Robust Routing Made Easy

Christoph Lenzen and Moti Medina(✉)

MPII, Saarland Informatics Campus, Saarbrücken, Germany
{clenzen,mmedina}@mpi-inf.mpg.de

Abstract. Designing routing schemes is a multidimensional and complex task that depends on the objective function, the computational model (centralized vs. distributed), and the amount of uncertainty (online vs. offline). We showcase *simple* and generic transformations that can be used as a blackbox to increase resilience against (independently distributed) faults. Given a network and a routing scheme, we determine a *reinforced* network and corresponding routing scheme that faithfully preserves the specification and behavior of the original scheme. We show that reasonably small constant overheads in terms of size of the new network compared to the old one are sufficient for substantially relaxing the reliability requirements on individual components. The main message in this paper is that the task of designing a robust routing scheme can be decoupled into (i) designing a routing scheme that meets the specification in a fault-free environment, (ii) ensuring that nodes correspond to fault-containment regions, i.e., fail (approximately) independently, and (iii) applying our transformation to obtain a reinforced network and a robust routing scheme that is fault-tolerant.

1 Introduction

When scaling up the size of systems, one inevitably faces the challenge of sufficiently enhancing reliability to ensure intended operation. Specifically, this applies to the communication infrastructure, which must remain operational despite failures of some components. Otherwise, isolated faults would bring down the entire system, which is impractical unless the failure probability of individual components is so small that it is likely that none of them fail. Existing designs and algorithms (that are considered practical) do account for lost messages and, in some cases, permanently crash-failing nodes or edges [4,9,12].

It is our understanding that handling stronger fault types is considered practically infeasible, be it in terms of complexity of implementations or the involved overheads. However, pretending that crash failures are the worst that can happen means that the *entire* system possibly fails whenever, e.g., we face a "babbling idiot" (i.e., a node erroneously generating many messages and congesting the network), excessive link delays (violating specification), or misrouting, corruption,

The full version of this extended abstract can be found in https://arxiv.org/abs/1705.04042.

P. Spirakis and P. Tsigas (Eds.): SSS 2017, LNCS 10616, pp. 187–202, 2017.
https://doi.org/10.1007/978-3-319-69084-1_13

or loss of messages. The current approach is to (i) use techniques like error correction, acknowledging reception, etc. to mask the effects of such faults, (ii) hope to detect and deactivate faulty components quickly (logically mapping faults to crashes), and (iii) repair or replace the faulty components after they have been taken offline. This strategy may result in significant disruption of applications; possible consequences include:

(I) Severe delays in execution, as successful message delivery necessitates to detect and deactivate faulty components first. (II) Failure to deliver correct messages and the resulting repeated attempts to do so (both by applications or routing algorithms) overload the network; the resulting congestion then renders the system inoperative as a whole. (III) Constraints on message delivery times are violated, breaking any real-time service. (IV) More generally, any instance of the classic fallacy of assuming that the network is reliable [16] may cause secondary errors.

In this paper, we challenge the belief that resilience to strong fault types is intractable in practice. We discuss generic approaches to reinforcing networks at small constant overheads (in terms of resources like nodes, links, latency, and energy) to achieve resilience to non-crash faults (up to fully Byzantine, i.e., arbitrary behavior). The proposed strategies are deliberately extremely simple, both in terms of applying them and analyzing them. Yet, they substantially reduce the required reliability on the component level to maintain network functionality, without losing messages or increasing latencies. We provide transformations that allow for directly reusing non-fault-tolerant routing schemes as a blackbox, avoiding the need to refactor working solutions. The main message we seek to convey is that being prepared for non-benign faults can be simple, affordable, and practical, and therefore enables building larger reliable networks.

The Challenge. We are given a synchronous network $G = (V, E)$ and a routing scheme. We seek to allocate additional resources (nodes, edges) to the network and provide a corresponding routing strategy to simulate the routing scheme on the original network despite non-benign node failures. The goals are to (i) use little additional resources, (ii) maximize the probability of uniformly independently random node failures the network is likely to withstand, (iii) ensure that the transformation is simple to implement, and (iv) interferes as little as possible with the existing system design and operation, e.g., does not change the reinforced system's specification. Note that both (iii) and (iv) are crucial for practical utility; significant refactoring of existing systems and/or accommodating substantial design constraints is rarely affordable.

This setting makes a number of simplifying assumptions. First and probably most notably, we assume independent failures. This is motivated by the fact that highly correlated faults necessitate high degrees of redundancy and thus overheads; clearly, a system-wide power outage, whether rare or not, cannot be addressed by adding extra nodes or edges that are connected to the same power source, but requires independent backup power. More generally, guaranteeing full functionality despite having f adversarially placed faults trivially requires node degrees larger than f. As there are many reasons why topologies of communication networks feature very small degrees in practice, assuming worst-case *distribution* of faults would hence come at too high of a cost. Instead, we aim at masking faults with little or no correlation among each other, arguing that resilience to such faults can be boosted significantly. Second, in this context we treat nodes and their outgoing links as fault-containment regions (according to [10]), i.e., they are the basic components our systems are comprised of. This choice is made for the sake of concreteness; similar results could be obtained when considering, e.g., edge failures, without changing the gist of results or techniques. With these considerations in mind, the probability of uniformly random node failures that the reinforced system can tolerate is a canonical choice for measuring resilience. Third, we focus on synchronous networks. This has several reasons: we believe synchrony helps in handling faults, both on the theoretical level (as illustrated by the famous FLP theorem [8]) and for ensuring correct implementation; it simplifies presentation, making it easier to focus on the proposed concepts; last but not least, we believe our approach to be of particular interest in the context of real-time systems, where the requirement of meeting hard deadlines makes synchrony an especially attractive choice.

Techniques and Results. Our first approach is almost trivial: We replace each node by $\ell \in \mathbb{N}$ copies and for each edge we connect each pair of copies of its endpoints, where ℓ is a constant.[1] Whenever a message would be sent over an edge in the original graph, it should be sent over each copy of the edge in the reinforced graph. If not too many copies of a given node fail, this enables each receiving copy to recover the correct message. Thus, each non-faulty copy of a node can run the routing algorithm as if it were the original node, guaranteeing that it has the same view of the system state as its original in the corresponding fault-free execution of the routing scheme on the original graph.

We observe that, asymptotically almost surely (a.a.s., with probability $1 - o(1)$) and with $\ell = 2f + 1$, this reinforcement can sustain an independent probability p of Byzantine node failures for any $p \in o(n^{-1/(f+1)})$. This threshold is sharp up to (small) constant factors: for $p \in \omega(n^{-1/(f+1)})$, a.a.s. there is some node for which all of its copies fail. If we restrict the fault model to omission faults (faulty nodes may skip sending some messages), $\ell = f+1$ suffices. The cost of this reinforcement is that the number of nodes and edges increase by factors of ℓ and ℓ^2, respectively. Therefore, already this simplistic solution can support non-crash faults of probability $p \in o(1/\sqrt{n})$ at a factor-4 overhead. Note that the simulation introduces no big computational overhead and does not change the way the system works, enabling to use it as a blackbox. Randomized algorithms can be simulated as well, provided that all copies of a node have access to a shared source of randomness; note that this requirement is much weaker than globally shared randomness: it makes sense to place the copies of a node in physical proximity to approximately preserve the geometrical layout of the physical realization of the network topology.

[1] Choosing concreteness over generality, we focus on the, in our view, most interesting case of constant ℓ. It is straightforward to generalize the analysis.

We then proceed to reducing the involved overhead further. To this end, we apply the above strategy only to a small subset E' of the edge set. Denoting by $v_1, \ldots, v_\ell$ the copies of node $v \in V$, for any remaining edge $\{v, w\} \in E \setminus E'$ we add only edges $\{v_i, w_i\}$, $i \in [\ell$, to the reinforced graph. The idea is to choose E' in a way such that the connected components induced by $E \setminus E'$ are of constant size. This results in the same asymptotic threshold for p, while the number of edges of the reinforced graph drops to $((1-\varepsilon)\ell + \varepsilon\ell^2)|E|$. For constant ε, we give constructions with this property for grids or tori of constant dimension and minor-free graphs of bounded degree. Again, we consider the case of $f = 1$ of particular interest: in many typical network topologies, we can reinforce the network to boost the failure probability that can be tolerated from $\Theta(1/n)$ to $\Omega(1/\sqrt{n})$ by roughly doubling (omission faults) or tripling (Byzantine faults) the number of nodes and edges.

The redundancy in this second construction is near-optimal under the constraint that we want to simulate an arbitrary routing scheme in a blackbox fashion, as it entails that we need a surviving copy of each edge, and thus in particular each node. While one may argue that the paid price is steep, in many cases it will be smaller than the price for making each individual component sufficiently reliable to avoid this overhead. Furthermore, we briefly argue that the simplicity of our constructions enables us to re-purpose the redundant resources in applications with less strict reliability requirements.

We conclude by suggesting open problems we consider of interest for further developing the proposed paradigm of reinforcement against non-benign faults.

Related Work. Local Byzantine faults were studied in [5,13] in the context of broadcast and consensus problems. Unlike its global classical counterpart, the f-local Byzantine adversary can control at most f neighbors of each vertex. This more restricted adversary gives rise to more scalable solutions, as the problems can be solved in networks of degree $O(f)$; without this restriction, degrees need to be proportional to the *total* number of faults in the network.

We also limit our adversary in its selection of Byzantine nodes, by requiring that the faulty nodes are chosen independently at random. As illustrated, e.g., by Lemma 1 and Theorem 1, there is a close connection between the two settings. Informally, we show that certain values of p correspond, asymptotically almost surely (a.a.s), to an f-local Byzantine adversary. However, we diverge from the approach in [5,13] in that we require a fully time-preserving simulation of a fault-free routing schedule, as opposed to solving the routing task in the reinforced network from scratch.

2 High-Level Overview

In this section, we highlight the utility of decoupling the task of designing a valid reinforcement from the task of designing a routing scheme over the input network: one can just plug in any routing scheme, for any objective, e.g., load minimization, maximizing the throughput, etc., in various models of computation, e.g., centralized or distributed, randomized or deterministic, online or offline, or oblivious. We now sketch the guarantees and (mild) preconditions of our blackbox transformation informally (for formal specification see Sect. 3).

Assumptions on the Input Network. We have two main assumptions on the network at hand: (1) We consider synchronous routing networks, and (2) each node in the network (alongside its outgoing links) is a fault-containment region, i.e., it fails independently from other nodes.

Valid Reinforcement Simulation Guarantees. Our reinforcements make a number of copies of each node. We have each non-faulty copy of a node run the routing algorithm as if it were the original node, guaranteeing that it has the same view of the system state as its original in the corresponding fault-free execution of the routing scheme on the original graph. Moreover, the simulation fully preserves all guarantees of the schedule, including its timing, and introduces no big computational overhead.

Unaffected Complexity and Cost Measures. When designing a routing scheme, one optimizes its complexity, e.g., in terms of running time for centralized algorithms, number of rounds for distributed algorithms, message size, etc. This is balanced against its quality with respect to the objective function of the problem at hand, e.g., load minimization, maximizing the throughput, minimizing the latency, etc. Moreover, there is the degree of uncertainty that can be sustained, e.g., whether the input to the algorithm is fully available at the beginning of the computation (offline computation) or revealed over time (online computation). Our reinforcements preserve all of these properties, as they operate in a blackbox fashion. For example, our machinery readily yields various fault-tolerant packet routing algorithms in the Synchronous Store-and-Forward model by Aiello et al. [1]. More specifically, from [6] we obtain a centralized deterministic online algorithm on unidirectional grids of constant dimension that achieves a competitive ratio which is polylogarithmic in the number of nodes of the input network w.r.t. throughput maximization. Using [7] instead, we get a centralized randomized offline algorithm on the unidirectional line with constant approximation ratio w.r.t. throughput maximization. In the case that deadlines need to be met the approximation ratio is, roughly, $O(\log^* n)$ [15]. As a final example, one can obtain from [3] various online distributed algorithms with sublinear competitive ratios w.r.t. throughput maximization.

Cost and Gains of the Reinforcement. The price of adding fault-tolerance is given by the increase in the network size, i.e., the number of nodes and edges of the reinforced network in comparison to the original one. Due to the assumed independence of node failures, it is straightforward to see that the (uniform) probability of sustainable node faults increases roughly like $n^{-1/(f+1)}$ in return for (i) a linear-in-f increase in the number of nodes and (ii) an increase in the number of edges that is quadratic in f. We then proceed to improve the construction for grids and minor-free constant-degree graphs to reduce the increase in the number of edges to linear in f. Based on this information, one can then assess the effort in terms of these additional resources that is beneficial, as less reliable nodes in turn are cheaper to build, maintain, and operate. We also note that, due to the ability of the reinforced network to ensure ongoing unrestricted operability in the presence of some faulty nodes, faulty nodes can be replaced or repaired *before* communication is impaired or breaks down.

Preprocessing. Preprocessing is used, e.g., in computing routing tables in Oblivious Routing [14]. The reinforcement simply uses the output of such a preprocessing stage in the same manner as the original algorithm. In other words, the preprocessing is done on the input network and its output determines the input routing scheme. In particular, the preprocessing may be randomized and does not need to be modified in any way.

Randomization. Randomized routing algorithms can be simulated as well, provided that all copies of a node have access to a shared source of randomness. We remark that, as our scheme locally duplicates the network topology, it is natural to preserve the physical realization of the network topology in the sense that all (non-faulty) copies of a node are placed in physical proximity. This implies that this constraint is much easier to satisfy than globally shared randomness.

3 Preliminaries

We consider synchronous routing networks. Formally, the network is modeled as a directed graph $G = (V, E)$, where V is the set of $n \triangleq |V|$ vertices, and E is the set of $m \triangleq |E|$ edges (or links). Each node maintains a state, based on which it decides in each round for each of its outgoing links which message to transmit. We are not concerned with the inner workings of the node, i.e., how the state is updated; rather, we assume that we are given a scheduling algorithm performing the task of updating this state and use it in our blackbox transformations. In particular, we allow for online, distributed, and randomized algorithms.

Probability-p Byzantine Faults $\mathsf{Byz}(p)$. The set of faulty nodes $F \subseteq V$ is determined by sampling each $v \in V$ into F with independent probability p. Nodes in F may deviate from the protocol in arbitrary ways, including delaying, dropping, or forging messages, etc.

Probability-p Omission Faults $\mathsf{Om}(p)$. The set of faulty nodes $F \subseteq V$ is determined by sampling each $v \in V$ into F with independent probability p. Nodes in F may deviate from the protocol by not sending a message over an outgoing link when they should. We note that it is sufficient for this fault model to be satisfied *logically.* That is, as long as a correct node can identify incorrect messages, it may simply drop them, resulting in the same behavior of the system at all correct nodes as if the message was never sent.

Simulations and Reinforcement. For a given network $G = (V, E)$ and a scheduling algorithm A, we will seek to *reinforce* (G, A) by constructing $G' = (V', E')$ and scheduling algorithm A' such that the original algorithm A is *simulated* by A' on G', where G' is subject to random node failures. We now formalize these notions. First, we require that there is a surjective mapping $P : V' \to V$; fix G' and P, and choose $F' \subseteq V'$ randomly as specified above.

Definition 1 (Simulation under Byz*(p)*). *Assume that in each round $r \in \mathbb{N}$, each $v' \in V' \setminus F'$ is given the same input by the environment as $P(v')$. A' is a* simulation *of A under Byz(p), if for each $v \in V$, a strict majority of the nodes $v' \in V'$ with $P(v') = v$ computes in each round $r \in \mathbb{N}$ the state of v in A in this round. The simulation is* strong, *if not only for each $v \in V$ there is a strict majority doing so, but all $v' \in V' \setminus F'$ compute the state of $P(v')$ in each round.*

Definition 2 (Simulation under Om*(p)*). *Assume that in each round $r \in \mathbb{N}$, each $v' \in V'$ is given the same input by the environment as $P(v')$. A' is a* simulation *of A under Om(p), if for each $v \in V$, there is $v' \in V'$ with $P(v') = v$ that computes in each round $r \in \mathbb{N}$ the state of v in A in this round. The simulation is* strong, *if each $v' \in V'$ computes the state of $P(v')$ in each round.*

Definition 3 (Reinforcement). *A* (strong) reinforcement *of a graph $G = (V, E)$ is a graph $G' = (V', E')$, a surjective mapping $P\colon V' \to V$, and a way of determining a scheduling algorithm A' for G' out of scheduling algorithm A for G. The reinforcement is* valid *under the given fault model (Byz(p) or Om(p)) if A' is a (strong) simulation of A a.a.s.*

Resources and Performance Measures. We use the following performance measures. (i) The probability p of independent node failures that can be sustained a.a.s. (ii) The ratio $\nu \triangleq |V'|/|V|$, i.e., the relative increase in the number of nodes. (iii) The ratio $\eta \triangleq |E'|/|E|$, i.e., the relative increase in the number of edges.

4 Strong Reinforcement Under Byz(p)

Given are the input network $G = (V, E)$ and scheduling algorithm A. Fix a parameter $f \in \mathbb{N}$ and set $\ell = 2f + 1$.

Reinforced Network G'. We set $V' \triangleq V \times [\ell]$, where $[\ell] \triangleq \{1, \ldots, \ell\}$, and denote $v_i \triangleq (v, i)$. Accordingly, $P(v_i) \triangleq v$. We define $E' \triangleq \{(v', w') \in V' \times V' \mid (P(v'), P(w')) \in E\}$.

Strong Simulation A' of A. Consider node $v' \in V' \setminus F'$. We want to maintain the invariant that in each round, each such node has a copy of the state of $v = P(v')$ in A. To this end, v'

(1) initializes local copies of all state variables of v as in A,
(2) sends on each link $(v', w') \in E'$ in each round the message v would send on $(P(v'), P(w'))$ when executing A, and
(3) for each neighbor w of $P(v')$ and each round r, updates the local copy of the state of A as if v received the message that has been sent to v' by at least $f+1$ of the nodes w' with $P(w') = w$ (each one using edge (w', v')).

Naturally, the last step requires such a majority to exist; otherwise, the simulation fails. We show that A' can be executed and simulates A provided that for each $v \in V$, no more than f of its copies are in F'.

Lemma 1. *If for each $v \in V$, $|\{v_i \in F'\}| \leq f$, then A' strongly simulates A.*

Proof. We show the claim by induction on the round number $r \in \mathbb{N}$, where we consider the initialization to anchor the induction at $r = 0$. For the step from r to $r+1$, observe that because all $v' \in V' \setminus F'$ have a copy of the state of $P(v')$ at the end of round r by the induction hypothesis, each of them can correctly determine the message $P(v')$ would send over link $(v, w) \in E$ in round $r+1$ and send it over each $(v', w') \in E$ with $P(w') = w$. Accordingly, each $v' \in V' \setminus F'$ receives the message A would send over $(w, v) \in E$ from each $w' \in V' \setminus F'$ with $P(w') = w$ (via the link (w', v')). By the assumption of the lemma, we have at least $\ell - f = f + 1$ such nodes, implying that v' updates the local copy of the state of A as if it received the same messages as when executing A in round $r+1$. Thus, the induction step succeeds and the proof is complete.

Resilience of the Reinforcement. We now examine how large the probability p can be for the precondition of Lemma 1 to be satisfied a.a.s.

Theorem 1. *Assume that $p \in o(1)$. Then the above construction is a valid strong reinforcement for the fault model $\mathsf{Byz}(p)$ if $p \in o(n^{-1/(f+1)})$. Moreover, if G contains $\Omega(n)$ nodes with non-zero outdegree, $p \in \omega(n^{-1/(f+1)})$ implies that the reinforcement is not valid.*

Proof. By Lemma 1, A' strongly simulates A if for each $v \in V$, $|\{v_i \in F'\}| \leq f$. If $p \in o(n^{-1/(f+1)}) \cap o(1)$, using $\ell = 2f+1$ and a union bound we see that the probability of this event is at least

$$1 - n \sum_{j=f+1}^{2f+1} \binom{2f+1}{j} p^j (1-p)^{2f+1-j} \geq 1 - n \sum_{j=f+1}^{2f+1} \binom{2f+1}{j} p^j$$

$$\geq 1 - n \binom{2f+1}{f+1} p^{f+1} \sum_{j=0}^{f} p^j \in 1 - n(2e)^f p^{f+1}(1+o(1)) = 1 - o(1).$$

Here, the second last step uses that $\binom{a}{b} \leq (ae/b)^b$ and that $p \in o(1)$, while the last step exploits that $p \in o(n^{-1/(f+1)})$.

On the other hand, for any $v \in V$, the probability that $|\{v_i \in F'\}| > f$ is independent of the same event for other nodes and larger than $\binom{2f+1}{f+1} p^{f+1}(1-$

$p)^f \geq (3/2)^f p^{f+1}(1-p)^f \in \Omega((3/2)^f p^{f+1})$, since $\binom{a}{b} \geq (a/b)^b$. Hence, if G contains $\Omega(n)$ nodes v with non-zero outdegree, $p \in \omega(n^{-1/(f+1)}) \cap o(1)$ implies that the probability that there is some node v with $|\{v_i \in F'\}| > f$ is in $1 - \left(1 - \Omega\left(\left(\frac{3}{2}\right)^f p^{f+1}\right)\right)^{\Omega(n)} \subseteq 1 - \left(1 - \omega\left(\frac{1}{n}\right) \cap o(1)\right)^{\Omega(n)} = 1 - o(1)$. If there is such a node v, there are algorithms A and inputs so that A sends a message across some edge (v, w) in some round. If faulty nodes do not send messages in this round, the nodes $w_i \in V' \setminus F'$ do not receive the correct message from more than f nodes v_i and the simulation fails. Hence, the reinforcement cannot be valid.

Remark 1. *For constant p, one can determine suitable values of $f \in \Theta(\log n)$ using Chernoff's bound. However, as our focus is on small (constant) overhead factors, we refrain from presenting the calculation here.*

Efficiency of the Reinforcement. For $f \in \mathbb{N}$, we have that $\nu = \ell = 2f + 1$ and $\eta = \ell^2 = 4f^2 + 4f + 1$, while we can sustain $p \in o(n^{-1/(f+1)})$. In the special case of $f = 1$, we improve from $p \in o(1/n)$ for the original network to $p \in o(1/\sqrt{n})$ by tripling the number of nodes. However, $\eta = 9$, i.e., while the number of edges also increases only by a constant, it seems too large in systems where the limiting factor is the amount of links that can be afforded.

5 Strong Reinforcement Under $\mathsf{Om}(p)$

The strong reinforcement from the previous section is, trivially, also a strong reinforcement under $\mathsf{Om}(p)$. However, we can reduce the number of copies per node for the weaker fault model. Given are the input network $G = (V, E)$ and scheduling algorithm A. Fix a parameter $f \in \mathbb{N}$ and, this time, set $\ell = f + 1$.

For details of the reinforcement, the simulation of algorithm A, and the corresponding proofs, we refer the reader to the full version. The resilience statement and the efficiency of the reinforcement are as follows.

Theorem 2. *There is a valid strong reinforcement for the fault model $\mathsf{Om}(p)$ if $p \in o(n^{-1/(f+1)})$. If G contains $\Omega(n)$ nodes with non-zero outdegree, then $p \in \omega(n^{-1/(f+1)})$ implies that the reinforcement is not valid.*

Efficiency of the Reinforcement. For $f \in \mathbb{N}$, we have that $\nu = \ell = f + 1$ and $\eta = \ell^2 = f^2 + 2f + 1$, while we can sustain $p \in o(n^{-1/(f+1)})$. In the special case of $f = 1$, we improve from $p \in o(1/n)$ for the original network to $p \in o(1/\sqrt{n})$ by doubling the number of nodes and quadrupling the number of edges.

6 More Efficient Reinforcement

In this section, we reduce the overhead in terms of edges at the expense of obtaining only a (non-strong) reinforcement. We stress that the obtained trade-off between redundancy (ν and η) and the sustainable probability of faults p is asymptotically optimal: as we require to preserve arbitrary routing schemes in a blackbox fashion, we need sufficient redundancy on the link level to directly simulate communication. From this observation, both for $\mathsf{Om}(p)$ and $\mathsf{Byz}(p)$ we can readily derive trivial lower bounds on redundancy that match the constructions below up to lower-order terms.

6.1 A Toy Example

Before we give the construction, we give some intuition on how we can reduce the number of required edges. Consider the following simple case. G is a single path of n vertices $(v_1, \dots, v_n)$, and the schedule requires that in round i, a message is sent from v_i to v_{i+1}. We would like to use a "budget" of only n additional vertices and an additional $(1+\varepsilon)m = (1+\varepsilon)(n-1)$ links, assuming the fault model $\mathsf{Om}(p)$. One approach is to duplicate the path and extend the routing scheme accordingly. We already used our entire budget apart from εm links! This reinforcement is valid as long as one of the paths succeeds in delivering the message all the way. The probability that one of the paths "survives" is $1-(1-(1-p)^n)^2 \leq 1-(1-e^{-pn})^2 \leq e^{-2pn}$, where we used that $1-x \leq e^{-x}$ for any $x \in \mathbb{R}$. Hence, for any $p = \omega(1/n)$, the survival probability is $o(1)$. In contrast, the strong reinforcement with $\ell = 2$ (i.e., $f = 1$) given in Sect. 5 sustains any $p \in o(1/\sqrt{n})$ with probability $1-o(1)$; however, while it adds n nodes only, it requires $3m$ additional edges. We need to add some additional edges to avoid that the likelihood of the message reaching its destination drops too quickly. To this end, we use the remaining εm edges to "cross" between the two paths every $h \triangleq 2/\varepsilon$ hops (assume h is an integer). This splits the path into segments of h nodes each. As long as, for each such segment, in one of its copies all nodes survive, the message is delivered. For a given segment, this occurs with probability $1-(1-(1-p)^h)^2 \geq 1-(ph)^2$. Overall, the message is thus delivered with probability at least $(1-(ph)^2)^{n/h} \geq 1-nhp^2$. As for any constant ε, h is a constant, this means that the message is delivered a.a.s. granted that $p \in o(1/\sqrt{n})$!

Remark 2. *The reader is cautioned to not conclude from this example that random sampling of edges will be sufficient for our purposes in more involved graphs. Since we want to handle arbitrary routing schemes, we have no control over the number of utilized routing paths. As the latter is exponential in n, the probability that a fixed path is not "broken" by F would have to be exponentially small in n. Moreover, trying to leverage Lovász Local Lemma for a deterministic result runs into the problem that there is no (reasonable) bound on the number of routing paths that pass through a single node, i.e., the relevant random variables (i.e., whether a path "survives") exhibit lots of dependencies.*

6.2 Partitioning the Graph

To apply the above strategy to other graphs, we must take into account that there can be multiple intertwined routing paths. However, the key point in the above example was not that we had path segments, but rather that we partitioned the nodes into constant-size regions and used a few edges inside these regions only, while fully connecting the copies of nodes at the boundary of the regions.

In general, it is not possible to partition the nodes into constant-sized subsets such that only a very small fraction of the edges connects different subsets; any graph with good expansion is a counter-example. Fortunately, many network topologies used in practice are not expanders. We focus in this section on grid networks and minor free graphs and show how to apply the above strategy in each of these families of graphs.

Grid Networks. We can generalize the above strategy to hypercubes of dimension $d > 1$.

Definition 4 (Hypercube Networks). *A q-ary d-dimensional hypercube has node set $[q]^d$ and two nodes are adjacent if they agree on all but one index $i \in [d]$, for which $|v_i - w_i| = 1$.*

The proof of the following lemma is in the full version.

Lemma 2. *For any $h, d \in \mathbb{N}$, assume that h divides $q \in \mathbb{N}$ and set $\varepsilon = 1/h$. Then the q-ary d-dimensional hypercube can be partitioned into $(q/h)^d$ regions of h^d nodes such that at most an ε-fraction of the edges connects nodes from different regions.*

Note that the above result and proof extend to tori, which also include the "wrap-around" edges connecting the first and last nodes in any given dimension.

Minor Free Graphs. Another general class of graphs that can be partitioned in a similar fashion are minor-free bounded-degree graph.

Definition 5 (H-Minor free Graphs). *For a fixed graph H, H is a minor of G if H is isomorphic to a graph that can be obtained by zero or more edge contractions on a subgraph of G. We say that a graph G is H-minor free if H is not a minor of G.*

For any such graph, we can apply a Corollary from [11, Corollary 2] which is based on [2] to construct a suitable partition.

Theorem 3 [11]**.** *Let H be a fixed graph. There is a constant $c(H) > 1$ such that for every $\varepsilon \in (0, 1]$, every H-minor free graph $G = (V, E)$ with degree bounded by Δ a partition $R_1, \ldots, R_k \subseteq V$ with the following properties can be found in time $O(|V|^{3/2})$: (i) $\forall i : |R_i| \leq \frac{c(H)\Delta^2}{\varepsilon^2}$, (ii) $\forall i$ the subgraph induced by R_i in G is connected. (iii) $|\{(u, v) \mid u \in R_i, v \in R_j, i \neq j\}| \leq \varepsilon \cdot |V|$.*

Remark 3. *Grids and tori of dimension $d > 2$ are not minor-free.*

We note that this construction is not satisfactory, as it involves large constants. It demonstrates that a large class of graphs is amenable to the suggested approach, but it is advisable to search for optimized constructions for more specialized graph families before applying the scheme.

Reinforced Network G'. Equipped with a suitable partition of $G = (V, E)$ into disjoint regions $R_1, \ldots, R_k \subseteq V$, we reinforce as follows. As before, we set $V' \triangleq V \times [\ell]$, denote $v_i \triangleq (v, i)$, define $P(v_i) \triangleq v$, and set $\ell \triangleq f + 1$. However, the edge set of G' differs. For $e = (v, w) \in E$,

$$E'_e \triangleq \begin{cases} \{(v_i, w_i) \mid i \in [\ell]\} & \text{if } \exists k' \in [k] : v, w \in R_{k'} \\ \{(v_i, w_j) \mid i, j \in [\ell]\} & \text{else.} \end{cases}$$

and we set $E' \triangleq \bigcup_{e \in E} E'_e$.

6.3 Simulation Under $\mathsf{Om}(p)$

The details of how to reinforce the network and to simulate algorithm A on this reinforced network as well as the corresponding proofs appear in the full version. The resilience statement and the efficiency of the reinforcement are as follows.

Resilience of the Reinforcement. Denote $R \triangleq \max_{k' \in [k]}\{|R_{k'}|\}$ and $r \triangleq \min_{k' \in [k]}\{|R_{k'}|\}$.

Theorem 4. *There is a valid reinforcement for the fault model $\mathsf{Om}(p)$ if $p \in o((n/r)^{-1/(f+1)}/R)$. Moreover, if G contains $\Omega(n)$ nodes with non-zero outdegree and $R \in O(1)$, $p \in \omega(n^{-1/(f+1)})$ implies that the reinforcement is not valid.*

Efficiency of the Reinforcement. For $f \in \mathbb{N}$, we have that $\nu = \ell = f + 1$ and $\eta = (1 - \varepsilon)\ell + \varepsilon\ell^2 = 1 + (1 + \varepsilon)f + \varepsilon f^2$, while we can sustain $p \in o(n^{-1/(f+1)})$. In the special case of $f = 1$ and, say, $\varepsilon = 1/5$, we improve from $p \in o(1/n)$ for the original network to $p \in o(1/\sqrt{n})$ by doubling the number of nodes and multiplying the number of edges by 2.4.

Remark 4. *For hypercubes and tori, the asymptotic notation for p does not hide huge constants. Lemma 2 shows that h enters the threshold in Theorem 4 as $h^{-d+1/2}$; as the cases of $d = 2$ and $d = 3$ are the most typical (for $d > 3$ grids and tori suffer from large distortion when embedding them into 3-dimensional space), the threshold on p degrades by factors of 11.2 and 55.9, respectively.*

6.4 Simulation Under $\mathsf{Byz}(p)$

The same strategy can be applied for the stronger fault model $\mathsf{Byz}(p)$, if we switch back to having $\ell = 2f + 1$ copies and nodes accepting the majority message among all messages from copies of a neighbor in the original graph.

Consider node $v \in V$. We want to maintain the invariant that in each round, a majority among the nodes v_i, $i \in [\ell]$, has a copy of the state of v in A. For $v' \in V'$ and $(w, P(v')) \in E$, set $N_{v'}(w) \triangleq \{w' \in V' \mid (w', v') \in E'\}$. With this notation, v' behaves as follows.

(1) It initializes local copies of all state variables of v as in A.
(2) It sends in each round on each link $(v', w') \in E'$ the message v would send on $(P(v'), P(w'))$ when executing A (if v' cannot compute this correctly, it may send an arbitrary message).
(3) It updates its state in round r as if it received, for each $(w, P(v')) \in E$, the message the majority of nodes in $N_{v'}(w)$ sent.

The proof of the following lemma is in the full version.

Lemma 3. *Suppose for each $k' \in [k]$, there are at least $f+1$ indices $i \in [\ell]$ so that $\{v_i \,|\, v \in R_{k'}\} \cap F' = \emptyset$. Then A' simulates A.*

Resilience of the Reinforcement. Denote $R \triangleq \max_{k' \in [k]}\{|R_{k'}|\}$ and $r \triangleq \min_{k' \in [k]}\{|R_{k'}|\}$.

Theorem 5. *Assume that $Rp \in o(1)$. The above construction is a valid reinforcement for the fault model $\mathsf{Byz}(p)$ if $p \in o((n/r)^{-1/(f+1)}/R)$. Moreover, if G contains $\Omega(n)$ nodes with non-zero outdegree and $R \in O(1)$, $p \in \omega(n^{-1/(f+1)})$ implies that the reinforcement is not valid.*

Proof. By Lemma 3, A' simulates A if for each $k' \in [k]$, there are at least $f+1$ indices $i \in [\ell]$ so that $\{v_i \,|\, v \in R_{k'}\} \cap F' = \emptyset$. For fixed k' and $i \in [\ell]$, $\Pr\left[\{v_i \,|\, v \in R_{k'}\} \cap F' = \emptyset\right] = (1-p)^{|R_{k'}|} \geq 1 - Rp$. Thus, analogous to the proof of Theorem 1, the probability that for a given k' the condition is violated is at most $\sum_{j=f+1}^{2f+1} \binom{2f+1}{j} (Rp)^j (1-Rp)^{2f+1-j} \in (2e)^f (Rp)^{f+1}(1+o(1))$. By a union bound over the at most n/r regions, we see that $p \in o((n/r)^{-1/(f+1)}/R)$ thus guarantees that the simulation succeeds a.a.s. As $r \leq R \in O(1)$, the proof of the second statement is analogous to the respective statement of Theorem 1.

Efficiency of the Reinforcement. For $f \in \mathbb{N}$, we have that $\nu = \ell = 2f+1$ and $\eta = (1-\varepsilon)\ell + \varepsilon\ell^2 = 1 + (2+2\varepsilon)f + 4\varepsilon f^2$, while we can sustain $p \in o(n^{-1/(f+1)})$. In the special case of $f=1$ and $\varepsilon = 1/5$, we improve from $p \in o(1/n)$ for the original network to $p \in o(1/\sqrt{n})$ by tripling the number of nodes and multiplying the number of edges by 4.2.

7 Discussion

In the previous sections, we have established that constant-factor redundancy can significantly increase reliability of the communication network in a blackbox fashion. Our constructions in Sect. 6 are close to optimal. Thus, one may argue that the costs are too high. However, apart from pointing out that the costs of using sufficiently reliable components may be even higher, we would like to raise a number of additional points in favor of the approach.

Node Redundancy. When building a reliable large-scale system, fault-tolerance needs to be considered on all system levels. Unless nodes are sufficiently reliable, node replication is mandatory, regardless of the communication network. In other words, the node redundancy required by our construction may not be an actual overhead to begin with. When taking this point of view, the salient question becomes whether the increase in links is acceptable. Here, the first observation is that any system employing node redundancy will need to handle the arising additional communication, incurring the respective burden on the communication network. Apart from still having to handle the additional traffic, however, the system designer now needs to make sure that the network is sufficiently reliable for the node redundancy to matter. Our simple schemes then provide a means to provide the necessary communication infrastructure without risking to introduce, e.g., a single point of failure during the design of the communication network; at the same time, the design process is simplified and modularized.

Dynamic Faults. Due to the introduced fault-tolerance, faulty components do not impede the system as a whole, so long as the simulation of the routing scheme can still be carried out. Hence, one may repair faulty nodes at runtime. If T is the time for detecting and fixing a fault, we can discretize time in units of T and denote by p_T the (assumed to be independent) probability that a node is faulty in a given time slot, which can be bounded by twice the probability to fail within T time. Then the failure probabilities we computed in our analysis directly translate to an upper bound on the expected fraction of time during which the system is not (fully) operational.

Adaptivity. The employed node- and link-level redundancy may be required for mission-critical applications only, or the system may run into capacity issues. In this case, we can exploit that the reinforced network has a very simple structure, making various adaptive strategies straightforward to implement. (i) One may use a subnetwork only, deactivating the remaining nodes and links, such that a reinforced network for smaller f (or a copy of the original network, if $f = 0$) remains. This saves energy. (ii) One may subdivide the network into several smaller reinforced networks, each of which can perform different tasks. (iii) One may leverage the redundant links to increase the overall bandwidth between (copies of) nodes, at the expense of reliability. (iv) The above operations can be applied locally; e.g., in a congested region of the network, the link redundancy could be used for additional bandwidth. Note that if only a small part of the network is congested, the overall system reliability will not deteriorate significantly.

8 Conclusion

In this work we analyze simple replication strategies for improving network reliability. While our basic schemes may hardly surprise, to the best of our knowledge the literature does not provide the kind of discussion given here. This, in turn, surprised us: simplicity is an important design feature, and we tried to convey the message that a number of significant advantages in overall system design arise from the proposed approach. In addition, we highlight that a (still simple) refined strategy results in near-optimal trade-offs under the constraint that arbitrary routing schemes are fully preserved. We consider this property highly useful in general and essential in real-time systems. Weaker guarantees may result in more efficient solutions, but also necessitate that other system levels must be able to handle the consequences.

Our work raises a number of follow-up questions. (i) Which network topologies allow for good partitions as utilized in Sect. 6? Small constants here result in highly efficient reinforcement schemes, which is key to practical solutions. (ii) Is it possible to guarantee strong simulations at smaller overheads? (iii) Can constructions akin to the one given in Sect. 6 be applied to a larger class of graphs?

References

1. Aiello, W., Kushilevitz, E., Ostrovsky, R., Rosén, A.: Dynamic routing on networks with fixed-size buffers. In: SODA, pp. 771–780 (2003)
2. Alon, N., Seymour, P., Thomas, R.: A separator theorem for graphs with an excluded minor and its applications. In: STOC, pp. 293–299. ACM (1990)
3. Angelov, S., Khanna, S., Kunal, K.: The network as a storage device: dynamic routing with bounded buffers. Algorithmica **55**(1), 71–94 (2009)
4. Cho, H., Leem, L., Mitra, S.: ERSA: error resilient system architecture for probabilistic applications. Trans. Comput.-Aided Des. Integr. Circ. Syst. **31**(4), 546–558 (2012)
5. Dolev, D., Hoch, E.N.: Constant-space localized byzantine consensus. In: Taubenfeld, G. (ed.) DISC 2008. LNCS, vol. 5218, pp. 167–181. Springer, Heidelberg (2008). doi:10.1007/978-3-540-87779-0_12
6. Even, G., Medina, M., Patt-Shamir, B.: Better deterministic online packet routing on grids. In: SPAA, pp. 284–293 (2015)
7. Even, G., Medina, M., Rosén, A.: A constant approximation algorithm for scheduling packets on line networks. In: ESA, pp. 40:1–40:16 (2016)
8. Fischer, M., Lynch, N., Paterson, N.: Impossibility of distributed consensus with one faulty process. J. ACM **32**(2), 374–382 (1985)
9. Kang, Y.H., Kwon, T., Draper, J.: Fault-tolerant flow control in on-chip networks. In: NOCS, pp. 79–86 (2010)
10. Kopetz, H.: Fault containment and error detection in the time-triggered architecture. In: ISADS, pp. 139–146 (2003)
11. Levi, R., Ron, D.: A quasi-polynomial time partition oracle for graphs with an excluded minor. ACM Trans. Algorithms **11**(3), 24:1–24:13 (2015)
12. Park, D., Nicopoulos, C., Kim, J., Vijaykrishnan, N., Das, C.R.: Exploring fault-tolerant network-on-chip architectures. In: DSN, pp. 93–104 (2006)
13. Pelc, A., Peleg, D.: Broadcasting with locally bounded byzantine faults. Inf. Process. Lett. **93**(3), 109–115 (2005)
14. Räcke, H.: Survey on oblivious routing strategies. In: Ambos-Spies, K., Löwe, B., Merkle, W. (eds.) CiE 2009. LNCS, vol. 5635, pp. 419–429. Springer, Heidelberg (2009). doi:10.1007/978-3-642-03073-4_43

15. Räcke, H., Rosén, A.: Approximation algorithms for time-constrained scheduling on line networks. Theory Comput. Syst. **49**(4), 834–856 (2011)
16. Rotem-Gal-Oz, A.: Fallacies of Distributed Computing Explained. http://www.rgoarchitects.com/Files/fallacies.pdf