Deadlock-Free Local Fast Failover for Arbitrary Data Center Networks

Brent Stephens
UW-Madison

Alan L. Cox
Rice University

***Abstract*—Today, given data center networks' sizes and bursty workloads, it is likely that at any moment there is packet loss due to some type of failure in the network. This paper focuses on solving the two most common types of data center network failures: congestion and routing failures. Recently, there has been demand for lossless Ethernet (DCB) in data center networks as a solution to congestion failures. However, DCB complicates fault tolerance by introducing a new type of failure, deadlock. If DCB is enabled, then all routing must be deadlock free. To the best of our knowledge, this paper describes the first ever deadlock-free approaches to local fast failover that can be combined with DCB, DF-FI and DF-EDST resilience. Moreover, in the evaluation, this paper shows that DF-EDST resilience, which is the paper's main contribution, can improve fault tolerance without adversely impacting performance when compared to a state-of-the-art approach to deadlock-free routing. If, however, a small reduction in aggregate throughput is acceptable, then it is possible to build routes such that only 0.00001% of the total flows in the network are likely to fail given 16 edge failures on networks with 1K-4K hosts.**

I. Introduction

Given both the size of today's data center networks and the bursty traffic patterns of many data center applications, at any point in time there is likely to be packet loss due to some kind of network failure. This paper focuses on simultaneously addressing the two most common kinds of failures in data center networks: congestion failures [1]–[3] and routing failures [4]. Congestion failures occur when the incoming load for a link is greater than the outgoing capacity. Routing failures occur when the physical network has a connection between two endpoints but there does not exist an installed route between them that does not use a failed link or switch, or, even worse, when packets are forwarded in a loop.

Recent research has proposed enabling lossless Ethernet (DCB) within data center networks [5]–[9]. Not only does DCB prevent congestion failures due to traffic patterns such as incast [6], [7], by enabling the elimination of TCP slow start, it also promotes shorter completion times for the medium-sized flows [6] that are common under many data center applications [1], [3]. However, DCB does not address routing failures. Concurrently, there is also much research on developing implementations of local fast failover to prevent routing failures [10]–[20]. In part, this research is motivated by a recent study finding that other existing failover approaches still lead to significant packet loss [4]. Ideally, one of the implementations of local fast failover could be combined with DCB to protect against both routing and congestion failures. Unfortunately, enabling DCB makes a new kind of failure, deadlock [21], possible, and no existing implementation of local fast failover guarantees deadlock-free routing.

Because a single group of deadlocked switch ports can render a data center network unusable, avoiding deadlocks is essential. Current approaches to deadlock-free routing when DCB is enabled either rely on minimal routing on fat tree topologies [5] or restricting routes to the paths defined by a set of edge disjoint spanning trees (EDSTs) [6]. Although some implementations of local fast failover have utilized either fat trees [16] or EDSTs [17], [20], this does not inherently guarantee deadlock-free routing. For example, an implementation might use non-minimal routes on a fat tree [16]. And, in fact, prior implementations using EDSTs to handle routing failures [17], [20] are not deadlock free.

To enable the use of DCB with local fast failover, this paper introduces the first ever approaches to local fast failover that guarantee deadlock-free routing for arbitrary network topologies. Our first approach takes the routes built by MPLS-FRR [13], Plinko [10], or FCP [12], all of which implement what we call *failure identifying* (FI) resilience, and assigns these routes to the network's virtual channels so as to prevent deadlock. In order to implement this approach, which we call DF-FI resilience, we modified an FI resilient routing algorithm from Stephens *et al.* [6] so that it never routes packets over a port more than once and adapted the virtual channel assignment algorithm from Domke *et al.* [22] so that it applies to backup routes. Although DF-FI resilience is desirable because it does not impact performance, we ultimately conclude that it has limited scalability because of its virtual channel requirements. For example, tolerating two arbitrary failures on some 1024-host topologies requires more than 8 virtual channels, which is the total number of virtual channels in DCB [23]. Moreover, this approach conflicts with the use of virtual channels in DCB for prioritizing traffic.

In contrast, our second approach, which is the main contribution of this paper, overcomes the scalability limitations of DF-FI resilience because it does not require the network to provide virtual channels. Under this approach, which we call DF-EDST resilience, packets transition between the paths defined by different EDSTs on network failures. Although transitioning packets arbitrarily between trees can lead to deadlocks, we prove that if the graph of allowed tree transitions is acyclic, then the resulting forwarding function is deadlock free. In effect, DF-EDST resilience removes all of the tree transitions that could cause cyclic dependencies from EDST resilience. Nonetheless, if the network provides even a limited number of virtual channels, DF-EDST resilience can exploit them to improve performance.

Given that we prove that DF-EDST resilience is deadlock free, the main questions that we will seek to answer are: What is the performance of DF-EDST routing and how fault tolerant are DF-EDST routes? However, the answers to these questions are dependent on a trade-off made in the construction of the tree-transition graph (TTG). Because the TTG must be acyclic, a tree can be used as an initial tree to increase performance through path diversity or as a backup tree to support fault tolerance for all routes, but not as both. However, even if all trees are used as initial trees, DF-EDST can still improve fault tolerance. This is because the initial trees may be connected in the TTG, and only traffic that is forwarded along the one initial tree in the TTG that does not have a successor cannot tolerate at least a single link failure. Moreover, if trees are set aside for fault tolerance at the cost of a small reduction in performance, each backup tree is guaranteed to protect *all* routes against *any* additional link failure. Further, not all failures affect all routes. Each backup tree also often reduces the expected probability of routing failure by an order of magnitude.

All of these results apply to arbitrary topologies. If a tree topology is used, then it is possible to do even better with respect to throughput. With a tree topology, minimal routing subject to any traffic engineering scheme may be used on one virtual channel and still be guaranteed deadlock free. Then, when a packet encounters a failure, it may transition to another virtual channel and follow a set of backup routes defined by DF-EDST resilience without any potential for deadlock. In effect, the only limits on fault tolerance are the connectivity of the network and the amount of forwarding table state.

In summary, the contributions of this paper are as follows:

Proving that DF-EDST resilience is deadlock free: In this paper, we prove that DF-EDST resilience is deadlock free as long as the graph of allowed tree transitions is acyclic.

Analyzing the fault tolerance of DF-EDST resilience: In this paper, we analyze the worst and best case fault tolerance of DF-EDST resilience. In the worst case, we show that if the height of the TTG is h, then DF-EDST is $(h-1)$-resilient. In the best case, we show that DF-EDST resilience can tolerate as many failures as the most number of trees reachable from an initial tree.

Evaluating DF-EDST resilience: We evaluate the trade-off between performance and fault tolerance for variants of DF-EDST resilience with different TTGs. We find that even on arbitrary topologies it is possible both to provide high throughput lossless forwarding and to provide forwarding where only 0.00001% of the routes in the network are likely to fail given 16 arbitrary edge failures and topologies with 1K–4K hosts.

The rest of this paper is organized as follows. First, Section II introduces background information on deadlock-free routing and fault tolerant routing. Next, Section III introduces DF-FI resilience, and Section IV introduces DF-EDST resilience. Section V and Section VI present our methodology and evaluation respectively. We discuss related work in Section VII. Finally, we conclude in Section VIII.

II. Background

This section provides background first on deadlock-free routing and then on fault-tolerant routing, focusing on data center networking. Although this paper is motivated by earlier work that uses lossless Ethernet to improve TCP and application performance [5], [6], [24], [25], we do not discuss these projects in detail. The specifics of the congestion control algorithm used in the network is independent of routing, so any of these approaches are implementable as long as routing is guaranteed to be deadlock free. Finally, there is a large amount of research on fault-tolerant routing [10]–[20], so we limit our discussion to the most closely related projects.

A. Deadlock-Free Routing

Instead of dropping packets when congestion occurs, lossless Ethernet, or DCB, provides *pause frames*, which allow congested ports to apply back pressure on the upstream ports causing the congestion. Before a buffer at an input port is overrun, the NIC/switch port will send a pause frame to the output port causing the congestion, which forces it to stop sending packets for long enough for traffic to drain from the network. As congestion occurs, these pause frames can cascade through the network. Consequently, it is possible for the network to reach a forwarding deadlock, which can render the network incapable of forwarding traffic [21]. Because of this, the forwarding function formed by all of the routes in the network must be deadlock free if DCB is enabled. Deadlock freedom can be thought of as an extension of loop freedom. While loop freedom on a hop-by-hop routed network requires that there is no cyclic dependency between the ports used to route a single flow, deadlock freedom requires that there is no cyclic dependency between the ports used to route all flows.

In addition to introducing the first deadlock-free routing algorithms, Dally and Sietz [21] also proved a sufficient condition for a forwarding function to be deadlock free. In general, they showed that deadlocks can arise whenever there exists a cycle in the network's channel dependency graph. To formally describe this concept, let $G = (V, E)$ be a bi-directional graph, where $V = \{1, \ldots, n\}$ is the set of vertices, $\{u, v\} \in E$ is the set of bi-directional edges, and $C = \{(u, v), (v, u) \forall \{u, v\} \in E\}$ is the set of unidirectional channels (ports/buffers), two for each edge. Let $(c_1, \ldots, c_s) \in R$ be the set of all routes installed in the network, with each route being represented as the sequence of channels that it uses. Let $D = (C, K)$ be the network's channel dependency graph, where C again is the set of network channels and $K = \{(c_i, c_{i+1}) \forall i \in \{1, \ldots, s-1\} \forall (c_1, \ldots, c_s) \in R\}$ is a set of directed edges representing the channel dependencies created by R. Under this model, Dally and Seitz proved that a network is deadlock free if D is acyclic.

While there is a significant amount of research on deadlock-free routing algorithms, those that are applicable to Ethernet use two techniques, either independently or in combination [26]. The first technique is to restrict routing to avoid channel dependency cycles. The second technique is to divide each channel into multiple deadlock-independent virtual channels that share the same physical channel and then assign routes to virtual channels so that the forwarding function is deadlock free. However, this second technique is only feasible if the required number of virtual channels is less than or equal to the number of virtual channels provided by the underlying hardware. In DCB, there are 8 virtual channels [23]. In Infiniband, there may be up to 16, although recent hardware has only supported 8 [22].

A relevant deadlock-free routing algorithm that uses the first technique is Up*/Down* [27] and its many variants [26]. Essentially, Up*/Down* builds a spanning tree of the network and assigns one direction of each link as *up* and the other as *down* based on this spanning tree. Because routes are only allowed to traverse up channels followed by down channels, in other words, a route cannot transition from a down channel to an up channel, there can never be a channel dependency cycle. This ensures that routing is deadlock free. Further, proving that minimal routing on fat trees is deadlock free is similar to proving that Up*/Down* routing is deadlock free [28].

To improve the performance of deadlock-free routing on well-connected data center topologies, recent research [6] has proposed using EDSTs, of which there are $k/2$ in a k-connected topology. While Up*/Down* builds a single tree and assigns directions to all links based on this tree, this approach builds multiple disjoint trees and assigns directions to links based on the tree that the links are a member of. Further, the performance of using EDSTs for deadlock-free routing can be improved by installing routes derived from different sets of EDSTs on different virtual channels. This increases the total number of available paths while still remaining deadlock free.

The most relevant deadlock-free routing algorithms that use virtual channels are LASH [29] and DFSSSP [22]. Both build paths between all sources and destinations oblivious to deadlocks and then use a heuristic to break any cyclic dependencies by assigning paths to virtual channels. While this approach allows for high performance routing, the major drawback is that the number of required virtual channels increases with topology size. Because the expected time complexity of DFSSSP is smaller than that of LASH, we focus on DFSSSP. Another relevant approach assigns IDs to switches, requiring that packets change virtual channels when they are forwarded to a switch with a lower ID than the current switch [30].

B. *Fault Tolerant Routing*

A routing failure occurs when there exists a path in the underlying network topology for a flow, but there does not exist an installed route that does not use a failed link or switch. Routing failures are often addressed by computing and installing a new end-to-end route, and this can be done with either distributed or centralized protocols. However, it is becoming increasingly important to handle routing failures at the switch local to the failure so as to avoid dropping packets while computing a new end-to-end route.

We consider two different approaches to providing local fast failover that can protect against multiple link failures on arbitrary topologies. The first approach, which we call *failure identifying* (FI) resilience, includes Plinko [10], MPLS-FRR [13], and a variant of FCP [12]. These approaches allow for arbitrary routing, even for backup routes, as well as arbitrary levels of resilience. This is because these approaches all mark packets so that the set of failures a packet has already encountered may be identified from its headers. This then enables a routing algorithm that can recursively build backup routes that protect against the failure of any link in any of the currently considered routes, if the routes exist. This process starts with the default routes and continues to build backup routes for the previous round of backup routes until the desired level of resilience is achieved. Because failures are explicitly marked in packet headers and are never removed, it is possible to guarantee that a packet in FI resilience will eventually reach the destination or be dropped because no path exists.

Yener *et al.* [20] introduced the concept of routing along edge-disjoint spanning trees (EDSTs) to provide fault tolerance. Because EDSTs are spanning trees, an alternate tree can be selected regardless of which switch needs to route around a failure. Because EDSTs do not have any edges in common, an alternate tree is guaranteed to avoid the edge that caused the previous tree to fail. In a k-connected topology, there exist $k/2$ EDSTs, and these $k/2$ EDSTs can be used to protect against the failure of $k/2-1$ arbitrary edges [17]. To ensure that packets are dropped in the event of a partition, a $k/2$ bit wide bitfield is added to the packet headers, and when a failed edge is encountered, the appropriate bit in the bitfield is set to represent that the tree the edge is a member of has failed. However, if the trees are arranged in a line and traversed in order, then tracking failed trees in a packet header is not necessary, a fact noted by Elhourani *et al.* [17].

Lastly, Feigenbaum *et al.* [11] introduced some important theory on routing failures. First, they define the concept of a t-resilient forwarding pattern, which is a forwarding pattern that, even given t arbitrary failures, defines a forwarding path between any two hosts as long as there exists a path between them in the underlying topology. Further, a t-resilient forwarding pattern never defines any infinitely long forwarding paths, *i.e.*, forwarding loops. Given this definition, Feigenbaum *et al.* proved that, even if packets are not modified to identify failures, then there always exists a 1-resilient forwarding function, but there does not always exist an ∞-resilient one.

III. DF-FI Resilience

This section discusses the implementation of DF-FI resilience, a deadlock-free variant of FI resilience [10], [12], [13]. Because FI resilience does not consider cyclic channel dependencies when building routes, a set of resilient routes could easily form a cyclic channel dependency, which can lead to deadlock. However, FI resilience can made deadlock free with a few simple changes, although doing so has the drawback that it requires a variable number of virtual channels. Specifically, implementing DF-FI resilience involves modifying the FI resilient routing algorithm so that it never has a packet use the same arc twice, and modifying the virtual channel assignment algorithm of Domke *et al.* [22] so that it assigns route branchings to virtual channels instead of individual routes.

The need for the first change is clear. In FI resilience, a route could be deadlocked on itself if it ever traverses the same link in the same direction twice. However, in FI resilience, packets may be in flight following multiple different routes for a single source and destination immediately after a failure, and any route with in-flight packets can cause a deadlock. To handle this, we assign *route branchings* to different virtual channels instead of routes. A route branching is the graph of routes that a packet can follow after it has been forwarded by a top-of-rack (TOR) switch across a port. Because packets may be in flight down any of the routes in a branching and packets are not allowed to change virtual channels to avoid dependencies across multiple virtual channels, a route branching is the smallest unit that is assignable to a virtual channel.

IV. DF-EDST Resilience

On the other hand, if routing is restricted, then it should be possible to provide deadlock-free local fast failover without the use of virtual channels. However, the principle difficulty in doing so is in providing a useful level of fault tolerance as well as high throughput forwarding.

EDSTs have previously been used to provide both deadlock-free routing [6] and fault tolerant routing [17], [20]. However, allowing packets to transition arbitrarily between trees prevents EDST resilience from being deadlock free.

To solve this problem, we introduce DF-EDST resilience. In DF-EDST resilience, routing is not only restricted to use paths defined by the EDSTs, but the graph of allowed tree transitions is restricted to be acyclic. This paper proves that this is sufficient to guarantee deadlock freedom.

In the rest of this section, we first present a proof that DF-EDST resilience is deadlock-free and then analyze its resilience and state requirements. After that, we discuss the inherent trade-off between performance and resilience that is introduced by restricting tree transitions and introduce several different tree transition graphs (TTGs).

A. DF-EDST Analysis

In DF-EDST resilience, all routes follow the paths defined by a set of EDSTs until a failure is encountered. Once a failure is encountered, a packet may only transition to the paths defined by the trees that are successors of the current tree in a tree transition graph (TTG). Given this forwarding model, we would like to prove the following theorem.

Theorem 1 *DF-EDST resilience is deadlock free if the TTG is acyclic.*

In order to prove this theorem, we use two properties. The first is that, if there exists a total ordering of channel requests, then the forwarding function is deadlock free. This follows from Dally and Seitz [21]. If there exists a total ordering of channel requests, then the channel dependency graph is acyclic, and thus the forwarding function is deadlock free.

The second property that we use is that there exists a total ordering of channel dependencies within an EDST. This follows from the proofs that Up*/Down* routing and routing on trees is deadlock free [27], [28]. The channels in an EDST can be divided into two groups, up channels and down channels. There exists a topological ordering of both the up and down channels, and the set of up channels is ordered before the down channels. Thus, there exists a total ordering of channel requests $c_{u1} < \ldots < c_{u(n-1)} < c_{d1} < \ldots < c_{d(n-1)}$ in a spanning tree of a network with n vertices.

Given these two properties, proving Theorem 1 is straightforward. Given a set of EDSTs T, every channel $c \in C$ is guaranteed to be a member of at most one tree. Let C_t be the set of all channels for a given tree $t \in T$. Because packet transitions between trees are restricted by a TTG, which is a DAG, there exists a topological ordering of the trees. This implies that there is a topological sorting of the set of the channels that belong to trees on a k-connected topology, $C_1 < \ldots < C_{k/2}$. Thus, there must exist a total ordering of channel requests $c_{1u1} < \ldots < c_{1d(n-1)} < \ldots < c_{(k/2)u1} < \ldots < c_{(k/2)d(n-1)}$ on a k-connected topology with n vertices. Thus, the forwarding function must be deadlock-free.

Further, DF-EDST resilience does not need to modify packet headers, either to mark the current tree or tree failures in a packet header. Because each input edge only belongs to a single EDST, the input edge of a packet identifies the current EDST. Because the TTG is acyclic, a packet is guaranteed to either reach its destination or be dropped as it transitions between trees as it will eventually reach a leaf tree that does not have a valid transition to any other EDST.

Theorem 2 *If a network uses the forwarding function $f_v(d, e_v, bm) \rightarrow e$, where $d \in D$ is a destination, $e_v \in E_v$ is the input port of a packet, and bm is a bitmask of the local port status, then DF-EDST resilience can always build a $\lfloor k/2-1 \rfloor$-resilient forwarding pattern on a k-connected topology.*

To understand this theorem, consider a TTG that is a line. Because a k-connected topology contains $\lfloor k/2 \rfloor$ EDSTs, there are $\lfloor k/2 \rfloor$ nodes in the TTG. Because EDSTs are spanning trees, regardless of which link in the tree fails, it will always be possible to transition to another tree. Because a packet starts forwarding on t_1 and only transitions from EDST t_i to t_j when it encounters a single edge failure, a packet must encounter $\lfloor k/2-1 \rfloor$ failures before it is forwarded along TTG $t_{\lfloor k/2 \rfloor}$. If a packet encounters a failure when forwarding along TTG $t_{\lfloor k/2 \rfloor}$, then it will be dropped because there are no more subsequent trees. Thus, DF-EDST resilience with a line TTG provides $\lfloor k/2-1 \rfloor$-resilience. This result implies that there is always a max(1, $\lfloor k/2-1 \rfloor$)-resilient forwarding function given a k-connected topology even if packets are not modified.

At initial consideration, this result may not seem practical as it provides poor forwarding throughput when implemented for deadlock-free routing as all initial forwarding paths use the same tree, like Ethernet with RSTP. However, this result is useful even in deadlock-free routing if this level of fault tolerance were to be only applied to mission critical data. Moreover, this result also has further implications given non-deadlock-free routing, where each destination may use a different TTG. This implies that existing networks that allow for fault-tolerant forwarding but do not allow for introducing new packet formats, such as IP-FRR [31] and OpenFlow [32], can build a forwarding function that is $\lfloor k/2 - 1 \rfloor$-resilient.

However, if a line TTG is not used, then DF-EDST may not be as resilient. Specifically, if $h(t)$ is the height of an EDST in the TTG, and IT is the set of initial trees that a packet may start forwarding over, then DF-EDST is $(min_{t \in IT} h(t) - 1)$-resilient. This implies that, if packets may start forwarding over any tree in the TTG ($IT = V_{TTG}$) then DF-EDST resilience is 0-resilient.

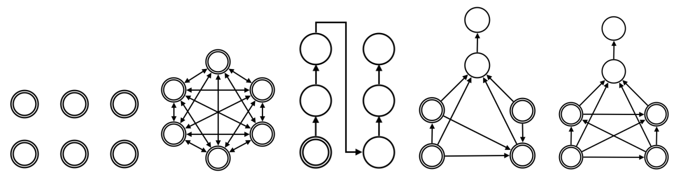
On the other hand, t-resilience is a conservative metric for characterizing fault tolerance. Given t-resilience, many routes will survive more than t failures. For example, given a set of failures, many packets will not even encounter a single failure, let alone all of the failures. Further, in DF-EDST resilience, a packet may be able to encounter $max_{t \in TTG} reachable(t)$ failures and still have a valid route, where $reachable(t)$ is a function that returns the total number of EDSTs reachable from a tree. Because of this, we evaluate both the resilience and average expected probability of routing failure of DF-EDST resilience.

B. DF-EDST Implementation

In this section, we describe how to actually implement DF-EDST resilience. First, we describe a number of different TTGs. Then, we compare and contrast them. After that, we discuss the routing algorithm needed for DF-EDST resilience. Lastly, this section finishes with a discussion.

1) TTGs: The proof in Section IV-A that DF-EDST resilience is deadlock free hints at the reason why implementing routes that provide both high throughput forwarding and a useful level of resilience can be difficult. If IT, the set of trees that a packet may initially be forwarded over, is too large, then not enough resilience may be guaranteed. However, if IT is too small, then the default routes may not provide enough path diversity to provide high throughput forwarding. Thus, the choice of TTG for use in a network presents a trade-off between performance and resilience. To explore this trade-off, we consider variants of 5 different TTGs.

The first two TTGs we consider are *NoRes* and *NoDFR*. These are the TTGs that result from prior work on using EDSTs for deadlock-free routing and fast failover, respectively. Although these TTGs are either not fault tolerant or not deadlock free, we consider them because they provide a baseline from which to compare the performance impact of DF-EDST resilience and a lower and upper bound on fault tolerance, respectively. Figures 1a and 1b illustrate these TTGs, with the initial trees being represented with double circles.



(a) **A Non-Resilient TTG** (b) **A Non-Deadlock-Free TTG** (c) **A Line TTG** (d) **A Rand 2-resilient TTG** (e) **A Max 2-resilient TTG**

Fig. 1: **Different TTGs for DF-EDST**

Algorithm 1 – DF-EDST Routing

```
Input: network topology $G = (V, E)$ where $V = \{1, \dots, n\}$, and a set of edge-
    disjoint trees $ET$.
Output: a forwarding pattern $f = (f_1, \dots, f_n)$. $\forall v \in V$, $f_v(d, ip, e, F_v) \rightarrow e$,
    where $d \in D$ is the destination, $ip \in E_v$ is the input port, $e \in E_v$ is an edge that
    must not be failed that is also used as the output edge, and $F_v \subseteq E_V$ is a set of
    edges that must be failed.
1) Build non-backup routes at each vertex for each destination for each input port
   tree: $\forall v \in V$, $\forall d \in D$, and $\forall ip \in E_v$, do:
 • Let $t \in ET$ be the tree that $ip$ is a member of, i.e., $ip \in t$. If $t$ does not exist,
   continue.
 • Let $e \in t$ be the output edge on $t$'s route to $d$.
 • Set $f_v(d, ip, e, \varnothing) := e$
 • Build backup routes that transition to all trees reachable from $t$:
 – Let $F_v := \{e\}$ be the set of edges that must be failed.
 – For each tree reachable from $t$: $\forall at \in reachable(t)$
   ∗ Let $ae \in at$ be the output edge on $at$'s route to $d$.
   ∗ Set $f_v(d, ip, ae, F_v) := ae$
   ∗ Let $F_v := F_v \cup \{ae\}$
```

The first TTG that we consider that is both fault tolerant and deadlock free is the *Line* TTG (Figure 1c). However, this TTG is still not practical. Because all forwarding starts off on a single spanning tree, performance will be as limited as traditional Ethernet with a single spanning tree that is built by RSTP. We consider this TTG because it bounds the fault tolerance achievable given DF-EDST resilience.

Lastly, we also consider two practical TTGs, the *Rand* and *Max* TTGs, which are presented in Figures 1d and 1e. These TTGs have many initial trees and guaranteed resilience for all routes. Further, they improve average fault tolerance by allowing packets to transition between initial trees. In the Rand TTG, the initial trees start out in a fully connected TTG, then cycles are randomly broken until the TTG is a DAG similar to how cycles are broken in the deadlock-free routing algorithm of Domke *et al.* [22]. To guarantee resilience, the non-initial trees are connected to a line. The Max TTG also arranges the non-initial trees in a line, but, in the Max TTG, the initial trees form a maximally-connected DAG, *i.e.*, the upper triangle of the adjacency matrix is all ones.

2) Routing: Prior work on EDST resilience [17] does not present a routing algorithm that is suitable for implementation in hardware forwarding tables. We introduce a routing algorithm for DF-EDST resilience suitable for implementation given TCAMs in Algorithm 1, and this algorithm is a generalized version of the routing algorithm for EDST resilience.

Given Algorithm 1, Equation 1 captures the state requirements of DF-EDST resilience if the current tree is marked in a packet's header.

$$\forall v \in V, |f_v(d, t, e_v, bm)| = |D| * \sum_{t \in TTG} reachable(t) + 1 \quad (1)$$

Because the path lengths for each tree may vary widely, to choose from the available initial trees, we use a random top-k approach similar to that taken by prior work on using EDSTs for deadlock-free routing [6]. For every switch and destination, the initial trees across all of the virtual channels are sorted by path length. If the best tree is within a factor of 1.35× of the shortest path distance, then forwarding table entries are installed such that a packet starts forwarding randomly over any of the up-to top-8 initial EDSTs whose path length is within a factor of 1.35×. Otherwise, default rules are installed that randomly route over any of the trees with a path length equal to that of the best initial EDST.

3) Discussion: On networks with low connectivity, the applicability of EDST resilience may be limited due to the number of EDSTs. However, using EDSTs for resilience can suffer from the opposite problem on data center networks. There are at least 20 EDSTs on the $1:1$ bisection bandwidth ratio topologies we evaluate. Even after applying NetLord-style network virtualization [33] to reduce state, installing rules for all 20 EDSTs on a 2048-host topology can require more than 10 Mbit of TCAM state for many of the TTGs we evaluate. Because of this, we consider ways to reduce the forwarding table state.

Specifically, forwarding table state can be reduced by reducing the number of virtual channels used, which can impact performance, or forwarding over a subset of the TTG for each destination, which can impact performance or fault tolerance. Reducing the number of virtual channels and thus the independent sets of EDSTs is simple to implement. On the other hand, using only a subset of the trees in the TTG is more complicated.

If only the initial trees are selected for the subset, then the forwarding table would be 0-resilient. Further, all switches must use the same TTG subset for a given destination so that rules for the entirety of a spanning tree are installed. However, if the subset of the TTG avoids the trees with the longest paths for the destination, then state may be reduced without significant impact to performance.

Given the trade-off between fault tolerance and performance in choosing a subset of the TTG for a destination, we chose the trees with the shortest average path length for the destination, subject to the resilience constraints of the TTG.

V. Methodology

This section presents our methodology for evaluating DF-FI and DF-EDST resilience[1], which is similar to prior work so as to allow comparisons [6], [10].

To evaluate DF-FI resilience, we implemented Plinko with the modifications discussed in Section III and then evaluated how many virtual channels are required for the DF-FI routes on both EGFT [34] and Jellyfish [35] topologies built with 64-port switches to support a range of hosts (1K-4K), bisection bandwidth ratios, ranging from $1:1$ (B1) to $1:6$ (B6), levels of resilience, and degrees of multipathing.

Similarly, we use the same topologies to evaluate DF-EDST resilience. Like prior work [6], we use a randomized algorithm for finding the set of EDSTs on a topology. Given the TTGs we introduced in Section IV-B1, we use simulations to compute the aggregate throughput achieved, the expected probability of routing failure, and the size of the forwarding tables. To compute forwarding throughput, we use a uniform random (URand) workload with a degree of four combined with Algorithm 1 from DevoFlow [36]. We then normalize the achieved throughput by the combined throughput capacity of the end hosts. To compute the probability of routing failure, we randomly select sets of edges of different sizes for failures. After evaluating the impact of at least 80 different sets of failed edges, we then report the average fraction of routes in the network that experienced a routing failure. To provide an upper bound on throughput, we compare DF-EDST resilience against ideal deadlock-oblivious shortest path routing. To compute the state requirements of DF-EDST, we assume, like prior work [10], that each entry requires 64-bits of TCAM state, which is the size of the port bitmask on 64-port switches. Further, we consider building forwarding tables both with and without the current EDST marked in packet headers, because marking packets can reduce forwarding table state. Lastly, because the total number of available virtual channels on DCB is 8, we only present state results assuming NetLord-style network virtualization with 8-way and 4-way multipathing, which reduces the state requirements from being proportional to the number of end-hosts to the number of switches times the degree of multipathing.

VI. Evaluation

In this section, we first present the results from our analysis of DF-FI resilience and then present our evaluation of DF-EDST resilience. For DF-FI resilience, we find that all but relatively small topologies and low levels of resilience require more virtual channels than are available on today's networks. However, this just further motivates our analysis of DF-EDST resilience. This analysis shows that DF-EDST resilience can improve fault tolerance by roughly an order of magnitude without impacting performance when compared with using EDSTs just for deadlock-free routing. However, if we are willing to accept a small reduction in aggregate forwarding throughput, often $<$5–10%, then the expected probability of routing failure given even tens of link failures can be reduced by 3 to 4 orders of magnitude.

A. DF-FI Resilience

For DF-FI resilience, there is one key question that we would like to answer: how many virtual channels are required to implement DF-FI resilience? If DF-FI resilience needs more virtual channels for a topology size and resilience level than are provided by the network, which is 8 for DCB, then

[1]https://github.com/bestephe/res-sim

	512-H 1-E/2-E/4-E/8-E	1024-H 1-E/2-E/4-E/8-E	2048-H 1-E/2-E/4-E/8-E
0-R	1/1/1/1	1/1/1/1	1/1/1/1
1-R	1/1/?/1	1/1/1/?	3/5/?/?
2-R	1/1/1/1	3/4/6/10	7/12/21/?
4-R	2/3/4/?	7/14/25/?	18/?/?/?

TABLE I: Number of VCs Required on (B1) EGFTs

this impies that DF-FI would not be implementable for this topology size and resilience level.

To answer this question, Table I shows the required number of virtual channels on (B1) EGFT topologies for varying topology sizes (*-H), levels of resilience (*-R), and degrees of multipathing (*-E). As expected, 0-resilience on EGFTs only requires one virtual channel. However, given resilience, this table shows that the number of virtual channels required by DF-FI resilience can be prohibitive. Providing even 8-way ECMP and 4-resilience required 10 or more virtual channels on all of the 1024-host topologies we evaluated. Similarly, 4-way ECMP and 1-resilience requires 10 or more trees on all of the 2048-host topologies we evaluated, and 1-way ECMP and 2-resilience is barely implementable on the 2048-host topologies we evaluated, typically requiring 8 virtual channels. Different bisection bandwidths and the Jellyfish topologies showed similar trends.

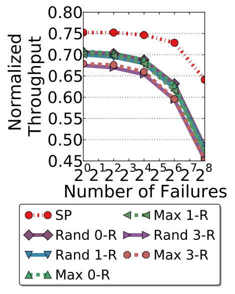
B. DF-EDST Resilience

Unlike EDST resilience, DF-EDST resilience represents a trade-off between performance and fault tolerance. Because of this, we are primarily interested in evaluating the aggregate throughput and expected probability of routing failure for different variants of the TTGs discussed in Section IV-B1. In addition, we evaluate the forwarding table state requirements of DF-EDST resilience.

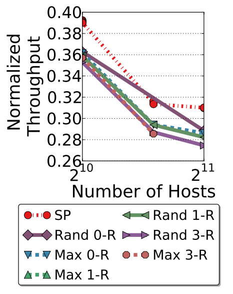
All things considered, DF-EDST resilience can match the performance of prior work on using EDSTs for deadlock freedom [6] while still reducing the probability of a routing failure by about an order of magnitude. Moreover, if a small impact on performance is acceptable, fault tolerance may be increased. Even providing 3-resilient routing often only reduces throughput by between 5–10%, and 3-resilience experiences 3–4 orders of magnitude fewer routing failures than non-fault-tolerant routing given 16 edge failures.

This paper introduces two practical TTGs, Rand and Max, which leads to the following question: Is one of these TTGs better than the other? Because the Rand TTG is less connected than the Max TTG, it could be expected to increase forwarding throughput and decrease fault tolerance. However, in our evaluation, we find that neither the Rand nor Max TTGs differ significantly in terms of throughput, expected probability of routing failure, or forwarding table state.

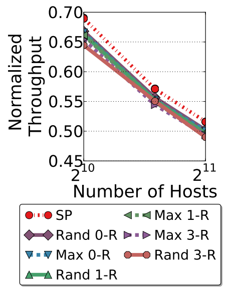
Forwarding table state may be reduced by either reducing the size of the TTG subset for each destination or by reducing the number of different TTGs on different virtual channels used to increased forwarding throughput, *i.e.*, reducing the degree of multipathing. Although it may be expected that these approaches would represent different points in the space



(a) **8-T 8-E 1024-hosts (B1 EGFT)**



(b) **8-T 8-E 16-F (B4 EGFT)**



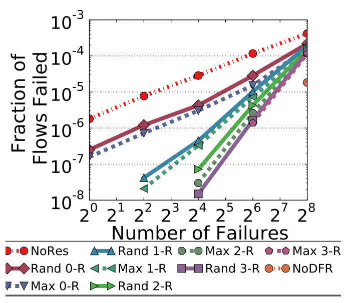
(c) **8-T 8-E 16-F (B4 Jellyfish)**

Fig. 2: **Throughput for the Rand and Max TTGs**

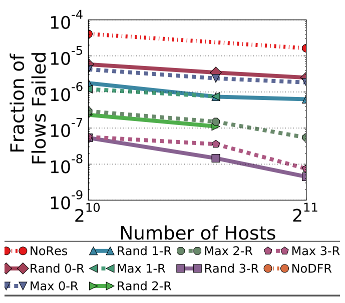
of trade-offs between throughput and resilience, we find that neither is inherently better than the other. For example, we found that using 12 trees per destination and 4-way multipathing (4 virtual channels) had the same aggregate forwarding throughput and forwarding table state and only slightly lower probability of routing failure when compared with using 8 trees per destination and 8-way multipathing (8 virtual channels). This result is particularly interesting because it implies that not all of the available virtual channels are required for high performance and fault tolerant deadlock-free routing.

In the rest of this section, we present our evaluation in more detail. To start off, Figure 2 shows the throughput achieved by variants of the Rand and Max TTGs with varying resilience (*-R), number of trees per destination (*-T), degrees of multipathing (*-E), and sorting the initial trees either for performance (P) or resilience (R) on variants of the Jellyfish and EGFT topologies with varying numbers of hosts and bisection bandwidths (B*). The first thing that these figures show is that the throughput of both the Rand and Max TTGs closely tracks that of deadlock-oblivious reactive shortest path routing (SP), especially given no or relatively few failures. To some extent, this is expected. Because every tree is an initial tree, the 0-R variants behave identically to the NoRes TTG given no failures. Although increasing resilience in the other *-R variants disallows a number of trees from being initial trees equal to the level of resilience, the impact of this on performance is also relatively small. For example, given no failures on the (B1) 1024-host EGFT topology, the Rand 0-R TTG reduced forwarding throughput by 6.5% when compared with SP, while the Rand 3-R TTG only reduced forwarding throughput by 10.9%. While this impact increases with topology size, the impact is still small, with the Rand 0-R TTG on a 2048-host (B2) EGFT reducing forwarding throughput by 6.7%, the Rand 3-R TTG reducing throughput by 14.3%, and the other TTGs falling somewhere in-between.

We also found that both the “8-T 8-E” and “12-T 4-E” TTG variants provide near-identical forwarding table throughput across a range of number of failures, topologies, and bisection bandwidths. However, the “12-T 8-E” and “16-T 8-E” TTG variants, while not shown, had forwarding throughput even closer to that of SP. This shows that, if more forwarding table state can be dedicated to DF-EDST resilience, then the impact



(a) **12-T 4-E 1024-hosts (B1)**



(b) **12-T 4-E 16-F (Jellyfish) (B4)**

Fig. 3: **Probability of Routing Failure given DF-EDST**

of forwarding throughput can be even further reduced.

To show how likely it is for DF-EDST routes to fail, Figure 3 presents the average number of routing failures of different variants of the Rand and Max TTGs on a range of Jellyfish and EGFT topologies. These figures show that a 3-resilient Max TTG can reduce the probability of routing failure by about three to four orders of magnitude when compared with non-resilient forwarding (NoRes), which is expected given that recent work has shown that linear increases in resilience result in exponential decreases in the average probability of routing failure [37]. This is an important result because we have previously shown that this TTG only reduces the aggregate throughput of the network by about 5–10%.

Additionally, these figures also show that even the 0-resilient variants of the Rand and Max TTGs are more fault tolerant than the NoRes TTG. Across a wide range of failures and topologies, the 0-resilient variants of the Rand and Max TTGs experience roughly an order of magnitude fewer routing failures than the NoRes TTG on average. This is important because, given 0 failures, the 0-resilient Rand and Max TTGs behave identically to the NoRes TTG. However, the big difference is that the 0-resilient Rand and Max TTGs use a TTG that allows for improved fault tolerance by allowing packets to transition between initial trees.

However, unlike with throughput, which was nearly equal for the 8-T 8-T variants and the 12-T 4-E variants, the 12-T 4-E variants are only slightly more fault tolerant. This is because not all initial trees in the Rand and Max TTGs provide the same level of resilience. What has a larger impact on fault tolerance than increasing the total number of trees is by increasing the number of trees set aside for guaranteed resilience, and the number of set aside trees is equal in the 8-T and 12-T variants with the same resilience.

Because Equation 1 predicts the state requirements of DF-EDST resilience, we omit any figures of the state requirements of different TTGs. Instead, we note that tree resilience requires 40Mbit of TCAM state to implement DF-EDST with 12-trees per destination and 4-way multipathing on a (B1) Jellyfish topology with about 3K hosts and a (B1) EGFT topology with about 5K hosts. For context, 40Mbit of TCAM state is the total amount of state available in Metamorphosis [38], a proposed SDN switch. Additionally, forwarding table state is reduced as bisection bandwidth is reduced. Given $1:4$ bisection bandwidth topologies, DF-EDST should scale to networks with about 8K hosts. However, if packets are not modified, the state requirements are increased. Both the EGFT and Jellyfish topologies require more 40Mbit for topologies with (B1) topologies with 2K hosts and (B2) and (B4) topologies with 3K hosts. Further, we find that the state requirements of of both the "12-T 4-E" and "8-T 8-E" are also nearly identical.

VII. Related Work

Although this work is motivated by recent research that uses lossless Ethernet to improve TCP or application performance [5], [6], [24], [25], [39], we avoid a detailed discussion of this work because it is orthogonal from deadlock-free or fault tolerant routing. Similarly, related work on deadlock-free routing for data centers has already been covered in Section II. Thus, we only focus on related work on local fast failover in this section, none of which are deadlock-free.

First, local fast failover groups have been a part of the OpenFlow standard since version 1.1 [40]. Although we are not aware of any switch that implements this part of the standard in hardware, DF-EDST resilience should be fully expressible in terms of OpenFlow fast failover groups.

The most closely related work to DF-EDST resilience was introduced by Elhourani *et al.* [17], who contributed an approach to resilience that uses arc-disjoint spanning trees (ADST) instead of EDSTs. On a k-connected network, there are k ADSTs per destination that can be used to provide $(k-1)$-resilience if tree failures are marked in packets.

Given the proof that DF-EDST is deadlock-free, it follows that arc-disjoint spanning trees (ADSTs) can also be used to provide deadlock-free local fast failover if the TTG is acyclic. However, because arcs are directed, ADSTs are directed trees, in contrast with EDSTS, which are undirected. This implies that that not every source and destination have a path defined by every ADST. Because of this, using ADSTs in this way could lead to limited path diversity and resilience.

Similarly, trying to use ADSTs to improve upon Theorem 2 is promising. If deadlock freedom is not required, ADST resilience may use a different set of ADSTs for each destination. Given a line TTG, ADST resilience does not need to modify packet headers. However, there is one large problem in using this forwarding model to improve upon Theorem 2. The proof that ADST resilience can build $(k-1)$-resilient routes relies on the property that the tree that is chosen after an arc failure must be the tree that includes the opposite arc of the failed arc, but this tree ordering may be different for each switch and set of failures. Given a line TTG, this ordering may not be respected. Thus, such a forwarding function is not guaranteed to be $(k-1)$-resilient. Currently, the resilience of ADSTs given an arbitrary topology is an open problem.

Although there has been much research on local fast failover, many existing implementations fail to meet our requirements. For example, ECMP, IP Fast Re-route [31], and Fat Tire [15] offer limited resilience. Packet re-cycling [41] and Borokhovich *et al.* [19] use inefficient paths. R-BGP [42] and F10 [16] rely on graph-specific properties. DDC [18] can temporarily incur significant stretch and can suffer from forwarding loops, although an IP TTL may terminate the forwarding of a packet, and KF [14] also allows loops.

Although these projects do not provide deadlock-free local fast failover, they can be complementary. For example, although DDC [18] can incur significant stretch and loop packets until a TTL expires, it can guarantee that a packet will reach its destination if the network is connected. Because DF-EDST resilience does not require all 8 VCs provided by DCB, lossy DDC can be implemented on one VC to provide connectivity for important or control traffic. Similarly, the fault tolerant routing algorithm presented by Borokhovich *et al.* [19] could also be used to ensure connectivity on a lossy VC without even allowing for forwarding loops.

VIII. Conclusions

In summary, we have introduced and evaluated two different approaches to implementing both deadlock-free and fault-tolerant forwarding for arbitrary data center networks, DF-FI resilience and DF-EDST resilience. First, we find that DF-FI resilience requires more virtual channels than are currently available to implement resilient forwarding on data center topologies with more than 2K hosts. Until further advances are made on the topic of guaranteeing that some configurations of backup routes are never possible and thus could never form deadlocks, DF-FI resilience is not a scalable solution.

However, this result also further motivates DF-EDST resilience. To enable DF-EDST resilience, we prove that DF-EDST resilience is deadlock-free as long as the TTG is acyclic. In DF-EDST resilience, the TTG determines both the performance and resilience of the forwarding function. Given this, we then evaluated the aggregate throughput, average probability of routing failure, and forwarding table state requirements of a number of different TTGs. We show that, with DF-EDST resilience, fault tolerant forwarding can be provided on any arbitrary well-connected data center topology without impacting throughput beyond that of using EDSTs to provide deadlock-free routing. In effect, the only extra cost of fault tolerance is additional forwarding table state. Moreover, if even three of the total EDSTs in the topology, of which there are often 10–20, are reserved for fault tolerance and guaranteed resilience (3-resilience), then the probability of routing failure can be reduced by 3–4 orders of magnitude with only a small impact on forwarding throughput.

Lastly, we evaluated the forwarding table sizes necessary to implement DF-EDST resilience. Even though implementing DF-EDST in our evaluation requires 40Mbit of TCAM state, the total amount of TCAM state provided by Metamorphosis [38], on topologies with at most about 8K hosts, we do not consider these forwarding table state requirements to be limiting. Because of the trade-off between fault tolerance and forwarding table state, fault tolerance can be reduced to enable DF-EDST on larger topologies. Further, these results motivate building switches with forwarding table hardware specifically designed for tree resilience. Given custom packet processing hardware that could search for the first non-failed tree, we believe the forwarding table state requirements of the resilient TTGs could be reduced to match that of using EDSTs for non-fault-tolerant but deadlock-free forwarding, $|D| * |V_{TTG}|$.

References

[1] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "DCTCP: Efficient packet transport for the commoditized data center," in *SIGCOMM*, 2010.

[2] Y. Chen, R. Griffith, J. Liu, A. D. Joseph, and R. H. Katz, "Understanding TCP Incast Throughput Collapse in Datacenter Networks," in *SIGCOMM*, Aug. 2009.

[3] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The nature of datacenter traffic: Measurements and analysis," *IMC*, 2009.

[4] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: measurement, analysis, and implications," in *SIGCOMM*, 2011.

[5] D. Zats, T. Das, P. Mohan, D. Borthakur, and R. H. Katz, "DeTail: Reducing the flow completion time tail in datacenter networks," EECS Department, University of California, Berkeley, Tech. Rep., 2012. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-33.html

[6] B. Stephens, A. L. Cox, A. Singla, J. Carter, C. Dixon, and W. Felter, "Practical DCB for improved data center networks," in *INFOCOM*, 2014.

[7] A. Phanishayee, E. Krevat, V. Vasudevan, D. G. Andersen, G. R. Ganger, G. A. Gibson, and S. Seshan, "Measurement and analysis of TCP throughput collapse in cluster-based storage systems," in *FAST*, 2008.

[8] R. Mittal, T. Lam, N. Dukkipati, E. Blem, H. Wassel, M. Ghobadi, A. Vahdat, Y. Wang, D. Wetherall, and D. Zats, "TIMELY: RTT-based congestion control for the datacenter," in *SIGCOMM*, 2015.

[9] 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*. ACM, August 2015. [Online]. Available: http://research.microsoft.com/apps/pubs/default.aspx?id=252307

[10] B. Stephens, A. L. Cox, and S. Rixner, "Plinko: building provably resilient forwarding tables," in *HotNets*, 2013.

[11] J. Feigenbaum, P. B. Godfrey, A. Panda, M. Schapira, S. Shenker, and A. Singla, "On the resilience of routing tables," in *Brief announcement, 31st Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*, July 2012.

[12] K. Lakshminarayanan, M. Caesar, M. Rangan, T. Anderson, S. Shenker, and I. Stoica, "Achieving convergence-free routing using failure-carrying packets," in *SIGCOMM*, 2007.

[13] P. Pan, G. Swallow, and A. Atlas, "RFC 4090 Fast Reroute Extensions to RSVP-TE for LSP Tunnels," May 2005.

[14] B. Yang, J. Liu, S. Shenker, J. Li, and K. Zheng, "Keep forwarding: Towards k-link failure resilient routing," in *INFOCOM*, 2014.

[15] M. Reitblatt, M. Canini, A. Guha, and N. Foster, "FatTire: Declarative fault tolerance for software defined networks," in *HotSDN*, 2013.

[16] V. Liu, D. Halperin, A. Krishnamurthy, and T. Anderson, "F10: A fault-tolerant engineered network," in *NSDI*, 2013.

[17] T. Elhourani, A. Gopalan, and S. Ramasubramanian, "IP fast rerouting for multi-link failures," in *INFOCOM*, 2014.

[18] J. Liu, A. Panda, A. Singla, P. B. Godfrey, M. Schapira, and S. Shenker, "Ensuring connectivity via data plane mechanisms," in *NSDI*, April 2013.

[19] M. Borokhovich, L. Schiff, and S. Schmid, "Provable data plane connectivity with local fast failover: Introducing OpenFlow graph algorithms," in *HotSDN*, 2014.

[20] B. Yener, Y. Ofek, and M. Yung, "Convergence routing on disjoint spanning trees," *Computer Networks*, vol. 31, no. 5, pp. 429 – 443, 1999.

[21] W. J. Dally and C. L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Transactions on Computers*, vol. C-36, no. 5, pp. 547–553, May 1987.

[22] J. Domke, T. Hoefler, and W. E. Nagel, "Deadlock-free oblivious routing for arbitrary topologies." in *IPDPS*. IEEE, 2011.

[23] "IEEE Data Center Bridging Task Group," http://www.ieee802.org/1/pages/dcbridges.html.

[24] A. Kalia, M. Kaminsky, and D. G. Andersen, "Using RDMA efficiently for key-value services," in *SIGCOMM*, 2014.

[25] A. Dragojević, D. Narayanan, M. Castro, and O. Hodson, "FaRM: Fast remote memory," in *NSDI*, 2014.

[26] J. Flich, T. Skeie, A. Mejía, O. Lysne, P. López, A. Robles, J. Duato, M. Koibuchi, T. Rokicki, and J. Sancho, "A Survey and Evaluation of Topology Agnostic Deterministic Routing Algorithms," *IEEE Transactions on Parallel and Distributed Systems*, 2011.

[27] M. Schroeder, A. Birrell, M. Burrows, H. Murray, R. Needham, T. Rodeheffer, E. Satterthwaite, and C. Thacker, "Autonet: A High-speed, Self-configuring Local Area Network Using Point-to-point Links," *IEEE Journal on Selected Areas in Communications*, 1991.

[28] X. Yuan, W. Nienaber, Z. Duan, and R. G. Melhem, "Oblivious routing for fat-tree based system area networks with uncertain traffic demands." in *SIGMETRICS*. ACM, 2007.

[29] T. Skeie, O. Lysne, and I. Theiss, "Layered Shortest Path (LASH) Routing in Irregular System Area Networks," *IPDPS*, 2002.

[30] P. M. Klausler, "Deadlock prevention in direct networks of arbitrary topology," May 14 2013, US Patent 8,441,933. [Online]. Available: http://www.google.com.ar/patents/US8441933

[31] P. Francois and O. Bonaventure, "An evaluation of IP-based fast reroute techniques," in *CoNext*, 2005.

[32] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, 2008.

[33] J. Mudigonda, P. Yalagandula, J. C. Mogul, B. Stiekes, and Y. Pouffary, "NetLord: a scalable multi-tenant network architecture for virtualized datacenters," in *SIGCOMM*, 2011.

[34] S. Ohring, M. Ibel, S. Das, and M. Kumar, "On generalized fat trees," *Parallel Processing Symposium, International*, vol. 0, p. 37, 1995.

[35] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey, "Jellyfish: Networking data centers randomly," in *NSDI*, April 2012.

[36] A. R. Curtis, J. C. Mogul, J. Tourrilhes, and P. Yalagandula, "DevoFlow: Scaling flow management for high-performance networks," in *SIGCOMM*, 2011.

[37] B. Stephens, A. L. Cox, and S. Rixner, "Scalable multi-failure fast failover via forwarding table compression," in *SOSR*. ACM, 2016.

[38] P. Bosshart, G. Gibb, H.-S. Kim, G. Varghese, N. McKeown, M. Izzard, F. A. Mujica, and M. Horowitz, "Forwarding metamorphosis: fast programmable match-action processing in hardware for SDN," in *SIGCOMM*, 2013.

[39] M. Alizadeh, A. Kabbani, T. Edsall, B. Prabhakar, A. Vahdat, and M. Yasuda, "Less is More: Trading a little Bandwidth for Ultra-Low Latency in the Data Center," *NSDI*, 2012.

[40] "OpenFlow switch specification, version 1.1.0," http://www.openflow.org/documents/openflow-spec-v1.1.0.pdf.

[41] S. S. Lor, R. Landa, and M. Rio, "Packet re-cycling: eliminating packet losses due to network failures," in *HotNets*, 2010.

[42] N. Kushman, S. Kandula, D. Katabi, and B. M. Maggs, "R-BGP: staying connected in a connected world," in *NSDI*, 2007.