On resilient routing

In this work we prove that each 3-connected graph has a 2-resilient routing scheme. We also present a counter-example of a 2-connected graph for which 2-resilient routing scheme does not exist. Therefore we provide more arguments for a conjecture that k-resilient static routing exists for all $(k+1)$-connected graphs but does not always exist for graphs with less connectivity.

1. DEFINITIONS

Static routing is a routing scheme in which decision how to forward packet is based only on destination address, inbound link and a set of *adjacent* failed links.

k-resilience is a property of static routing to guide a packet to the destination in case any k links failed such that there exists a path to destination.

A graph is called **k-connected** (or edge-k-connected) if it is impossible to make the graph disconnected by removing less than k edges.

2. 2-RESILIENCE FOR 3-CONNECTED GRAPHS

In this section we will consider a 3-connected undirected graph $G = \langle V, E \rangle$ with a single destination d. We will consider only one destination, as static routing may be represented by a set of separate routings for all destinations.

Gopalan and Ramasubramanian [**?**] showed that there exists 3 edge independent trees rooted at the node d. Each edge can belong to at most 2 trees (in different directions). From each node except d there is exactly one outgoing edge in each tree. All trees have d as a root. Edge independence property means that for each node s paths from it to d in all trees are edge-disjoint. Figure 1 shows an example of a 3-connected graph (clique with 4 nodes) with 3 edge-independent trees in it.

We can utilize these trees to construct 2-resilient routing. Let us call these trees Red, Blue and Black. Consider we already constructed trees. Therefore,

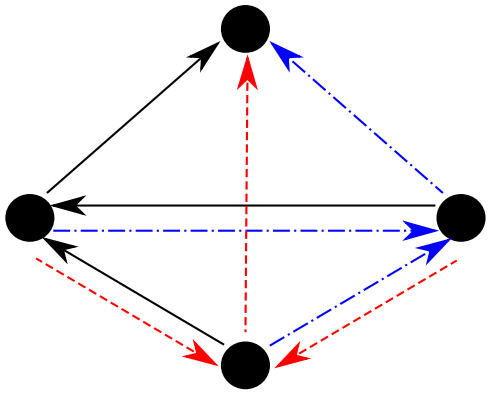


Figure 1: A 3-connected graph with marked three edge-independent trees rooted at the top node.

each directed link may be colored red, blue or black color (or be uncolored). Each node has an outgoing link of each color. For each node routing decisions will be as follows:

1. if node is the origin of a packet — use Blue outgoing link.
2. if node is a destination d — no routing needed.
3. if outgoing link of the same color as the inbound link has not failed — use it (i.e. continue to use the same tree).
4. if inbound link is Blue and outbound Blue link failed: Use the outgoing link of the same color as the reversed outgoing Blue link. If reverse link is uncolored, use any of the remaining two trees. If it is impossible (i.e., both outgoing links failed) — use the remaining outgoing edge.
5. if inbound link is Red or Black and outgoing link of the same color had failed: Use the third (non Blue and non inbound) tree.

The decisions are made only based on the inbound link and set of failed adjacent links. Therefore it is a static routing. Such routing will always guide packet to the destination d for any 2 links failures.

Proof: Packet is routed along Blue tree before it encounters a failure. It may reach destination just using a Blue tree, or encounter an error somewhere along the path. Let us denote the failed link as A.

Let us consider that outgoing Blue edge has reverse edge colored as Red (the same proof can be made for the other color). From now on, the packet will use the Red tree. It may reach a destination or encounter a failure. Because directed trees do not have any cycles, the packet cannot reach the link A again. Therefore that failure is in another link B. From now on, packet will be using the third — Black tree. Because there are no cycles in directed trees, packet will never reach link B. Because link A is colored in Blue and Red, packet can not reach it either. Therefore, it will reach the destination. If while using the Blue tree a packet encounters two failed outgoing links, it can use the third colored outgoing link. Because there are no cycles, the remaining path will not traverse any of the two failed links.

3. 1-RESILIENCE

It was proven in [**?**] that for each network there exists a 1-resilient routing. However, there is an easier proof for 1-resilience using independent trees.

Using 2 edge-independent trees [**?**] we can easily construct 1-resilient routing for 2-connected graphs. Each node has at least one outgoing edge belonging to each tree. Packet should be routed along the first tree unless it reaches failed link. From that point it should be routed using the second tree. Because each directed link belongs to only one tree, it is possible to infer the tree the packet is following without any additional information.

Because each 1-connected graph can be represented as a tree made of 2-connected components, we can construct 1-resilient routing for every graph. Each component has one special node such that all paths to the destination have to traverse that node. Therefore we can construct resilient routing in each 2-connected component separately and add special rules for such special nodes — they always have to forward packet toward next 2-connected component if it is possible. If corresponding link had failed — graph is no longer connected and nothing can be done in that case.

4. COUNTEREXAMPLE: 2-CONNECTED GRAPH WITHOUT 2-RESILIENT ROUTING

The 2-connected graph without 2-resilient routing is presented in figure 2a. There are many graphs with such property, but presented is one of the smallest. In this section we will prove analytically that no static routing scheme will be 2-resilient.

The destination is the node 1. The nodes 2, 3, 4, and 5 all have 3 links with other nodes. Only these nodes need to make some routing decisions because all other nodes have at most 2 links and routing is trivial for them. Small dots on the picture are intermediate nodes. They are needed to simplify the proof. We will consider only failures between intermediate nodes thus a routing scheme for each node will be just a table with three entries: Where to send packet arrived through each of the three links.

Let $f(e_i) \in e1, e2, e3$ be the routing table for some node, e_i for $i = 1, 2, 3$ be adjacent links for a given node. Firstly, $f(e_i) \neq e_i$, i.e. no node bounce a packet back to the link it arrived from. Because a packet will loop in the link e_i in case that link had a dead-end ahead created by a link failure between two intermediate nodes (small dots in figure 2a). Secondly, all entries in the routing table are distinct. Otherwise, if $f(e_1) = f(e_2) = e_3$ then two dead-ends ahead of links e_3 and $f(e_3)$ will create a loop for a packet originating from the other end of the link e_3. Because there is a dead-end, it has no other way but to go to the current node. Then it will go to $f(e_3)$ which is either e_1 or e_2. Because there is a dead-end a packet will return to the node and then go back to the e_3 where it will continue to loop.

Therefore, each of the four key nodes will have to do routing in a round manner — all 3 links will have some order and packets incoming using one link will leave the node using next link in the order. There are only two cyclic permutations for 3 elements, therefore there are only 16 possible routing schemes for that graph and we will show that all of them have loops.

First, we will consider nodes 3 and 5. There are 4 routing schemes for these two nodes. But because of graph symmetry there are only 2 distinct among them. In one nodes 3 and 5 have same (clockwise) order of edges. In the second routing they have different orders of edges. If they have different order then two link failures like shown in figure 2b can create a loop for packets originating on the first intermediate node between nodes 3 and 2. That packet will visit node 3. After that it will be routed towards node 4. After that regardless of ordering at node 4 packet will go to node 5. In that node it will be routed back to node 3. From node 3 packet will be routed toward node 2, therefore, closing the loop.

If there exists a 2-resilient routing then nodes 3 and 5 have the same order of edges in their routing tables. Now we will consider the case there these nodes both have clockwise order. Another option can be omitted due to graph symmetry. There are 4 routing schemes for different ordering at nodes 2 and 4. All routings and corresponding loops with needed link failures are shown in figures 2c-2f

Therefore, we have shown that there are no static routing scheme which will be 2-resilient. Note, that given proof assumes there may be up to two simultaneous link failures to argue that routing at each node should be in a round manner. Thus it does not contradict with existence of 1-resilient routing scheme for any graph.

5. 3-RESILIENCE IN A GRID

In this section we present a 3-resilient routing for a grid network. We consider a rectangular grid with additional links wrapped around borders (see Fig. 3). This is a 4-connected graph, thus according to conjecture we expect 3-resilience.

Our routing scheme relies on graph decomposition to 2 edge-independent Hamiltonian cycles. Such decomposition always exists. We provide patterns for different parity of grid dimensions which are extendible by adding two rows or columns for such decomposition. Figure 3 shows all possible parity cases. Two cycles are marked with different line types. Repeatable blocks are highlighted with curve brackets.

The routing will be as following. Each cycle is directed in either way. Then packet starts to traverse the first cycle along the direction. If packet encounters a failed link it should start to traverse the same cycle backwards. If the packet encounters a second failed link it should start to traverse the second cycle along the direction. If packet encounters a third failed link it should start to traverse the same cycle backwards. If at any time a packet arrives to the destination it stops to route. Note that each node has exactly 4 links: inbound and outbound links in both cycles. Therefore it is always possible to switch to the second cycle. Because cycles are edge-independent, a packet will never encounter the same failed link twice. Therefore for any 3 failed links any packet will be guided to the destination using proposed routing scheme.

6. PROBABILISTIC ROUTING

We present a following generalization of static routing in which in all nodes there are probabilities of forwarding packet along each outgoing link that depend on an inbound link, destination address and a set of failed adjacent links. Each static routing is a special case of probabilistic routing in which for each case all probabilities except one choice are zero.

Table 1 shows notations used in this section.

Notation	Comment
V	set of nodes
E	set of edges
$F \subset 2^E$	set of such link failures that the graph is still connected and there are no more than K link failures
$N_v^s \subset V$	set of adjacent nodes for node v not counting edges from set $s \in E$
$E_v \subset E$	set of edges adjacent to node v
S	set of all situations where a routing decision is made. $S = \{(u, e, T) \vert u \in V, e \in E_u, T \subset E_u\}$

Table 1: Notations employed in this section

The deterministic routing can be formulated as a function $w : (u, e, T) \in S \to N_v^T$, i.e., for each node, an inbound link and a set of adjacent failed links there is a neighbor node to which a packet should be routed. There should also be initial step decisions $w' : (u, T) \to N_v^T$.

The probabilistic routing can be formulated as a set of probabilities: $P_{u,e,T,v}$ — a probability to send a packet to node v from node u if it has arrived using link $e \in E_u$ and links from set $T \subset E_u$ failed. These variables should comply to the following restrictions:

$$\forall (u, e, T) \in S, v \in N_v^T : P_{u,e,T,v} \geq 0$$

$$\forall (u, e, T) \in S : \sum_{v \in N_v^T} P_{u,e,T,v} = 1$$

Let $x_{u,v}^c$ be the expected number of hops before a packet forwarded from node $u \in V$ to node $v \in V$ will be delivered to destination $d \in V$ if links from the set $c \in F$ have failed. The following equations for x can be formulated:

$$\forall c \in F, e = (u, v) \in E/c, v = d : x_{u,v}^c = 1$$

$$\forall c \in F, e = (u, v) \in E/c, v \neq d :$$

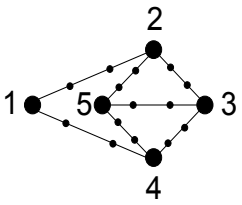
$$x_{u,v}^c = 1 + \sum_{t \in N_u^c} x_{v,t}^c P_{u,(u,v),c \cap E_u,t}$$

Then we can introduce an initial step probabilities $Q_{u,T,v}$ — probability to send a packet originated at node u towards node $v \in N_u^T$ if links from set $T \subset E_u$ have failed. These variables comply to following restrictions:

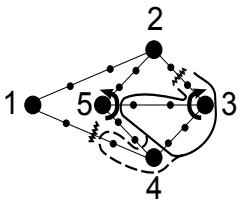
$$\forall u \in V, T \subset E_u, v \in N_u^T : Q_{u,T,v} \geq 0$$

$$\forall u \in V, T \subset E_v : \sum_{v \in N_u^T} Q_{u,T,v} = 1$$

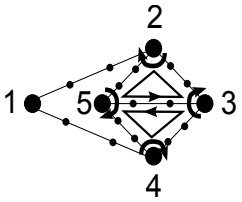
Now we can define an expected delivery time for



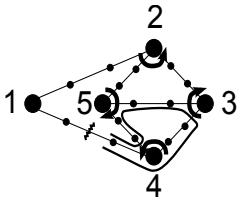
(a) Graph



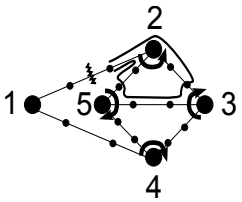
(b) Regardless of ordering at node 4 there is a loop starting near node 3



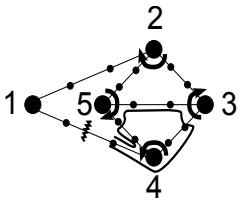
(c) Packet will loop if it starts on path between nodes 5 and 3 regardless of its direction



(d) Packet starting near node 4 will always loop

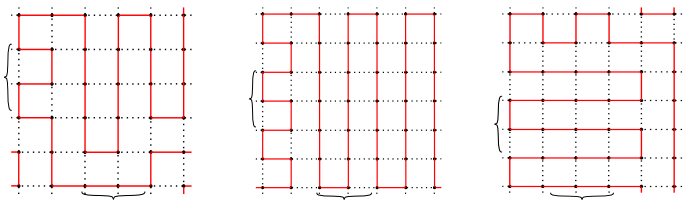


(e) Loop starts near node 2



(f) Loop starts near node 4

Figure 2: Counterexample graph with different possible orderings at nodes and corresponding loop paths



(a) Odd-Odd (b) Even-Even (c) Even-Odd

Figure 3: Grid decomposition patterns to 2 Hamiltonian cycles for Odd-Odd, Even-Even and Even-Odd grid graphs

packets originating in node u for a failure set c:

$$H_u^c = \sum_{(u,v) \in N_u^c} Q_{u,c \cap E_u,v} \times X_{u,v}$$

Given decision probabilities Q and P we can calculate H for all nodes and failure sets solving system of linear equations to find X and use it in the last equation.

After that we can formulate an optimization problem to find suitable probabilistic routing for given network:

$$minimize : \sum_{u \in V, c \in F} \left(H_u^c\right)^3$$

We do not simply take average of all expected delivery times, as we want to give more weight for worst case, thus optimizing not only average performance but also a worst case.

We used gradient descent and local optimizations method to find best routings for grid networks of sizes 3×3, 5×3, 5×5 and 3×7. It is still computation hard problem as we have to compute H for all possible failure sets which is $O(|E|^3)$ for grid networks.