Attack-Tolerance in Structured Networks via Multipath Routing

EDWARD L. PLATT, University of Michigan
DANIEL M. ROMERO, University of Michigan

Networks with single points of failure are particularly susceptible to adversarial faults, in which an attacker creates faults strategically at central points. Centralized network architectures necessarily exhibit this vulnerability, leading to a growing popularity in decentralized architectures. However, decentralized architectures do not guarantee the absence of single points of failure, and networks such as the Internet and World-Wide Web have been shown theoretically and historically to be highly susceptible to adversarial faults. We develop a partial trust model which makes weaker transitivity assumptions than common web of trust models. Using this model, we propose a general multipath fault tolerance technique for structured networks which provides asymptotic security against adversarial faults. We also provide a specific algorithm for implementing this technique on the butterfly network topology. When network topology can be dictated, these results can be used to create scalable, attack-tolerant infrastructures. More generally, our results provide a formalism for evaluating the effects of network structure on adversarial fault tolerance.

CCS Concepts: •**Security and privacy**→ **Formal methods and theory of security;** •**Networks**→ **Network architectures;** •**Software and its engineering**→ **Peer-to-peer architectures;**

Additional Key Words and Phrases: censorship, decentralization, fault tolerance, multipath, networks, peer-to-peer, trust

1. INTRODUCTION

Large-scale communication networks, exemplified by the Internet, have become ubiquitous and critical infrastructural for communities, organizations, and markets. As with any critical infrastructure, the cost of a failure can be immense, so methods for tolerating various kind of faults are an important and ongoing area of research [Albert et al. 2000; Sterbenz et al. 2010; Zin et al. 2015]. The key question is: what are the techniques and network structures that can be used to create robust, fault-tolerant systems?

Many complex networks, including the Internet and World-Wide Web, exhibit scale-free structure [Barabsi and Albert 1999; Barabsi and others 2009]. While scale-free networks tolerate random faults well, they are highly susceptible to adversarial faults, i.e. attacks [Albert et al. 2000]. For example, in 2008, YouTube suffered a worldwide outage for several hours when a service provider in Pakistan advertised false routing information [Hunter 2008]. The action (known as a *black hole attack*) was intended to censor YouTube within Pakistan only, but resulted in a worldwide cascading failure. Such vulnerabilities are not limited to any one system or protocol, but an attribute of complex communication networks themselves. General techniques for understanding and mitigating these vulnerabilities are needed.

We consider a setting in which a source node in a network attempts to route information to a target node by forwarding it through the links of the network. We assume that some nodes in the network may be compromised by an attacker and will forward the wrong information to their neighbors or block the information altogether. In order

Author's addresses: E.L. Platt and D.M. Romero, School of Information; email: {elplatt,drom}@umich.edu.

EC'16, July 24–28, 2016, Maastricht, The Netherlands. ACM 978-1-4503-3936-0/16/07.
http://dx.doi.org/10.1145/XXXXXXX.XXXXXXX

to evaluate network-based techniques for mitigating such attacks, we develop a decentralized *partial trust* model, similar to a web of trust [Ferguson and Schneier 2003; Zimmermann 1995], but making weaker assumptions of trust transitivity. While privacy and secrecy are often concerns in secure communication, it is not the focus of this paper. Existing techniques for privacy and secrecy are relatively mature compared to those for tolerating blocked or corrupted messages. In fact, an attacker might use message disruption as a fallback when surveillance isn't feasible. However, we would like to stress that the technqiues presented in this paper are entirely compatible with, and in some cases could enhance, existing privacy and secrecy techniques.

Given the above framework, we develop and evaluate a *multipath routing* technique to reduce the likelihood that compromised nodes are undetected during the information routing process by allowing multiple copies of a message to be sent through independent paths, which can be used for error-detection and error-correction by the receiver.

The multipath routing technique relies on the existence and discoverability of multiple redundant paths between the source and the destination. Thus, network structure and routing algorithms play a key role in the success of the technique. We formally evaluate the effects of network structure on attack-resistance and show that the probability of undetected errors decreases exponentially with the number of independent paths between source and destination.

The mutlipath routing technique is well-suted for *structured networks*, in which links between nodes are constrained to a particular architecture. Specifically, we employ the butterfly network topology, used elsewhere in parallel computing [Kshemkalyani and Singhal 2008] and peer-to-peer [Korzun and Gurtov 2013; Lua et al. 2005] applications. For this network topology, we propose an efficient, attack-tolerant multipath routing algorithm.

Our main contributions are:

— We develop a decentralized *partial trust model*, which makes weaker transitivity assumptions than previous web-of-trust models, and use this model to quantify the effects of network structure on fault tolerance;
— We show that as the number of disjoint paths between the neighborhoods of the sender and reciever grows, the probability of successfully detecting adversarial faults approaches 1 exponentially. Furthermore, we show that the number of disjoint paths is a function of network structure;
— We present a scalable, efficient, and attack-tolerant multipath routing algorithm on the butterfly network topology.

This paper is organized as follows. Section 2. reviews background and related work. Section 3. discusses the desired properties of attack-tolerant network infrastructure. Section 4. proposes the partial trust model. Section 5. proposes a general technique for multipath fault tolerance on structured networks. Section 6. proposes a specific application of this technique to the butterfly network topology. Section 7. discusses these results. And Section 8. concludes.

2. BACKGROUND AND RELATED WORK

In *structured networks*, link structure is predetermined. Such networks can be designed to have favorable structural and routing properties, at the expense of complicating the addition or removal of nodes. Structured networks have been a popular tool in parallel and distributed computing architectures [Kshemkalyani and Singhal 2008]. More recently, peer-to-peer systems based on distributed hash tables have used structured "overlay" networks to map table keys to local TCP/IP routes [Korzun and Gurtov 2013; Lua et al. 2005].

Many distributed consensus protocols (such as those used by crypto-currencies) are designed to provide tolerance against arbitrary or adversarial faults. Byzantine agreement protocols [Castro et al. 1999; Lamport et al. 1982] provide tolerance against arbitrary faults (including attacks) under some circumstances, but are limited to small networks due to poor scalability. Proof-of-work [Dwork and Naor 1993; Nakamoto 2008] and proof-of-stake [King and Nadal 2012] provide better scalability, but wasteful of computational and energy resources. Federated Byzantine Agreement (FBA) [Mazires 2015] is scalable and optimal, guaranteeing that the protocol only fails when success is impossible, although it does not provide a way to evaluate the probability of failure. All of these protocols rely on the assumption that their cryptography cannot be compromised. While cryptography can be extremely resistant to technological attacks, it can still be compromised through coercion (e.g., legal action). In additon, the fault-tolerance of FBA depends on redundancy in the network structure but leaves an unanswerd question: how should a network be structured to achieve fault tolerance? This paper addresses both of these issues by analyzing the role of network structure on adversarial faults, without relying on assumptions of cryptographic integrity.

Multipath routing uses multiple paths when routing a message through a network, in contrsast to traditional *unipath* routing, which uses a single path. Multipath routing can have many benefits, including reduced congestion, increased throughput, and more reliability [Qadir et al. 2015]. Many of these routing protocols offer better security [Zin et al. 2015]. Some approaches utilize redundant paths as backups for increased fault tolerance [Alrajeh et al. 2013], and some specificially protect against adversarial faults [Khalil et al. 2010; Kohno et al. 2012; Lou and Kwon 2006]. The method of Liu et al. [Liu et al. 2012] routes multiple messages first to random peers and then to their final destination. The butterfly algorithm we present takes a conceptually similar approach. Most work on multipath routing has been motivated by applications related to wireless sensor networks (WSNs), and have thus focused on ad-hoc, unstructured networks, often having a central base station. The trust model presented in this paper provides a more general way to evaluate the effect of network structure on adversarial fault-tolerance.

3. ATTACK-TOLERANT NETWORKS INFRASTRUCTURE

In this section, we describe the functional properties required of an attack-tolerant network infrastructure and how those properties translate into constraints on network structure. We pay special attention to a property we call *stabilizing asymmetry*.

3.1. Functional Properties

Scalability. For large scale networks it is important that the infrastructure allows for the network to grow while remaining functional. In practice, people, devises, and connections, have limited capabilities and these limitations need to be considered as part of the design of the infrastructure.

Decentralization. Systems having single points of failure are less tolerant against faults at those points, which both raises the likelihood of a failure and creates targets where attackers might be able to focus their resources efficiently. Attack-tolerant infrastructure must be decentralized in order to minimize single points of failure.

Stabilizing asymmetry. In the context of international conflict, [Mack 1975] observed that power imbalance usually determines the outcome of a conflict (with the more powerful side winning) except in the special case of *asymmetric conflicts*. In asymmetric conflicts, the same level of resource expenditure yields different results for different parties. There are two possible cases: the attacker's resources are either more or less effective than the defender's. We call the latter case stabilizing asymmetry, because it lowers the incentive to attack. With this in mind, an attack-resistant infrastructure will benefit from a high level of stabilizing asymmetry.

3.2. Structural Properties

We now describe the structural properties of a network that allow it to achieve the functional properties described above.

Sparsity and low diameter. To achieve scalability, networks must be *sparse* and have a *low diameter*. In practical settings, humans and devices have an upper limit on the number of connections they can maintain. In sparse networks, the number of links grows slowly as the network grows in size, allowing the network to scale without exceeding the nodes' capacity for links. Similarly, low-diameter guarantees that as a network grows, a short path will still exist between any pair of nodes. While low diameter guarantees a path exists, paths are only useful if an efficient *routing* algorithm exists to find them.

Vertex transitivity. In vertex transitive networks, for any pair of nodes, an edge-preserving map always exists from one node to the other. In other words, all nodes occupy structurally indistinguishable positions in the network. Networks with *vertex transitivity* are decentralized in the sense that they structure does no allow for single points of failure.

Redundancy. Redundancy in a network refers to the extent to which nodes are connected to each other by multiple independent paths. Redundancy can help reduce single points of failure as well as increase fault-tolerance. One measure of redundancy is given by the ratio of edges to nodes [Baran and others 1964]. A single point of failure occurs when a node holds a uniquely central position, and can be eliminated by adding alternative paths around the node, which increases the network's redundancy. Similarly, as we will later show, redundancy in a network can enable fault-tolerance techniques that reduce the effectiveness of attacks and creating stabilizing asymmetry. When secrecy is desired, redundancy can help by allowing messages to be divided into parts and sent along different paths so even if one path is compromised, the whole message is not revealed [].

3.3. Trust

The sparsity requirement has an important consequence for network infrastructure: as a network grows, the number of nodes will exceed the number of links a given node can support and many pairs of nodes will only be indirectly connected, with all paths between them passing through at least one intermediary node. These intermediary nodes are targets for potential attacks. More specifically, the threat model is as follows: Alice sends a message to Bob via one or more intermediaries, but some intermediaries may try to either block or alter the message. The ability to tolerate these types of adversarial attacks depends on Alice and Bob's ability to assume that some nodes have not been compromised—to *trust* them.

Both centralized and decentralized approaches are commonly used to create trust infrastructures. Centralized approaches such as *public key infrastructure* (PKI) suffer from a number of vulnerabilities [Ellison and Schneier 2000], which stem largely from the single points of failure inherent to centralization. Alternatively, the decentralized *web of trust* model [Ferguson and Schneier 2003; Zimmermann 1995] allows individuals to choose who they trust initially. Trust is then extended to new individuals if they are vouched for by a currently-trusted individual. However, the web of trust approach assumes that trust can be extended transitively, which is not generally true [Christianson and Harbison 1997]. We note that while such approaches are decentralized in the sense that no central authority holds the keys, they do not necessarily incorporate

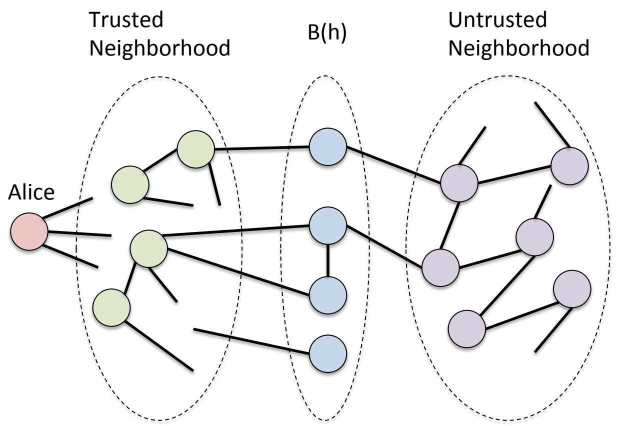


Fig. 1. Illustration of the *partial trust* model. Edges represent direct, mutual trust relationships. Alice (v) trusts all nodes less than h hops away—her *trusted neighborhood* $T_h(v)$. Beyond that, the nodes at distance h form her *trust boundary* $B_h(v)$. We assume adversaries are unable to compromise nodes in the trusted neighborhood. By compromising all nodes in the trust boundary, an adversary can ensure that all communications leaving Alice's trusted neighborhood are compromised.

redundancy and the network they form may not be vertex transitive. Thus they still may suffer from single points of failure.

4. PARTIAL TRUST

We now propose a decentralized trust model based on weaker transitivity assumptions than existing web of trust models. In following sections, we will use this model along with network redundancy to minimize vulnerabilities due to single points of failure. Our model is based on the assumption of partially-transitive trust. While this assumption is still idealized, it is a more realistic improvement on fully-transitive models.

Our model (Figure 1) is represented as an undirected graph, with nodes representing individual agents and edges representing mutual trust relationships. We define a *trust radius* h and assume that trust is transitive up to h hops away. In other words, the nodes v and w trust each other if the distance between v and w, $d(u, v)$ is less than h. For a given node v, we call the set of nodes that v trusts the *trusted neighborhood* $T_h(v)$, and all nodes at exactly distance h from v the *trust boundary* $B_h(v)$:

$$T_h(v) = \{w \mid d(v, w) < h\} \quad (1)$$
$$B_h(v) = \{w \mid d(v, w) = h\}\,. \quad (2)$$

We assume that v is very familiar with nodes in its trusted neighbor and v would know if any of these nodes were compromised by an attacher. Hence, an adversary of v has no incentive to compromise a node in $T_h(v)$. We thus assume that no nodes within the trusted neighborhood are compromised by an adversary of v. However, by compromising all nodes in the trust boundary, an adversary can ensure that all communications leaving the trusted neighborhood are compromised.

Now we consider a source v (Alice) and a destination w (Bob), shown in Figure 2. We also consider an adversary who aims at disrupting all possible communication paths between Alice and Bob. Since the adversary cannot compromise any nodes in the trusted neighborhoods of Alicia or Bob, the only option is to compromise nodes outside of the trust neighborhoods. An adversary could, for example, compromise all possible paths between Alice and Bob by compromising either Alice or Bob's entire trust boundary. This would require compromising $\min(|T_h(v)|, |T_h(w)|)$ nodes. In a ver-

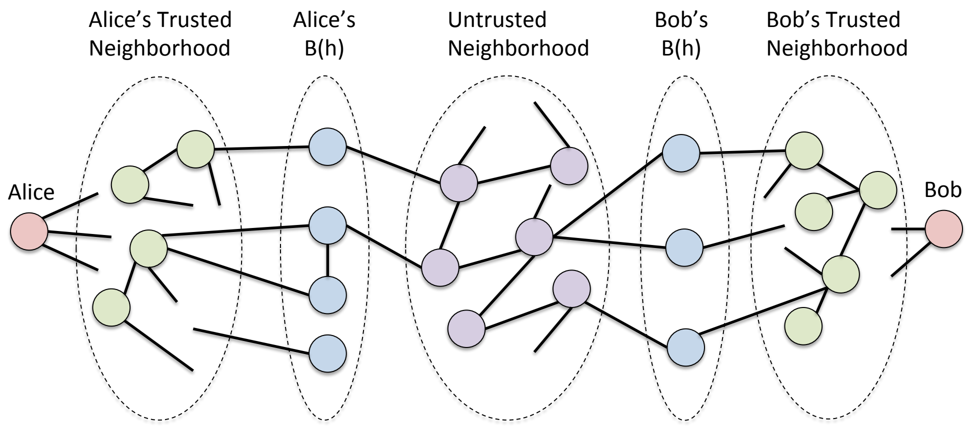


Fig. 2. Partial trust model with sender (Alice) and receiver (Bob). If disjoint paths exist connecting each node in Alice's trust boundary to a node in Bob's, than an adversary needs to compromise at least $|T_h(v)|$ nodes to compromise all possible paths between Alice and Bob.

tex transitive network, these two trust boundaries will be the same size. Alternatively, the adversary could try to compromise a small set of nodes such that all paths between Alice and Bob pass through one of these nodes. However, If disjoint paths exist connecting each node in Alice's trust boundary to a node in Bob's, then the cut size between the two can never be smaller than $|T_h(v)| = |T_h(v)|$, so an adversary who compromises fewer nodes cannot compromise all possible paths between Alice and Bob.

5. MULTIPATH FAULT TOLERANCE

When the assumptions of the partial trust model hold, a network can achieve robust fault tolerance by utilizing the redundancy of multiple paths between the sender and receiver. Furthermore, this fault tolerance can remain robust even when faults are adversarially chosen by an attacker who has access to more resources than the sender and receiver. On a structured network, multipath fault tolerance reveals not just when an error has occured, but also provides information about which paths provide conflicting information, which could be used to help locate compromised nodes.

In the terminology of fault tolerance, a *fault* occurs when one component of a system behaves incorrectly (e.g., a routing node blocking or altering a message). The result of that fault (e.g., a recipient receiving conflicting messages) is an *error* state. If the error is undetected or incorrectly corrected, the system is said to have experienced a *failure* (e.g., an altered message is accepted as authentic). We are concerned in particular with *adversarial faults*, which (as opposed to random faults) are chosen strategically (within the assumptions of the trust model) to maximize the liklihood of a failure. In this section, we focus on the simplest possible question: when using multipath fault tolerance, what is the probability that adversarial faults will cauase an undetectable error, and thus guarantee a failure?

They key idea behind multipath fault tolerance is that several copies of a message are sent, along multiple routes, rather than using traditional unicast routing. If any of the message copies differ from one another or do not arrive, the receiver can conclude that an error has occurred, and possibly correct it, depending on the magnitude of the error and the desired fault tolerance properties.

In the simplest such scheme, a sender (Alice) broadcasts copies of a message along all possible routes to the receiver (Bob). Such an approach takes advantage of all possible redundancy in the network, and requires an adversary (Eve) to compromise at least one node on each path in order to create an undetectable error. But how many nodes is that? The answer depends on the smallest *vertex cut* of the untrusted portion of the network—the smallest set of nodes than can be removed to separate the network into two components. Any path between Alice and Bob must pass through all such sets, so to guarantee an undetectable error, Eve must compromise at least as many nodes as are in the smallest vertex cut. Alice and Bob's trust boundaries are both vertex cuts, so the smaller of the two is an upper bound. It is not difficult to construct networks in which the trust boundary is exactly the smallest vertex cut, and we will show this to be true for the networks we consider below. We denote the size of the smallest vertex cut—the maximum number of disjoint paths between Alice and Bob's trusted neighborhoods—as B.

Realistically, the sender may only have the resources to send a limited number of copies, or the network may not be able to handle the congestion caused by sending all messages along all paths. However, we will now show that a network can provide a high level of fault tolerance with only a few copies of a message, provided the number of disjoint paths B is large.

In a more sophisticated multipath fault tolerance scheme, Alice randomly chooses k paths and sends a copy of her message on each. We further assume that Alice posesses a routing algorithm (such as described in the next section) that enables her to construct paths that do not intersect between trusted neighborhoods. So the maximum number of paths Alice could choose is B, otherwise two paths would have to intersect in the minimum vertex cut. Let us assume that Eve is capable of compromising up to l nodes. Eve's best adversarial strategy is now to compromise as many nodes in the smallest vertex cut as possible, succeeding when each of Alice's paths include one of the compromised nodes. Since Alice chooses paths randomly, all nodes in the vertex cut are equally likely to contain a path, so Eve can do no better than also choosing randomly. If $k > l$, at least one message will get through uncompromised and all errors are detectable. Otherwise, the probability of Eve producing an undetectable error is the probability that all of Alice's paths contain compromised nodes:

$$p_f = \frac{l!(B-k)!}{B!(l-k)!}. \tag{3}$$

Letting $k = \alpha B$ and $l = \beta B$, then applying Stirling's approximation gives:

$$p_f \approx \frac{\sqrt{\beta(1-\alpha)}}{\sqrt{\beta-\alpha}} \left[\left(\frac{\beta-\alpha}{1-\alpha} \right)^{\alpha} \left(\frac{\beta}{\beta-\alpha} \right)^{\beta} (1-\alpha) \right]^{B}. \tag{4}$$

Figure 3 shows the value of p_f as a function of k and l for various values of B. Equation (4) shows that while p_f depends on the fractions of redundant paths actually utilized α and compromised β, it decreases exponentially in the total number of *possible* redundant paths. This result is significant because B depends only on the network structure, not on the amount of resources available to the sender or the adversary. In other words, senders can tolerate attacks from significantly more powerful adversaries, as long as the network structure provides large B. Formally we have shown that under the partial trust assumption, multipath fault tolerance can provide *asymptotic security* against adversarial faults if B can be made arbitrarily large. Similarly, because the cost of an attack is prohibitively large compared to the cost of protecting against an attack, this architecture creates what a stabilizing asymmetry. By putting attackers at a disadvantage, attacks are not only tolerated but also disincentivized.

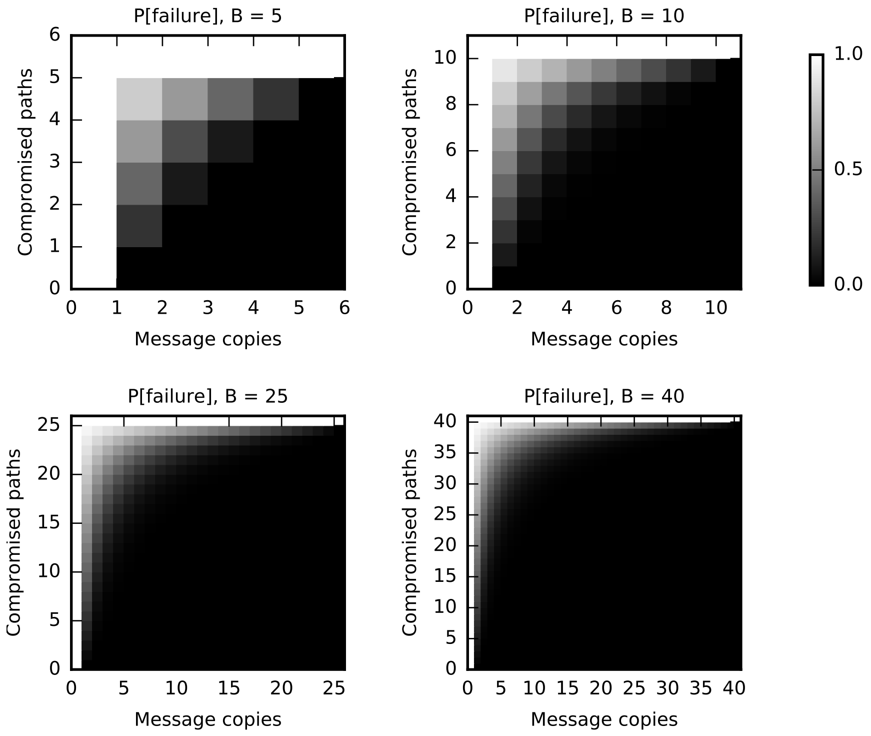


Fig. 3. The probability of an undetectable error as a function of the number of message copies and the number of adversarial faults, for various numbers of redundant paths.

6. MULTIPATH ROUTING ON THE BUTTERFLY TOPOLOGY

6.1. Butterfly Network Topology

We now present a decentralized, fault-tolerant, multipath routing algorithm on a specific topology known as the butterfly network topology [Kshemkalyani and Singhal 2008]. Several variations on the butterfly network exist. Specifically, we utilize the wrap-around butterfly. We denote the m-dimensional, directed wrap-around butterfly as $\mathrm{wBF}(m)$:

$$\begin{aligned}
\mathrm{wBF}(m) &= (V, E_{\downarrow} \cup E_{\rightarrow}) && (5)\\
V &= \mathbb{Z}_m \times \mathbb{Z}_2^m && (6)\\
E_{\downarrow} &= \{((l,z),(l+1 (\mathrm{mod}\ m), z)\} && (7)\\
E_{\rightarrow} &= \{(l,z),(l+1 (\mathrm{mod}\ m),(z_0,\ldots,z_{l-1},z_l \oplus 1, z_{l+1},\ldots,z_{m-1})\}, && (8)
\end{aligned}$$

where $\mathbb{Z}_m$ is the set of integers modulo m, and $\oplus$ represents the addition modulo 2. Each node is associated with a level l and an m-bit string z known as *the place-within-level*. There are two types of edges, shown in Figure 4. Down edges ($E_{\downarrow}$) connect nodes sharing the same z value in a cycle of increasing level l. Down-right edges ($E_{\rightarrow}$) also link to a node of level $l+1$, but one having the place-within-level equal to z with the lth bit inverted.

The wrap-around butterfly network has a number of properties desirable for a scalable, decentralized communication network.

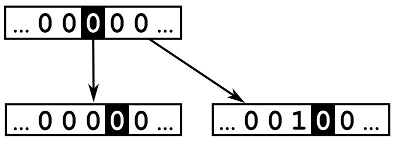


Fig. 4. Schematic illustration of the two types of edges in a directed butterfly network. The node (l, z) is shown as the bit string z with a square around the lth bit. "Down" edges increment l, leaving z unchanged, while "down-right" edges increment l and invert the lth bit of z.

Vertex-transitive. Because the wraparound butterfly is vertex transitive, it is maximally decentralized. We will also use this property throught the proof, noting that the problem of finding a route between arbitrary nodes $\tilde{v}$ and $\tilde{w}$ can be reduced to finding a route from node $(0, 0)$ to some node w.
Small-diameter. For any two nodes, the length of the shortest path between them is $O(\log N)$, where N is the number of nodes in the network.
Sparse. With a constant degree of 4, the wraparound butterfly is extremely sparse, and can scale indefnitely without node degree becoming a limitation.
Redundant. The number of independent paths between the h-trusted neighborhoods of two nodes in the wraparound butterfly increases exponentially in h. We prove this assertion below by giving an algorithm to explicitly construct these paths.

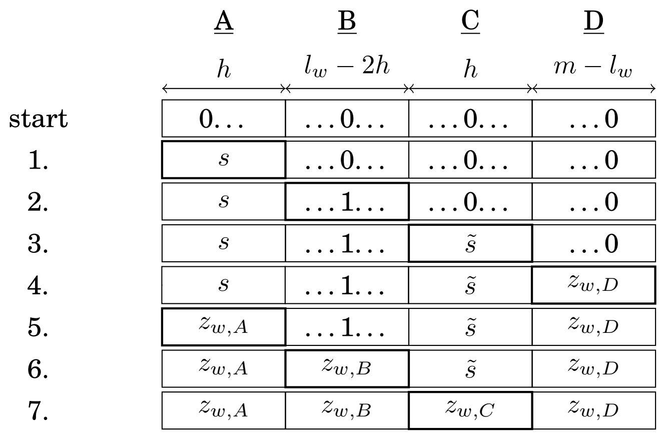
6.2. Routing Algorithm: Overview

This section gives an informal overview of the multipath routing algorithm. In a wraparound butterfly network with trusted radius h, the algorithm constructs 2^h disjoint paths between the trusted neighborhoods of any two nodes v and $w = (l_w, z_w)$. Because of vertex transitivity, we can make the simplifying assumption that $v = (0, 0)$ without loss of generality. Each path is parameterized by an h-bit string $s \in \mathbb{Z}_2^h$ (see Table I). Each path cycles around the levels of the butterfly network at most twice.

During the first cycle, the place-within-level z of all nodes is set such that no two paths overlap outside the trusted neighborhoods. During the second cycle, the place-within-level is set to its destination value and the algorithm terminates when the destination level is reached.

More specifically, consider any node, (l, z), in the path corresponding to s. The algorithm can be understood by dividing the m bits of z into four segments: A, B, C, D (see figure 5). A is the first h bits, and C is the h bits preceeding index l_w. B is formed by the remaining bits between A and C, while D refers to the remaining bits after C. Hence, $A, B, C,$ and D occupy h, $l_w - 2h$, h, and $m - l_w$ bits of z, respectively (we leave the caase of overlapping A and C for the formal proof). Note that due to the construction of the network, as we walk through any path $\{p_1, p_2, \ldots\}$, the level l_{t+1} of node p_{t+1} is always one higher (mod m) than level l_t the previous node p_t. Also, z_{t+1} is always the same as the z_t, except for possibly the $(l+1)^{th}$ bit of z_{t+1}, which can be either 0 or 1. Hence, as the routing algorithm progresses through the levels l, the segments of z get updated in order.

There are 7 stages of the algorithm to construct a path from v to w. Stages (1) through (4) consist of the first cycle thought the levels of the network and stages (5) though (7) consists of the second (potentially partial) cycle. We start at the source node $v = (0, 0)$ and begin constructing the path. During stage (1), A is set to match s. During stage (2) the bits of B are inverted. During stage (3) a cyclic permutation of s is placed into C (the details of the permutation are necessary for the overlapping case, but not relevant for this overview). After this, in stages (4–7), D, A, B, and C are set to their



	$\underline{A}$	$\underline{B}$	$\underline{C}$	$\underline{D}$
	h	$l_w - 2h$	h	$m - l_w$
start	$0\ldots$	$\ldots0\ldots$	$\ldots0\ldots$	$\ldots0$
1.	s	$\ldots0\ldots$	$\ldots0\ldots$	$\ldots0$
2.	s	$\ldots1\ldots$	$\ldots0\ldots$	$\ldots0$
3.	s	$\ldots1\ldots$	$\tilde{s}$	$\ldots0$
4.	s	$\ldots1\ldots$	$\tilde{s}$	$z_{w,D}$
5.	$z_{w,A}$	$\ldots1\ldots$	$\tilde{s}$	$z_{w,D}$
6.	$z_{w,A}$	$z_{w,B}$	$\tilde{s}$	$z_{w,D}$
7.	$z_{w,A}$	$z_{w,B}$	$z_{w,C}$	$z_{w,D}$

Fig. 5. Progression of place-within-level z as the multipath routing algorithm cycles through the levels of the butterfly network.

Table I. Butterfly Multipath Routing Variables

NAME	VARIABLE
butterfly dimension	$m \in \mathbb{Z}_+$
node level	$l \in \mathbb{Z} : 0 \leq l < m$
node place within level	$z \in \mathbb{Z}_2^m$
trust radius	$h \in \mathbb{Z} : 1 \leq h \leq \lfloor m/2 \rfloor$
path index	$s \in \mathbb{Z}_2^h$

destination values (see figure 5). Figure 6 shows an explicit example of the routing algorithm for one value of s.

We now argue that in the above routing algorithm, given $s, s' \in \mathbb{Z}_2^h$ such that $s \neq s'$, the corresponding paths, P_s and $P_{s'}$, are disjoint when they are both outside of the trusted neighborhoods of v and w. We informally refer to different segments of nodes in s and s' as A, B, C, D and A', B', C', D', respectively.

First note that any overlap between the two paths must occur either while they are at the same stage of the routing, or while one is at stage (1) and the other one at (5), one is at stage (2) and the other one at (6), or one is at stage (3) and the other one at (7). This is because in any other case, the nodes in the paths will be at different levels, l, and thus they cannot overlap. Furthermore, when a path is in stage (1) or (7), it is in the trusted neighborhoods of v or w, hence we are not concerned with overlaps within stages (1) or (7).

When both paths are in stages (2), (3), or (4), they do not overlap because segment A is set to s and segment A' is set to s'. When both paths are in stages (5) or (6), they do not overlap because segment C is set to a cyclic permutation of s and segment C' is the same cyclic permutation of s'. When one path is in stage (1) and the other one is in stage (5), say P_s is in (1) and $P_{s'}$ is in (5), they do not overlap because all bits in B are set to the original bits in the source node and all bits in B' are set to the inverted bits of the source node. When one path is in stage (2) and the other one is in stage (6), they do not overlap because at least one pair of bits in B and B' are still inverted while stage (6) is being processed.

6.3. Routing Algorithm: Specification and Proof

We now specify the multipath routing scheme and prove that provides 2^h disjoint paths between the h-hop trusted neighborhoods of any two nodes in an m-bit wrap-around

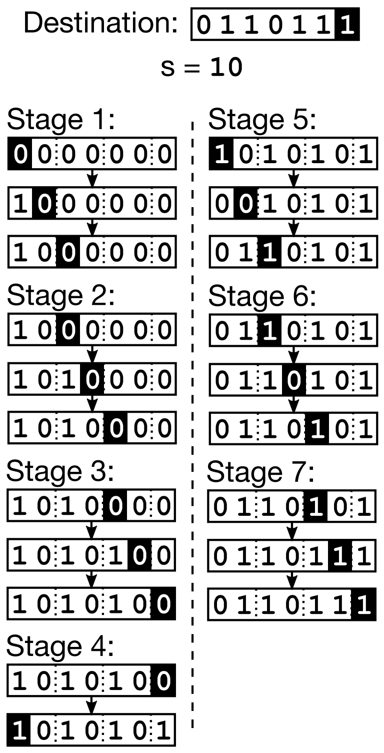


Fig. 6. An example of one path as constructed by the proposed multipath routing algorithm. The path is shown for $s = 10_2$ and $w = (6, 0110111_2)$.

butterfly network. Utilizing vertex transitivity, we label the source node as $(l^{(0)}, z^{(0)}) = (0, 0)$ and denote the destination node as $w = (l_w, z_w)$, without loss of generality.

Let s be an h-bit binary string with s_i denoting the bit at index i. There are 2^h such strings. Let $v_s^{(t)} = (l^{(t)}, z^{(t)})$ be the node at position t in the path parameterized by s. For convenience, we will omit the subscript s when it is obvious from context. We define three distinct partitions of m-bit binary strings. Let $Q_{v^{(0)}}$ ($\overline{Q_{v^{(0)}}}$) be the set of m-bit strings in which the bits at all indeces $h \le i < l_w - h$ match (do not all match) those of $z^{(0)}$. Note that $Q_{v^{(0)}}$ is trivially all m-bit strings if $l_w < 2h$. Let R_s ($\overline{R_s}$) be the set of m-bit strings with the lowest h bits all matching (not all matching) the bits of s. Let S_s ($\overline{S_s}$) be the set of m-bit strings with the h bits preceeding index l_w all matching (not all matching) the bits of $\tilde{s}$, where $\tilde{s}$ is a cyclic permutation of s defined below.

$$Q_{v^{(0)}} = \{z \in \mathbb{Z}_2^m \mid \neg\exists i \in \mathbb{Z} : h \le (i < l_w - h) \wedge z_i \ne z_i^{(0)}\} \tag{9}$$
$$\overline{Q_{v^{(0)}}} = \mathbb{Z}_2^m \setminus Q_{v^{(0)}} \tag{10}$$
$$R_s = \{z \in \mathbb{Z}_2^m \mid \forall i \in \mathbb{Z}_2^h z_i = s_i\} \tag{11}$$
$$\overline{R_s} = \mathbb{Z}_2^m \setminus R_s \tag{12}$$
$$S_s = \{z \in \mathbb{Z}_2^m \mid \forall i \in \mathbb{Z}_2^h z_{(l_w - h + i)} = \tilde{s}_i\} \tag{13}$$
$$\overline{S_s} = \mathbb{Z}_2^m \setminus S_s \tag{14}$$
$$\tilde{s}_i = s_{(i + l_w) \bmod h}. \tag{15}$$

We will make use of the fact that:

$$s \neq s' \implies S_s \cap S_{s'} = R_s \cap R_{s'} = \emptyset. \tag{16}$$

Routes are contructed in 7 stages. The network topology dictates that $l^{(t+1)} = l^{(t)} + 1$ (mod m), so we let $l = t$ (mod m). and that $z^{(t+1)}$ is equal to $z^{(t)}$ with or without the bit in index $l^{(t)}$ inverted, depending on whether the down or down-right edge was taken at step t.

Stage 1: *($0 \leq t < h$).* Down or down-right edges are chosen such that the tth bit of $z^{(t+1)}$ is equal to the tth bit of s. Throughout Stage 1, all nodes are within the sender's trusted neighborhood. Throughout Stage 1, $z^{(t)} \in Q_{v^{(0)}}$. At the end of Stage 1, $z^{(h)} \in S_s$, and $z^{(t)}$ will remain so until the level cycles to 0 at $t = m$.
Stage 2: *($h \leq t < l_w - h$).* Edges are chosen to make the tth bit of $z^{(t+1)}$ the inverse of the tth bit of $z^{(0)}$. Note that this stage does not occur when $l_w < 2h$. If this stage occurs, then $z^{(t)} \in \overline{Q_{v^{(0)}}}$ until these levels are reached again in stage 6.
Stage 3: *($l_w - h \leq t < l_w$).* The bits of $z^{(t)}$ are chosen to match $\tilde{s}$, such that after the stage is complete, $z^{(t)} \in R_s$.
Stage 4: *($l_w \leq t < m$).* Paths are chosen such that the tth bit of $z^{(t+1)}$ matches that of the destination node z_w. This stage will not occur if $l_w > m - h$.
Stage 5: *($m \leq t < m + h$).* There are two cases. If $2h < l_w < m - h$, then there is no overlap between the indeces defining R_s and S_s. In this case, the first h bits of $z^{(t)}$ are set to match z_w. Otherwise there is some overlap between the indeces defining R_s and S_s. In this case, the each of the first h bits of $z^{(t)}$ is either kept the same if $l_w - h \leq l < l_w$, or set to the corresponding bit of z_w otherwise. In this stage and after, $z^{(t)}$ is no longer guaranteed to be in R_s. However, $z^{(t)}$ remains in S_s during and after this stage.
Stage 6: *($m + h \leq t < m + l_w - h$).* In this stage, edges are chosen to set the bits of $z^{(t)}$ to their corresponding value in z_w. $z^{(t)} \in \overline{Q_{v^{(0)}}}$ throughout this stage, but not afterwards.
Stage 7: *($m + l_w - h \leq t < m + l_w$).* The h bits of $z^{(t)}$ preceeding index l_w are set to match z_w. All nodes in this stage are within h hops of w and thus in its trusted neighborhood. After this stage, $v^{(m+l_w)} = w$ and routing is complete.

For all 2^h of these paths, the components excluding nodes within h hops of v or w (in their trusted neighborhoods) are pairwise disjoint, which we now prove. Nodes from two paths can only coincide if their levels are the same. Nodes which share a level must either be in the same stage, or 4 stages apart. Let (a,a') denote a pair of sub-paths corresponding to stage a of one path and stage a' of another. Excluding paths that intersect in their trusted neighborhoods, (1,1) and (7,7), we have reduced the list of possible intersections to the following cases: (2,2), (3,3), (4,4), (5,5), (6,6), (1,5), (2,6), and (3,7). Nodes in stages 2–4 belong to R_s so cannot overlap with any stage 2–4 nodes from another path, eliminating (2,2), (3,3), and (4,4). Similarly, nodes in stages 4–6 belong to a unique S_s, eliminating (5,5) and (6,6). Nodes in stage 1 belong to $Q_{v^{(0)}}$ while those in stage 5 belong in its complement, eliminating (1,5). Similarly, for all l in stage 2, $z^{(l)}$ is equal to $z^{(0)}$, while in stage 6, $z^{(l)}$ is the inverse, eliminating (2,6) This leaves only (3,7), a collision which can occur only for only one path (with s matching the first h bits of z_w), and which enters the trusted neighborhood in stage 3. For this single path, we can proceed directly from stage 2 to stage 7, eliminating the last possible collision.

Thus, we have shown that assuming the partial trust model, with trust transitive for h hops, we can construct 2^h paths on a wraparound butterfly topology which do not intersect outside the trusted neighborhoods of the source and destination. This is a lower bound on the value B in Equation (3), showing that the decentralized, redundant, structured networks such as the butterfly can have a very low probability of failure when faced with adverarial faults, even from a very powerful attacker.

7. DISCUSSION

While decentralized architectures have recieved much attention, methods for classifying, and evaluating decentralization are in their infancy. The trust model and multipath fault tolerance technique proposed in this paper provide a practical fault tolerance architecture for applications when network structure can be imposed, as well as a more general method for evaluating the role of network structure on fault tolerance.

Fault-tolerant network infrastructures have many direct applications. Areas such as cryptocurrency, secure multiparty computation, and wireless sensor networks have immediate need for scalable, fault-tolerant infrastructures. Many Internet services—email, social networks, cloud storage—are still highly centralized and vulnearble to technical and non-technical attacks. Decentralized fault tolerance is one approach to securing these important services and making networked communication safer.

While the primary purpose of our proposed fault tolerance technique is the threat of message blocking or corruption, it can provide some improvements to secrecy and privacy. For example, if a message is broken into parts and each part is sent along a different random subset of routes, an attacker who succesfully compromises one part of the message will still be unable to compromise the rest.

Our present contributions are limited in several ways. The partial trust model, while making fewer assumptions than web of trust approaches, still depends on some transitivity of trust, which may be a questionable assumption in many cases. Better trust models are still needed.

The primary limitation of our structured network approach is that link structure is imposed rather than self-organized. This requirement may be appropriate for appliations such as wireless sensor networks, or slow-changing institutional infrastructure, in which the architect has considerable control over link structure. But in general, it remains an open question whether self-organized networks can be restructured, or buit from the ground up in such a way that structured multipath fault tolerance technqiues can be applied. Similarly, structured network topologies such as the butterfly are defined only for specific values. Although these values can be as large as desired, how to apply methods such as ours to networks with an intermediate number of nodes remains an open question.

8. CONCLUSION

We have developed a partial trust model which enables a formal analysis of the effect of decentralized network structure on fault tolerance properties. We have also proposed a general technique for achieving and evaluating adversarial fault tolerance in a structured network with partial trust. We found that the probability of an adversary causing an undetectable error decreases exponentially as the number of redundant paths between the neighborhoods of the sender and receiver grows. This property puts attackers at a disadvantage, creating a stabilizing asymmetry which disincentivizes attack. We also presented a specific implementation of the multipath fault tolerance technique on a wraparound butterfly network topology and proved that the number of redundant paths grows exponentially with number of network hops that can be trusted. Our results are directly applicable to architectures in which the link structure can be imposed by the architect, and provide a formalism that can be used more generally to evaluate the role of network structure and redundancy on the fault tolerance of decentralized networks.

ACKNOWLEDGMENTS

REFERENCES

Rka Albert, Hawoong Jeong, and Albert-Lszl Barabsi. 2000. Error and attack tolerance of complex networks. *nature* 406, 6794 (2000), 378–382.

Nabil Ali Alrajeh, Mohamad Souheil Alabed, and Mohamed Shaaban Elwahiby. 2013. Secure ant-based routing protocol for wireless sensor network. *International Journal of Distributed Sensor Networks* 2013 (2013).

Albert-Lszl Barabsi and Rka Albert. 1999. Emergence of scaling in random networks. *science* 286, 5439 (1999), 509–512.

Albert-Lszl Barabsi and others. 2009. Scale-free networks: a decade and beyond. *science* 325, 5939 (2009), 412.

Paul Baran and others. 1964. On distributed communications. *Volumes I-XI, RAND Corporation Research Documents, August* (1964), 637–648.

Miguel Castro, Barbara Liskov, and others. 1999. Practical Byzantine fault tolerance. In *OSDI*, Vol. 99. 173–186.

Bruce Christianson and William S Harbison. 1997. Why isn't trust transitive?. In *Security protocols*. Springer, 171–176.

Cynthia Dwork and Moni Naor. 1993. Pricing via processing or combatting junk mail. In *Advances in CryptologyCRYPTO92*. Springer, 139–147.

Carl Ellison and Bruce Schneier. 2000. Ten risks of PKI: What you're not being told about public key infrastructure. *Comput Secur J* 16, 1 (2000), 1–7.

Niels Ferguson and Bruce Schneier. 2003. *Practical cryptography*. Wiley, New York.

Philip Hunter. 2008. Pakistan YouTube block exposes fundamental internet security weakness: Concern that pakistani action affected youtube access elsewhere in world. *Computer Fraud & Security* 2008, 4 (2008), 10–11.

Issa Khalil, Saurabh Bagchi, Cristina N Rotaru, and Ness B Shroff. 2010. UnMask: Utilizing neighbor monitoring for attack mitigation in multihop wireless sensor networks. *Ad Hoc Networks* 8, 2 (2010), 148–164.

Sunny King and Scott Nadal. 2012. *Ppcoin: Peer-to-peer crypto-currency with proof-of-stake*. August.

Eitaro Kohno, Tomoya Okazaki, Mario Takeuchi, Tomoyuki Ohta, Yoshiaki Kakuda, and Masaki Aida. 2012. Improvement of assurance including security for wireless sensor networks using dispersed data transmission. *J. Comput. System Sci.* 78, 6 (2012), 1703–1715.

Dmitry Korzun and Andrei Gurtov. 2013. *Structured peer-to-peer systems: fundamentals of hierarchical organization, routing, scaling, and security*. Springer, New York, NY.

Ajay D Kshemkalyani and Mukesh Singhal. 2008. *Distributed computing: principles, algorithms, and systems*. Cambridge University Press.

Leslie Lamport, Robert Shostak, and Marshall Pease. 1982. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)* 4, 3 (1982), 382–401.

Anfeng Liu, Zhongming Zheng, Chao Zhang, Zhigang Chen, and Xuemin Shen. 2012. Secure and energy-efficient disjoint multipath routing for WSNs. *Vehicular Technology, IEEE Transactions on* 61, 7 (2012), 3255–3265.

Wenjing Lou and Younggoo Kwon. 2006. H-SPREAD: a hybrid multipath scheme for secure and reliable data collection in wireless sensor networks. *Vehicular Technology, IEEE Transactions on* 55, 4 (2006), 1320–1330.

Eng Keong Lua, Jon Crowcroft, Marcelo Pias, Ritu Sharma, and Sharon Lim. 2005. A survey and comparison of peer-to-peer overlay network schemes. *Communications Surveys & Tutorials, IEEE* 7, 2 (2005), 72–93.

Andrew Mack. 1975. Why big nations lose small wars: The politics of asymmetric conflict. *World Politics* 27, 02 (1975), 175–200.

David Mazires. 2015. *Stellar Consensus Protocol: A Federated Model for Internet-level Consensus*.

Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. *bitcoin.org* (2008), 28.

Junaid Qadir, Anwaar Ali, Kok-Lim Alvin Yau, Arjuna Sathiaseelan, and Jon Crowcroft. 2015. Exploiting the power of multiplicity: a holistic survey of network-layer multipath. *Communications Surveys & Tutorials, IEEE* 17, 4 (2015), 2176–2213.

James PG Sterbenz, David Hutchison, Egemen K etinkaya, Abdul Jabbar, Justin P Rohrer, Marcus Schller, and Paul Smith. 2010. Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines. *Computer Networks* 54, 8 (2010), 1245–1265.

Philip R Zimmermann. 1995. *The official PGP user's guide*. MIT press.

Shazana Md Zin, Nor Badrul Anuar, Miss Laiha Mat Mat Kiah, and Ismail Ahmedy. 2015. Survey of secure multipath routing protocols for WSNs. *Journal of Network and Computer Applications* 55 (2015), 123–153.