Local Fast Rerouting with Low Congestion: A Randomized Approach

Gregor Bankhamer [†]
Department of Computer Sciences
University of Salzburg, Austria
gbank@cs.sbg.ac.at

Robert Elsässer [†]
Department of Computer Sciences
University of Salzburg, Austria
elsa@cs.sbg.ac.at

Stefan Schmid
Faculty of Computer Science
University of Vienna, Austria
stefan_schmid@univie.ac.at

***Abstract*—Most modern communication networks include fast rerouting mechanisms, implemented entirely in the data plane, to quickly recover connectivity after link failures. By relying on *local* failure information only, these data plane mechanisms provide very fast reaction times, but at the same time introduce an algorithmic challenge in case of multiple link failures: failover routes need to be robust to additional but locally unknown failures downstream.**

This paper presents local fast rerouting algorithms which not only provide a high degree of resilience against multiple link failures, but also ensure a low congestion on the resulting failover paths. We consider a randomized approach and focus on networks which are highly connected before the failures occur. Our main contribution are three simple algorithms which come with provable guarantees and provide interesting resilience-load tradeoffs, significantly outperforming *any* deterministic fast rerouting algorithm with high probability.

I. Introduction

Emerging applications, e.g., in the context of industrial, tactile or 5G networks, come with stringent latency and dependability requirements. To meet such requirements, many communication networks feature Fast Re-Route (FRR) mechanisms [1], [2], [3], [4]: *local* failover mechanisms in the data plane which avoid the time-consuming advertisement and collection of failure information and re-computation of routes in the control plane [5], [6]. Rather, these mechanisms rely on a *pre-defined* logic, often implemented in terms of conditional failover rules [7]. However, while such mechanisms are attractive and widely used to deal with single failures, they introduce an *algorithmic challenge* in the presence of *multiple* link failures, as they are common in large networks such as datacenter and Internet networks [8], [9], [10], [11], [12], [13]: rerouting decisions need to be made based on *incomplete* information about the failure scenario, and in particular, about failures *downstream*. The problem becomes particularly challenging if the rerouted flows should not only preserve connectivity under failures but also a low *load*, an important criteria in practice: congested routes threaten dependability and indeed, congestion is a main concern of any traffic engineering algorithm.

Recently, a series of negative results have been obtained on what can be achieved using *deterministic* fast rerouting algorithms (e.g., [14]). In particular, it has been shown that even on networks which are still highly-connected after failures, the congestion resulting from *any* deterministic local fast failover algorithm is bound to be high in the worst case [15], [16], i.e., *polynomial* in the number of link failures.

This paper initiates the study of *randomized* algorithms to provide high resiliency and low congestion at the same time. In particular, we will show that using a randomized approach, the congestion can be reduced from polynomial to polylogarithmic, with high probability, hence breaking deterministic congestion lower bounds.

A. Model in a Nutshell

In a nutshell, we consider the fundamental problem of congestion-minimal fast rerouting on a complete undirected network $G = (V, E)$ connecting n nodes (routers and hosts). Such complete networks are typically studied in the related work and can be seen as an approximation of highly-connected networks as they arise, e.g., in the context of datacenters. Furthermore, it is known that solutions for complete networks translate into multihop solutions in certain cases and hence form building blocks. More details will follow.

The network links (henceforth called *edges*) of G are subject to multiple concurrent failures, determined by an *oblivious* adversary: an adversary who knows the failover protocol including the used probability distributions, but not the generated random values nor the resulting loads. (Later in this paper we will discuss even stronger adversaries.)

The goal is to pre-define local failover rules for the different nodes V such that traffic is rerouted to the destination while balancing the *load*: the maximal number of flows (a continuous stream of packets) crossing any node or link in the network. The failover rules, which depend on the set of failed edges incident to the given node as well as certain packet header information, should be as simple as possible. Also, these rules should be static, i.e., the routing table is not allowed to be updated during the whole routing procedure, except in the case of some "bad" event – which can only happen with a low probability in the adversarial model we consider (see Section V for a discussion). For a brief overview on the applicability of different failover routing mechanisms, see Section I-D. In particular, we in this paper consider three such *destination-based* rule types: one where only the destination

[†] The authors' work was supported by grant no. P 27613 of the Austrian Science Fund (FWF), "Distributed Voting in Large Networks"

can be matched, one where we can also match the hop count, and one where we can modify the hop count. To measure the performance of our failover approaches we assume that the network follows a *all-to-one* communication pattern [15], [16]: one of the V nodes is selected as a distinguished destination d, and one flow originating from each node $V \setminus \{d\}$ needs to be routed to d. The adversary may know d.

B. The Deterministic Case Lower Bound

The authors of [15] showed that deterministic failover algorithms are bound to result in a high load even in case of an initially completely connected network which is still highly connected after the failures [15]. The proof has been generalized further by Pignolet et al. in [16]. More specifically, the authors showed that: (1) when only relying on destination-based failover rules, an adversary can always induce a maximum edge load of $\Omega(\varphi)$ by cleverly failing φ edges; (2) when failover rules can also depend on the source address, an edge load of $\Omega(\sqrt{\varphi})$ can be achieved, when failing φ many edges.

When considering the node load only, this bound can be extended and account for further information that may be used by the routing rules. Particularly, if we require that some packet starting from node v takes the same path under the same set of underlying edge failures (i.e., the packets' paths are oblivious and may not change depending on the other traffic moving around the network), a node load of $\Omega(\sqrt{\varphi})$ can be generated by the adversary. Note that this extension allows for including the hop counter inside the routing rule without weakening the result of the lower bound.

C. Our Results

The main contribution of this paper are three randomized fast rerouting algorithms which not only provide a high resilience to multiple link failures but also an exponentially lower load than any possible deterministic algorithm.

We present the three failover strategies in turn. Assuming up to $\varphi = O(n)$ edge failures, the first algorithm ensures that a load of $O(\log n \log\log n)$ will not be exceeded at most nodes, while the remaining $O(\text{polylog}\, n)$ nodes reach a load of at most $O(\text{polylog}\, n)$. As we consider randomized approaches, we require the above statement to hold *with high probability*[1]. The second approach we present reduces the edge failure resilience to $O(n/\log n)$, however it is purely *destination-based* and achieves a congestion of only $O(\log n \log\log n)$ at *any* node w.h.p. Finally, by assuming that the nodes do have access to $O(\log n)$ shared permutations of V, which are not known to the adversary, the node load can be reduced even further. That is, a maximum load of only $O(\sqrt{\log n})$ occurs at *any* node w.h.p.

While our focus lies on complete networks, these networks constitute a major open problem in the literature today and it is known that such solutions may be generalizable to other networks using specific network decompositions based on arborescence covers [17], [18] (namely if the cover is based on Hamiltonian cycles). Furthermore, such networks provide a good approximation of datacenter networks which are usually highly connected, providing (almost) full bisection bandwidth.

[1]Here we use the well established notion of *with high probability*, or w.h.p. , denoting probability of at least $1 - n^{-\Omega(1)}$.

D. Further Related Work

Link failures are the most common failures in communication networks [19], [20], [21], [22] and it is well-known that ensuring connectivity via the control plane can be slow [4], [14], even if it is centralized [23] or based on link reversal [24], [25], [26], [27]. Data plane based failover mechanisms which do not require table reconfigurations can be orders of magnitudes faster [14] but are algorithmically difficult to configure. Feigenbaum et al. [14] showed that it is not possible to achieve an "ideal (static) resilience" in arbitrary networks using local fast rerouting and without header rewriting [17], [28]: it is impossible to define failover rules such that connectivity is preserved as long as the network is physically connected. Furthermore, Chiesa et al.'s conjecture [29], [30] that is at least always possible to preserve connectivity in a k-(edge-)connected network if there are at most $k-1$ link failures, remains an open problem. Existing results on failover algorithms either do not account for load (see e.g., the interesting randomized failover approaches in [29], [30]) or are deterministic [31], [32]. Note that in [29], [30] the authors mainly focus on static-routing-resiliency of different procedures in arbitrary k-connected graphs, where k can be any number between 2 and n, and derive several important results. Their randomized algorithm is based on arborescence-based routing, which requires the precomputation [33] and the (distributed) storage of sets of certain spanning trees, leading to significant memory complexity. Furthermore, as randomized decisions have to be taken during the routing of each packet, possible TCP-flows may be disrupted.

E. Organization

The remainder of this paper is organized as follows. In the following Sections II to IV we present our three algorithms and provide an extensive analysis of their behavior. Then, in Section V, we discuss extensions of our results, considering additional failure scenarios as well as stronger adversaries. After reporting on simulation results in Section VI, we conclude in Section VII. Due to space constraints, some simple proofs are omitted in this version of the paper.

II. Beating Deterministic Approaches with Three Permutations

This section presents our first failover algorithm. While it is simple as forwarding is only based on the destination and hopcount header fields, it ensures w.h.p. very low loads even under a large number of link failures.

From the point of view of some fixed node v the first protocol, we call it *3-Permutations*, works as follows. For destination d, the node v stores three permutations $\pi^{(1)}_{v,d}$, $\pi^{(2)}_{v,d}$ and $\pi^{(3)}_{v,d}$ of all nodes $u \in V \setminus \{d\}$. Upon receiving a packet p intended for d, the node v first tries to forward it directly via the link (v, d). In case this link failed, v inspects the

```
Input: A packet with destination $d$ and hop count $h(p)$
1: if $(v,d)$ is intact then forward $p$ to $d$ and return
2: else set $i = \arg\max_{j\in\{1,2,3\}}\{h(p) \geq (j-1)C_1\}$ and send $p$ to first directly reachable node in $\pi_{v,d}^{(i)}$
3: increase $h(p) \mathrel{+}= 1$
```

Fig. 1. *3-Permutations* protocol. Point-of-view of some node v

current hop counter of p, denoted by $h(p)$. Depending on $h(p)$, the node v then chooses one of the three permutations $\pi_{v,d}^{(i)}$ and forwards the packet to the first *reachable* partner w in this permutation. We call a node w *reachable* from v, if the direct link (v,w) is not failed. The criteria for selecting which permutation to use is simple. In case $h(p) < C_1$ for a value $C_1 = O(\log n)$, permutation $\pi_{v,d}^{(1)}$ is consulted. For $C_1 \leq h(p) < 2C_1$ the permutation $\pi_{v,d}^{(2)}$ is used and in any remaining case $\pi_{v,d}^{(3)}$ is utilized. In any case the packets hop counter is increased by 1 before handing it to the next node. A concise description is given in Fig. 1.

Our main contribution is related to the way how these permutations are selected. Instead of opting for a deterministic protocol, we assume that each node v chooses the permutations $\pi_{v,d}^{(i)}$ out of all possible permutations of nodes $u \in V \setminus \{d\}$ uniformly and at random. As the adversary is *oblivious*, these permutations are not known to it and it needs to essentially blindly select edges for manipulation. Note however that this approach comes with a challenge. This random creation of failover routes may introduce temporary cycles into the packets routing paths. One important idea of the *3-Permutations* protocol is that most packets p will reach destination d solely relaying on the failover entries given by $\pi_{v,d}^{(1)}$. And, only in case p ends up trapped in a cycle, further permutations will be used allowing it to escape said cycle. We show that w.h.p., at most $O(\log^2 n \log\log n)$ load will reside at any node, even if the adversary is allowed to destroy a linear amount of edges.

Theorem 1. *Assume that the adversary fails at most $\alpha{\cdot}n$ edges where $\alpha < 1$ is a non-negative constant. Then, if all nodes perform all-to-one routing to any destination d and follow the* 3-Permutations *protocol, a maximum of*

$$O(\log n \cdot \log\log n)$$

flows will pass at all but $O(\log^2 n)$ nodes. Furthermore, the remaining nodes, except for d, will receive a load of at most $O(\log^2 \cdot \log\log n)$ and every packet will travel $O(\log n)$ hops. These statements hold w.h.p.

Now, in order for the nodes to follow this protocol they require the value C_1. This value upper-bounds the number of hops needed by any packet to reach the destination d, unless it is trapped in a cycle due to the permutation $\pi_{v,d}^{(1)}$. We will see that C_1 can be bounded from above by $16\log_{(1/\alpha)} n$. If α is not known to the nodes, then C_1 can be set to some value in $\omega(\log n)$. This slightly changes the result of Theorem 1 where up to $O(\log^2 n)$ many nodes will receive a load of $O(C_1 \cdot \log n \log\log n)$.

When it comes to memory complexity, each node may store 3 permutations of n nodes for every destination d. A naive approach would therefore require routing tables of size $O(n^2 \log n)$ bits to be prepared for routing to any arbitrary destination d. This can be overcome as follows. First, each node v only computes 3 random permutations $\pi_v^{(i)}$, $i = 1,2,3$ on *all* nodes. The permutation $\pi_{v,d}^{(i)}$ for each $d \in V$ is simply $\pi_v^{(i)}$, and thus we apply $\pi_v^{(i)}$ to obtain our failover strategy regardless of d. Note that if the edge (v,d) is not failed, then any packet with target d that reaches v is sent directly from v to d. If, however, (v,d) is failed, then such a packet is sent to the first node w in $\pi_v^{(i)}$ for which (v,w) is not failed. Secondly, note that the nodes only consult their permutations up until the first reachable node (see Line 2 of Fig. 1). Even if all αn failed edges are incident to the same node v, then at least one of the first $3\log_{(1/\alpha)} n$ nodes in each of the permutations $\pi_{v,d}^{(i)}$ will be directly reachable from v w.h.p. This follows from the fact that the adversary does not know the random bits generated at some node. Therefore, nodes may truncate the permutations, storing only the first $3\log_{(1/\alpha)} n$ entries of each permutation. In the low probability event that none of the first $3\log_{(1/\alpha)} n$ is directly reachable from v (due to the failed edges), another $3\log_{(1/\alpha)} n$ nodes are selected uniformly at random – without replacement. Employing these improvements yields an improved total memory complexity of $O(\log^2 n/\log(1/\alpha)) = O(\log^2 n)$ per node.

In the following, we will perform the analysis w.r.t some arbitrary but fixed destination d. As we will establish probabilistic guarantees of at least $1-n^{-(1+\Omega(1))}$ for the statements in Theorem 1, applying the union bound will show that the results indeed hold for arbitrary destinations d w.h.p.

Regarding the tightness of our result, assume all $\alpha \cdot n$ failed edges are so called *destination edges*, i.e., edges incident to the destination. Then, the flow starting at the other end v of such a (failed) edge will first be sent to a node selected uniformly at random from the set $V \setminus \{d\}$. The resulting distribution of the load can be seen as the outcome of throwing αn balls into $n-1$ bins [34], and the maximum load will reach $\Omega(\log n/\log\log n)$ at some node w.h.p.

A. Notation and Conventions

Let $\pi_{v,d}^{(i)}$ for $i \in \{1,2,3\}$ denote the permutations for some fixed node v when receiving a packet with destination d. We will in the following consider a fixed destination d, and therefore omit it from the indices. Additionally, let $\pi_v^{(i)}(j)$ for $1 \leq j \leq n-1$ denote the j-th node in v's permutation. Furthermore we define $\mathcal{F}$ to be the set of failed edges and further partition it into $\mathcal{F}^{(in)}$ and $\mathcal{F}^{(out)}$. The former contains all failed inner edges , i.e (u,v) with $u,v \neq d$. The latter consists of the remaining failed edges, which are incident to the destination. We furthermore define constants ε and γ such that $|\mathcal{F}^{(out)}| \leq \varepsilon \cdot n$ and $|\mathcal{F}^{(in)}| \leq \gamma \cdot n$, both smaller than α and subject to $\varepsilon+\gamma < 1$. Also, let V_G be the set of *good* nodes, that is, nodes $v \in V \setminus \{d\}$ with $(v,d) \notin \mathcal{F}^{(out)}$. Conversely let $V_B = V \setminus (V_G \cup \{d\})$ be the set of *bad* nodes, s.t. for $v \in V_B$ it holds that $(v,d) \in \mathcal{F}^{(out)}$. If we state that we apply Chernoff bounds for a random variable X, we mean $P[X \geq (1+\delta)\mu] \leq \exp(\min\{\delta, \delta^2\}\mu/3)$, where $\mu = E[X]$. For lower tails we use $P[X \leq (1-\delta)\mu] \leq \exp(\delta^2\mu/2)$ with $\delta < 1$. Besides w.h.p., we introduce the following abbreviations: Instead of *with probability*, *uniformly at random* and *random variable*, we will use w.p., u.a.r. and r.v. respectively. Finally, $\log n$ will denote $\log_2 n$.

B. Analysis

We first establish some structural properties that describe the paths that the packets take according to our failover strategy. For $i \in \{1,2,3\}$, we define the directed subgraphs $G^{'(i)} = (V \setminus \{d\}, E^{'(i)})$ with edge sets $E^{'(i)} = \{(v, \pi_v^{(i)}(1)) | v \in V_B\}$. Under assumption that $\mathcal{F}^{(in)} = \emptyset$, this graph depicts the possible paths a packet traverses to either the good nodes V_G or to some possibly existing cycle. It is easy to see that any graph $G^{'(i)}$ hosts multiple trees, each rooted in some $v \in V_G$. The other components in $G^{'(i)}$ consist of cycles, in which each node acts as the root of a tree. Note that such a root may have no children at all. The first important result of our analysis is that the size of these structures will be at most $O(\log n \cdot \log\log n)$ w.h.p.

In the next step, we will account for the failures in $\mathcal{F}^{(in)}$. Here we will use the fact the permutations are chosen completely at random. We will see that only $O(\log n)$ of all edges in the graphs $G^{'(i)}$ are failed, which reinforces the intuition that failing inner edges has little effect compared to the failure set $\mathcal{F}^{(out)}$. This approach will allow us to construct the graphs $G^{''(i)}$, which now account for inner edge failures and correctly depict the paths that the packets traverse.

Finally we put everything together and use the graphs $G^{''(i)}$ to show the result of Theorem 1. Additionally we will see that 3 permutations per node do indeed suffice for *any* packet to be routed to d w.h.p.

a) Measuring Forests: As mentioned we start by analyzing the graphs $G^{'(i)}$. We will for now consider some fixed $i \in \{1,2,3\}$ and omit the superscript (i). That is, we consider the graph G' together with the edge set $E' = \{(v, \pi_v(1)) | v \in V_B\}$. As already discussed, only the existence of some cycles between nodes in V_B prevents G' from being a forest. A rough perspective on G' is given in Fig. 2.

We start by establishing some structural properties of the graph G'. The proof of the following result relies on standard techniques and is omitted due to space constraints.

Lemma 1. *The graph G' does not contain paths or cycles of length larger than $4\log_{1/\varepsilon} n$ w.p. $1 - n^{-3}$. Additionally, the number of cycles in G' is $O(\log n)$ w.p. $1 - n^{-3}$.*

Consider again the graph $G' = (V', E')$ with $V' = V \setminus \{d\}$ and some fixed node $v \in V_G$. Remember that such a node is the root of a tree in G', induced by the edges $(w, \pi_w(1))$ for $w \in V_B$. Let now L_i denote the set of nodes at level i of v's tree, where $L_0 = \{v\}$. We now construct $L_1, L_2, ...$ step by step as follows. In the i-th step construct $L_i = \{w | w \in (V_B \setminus \bigcup_{j=0}^{i-1} L_j) \wedge \pi_w(1) \in L_{i-1}\}$. One can see this as constructing the tree $v \in V_G$ layer-by-layer, by adding nodes with outgoing edges connected to nodes in the set L_i in the i-th step. We define the r.v. $X_i = |L_i|$ and when fixing $v \in V_G$ we will in the following call $\{X_i\}$ the *layer sequence*, or in short *sequence*, corresponding to v. The step-wise construction described above directly yields the following lemma.

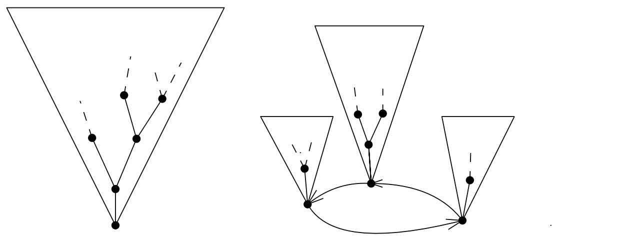
Fig. 2. The structures contained in the subgraph G'. On the left, a tree rooted in some $v \in V_G$ is presented. On the right, we have a cycle and each node of the cycle is again a root of a tree.

Lemma 2. *Fix some root node $v \in V_G$ in G' together with its layer sequence $\{X_i\}$. Then, it holds for level i that $X_i \sim B(m,p)$ with*

$$m = |V_B| - \sum_{j=1}^{i-1} X_j \text{ and } p = \frac{X_i}{|V'| - \sum_{j=0}^{i-2} X_j}$$

This means that we can describe the tree rooted in $v \in V_G$ by a sequence of binomial distributions, whose expected value depends on the previous layers. The following statement gives us a bound on $m \cdot p$ w.h.p., showing that the set of nodes at level i indeed decreases exponentially fast.

Lemma 3. *Let $v \in V_G$ be a root node in G' together with its corresponding layer sequence $\{X_i\}$. Then, for $i < \log^2 n$ it holds that $E[X_{i+1}] \leq X_i \cdot \varepsilon(1 + o(1)) \leq \beta$. Additionally, w.p. $1 - n^{-3}$ it holds for all $i \leq \log^2 n$ that $X_i < C \log n$. Here $0 < \beta < 1$ and $0 < C$ are constants depending on ε.*

Proof: We show by induction on i that with probability $p_j > (1 - i \cdot n^{-4})$ it holds for all $j \leq i$ that $X_j < C \log n$. Clearly w.p. 1 it holds that $X_0 = 1 < C \log n$. Now, consider $i \leftarrow (i+1)$ and observe that

$$\begin{aligned} &E[X_{i+1} | X_0, ..., X_i < C \log n] \\ &\quad < (|V_B| - \text{polylog}\, n) \frac{X_i}{|V'| - \text{polylog}\, n} \\ &\quad < \varepsilon \cdot X_i \cdot \left(1 + \frac{\text{polylog}\, n}{n}\right), \end{aligned} \tag{1}$$

where we used the induction hypothesis and that $i < \log^2 n$ as well as $|V_B| \leq \varepsilon n$. By Lemma 2, X_{i+1} follows a binomial distribution. Therefore we apply Chernoff bounds with $\delta \approx \varepsilon^{-2} - 1$ (c.f Section II-A) and obtain that $P[X_{i+1} \geq C \log n \mid X_1, ..., X_i < C \log n] < n^{-4}$ for large enough constant C. Hence we established the desired property w.p. $p_{i+1} \geq p_i \cdot (1 - n^{-4}) > (1 - (i+1)n^{-4})$ and conclude the induction. Using that $P[X_{i+1} \leq n] = 1$, for $i < \log^2 n$ we can bound $E[X_{i+1}] < E[X_{i+1} \mid X_0, ..., X_i < C \log n] + n \cdot i \cdot n^{-4}$, since $P[X_i < n] = 1$. This, together with (1), yields the first statement of the lemma. ■

According Lemma 1 packets at most $O(\log n)$ hops until reaching the destination. This implies the following.

Corollary 1. *Consider* any $v \in V_G$ *of* G' *with its corresponding layer sequence* $\{X_i\}$. *Then, for* $i > C' \log n$ *it holds that* $X_i = 0$ *w.p. at least* $1 - n^{-3}$.

Clearly for some fixed node $v \in V_G$ with sequence $\{X_i\}$, our main interest lies in the value $X = \sum_{i \geq 0} X_i$. In the following we say that $X_{i-1} < a$ *increases* into the interval $[a, b)$, iff $X_i \in [a, b)$. Analogously we say $X_{i-1} \geq b$ *decreases* into the same interval iff $X_i \in [a, b)$.

Lemma 4. *Consider again a root* $v \in V_G$ *and the corresponding layer sequence* $\{X_i\}$. *Then, at most* $O(\hat{\beta}^{-j})$ *members of* $\{X_i\}$ *will increase into the interval*

$$\left[C \log n \cdot \hat{\beta}^j, C \log n \cdot \hat{\beta}^{j-1}\right)$$

w.p. at least $1 - n^{-3}$. *Here* $j < \log(\log n / \log \log n)$, *and* $\hat{\beta}$ *is a constant* $\beta < \hat{\beta} < 1$, *with* β *and* C *being the constants defined in Lemma 3.*

Proof: In the following we consider the elements of the sequence $\{X_i\}$ one after the other, starting with X_0. By Lemma 2 and Lemma 3 we know that the i-th value X_i follows a binomial distribution with mean less than $X_{i-1} \cdot \beta$. Using Chernoff bounds together with the fact that $\beta < 1$ is a constant, we obtain for any $t \geq 0$

$$Pr[X_i \geq t \mid X_{i-1} \leq t] \leq \exp(-\Omega(t)). \tag{2}$$

Note that for $t = C \log n \cdot \hat{\beta}^j$, (2) bounds the probability that X_{i-1} increases into $[C \log n \cdot \hat{\beta}^j, C \log n \cdot \hat{\beta}^{j-1})$. Now, from Corollary 1 it follows that at most $C' \log n$ elements may increase into the interval mentioned before. Therefore we can majorize the total number of increases into the interval by $B(C' \log n, \exp(-c \cdot t))$, where c is the constant hidden in $\Omega(t)$ in (2). Now, using the well-known upper bound on the binomial coefficient we get

$$Pr\left[B\left(C' \log n, \exp\left(-cC \cdot \log n \cdot \hat{\beta}^j\right)\right) = \frac{5}{cC} \cdot \hat{\beta}^{-j}\right] < \tag{3}$$

$$\left(O(1) \log n \cdot \hat{\beta}^j\right)^{O(\hat{\beta}^{-j})} \left(\frac{1}{e}\right)^{5 \log n} < \frac{1}{n^{3.4}}$$

for n large enough and $j < \log(\log n / \log \log n)$. ■

We are finally ready to state that no tree contained in G' consists of more than $O(\log n \cdot \log \log n)$ nodes. The proof is omitted due to space constraints.

Lemma 5. *Let* $v \in V_G$ *be a root and* $\{X_i\}$ *the corresponding layer sequence. Then it holds w.p. at least* $1 - \text{polylog}\, n / n^3$ *that* $\sum_i X_i < O(\log n \cdot \log \log n)$

When applying the union bound, this gives us that *no* tree with root $v \in V_G$ of G' will exceed size $O(\log n \cdot \log \log n)$ w.h.p. However, remember that another type of component exists in G', namely cycles that may have additional nodes attached to them. Fix one of these components and let $C = \{v_1, ..., v_{|C|}\}$ be the set of nodes of the cycle. Furthermore define $A_i := \{w \mid w \in (V_B \setminus C) \wedge$ a path from w to v_i exists in $G'\} \cup \{v_i\}$. Then, this set induces a tree in G' unless the loop (v_i, v_i) is contained in E' which implies $|C| = 1$. In any case, one may see the component as being induced by the set $\bigcup_i A_i$ as a forest of trees, whose roots lie on the cycle C. To determine the total size of the set $\bigcup_{i=1}^{|C|} A_i$, we look at the growth of them layer-by-layer. However, this time we let all $|C|$ of these trees grow at the same time, again uncovering edges step-by-step. That is, $L_0 = C$ and construct any set L_i with $X_i = |L_i|$ just as we did when considering the roots $v \in V_G$, i.e., $L_i = \{w \mid w \in (V_B \setminus \bigcup_{j=0}^{i-1} L_j) \wedge \pi_w(1) \in L_{i-1}\}$. Observe that for this fixed cycle we can describe the sequence $\{X_i\}$ by the number of nodes in $V_B \setminus C$ being at distance i from the cycle. When replacing the set V_B by $V_B \setminus C$, the result of Lemma 2 also holds for this layer sequence. As Lemma 1 guarantees that $X_0 < O(\log n)$, the statement of Lemma 3 follows accordingly and allows us to repeat the whole approach.

Corollary 2. *The results of Lemma 3, Corollary 1, Lemma 4 and finally Lemma 5 also hold for the sequence* $\{X_i\}$ *of nodes in distance* i *to some fixed cycle in* G'.

b) Accounting for Inner Edge Failures: So far we established that none of the components in G' will be of total size more than $O(\log n \cdot \log \log n)$ w.h.p. Remember however that we still need to account for the failures in $\mathcal{F}^{(in)}$. We start by showing that only $O(\log n)$ of them lie on any path in G', even if the adversary fails $\Theta(n)$ inner edges. The proof is based on standard techniques and omitted here due to space constraints.

Lemma 6. *The number of nodes* $v \in V$ *that have their first failover edge* $(v, \pi_v(1))$ *destroyed by the adversary is* $O(\log n)$ *w.p.* $1 - n^{-3}$

In the following we will transfer $G^{'(i)}$ into $G^{''(i)}$ for any $i = 1, 2, 3$. The basic idea will be to remove edges $(v, \pi_v^{(i)}(1))$ that lie in $\mathcal{F}^{(in)}$ from $G^{'(i)}$ and replace them with the first non-failed edge in the permutation $\pi_v^{(i)}$. This way, the graph $G^{''(i)}$ represents the correct path of the packets with hop counter $(i-1)C_1 \leq h(p) < iC_1$. One may see the construction of $G^{''(i)}$ as cutting out subtrees of size $O(\log n \cdot \log \log n)$ of the structures in $G^{'(i)}$ and then letting them dock on some components in $G^{'(i)}$. This is visualized in Fig. 3. Besides the fact that some components in $G^{''(i)}$ are extended by docking subtrees, a new type of structure can be created. Some of the launched subtrees may connect with each other and thereby form a new type of cycle. It is easy to show the following.

Lemma 7. *Consider the graph* $G^{''(i)}$. *Then, none of the components contained in* $G^{''(i)}$ *will consist of more than* $O(\log n \cdot \log \log n)$ *nodes. Furthermore, the number of con-*

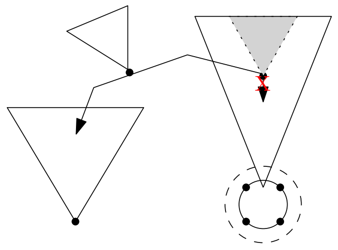

Fig. 3. Failed edges $(v, \pi_v(1))$ may cause subtrees relocate.

tained cycles remains in $O(\log n)$ with each not exceeding length $O(\log n)$. Additionally, any packet that is not trapped in some cycle takes at most $C_1 = 16 \log_{1/\varepsilon} n$ steps to reach d. All above statements hold w.p. $1 - O(n^{-2})$

c) From Forests to Load: To determine the total load some node v receives we look at the graphs $G^{''(i)}$ one after another. When starting the *all-to-one* routing, each node initiates a flow and all of them follow paths according to the single outgoing edges in $G^{''(1)}$. Now, most of the flows will reach the destination d after at most C_1 many hops. However, some might be trapped inside a cycle. In both cases we will consider the cost $T_1(v)$ that occurred at any node v until this point. Upon reaching a hop value of C_1 the flows currently trapped in a cycle will move according to the permutation $\pi_v^{(2)}$. If we know where these cycle nodes are located in $G^{''(2)}$, we can again track the flows' paths and determine the loads caused by the next C_1 many hops, denoted $T_2(v)$. Finally, we repeat this approach one more time in the same manner and obtain the values $T_3(v)$. In the following we say that flows *exit* the system $G^{''(i)}$ if their respective hop counter reaches iC_1. Similarly we say flows *enter* the system $G^{''(i)}$, if they reach hop value $(i-1)C_1$ and $G^{''(i)}$ becomes relevant for their failover paths for the first time. Finally, we will argue that $T_i(v) = 0$ w.h.p. for any v and $i \geq 4$, which is why we originally only required three permutations, and deduce that the total load v receives is $T(v) = T_1(v) + T_2(v) + T_3(v)$. We can show the following:

Lemma 8. *Let $i \geq 1$ and assume that every component in $G^{''(i)}$ is entered by $O(\log n \cdot \log \log n)$ total flows. Then, every node v that is not contained in a cycle will receive a load of at most $T_i(v) = O(\log n \cdot \log \log n)$. Nodes contained in a cycle in $G^{''(i)}$ will receive $T_i(v) = O(\log^2 n \cdot \log \log n)$ load until the flows exit $G^{''(i)}$ w.p. $1 - O(n^{-2})$*

Clearly, for $i = 1$ and $G^{''(1)}$ the assumption of this lemma holds. This is because Lemma 7 guarantees that all structures are of size $O(\log n \cdot \log \log n)$ and each node starts sending one flow of packets. The next lemma deals with the inductive step and $i > 1$. (Proof omitted due to space constraints.)

Lemma 9. *Assume that the assumption of Lemma 8 holds for $G^{''(i)}$. Then, also in $G^{''(i+1)}$ no component will be entered by more than $O(\log n \log \log n)$ flows w.p. $1 - O(n^{-2})$*

Above lemma concludes the inductive step, showing that Lemma 8 is applicable to all $i \in \{1, 2, 3\}$. We now state that it is indeed enough to only look at the graphs $G^{''(i)}$ where $1 \leq i \leq 3$ to determine the maximum load.

Lemma 10. *Fix an arbitrary packet p sent by a node $v \in V_B$. Then, p ends up in a cycle in each $G^{''(1)}, G^{''(2)}$ and $G^{''(3)}$ w.p. at most $\operatorname{polylog}/n^3$. Additionally, it holds that $T_i(v) = 0$ for all nodes $v \in V \setminus \{d\}$ and $i \geq 4$ w.p. $1 - \operatorname{polylog}/n^2$.*

Note that this lemma also implies that $O(\log n)$ hops suffice for any packet to reach the destination. We established in Lemma 9 that the assumption in Lemma 8 is indeed fulfilled for $G^{''(1)}, G^{''(2)}$ and $G^{''(3)}$. Hence, the total load any node receives that is not contained in a cycle in either $G^{''(2)}$ or $G^{''(1)}$ is $O(\log n \cdot \log \log n)$. And those that lie on a cycle will have load of $O(\log^2 n \cdot \log \log n)$, where according to Lemma 7 at most $O(\log^2 n)$ such nodes exist. Theorem 1 follows accordingly.

III. Circumventing Cycles by Partitioning

One of the major challenges of the *3-Permutations* protocol is to cope with temporary cycles, which may be introduced when employing randomization into the failover strategy. For packets with large hop counts, which indicate the existence of a cycle, we effectively provided different failover routes. In the following we present the *Intervals* routing protocol that 1) does not introduce any cycles in the packets routing paths w.h.p. and 2) is purely destination based. This comes however at the cost of smaller maximum resilience against failures.

The *Intervals* protocol works as follows. We assume that every node is given a unique ID or address, which is known to the other nodes. Therefore we can enumerate the nodes by $v_1, v_2, ..., v_n$. Let now α be some small constant $0 < \alpha < 1$. We partition the nodes of the graph into consecutive sets of size $I = n/4 \log_{1/\alpha} n$. That means, the i-th set R_i contains nodes with addresses in the range

$$\Big[(i-1) \cdot \frac{n}{4 \log_{1/\alpha} n} + 1 \ , \ i \cdot \frac{n}{4 \log_{1/\alpha} n}\Big].$$

Assume that the value α is chosen such that both, the interval bounds and the number of intervals, are integers. The next step is similar to Section II. Every node tries to directly forward a packet to the desired destination d, if the direct link is available. Otherwise, again a permutation $\pi_{v,d}$ of nodes is consulted and the packet is sent to the first reachable partner in $\pi_{v,d}$. The crucial difference to the *3-Permutations* protocol is the following: for some node v that lies in the interval R_i, the permutation $\pi_{v,d}$ is a permutation of the nodes in $R_{(i+1)} \setminus \{d\}$. Hence, only edges ranging from nodes in the set R_i to nodes in R_{i+1} are considered as possible failover edges. To allow for a proper protocol, we assume that nodes of the rightmost interval choose failover edges into R_0. We will see in the upcoming analysis that this protocol will not create any cycles in the routing paths w.h.p. The following statement allows the adversary to fail up to $\Omega(n/\log n)$ many edges. Note that this protocol is purely destination based and therefore any

```
Input: A packet with destination $d$
1: if $(v, d)$ is intact then forward $p$ to $d$ and return
2: Forward $p$ to first directly reachable node in $\pi_{v,d}$
```

Fig. 4. *Intervals* protocol. Point-of-view of some node v

deterministic scheme operating under this constraint would allow $\Omega(n/\log n)$ load to be created by the adversary [15].

Theorem 2. *Assume the adversary is allowed to fail up to $\alpha \cdot I$ many edges, for some arbitrary constant $0 < \alpha < 1$ where $I = n/(4 \log_{1/\alpha} n)$. Then, when considering all-to-one routing to any destination d, the* Intervals *protocol guarantees a maximum of*

$$O(\log n \cdot \log\log n)$$

load at any node except d and edge w.h.p. Additionally, no packet will perform more than $O(\log n)$ hops w.h.p.

For $\alpha = 1/e$ above statement provides the maximum resilience of $n/(4e \log_e n)$. Furthermore, assume the adversary fails $\Omega(n/\log n)$ destination edges (v, d), with all such nodes v being in the same interval R_i. Then, similar as in the case of the *3-Permutation* protocol, a balls-into-bins argument [34] shows that after all nodes in R_i send their flows to randomly chosen nodes in R_{i+1}, at least one node will have load $\Omega(\log n/\log\log n)$ w.h.p.

Concerning the notation we will carry over everything defined in Section II-A. Additionally we extend the notation for the sets of failed edges as $\mathcal{F}_i^{(in)}$ and $\mathcal{F}_i^{(out)}$. These sets only contain failed edges started in the i-the interval and, in case of $\mathcal{F}_i^{(in)}$, have partners in the interval $i + 1$. Just as in the protocols description we denote the set of nodes inside the i-th interval as R_i where $I = |R_i|$.

We remark that the result of Theorem 2 also holds, if we require that $|\mathcal{F}_i^{(in)}| < \varepsilon \cdot I$ and $|\mathcal{F}_i^{(out)}| < \gamma \cdot I$ *for any* interval R_i, as long as $\varepsilon + \gamma \leq 1 - \Delta$ for some constant $\Delta > 0$. As the sets $\mathcal{F}_i^{(in)}$ and $\mathcal{F}_i^{(out)}$ are specified for some fixed destination d, we require this property for all possible destinations d.

Regarding the memory complexity, for a fixed destination each node needs a permutation of $O(n/\log n)$ nodes, hence $O(n)$ bits in total. As in case of the *3-Permutations* protocol, the set of permutations $\{\pi_{v,d} \mid d \in V\}$ some node v requires can be derived from a single permutation π_v, and only the first $3 \log_{1/\alpha} n$ entries of each permutation need to be stored (see corresponding description on page 3). This allows for a reduction of the *total* memory required per node to $O(\log^2 n/\log(1/\alpha))$.

A. Analysis

A main motivation behind Section III is to eliminate the need to perform any kind of cycle resolution. To that end we will start by showing that our protocol will not introduce any cycles into the packets routing paths and at the same time derive that the hop count of *any* packet remains in $O(\log n)$.

To show that the maximum load occurring lies in $O(\log n \cdot \log\log n)$, we will take a similar approach as in Section II-B. That is, we fix some node $v \in V_G \cap R_i$ in some interval. Then, all the sources of packets that are forwarded over the edge (v, d) will form a tree rooted in v. We will again argue that, in expectation, the number of nodes per level of this tree decreases exponentially fast and reuse parts of Section II-B.

a) Cycles: Some packet located in R_i has only two possibilities for its next hop. Either it is directly forwarded to the destination, or it is forwarded to a node of the set R_{i+1}. To end up in a cycle it needs to traverse a sequence of $4 \log_{1/\alpha} n$ intervals, hitting a node $v \in V_B$ with every hop. This is unlikely and formalized as follows.

Lemma 11. *Let p be an arbitrary packet to be routed to destination d. Then, its routing path will not contain any cycles and it will reach the destination after $4 \cdot \log_{1/\alpha} n + 1$ hops w.p. at least $1 - n^{-4}$.*

The proof is simple and excluded due to space constraints.

b) Carrying over Previous Results: In the following we show that the result of Lemma 5 also holds when nodes follow the *Intervals* protocol. That is, for some fixed node $v \in V_G$ we construct a tree as follows. Assume $v \in R_i$ and let $L_0 = \{v\}$ denote the root of this tree. The j-th level of the tree associated with v is defined by $L_j = \{w \mid (w \in R_{i-j} \cap V_B) \wedge (\pi_w(k) \in L_{i-1}) \wedge (\forall \ell < k : (w, \pi_w(\ell)) \in \mathcal{F}^{(in)}) \wedge ((w, \pi_w(k)) \notin \mathcal{F}^{(in)})\}$. That is, L_j is the set of nodes whose packets reach v in exactly j hops (for easier readability we neglect the fact that wrap-around might occur). One can easily show the following.

Lemma 12. *Let $v \in V_G$ be arbitrary and assume that $v \in R_i$. Furthermore let $X_j = |L_j|$, where L_j is the j-th level of the tree associated with v. Then, it holds that $E[X_{j+1}] \leq X_j \cdot \alpha$.*

With this we established a statement similar to the first part of Lemma 3. It is easy to see that the size of each level X_j of the tree follows a binomial distribution when constructing the tree level-by-level. Furthermore, Lemma 11 establishes the property of Corollary 1 and since Lemma 12 guarantees that the levels shrink exponentially fast in expectation, the statement $X_i < C \log n$, C large enough, follows by applying Chernoff bounds. That said, we established all necessary requirements and a simple repetition of the corresponding analysis allows us to reuse Lemma 4 and Lemma 5. Summarizing, we get the following and conclude the proof of Theorem 2.

Corollary 3. *Let $v \in V_G$ be a good node and let $\{X_i\}$ be defined as in Lemma 12. Then it holds that $\sum_i X_i = O(\log n \cdot \log\log n)$ w.p. $1 - \operatorname{polylog} n/n^3$.*

IV. Further Reducing the Congestion

In this section we present a third protocol, called *Shared-Permutations*, that improves the bound of the maximum load observed in Theorem 1 and Theorem 2 under the assumption that the nodes share a common but randomized permutation. This could for example be achieved by computing parts of the routing tables starting from the same seed for the random generator, which is unknown to the adversary. Additionally we assume again that the packet headers are equipped with a hop

```
Input: A packet with destination $d$ and hop count $h(p)$
1: if $(v,d)$ is not failed then forward $p$ to $d$ ; $h(p)$ += 1 and return
2: if $h(p) < E_2$ then $v' \leftarrow$ successor of $v$ in $\pi_{h(p),d}$
3:     if $(v,v')$ is not failed then forward $p$ to $v'$
4:     else $h(p) \leftarrow E_2$.
5: if $h(p) \geq E_2$ then forward $p$ to first directly reachable node according to $\pi_{v,h(p),d}$
6: increase $h(p)$ += 1
```

Fig. 5. *Shared-Permutations* protocol. Point-of-view of some node v

field of size $O(\log\log n)$ bits, which is initially set to 0 and may be accessed by the network partners.

The *Shared-Permutations* protocol works as follows. Again we consider an arbitrary but fixed destination d. Every node $v \in V$ is equipped with permutations $\pi_{0,d}, \pi_{1,d}, ..., \pi_{C_1,d}$ of all nodes $V \setminus \{d\}$, where C_1 is a value $O(\log n)$ to be specified later. Now, contrary to the *3-Permutation* protocol, these permutations are assumed to be *globally* agreed upon without being known to the adversary. Furthermore, each permutation is chosen u.a.r out of the set of all possible permutations. Additionally we assume that v stores C_2 additional permutations $\pi_{v,j,d}$ on $V \setminus \{d\}$, only known to v itself and chosen u.a.r. Here $j \in \{E_2, E_2+1, \ldots, E_2+C_2+1\}$ for $E_2 = C_1 + 1$ and C_2 is another value in $O(\log n)$.

Assume now that a packet p with destination d arrives at node $v \in V$ and denote its current hop counter by $h(p)$. First of all, if the link (v,d) is not failed the packet is directly forwarded to the destination. Otherwise if $h(p) < E_2$, the node v forwards it via a link (v,v') where v' denotes the node following v in the global permutation $\pi_{h(p),d}$. In case this link is failed, v raises the hop counter of p to E_2 instead and forwards it to the *first non-failed* edge according to $\pi_{v,E_2,d}$. The case we did not consider yet is $h(p) \geq E_2$. In this case p is routed over the first reachable partner in $\pi_{v,h(p),d}$. Finally, in every case, $h(p)$ is increased by one. A pseudo-code describing this algorithm is given in Fig. 5. The common global permutations allow the flow to be distributed more evenly among the network, reducing the congestion to $O(\sqrt{\log n})$, even if $\Omega(n)$ edges are failed by the adversary.

Theorem 3. *Assume that the adversary is allowed to fail $\alpha \cdot n$ edges total, where $\alpha < 1$ is a constant. When performing all-to-one routing to any destination d, the* Shared-Permutations *protocol guarantees a maximum flow of*

$$O(\sqrt{\log n})$$

on any node (except d) and edge w.h.p. Additionally, no packet traverses more than $O(\log n)$ hops w.h.p.

Assuming it is possible for the nodes to agree on common permutations that are not known to the adversary, the maximum load can be decreased by more than a factor $\sqrt{\log n}$ compared to the protocols in Sections II and III. Note that this result breaks the $\Omega(\log n / \log\log n)$ lower bound of the *3-Permutations* and *Intervals* protocols.

Regarding space complexity, our nodes are required to store $O(\log n)$ permutations of n nodes per destination. Therefore in the most simple case we require $O(n^2 \cdot \log^2 n)$ bits at most. However, the same improvements as described in Sections II and III can be made to store the permutations more efficiently and achieve a memory complexity of $O(\log^3 n / \log(1/\alpha))$ bits per node. Note that the protocol requires knowledge of the values C_1 and C_2, which can both be set to $5\log_{1/\alpha} n$. These values are given in Lemma 14 and Lemma 17, together bounding the maximum number of hops any packet performs until it reaches the destination d w.h.p. If α is not known to the nodes, then a slow growing function in $\omega(\log n)$ can be used for C_1 and C_2, which comes at the cost of slightly increased memory complexity.

A. Analysis

Throughout the analysis we will consider a fixed destination d and omit the corresponding index from all permutations. We will use the notation from Section II-A and start by neglecting any failed edges in the set $\mathcal{F}^{(in)}$. Next, we assume that each node sends 1 packet with destination d from each node $v \in V$. We will consider the number of packets that have i hops while still not having reached the destination d and see that this set decreases exponentially fast. Additionally no packet traverses more than $C_1 < E_2$ many hops w.p. at least $1 - n^{-2}$. Therefore, without any inner edge failures, the only relevant permutations for our failover strategy are $\pi_0, ..., \pi_{E_2-1}$.

Finally we will account for the failures in $\mathcal{F}^{(in)}$ and make use of the permutations $\pi_{v,j}$. We will consider the maximum load caused by the flows after reaching hop value E_2 separately and deduce that this value is $O(\sqrt{\log n})$ w.h.p.

a) Staying in Line: We start by neglecting the failures in $\mathcal{F}^{(in)}$, i.e we assume first that $|\mathcal{F}^{(in)}| = 0$ and $|\mathcal{F}^{(out)}| < \varepsilon \cdot n$. Furthermore, assume that every node $v \in V \setminus \{d\}$ starts sending a single packet to destination d. Then, the number of packets that pass through some node v is equivalent to the number of flows passing through v. Notice, that due to the global permutations, no node will be visited by more than 1 packet with the same hop value. Consider the set of packets hop-for-hop and denote H_i as the set of packets that reached hop i at some point without reaching the destination. Clearly $|H_0| = (n-1)$, $|H_1| = |V_B|$ and we can show the following.

Lemma 13. *Let H_i denote the set of packets that still have not reached d after i hops. Then, for $|H_i| = \Omega(\log n)$ it holds w.p. at least $1 - n^{-4}$ that $|H_{i+1}| \leq |H_i| \cdot \sqrt{\varepsilon}$.*

This allows us to bound the maximum number of hops required for any packet to reach destination d. It is easy to show the following.

Lemma 14. *Fix some packet p with destination d. Then, assuming $\mathcal{F}^{(in)} = 0$, it requires at most C_1 steps to reach the destination w.p. at least $1 - n^{-3}$. Here C_1 is a value bounded above by $5\log_{1/\varepsilon} n$.*

Remember the following invariant: when fixing some node v and hop value i, the node v will receive *at most* 1 packet with such hop value. This leads to following result.

Lemma 15. *Consider some node $v \in V$ and assume $|\mathcal{F}^{(in)}| = 0$. Then, if every node sends 1 packet with destination d, v will by visited by packets $O(\sqrt{\log n})$ times w.p. at least $1 - n^{-3}$.*

Proof: Let i^* be the first time such that H_{i^*} reaches size $O(\log n)$. We start by showing that throughout hops $1 \leq i < i^*$ the node v is visited by at most $O(\sqrt{\log n})$ packets in total. For this range of i, we know according to Lemma 13 that the size of H_i will decrease exponentially fast, i.e $|H_{i+1}| < |H_i| \cdot \beta$ for some constant $0 < \beta < 1$ that depends on ε. Fix now some node v and let $Y_i = 1$ iff v receives a packet with hop value i at some point, and 0 otherwise. Similar as in the proof of Lemma 13, we argue that the packets with hop value i are distributed uniformly – and independently of any earlier hops – among the nodes of the network. Hence $P[Y_i = 1] = H_i/(n-1)$ and we are interested in $Y = \sum_{i=1}^{i^*} Y_i$. Note that $P[Y_i = 1] = H_i/n < \varepsilon \cdot \beta^i$, where we wrote n instead of $n-1$ for ease of readability. Then

$$P[Y = k] = \sum_{\substack{S \subseteq \{1, \dots O(\log n)\} \\ \text{with } |S| = k}} \Big(\prod_{j \in S} P[Y_j = 1] \cdot \prod_{\ell \in \{1 \dots O(\log n)\} \setminus S} 1 - P[Y_\ell = 1] \Big).$$

The second product can be crudely bounded above by 1. The first product reaches its maximum value for $S = \{1, ..., k\}$ and is bounded by $(\varepsilon\beta \cdot \varepsilon\beta^2 \cdot ... \cdot \varepsilon\beta^k)$. Therefore we get

$$P[Y = k] < \left(\frac{e \cdot C \log n}{k} \right)^k \varepsilon^k \cdot \beta^{k(k-1)/2}.$$

Now assume $k = C' \cdot \sqrt{\log n}$ for some sufficiently large constant C'. In this case

$$P[Y = k] < O\left(\sqrt{\log n}\right)^{C'\sqrt{\log n}} \cdot \left(\frac{1}{n}\right)^5.$$

Clearly the first term lies in $o(n)$. When increasing k further, the probability will only get smaller. Applying the union bound over the remaining $O(\log n) - C'\sqrt{\log n}$ larger values of k, we get $P[Y \geq C'\sqrt{\log n}] < n^{-3}$. Adding 1 for the packet that was initialized on v yields the result. ■

Clearly this implies that both, the maximum node load and the edge load are $O(\sqrt{\log n})$ in the case of $|\mathcal{F}^{(in)}| = 0$.

b) Accounting for Inner Edge Failures: We consider two copies, $S^{(out)}$ and S of our initial graph in which the adversary failed at most αn edges according to its strategy. In $S^{(out)}$ we repair all failures $\mathcal{F}^{(in)}$, which results in ignoring inner edge failures just as described above. Again we will consider the equivalent point-of-view of each node sending a single packet to d instead of a consecutive flow. The idea in the following is to consider only $S^{(out)}$ and each time an inner-edge (u, v) is chosen for communication that is failed in the original graph, the packet is copied and placed with hop count E_2 on u in S. This way S will only contain packets with hop count of at least E_2. The packet in $S^{(out)}$ will however continue as if the edge was intact. Note that, by Lemma 14, S will w.h.p. only consist of packets that are redirected because of inner edge failures. The idea behind the analysis is the following: Let $S^{(out)}$ run until all packets reached the destination d and determine the number of packets starting in S. We then let the system S run and it is easy to see that we can majorize the load some node v receives in the original process by adding up the loads of v in $S^{(out)}$ and S respectively. This is because in $S^{(out)}$ we do not remove packets but copy them to S instead.

In Lemma 15 we already established the load some node v receives in $S^{(out)}$. The following can be shown w.r.t. the number of packets that are initialized in the system S.

Lemma 16. *Consider the number of packets p that reach a load of E_2 at some point. Then, at most $O(\log n \cdot \sqrt{n})$ of them will exist w.p. at least $1 - 2 \cdot n^{-3}$.*

As all packets in S have hop count at least E_2, only the *local* permutations $\pi_{v,j}$ are used as part of our failover strategy. While this leads to nodes possibly receiving multiple packets of the same hop value, the number of initial packets lies in $O(\log n\sqrt{n})$ only. The following statement can be shown.

Lemma 17.
1) *Fix some arbitrary packet in S. Then, it will reach the destination after at most C_2 hops for $C_2 < 3\log_{1/\alpha} n$ w.p. at least $1 - O(n^{-3})$.*
2) *Each node in S is reached by at most $O(\sqrt{\log n})$ packets in total and w.p. at least $1 - O(n^{-3})$.*

We established that in both systems, $S^{(in)}$ and S, each node has a load of $O(\sqrt{\log n})$. Theorem 3 follows accordingly.

V. Further Remarks

Stronger adversaries. So far we have focused on oblivious adversaries who only know the random distribution but not the actually sampled values (a common assumption in the literature). However, our algorithms can easily be extended to deal also with more adaptive adversaries: to defeat adversaries who aim to infer network-internal loads (e.g., leveraging physical access or using tomographic techniques), we can simply regenerate random permutations periodically. That is, the *3-Permutations* and *Intervals* algorithms have the attractive property that they allow to regenerate such permutations quickly, locally, and without coordination: each node can independently regenerate random numbers over time to enhance security. Note, this also allows our algorithms to recover in case the $1/n^{\Omega(1)}$ probability event occurs that may lead to higher loads than specified in the theorems.

Lower Amount of Failures. Throughout the previous sections we assumed that the adversary destroys up to either linear or $O(n/\log n)$ many edges. A lower amount of edge failures affects the performance of the algorithms as follows. In case $n^{1-\delta}$ edges are destroyed for constant $\delta > 0$ it can be shown (by a slightly adapted repetition of the existing analysis) that all three of our algorithms guarantee w.h.p. a maximum load of $O(1)$ on most, or even all, of the nodes. That is, the Intervals and Shared-Permutations protocols achieve this result w.r.t. all

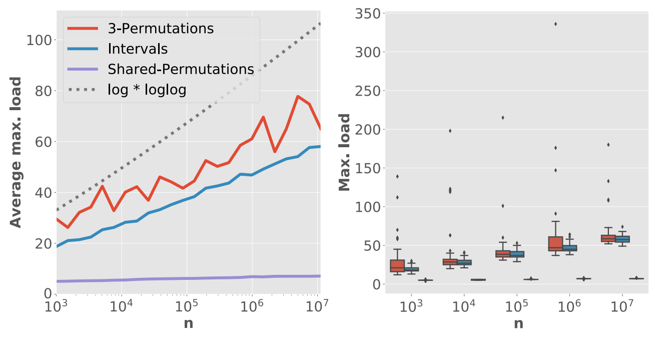

Fig. 6. *Left:* Log-plot of the average maximum load for networks of increasing size. *Right:* Maximum load distribution for selected n values. 50 runs for each value of n (colors correspond to same algorithms as in left figure).

nodes in $V \setminus \{d\}$. In case of the *3-Permutations* algorithm $O(\log n)$ nodes may receive a load of up to $O(\log n)$.

Random Edge Failures. Previously we assumed the existence of a malicious adversary selecting the edges to be failed. However, in the context of e.g. hardware failures or power outages, it might make sense to model the set of failed edges as selected u.a.r. out of all $\sim n^2/2$ edges. In this case, w.h.p., all but $O(\log n)$ nodes will forward their incoming traffic directly to the all-to-one routing destination d. Then, it can be shown that all our algorithms only induce a congestion of $O(1)$ w.h.p.

VI. Simulations

To complement our theoretical results and obtain insights into the specific distributions of the maximum load, we conducted several simulations.

a) Setup: We study fully meshed networks of different sizes. We preceded each simulation run by selecting a distinguished node $d \in V$ as destination and failing $\alpha \cdot n$ edges with $\alpha = 1/2$. We chose to distribute all these failures on edges incident to the destination node d. Intuitively, failing destination edges is the best strategy for the adversary as most so-called inner edges (i.e., edges not incident to d) are not part of any failover route. Note that in the analysis of the *3-Permutations* protocol (cf. Lemma 6), we showed that the algorithm tries to route over at most $O(\log n)$ of such failed inner edges w.h.p., even if there are $n/2$ failed inner eges. Additionally, we selected the set of failed destination edges uniformly at random: as the random bits generated by the nodes are unknown to the adversary and we operate in a complete network, there is no better strategy for the adversary. To initiate a flow from each node to the destination d, each node $v \in V \setminus \{d\}$ generates at the beginning a packet to be sent to destination d (all-to-one routing towards d) and incoming packets are forwarded according to one of our three proposed algorithms. To account for the maximum flow passing through some node, the maximum load is defined as the maximum number of packets that traverse any node $v \in V \setminus \{d\}$ during the whole execution of the algorithm.

b) Parameters: We considered 24 complete networks with sizes ranging from 10^3 to 10^7 nodes. For every such network, 50 simulations were performed w.r.t. each protocol. In the simulated *3-Permutations* and *Shared-Permutations* algorithms, we set $C_1 = \lceil 1.5 \log n \rceil$ as this value worked well in practice. As only edges incident to the destination were failed, the C_2 parameter of the *Shared-Permutations* protocol does not need to be specified. In case of the *Intervals* protocol, we failed half of the destination edges in each interval, allowing for comparison to our other algorithms, and set the interval size to $\lceil 4n/\log_{(1/\alpha)} n \rceil$ as required by Theorem 2.

c) Results: The log-plot on the left of Fig. 6 shows the averaged maximum load for increasing network sizes. We observe that the average maximum load w.r.t. all algorithms for all network sizes lies below $\log n \log \log n$. This reflects the theoretical results we derived for our *Intervals* and *Shared-Permutations* protocols, which ensure a maximum congestion of $O(\log n \cdot \log \log n)$ and $O(\sqrt{\log n})$, respectively (cf. Theorems 2 and 3). Surprisingly, the *3-Permutations* protocol achieves similar average maximum load as the *Intervals* protocol, which is significantly smaller than the theoretical upper bound derived in Theorem 1. This can be explained as follows: In many simulation runs of the *3-Permutations* protocol *no* temporary cycles occur at all. However, as discussed in Section II-B Lemma 8, a load of $\omega(\log n \log \log n)$ is only induced on nodes that lie on a (temporary) cycle (see cycle structure in Fig. 2). This phenomena is further analyzed in the second plot on the right side of Fig. 6. It contains 5 triples of box plots, where each such triple describes the distribution of the maximum load w.r.t. the three protocols over 50 runs each. While the results of the *Intervals* and *Shared-Permutations* simulations are tightly concentrated, the *3-Permutations* protocol sometimes induces a congestion above value $\log n \log \log n$. These outliers correspond to runs in which the failover paths of some packets contain temporary cycles of short length (at most 3 nodes). Packets following a failover path that reaches a node on such a cycle may hit each of these cycle nodes up to $\Omega(\log n)$ times before exiting the cycle (after roughly C_1 hops). In case the failover paths of $\omega(\log \log n)$ packets share the same such cycle, each node lying on this cycle will receive a load of $\omega(\log n \cdot \log \log n)$. However, also these outliers seem to adhere the theoretical bound of $O(\log^2 n \cdot \log \log n)$ as specified by Theorem 1.

VII. Conclusion

We initiated the study of randomized congestion-minimizing fast rerouting algorithms and presented three algorithms which provably ensure significantly lower loads than any deterministic algorithm. We understand our work as a first step and believe that it opens several interesting avenues for future research. In particular, it would be interesting to fill the gap between our derived upper and lower bounds which do not perfectly match yet. More generally, it will be interesting to extend our study of randomized approaches to more general communication patterns and network structures.

Acknowledgments. We would like to thank the anonymous reviewers and our shepherd K.K. Ramakrishnan for the helpful comments that allowed us to improve the paper.

References

[1] A. Atlas and A. Zinin, "Basic specification for IP fast reroute: Loop-free alternates," in *Request for Comments (RFC) 5286*, 2008.

[2] P. Pan, G. Swallow, and A. Atlas, "Fast reroute extensions to RSVP-TE for LSP tunnels," in *Request for Comments (RFC) 4090*, 2005.

[3] T. Holterbach, E. C. Molero, M. Apostolaki, A. Dainotti, S. Vissicchio, and L. Vanbever, "Blink: Fast connectivity recovery entirely in the data plane," in *Proc. 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2019.

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

[5] ISO, "Intermediate ststem-to-intermediate system (is-is) routing protocol," ISO/IEC 10589, 2002.

[6] J. Moy, "OSPF version 2," RFC editor, https://tools.ietf.org/html/rfc2328, RFC 2328, 1998.

[7] Switch Specification 1.3.1, "OpenFlow," in *https://bit.ly/2VjOO77*, 2013.

[8] L. Shen, X. Yang, and B. Ramamurthy, "Shared risk link group (SRLG)-diverse path provisioning under hybrid service level agreements in wavelength-routed optical mesh networks," *IEEE/ACM Transactions on Networking*, vol. 13, no. 4, pp. 918–931, 2005.

[9] J. Tapolcai, B. Vass, Z. Heszberger, J. Bıró, D. Hay, F. A. Kuipers, and L. Rónyai, "A tractable stochastic model of correlated link failures caused by disasters," in *Proc. IEEE INFOCOM*, 2018.

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

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

[12] A. K. Atlas and A. Zinin, "Basic specification for IP fast-reroute: loop-free alternates," *IETF RFC 5286*, 2008.

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

[14] J. Feigenbaum, B. Godfrey, A. Panda, M. Schapira, S. Shenker, and A. Singla, "Brief announcement: On the resilience of routing tables," in *Proc. ACM PODC*, 2012.

[15] 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.

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

[17] M. Chiesa, I. Nikolaevskiy, S. Mitrovic, A. Panda, A. Gurtov, A. Madry, M. Schapira, and S. Shenker, "The quest for resilient (static) forwarding tables," in *Proc. IEEE INFOCOM*, 2016.

[18] K.-T. Foerster, Y.-A. Pignolet, S. Schmid, and G. Tredan, "CASA: congestion and stretch aware static fast rerouting," in *Proc. IEEE INFOCOM*, 2019.

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

[20] G. Iannaccone, C.-n. Chuah, R. Mortier, S. Bhattacharyya, and C. Diot, "Analysis of link failures in an IP backbone," in *Proc. ACM SIGCOMM Workshop on Internet Measurment*, 2002.

[21] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C. N. Chuah, Y. Ganjali, and C. Diot, "Characterization of failures in an operational IP backbone network," *IEEE/ACM Transactions on Networking*, vol. 16, no. 4, pp. 749–762, Aug 2008.

[22] A. J. González and B. E. Helvik, "Analysis of failures characteristics in the UNINETT IP backbone network," in *2011 IEEE Workshops of International Conference on Advanced Information Networking and Applications*, 2011.

[23] B. Yang, J. Liu, S. Shenker, J. Li, and K. Zheng, "Keep Forwarding: Towards k-link failure resilient routing," in *Proc. IEEE INFOCOM*, April 2014, pp. 1617–1625.

[24] E. Gafni and D. Bertsekas, "Distributed algorithms for generating loop-free routes in networks with frequently changing topology," *Trans. Commun.*, vol. 29, no. 1, pp. 11–18, 1981.

[25] M. S. Corson and A. Ephremides, "A distributed routing algorithm for mobile wireless networks," *Wireless networks*, vol. 1, no. 1, pp. 61–81, 1995.

[26] V. D. Park and M. S. Corson, "A highly adaptive distributed routing algorithm for mobile wireless networks," in *Proc. IEEE INFOCOM*, 1997.

[27] J. L. Welch and J. E. Walter, "Link reversal algorithms," *Synthesis Lectures on Distributed Computing Theory*, vol. 2, no. 3, pp. 1–103, 2011.

[28] 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.

[29] M. Chiesa, A. V. Gurtov, A. Madry, S. Mitrovic, I. Nikolaevskiy, M. Schapira, and S. Shenker, "On the resiliency of randomized routing against multiple edge failures," in *Proc. ICALP*, 2016.

[30] M. Chiesa, I. Nikolaevskiy, S. Mitrovic, A. Gurtov, A. Madry, M. Schapira, and S. Shenker, "On the resiliency of static forwarding tables," *IEEE/ACM Transactions on Networking (TON)*, vol. 25, no. 2, pp. 1133–1146, 2017.

[31] K.-T. Foerster, A. Kamisinski, Y.-A. Pignolet, S. Schmid, and G. Tredan, "Bonsai: Efficient fast failover routing using small arborescences," in *Proc. 49th IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2019.

[32] K.-T. Foerster, Y.-A. Pignolet, S. Schmid, and G. Tredan, "Casa: Congestion and stretch aware static fast rerouting," in *Proc. IEE INFOCOM*, 2019.

[33] A. Bhalgat, R. Hariharan, T. Kavitha, and D. Panigrahi, "Fast edge splitting and Edmonds' arborescence construction for unweighted graphs," in *Proc. SODA*, 2008.

[34] M. Raab and A. Steger, "Balls into bins – A simple and tight analysis," in *Randomization and Approximation Techniques in Computer Science*. Springer Berlin Heidelberg, 1998, pp. 159–170.