Brief Announcement: On the Resilience of Routing Tables

Joan Feigenbaum
Yale University
joan.feigenbaum@yale.edu

Brighten Godfrey
UIUC
pbg@illinois.edu

Aurojit Panda
UC Berkeley
apanda@cs.berkeley.edu

Michael Schapira
Hebrew University
schapiram@huji.ac.il

Scott Shenker
UC Berkeley
shenker@eecs.berkeley.edu

Ankit Singla
UIUC
singla2@illinois.edu

Categories and Subject Descriptors

C.2.2 [**Network Protocols**]: Routing Protocols

Keywords

Internet routing, fault tolerance

ABSTRACT

Many modern network designs incorporate "failover" paths into routers' forwarding tables. We initiate the theoretical study of such *resilient routing tables*.

1. INTRODUCTION

The core mission of computer networks is delivering packets from one point to another. To accomplish this, the typical network architecture uses a set of forwarding tables (that dictate the outgoing link at each router for each packet) and a routing algorithm that establishes those forwarding tables, recomputing them as needed in response to link failures or other topology changes. While this approach provides the ability to *recover* from an arbitrary set of failures, it does not provide sufficient *resiliency* to failures because these routing algorithms take substantial time to reconverge after each link failure. As a result, for periods of time ranging from 10s of milliseconds to seconds (depending on the network), the network may not be able to deliver packets to certain destinations. In comparison, packet forwarding is several orders of magnitude faster: a 10 Gbps link, for example, sends a 1500 byte packet in 1.2 μsec.

In order to provide higher availability we must design networks that are more resilient to failures. To this end, many modern network designs incorporate various forms of "backup" or "failover" paths into the forwarding tables that enable a router (or switch), when it detects that one of its attached links is down, to use an alternate outgoing link. We call these *resilient routing tables* since they embed failover information into the routing table itself and do not entail changes in packet headers (and so require no change in the low-level packet forwarding hardware). Because these failover decisions are purely local — based only on the packet's destination, the packet's incoming link, and the set of active incident links — they occur much more rapidly than the global recovery algorithms used in traditional routing protocols and thus result in many fewer packet losses.

While such resilient routing tables are widely used in practice (*e.g.,* ECMP), there has been little theoretical work on their inherent power and limitations. In this short note we initiate the theoretical study of resilient routing tables and take the first few steps in this research direction. We prove that routing tables can always provide resilience against single failures (so long as the network remains topologically connected). We show, in contrast, that perfect resilience is not achievable in general (*i.e.,* there are cases in which no set of routing tables can guarantee packet delivery even when the graph remains connected). We leave open the question of closing the large gap between our positive and negative results. Other interesting open questions include exploring resilient routing tables in the context of specific families of graphs, randomized forwarding rules, and more.

The literature is replete with discussions of how to make routing more resilient, but these approaches differ from ours in one or more important respects: (a) use bits in the packet headers to determine when to switch from primary to backup paths (this includes MPLS Fast Reroute [6]); (b) encode failure information in packet headers to allow nodes to make failure-aware forwarding decisions (see [3, 5] and work on fault-tolerant compact routing [7]); (c) use graph-specific properties to achieve resilience [2]; and (d) modify routing tables on the fly [4].

Because of space limitations, full proofs can be found in our technical report [1].

2. MODEL

The network is modeled as an undirected graph $G = (V, E)$, in which the vertex set V consists of source nodes $\{1, 2, \ldots, n\}$ and a *unique* destination node $d \notin [n]$. Each node $i \in [n]$ has a *forwarding function* $f_i^d : E_i \times 2^{E_i} \to E_i$, where E_i is the set of node i's incident edges. f_i^d maps incoming edges to outgoing edges as a function of which incident edges are up. We call an n-tuple of forwarding functions $f^d = (f_1^d, \ldots, f_n^d)$ a *forwarding pattern*.

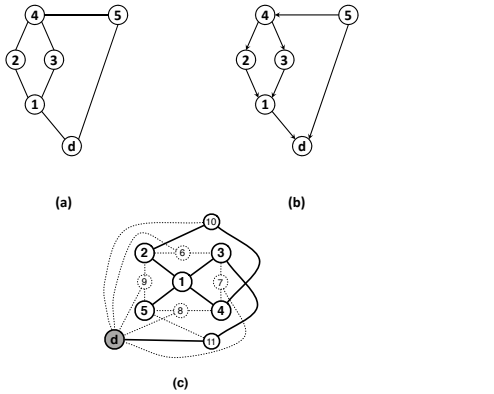
Consider the scenario that a set of edges $F \subseteq E$ fails. A *forwarding path* in this scenario is a loop-free route in the graph $H^F = (V, E \setminus F)$ such that for every two consecutive edges e_1, e_2 on the route which share a mutual node i it holds that $f_i^d(e_1, E_i \setminus F) = e_2$.

Intuitively, our aim is to guarantee that whenever a node

Copyright is held by the author/owner(s).
PODC'12, July 16–18, 2012, Madeira, Portugal.
ACM 978-1-4503-1450-3/12/07.

is connected to the destination d, it also has a forwarding path to the destination. Formally, we say that a forwarding pattern f is *t-resilient* if for every failure scenario $F \subseteq E$ such that $|F| \leq t$, if there exists some route from a node i to d in H^F then there also exists a forwarding path from i to d in H^F.

3. POSITIVE RESULT



We now present our main result, which establishes that for every given network it is possible to efficiently compute a 1-resilient forwarding pattern.

THEOREM 3.1. *For every network there exists a 1-resilient forwarding pattern and, moreover, such a forwarding pattern can be computed in polynomial time.*

We prove Theorem 3.1 constructively; we present an algorithm that efficiently computes a 1-resilient forwarding pattern. We now give an intuitive exposition of our algorithm. We first orient the edges in G so as to compute a directed acyclic graph (DAG) D in which each edge in E is utilized. Our results hold regardless of how the DAG D is computed. An example network and corresponding DAG appear in figures (a) and (b), respectively. The DAG D naturally induces forwarding rules at source nodes; each node's incoming edge in D is mapped to its first active outgoing edge in D, given some arbitrary order over the node's outgoing edges (*e.g.*, node 4 in the figure forwards traffic from node 5 to node 2 if the edge to 2 is up, and to node 3 otherwise).

Intuitively, the next step is to identify a "problematic" node, that is, a node that is bi-connected to the destination in G but not in the partial forwarding pattern computed thus far, and add forwarding rules so as to "fix" this situation. Once this is achieved, another problematic node is identified and fixed, and so on. Observe that nodes 1-4 in the figure are all problematic. Observe also that adding the two following forwarding rules fixes node 4 (*i.e.*, makes node 4 bi-connected to the destination in the forwarding pattern): (a) when both of node 4's outgoing edges in D are down, traffic reaching 4 from node 5 is sent back to 5; and (b) when node 5's direct edge to the destination is up, traffic reaching node 5 from node 4 is sent along this edge. Thus, the algorithm builds the forwarding functions at nodes gradually, as more and more forwarding rules are added to better the resilience of the forwarding pattern.

Implementing the above approach, though, requires care; the order in which problematic nodes are chosen, and the exact manner in which forwarding rules are fixed, are important. Intuitively, our algorithm goes over problematic nodes in the topological order $<_D$ induced by the DAG D (visiting problematic nodes closer to the destination in D first), and when fixing a problematic node i, forwarding rules are added until a minimal node in $<_D$ whose entire sub-DAG in D does not traverse i is reached. We prove that this scheme outputs the desired forwarding pattern in a computationally-efficient manner.

4. NEGATIVE RESULT

We say that a forwarding pattern f is *perfectly resilient* if it is ∞-resilient — so that regardless of the failure scenario $F \subseteq E$, if there exists some route from a node i to the destination d in H^F then there also exists a forwarding path from i to d in H^F. We next prove that forwarding patterns cannot always achieve perfect resilience.

THEOREM 4.1. *There exists a network for which no perfectly resilient forwarding pattern exists.*

We now present the intuition for the proof of Theorem. Consider the example network in figure (c). We show that after certain failures, no forwarding pattern on the original graph allows each surviving node in the destination's connected component to reach the destination. In figure (c), the surviving links are shown in bold; all other links fail.

We argue that in any perfectly resilient forwarding pattern f^d, node 1 has to route packets in some cyclic ordering of its neighbors. By the topology's symmetry, we can suppose w.l.o.g. that this ordering is 2, 3, 4, 5, 2, *i.e.*, f^d is defined such that 1 forwards packets from 2 to 3, packets from 3 to 4, *etc.* Note that a forwarding loop is formed when a packet repeats a directed edge in its path (rather than just a node). To show that this occurs, consider the path taken by packets sent by 5 after the failures. It can be shown that to achieve perfect resilience, packets sent $1 \rightarrow 2$ must not loop back and so must travel $2 \rightarrow 10 \rightarrow 4 \rightarrow 1$. As a result the packet travels $5 \rightarrow 1 \rightarrow 2 \rightarrow 10 \rightarrow 4 \rightarrow 1 \rightarrow 5 \rightarrow 1$ which is a loop since the edge $5 \rightarrow 1$ is repeated.

5. REFERENCES

[1] Joan Feigenbaum, P. Brighten Godfrey, Aurojit Panda, Michael Schapira, Scott Shenker, and Ankit Singla. Technical report YALEU/DCS/TR-1454. On the resilience of routing tables. 2012.

[2] Nate Kushman, Srikanth Kandula, Dina Katabi, and Bruce M. Maggs. R-BGP: Staying connected in a connected world. In *NSDI*, 2007.

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

[4] Junda Liu, Baohua Yan, Scott Shenker, and Michael Schapira. Data-driven network connectivity. In *HotNets*, 2011.

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

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

[7] Koichi Wada and Kimio Kawaguchi. Efficient fault-tolerant fixed routings on (k+1)-connected digraphs. *Discrete Applied Mathematics*, 1992.