Public Review for

Local Fast Failover Routing With Low Stretch

Klaus-Tycho Foerster, Yvonne-Anne Pignolet
Stefan Schmid, Gilles Tredan

Many techniques have been proposed to cope with link and node failures in networks. A popular approach is to rely on a dynamic routing protocol to detect failures and recompute the forwarding tables when needed. However, not all networks use dynamic routing protocols. This paper tackles the problem of computing static conditional forwarding tables that can cope with multiple failures. When flows need to be rerouted, an important metric is the path stretch. The authors provide a lower bound on the stretch achievable by any failover mechanism and propose algorithms that are optimal for many interesting topologies.
The reviewers liked the algorithms proposed in the paper, their evaluation and the proofs. This is not the final word on this topic and reviewers expect that the paper will spur additional interesting work in the area, e.g. by including traffic load in additional to the path stretch in the proposed algorithms.

Public review written by
Katerina Argyraki
EPFL

Local Fast Failover Routing With Low Stretch

Klaus-Tycho Foerster[1], Yvonne-Anne Pignolet[2], Stefan Schmid[3], Gilles Tredan[4]
[1] Aalborg University, Denmark, [2] ABB Corporate Research, Switzerland
[3] University of Vienna, Austria, [4] CNRS-LAAS, France
ktfoerster@cs.aau.dk, yvonne-anne.pignolet@ch.abb.com, stefan_schmid@univie.ac.at, tredan@laas.fr

ABSTRACT

Network failures are frequent and disruptive, and can significantly reduce the throughput even in highly connected and regular networks such as datacenters. While many modern networks support some kind of local fast failover to quickly reroute flows encountering link failures to new paths, employing such mechanisms is known to be non-trivial, as conditional failover rules can only depend on *local* failure information. While over the last years, important insights have been gained on how to design failover schemes providing *high resiliency*, existing approaches have the shortcoming that the resulting failover routes may be unnecessarily long, i.e., they have a large stretch compared to the original route length. This is a serious drawback, as long routes entail higher latencies and introduce loads, which may cause the rerouted flows to interfere with existing flows and harm throughput.

This paper presents the first deterministic local fast failover algorithms providing provable resiliency and failover route lengths, even in the presence of many concurrent failures. We present stretch-optimal failover algorithms for different network topologies, including multi-dimensional grids, hypercubes and Clos networks, as they are frequently deployed in the context of HPC clusters and datacenters. We show that the computed failover routes are optimal in the sense that no failover algorithm can provide shorter paths for a given number of link failures.

CCS Concepts

•Networks → Routing protocols;

Keywords

Fast Reroute, Static Resiliency, Network Algorithms

1. INTRODUCTION

Many computer networks are mission critical and dependability requirements are increasingly stringent. It is hence not surprising that network reliability is one of the main concerns of operators [21]. Ensuring a high network availability however is often non-trivial, especially under frequent and concurrent link failures, which are becoming more likely with the increasing scale of communication networks including datacenter [9], backbone [14,16] or enterprise [17] networks, but also due to virtualization and shared risk link groups [22].

For fast failover, most modern computer networks provide some kind of *local failover mechanism*: routers and switches store conditional forwarding rules which take effect only if an incident link or port is unavailable. In MPLS networks, this mechanism is known as *MPLS Fast Reroute*: upon a link failure, packets are sent along precomputed alternate paths. In Software-Defined Networks (SDNs) and in particular in OpenFlow, local fast failover is supported using group tables.

Local fast failover can be significantly faster than, e.g., reconvergence using link state IGPs or indirections via a (remote) OpenFlow controller. At the same time, local fast failover mechanisms are severely limited in terms of the possible routes that can be chosen *under multiple failures*: as failover decisions need to be pre-computed and can only rely on information about the availability of links directly incident to the respective router or switch, the resulting paths can be far from optimal. Accordingly, local fast failover mechanisms are often used as a first line of defense, after which more rigorous and global path reconfigurations are performed.

Over the last years, the question of how to compute local fast failover rules has received signficant interest by the networking community, and many solutions are known that provably ensure a high robustness even in the presence of multiple concurrent failures [3,4,5,6,15,16,18,19]. However, while the resiliency is fairly well-understood, less is known about the "quality" of the resulting failover routes, in terms of the *length* of the resulting detour.

Long failover routes (compared to the original route) are undesirable, for several reasons: Long routes require more resources as bandwidth needs to be allocated *on each additional hop*. This also makes it more likely that the load introduced by the rerouted flows leads to interference with existing flows, harming throughput. Longer routes are also likely to increase latency (additional queuing delay for each hop).

Contributions. This paper initiates the study of deterministic local fast failover algorithms which are not only very robust (to a large number of simultaneous failures) but which also account for the length (resp. the *stretch*) of the detours resulting from local rerouting.

In particular, we propose to allocate failover paths such that depending on the actual failures, flows *preserve distances to their destination.* We consider different fundamental network topologies, namely torus, hypercube, fat-tree (Clos), and BCube networks as they typically occur in high-performance clusters and datacenters. We present several failover algorithms and prove that they come with attractive worst-case guarantees: all algorithms presented in this paper ensure an *optimal* stretch, among the class of all deterministic local fast failover algorithms.

Example and Limitation of Prior Work. In order to illustrate our problem and show the need for route-length

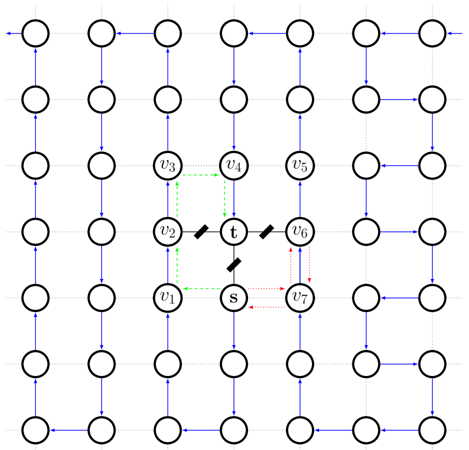

Figure 1: Example of a 7×7 torus with three failures (*crossed out links*) incident to the destination t, with three different failover strategies using *blue* (solid), *red* (dotted), and green (dashed) arcs, respectively.

aware algorithms, we consider a 2-dimensional torus in Fig. 1.

In this simple example, there is just a single flow, from s to t. In the absence of failures, s can communicate with t directly: the two nodes are adjacent. In case of incident link failures, any node v will apply alternative conditional rules which have been pre-installed. These rules can only be conditioned on the availability of links incident to v and the in-port, i.e., rerouting decisions are purely local. See later (and related work [4,6,15,18]) for a more detailed model.

We are interested in strategies to pre-compute such conditional failover rules which ensure that s is still able to route to t, even in the presence of multiple failures. In the example in Fig. 1, node s may have a conditional failover rule which reroutes traffic to v_7 if (s,t) is unavailable. Assume that v_7 will forward traffic to v_6, which cannot reach t either. If v_6 has a conditional rule to forward traffic to v_7 if link (v_6,t) is unavailable, a forwarding loop may result (indicated by dotted *red* arcs). Prior work provided important insights in how failover rules should be defined in order to avoid such loops, even under multiple link failures, for example, relying on Hamiltonian cycles [5]. However, in the worst case, these schemes can result in very long paths: if s resorts to the alternative route indicated by solid *blue* arcs, the stretch (i.e., the actual length of the failover route divided by the shortest possible distance) is in the order of $\Omega(n)$. Thus, we are aiming to ensure that failover paths "preserve locality", without sacrificing resiliency. In other words, we are interested in algorithms to compute static failover rules which result in paths like the one indicated by dashed *green* arcs.

Related Work. There exists much work on routing schemes resilient to single failures as well as routing schemes resilient to multiple failures with dynamic failover tables (hence link reversal approaches [8] can be used). However, not much is known about the design of resilient static forwarding tables, especially if packet-header rewriting or packet-duplication is impossible or undesired (the former consumes header space and the latter introduces additional loads).

The most closely related works to ours are by Chiesa et al. [5,6], Stephens et al. [18,19], and Pignolet et al. [15], who developed robust failover schemes using static forwarding tables. These approaches provide a very high resilience, however, they do not guarantee any non-trivial deterministic bounds for the resulting path lengths.

This paper aims to fill this gap, by initiating the study of deterministic local algorithms for short failover paths. In addition to new approaches, we build upon the concepts of *arc-disjoint spanning arborescences*, which were also exploited in prior work by Chiesa et al. [5,6] and are reminiscent of work for homotopic routing problems [12].

Organization. We present our formal model in § 2, and derive a lower bound on the stretch in § 3. § 4 describes optimal low-stretch failover algorithms for grids, tori and hypercubes, and § 5 describes optimal failover algorithms for Clos and BCube networks. We conclude in § 6.

2. MODEL

We consider a network $G = (V,E)$ (a so-called *symmetric digraph*) connecting n nodes (switches, routers, hosts) $V = \{v_1,\dots,v_n\}$ using bidirected (i.e., full-duplex) links E, that is, on a link (u,v) packets can be sent from u to v and in the opposite direction on (v,u). We assume that rules can match packet header fields as well as the in-port[1].

Each node v forwards packets according to two kinds of pre-installed flow rules:

1. *Default flow rules* describe the forwarding behavior for packets at v *if all links incident to v are available.*
2. *(Conditional) failover flow rules* describe how v should forward packets in case of incident link or node failures.

Both the default and the failover flow rules are pre-installed and static. In particular, at the time the failover rules are installed, let us call it t_0, the set F of directed link failures is not known yet. Moreover, for simplicity, we assume that the $f = |F|$ link failures occur simultaneously at some later time $t_1 > t_0$. While this simplifies the analysis presented later in this paper, our algorithms are also robust against failures occurring at different points in time.

Without loss of generality, it is sufficient to focus on an arbitrary single flow, from source s to destination t. Moreover, we do not allow packet tagging: while marking packets is known to improve the robustness of routing [3,6], it may not be impossible in practice to add additional header fields or to reuse existing fields, as they are needed by other protocols.

Our goal is to devise a (deterministic) algorithm A that computes failover rules for the network nodes such that it holds for every flow of packets from s to t:

1. *Resiliency:* The route taken by packets from s according to algorithm A leads to t despite f link failures, as long as there is still a path from s to t.
2. *Low stretch:* Let ℓ_0 denote the minimum distance packets have to travel from s to t without failures (shortest path routing) and let ℓ_1 denote the route length using algorithm A, given the worst possible f failures. A good failover routing algorithm minimizes the additive stretch, defined as $\sigma = \ell_1 - \ell_0 \geq 0$. Moreover, we consider *local stretch-optimal* failover algorithms, optimal algorithms in the class of local algorithms for which $\ell_1 \leq \ell_1^*$, where ℓ_1^* denotes the *worst case* route length of any deterministic local failover algorithm A^* subject to f worst case failures.

[1]The in-port is crucial for resiliency. E.g., consider a network with a dead-end, e.g., a node v which can only be reached via a link from u after the failures. As packets are forced to return back to u along link (v,u), i.e., the same link from which they arrived, matching the in-port is needed to facilitate a different routing decision at v, avoiding a loop.

3. A LOWER BOUND

We start by pointing out a fundamental limit for low-stretch local fast failover. The following lower bound on the stretch σ is expressed in terms of the network's girth (the length of the shortest undirected cycle contained in the network) and will also serve us to prove that the algorithms presented later are local stretch-optimal.

THEOREM 1. *Consider any network G of some given girth γ, and an arbitrary local failover algorithm ALG. Then, there exists a set of f link failures such that the length of the failover path of ALG increases by at least $(\gamma - 2) \cdot f$.*

PROOF. The claim follows by induction over the number of failures. Given a route from s to t we construct increasingly large failure sets by iteratively removing the *last* link of the current failover path chosen by ALG; the is the link whose destination is t. Note that the claim holds for the first failure: let $v_0 = s, v_1, \ldots, v_\ell = t$ be the route taken in absence of failures. By failing the last traversed link $e = (v_{\ell-1}, t)$, a failover path $v_0 = s, \ldots, v_{\ell-1}, v'_1, \ldots, v'_{\ell'} = t$ used for routing instead. We observe that in this case, i.e., by setting $F = \{e\}$, the route length increases by at least $\gamma - 2$ hops since the girth of G is γ and there cannot be a path from $v_{\ell-1}$ to t with fewer than $\gamma - 1$ hops.

For the induction step, we note that by deleting links, the girth cannot decrease. We use the following indistinguishability argument. Let F and F' be two different (possibly empty) sets of failed links. Moreover, let $\Gamma_X(v)$ be the incoming and outgoing links incident to v which have not failed, i.e., are not part of set X. Therefore, since ALG is local and deterministic, if $\Gamma_F(v) = \Gamma_{F'}(v)$ any node v receiving a packet from a given port will forward the packet the same way under F and F', even if $F \neq F'$: the failures are *locally indistinguishable*. Thus, for any path interval between two nodes u and v in which failure sets are indistinguishable, the routing paths $p_F(u, v)$ and $p_{F'}(u, v)$ will be the same.

When we fail the next link, $e' = (v'_{\ell'-1}, t)$, we know that nodes $v'_{\ell'-1}$ and t have not been visited before by the failover path and hence the failure of e' is not visible in the prefix of the path. Thus, we can again increase the path by at least $\gamma - 2$ hops, independently of the previous failures. □

Interestingly, the above theorem also holds in the case of undirected link failures, where each failure affects edges of both directions (i, j) and (j, i). We will use Thm. 1 later to show the optimality of our algorithms for various topologies.

4. GRIDS, HYPERCUBES, AND TORI

In this section, we will first present an algorithm to compute a stretch-optimal failover path for a 2-dimensional torus network. We will then move on and consider more general topologies, namely higher-dimensional grids and tori, as well as hypercubes, as they often occur in real networks.

4.1 The 2-Dimensional Torus

In this section we study the 2-dimensional torus, i.e., a 2-dimensional grid with *boundary* resp. wrap-around links. We first briefly review prior work, then present a scheme for $k_1 \times k_2$ tori, $k_1, k_2 \geq 5$, and lastly discuss smaller tori.

Hamiltonian cycles for the 2-dimensional torus. Chiesa et al. [5] designed a 3-resilient failover routing scheme that applies to the 2-dimensional torus by the specification of two bidirected-link-disjoint Hamiltonian cycles. If a packet encounters a failed link, the current Hamiltonian cycle is traversed in the other direction, and the next failed link on the path causes a switch to the second cycle, where the idea is repeated. The resiliency of this scheme is optimal due to the uniform out-degree of 4, but its stretch is $\Omega(n)$, see Fig. 1.

Optimal stretch for 2-dimensional tori of width ≥ 5. We now show how to combine optimality for both resiliency and stretch for any $k_1 \times k_2$ torus, as long as $k_1, k_2 \geq 5$. The general idea is to construct a geometric routing scheme, which "backs off" to a detour when a failure is encountered.

THEOREM 2. *Let G be a $k_1 \times k_2$ torus graph with $k_1, k_2 \geq 5$. There is a 3-resilient routing scheme for G with an additive stretch of $2f$ for $f \leq 3$ failures.*

PROOF. W.l.o.g., let t be centered at x_1, x_2-coordinates $(0, 0)$, and divide the grid into four quadrants of cyclic order: $(++)$ for $x_1 \geq 0, x_2 > 0$, $(-+)$ for $x_1 < 0, x_2 \geq 0$, $(--)$ for $x_1 \leq 0, x_2 < 0$, and $(+-)$ for $x_1 > 0, x_2 \leq 0$. Next, we fix a shortest-path routing tree T_t to be used for 0 failures, as shown in Fig. 2. For each node $v \neq t$, we denote the unique outgoing link towards t by $e_t(v)$ (omitting v if clear from the context). Another way to describe T_t is as follows:

- $(++)$: decrease x_1 to 0 when $x_1 > 0$, then decrease x_2
- $(-+)$: decrease x_2 to 0 when $x_2 > 0$, then increase x_1
- $(--)$: increase x_1 to 0 when $x_1 < 0$, then increase x_2
- $(+-)$: increase x_2 to 0 when $x_2 < 0$, then decrease x_1

We always assume an initial orientation towards the link e_t. The link directly counter-clockwise ("left") / clockwise ("right") is denoted by e_ℓ, e_r, respectively, and the remaining backward link is denoted by e_b. The forwarding mechanism can then be described in the following list items I, II, III:

I: If e_t is up and the incoming port is not e_t, then use e_t,

II: if node v is at the end of $e_\ell(w)$ with $x_1(w) = 0$ or $x_2(w) = 0$ and the in-port is $e_\ell(v)$, traverse $e_b(v)$,

III: else, traverse the following links, ordered by priority, if they are not failed: e_ℓ, e_b, e_r.

The correctness of the above provided algorithm can be proven by careful case distinction. Due to space constraints, we only sketch the main ideas in the following.

Observe that if the current quadrant has no link failures, we will route along a shortest path. Since there are at most three link failures, at least one quadrant will have no failures.

Next, assume that for every node v, it holds: if $e_t(v)$ failed, then $e_\ell(v)$ is up, i.e., at most one failure hits per node. When e_t failed and we fall back to using e_ℓ, the distance to the destination is decreased, unless v has a x_1 or x_2 coordinate of zero. In the latter case, as e_ℓ is always up, we will be at equal distance to t after two steps. As such, we will eventually reach a quadrant without failures (three failures allowed, four quadrants), reaching the destination. Furthermore, every failure induces an additional stretch of at most two.

For a packet to use e_r at some node, three failures at the current/previous node are required, after which t is reached by local rerouting with an additive stretch of at most 6. If a quadrant is switched, routing rule II can come into play, preventing the previous quadrant to be entered again.

To evoke the use of e_b as a boundary link, two failures at the current/previous node are required. The next quadrant can have at most one failure, where the minimum torus width of 5 ensures delivery within the promised stretch bounds.

Hence, we can now assume that no boundary links will be activated by our routing scheme, and that at most two

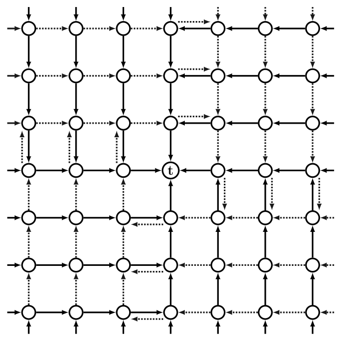

Figure 2: e_t rules, drawn in solid. The "secondary" rules e_ℓ are drawn dotted. Boundary links, here not drawn, are neither part of e_t or e_ℓ, all are e_b or e_r. Note that every node, except for t, belongs to exactly one quadrant.

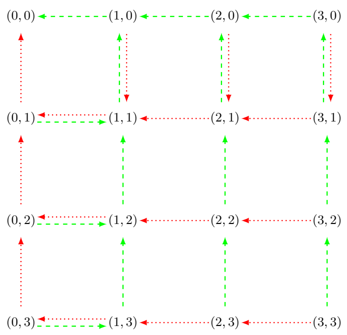

Figure 3: Two arc-disjoint arborescences as in Lemma [1] for a finite 4×4 grid, with destination $t = (0, 0)$ and roots $r_1 = (1, 0)$, $r_2 = (0, 1)$. The arcs of the trees T_1 and T_2 are drawn in green (dashed) and *red* (dotted), respectively.

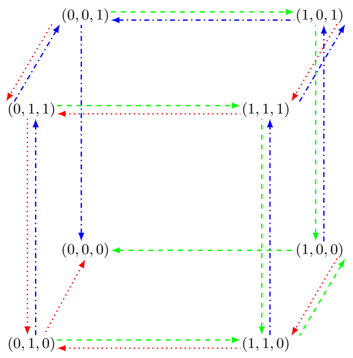

Figure 4: 3-dimensional Hypercube routing (finite $2 \times 2 \times 2$ grid) for the destination $t = (0, 0, 0)$, with three trees, drawn in green (dashed), *red* (dotted), *blue* (dashdotted).

failures hit a packet per node. I.e., when e_b is used by a packet (by two failures at the current node or by failing e_ℓ in addition to e_t being the incoming port), at most one more failure can occur subsequently (or some time before), yielding the desired stretch bounds of 4(+2). Due to the girth being 4, the additive stretch of $2f$ is optimal, see Thm. [1]. □

Small tori of width 3 and 4. Our above scheme remains two-resilient on smaller tori, as using boundary links requires two failed links in the current quadrant, leaving the other quadrants failure-free. Three failures can result in a loop though: E.g., on a $3 \times k_2$ torus graph, three link failures can change the distance from t to a $v \in N(t)$ from one to $k_2 - 1 \in \Omega(n)$. Furthermore, on $3 \times k_2$ tori, the girth is 3. Since the method of Chiesa et al. [5] is applicable to be 3-resilient on all $k_1 \times k_2$ tori with $k_1, k_2 \geq 3$, we conjecture that dedicated constructions with optimal additive stretch also exist for smaller 2-dimensional tori.

4.2 General Grids, Hypercubes and Tori

In this section, we present a resilient failover scheme for the d-dimensional finite grid with optimal additive stretch. Our scheme applies to hypercubes, as they are a special case of finite grids. Furthermore, we can utilize our scheme for tori, where stretch is optimal for a width of ≥ 4.

Preliminary definitions. We define a d-torus as a d-dimensional torus graph with $k_1 \times \cdots \times k_d$ nodes and integral coordinates $\left(\left[-\left\lfloor\frac{k_1-1}{2}\right\rfloor, \left\lfloor\frac{k_1}{2}\right\rfloor\right], \ldots, \left[-\left\lfloor\frac{k_d-1}{2}\right\rfloor, \left\lfloor\frac{k_d}{2}\right\rfloor\right]\right)$, $k \geq 3$, and the destination t being located at $(0, \ldots, 0)$. In addition to the links between nodes at distance 1 from each other, there are links between nodes $(x_1, \ldots, x_d), (x'_1, \ldots, x'_d)$ with $x_i = -\left\lfloor\frac{k_i-1}{2}\right\rfloor$ and $x'_i = \left\lfloor\frac{k_i}{2}\right\rfloor$ and $x_j = x'_j$ for all $j \neq i$, the so-called boundary links. We will also study finite d-grids, which are d-tori, with $k \geq 2$, but without boundary links. A special case are hypercubes, where $k = 2$ holds in all dimensions. W.l.o.g the destination t is set to $(0, \ldots, 0)$, adjusting the other coordinates accordingly. Note that t can, e.g., be in a "corner", where t has only d neighbors.

Upper bounds on resiliency. From prior work it is implied that d-dimensional hypercubes, a special case of finite d-grids, have a resiliency of $d - 1$ [5]. To the best of our knowledge, static failover resiliency for higher-dimensional d-tori was not yet formally studied. A resiliency of $2f - 1$ for $f \leq d$ can be proven by combining methods in [5, 20].

For the upcoming proof of Theorem [3] we leverage *rooted spanning arborescences*, a known approach for resilient routing [6]. For directed graphs, let (u, v) denote a directed arc from node u to v. A directed subgraph T is an r-rooted spanning arborescence of (the directed version of) G if (i) $r \in V(G)$, (ii) $V(T) = V(G)$, (iii) r is the only node without outgoing arcs and (iv), for each $v \in V \setminus \{r\}$, there exists only a single outgoing edge and a directed path from v to r.

When clear from the context, we use the term arborescence to refer to an r-rooted spanning arborescence. A set of arborescences $\mathcal{T} = \{T_1, \ldots T_k\}$ are arc-disjoint if no pair of arborescences in $\mathcal{T}$ share common directed arcs, i.e., if $(u, v) \in E(T_i)$ then $(u, v) \notin E(T_j)$ for all $i \neq j$.

It is known that k arc-disjoint arborescences exist in any k-connected graph [7] and can be computed efficiently [2], in a runtime of $O(|E| + nk^3 \log^2 n)$. As thus, the total complexity of computing our following failover schemes will be low, even more so for the already discussed case of the 2-dimensional torus, where we provided the rules explicitly.

Theorem 3. *There is a resilient routing scheme for the d-dimensional finite grid or hypercube with optimal additive stretch of $2f$ for $f \leq d - 1$ failures.*

Our proof will rely on the following Lemma [1], giving us d pairwise arc-disjoint shortest path arborescences to the d neighbors of the destination in an d-dimensional orthant. A d-dimensional orthant is the d-dimensional analogon of the 2-dimensional quadrant or 3-dimensional octant. For ease of construction, we only consider orthants that have a width of at least two nodes in every dimension.

Lemma 1. *Let G be a finite d-dimensional grid or hypercube with destination t, and let G' be a d-dimensional orthant of G. Let $N'(t)$ be the d neighbors of t in G'. For each $v \in N'(t)$, there is a shortest path arborescence T_v in $G' \setminus \{t\}$, such that these d arborescences are pairwise arc-disjoint.*

Proof. W.l.o.g., we can assume that G' is the d-dimensional orthant of G where all coordinates are non-negative. For completeness reasons, we start with $d = 1$: the whole orthant is just a line, where we orient the only possible arborescence T_{x_1} towards t. For illustrative purposes, the two arborescences for $d = 2$ are presented in Fig. [3]. Formally, for $d = 2$, we use the 1-dimensional construction for both arborescences T_1, T_2, with $r_1 = (1, 0)$ and $r_2 = (0, 1)$, as follows:

the 1-dimensional orthant $G'_{T_1,1}$ spanned by $(0,0) + x'_1(1,0)$, $x_1 \in \mathbb{N}_{\geq 0}$ has the same arborescence as in in $d = 1$, analogously for T_2 in $G'_{T_2,1}$ with $(0,0) + x'_2(0,1)$, $x_2 \in \mathbb{N}_{\geq 0}$. Arcs incident to nodes not in $G'_{T_1,1} \cup G'_{T_2,1}$ are added to the arborescences as follows: for T_1, add arcs that are directed such that they always decrease the distance in the x_2-coordinate, while for T_2, always the distance in the x_1 coordinate is decreased. For T_1 in nodes in $G'_{T_2,1}$, we add arcs that enforce the following rule: increase the distance in x_1, analogously for T_2 in $G'_{T_1,1}$. Both arborescences are arc-disjoint, and provide shortest paths, as they always decrease the distance to the respective destination r_1, r_2 in every hop, see Fig. 3.

We will now use $d = 2$ as a base case for induction. Assuming Lemma 1 holds for the case of d dimensions, we will show how to extend it to $d + 1$ dimensions.

In a $d+1$-dimensional orthant G'_{d+1}, there are $d+1$ different d-dimensional orthants spanned from the destination by a linear combination of d of the $d+1$ different root coordinates as vectors. However, each of these d-dimensional orthants only has "blueprints" for d different trees, and we have no routing rules for nodes outside of these orthants.

We first perform an intermediate step, where we match each of the $d + 1$ d-dimensional orthants with one root that takes part in spanning it, where the chosen blueprint is defined by the contained remaining roots. A perfect matching always exists, applying Hall's marriage theorem [11]. For ease of notation, we denote the $d + 1$ d-dimensional orthants by $G'_{T_1,d}, \dots, G'_{T_{d+1},d}$. For an example with $d = 3$, see Fig. 4, where the 3 contained $2d$ blueprints are provided by Fig. 3.

We now add the outgoing arcs for nodes v not contained in the $d + 1$ orthants for each T_i, such that the distance to $G'_{T_i,d}$ along the vector orthogonal to $G'_{T_i,d}$ is decreased.

It remains to specify the arcs in the $d + 1$ d-dimensional orthants. First, we apply the d-dimensional routing blueprint to all d roots contained in the respective orthant. For each of the $d + 1$ d-dimensional orthants, it remains to add the arcs for the one root not contained in it. To this end, we assign the outgoing arc for the respective arborescence along the vector orthogonal to the respective orthant.

Observe that the constructed arborescences inside the orthants are arc-disjoint by the induction assumption. By construction, when considering all these rules in total, along with the arcs incident to nodes not contained in any of the $d + 1$ d-dimensional orthants, we have arc-disjointness. For the arcs constructed in the previous paragraph, observe that they are in the opposite direction of the arcs "entering" the orthants, i.e., all $d + 1$ arborescences are arc-disjoint, and as they decrease the distance to their root in every step, they have the shortest path property in grids. $\square$

Putting it all together. Chiesa et al. [6] showed how decompositions of G into arborescences $\mathcal{T}$ can be used for failover routes: when encountering a failed link at node v, v forwards the packet along a different arborescence T_j, where the current tree is determined by the incoming port. By defining a circular order on the arborescences, one obtains $(d-1)$-resiliency of such circular-arborescence routing on d arc-disjoint arborescences, as each directed link/arc is used in at most one arborescence, which we use to prove Thm. 3.

Proof. We define a circular order on the arborescences promised by Lemma 1, switching to the next arborescence in case a failure is encountered, where the current arborescence can be identified by the incoming port. As the arborescences are provided only for each orthant, we remove multiple outgoing arborescence links at the orthant boundary nodes arbitrarily, until only one remains, obtaining d arc-disjoint arborescences for the whole graph, yielding the stated resiliency of $d - 1$. It remains to show the stretch property. As the arborescences have the shortest path property, switching between two arborescences induces only an additive stretch of at most 2, with the girth of finite grids and hypercubes being 4. By starting the routing on an arborescence with minimum distance to the destination t, the theorem follows. $\square$

Corollary 1. *There is a resilient routing scheme for the d-dimensional tori of minimum width 4, with optimal additive stretch of $2f$ for $f \leq d - 1$ failures.*

Proof. Resiliency follows analogously by ignoring the boundary links, but it remains to show the stretch property. However, as we "centered" the destination t in the torus, shortest paths to the destination in a current orthant are also always shortest paths in the torus. Lastly, a minimum width of 4 enforces a girth of 4. $\square$

5. CASE STUDIES: CLOS AND BCUBE

The failover algorithms presented so far addressed grid and hypercube graphs connecting *"nodes"*, which raises the question how to instantiate these approaches for actual datacenter networks connecting *servers* and *routers/switches*, using topologies such as Clos [1], F10 [13] and BCube [10]. However, it turns out that our techniques for low-stretch rerouting are directly applicable to such networks as well, sometimes even at an improved robustness.

First, topologies like Clos and F10 are decomposable into trees of connected complete bipartite graphs and can be made resilient using the circular arborescence constructions for *shared-link-failure-free routing functions* introduced in [5,6]. It is easy to see that here the stretch is bounded by an additive term of $2f$, where $f < k$ and k is the number of partially overlapping rooted trees describing the Clos and F10 topology, using a simple induction argument over the tree layers. Since Clos and F10 networks form bipartite graphs with girth 4 we can apply Theorem 1 to derive the optimality of this construction with respect to stretch.

A second important class of datacenter topologies are formed by hypercubic networks. As an example, we consider the BCube: BCube topologies are organized in k layers of server blocks of size n. Each of the total n^k servers is provided with a k-digits address written in base n. That is, server a has $address(a) = a_0 a_1 \dots a_{k-1}$ with $a_i \in \{0, 1, \dots, n-1\}, \forall i$. At the physical level, each server a is connected to k n-port switches: these switches connect a to all other servers at hamming distance 1 from a (that is, to servers b where $\sum_{i=0}^{k-1}(address(b)[i] \neq address(a)[i]) = 1$). A switch deployed at level i will connect n servers sharing identical addresses with the exception of the i^{th} digit.

Therefore, at the logical (server-to-server) level, all servers at hamming distance one are connected as a clique. In particular, consider a BCube with parameter $n = 2$, which is essentially a standard k-dimensional hypercube at the logical level. We exploit this observation to apply the low-stretch failover algorithm for hypercubes to the physical topology of BCubes as follows. Let $G = BCube(n, k)$. Let $s \in V$ a source of address $s_0 s_1 \dots s_{k-1}$. We can assume without loss of generality that the target address $t = t_0 \dots t_{k-1} = 0^k$. Consider the set of servers V' whose address is the combination of the address elements of s and the address elements of d when they differ. For elements where s and t are the same, we pick the values according to a permutation σ to add further servers to the set. Thus we have $|V'| = 2^k$, i.e., every address element of servers in V' can be one of two values. Formally, $V' = \{v \in V \text{s.t.} \forall i \in 0..k-1, (address(v)[i] = t_i)$ or $((address(v)[i] = s_i$ if $s_i \neq t_i)$ or $(address(v)[i] = \sigma(d_i)$ if $s_i = t_i))\}$. Consider the subgraph of G induced by those nodes: it is the hypercube $H(k)$. Moreover, observe that this hypercube nicely spans the physical topology: each node has a degree k in $H(k)$, and each of those k neighbors is reachable through a different switch.

Deploying a circular arborescence scheme on $H(k)$ is thus possible, since each node has a direct correspondence between its in-port and the corresponding neighbor in the topology, and since each node knows its position in $H(k)$ by comparing its address with the destination's address.

One interesting observation to make here (and in contrast to the graph models considered in previous sections) is that the flows between servers can be even more resilient than the degree of the individual switches, i.e., the local failover routing scheme can tolerate complete switch failures. In such a setting, each 2-port switch connects a pair of servers, and hence, at the physical level, the network is only 2 connected. However, using robust routing, we can ensure $(k-1)$-resiliency for flows between servers: the redundancy offered by the k switches provides servers with a greater robustness.

6. CONCLUSION

This paper presented the first deterministic local fast failover algorithms which guarantee not only resiliency but also short routes. As such, our solutions can lead to significantly shorter failover routes with optimal stretch.

We described our algorithms in terms of bidirected links. Our approach has however implications also for undirected failures, where an adversary fails links in "both directions". In such a scenario our 2d-torus algorithm has identical performance. However, when using algorithms relying on directed arborescences, a single undirected failure can impact two arborescences. Thus, in this case, the resiliency is halved and the stretch is doubled.

Our work opens several interesting directions for future research. For example, this paper has focused on a hop distance metric, which is natural to capture resource allocations and loads; however, it would be interesting to generalize our results to arbitrary link weights (e.g., representing latencies). The main open question in terms of algorithms however concerns low-stretch failover algorithms for more general graph classes or even arbitrary graphs. The study of rerouting techniques with header re-writing constitutes another interesting direction for future research.

Acknowledgments. Research partly supported by the Villum project ReNet & Aalborg University's PreLytics project.

7. REFERENCES

[1] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. *ACM SIGCOMM CCR*, 38(4):63–74, 2008.

[2] A. Bhalgat, R. Hariharan, T. Kavitha, and D. Panigrahi. Fast edge splitting and edmonds' arborescence construction for unweighted graphs. In *Proc. SODA*, 2008.

[3] M. Borokhovich, L. Schiff, and S. Schmid. Provable data plane connectivity with local fast failover: Introducing openflow graph algorithms. In *Proc. ACM SIGCOMM HotSDN*, 2014.

[4] M. Borokhovich and S. Schmid. How (not) to shoot in your foot with sdn local fast failover: A load-connectivity tradeoff. In *Proc. OPODIS*, 2013.

[5] M. Chiesa, A. Gurtov, A. Madry, S. Mitrovic, I. Nikolaevkiy, A. Panda, M. Schapira, and S. Shenker. Exploring the limits of static failover routing (v4). *arXiv:1409.0034 [cs.NI]*, 2016.

[6] M. Chiesa, I. Nikolaevskiy, S. Mitrovic, A. V. Gurtov, A. Madry, M. Schapira, and S. Shenker. On the resiliency of static forwarding tables. *IEEE/ACM Trans. Netw.*, 25(2):1133–1146, 2017.

[7] J. Edmonds. Edge-disjoint branchings. *Combinatorial algorithms*, 9(91-96):2, 1973.

[8] E. Gafni and D. Bertsekas. Distributed algorithms for generating loop-free routes in networks with frequently changing topology. *Trans. Commun.*, 29(1):11–18, 1981.

[9] P. Gill, N. Jain, and N. Nagappan. Understanding network failures in data centers: measurement, analysis, and implications. *SIGCOMM CCR*, 41:350–361, 2011.

[10] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu. Bcube: a high performance, server-centric network architecture for modular data centers. In *Proc. ACM SIGCOMM*, 2009.

[11] P. Hall. On representatives of subsets. *Journal of the London Mathematical Society*, s1-10(1):26–30, 1935.

[12] M. Kaufmann and K. Mehlhorn. A linear-time algorithm for the homotopic routing problem in grid graphs. *SIAM J. on Computing*, 23(2):227–246, 1994.

[13] V. Liu, D. Halperin, A. Krishnamurthy, and T. E. Anderson. F10: A fault-tolerant engineered network. In *Proc. USENIX NSDI*, 2013.

[14] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, and C. Diot. Characterization of failures in an ip backbone. In *Proc. IEEE INFOCOM*, 2004.

[15] Y.-A. Pignolet, S. Schmid, and G. Tredan. Load-optimal local fast rerouting for dependable networks. In *Proc. DSN*, 2017.

[16] S. Schmid and J. Srba. Polynomial-time what-if analysis for prefix-manipulating mpls networks. In *Proc. IEE INFOCOM*, 2018.

[17] A. Shaikh, C. Isett, A. Greenberg, M. Roughan, and J. Gottlieb. A case study of ospf behavior in a large enterprise network. In *Proc. IMW*. ACM, 2002.

[18] B. Stephens, A. L. Cox, and S. Rixner. Plinko: Building provably resilient forwarding tables. In *Proc. ACM HotNets*, 2013.

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

[20] R. Stong. Hamilton decompositions of cartesian products of graphs. *Disc. Math.*, 90(2):169 – 190, 1991.

[21] Y. Wang, H. Wang, A. Mahimkar, R. Alimi, Y. Zhang, L. Qiu, and Y. R. Yang. R3: resilient routing reconfiguration. *ACM SGICOMM CCR*, 40(4):291–302, 2010.

[22] D. Xu, Y. Xiong, C. Qiao, and G. Li. Failure protection in layered networks with shared risk link groups. *IEEE network*, 2004.