Non-Shortest Path Routing in Lossy Data Center Networks

Tal Mizrahi
Huawei

Shahar Belkar
Huawei

Oren Spector
Huawei

Reuven Cohen
Technion

***Abstract*—Over the past decade, high-speed data center networks have evolved rapidly. Beyond traditional Clos and Fat-tree topologies, a variety of innovative architectures have gained prominence, driven by the escalating demand for high throughput and low latency in machine learning workloads. Conventional shortest-path routing approaches often prove inadequate for some of these topologies, leading to bottlenecks and congestion that reduce throughput. In this paper, we introduce a routing approach that leverages both shortest and non-shortest paths to enable high throughput across diverse data center topologies. Paths are selected using a lightweight measurement protocol that estimates the one-way delay over multiple paths and selects the most available one. Our approach is specifically designed for lossy networks that do not implement an L2 flow-control protocol, such as Priority Flow Control (PFC), thereby minimizing the risks of head-of-line blocking and the costs associated with deadlock prevention.**

***Index Terms*—RDMA, multipath, telemetry.**

I. Introduction

Shortest path routing has been the dominating routing approach for decades due to its simplicity and its high performance in typical topologies and traffic workloads. In topologies that have many shortest paths between source-destination pairs, like Fat-trees [1], Equal Cost Multi-Path (ECMP) routing has been used to better utilize the network bandwidth. Using ECMP, different flows between the same source-destination pair can be routed over different shortest paths.

In the last years, a few alternative data center topologies have been proposed to Fat-tree [5], [7], [12], [13]. Their main purpose is to decrease the infrastructure cost by increasing the number of endpoints (hosts) N for a given diameter D and switch radix k, while still maintaining high global bandwidth [5]. Compared to Fat-tree, these topologies have a small number of shortest paths and many non-shortest paths between every source-destination pair. When traffic distribution is skewed, routing only on the shortest paths is sub-optimal, as these paths become a bottleneck while the network is still underutilized. Hence, non-shortest path routing protocols have been proposed for these topologies [11], [19].

All data center non-shortest path routing protocols we are aware of [5], [7], [11]–[13], [19] have been proposed for lossless networks; i.e., networks that employ a link layer flow control protocol, such as Priority Flow Control (PFC) [10], to prevent packet losses due to congestion. Lossless networks have become the common practice in data centers mainly because the dominant data-center transport protocol in the last two decades has been RDMA. All known non-shortest path routing protocols use the concept of adaptive routing in order to balance the traffic between shortest and non-shortest paths. With adaptive routing, a switch that receives a packet can decide whether to forward it over a shortest or a non-shortest path, depending on the present load on these paths. Non-shortest path routing protocols vary from each other in the way a switch predicts the load on each path. The simplest way is that the switch considers only the load on its local ports [15]. More complex protocols allow each switch to obtain up-to-date information about the load on remote links [2].

Another aspect that should be considered is that running large lossless data centers involves a significant throughput penalty. The fundamental reason is that in addition to blocking connections that contribute to congested links, layer 2 flow-control protocols also block connections that do not contribute to any such links. This problem is known as head-of-the-line blocking. In order to eliminate PFC[1], the Transport protocol must have a loss-aware congestion control protocol, and should use selective-repeat rather than go-back-N retransmission policy [20]. RDMA over Ethernet (RoCEv2) uses go-back-N because the underlying network is assumed to be lossless, in which case there is no difference between the performance of a simple go-back-N retransmission and the more complex Selective-Repeat retransmission.

We have developed an advanced transport protocol called the Lightweight Efficient Retransmission Protocol (LERP). This protocol enables the receiver to accept out-of-order packets, caused by losses of other packets, and correctly position them in memory, a technique known as direct data placement. To achieve this, each packet includes the precise memory location of its payload. LERP utilizes Selective Acknowledgment (SACK) messages, allowing the receiver to notify the sender of missing packets. By relying heavily on fast retransmission, LERP eliminates the need to wait for timeouts, significantly improving efficiency. For fast retransmission to work correctly, all packets within the same flow must traverse a single routing path, ensuring that when the sender receives a SACK, it can accurately identify and retransmit the lost packet. LERP's congestion control algorithm is based on DCTCP [3].

[1]Throughout the paper, we use "PFC" as a code name for any Layer 2 flow control protocol that guarantees a lossless network. All these protocols are similar, and they all suffer from the head-of-the-line blocking problem.

ISBN 978-3-903176-72-0

Number of connections	Load	Throughput without PFC	Throughput with PFC	Gain of removing PFC
16	0.03	1,360	1,288	5.59%
32	0.04	2,177	2,002	8.74%
64	0.07	3,415	3,038	12.41%
128	0.1	4,993	4,374	14.15%
256	0.15	6,769	6,026	12.33%
512	0.19	8,774	7,576	15.81%
1,024	0.23	10,689	8,703	22.82%
2,048	0.27	12,716	9,989	27.3%
4,096	0.32	14,868	10,455	42.21%
8,192	0.38	17,883	10,585	68.95%
16,384	0.47	22,211	10,961	102.64%
32,768	0.6	28,734	10,976	161.79%
65,535	0.73	34,795	10,894	219.4%

TABLE I: The performance in Fat-tree with and without PFC in a 3-level Fat-Tree with $1:4$ blocking ratio and 512 hosts.

One of the primary benefits of our proposed routing scheme is its ability to effectively prevent deadlock scenarios, which can arise in various network topologies due to PFC circular dependencies. Unlike alternative methods that rely on multiple virtual circuits (VCs), our approach utilizes a single VC, making it both more efficient and straightforward.

By eliminating PFC and using LERP, the data-center performance is significantly improved. As an example, consider Table I, which shows simulation results for a 3-level Fat-Tree with $1:4$ blocking ratio and 512 hosts. Each row shows a different number of connections that are randomly established, using uniform selection of source-destination pairs. Each of this connection sends a long message and the total throughput is measured when all the connections are active. Each row shows the total throughput obtained for these connections when RoCEv2 is used together with PFC, and when LERP is used without PFC. In this scenario, removing PFC can result in efficiency gains of over 200% . Moreover, these improvements become even more significant as the network size increases.

The significant throughput gain obtained for Fat-trees by removing PFC raises the question whether a similar improvement can be obtained for topologies that require non-shortest path routing. While we answer in the affirmative, we show that for such networks it is not enough to change the RDMA Transport protocol, and it also needed to change the routing policy. State-of-the-art routing schemes for topologies that contain many non-shortest paths require the switches to decide whether to route a received packet over a shortest path, or to deflect it to a non-shortest path, a concept known as "adaptive routing".

But as explained in Section II, when PFC is removed and packets can be lost, fast retransmit and congestion control are essential. Unfortunately, these crucial components do not work well with adaptive routing. Thus, we propose a loose source-routing scheme, which, together with inband telemetry, allows the sender to choose the best path to the destination for all the packets of a considered flow, without using adaptive routing. If this path becomes overloaded, and another path becomes more attractive, the sender moves the flow to the new path, a process also known as rerouting. Each path chosen by the source can be either a shortest path or a non-shortest path.

Our evaluation is focused on the Dragonfly topology, as this is the most popular topology that uses RDMA over non-shortest paths. We show that by removing PFC and running the proposed telemetry-based routing protocol, the throughput of RDMA over Dragonfly networks can be significantly improved, by up to a few hundred of percent. However, the proposed routing scheme is topology-independent, and it can be applied to other topologies for which non-shortest path routing is needed.

The rest of this paper is organized as follows. Section II describes the challenges of non-shortest path routing and presents the protocols we used as benchmarks in our evaluation. Section III introduces our telemetry-based non-shortest path routing approach. Evaluation results are presented and discussed in Section IV and concluding remarks are presented in Section V.

II. Background: The Challenge of Non-shortest Path Routing in Lossy Networks

A. The Dragonfly Topology

Dragonfly is a topology that uses groups of switches[2] with high radix (degree). The switches in each group are connected to each other in full mesh, using intra-group short-distance links. Each group is connected to each of the other groups using at least one long-distance link. In practical deployments, switches and end-points belonging to the same group are assumed to be collocated in a small number of chassis or cabinets.

The Dragonfly topology is a hierarchical network with three levels: switch, group, and network. A group consists of a switches that are fully connected to each other. At the bottom level, each switch has p local links to hosts, a–1 local links to other switches in the same group, and h global links to switches in other groups. Hence, the radix of each switch is $k = p + a + h - 1$. Each group has $a \cdot p$ links to hosts and $a \cdot h$ global links to other groups. Figure 1 shows an example of a Dragonfly topology with 9 groups. In this example, $p = 2$, $a = 4$ and $h = 2$.

[2]In Dragonfly's terminology, switches are also called "routers".

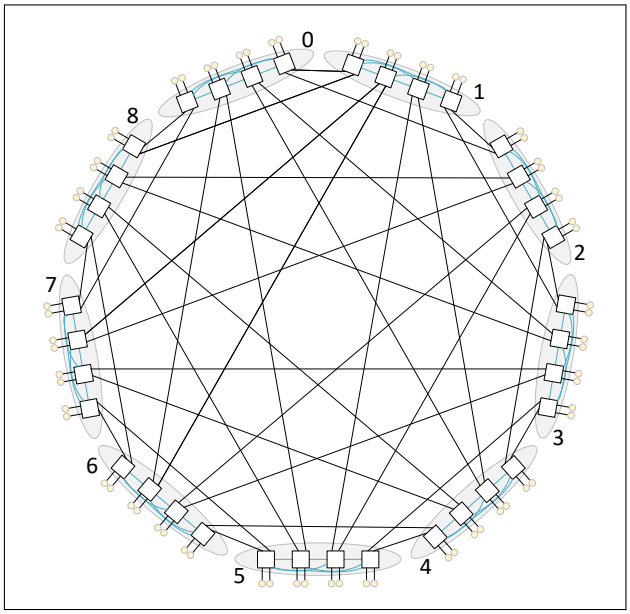


Fig. 1: Dragonfly topology.

Dragonfly is mainly used for large HPC clusters, and it has two main advantages over a Fat-tree topology. First, the number of hosts that can be connected to a network with a certain number of switch ports is larger than a comparable Fat-tree. Second, the average length of the shortest path in Dragonfly is shorter compared to a Fat-tree with the same number of switch ports. However, routing in a Dragonfly network is more difficult than in a Fat-tree network. In a Fat-tree, all the paths between every source-destination pair are shortest paths, which can be safely used by every simple routing protocol that uses ECMP (Equal Cost Multiple Paths). In contrast, a Dragonfly topology contains both shortest and no-shortest paths. Figure 4 shows two paths from a host in group 0 to a host in group 6: the left (purple) path is a shortest path, while the right (red) path is a non-shortest path. As long as traffic load is uniform, shortest path routing in Dragonfly maximizes the throughput of the network. However, when traffic is non-uniform, a routing protocol that uses both shortest and non-shortest paths outperforms a protocol that only uses shortest paths. For example, this is the case if all the traffic in the network of Figure 1 is from group 0 to group 6.

B. Background on Non-shortest Path Routing in Lossless Networks

Routing over non-shortest paths is also known as Valiant routing [7]. Valiant routing is used when the source and destination belong to different groups, and the direct link between the two groups is overloaded. In this case a non-shortest path that traverses some intermediate group can be used. We therefore divide the Dragonfly inter-group paths into two categories:

- MIN is a shortest path, which uses a direct link between the source and destination group. A MIN path is divided into three components: (a) in the source group, from the source node to the global link leading to the destination group; (b) the direct global link; (c) in the destination group, from the global link to the destination node.
- VAL is a non-shortest path, which uses an intermediate group in order to bypass a congested direct global link between the source group and the destination group. Using VAL, a packet is first routed to an intermediate group and then to the destination group. Thus, two global links are traversed and the whole path can be divided into five components: (a) in the source group, from the source node to the global link leading to the intermediate group; (b) the first global link; (c) in the intermediate group, from the first to the second global link; (d) the second global link; (e) in the destination group, from the second global link to the destination node.

The switches in each group are usually connected to each other in a full mesh. But even if this is not the case, we assume that intra-group routing is always conducted over the shortest paths. This is the case when the source and destination belong to the same group, when a packet needs to reach a non-destination switch in order to traverse the global link to which this switch is connected, or when a packet reaches the destination group and needs to reach the destination node. The scheme we propose can be extended for non-shortest path intra-group routing, but the performance improvement will be very small and cannot justify the added complexity.

There are two main challenges in Dragonfly routing. The first is to decide when to use MIN and when to use VAL. The second is which intermediate group to traverse when VAL is used. The answers to these problems depend on the loads on the various global links. But these loads change very frequently, and they are unknown to the switches that need to make these decisions for each packet they receive.

Universal Globally Adaptive Load-balanced routing (UGAL) [19] was developed to make a routing decision for each packet. The decision which path to use is made by the first-hop switch, based on its estimation of the load on the network global links. The main challenge is how to efficiently and rapidly distribute up-to-date information about the state of the global links to all network switches. A solution proposed by [11], called PiggyBack (PB), is that each switch disseminates link state information of its global links to all adjacent switches by piggybacking on data packets or by sending status messages proactively on idle links. Each switch then maintains the most recent link state information for every global link in its group. A routing decision is made using this global state information in the following way. When a packet has to be transmitted, the source switch considers the load $Ldir$ on the (direct) global link leading to the destination group, the link l whose load $Lmin$ is minimum, and the average load on all links $Lavg$. If $Ldir < Lmin + Lavg + T$, where T is a constant that filters out transient imbalances, the packet is routed using MIN over the direct link. Otherwise, it is routed using VAL via the link l whose load is minimum to an intermediate group G. The idea behind this equation is that $Lavg$ is sufficiently close

to the unknown load over the link between the intermediate group G to the destination group.

We use UGAL together with PB as our benchmark protocol for lossless networks. In this case, packet are not lost, but are likely to be received out of order. The Transport protocol used for this benchmark is the standard RDMA over Ethernet (RoCE2), which uses Go-back-N. The receiver is equipped with a reordering buffer, which holds out-of-order packets before processing them and sending ACKs. A reordering buffer is not needs for LERP, despite of the fact that it can process packets that are received out of order, because the processing of these packets is not delayed and these packets are directly placed in the user memory.

The above protocol, as well as any other protocol that requires the switches to make per-packet routing decisions does not work in lossy networks for two reasons. The main reason is that congestion control is a critical components in such networks. Congestion control gives the sender a feedback about the path over which the packets have been transmitted. The feedback received by the sender is translated into "tokens", which can be consumed for transmitting new packets. For this approach to work, the feedback and the tokens must be related to the same path. If the sender receives a feedback from switches on one path and it uses its tokens for sending packets over a different path, congestion control does not work, the loss rate is high, and the performance of the throughput of the Transport protocol will be low.

Another reason why per-packet routing does not work well in lossy networks is that fast retransmission is a critical component in any transport protocol for such networks, including LERP. Fast retransmission means that if the receiver receives packet j before packet i although $i < j$, then packet i is considered lost and it is immediately retransmitted. If the packets are not routed along the same path, they are likely to be received out of order even if there is no loss, and fast retransmission does not work well.

C. Deadlock Prevention

While the main problem of using a lossless network is head-of-the-line blocking, such networks have another problem: deadlocks due to circular dependency between buffers [6], [8], [9], [18]. As an example, consider Figure 2. The figure shows three Dragonfly groups. In each group only two switches are shown. The figure shows also three local links (one inside each group) and three global links. There are three connections: blue, green and red. Each connection starts and ends in a host (circle).

Due to the red flow, the ability of buffer $b2$ to send packets depends on the progress of $b3$. We denote this dependency by $b2 \rightarrow b3$. Also due to the red connection, we have $b3 \rightarrow b4$, $b4 \rightarrow b5$, $b5 \rightarrow b6$ and $b6 \rightarrow b7$. We can summarize that due to the red connection: $b2 \rightarrow b3 \rightarrow b4 \rightarrow b5 \rightarrow b6 \rightarrow b7$.

In the same way, due to the blue connection, we have $b6 \rightarrow b7 \rightarrow b8 \rightarrow b9 \rightarrow b10 \rightarrow b11$, and due to the green connection, we have $b10 \rightarrow b11 \rightarrow b12 \rightarrow b1 \rightarrow b2 \rightarrow b3$.

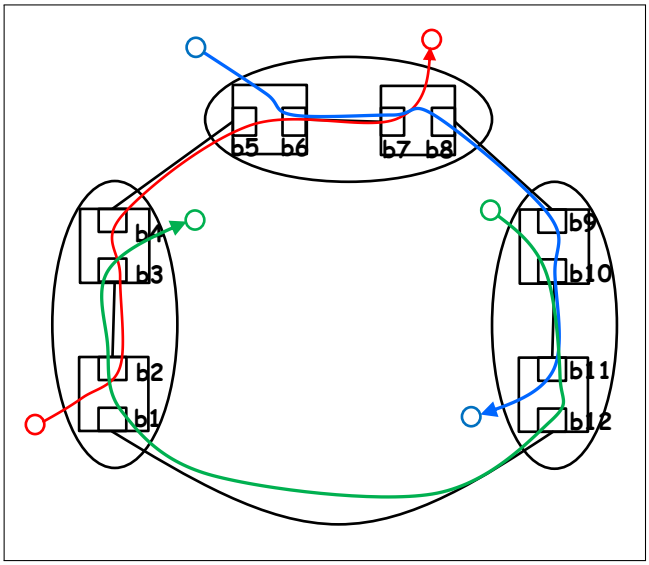


Fig. 2: Deadlock example in a Dragonfly network. Three connections (red, blue and green) have a circular dependency: if PFC blocks one of the connections it may lead to blocking the other two connections as well, resulting in a deadlock.

Consequently, the three connections form the following circular dependency: $b1 \rightarrow b2 \rightarrow b3 \rightarrow b4 \rightarrow b5 \rightarrow b6 \rightarrow b7 \rightarrow b8 \rightarrow b9 \rightarrow b10 \rightarrow b11 \rightarrow b12 \rightarrow b1$. A circular dependency is an indication for a potential deadlock.

The most common approach for preventing deadlocks is to use multiple virtual circuits (VCs), and to allocate a different pool of L2 buffers to each VC [14]. The example above can be avoided by using three VCs. While employing numerous VCs can be an effective method to prevent deadlocks, it comes with several significant drawbacks. Firstly, the allocation of buffers to multiple VCs may lead to inefficient utilization. Secondly, the increased complexity of managing multiple VCs can lead to higher overheads in terms of both hardware and software resources. Moreover, the added complexity can also complicate network configuration and maintenance, making it more challenging to diagnose and resolve network issues. Finally, the scalability of the network may be hindered, as the number of VCs required to maintain deadlock-free operations can grow with the size of the network. Therefore, while multiple VCs can be a solution to deadlocks, they also introduce trade-offs that must be carefully considered in the design and management of lossless networks.

Since the routing scheme proposed in the paper does not require a lossless network, it does not need to address the deadlock problem described above.

III. Telemetry-Based Non-Shortest Path Routing

As discussed in the previous sections, we need to find a routing protocol that, on one hand, guarantees that all the packets of the same connection are transmitted along the same path, while, on the other hand, decides which path to use according to the actual traffic in the network. Our proposed

routing protocol is different from the previous non-shortest path protocols in several aspects:

- The decision whether to use a shortest or non-shortest path, and which path to use, is made by the sending NIC. The switches need to follow the instructions of the sender, which are given in a special routing field in the packet's header, and they do not need to consider the load in the network.
- The sender uses a telemetry protocol in order to explore multiple potential paths, both shortest and non-shortest, and to decide which of them to use.
- All the packets of the same connection are routed over the same chosen path. If the sender decides that the chosen path is not good enough, it can reroute the connection, by instructing the switches to use a different path.

We use a lightweight measurement protocol that we previously introduced in [16]. This protocol allows a source node to measure the one-way queuing delay on the forward path to a given destination. From this delay, the source learns about the load of the path, and decides which path to use. A brief description of the lightweight measurement protocol follows.

The protocol uses a request-response handshake that is invoked by a source node to a given destination: the source sends a Path Test Message (PTM) to the destination, which responds with a Path Test Reply (PTR). The PTM and PTR are short messages that include an **Accumulated Queuing Delay** field, representing the sum of the queuing delay values in all the switches along the path taken by the packet. A transit node that receives the PTM or PTR updates the Accumulated Queuing Delay field by adding its local transit delay.

The Accumulated Queuing Delay is incorporated in a fixed-size shim header that can be piggybacked onto control plane probe packets and/or onto data packets. For example, in IPv6 the shim header can be incorporated using an IPv6 extension header that includes the Transit Measurement Option[3], as shown in Figure 3.

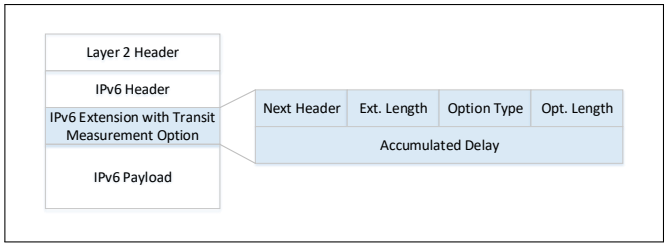


Fig. 3: Transit Measurement (TM) Option in IPv6.

The PTR includes the Accumulated Queuing Delay field of the corresponding PTM[4]. Thus, at the end of the PTM-PTR exchange the source node has the accumulated delay of all the queues for the forward path. Notably, this is obtained without relying on clock synchronization between the participating devices.

In addition to using explicit PTM and PTR messages, the delay field can be piggybacked onto data packets and reflected in ACK packets, thus eliminating the need for control plane messages and allowing continuous measurement. The Accumulated Queuing Delay field allows the source to compare the load on different paths, and to choose the best, less congested, path. This is in contrast to measuring and relying on round-trip time (RTT) metrics, which contain both the queuing time and propagation time for both the forward path and the backward path.

Disabling PFC is the key factor for improving the performance using the proposed scheme. To minimize the loss rate, each connection uses an ECN-based congestion control, which is very similar to DCTCP [4]. Our evaluation includes a comparison between using the standard Go-back-N RDMA and using our own Selective-Repeat variant, called Lightweight Efficient Retransmission Protocol (LERP).

The scheme is invoked when the sender has many packets to send to the same destination. This means that it either has many RDMA WQEs (Work Queue Elements) on the same connection, or even a single WQE of a long message, which is a typical machine learning scenario. The sender starts by sending the first packets over a random shortest path ($val\_id = undefined$). In addition, the sender sends PTM (Path Test Message) packets through a few random alternative, shortest and non-shortest, paths. Each PTM is responded by a PTR (Path Test Reply), sent by the receiver. The sender waits for all the PTRs to be received. It is not enough to wait only for the first PTR, assuming that the first PTR returns over the best path, because the latency of the returned PTR depends not only on the latency of the forward path's queues, but also the latency of the return path's queues, which is irrelevant.

To tolerate a possible PTM/PTR loss, the sender maintains a timeout after which it does not wait for additional PTRs. To this end, the time until the first PTR returns is considered as R and the sender waits an additional $2R$. If all the PTRs arrive between $[R, 3R]$, there is no need to wait for the expiration of the timeout. However, if the timeout expires, the best path is selected from the paths for which a PTR arrived.

If an alternative path is found with a better forward queueing delay, the sender reroutes the connection in the following way. In general, the rerouting approach depends on whether the network is lossy or lossless. We describe here the lossy version, as the lossless version requires the receiver to be aware of the process and to maintain a reordering buffer. In the lossy version, the sender simply stops using the old path and starts using the new path with slow-start. The receiver is not aware of rerouting, and it does not maintain a reordering buffer. To ensure that good packets are not dropped due to rerouting, the sender waits some time in idle state before the transition to the new path, and does not use any path. This time is equal to the difference between the latency of the old path and the new path. The impact of this waiting time on the performance is negligible.

[3]Submitted as an IETF Internet Draft [17].

[4]In addition, the PTR has a field for the Accumulated Queuing Delay of the PTR itself. This allows to measure the total queuing delay of the reverse path. But this is not needed for our telemetry-based routing protocol

An alternative path is considered better than the original one if its latency is shorter than $[the\ latency\ of\ the\ original\ path] \cdot \Delta 1$ and the difference between the two latencies is larger than $\Delta 2 \cdot T$, where T is the time to transmit an MTU packet, where $\Delta 1$ and $\Delta 2$ are constants. Based on our simulations, we heuristically chose the values for these two constants, $\Delta 1$ was 0.8 and $\Delta 2$ was 75. The latency of the used path is continuously measured, using the piggybacked delay field in every data packet, and is reported in the corresponding ACK. In addition to the initial PTMs and the piggybacked delay fields, a PTM is sent every $max\{cwnd, \Delta 3\}$ data packets. Our simulations revealed that a reasonable value for $\Delta 3$ is 100. PTMs are sent in a round robin fashion over all candidate paths. The set of candidate paths consists of the shortest path as well as all the non-shortest paths going through the different groups. This means that that there are $|G|-2+1 = |G|-1$ candidate paths, where $|G|$ is the number of groups. $|G|-2$ is the number of possible intermediate groups in a non-shortest path, and the "$+1$" accounts for the shortest path.

For ECMP, it is assumed that path selection is performed by the network devices based on the IP/UDP 5-tuple. Thus, for RoCEv2, when the sender use one shortest path and it wants to test another shortest path, it can vary the UDP source port number, causing the network to randomly select one of the shortest paths (which, with some probability, is different than the previous one).

In our scheme, to allow the sender to select a non-shortest path, we allocate the most significant bit in the UDP source port number, which determines whether the packet should be forwarded through a shortest path or through a non-shortest path. The remaining 15 bits in the field determine which shortest or non-shortest path to use. If the first bit is 0, the switches run a hash function on these 15 bits to choose one of the shortest paths. If the first bit is 1, the switches run a hash function on these 15 bits to choose one of the non-destination groups through which the non-shortest path traverses. The sender does not need to be aware of the topology, or to be aware of the connection between a specific UDP source port value and a specific path. However, the UDP source port values used in the PTM packets allow the source to determine the chosen path by choosing the UDP source port of the PTM with the best performance. Figure 4 illustrates an example in which a packet from a host in group 0 is forwarded to a host in group 6. The figure shows two possible paths: a shortest path (purple), in which the packet is forwarded directly from group 0 to group 6, and a non-shortest paths, in which the packet is forwarded through the intermediate group 1.

IV. Evaluation

A. Simulation Description

This section presents ns3 simulation results for the proposed telemetry-based non-shortest path routing scheme. It also compares the performance of the proposed scheme to that of previous schemes.

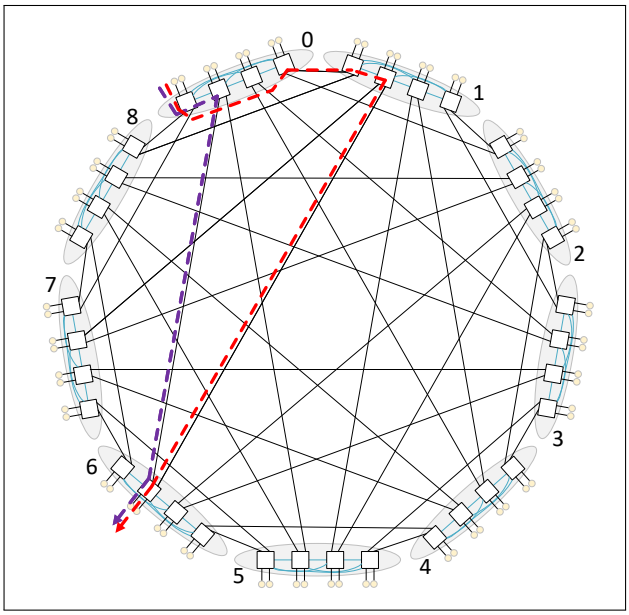


Fig. 4: Dragonfly: shortest (purple) and non-shortest path (red).

Traffic distribution. The different routing protocols are evaluated for three traffic workloads. The first is a uniform distribution workload (UNF), where each source is randomly and uniformly chosen from all the hosts, and each destination is randomly and uniformly chosen from all the hosts that are not in the source's group. In the second workload, called adversarial 1 (ADV1), two groups are randomly and uniformly chosen, Group1 and Group2. Then, for every source-destination pair, the source host is randomly and uniformly chosen from Group1 and the destination host from Group2. This implies that all the traffic in this case flows from Group1 to Group2. In the third workload, called adversarial 2 (ADV2), one group is randomly and uniformly chosen, Group1. Then, for each source-destination pair, the destination is randomly and uniformly chosen from Group1 and the source is is randomly and uniformly chosen from all other groups.

Number of source-destination pairs. In each simulation run, N pairs of source-destination servers are drawn from different groups. Each pair represents an RDMA connection with data transmitted from the source to the destination. The simulation is repeated between 10 and 20 times, and then averaged, to ensure consistent results. To draw a graph of the throughput versus the load, we use different values of N, between two and a few thousands.

Throughput measurement. The throughput is measured when all connections are active and congestion control is stable. During this period, the throughput is roughly constant. The idea is to ignore temporary conditions that happen at the beginning and the end of the simulation run.

Topologies. To analyze the scalability of the proposed protocol, three Dragonfly topologies were used in the simulation:

- **D1:** 9 groups, 4 switches per group, 7 ports per switch

Topology	Number of groups	Hosts per group	Total number of hosts	UNF: num. of pairs	ADV1: num. of pairs	ADV2: num. of pairs
D1	9	8	72	4608	64	512
D2	19	18	342	110808	324	5832
D3	33	32	1056	1081344	1024	32768

TABLE II: Dragonfly topologies and the number of source-destination permutations.

and 2 hosts per switch. Using the standard Dragonfly notations, defined in Section II, for this topology $h = 2$, $a = 2 \cdot h = 4$, and $p = h = 2$. In this topology there are 72 hosts and 252 switch ports.

- **D2:** 19 groups, 6 switches per group, 11 ports per switch and 3 hosts per switch. Thus, $h = 3$, $a = 2 \cdot h = 6$, and $p = h = 3$. In this topology there are 342 hosts and 1254 switch ports.
- **D3:** 33 groups, 8 switches per group, 15 ports per switch and 4 hosts per switch. Thus, $h = 4$, $a = 2 \cdot h = 8$, and $p = h = 4$. In this topology there are 1056 hosts and 3960 switch ports.

Table II summarize the Dragonfly topologies we use. In addition, it indicates for each topology the number of source-destination pairs. These numbers are computed in the following way:

- For UNF: the number of pairs is equal to $total\_hosts * (total\_host–hosts\_per\_group)$.
- For ADV1: the number of pairs is equal to $hosts\_per\_group * hosts\_per\_group$.
- For ADV2: the number of pairs is equal to $(number\_of\_groups–1) * hosts\_per\_group * hosts\_per\_group$.

The compared schemes. The following four schemes have been simulated and compared in this section:

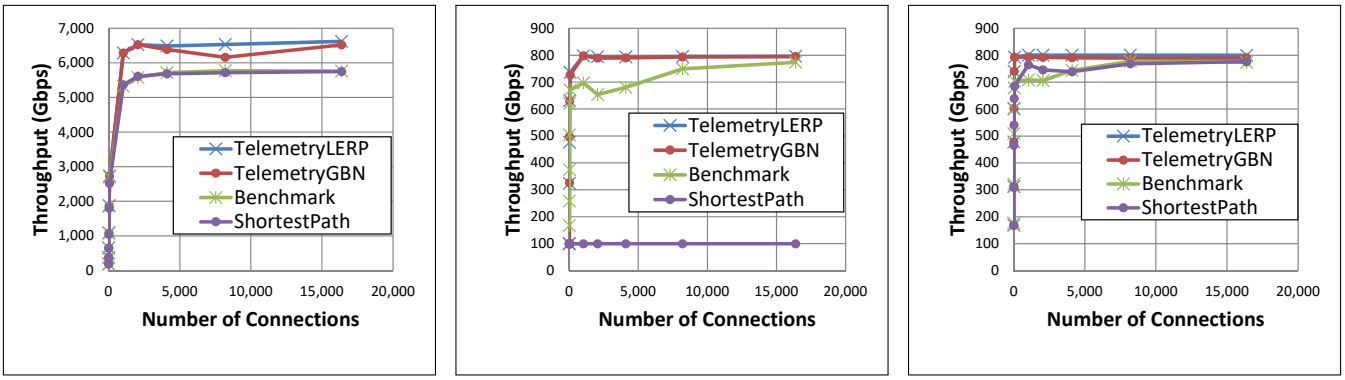
- **TelemetryLERP:** The telemetry-based routing scheme, as described in Section III, with LERP as a selective-repeat Transport (RDMA) protocol and without PFC. For this scheme, an ECN-based congestion control protocol, which is similar to DCTCP [4], is used. Note that the use of selective repeat obviates the need for the sender to wait before the transition to the new path, which was described in Section III.
- **TelemetryGBN:** The telemetry-based routing scheme, as described in Section III, with the standard RoCEv2 go-back-N as a Transport protocol. For this protocol PFC is disabled and an ECN-based congestion control protocol, which is similar to DCQCN [21], is used.
- **Benchmark:** A scheme based on an ideal version of the PB protocol, as described in Section II. In this ideal version, each switch knows the exact load on each global link in its group, even if this is not its local link. Since this scheme uses PFC and adaptive routing, we implemented virtual circuits in order to prevent PFC deadlocks, as described in Section II-C, and also a reordering receiver buffer because different packets can be routed on different routes. Although this scheme uses a lossless network, we implemented an ECN-based congestion control, which is similar to DCQCN [21], in order to reduce the impact of PFC head-of-the-line blocking.
- **ShortestPath:** The shortest path routing protocol, as described in Section II. This scheme also uses PFC. Thus, we implemented virtual circuits in order to prevent PFC deadlocks, as described above, and also a reordering receiver buffer because different packets can be routed on different routes. We also implemented an ECN-based congestion control, which is similar to DCQCN [21], in order to reduce the impact of PFC head-of-the-line blocking. The main difference between this scheme and Benchmark is that this scheme always route inter-group packets over a single direct global link, whereas Benchmark considers the load on this and other global links, and may decide to route via intermediate groups.

B. Simulation Results

Figures 5-7 depcit the throughput of the four protocols that were considered for topologies D1-D3 under the various workloads (UNF, ADV1 and ADV2). Note that the curves of TelemetryLERP and TelemetryGBN are almost aligned in most of the figures. The TelemetryLERP curve is denoted by 'X' markers, while TelemetryGBN is denoted by a red line.

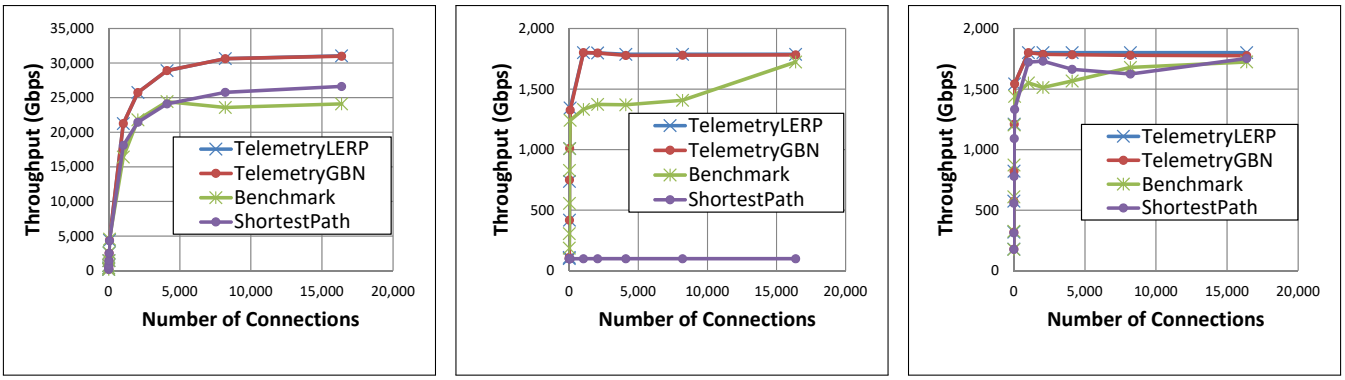
It is evident that the proposed TelemetryLERP performs significantly better than both ShortestPath and Benchmark. Figure 5a shows that for medium and high loads (1000 connections or more), the improvement of TelemetryLERP is over 15%, attributed to the fact that PFC is not used and the high head-of-the-line-blocking penalty is avoided. Note that in addition to this throughput improvement, we observed in our simulations that our protocol also significantly reduces the number of buffers in the switches because PFC deadlocks are not possible and VCs are not needed. In some cases the throughput of TelemetryLERP is slightly better TelemetryGBN, since using LERP allows quicker recovery from occasional losses than GBN.

Figure 5b illustrates the throughput in the ADV1 workload, in which traffic is sent between two randomly chosen groups. The ShortestPath curve yields a constant throughput of 100Gb/s, as there is a single 100Gb/s link between every two groups, and only this link can be used in this routing approach. The other three protocols use non-shortest paths as well. When using non-shortest paths, the maximum throughput for ADV1 is 800Gb/s: 100Gb/s for the direct link between the source and destination groups, and 100Gb/s when using each of the other seven groups as an intermediate group. It is evident that the TelemetryLERP achieves this upper bound very quickly. This protocol performs over 20% better than Benchmark in low and medium loads.



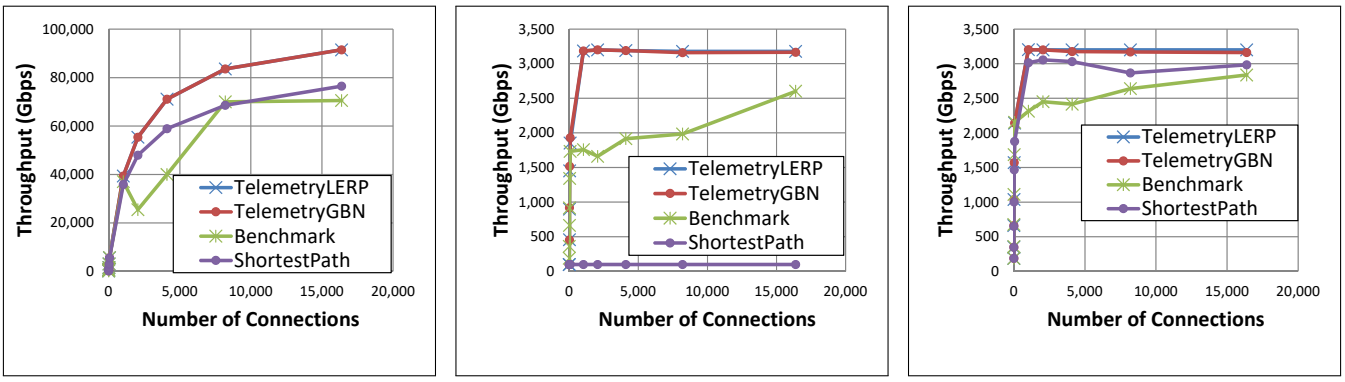
(a) D1 uniform (UNF) (b) D1 adversarial 1 (ADV1) (c) D1 adversarial 2 (ADV2)

Fig. 5: D1 topology: throughput as a function of the number of connection for each of the traffic distribution patterns (UNF, ADV1 and ADV2).



(a) D2 uniform (UNF) (b) D2 adversarial 1 (ADV1) (c) D2 adversarial 2 (ADV2)

Fig. 6: D2 topology: throughput as a function of the number of connection for each of the traffic distribution patterns (UNF, ADV1 and ADV2).



(a) D3 uniform (UNF) (b) D3 adversarial 1 (ADV1) (c) D3 adversarial 2 (ADV2)

Fig. 7: D3 topology: throughput as a function of the number of connection for each of the traffic distribution patterns (UNF, ADV1 and ADV2).

Figure 5c shows the throughput of the four considered protocols for D1 and the ADV2 traffic workload, in which traffic is transmitted from all the groups to a random destination group. TelemetryLERP and TelemetryGBN have the best performance, reaching the maximum possible throughput of 800Gb/s. The throughput of the ShortestPath approach in this workload is just slightly lower than TelemetryLERP, as the head-of-line blocking scenarios that affected the ShortestPath

Traffic distribution pattern	Load (light/heavy)	Topology D1	Topology D2	Topology D3
UNF	light load	16%	20%	78%
UNF	heavy load	13%	29%	30%
ADV1	light load	20%	35%	92%
ADV1	heavy load	3%	3%	22%
ADV2	light load	12%	18%	38%
ADV2	heavy load	3%	5%	11%

TABLE III: Summary of the TelemetryLERP throughput improvement percentage compared to the Benchmark throughput in each of the traffic distribution patterns.

approach in the UNF scenarios are negligible in the ADV2 case. The Benchmark protocol has a slightly lower throughput, as the use of adaptive routing is counter-productive in this all-to-one workload.

Figure 6 illustrates the throughput results for the three traffic workloads in the D2 topology. The results show the same trend as Figure 5 with a better improvement ratio in some cases due to the large scale. TelemetryLERP performs by up to 25% better than Benchmark in the UNF workload, by up to 30% better than Benchmark in the ADV1 workload, and up to 18% better than Benchmark in the ADV2 workload. Figure 7 shows the results for the D3 topology, with TelemetryLERP achieving up to 30% improvement compared to the Benchmark in the UNF workload, up to 100% improvement in the ADV1 workload, and up to 30% improvement in the ADV2 workload.

Table III summarizes the simulation results for all topologies and traffic patterns. The table shows the throughput improvement percentage of the telemetry-based LERP over the benchmark protocol for all topologies and for the three considered traffic distributions. We can see that by increasing the scale of the topology, the relative improvement of the telemetry-based LERP increases. For this reason, we believe that the motivation to use this protocol will increase in the future, as network topologies evolve to larger scales.

Taking the three traffic workloads into consideration, it is evident that the proposed TelemetryLERP consistently outperforms the other routing approaches for all the considered network scales.

V. Conclusion

We introduced a routing approach for data center networks that utilizes both the shortest and non-shortest paths by measuring the available path using a lightweight measurement protocol. Our approach is shown to provide higher throughput by an order of magnitude compared to shortest path routing in the Dragonfly topology, and a throughput improvement of up to 92% compared to state-of-the-art non-shortest path routing approaches.

References

[1] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. *ACM SIGCOMM computer communication review*, 2008.

[2] M. Alizadeh, T. Edsall, S. Dharmapurikar, R. Vaidyanathan, K. Chu, A. Fingerhut, V. T. Lam, F. Matus, R. Pan, N. Yadav, et al. Conga: Distributed congestion-aware load balancing for datacenters. In *Proceedings of the 2014 ACM conference on SIGCOMM*, 2014.

[3] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan. Data Center TCP (DCTCP). In *SIGCOMM*, 2010.

[4] S. Bensley, D. Thaler, P. Balasubramanian, L. Eggert, and G. Judd. Data Center TCP (DCTCP): TCP Congestion Control for Data Centers. RFC 8257, 2017.

[5] M. Besta and T. Hoefler. Slim fly: A cost effective low-diameter network topology. In *SC'14: proceedings of the international conference for high performance computing, networking, storage and analysis*, pages 348–359. IEEE, 2014.

[6] A. Gangidi, R. Miao, S. Zheng, S. J. Bondu, G. Goes, H. Morsy, R. Puri, M. Riftadi, A. J. Shetty, J. Yang, et al. Rdma over ethernet for distributed training at meta scale. In *Proceedings of the ACM SIGCOMM 2024 Conference*, pages 57–70, 2024.

[7] M. García, E. Vallejo, R. Beivide, M. Odriozola, and M. Valero. Efficient routing mechanisms for dragonfly networks. In *42nd International Conference on Parallel Processing, ICPP 2013, Lyon, France, October 1-4, 2013*, 2013.

[8] S. Hu, W. Bai, G. Zeng, Z. Wang, B. Qiao, K. Chen, K. Tan, and Y. Wang. Aeolus: A building block for proactive transport in datacenters. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, pages 422–434, 2020.

[9] S. Hu, Y. Zhu, P. Cheng, C. Guo, K. Tan, J. Padhye, and K. Chen. Tagger: Practical pfc deadlock prevention in data center networks. In *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies*, pages 451–463, 2017.

[10] IEEE. Priority-based Flow Control. *IEEE Std. 802.1Qbb*, 2011.

[11] N. Jiang, J. Kim, and W. J. Dally. Indirect adaptive routing on large scale interconnection networks. In *International Symposium on Computer Architecture (ISCA)*, 2009.

[12] G. Kathareios, C. Minkenberg, B. Prisacari, G. Rodriguez, and T. Hoefler. Cost-effective diameter-two topologies: Analysis and evaluation. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–11, 2015.

[13] J. Kim, W. J. Dally, S. Scott, and D. Abts. Technology-driven, highly-scalable dragonfly topology. In *2008 International Symposium on Computer Architecture*, pages 77–88. IEEE, 2008.

[14] P. Majumder, S. Kim, J. Huang, K. H. Yum, and E. J. Kim. Remote control: A simple deadlock avoidance scheme for modular systems-on-chip. *IEEE Transactions on Computers*, 70(11):1928–1941, 2020.

[15] Mellanox. How To Configure Adaptive Routing and SHIELD. https://community.mellanox.com/s/article/how-to-configure-adaptive-routing-and-shield, 2019.

[16] T. Mizrahi, S. Belkar, and R. Cohen. RDMA Path Selection Using Lightweight Network Telemetry. In *IEEE MedComNet*, 2025.

[17] T. Mizrahi, T. Zhou, S. Belkar, and R. Cohen. The Transit Measurement Option. Internet-Draft, Work in Progress draft-mzbc-ippm-transit-measurement-option-00, Internet Engineering Task Force, 2022.

[18] K. Qian, W. Cheng, T. Zhang, and F. Ren. Gentle flow control: Avoiding deadlock in lossless networks. In *Proceedings of the ACM Special Interest Group on Data Communication*, pages 75–89. 2019.

[19] A. Singh. *Load-balanced routing in interconnection networks*. Stanford University, 2005.

[20] Z. Wang, L. Luo, Q. Ning, et al. SRNIC: A scalable architecture for RDMA NICs. In *NSDI 23)*, 2023.

[21] Y. Zhu, H. Eran, D. Firestone, C. Guo, M. Lipshteyn, Y. Liron, J. Padhye, S. Raindel, M. H. Yahia, and M. Zhang. Congestion control for large-scale RDMA deployments. In *SIGCOMM*, 2015.