per configuration) as well as the *number* and *type of features* in the feature vector. As shown in Figure 5 the feature that affects the results most is the number of endpoints per optical channel, followed by the number of intra-group router connections. This result makes sense given that optical channels should not be kept idle, while intra-group router connections should not be oversubscribed, as this would result in underutilizing the optical links, which are the expensive resource in the system. Alternatively, one can use a scheme like this to predict the optimal configuration for a network given SLAs, or a power and cost budget. Although getting the exact configuration for the absolute optimum is an NP-hard problem, linear regression or more so-

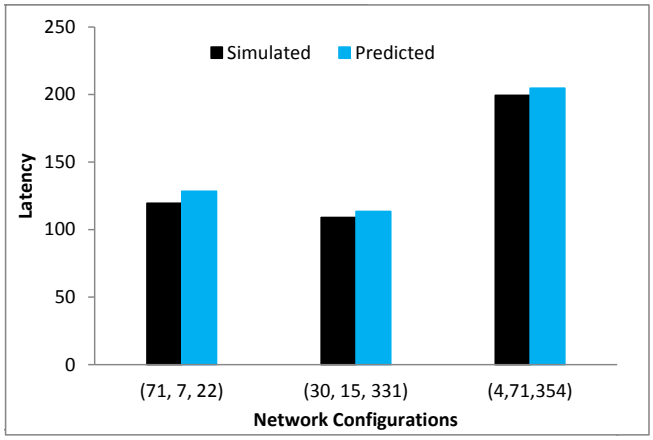

Figure 14: Performance Predictive Scheme Validation: Comparison between predicted and simulated performance for three possible network configurations.

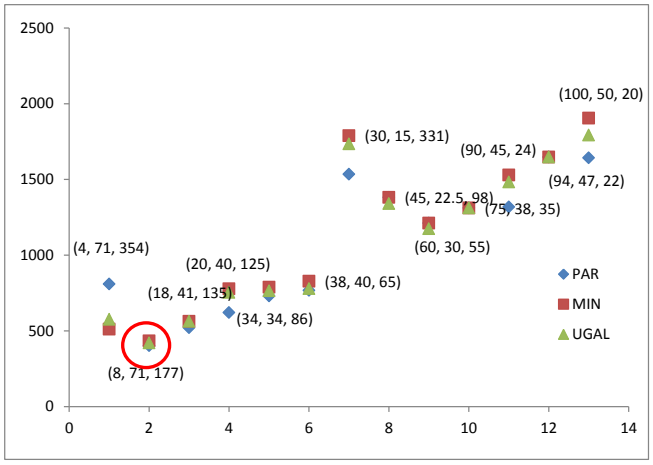

Figure 15: Performance-power-cost producto for different network configurations (Topology and Routing).

phisticated ML schemes, can offer design suggestions within an error margin of the optimal configuration.

6. FUTURE WORK

Although PAR performs better than MIN or UGAL, piggyback routing (as introduced in [10]) would provide a further advantage over PAR. While both our hotspot traffic and our adversarial traffic pattern wound up being fairly innocuous, it is conceivable that other, more adversarial traffic distributions could be used in a datacenter using this interconnection network.

Since our goal was to optimize datacenter costs, we aimed to decrease the number of global optical links. Alternatively, we could have switched from optical links to global metal links using repeaters. Since this would decouple network cost from the number of global links, we would likely wind up with a very different Pareto optimal frontier for our design space. We chose not to perform a design using global metal links due to signal integrity concerns (this would have required the use of repeaters) which would have vastly complicated our process for costing our datacenter.

7. CONCLUSIONS

In this paper, we presented an architecture for a datacenter interconnection network that uses a dragonfly topology to optimize for cost. Our architecture was designed to optimize cost through picking a topology that sits in a flat plane for cost. After constructing this cost curve, we performed a space exploration to determine which topologies provided best performance and power efficiency at a given price. Our design space exploration led to the realization that our chosen topology is nearly on the optimal frontier for the performance per cost curve, and is one of the more power efficient designs. As shown in Figure 15, our design optimizes the product of latency, power, and cost over all other network configurations of the exhaustive search exploration. Since performance is impacted by the balance of latency over core links and cost is dominated by global link costs, the lone more cost efficient topology chose to have larger groups with fewer endpoints per router. This topology (41 nodes per router, 18 routers per group, 133 groups) had a better balance of traffic between local and global interconnects while keeping the number of global links approximately equal. While this new topology has marginally better (5%) performance-per-cost, it is approximately 30% less power efficient than our original proposed topology (71 nodes per router, 8 routers per group, 177 groups).

In modern datacenter installations, capital expenditures and operating costs dominate design choices. By acheiving near-minimal total component cost in our datacenter and by achieving a very formidable performance-to-power ratio, we have designed a topology that is very practical and very efficient. Although we do not have maximum performance (based off of the guidelines in [7] as well as our design space exploration), we achieve a 4x cost improvement and a 1.2x power efficiency over the maximum performance router for our topology, and achieve the minimum cost-latency-power product over all topologies in our design space.

8. REFERENCES

[1] William J. Dally, Brian Towles. “Principles and practices of interconnection networks”. Morgan Kaufmann, 2004.

[2] “InfiniBand in the Enterprise Data Center: Scaling 10Gb/s Clustering at Wire-Speed”. White Paper, Mellanox Technologies, 2006.

[3] R.N. Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat. “PortLand: A Scalable Fault-Tolerant Layer 2 Data Center Network Fabric”, In Proc. of SIGCOMMâĂŽĂĎÃt’09, August 2009, Barcelona, Spain.

[4] M. Al-Fares, A. Loukissas, A. Vahdat. “A Scalable, Commodity DataCenter Network Architecture”. In Proc. of SIGCOMMâĂŽĂĎÃt’08, August 2008, Seattle, Washington, USA.

[5] Chen, Yanpei and Griffith, Rean and Liu, Junda and Katz, Randy H. and Joseph, Anthony D. “Understanding TCP incast throughput collapse in datacenter networks”. In Proceedings of the 1st ACM workshop on Research on enterprise networking. WREN ’09, Barcelona, Spain, 2009.

[6] Mellia, M. and Zhang, H. “TCP model for short lived flows”. Communications Letters, IEEE. February, 2002.

[7] Kim, J. and Dally, W. and Scott, S. and Abts, D.”Cost-Efficient Dragonfly Topology for Large-Scale Systems”. IEEE Micro, January - February, 2009.

[8] Vogels, W., D. Follett, J. Hsieh, D. Lifka, and D. Stern. “Tree-Saturation Control in the AC3 Velocity Cluster Interconnect”. White Paper, 2006.

[9] Luxtera. “Fiber Will Displace Copper Sooner Than You Think”. White Paper, 2005.

[10] Nan Jiang, John Kim, and William J. Dally. 2009. “Indirect adaptive routing on large scale

interconnection networks." SIGARCH Comput. Archit. News 37, 3 (June 2009), 220-231.

[11] Albert Greenberg, James R. Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A. Maltz, Parveen Patel, and Sudipta Sengupta. 2009. VL2: a scalable and flexible data center network. In Proceedings of the ACM SIGCOMM 2009 conference on Data communication (SIGCOMM '09). ACM, New York, NY, USA, 51-62.