Packet Re-cycling: Eliminating Packet Losses due to Network Failures

Suksant Sae Lor
s.lor@ee.ucl.ac.uk

Raul Landa
r.landa@ee.ucl.ac.uk

Miguel Rio
m.rio@ee.ucl.ac.uk

Network and Services Research Laboratory
Department of Electronic and Electrical Engineering
University College London, UK

ABSTRACT

This paper presents Packet Re-cycling (PR), a technique that takes advantage of cellular graph embeddings to reroute packets that would otherwise be dropped in case of link or node failures. The technique employs only one bit in the packet header to cover any single link failures, and in the order of $\log_2(d)$ bits to cover all non-disconnecting failure combinations, where d is the diameter of the network. We show that our routing strategy is effective and that its path length stretch is acceptable for realistic topologies. The packet header overhead incurred by PR is very small, and the extra memory and packet processing time required to implement it at each router are insignificant. This makes PR suitable for loss-sensitive, mission-critical network applications.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols—*Routing protocols*; C.2.6 [**Computer-Communication Networks**]: Internetworking

General Terms

Algorithms, Design, Reliability

Keywords

Internet routing, transient failures, fast re-routing

1. INTRODUCTION AND MOTIVATION

Although resilience to link (and node) failures has been programmed in the "DNA" of the Internet, service reliability has not improved proportionally to the levels required by future and some current applications. Many mission critical applications in security, transport and health, along with "five nines" telephony, require better resilience than that provided by the current Internet routing convergence process. In some cases, routing reconvergence may require minutes in order to conclude. During this period, large numbers of packets may be dropped. If, for instance, a heavily loaded OC-192 link is down for a second, more than a quarter of a million packets could be lost, given an average packet size of 1 kB. Moreover, intermittent failures can trigger multiple reconvergence episodes, which lead to routing instability.

Many approaches have been proposed to overcome the routing disruption problem in IP networks, including IP fast reroute (IPFRR) [2, 4, 6, 7, 9, 10, 11], multipath and multihomed routing [15, 19, 20], and resilient overlays (RON) [1]. Many of these techniques handle only single failures; very few address dual link failures. Although a resilient routing strategy using Failure-Carrying Packets (FCP) [8] has been proposed to allow full recovery from any number of link failures, the technique requires considerable overhead on each packet header, as well as nontrivial computation resources on each router (which must re-compute the shortest path available to a given destination for each failure combination it receives within a failure carrying packet).

Path repair mechanisms under the IPFRR framework [17] perform precomputation of the alternate paths used to reroute packets. Thus, once a failure is detected, rerouting can be performed locally without any additional signalling. Similarly, in this paper we propose a novel contingent forwarding approach which combines traditional routing with a Packet Re-cycling (PR) mechanism. The technique allows normal routing operations in failure-free scenarios, and rerouting under failure cases. However, unlike other fast reroute mechanisms, PR can guarantee full repair coverage for any number of failures, as long as the network remains connected.

The rest of this paper is organised as follows. Section 2 provides the overview of our rerouting strategy. We describe cellular graph embeddings, on which our technique is based, in Section 3. We present the protocol in Section 4, and arguments for its correctness in Section 5. Section 6 illustrates preliminary results of PR. Various issues of PR are discussed in Section 7. We conclude the paper in Section 8.

2. PACKET RE-CYCLING OVERVIEW

Packet Re-cycling (PR) is a novel fast reroute approach that extends routing with a recovery protocol that, in case of failure, reroutes packets along precomputed backup paths. What distinguishes PR from other schemes is the definition of these paths. Essentially, PR relies on a system of cycles where each unidirectional link in the network is associated with a unidirectional cycle that can be used to bypass it if it fails, through a process called *cycle following*. This is done in such a way that, if further failures are encountered in links along the backup path, the backup paths of these links are guaranteed to avoid previously encountered failures, for all failure cases where a path still exists between the source and the destination. This is achieved without including failure information in the packet header, or recalculating routing tables in real time.

Each PR-enabled router initialises the protocol by constructing its routing table using a conventional shortest path algorithm (*e.g.* Dijkstra's algorithm). The tables resulting from this process allow forwarding during failure-free conditions, and do not impose any additional requirements on the current routing paradigm. To take advantage of PR, routers implement a *cycle following table*, an additional information repository that is used to forward packets along their backup paths during the cycle following process. The additional information required to populate the cycle following table is obtained from the *cellular embedding* of the network graph, which is done offline prior to the initialisation of the protocol, and then installed in all routers in the network. We now discuss the useful properties of cellular graph embeddings on which PR is based.

3. CELLULAR GRAPH EMBEDDINGS

An *embedding* of a graph $\mathcal{G}$ on a closed surface $\mathcal{S}$ is a way of drawing $\mathcal{G}$ on $\mathcal{S}$ so that no two distinct nodes coincide and there are no edge crossings. This means that, in an embedding, links become *lines* on $\mathcal{S}$ which only meet at the nodes, that become *points* on $\mathcal{S}$. Thus, an embedding of $\mathcal{G}$ in $\mathcal{S}$ is a way of arranging the nodes and links of $\mathcal{G}$ in space so that all of them lie on $\mathcal{S}$ and no link crossings occur.

A particular kind of embedding is specially useful to us: the *minimum genus embedding* [14]. This is because it provides a *cellular cycle system*, a system of cycles in $\mathcal{G}$ so that every link is included in exactly two cycles. This kind of embedding of $\mathcal{G}$ on $\mathcal{S}$ has the property that the *cells* (the areas over $\mathcal{S}$ that are delimited by each cycle in the system) are topologically equivalent to open discs. Furthermore, since $\mathcal{S}$ is fully covered by them, they form a *partition* of $\mathcal{S}$ (a set of maximal, connected subsets).

We shall restrict our attention to *orientable* surfaces, such as the sphere or the torus. A property of these surfaces is that we can assign an *orientation* to our cellular cycle system in a consistent manner (*clockwise* or *anti-clockwise*), and when this is done, each link will be associated with two cycles, each one traversing the link in opposite directions. This will allow us to associate, for data transmission over each link and in any particular direction, a *main cycle* that represents the direction of data flow in failure-free conditions and a *complementary cycle*, in the opposite direction, that can be used as a backup if the link has failed. As we shall see in Section 4.2, this allows our protocol to systematically avoid failures in much the same way that the right-hand rule allows the solution of labyrinths.

4. PACKET RE-CYCLING

We now describe the operation of PR. When considering failure coverage, we assume that failures are bidirectional.

4.1 Routing and Cycle Following Tables

The cycle following table of a router is a three-column table with i entries, where i is the number of the interfaces in the router. We use the network illustrated in Figure 1(a) as an example for the construction of cycle following tables. In this case, node D has three interfaces and hence, its local cycle following table has three entries. The first column indicates the incoming interface for each entry, while the second and third columns store next hop information that enables forwarding along backup paths. In particular, the second column stores the outgoing interface under cycle following, while the third column stores the outgoing interface under failure avoidance. For reasons that will become clear in Section 4.2, this last column stores the next hop over the *complementary cycle* of the cycle defined by the first two columns. We now clarify this with an example. Let I_{YX} represent an interface at node X receiving packets from node Y. Table reftable:cycling shows the cycle following table at D.

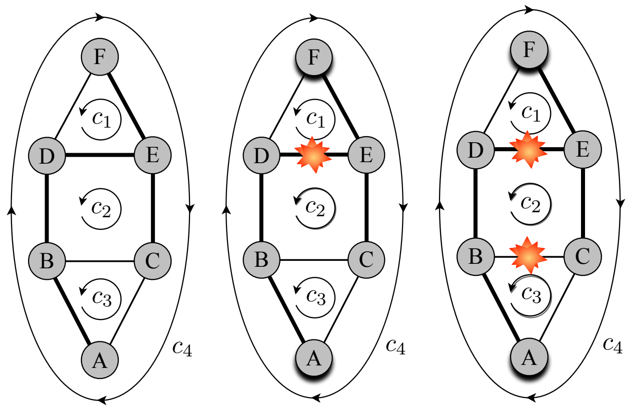
Table 1: Cycle following table at node D.

Interfaces		
Incoming	**Cycle Following**	**Complementary**
I_{BD}	I_{DF} (c_4)	I_{DE} (c_1)
I_{ED}	I_{DB} (c_2)	I_{DF} (c_4)
I_{FD}	I_{DE} (c_1)	I_{DB} (c_2)

The cycle following table is constructed based on the oriented embedding of the network graph. For example, in Table 1, the outgoing interface I_{DE} corresponds, in positive orientation, to cycle c_1 and, in negative orientation, to cycle c_2. This means that, if we want a packet entering D through I_{FD} to follow its positively oriented cycle c_1, we would need the packet to be forwarded to interface I_{DE}. To determine the complementary outgoing interface, we note that cycle c_1 is complementary to cycle c_2 over the link implied by the outgoing interface I_{DE}. Thus, in this case the complementary cycle is c_2 and the value installed in the complementary outgoing interface of the cycle following table is I_{DB}, the next hop interface over c_2 from D. In the same way, a packet entering D through I_{ED} should be forwarded using I_{DB} in order to follow its main cycle c_2, and through I_{DF} to follow c_4, its complementary cycle[1].

As is clear from the preceding discussion, the first two columns of the cycle following table can be used to forward

[1]It may seem that cycle c_4 has opposite orientation to the other cellular cycles; this is an artifact of the stereographic projection used to represent on the plane what is really an embedding on the sphere. It can be trivially verified that in Figure 1(a) each link belongs to exactly two cycles, each one flowing in opposing direction, thus satisfying the conditions for a cellular embedding.



(a) Embedding (b) Single failure (c) Multiple failures

Figure 1: Different failure scenarios.

packets along the cellular cycles of the network, and essentially are an implementation for the *rotation system* induced by the cellular embedding [14] (note that the forwarding table is a permutation over the output interfaces). We now explain how we use the output interface on the third column to define alternate routes if the corresponding outgoing interface in the second column has failed.

4.2 The Cycle Following Protocol

Once routing and cycle following tables are constructed, PR can guarantee full failure recovery from any single link failure in 2-connected networks without any additional information, and requiring a single bit on the packet header. To achieve this, routers forward packets normally according to the routing table, until a failure is detected. At that point, the detecting router marks packets which were to be routed along the failed link with a single bit (the *PR bit*) to indicate that they must be forwarded using cycle following tables instead of routing tables, and will forward them along the complementary interface associated with the failed outgoing interface. Routers receiving packets with the *PR bit* set will forward them as indicated by the cycle following interface associated with their ingress interface. Since this will, by design, forward packets along a cellular cycle (in this particular case, the complementary cycle of the failed link), they will eventually reach the router at the other side of the failed link. When that router attempts to send these packets over the failed link again to continue cycle following, the failure will be encountered one more time, and this will be interpreted as a signal that cycle following is no longer necessary, and normal shortest path routing can resume (this will be the first *termination condition* presented in this paper). Thus, these packets will resume shortest path routing after the failed link, and they will not encounter the failure again.

Figure 1(b) illustrates this single failure scenario. Let node A send a packet with node F as the destination, with the shortest path tree from all other nodes to F marked with thicker edges. Under PR, the packet would be forwarded along $A \to B$ and $B \to D$, as implied by the shortest path tree. Normally, D would then attempt to forward the packet to node E through interface I_{DE}. However, since link $D \to E$ is down, node D sets the *PR bit* in the packet and forwards it to the complementary outgoing interface of I_{DE}, which is I_{DB}. This initiates the cycle following process. When routers B and C receive the packet with the *PR bit* set, they will forward it using their normal cycle following tables, so that it follows cycle c_2, the complementary cycle of c_1 along the failed link. Once the packet arrives at node E and encounters the failure from the other side, the *PR bit* is cleared and the packet forwarded to node F via the conventional shortest path.

It is interesting to note that even this simple scheme can protect a network from specific instances of multiple link failures. If, for instance, we had failures not only on link $D \to E$ but also on link $A \to B$, shortest path forwarding would fail at $A \to B$ and thus packets would first follow cycle c_3 (complementary to c_4 over $A \to B$) to reach B, where normal routing would resume - only to fail again in D, due to $D \to E$ being down. From here, protocol recovery is identical to the previous example.

4.3 The Decreasing Distance Termination Condition

We now describe how the cycle following protocol of Section 4.2 can be improved, so that packets can be reliably protected from any number of link failures in any combination, as long as a path exists between the source and destination nodes. First, we note that the simple protocol described in Section 4.2 would, by itself, lead to forwarding loops. We use Figure 1(c) as an example. Again, let A be the source and F the destination. When node D detects a failed link, $D \to E$, it will send the packet along the complementary cycle of I_{DE}, following c_2 back through $D \to B$ and $B \to C$. However, without additional logic, when the packet encounters the second failure at $B \to C$, it will be routed according to the shortest path and will encounter the failed link $D \to E$ again, creating a forwarding loop.

In order to solve this problem, we enrich PR by adding an additional column in the routing table that stores a strictly increasing function of the links along the shortest path to each existing destination. We will call the value of this function the *distance discriminator*. Candidates for this function include the the number of hops to the destination or the sum of the link weights, both along the shortest path.

We then update the protocol with a more detailed termination condition, as follows. The first router that detects a failure will set the *PR bit* as before, but in addition, it will mark the packet header with the distance discriminator to the destination, as calculated by the router behind the link failure. We call the bits necessary to encode this number the *DD bits*. When a packet encounters further failures while cycle following, each failure-detecting router will compare the distance discriminator from itself to the destination with that one encoded in the *DD bits*. If its own is smaller, it will clear the *PR bit* and route along the shortest path. If its distance discriminator is larger or equal, it will forward the packet along the complementary cycle of the failed interface.

This mechanism eliminates the aforementioned forwarding loop issue. We present an scenario based on Figure 1(c), and using the number of hops to the destination as a distance discriminator. When node D detects $D \rightarrow E$ as down, it will set the *PR bit* and set 2 as the value of the *DD bits*. When node B finds itself unable to forward over $B \rightarrow C$, it will compare its own distance discriminator to the destination node F with the *DD bits*, thus determining that the packet must not be forwarded using conventional routing tables, but cycle following tables. Thus, B will forward the packet over I_{BA} following c_3, which will bring it to C after being forwarded by A. When C tries to forward the packet though I_{CB}, the termination condition will be triggered, and C will compare its own distance to F with the *DD bits*, thus determining that cycle following must continue. The packet will therefore follow c_2, where E will once again trigger termination criteria logic when it tries to forward the packet along I_{ED}. This time, however, when E compares its own distance to F with that one encoded in the *DD bits*, it will clear the *PR bit* and forward it along the shortest path.

In PR, the network embedding is computed offline, on a server designated for that purpose. Once it is available, appropriate cycle following tables are uploaded to all routers. As long as this process concludes correctly, all nodes will forward packets consistently under both normal and failure scenarios. The network embedding (with its corresponding cycle following tables) needs to be recomputed only when the network topology experiences a long-term change, such as when new links are introduced.

5. PROTOCOL PROPERTIES

In this section we present informal arguments for the correctness of the protocol presented in Section 4.3. We will proceed from the definitions presented in Section 3. First, we consider the effects of applying the cycle following protocol of Section 4.2 with no termination criteria. Then, we posit that the termination criteria presented in Section 4.3 are sufficient to ensure termination. Finally, we use these two ideas to argue that PR delivers packets to their destinations if a route exists.

5.1 Cycle Following Properties

We consider a network graph that has been cellularly embedded on an orientable, closed surface $\mathcal{S}$ using a cellular cycle system. As detailed in Section 3, the cellular embedding of $\mathcal{G}$ guarantees that each link l will either separate two different cells, or separate a single cell that is "curved", so that it meets itself along l (in this case, the main cycle and its complement are the same). Of course, this also applies to arbitrary compact regions over $\mathcal{S}$.

It will be useful to draw parallels between topological operations involving regions in $\mathcal{S}$ (and their boundaries) and the behaviour of the cycle following protocol. Thus, we propose a *join* operation, to be performed on regions in $\mathcal{S}$ that share a link along their boundaries. A join performed between two of these regions will consist of removing this shared link, taking the resultant connected region as the result of the operation. Then, we have that when a packet encounters a link failure, the path it will follow under the guidance of the cycle following protocol with no termination conditions coincides with a boundary component of the region obtained by joining all cells in $\mathcal{S}$ that have at least one failed link on their boundaries. Examples of this can be found in Figures 1(b) and 1(c), where the route that a packet follows after encountering failures corresponds to the boundary of a region constructed by joining all the cells involved in failures: c_2 in Figure 1(b), and the boundary of a region including c_2 and c_3 in Figure 1(c). These cases are trivial, since all regions are simply connected; [12] provides a proof for the general case. We now make use of this idea to explore the conditions under which the protocol terminates.

5.2 Termination Properties

The cycle following algorithm of Section 4.2 induces continuous looping in the network; it is only with explicit termination conditions that this can be prevented. Of course, the conditions proposed in Section 4.3 are clearly sufficient for those failure episodes involving a single link. For episodes involving many links, we need to consider the route that packets will take as a result of these failures, and whether the termination conditions are sufficient in these cases. It can be proved that this is so [12]. The reason for this is that, when generating new regions by joining cells, the result will always be a set of disconnected regions surrounding those nodes and links inaccessible to any given packet source. Then, if a curve following the route over the shortest path tree to the destination encounters one of these regions, it must cross its boundary at least *twice*: once going in, and once going out (otherwise, either the source or the destination would lie within the inaccessible region, implying that there is no available path between them and thus no recovery is possible). By definition, the intersection point going out will have a lower distance discriminator than the intersection point going in, which is where the *PR* and *DD bits* were initially set. Since the packets will follow the boundary of the region, the protocol is guaranteed to terminate at that intersection point.

5.3 Forwarding Loop Resolution

We now argue that the argument presented in Section 5.2 implies that the termination criteria of Section 4.3 are sufficient to ensure that packets do not loop continuously. To this end, we note that the progress of a packet through the network is characterised as a set of intercalated episodes of conventional routing and cycle following. By definition, the distance discriminator decreases with each hop when routing. Furthermore, we know that cycle following episodes will terminate, and they will always do so in nodes with lower distance discriminators than that where they started. Thus, since these two kinds of forwarding can only decrease the distance discriminator, which is itself finite (we consider only connected networks with finite weights), the destination will eventually be reached in a finite number of steps.

6. PRELIMINARY RESULTS

We compare the operation of PR with that of alternative schemes, based on incurred overheads and path length stretch. FCP and full routing protocol reconvergence are used as benchmarks, since they are among the few techniques that can handle multiple failures. We evaluate PR using a Java-based simulator in three different ISP topologies: Abilene [21], Teleglobe [18] and Géant [5] .

Routing using FCP requires on-demand computation at nodes, while traditional reconvergence requires, in addition, the flooding of failure information throughout the network in order to maintain routing consistency. Although FCP can reduce its computational overhead by requiring routers to maintain per-flow routing state, computation of new routes when an FCP arrives at each router is unavoidable. PR, on the other hand, requires the offline computation of the cellular graph embedding. However, it does not require any additional computational overhead after this initialisation. The amount of memory that PR requires within each router (a cycle following table and an additional column in the routing table) is acceptable, as other resilient approaches with precomputation have similar requirements.

Regarding the overheads in each packet header, FCP employs more bits in the packet header than are currently available, making its deployment difficult. Our routing strategy requires a single *PR bit* to indicate the forwarding mechanism to use, and enough *DD bits* to store the distance discriminator from each point of failure to each given destination (if we use the number of hops, we require in the order of $\log_2(d)$ bits, where d is the network diameter). We suggest the use of the set of bits available within pool 2 of the DSCP field, reserved for experimental or local use [16].

Consistently with prior work, we define the *stretch* of a path as the ratio between the total path cost while cycle following and the path cost of the normal shortest path. Figure 2 illustrates the stretch of Abilene, Teleglobe and Géant under different failure scenarios. As PR trades off path length for reliability, path stretch is usually higher than that achieved with FCP. However, since this higher path length stretch is the only price to be paid at forwarding time for full failure protection using very few bits on the packet header, no real-time computations and only very limited memory requirements on routers, we think that PR presents an attractive alternative to other other techniques for full-failure protection.

7. DISCUSSION

PR is an engineering solution that enables network providers and equipment manufacturers to perform several trade-offs. By performing relatively expensive computations offline, PR releases routers from real-time route recalculation when failures occur, and by providing an ordered basis for the exploration of backup paths, it allows full failure protection through increased stretch for the saved packets. Overall, PR can be of great usefulness in many IPv4/IPv6 deployment scenarios, particularly because its exceedingly modest requirements in term of packet header space, which might be very useful in cases were such space is restricted or the use of IP options is difficult. Depending on the desired deployment strategy, ISPs can include extra rules and policies to limit PR to certain types of traffic (for example by limiting it to certain classes identifiable by the remaining DSCP bits).

Although PR is designed to offer intra-domain routing resilience, we are working on extending this approach to prefixes outside the boundaries of the ISP announced through BGP. Multihomed ISPs that receive several announcements for the same prefix via different outgoing links can map this onto a connectivity graph, and use our technique to obtain cycle following routes.

As with all alternate forwarding schemes, PR must cater for the possibility of *link flapping*. This can be done simply by ensuring that link state transitions only happen after the link has been idle for long enough to ensure that packets that encountered the link in its *failed* state do not encounter it again in its *normal* state while cycle following.

Regarding embedding complexity, the general case is NP-hard [14]. However, linear time algorithms exist for graph embedding on surfaces of known genus [13], which may provide useful 2-cell embeddings for arbitrary networks at the cost of increased stretch. In the case of planar graphs, very efficient $O(n)$ algorithms are available [3]. We leave a detailed analysis of the implementation issues of network embedding algorithms for future work.

8. CONCLUSION

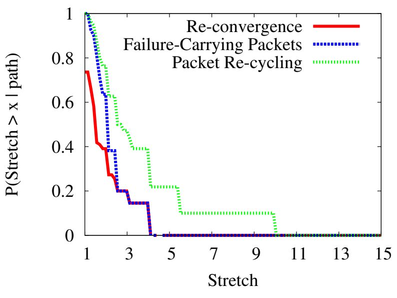
In this paper we proposed PR, a contingent forwarding technique that takes advantage of a cellular graph embedding for fast packet rerouting in the event of link failures. Our strategy requires less overheads than alternative proposals, while providing full protection against any number of failures. In particular, PR requires a very small number of bits in the packet header, very limited memory on routers, and no real-time recalculations. We compared the path length stretch of PR with those of FCP and normal reconvergence, showing it to be tolerable for many realistic deployments.

9. REFERENCES

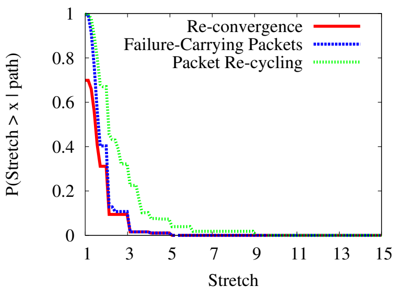
[1] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. Resilient Overlay Network. In *ACM SOSP*, 2001.

[2] A. Atlas and A. Zinin. Basic specification for IP fast reroute Loop-Free Alternates. RFC 5286, 2008.

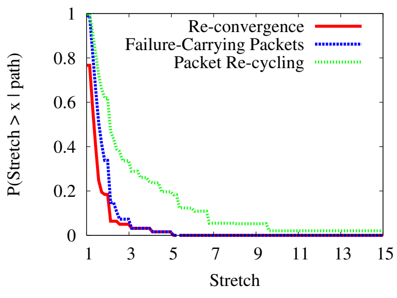
[3] J. M. Boyer and W. J. Myrvold. On the cutting edge: Simplified $O(n)$ planarity by edge addition. *Journal of*



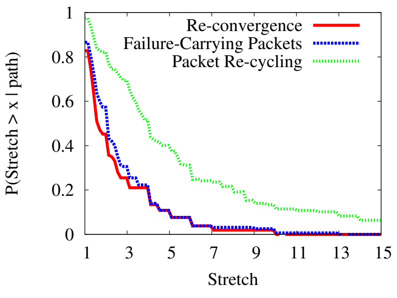
(a) Abilene with single failures



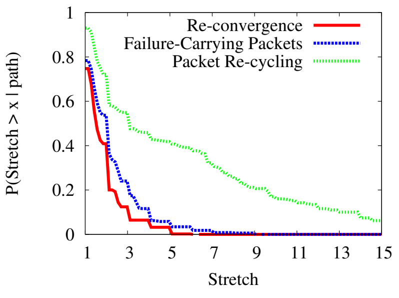
(b) Teleglobe with single failures



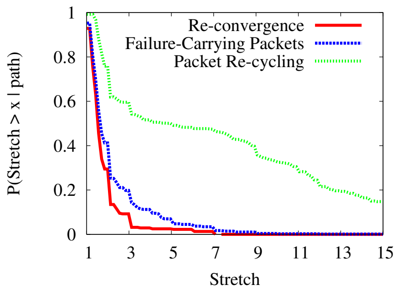
(c) Géant with single failures



(d) Abilene with 4 failures



(e) Teleglobe with 10 failures



(f) Géant with 16 failures

Figure 2: Stretch comparison between reconvergence, FCP and PR.

Graph Algorithms and Applications, 8:2004, 2004.

[4] S. Bryant, M. Shand, and S. Previdi. IP fast reroute using not-via addresses. IETF Internet draft, 2010.

[5] Géant. Network topology. Online, 2009. `http://www.geant.net/Network/NetworkTopology/pages/home.aspx`.

[6] S. Kini, S. Ramasubramanian, A. Kvalbein, and A. F. Hansen. Fast recovery from dual link failures in IP networks. In *IEEE INFOCOM*, 2009.

[7] A. Kvalbein, A. F. Hansen, T. Cicic, S. Gjessing, and O. Lysne. Fast IP network recovery using multiple routing configurations. In *IEEE INFOCOM*, 2006.

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

[9] S. Lee, Y. Yu, S. Nelakuditi, Z.-L. Zhang, and C.-N. Chuah. Proactive vs. reactive approaches to failure resilient routing. In *IEEE INFOCOM*, 2004.

[10] A. Li, X. Yang, and D. Wetherall. SafeGuard: safe forwarding during route changes. In *ACM CoNEXT*, 2009.

[11] S. S. Lor, R. Landa, R. Ali, and M. Rio. Handling transient link failures using Alternate Next Hop Counters. In *IFIP Networking*, 2010.

[12] S. S. Lor, R. Landa, and M. Rio. Packet re-cycling: Eliminating packet losses due to network failures (extended version). Online, 2010. `http://www.ee.ucl.ac.uk/~mrio/papers/hotnets2010extended.pdf`.

[13] B. Mohar. A linear time algorithm for embedding graphs in an arbitrary surface. *SIAM J. Discret. Math.*, 12(1):6–26, 1999.

[14] B. Mohar and C. Thomassen. *Graphs on Surfaces*. The Johns Hopkins University Press, 2001.

[15] M. Motiwala, M. Elmore, N. Feamster, and S. Vempala. Path splicing. In *ACM SIGCOMM*, 2008.

[16] K. Nichols, S. Blake, F. Baker, and D. Black. Definition of the differentiated services filed (DS field) in the IPv4 and IPv6 headers. RFC 2474, 1998.

[17] M. Shand and S. Bryant. IP fast reroute framework. RFC 5714, 2010.

[18] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson. Measuring ISP topologies with Rocketfuel. *IEEE/ACM ToN*, 12(1):2–16, 2004.

[19] W. Xu and J. Rexford. MIRO: multi-path interdomain routing. In *ACM SIGCOMM*, 2006.

[20] X. Yang and D. Wetherall. Source selectable path diversity via routing deflections. In *ACM SIGCOMM*, 2006.

[21] Y. Zhang. The Abilene topology and traffic matrices. Online, 2004. `http://www.cs.utexas.edu/~yzhang/research/AbileneTM/`.