applied sciences

MDPI

Article

Enhancing Robustness of Per-Packet Load-Balancing for Fat-Tree

Chansook Lim

Department of Software & Communications Engineering, Hongik University, Sejong 30016, Korea; chansooklim@hongik.ac.kr

Abstract: Fat-tree networks have many equal-cost redundant paths between two hosts. To achieve low flow completion time and high network utilization in fat-tree, there have been many efforts to exploit topological symmetry. For example, packet scatter schemes, which spray packets across all equal-cost paths relying on topological symmetry, work well when there is no failure in networks. However, when symmetry of a network is disturbed due to a network failure, packet scatter schemes may suffer massive packet reordering. In this paper, we propose a new load balancing scheme named LBSP (Load Balancing based on Symmetric Path groups) for fat-trees. LBSP partitions equal-cost paths into equal sized path groups and assigns a path group to each flow so that packets of a flow are forwarded across paths within the selected path group. When a link failure occurs, the flows affected by the failure are assigned an alternative path group which does not contain the failed link. Consequently, packets in one flow can still experience almost the same queueing delay. Simulation results show that LBSP is more robust to network failures compared to the original packet scatter scheme. We also suggest a solution to the queue length differentials between path groups.

Keywords: fat-tree; per-packet load-balancing; packet-reordering

Citation: Lim, C. Enhancing Robustness of Per-Packet Load-Balancing for Fat-Tree. *Appl. Sci.* **2021**, *11*, 2664. https://doi.org/10.3390/app11062664

Academic Editor: Christos Bouras

Received: 15 January 2021
Accepted: 15 March 2021
Published: 17 March 2021

1. Introduction

Modern datacenter networks have common characteristics such as high bandwidth and low latency. In addition, many typical datacenter network topologies have rich path diversity and symmetry [1,2]. Since many applications that run in datacenter networks require high throughput or short flow completion time, there have been extensive research on how to achieve those objectives through load-balancing [3].

Depending on the granularity of load balancing, approaches are categorized into flow-level and packet-level. One of the major factors in deciding granularity of load-balancing is the issue of packet reordering. Flow-level load-balancing is favorable to in-order delivery, but prone to low network utilization and long flow completion time, so there have been a lot of efforts to address the issues [2–7]. Packet-level load-balancing is more effective in spreading traffic evenly than flow-level load-balancing, although it needs to address the packet reordering issue. Many proposals have been made to make packet-level load-balancing more viable by exploiting the unique characteristics of datacenter networks [8–13]. A notable approach is to exploit the topological symmetry more directly. The authors of [9] showed the effectiveness of RPS (Random Packet Spraying), which sprays packets of a flow randomly across equal-cost paths. When RPS is used in symmetric datacenter networks, the packets belonging to the same flow experience almost the same queuing delay, and consequently most of them arrive at the destination in order. The study also shows that asymmetry due to a link failure results in queue length differentials, which adversely impacts TCP performance of RPS. DRILL [12] also exploits a key characteristics of a symmetric Clos network. DRILL employs per-packet load-balancing based on queue occupancies. To handle topological asymmetry, DRILL decomposes the network into a minimal number of components, such that the paths within each component are symmetric.

DRILL assigns each flow to a component and applies per-packet load-balancing across paths inside the component.

A recent proposal for architectural fault-tolerance of data center networks [14] shows the possibility that the network topology can keep symmetric more stably. Considering this technological trend, it is important to explore how to more effectively exploit topological symmetry and high degree of path diversity.

We propose a proactive per-packet load-balancing scheme named LBSP (Load Balancing based on Symmetric Path groups) for fat-trees. In LBSP, multiple equal-cost paths are partitioned into path groups of equal size. For each flow, packets belonging to the flow are sprayed across the paths within one path group chosen based on the destination address. A notable feature is that path groups are of equal size. Hence, symmetry holds not only between paths inside a path group, but also between path groups in terms of the number of available paths for each flow. If a link failure occurs, LBSP selects an alternative path group for the flows affected by the failure so that each flow is assigned a path group which does not contain a failed link. Consequently, even in the presence of failures, for each flow, LBSP conserves symmetry of multiple paths that the packets of the flow pass through.

A strength of LBSP is that, in a normal state (with no failures), it does not have to maintain the information on mapping a flow to a path. Only when an alternative path group needs to be used, the information is maintained for the relevant address range. In addition, the simple rule on mapping a flow to a path group facilitates hardware implementation for speedup.

To evaluate LBSP, we simulated three-tier fat-trees and compared flow completion time between LBSP and the original packet scatter scheme. Simulation results show that LBSP is much more robust to network failures than the original packet scatter schemes.

Since LBSP forwards packets based on the destination address, traffic distribution between path groups can be uneven. To reduce the queue length differentials between path groups, we propose a solution that uses not only the destination address but also the input port number.

The rest of this paper is structured as follows. In Section 2, we explain the background and motivation. In Section 3, we describe our scheme LBSP in detail. Section 4 demonstrates the simulation results. In Section 5, we propose a solution to reduce the queue length differentials between path groups. Section 6 describes the related work. In Section 7, we conclude.

2. Background and Motivation

2.1. Assumption on Fat-Tree

In this paper, we consider packet forwarding in fat-trees. We use the three-tier k-ary fat-tree architecture proposed in [1]. A k-ary fat-tree has k pods, each containing two layers of $k/2$ switches. Each k-port switch (ToR switch) in the lower layer is directly connected to $k/2$ hosts. Each of the remaining $k/2$ ports is connected to $k/2$ of the k ports in the aggregation layer of the hierarchy. There are $(k/2)^2$ k-port core switches. Each core switch has one port connected to each of k pods. The ith port of any core switch is connected to pod i such that consecutive ports in the aggregation layer of each pod switch are connected to core switches on $(k/2)$ strides. We note that each ToR switch is connected with $k/2$ aggregation switches and each aggregation switch with $k/2$ core switches. In general, a fat-tree built with k-port switches supports up to $k^3/4$ hosts. A fat-tree architecture in [1] can achieve the full bisection bandwidth of a network. In this paper, we assume that $k \geq 8$ is a power of 2. The links between ToR switches and aggregation switches are homogeneous and the links between aggregation switches and core switches are homogeneous. In a fat-tree, path selection is made only in the stage of upward forwarding because downward forwarding is deterministic once upward forwarding is determined.

2.2. Per-Packet Load Balancing and Packet Reordering

Packet reordering can affect performance of TCP. Since fast retransmit of TCP degrades performance severely in multipath routing environments where out-of-order packets are produced persistently, packet-level scheduling schemes usually entail measures to handle out-of-order packets. DeTail [8], a cross-layer network stack, constructs a lossless fabric in the link layer. When the queue fills up beyond a threshold, flow control messages are propagated back to quench the source. As a consequence, packets are seldom lost due to congestion, so DeTail simply disables fast retransmit and fast recovery in TCP. Out-of-order packets are reordered at the end-hosts. In Fastpass [10], a centralized arbiter assigns a timeslot and a path to each packet with the aim of achieving nearly zero queues. The arbiter requires complete knowledge of every host's traffic demand in the network and communicates with end-hosts for each request for packet scheduling. Presto [11] mitigates packet reordering by balancing traffic load in units of flowcell whose size is set to the maximum TCP Segment Offload size (64 KB). When out-of-order packets are to be reordered at the end-hosts, the resequencing delay must be short.

2.3. Exploiting Topological Symmetry

Some proposals employ per-packet load-balancing approaches that exploit topological symmetry of data center networks. The authors of [9] demonstrated the efficacy of RPS (Random Packet Spraying), which sprays packets of a flow randomly across the equal-cost paths. As far as topological symmetry holds, all the packets belonging to the same flow experience almost the same queuing delay even if they pass through different paths. Hence, it leads to only a few out-of-order packets.

However, if the network becomes asymmetric due to failures, spraying packets across multiple paths with unbalanced traffic may cause massive packet reordering. Figure 1 shows an example in an 8-ary fat-tree. For better visibility, Figure 1 shows only the switches and links that the flows specified by three source-destination pairs pass through. If the link connecting Switch A11 and Switch T11 fails, packets of the flow from S3 to D3 may experience different queueing delay depending on which core switch they pass through. That is, packets passing through core switches C12, C13, C14, or C15 may observe shorter queueing latency than packets passing through the other core switches. As a result, the flow from S3 to D3 may suffer TCP performance degradation due to packet reordering.

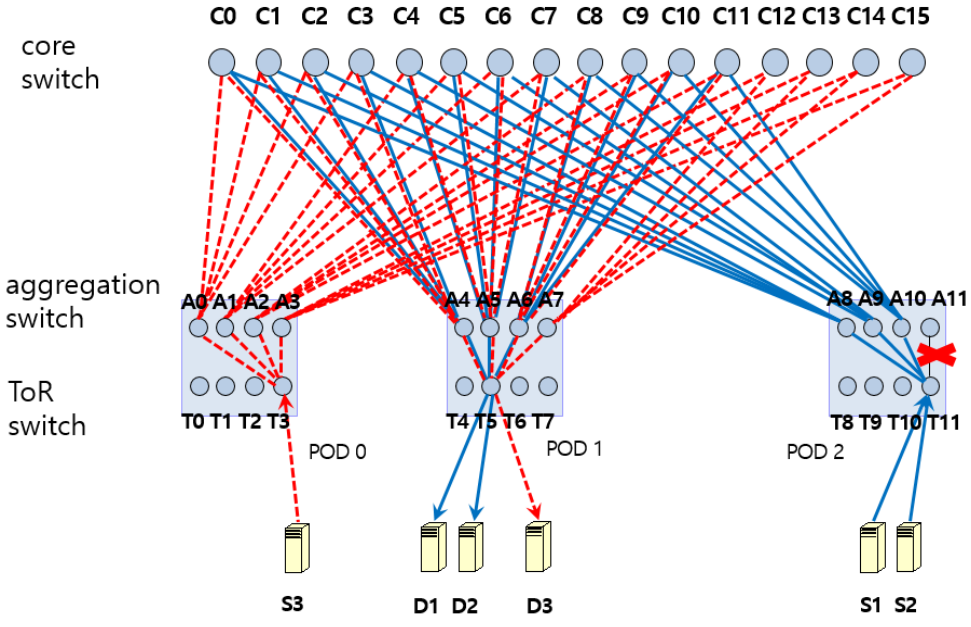


Figure 1. Illustration of a packet scatter scheme in an 8-ary fat-tree.

Hence, the approaches relying on symmetry of a network need to cope with failures which result in asymmetry. In [9], a variant of RED called SRED (Selective RED) is proposed to be used together with RPS. If a flow that cannot use all the multiple paths makes differentials of queue lengths between switches, SRED drops the packets belonging to the flow more aggressively. By doing this, SRED prevents flows that cannot use all the equal-cost paths from affecting the other "normal" flows. DRILL (Distributed Randomized In-network Localized Load-balancing) [12] employs per-packet decisions at each switch based on local queue occupancies and randomized algorithms to distribute load. DRILL compares queue lengths of two random ports plus the previously least-loaded port and sends the packet to the least loaded of these. When a failure causes topological asymmetry, DRILL decomposes the network into a minimal number of symmetric components, assigns each flow to a component, and applies load-balancing across paths inside the component. Although DRILL provides a more generalized solution, it is notable that the time complexity of constructing a labeled multigraph needed for decomposition is $O(L^2V^2)$, where L and V are the number of leaves and the number of vertices, respectively, in the case of a symmetric leaf-spine datacenter. Therefore, while decomposition is done instantaneously for small sized networks, it may take considerable time to perform decomposition for large-scaled networks.

We argue that, for topologically symmetric networks such as fat-trees, per-packet load-balancing can be enhanced at lower overhead costs by fully exploiting symmetry. If a network keeps more stably symmetric and concurrent failures are rare, it would be desirable to exploit symmetry to the full. Recent fault-tolerance-related research enables us to envision more robust network architectures, which support our approach. For instance, *shareable backup* [14] realizes fast recovery (in sub-milliseconds) from failures using backup switches. In shareable backup, the entire data center shares a pool of backup switches. When a switch or link failure occurs, the internal connections to it on all the circuit switches are reconfigured to connect to a backup switch, which replaces a failed switch to restore full capacity. This implies that, after being in a transient asymmetric state, a network can revert to a symmetric state shortly.

3. Load Balancing Based on Symmetric Path Groups (LBSP)

3.1. Basic Idea

The proposed scheme LBSP aims to achieve high TCP performance by exploiting a symmetric portion of redundant paths even when symmetry of the whole network is disturbed due to failures. The main features of LBSP are as follows.

- Initially, LBSP partitions shortest paths into fixed equal sized symmetric subsets called path groups.
- In a normal state without a failure, LBSP assigns a (default) path group to a flow based on the least significant bits of the destination address.
- In the case of failure detection, LBSP selects an alternative path group instead of the default path group for the flows affected by the failure.

LBSP partitions multiple equal-cost paths between two hosts into more than one path group of the same size. The rationale for partitioning multiple equal-cost paths is that, in the presence of network failures, we can reduce packet reordering by forwarding packets through a portion of multiple paths which are still symmetric. DRILL [12] also adopts the idea of partitioning multiple shortest paths between a source and destination pair into components. However, since DRILL partitions a network into a minimal number of symmetric components, the component sizes may be different. In addition, when there is no failure in a fat-tree, the whole network has only one component. In contrast, LBSP partitions the shortest paths into fixed equal-sized symmetric path groups. This enables the topological symmetry property to be satisfied not only between paths inside a path group but also between path groups. This reduces the complexity for handling asymmetry.

For each flow, a path group is selected according to a rule based on the destination address. This allows every packet belonging to the same flow to use the same path group.

For ease of explanation, we consider an 8-ary fat-tree, although our scheme is scalable to larger fat-trees. In the 8-ary fat-tree depicted in Figure 2, consider a packet going from Host 0 to Host 112. If no restriction is imposed on path selection, the packet can choose one of the 16 shortest upward paths which pass through different core switches. Now we partition the links between the Switch T0 and the aggregation Switches A0, A1, A2, and A3 into two link groups, which are indicated by the red solid lines and the red dashed lines in Figure 2. Similarly, for each of the aggregation Switches A0, A1, A2, and A3, the links between the aggregation switch and its neighbor core switches are partitioned into two link groups, represented by the blue solid lines and the blue dashed lines in Figure 2. If we combine the first partition with the second partition, as shown in Table 1, the 16 symmetric upward paths are partitioned into four path groups, each of which consists of four paths. The flow from Host 0 to Host 112 is assigned one of the four path groups. For simplicity, we assign a link group to each flow based on the parity of the least significant bits of the destination address. Within a link group, packets are forwarded in a round-robin manner. This method guarantees that the flows selecting the same path group observe almost the same queue occupancies.

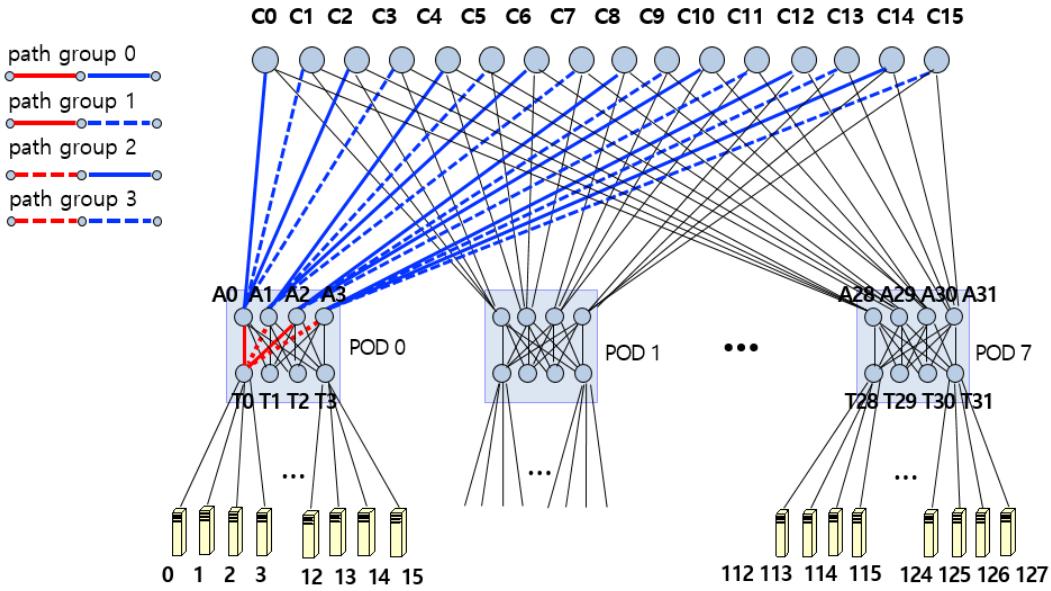


Figure 2. Illustration of partitioning 16 equal-cost paths into four symmetric path groups.

Table 1. Path groups and link groups in 8-ary fat-tree.

Path Group	Link Group (ToR Switch–Aggregation Switch)	Link Group (Aggregation Switch–Core Switch)
0	0	0
1	0	1
2	1	0
3	1	1

If a link failure occurs, LBSP tries to select an alternative path group which does not contain failed links. Figure 3 illustrates selecting an alternative path group. Suppose that the default path group for the flow S2 → D2 is the paths that pass through Switches A9 and A11. If Switch T11 detects the link failure between Switch A11 and itself, it assigns the flow an alternative link group that passes through Switches A8 and A10. Then, at Switches A8 and A10, the flow S2 → D2 is assigned one of the two link groups. Suppose that the flow S2 → D2 is assigned the same path group as the flow S1→D1 according to the least significant bits of the destination address. Then, as indicated by the green dashed lines, the paths for the flow S2 → D2 are as follows: S2 → T11 → {A8,A10} → {C0,C2,C8,C10}

→ {A4,A6} → T5 → D2. It is important to note that the two flows S1 → D1 and S2 → D2 using the same path group will observe almost the same queue lengths.

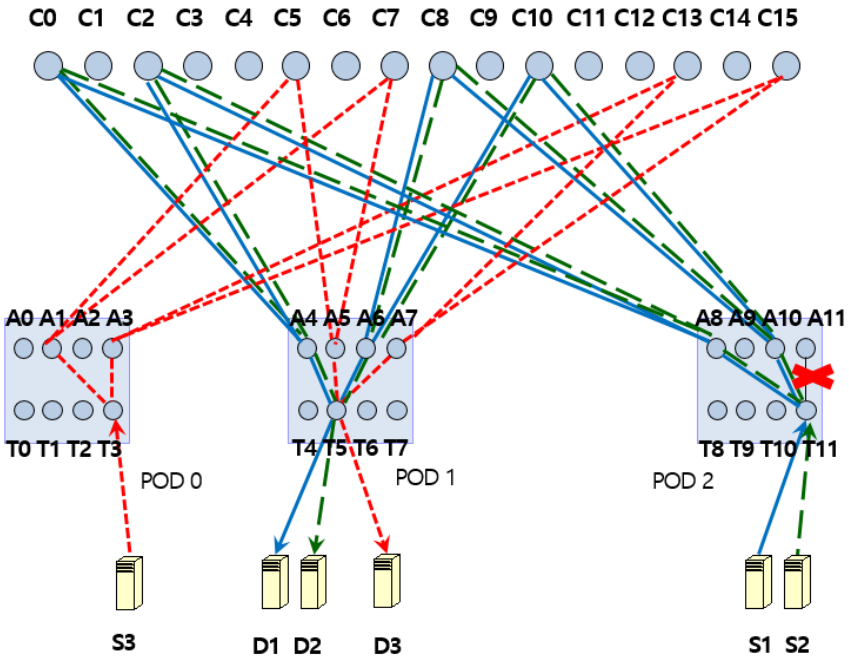


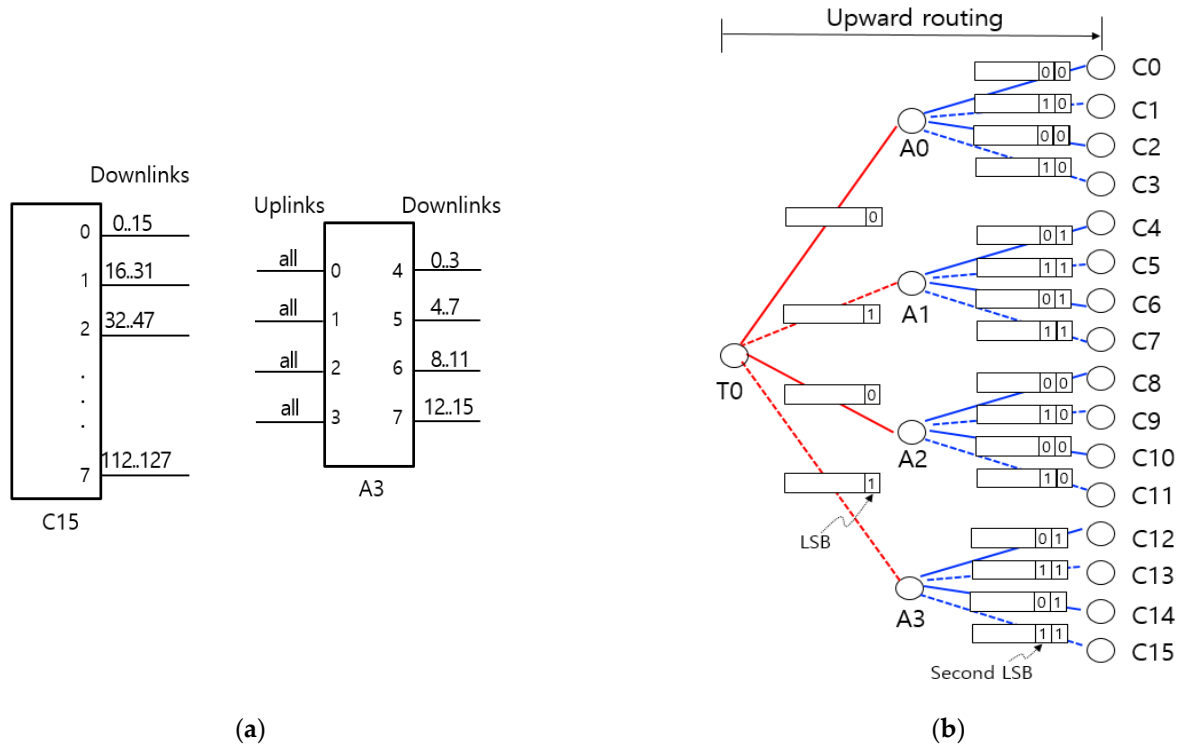
Figure 3. Illustration of LBSP (Load-Balancing based on Symmetric Path groups) selecting an alternative path group when a failure occurs on the link between aggregation Switch A11 and ToR Switch T11.

3.2. Partitioning Paths and Assigning Path Groups

The practical design of LBSP was inspired by Flexible Interval Routing (FIR), which Gomez et al. used to implement a routing algorithm for fat-trees [15]. We associate each output port with the range of the destination addresses. The range indicates all the destination hosts which are reachable from the output port. Since each upward output port leads to every host in a fat-tree, the range of destination addresses contains all the host addresses. In contrast, each downward output port has a limited range. Figure 4a illustrates the ranges associated with the output ports of an aggregation switch and a core switch in a fat-tree shown in Figure 2.

In a fat-tree, once upward paths are selected, the downward paths are deterministic. In an 8-ary fat-tree, a flow from a pod to another pod has 16 shortest paths. LBSP partitions all the 16 shortest paths into four path groups using the parity of each port number of each switch on the upward paths. Since partitioning the shortest paths in this way makes consecutive packets pass through different ports in a ToR switch, it helps reduce latency when a round-robin manner is used.

LBSP selects a path group for each flow based on the least significant bits of the destination address. That is, for upward forwarding, each ToR switch and aggregation switch selects uplinks based on the LSB and the second LSB, respectively. Across paths within a link group, packets of a flow are forwarded in a round-robin manner. Hence, flows can use increased bandwidth with a higher probability compared to flow-level load balancing. For downward forwarding, each switch forwards a packet according to the range of destination address. Figure 4b illustrates partitioning paths and assigning path groups to source-destination pairs according to the least significant bits of the destination address. If each output port has a mask register that indicates which bit of the destination address is to be compared, then in a normal state without failures, LBSP does not have to keep the information on which specific flow is assigned to which path group.



(**a**) (**b**)

Figure 4. (**a**) Illustrations of destination address ranges associated with downlink ports of the core switch C15 and the aggregation Switch A3 in the fat-tree shown in Figure 2; and (**b**) an illustration of partitioning 16 shortest paths into four path groups and assigning path groups according to the least significant bits of the destination address.

3.3. Handling Asymmetry due to Failures

3.3.1. Failure Notification

For failure detection, each switch sends hello messages to its neighbor switches. On detecting a failure, it floods a failure message in the network. A failure message contains the information on the failure type and the address range affected by the failure. With the failure types, a switch can figure out which link group should not be selected. Table 2 summarizes the failure types.

Table 2. Failure Types.

Type	Type of Failed Link	Affected Link Group
0	ToR switch–aggregation switch	Link Group 0
1	ToR switch–aggregation switch	Link Group 1
2	aggregation switch–core switch	Link Group 0
3	aggregation switch–core switch	Link Group 1

A failure message from a switch detecting a failure of a downlink contains the address range associated with the port incident to the failed link. In contrast, a failure message from a switch detecting a failure of an uplink contains the address range of all the hosts. In the fat-tree shown in Figure 2, suppose that the link between A31 and T31 is down. The failure message sent from A31 will contain the address range of [124 . . . 127] and the failure type of 1. Suppose that Switch T3 receives the failure message. Then, T3 cannot select Link Group 1 for the flows with the destination address in the range 124–127.

In the case of a three-tier fat-tree, the failure messages take at most three hops to reach all the aggregation switches and ToR switches that need the failure information to select a link group.

3.3.2. Modifying the Forwarding Tables

As previously described, when the network has no failure, upward forwarding in LBSP is performed as predefined. If a switch receives a message about a failure which affects the flows passing through itself, it modifies the forwarding table so that an alternative link group is chosen for packets with the destination addresses within the range indicated in the failure message.

For instance, consider a flow from Host 15 to Host 115, as shown in Figure 5. Since the LSB of the destination address (115) is 1, Switch T3 forwards packets using output Ports 1 and 3. Similarly, since the second LSB of 115 is 1, Switches A1 and A3 selects output Ports 1 and 3 for the flow. Suppose that the link between the aggregation Switch A31 and the core switch C15 is down. The failure message conveys the information that the affected address range is [112 . . . 127] and the failure type is 3. It means that the flows with the address in the range of 112–127 are affected by the failure and cannot be forwarded through Link Group 1 between an aggregation switch and a core switch. Note that failure Type 3 does not affect the forwarding tables of ToR switches. On receiving the failure message, aggregation Switches A1 and A3 modify their forwarding tables as follows.

Destination Address Range	Link Group (Output Ports)
112 . . . 127	Link Group 0 (Ports 0, 2)

Suppose that Host 15 sends packets to Host 115 at that point. Since the LSB of 115 is 1, Switch T3 forwards packets using Ports 1 and 3 as predefined. However, Switches A1 and A3 select Ports 0 and 2 instead of Ports 1 and 3, because the modified forwarding table indicates that packets whose destination addresses are in the range [112 . . . 127] must be forwarded through output Ports 0 and 2.

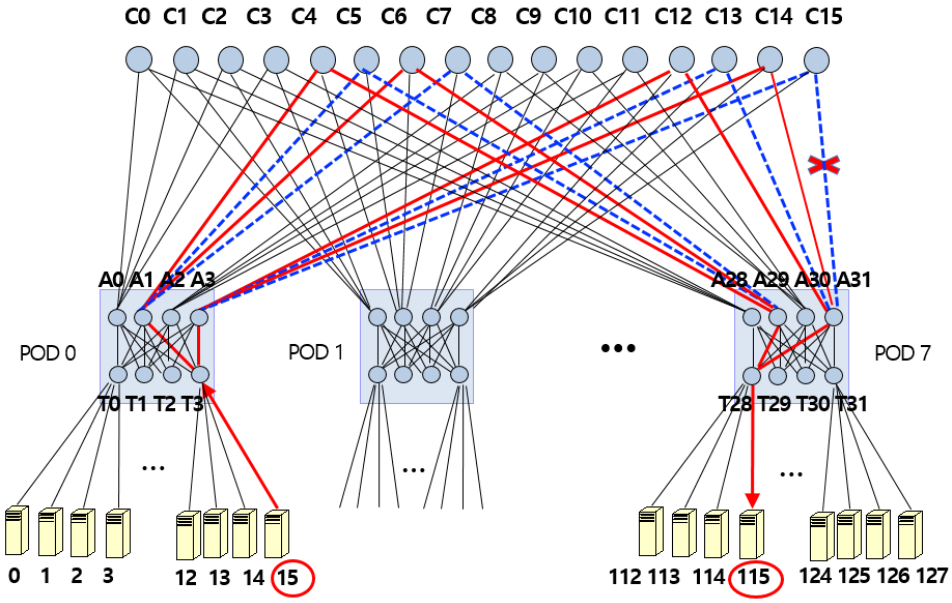


Figure 5. LBSP selects an alternative path group on detecting a failure of the link between core switch C15 and aggregation Switch A31. The red lines indicate the paths that the flow from Host 15 to Host 115 pass through in this case.

3.4. Discussions

LBSP is economical in that it does not require a lot of state information and can be implemented with simple hardware. In a normal state without a failure, LBSP can forward packets upwards by simply mapping the least significant bits of the destination address to a path group. The bit comparison needed can be implemented by using the mask register as in FIR [15] so that only the specific bit(s) of the destination address is compared. Only

when a failure is notified, the entries for the corresponding address range and link group are maintained in the forwarding table. To indicate whether the forwarding table needs to be searched, the restriction register proposed for FIR in [15] can be used.

While we describe the LBSP scheme using 8-ary fat-trees for simplicity, LBSP performs better for a higher fan-out ratio (k). For k-ary fat-trees with $k \geq 16$, we have two choices on partitioning shortest paths: either having more path groups or having more paths in each path group. Having more path groups enhances robustness to failures because each switch has more alternative path groups. On the other hand, having more paths in a path group helps reduce the flow completion time for each flow.

Effectiveness of LBSP becomes more evident in fat-trees with high oversubscription ratios, where queues build up more. If a failure occurs in such a network, network asymmetry and queue buildup may jointly cause severe packet reordering. LBSP is effective also in the presence of a degraded link. When packets in a TCP flow are spread evenly across multiple paths, performance of the flow depends on the slowest path. LBSP deals with a degraded link and a disconnected link in the same way by selecting an alternative path group.

Some might be concerned that selecting an alternative path group in the presence of a failure can significantly reduce network utilization. The proportion of flows that need to select an alternative path due to a link failure is not significant. For k-ary fat-tree, the proportion of the source-destination pairs to be rerouted due to a link failure between a ToR switch and an aggregation switch is $\left(\frac{k^3}{4} - \frac{k}{2}\right)\frac{k}{2} / \begin{pmatrix} k^3/4 \\ 2 \end{pmatrix}$ Hence, in the illustrative 8-ary fat-tree of Figure 3, if a failure occurs on the link between a ToR switch and an aggregation switch, the flows which communicate with the four hosts connected to the ToR switch are affected, and the proportion of the source-destination pairs to be rerouted is approximately 6%. Hence, network utilization is not significantly affected.

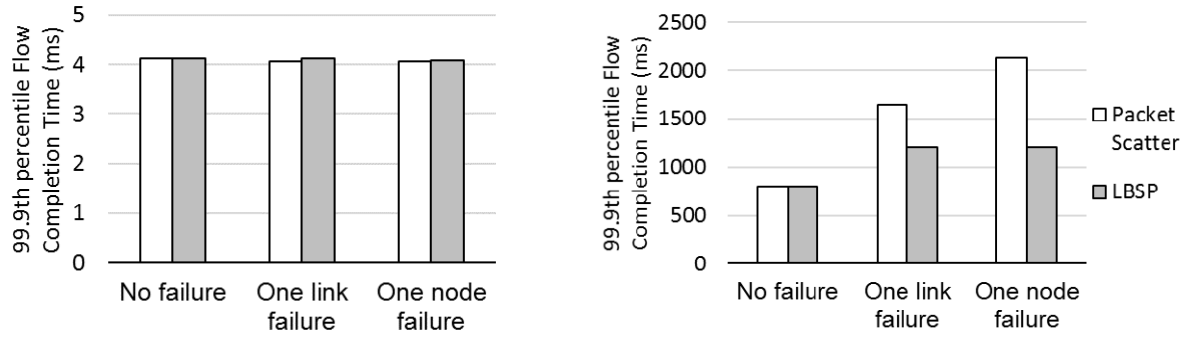
While LBSP assigns a path group to a flow simply based on the least significant bits of the destination address, entropy can be increased by hashing several fields in the packet header to a path group as in ECMP (Equal-cost Multiple Path). ECMP, which is the de facto flow-level load balancing scheme, distributes flows across available multiple paths by statically hashing some fields in the packet header. Hence, ECMP does not require state information, either. It is known that ECMP works well for a large number of flows with sufficient entropy but that it may underutilize a network when a few long flows collide on some path [6]. We expect that, by employing ECMP-like hashing and utilizing multiple symmetric paths, LBSP can mitigate the shortcomings of ECMP without causing excessive packet ordering.

4. Simulation Results

We performed simulations for an 8-ary fat-tree and a 16-ary fat-tree using ns-2. All links have bandwidth of 1 Gbps, queue size of 250 packets, and propagation delay of 50 µs. At the transport layer, we used TCP-Reno and set the retransmission timer to 20 ms. For tests with network failures, we simulated link failures, node failures, and degraded links. For link failure tests, we disconnected a link connecting an aggregation switch and a ToR switch. For node failure tests, we made an aggregation switch down. A node failure causes multiple link failures simultaneously. We measured the flow completion time, which is important in datacenter networks.

Short-lived flows in an 8-ary fat-tree

First, we simulated all-to-all short-lived flows of size 32 KB with load factor 0.8 repeatedly for about 3.3 s. In total, 128 flows were generated using a random permutation. Figure 6a shows the 99.9th percentile of the flow completion times. Neither a link failure nor a node failure affects the flow completion time of TCP flows significantly. Since the flows are short, the queues do not build up enough to cause packet reordering or packet drops. Hence, the original packet scatter scheme, which distributes packets using more paths, performs marginally better than LBSP.



(**a**) (**b**)

Figure 6. Flow completion time in tests for an 8-ary fat-tree: (**a**) result of all-in-all short-lived flow tests; and (**b**) result of long-lived flow tests.

Long-lived flows in an 8-ary fat-tree

To validate the functionality of LBSP for long-lived flows, we chose to use a similar test scenario to the one used in [6] considering the limit of our simulation environment. We generated a small number of 100 MB flows from hosts on ToR switches in two specific pods transmitting to hosts on another specific ToR switch in a different pod. We simulated four flows with size 100 MB repeatedly for about 20 s. The source and destination hosts were selected as depicted in Figure 3. For the source-destination pairs <S1, D1>, two flows were generated. For each of the other source destination pairs <S2, D2> and <S3, D3>, one flow was generated. This scenario is not advantageous to LBSP because <S1,D1> and <S2,D2> were selected so that, under the link failure, the three flows from S1 and S3 take the same paths. Figure 6b shows the 99.9th percentile of flow completion times for the original packet scatter scheme and LBSP. When there is no failure in the network, both schemes perform almost equal. When a failure occurs at a link between a ToR switch and an aggregation switch, as shown in Figure 3, LBSP performs better than the original packet scatter scheme despite the disadvantageous scenario. For the original packet scatter scheme, the flow from S3 suffers the greatest degradation of performance. The reason is that the packets from S3 arrive at D3 out of order due to different queuing delays. In LBSP, however, the flows seldom experience packet reordering. When an aggregation switch in the source pod of the three flows fails, the gap between two schemes becomes greater than that shown in Figure 6b.

Long-lived flows in a 16-ary fat-tree with oversubscription ratios

We simulated one link failure in a 16-ary fat-tree. We partitioned shortest paths into the same number of path groups as in an 8-ary fat-tree. Consequently, since each path group has more paths, the probability of packet loss due to congestion decreases. We disconnected a link connecting a ToR switch and an aggregation switch as in the tests for an 8-ary fat-tree. We generated five flows going to one pod: four flows coming from the pod containing the disconnected link and one flow from another pod. Figure 7a shows the result. For oversubscription ratio of 1:1, a link failure or a node failure does not significantly affect TCP flow completion time of the original scatter scheme. This was also the case when we simulated oversubscription ratio of 4:1 by reducing the bandwidth of the links connecting core switches and aggregation switches to 250 Mbps. However, when the oversubscription ratio was set to 8:1, the flow completion time of the original packet scatter scheme drastically increases, even when only one link is disconnected. In this case, there was no packet drop. However, it was observed that queue occupancy irregularly increased in many links connecting core switches and aggregation switches. It implies that the destination host suffered a lot of packet reordering. In contrast, the flow completion time of LBSP is not affected.

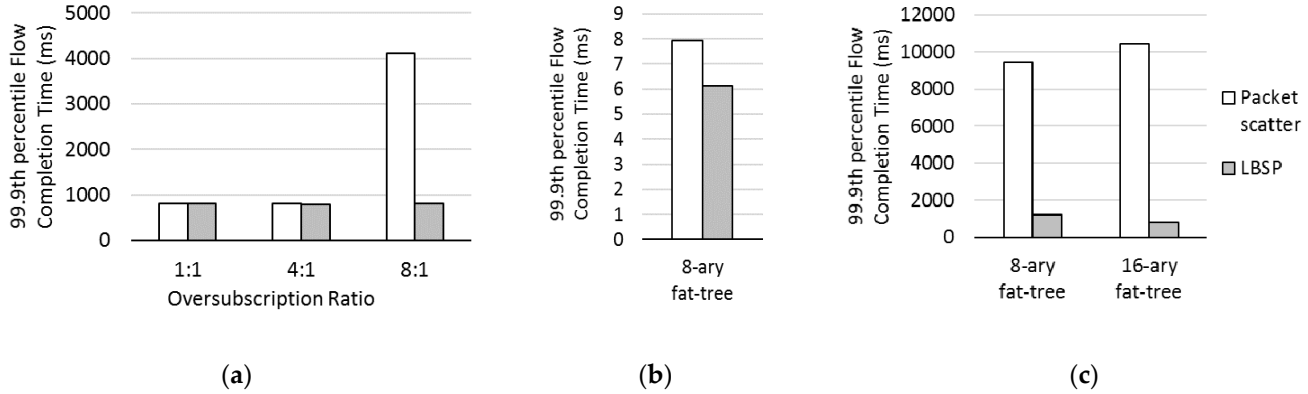


Figure 7. (**a**) Flow completion time in long-lived flow tests for 16-ary fat-tree with one link failure; (**b**) result of a short-lived flow test with degraded links; and (**c**) result of long-lived flow tests with degraded links.

Fat-trees with degraded links

We simulated asymmetries by degrading the bandwidth of a link from 1 Gbps to 100 Mbps for an 8-ary fat-tree and a 16-ary fat-tree both. We set the bandwidth of one link connecting a ToR switch and an aggregation switch to 100 Mbps. Short-lived flow tests were performed using flows with size 32 KB for the 8-ary fat-tree and long-lived flow tests using flows with size 100 MB for the 8-ary and 16-ary fat-trees both.

Figure 7b shows the results for short-lived flow tests. We can see that the effect of a degraded link on flow completion time is larger than that of a disconnected link which is shown in Figure 6a. Note that performance of LBSP is less affected by a degraded link compared to that of the original packet scatter scheme. Figure 7c shows the results for long-lived flow tests. The impact of degraded links on performance of the original packet scatter scheme is enormous. For both 8-ary and 16-ary fat-trees, the original packet scatter scheme experiences over eleven times longer flow completion time than in a normal condition. Since there is no packet drop, the performance degradation is only due to out-of-order packets. LBSP can avoid such huge degradation of performance by selecting an alternative path group that does not contain the degraded link.

5. Handling Queue Length Differentials

5.1. Introducing States in Link Group Selection

As mentioned above, the basic forwarding decision of LBSP is made based on the destination address. If the traffic load is high and the LSB and second LSB of the destination addresses are not evenly distributed, it can cause a large queue differential between link groups which leads to low network utilization and delayed response time. If a slight degree of transient packet reordering is allowed, we can reduce the queue length differentials between link groups by permitting more flexibility in selecting a path group.

First, we consider a ToR switch. For each ToR switch, we maintain three states, States 0, 1, and 2, as shown in Figure 8. A switch starts from State 0. At State 0, the switch selects a link group purely based on the LSB and second LSB of the destination address. A ToR switch monitors the queue lengths of the link groups. If the queue length of Link Group 0 exceeds a given threshold and is t times longer than that of Link Group 1, the ToR switch transitions to State 1. Conversely, if the queue length of Link Group 1 exceeds the threshold and is t times longer than that of Link Group 0, the ToR switch transitions to State 2. Since being in State 1 indicates that the output ports belonging to Link Group 0 are more crowded, we migrate a part of the traffic from Link Group 0 to Link Group 1 by relaxing the condition for selecting Link Group 1. More specifically, at State 1, packets with the destination addresses LSB 0 can also be forwarded to ports in Link Group 1 if the parity of the input port (through which the packet came in) is 1. Likewise, at State 2, a part of the traffic for Link Group 1 is forwarded to the ports in Link Group 0. Table 3 summarizes how

to determine the link group for a packet in each state. When the queue lengths between two link groups are almost equal while being in State 1 or State 2, the switch returns to State 0.

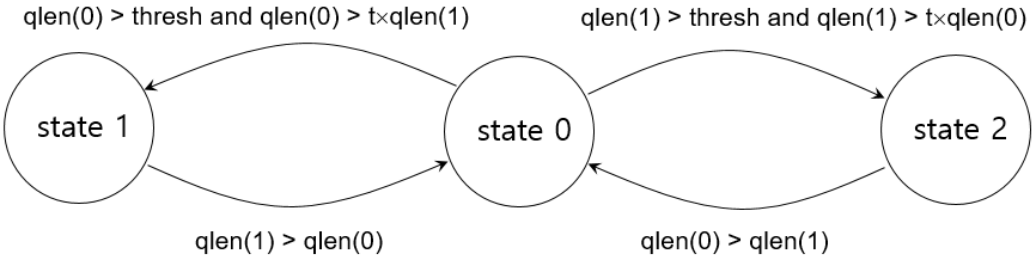


Figure 8. A switch is in one of the three states. qlen(i) indicates the queue length of an output port in link group i.

Table 3. Determining a link group for a packet in each state.

if (this switch is a ToR switch) **then**		
D ← LSB of the destination address		
else if (this switch is an aggregation switch) **then**		
D ← second LSB of the destination address		
end		
S ← parity of the input port through the packet came in		
State 1	State 0	State 2
if (D == 0) **then**	**if** (D == 0) then	**if** (D == 1) **then**
if (S == 0) **then**	group = 0	**if** (S == 0) **then**
group = 0	**else**	group = 0
else	group = 1	**else**
group = 1	**end**	group = 1
end		**end**
else		**else**
group = 1		group = 0
end		**end**

Aggregation switches essentially use the same mechanisms for state transition and link group selection as ToR switches, although we need to consider one difference. At the first hop, all the packets belonging to one flow are forwarded through the ports within the same ToR switch anyway. In contrast, at the second hop, the packets belonging to one flow are forwarded through a pair of aggregation switches. To make sure that the packets of the same flow encounter the same group of flows and experience almost the same queueing delay, the two aggregation switches must select the same link groups for each flow. To this end, a pair of aggregation switches (e.g., A0 and A2 in Figure 5) transitions to the same state simultaneously by exchanging the state information. In addition, aggregation switches use a looser condition for a state transition compared to a ToR switch, because the state transition needs a message exchange between the aggregation switches. In the numerical analysis in Section 5.2, we set the threshold and the t value to 5 and 1.5, respectively, for ToR switches, whereas, for aggregation switches, the threshold and the t value are set to 20 and 2, respectively.

When a switch transitions from State 0 to State 1 or State 2 and a flow starts to select an alternative link group with a shorter queue length, transient packet reordering can occur.

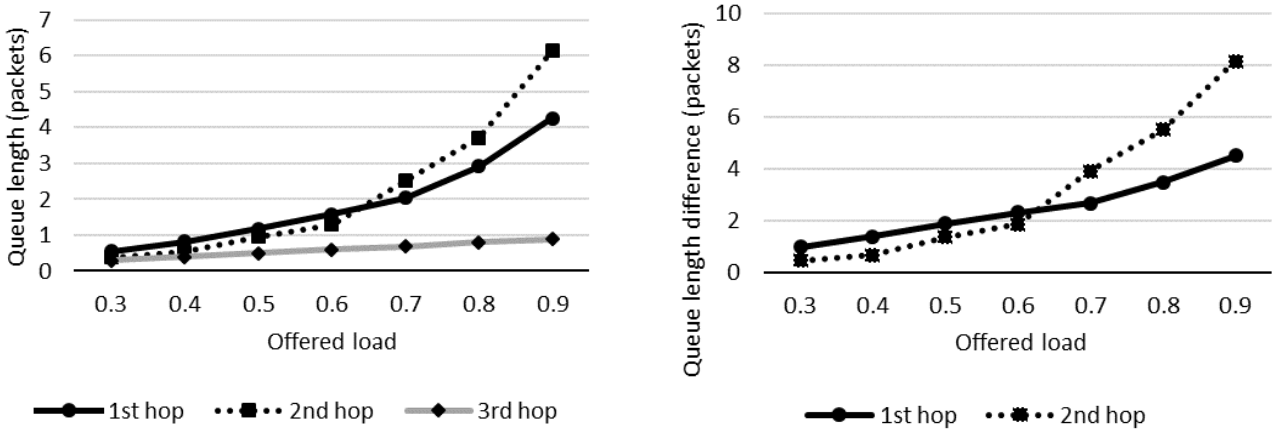
5.2. Numerical Results

We calculate the queue lengths under the mechanism presented in Table 3, by presenting the numerical results about the queue length for each hop in an 8-ary fat-tree. Adopting the time-slot model that is used in [16], we made similar assumptions. All packets have the same length. All switches are output-queued, and the capacity of each port is 1. Each

output port transmits one packet at each time-slot if the queue is not empty. Permutation traffic patterns are used and therefore once a host is selected for packet generation, the destination address of the generated packet is determined accordingly. In each time slot, each host generates packets using Bernoulli distribution.

To evaluate the behavior of a round robin manner, we randomly select a host for each pod in each time-slot and then generate four consecutive packets with the offered load ρ on each input port of each ToR switch. Packets are sent for 1000 time slots. We repeated this process for 100 randomly generated traffic permutation patterns. We varied the offered load ρ from 0.3 to 0.9.

We calculated not only the average queue lengths but also the difference between the queue lengths of different link groups. Recall that the queue lengths at the ports in the same link group are always almost equal. Figure 9a shows the average queue lengths for the upward hop outputs. We can see that traffic distribution mechanism prevents the queue lengths from growing excessively. As the offered load ρ increases, the queue lengths at the second hop (aggregation switches) are longer than those at the first hop (ToR switches). The reason is that aggregation switches less frequently perform state transitions in order to reduce message exchanges between aggregation switches. At the third hop (i.e., output ports of core switches), the average queue length increases linearly with respect to the offered load ρ, and remains below 1. Figure 9b shows the difference between the queue lengths of different link groups. Since aggregation switches less frequently perform state transitions, the queue length differences at the second hop are larger than those at the first hop.



(**a**) (**b**)

Figure 9. (**a**) Average queue lengths; and (**b**) difference between the queue lengths of link groups.

6. Related Work

There have been many efforts to enhance load-balancing in datacenter networks. We summarize the efforts in two categories: flow-level load-balancing and packet-level load-balancing.

Flow-level load balancing

ECMP (Equal Cost Multiple Path) is a standard forwarding scheme which is supported by commodity switches. ECMP distributes traffic across equal-cost paths based on flow hashing. VL2 [2] uses a variant of VLB (Valiant Load Balancing) that employs a random flow-level distribution to avoid packet reordering. In this variant of VLB, each host selects an intermediate switch (i.e., core switch) independently and randomly. This scheme may consume additional network capacity by extending a path length, but it can improve utilization of bisection bandwidth. Hedera [4] is a centralized flow scheduling scheme aimed at maximizing network utilization. Collecting information on flows, Hedera

dynamically computes paths so that flows avoid busy links. Hedera requires a picture of the whole routing and traffic demands of all flows. CAAR (Congestion-Aware Adaptive foRwarding) [5] is a distributed forwarding scheme where each flow selects the most underutilized path. The flow can be redirected to an underutilized path in transmission if the selected path becomes congested. Each switch selects the next hop based on its neighbors' queue length which is updated in each time slot. Similarly, FlowBender [6] reroutes flows when congestion is detected. In FlowBender, however, congestion is detected using end-to-end notification ECN and rerouting is initiated by end-hosts detecting congestion. In [17], the authors presented a theoretical solution to DLBP (Dynamic Load Balancing Problem) in hybrid switching data center networks with converters. MPTCP [7] generates several subflows for a flow and assigns each subflow one of multiple paths. Since each subflow performs congestion control independently, it mitigates the problem due to packet reordering. MPTCP requires significant changes to protocol stacks for end-hosts. To improve the efficacy of MPTCP for many-to-one traffic, DCMPTCP [18] has been proposed.

Packet-level load balancing

DeTail [8] is a cross-layer network stack to reduce the long-lived flow completion time tail. At the link layer, DeTail employs flow control based on port buffer occupancies to construct a lossless fabric. At the network layer, DeTail performs per-packet adaptive load balancing. Each switch dynamically picks a packet's next hop using the congestion information obtained from port buffer occupancies. Out-of-order delivery is reordered at the end-hosts. The study in [9] shows the feasibility RPS (Random Packet Spraying) in which packets of every flow are randomly assigned to one of the available shortest paths to the destination. It shows experimentally that RPS works well in symmetric topologies. To handle asymmetry due to failures, SRED (Selective RED) is proposed for use along with RPS. SRED selectively enables RED only for flows that induce queue length differentials. If a flow that cannot use all the multiple paths makes differentials of queue lengths between switches, SRED drops the packets belonging to the flow more aggressively. To this end, SRED requires a topology aware centralized fault manager, which configures end hosts or ToR routers to mark all packets of flows affected by a link failure so that downstream routers can employ RED only on marked packets. Fastpass [10] is a system where a logically centralized arbiter controls each packet's timing and path. The ultimate goal of the arbiter is to make queue lengths almost zero. At large scales, Fastpass needs several arbiters. Presto [11] performs proactive load-balancing with granularity in-between flow-level and packet-level. Edge vSwitches break each flow into discrete units of packets, called flowcells whose size is the maximum TCP Segment Offload size (64 KB). Presto employs a centralized approach. A centralized controller collects the network topology, partitions the network into a set of multiple spanning trees, and assigns each vSwitch a unique forwarding label in each spanning tree. When a failure occurs, the controller prunes the spanning trees that are affected by the failure or enforces a weighted scheduling algorithm over the spanning trees. Packet reordering is handled by the GRO (Generic Receive Offload) handler modified to reduce computational overhead. DRILL (Distributed Randomized In-network Localized Load-balancing) [12] is a datacenter fabric for Clos networks which employs per-packet decisions at each switch based on local queue occupancies and randomized algorithms to distribute load. DRILL compares queue lengths of two random ports and the previously least-loaded port, and sends the packet to the least loaded of these. To handle asymmetry, DRILL partitions network paths into groups, each of which is symmetric and applies micro load balancing inside each path group. TTC (Thresholded Two-Choice) [13] has been proposed to improve downward load-balancing in fat-trees based on the observation that DRILL [12], which well balances upward traffic, may perform poorly on downward load balancing since local path decisions are agnostic to downward congestion. Similar to DRILL, TTC also uses a two-choice algorithm. However, to improve downward load-balancing, TTC makes local routing decisions from two choices: a default path using the deterministic D-mod-k scheme [15] and a random choice. The random choice is selected only when the load of the default path exceeds that of the random choice by a threshold. However, TTC

has yet to handle link failures and packet reordering. More recently, VMS (Virtual Multi-channel Scatter) [19] has been proposed. VMS is a packet-level load-balancing scheme which runs in the virtual switch layer.

7. Conclusions

We propose a new per-packet load balancing scheme LBSP that exploits symmetric redundant paths in fat-trees. LBSP is focused on enabling packets in a flow to experience almost the same queueing delays even when the whole network becomes asymmetric due to network failures. To this end, LBSP partitions equal-cost paths between two hosts into more than one path groups of the same size. Each flow is assigned a path group based on the least significant bits of the destination address. Packets of a flow are forwarded across the links of the selected path group. When a failure is detected, LBSP selects an alternative path group for the flows affected by the failure so that the multiple paths that the packets of the flows pass through are symmetric. Only when a failure occurs, LBSP maintains the state information to reroute the affected flows. Simulation results show that LBSP is much less affected by network failures than the original packet scatter scheme.

Since LBSP selects a path group based on the destination address, the difference between the queue lengths of different link groups can be large for heavy load. To mitigate the problem, we propose a solution which migrates a part of the traffic from the default link group to the alternative link group.

Funding: Not applicable.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Al-Fares, M.; Loukissas, A.; Vahdat, A. A scalable, commodity data center network architecture. *ACM SIGCOMM Comput. Commun. Rev.* **2008**, *38*, 63–74. [CrossRef]
2. Greenberg, A.; Hamilton, J.R.; Jain, N.; Kandula, S.; Kim, C.; Lahiri, P.; Maltz, D.A.; Patel, P.; Sengupta, S. VL2: A scalable and flexible data center network. *ACM SIGCOMM Comput. Commun. Rev.* **2009**, 51–62. [CrossRef]
3. Zhang, J.; Yu, F.R.; Wang, S.; Huang, T.; Liu, Z.; Liu, Y. Load balancing in data center networks: A survey. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 2324–2352. [CrossRef]
4. Al-Fares, M.; Radhakrishnan, S.; Raghavan, B.; Huang, N.; Vahdat, A. Hedera: Dynamic flow scheduling for data center networks. *Nsdi* **2010**, *10*, 89–92.
5. Zhang, J.; Ren, F.; Huang, T.; Tang, L.; Liu, Y. Congestion-aware adaptive forwarding in datacenter networks. *Comput. Commun.* **2015**, *62*, 34–46. [CrossRef]
6. Kabbani, A.; Vamanan, B.; Hasan, J.; Duchene, F. Flowbender: Flow-level adaptive routing for improved latency and throughput in datacenter networks. In Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies, Sydney, Australia, 2–5 December 2014; pp. 149–160.
7. Raiciu, C.; Barre, S.; Pluntke, C.; Greenhalgh, A.; Wischik, D.; Handley, M. Improving datacenter performance and robustness with multipath TCP. *ACM SIGCOMM Comput. Commun. Rev.* **2011**, *41*, 266–277. [CrossRef]
8. Zats, D.; Das, T.; Mohan, P.; Borthakur, D.; Katz, R. DeTail: Reducing the flow completion time tail in datacenter networks. In Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, Helsinki, Finland, 13–17 August 2012; pp. 139–150.
9. Dixit, A.; Prakash, P.; Hu, Y.C.; Kompella, R.R. On the impact of packet spraying in data center networks. In Proceedings of the 2013 IEEE INFOCOM, Turin, Italy, 14–19 April 2013; pp. 2130–2138.
10. Perry, J.; Ousterhout, A.; Balakrishnan, H.; Shah, D.; Fugal, H. Fastpass: A centralized zero-queue datacenter network. *ACM SIGCOMM Comput. Commun. Rev.* **2014**, 307–318. [CrossRef]
11. He, K.; Rozner, E.; Agarwal, K.; Felter, W.; Carter, J.; Akella, A. Presto: Edge-based load balancing for fast datacenter networks. *ACM SIGCOMM Comput. Commun. Rev.* **2015**, *45*, 465–478. [CrossRef]
12. Ghorbani, S.; Yang, Z.; Godfrey, P.; Ganjali, Y.; Firoozshahian, A. DRILL: Micro load balancing for low-latency data center networks. In Proceedings of the Conference of the ACM Special Interest Group on Data Communication, Los Angeles, CA, USA, 21–25 August 2017; pp. 225–238.

13. Wang, S.; Luo, J.; Wong, W.S. Improved Power of Two Choices for Fat-Tree Routing. *IEEE Trans. Netw. Serv. Manag.* **2018**, *15*, 1706–1719. [CrossRef]
14. Wu, D.; Xia, Y.; Sun, X.S.; Huang, X.S.; Dzinamarira, S.; Ng, T. Masking failures from application performance in data center networks with shareable backup. In Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication, Budapest, Hungary, 20–25 August 2018; pp. 176–190.
15. Gomez, C.; Gilabert, F.; Gomez, M.E.; López, P.; Duato, J. Deterministic versus adaptive routing in fat-trees. In Proceedings of the 2007 IEEE International Parallel and Distributed Processing Symposium, Rome, Italy, 26–30 March 2007; pp. 1–8.
16. Cao, J.; Xia, R.; Yang, P.; Guo, C.; Lu, G.; Yuan, L.; Zheng, Y.; Wu, H.; Xiong, Y.; Maltz, D. Per-packet load-balanced, low-latency routing for clos-based data center networks. In Proceedings of the ninth ACM Conference on Emerging Networking Experiments and Technologies, Santa, Barbara, CA, USA, 9–12 December 2013; pp. 49–60.
17. Zheng, J.; Zheng, Q.; Gao, X.; Chen, G. Dynamic load balancing in hybrid switching data center networks with converters. In Proceedings of the 48th International Conference on Parallel Processing, Kyoto, Japan, 5–8 August 2019; pp. 1–10.
18. Dong, E.; Fu, X.; Xu, M.; Yang, Y. Low-Cost Datacenter Load Balancing with Multipath Transport and Top-of-Rack Switches. *IEEE Trans. Parallel Distrib. Syst.* **2020**, *31*, 2232–2247. [CrossRef]
19. Zhang, Y.; Bi, J.; Li, Z.; Zhou, Y.; Wang, Y. VMS: Load Balancing Based on the Virtual Switch Layer in Datacenter Networks. *IEEE J. Select. Areas Commun.* **2020**, *38*, 1176–1190. [CrossRef]