Recursive Loop-Free Alternates for Full Protection Against Transient Link Failures

Suksant Sae Lor, Redouane Ali, Raul Landa, and Miguel Rio
Department of Electronic & Electrical Engineering
University College London, UK
Email: {s.lor, r.ali, r.landa, m.rio}@ee.ucl.ac.uk

***Abstract*—In this paper, we propose a routing technique, "recursive Loop-Free Alternates (rLFAs)", to alleviate packet loss due to transient link failures. The technique consists of a backup path calculation with corresponding re-routing scheme based on the Loop-Free Condition (LFC) as defined in the basic specification for IP Fast Re-Route (IPFRR). Under this routing strategy, nodes calculate backup paths by modifying the weights of links in the primary shortest path tree. If a failure occurs, the detecting node determines the number of recursions, which indicates the number of times packets must be forwarded along the alternate next hops to bypass the failed link. This technique guarantees full repair coverage for single link failures. We evaluate the performance of our proposed technique through simulations and show that the incurred overheads, the stretch of its pre-computed alternate paths, and the failure-state Maximum Link Utilisation (MLU) are minimal.**

I. Introduction

Network reliability has been an important research topic over the past few decades due to the growth of sensitive services and applications. However, the current routing paradigm fails to satisfy the requirement for high reliability networks. Self-healing operation such as re-convergence does not provide sufficient resilience due to the time required to complete the process, hence the need for fast re-route mechanisms to cope with these issues.

The work described in [1] provides an analysis of link failures in today's backbone Internet and shows that about 50% of failures are transient, where primary routes are unavailable for short time durations (<1 minute) until the network re-converges and new routes are computed. Furthermore, it has been shown in [2] that about 70% of link failures occur one at a time *i.e.*, single link failures.

To account for this issue, several approaches based on IP Fast Re-Route (IPFRR) [3], in which alternate paths are pre-computed for fast re-route in presence of failures, have been proposed to alleviate packet loss rate due to failures. Some of the proposed algorithms include Loop Free Alternates (LFAs) [4], not-via addresses [5], Failure Carrying Packet (FCP) [6] and Multiple Routing Configuration (MRC) [7]. However, these schemes either do not guarantee full protection, are impractical in presence of frequent failures or require excessive computational and memory overheads.

Our proposed algorithm also falls within the IP Fast-Re-Route framework but does not suffer from the drawbacks just mentioned. In our proposed scheme, a node adjusts a counter for the number of recursions, *i.e.*, number of hops, necessary for forwarding through the alternate path before packets can be routed via the primary path again. This approach guarantees full coverage against single failures and incurs low computational and memory overheads.

The rest of this paper is organised as follows. In Section II, we describe the basic concepts of loop-free re-routing. We introduce our technique as an alternative to the basic mechanisms in Section III and prove the key properties of our routing strategy to ensure it is complete and correct. Section IV evaluates the performance of the routing technique with respect to incurred stretch of alternate paths, overheads and network loads. We conclude the paper in Section V.

II. Loop-Free Re-Routing

The concept of fast re-route in IP networks is not new. Several techniques have been proposed based on the IPFRR framework. These techniques focus primarily on mechanisms for repairing paths rather than mechanisms for fast failure detection. For examples, Loop-Free Alternates (LFAs) [4] and not-via addresses [5] are well-known techniques for providing alternate paths in the presence of failures. Other proposals such as Multiple Routing Configurations (MRC) [7] and Failure Carrying Packets (FCP) [6] have similar objectives, that is, to allow routers to react to failures as quickly as possible. However, each of these strategies has drawbacks. LFAs do not guarantee full protection even if an alternate path may be available. Not-via addresses require nodes to perform IP-in-IP tunnelling, which may degrade their performance. MRC requires excessive computational and memory overheads in order to generate different routing configurations that allow routers to bypass failures. Although, unlike other techniques, FCP is capable of handling multiple failures, it requires intensive computations and very large packet overhead. This paper focuses on LFAs and not-via addresses as they are practical in current IP networks.

A. Loop-Free Alternates

The basic specification of LFAs is defined in [4]. In general, LFAs can be categorised according the following conditions, which determine the ability of nodes to serve as alternate next hops: 1) loop-free condition (LFC); 2) node-protection condition (NPC); 3) downstream condition (DSC); and 4) equal-cost multi-path condition (ECMP). LFC is the basic condition used

978-1-4244-7755-5/10/$26.00 ©2010 IEEE

to guarantee loop-free forwarding in the presence of failures. However, its coverage does not include node failures. NPC recovers packets from node failures via longer alternate paths. Note that, LFAs do not handle multiple failures which might cause forwarding loops. However, LFAs that satisfy DSC and ECMP conditions do not suffer from this problem. Our routing technique focuses on handling single link failures based on (1).

$$cost(n_i, d) < cost(n_i, s) + cost(s, d) \quad (1)$$

where n_i is a neighbour of s, the detecting node, and d is the destination router.

LFAs do not require any major modifications to the existing routing paradigm. The techniques make use of additional information about the alternate next hops pre-computed by each router. Due to their simple implementation and minimal requirements, LFAs become good alternatives for handling single failures where full protection is not necessary.

B. Not-Via

The not-via addresses are defined in [5]. In the presence of failures, these addresses are used to protect specific interfaces; therefore two IP addresses are required for each interface in the network. Basically, not-via addresses are used to deviate the traffic around the failed elements by employing IP-in-IP tunnelling. Not-via addresses specify the failed nodes so that packets being re-routed along the repair paths do not encounter the failures. Although the computation of not-via addresses is based on nodes removal, a link failure is assumed if the detecting node is directly connected to the destination to guarantee full coverage of recoverable failures.

Although not-via addresses provide 100% repair paths for any single failure in arbitrary networks, these paths are considerably longer than the optimal shortest paths after network re-convergence.

III. Recursive Loop-Free Alternates

Computing backup paths can be simple with the help of source-based routing. However, designing hop-by-hop resilient routing protocols that can recover from failures quickly and with low complexity has proven difficult.

We propose a re-routing technique based on recursive Loop-Free Alternates (rLFAs) to provide loop-free repair paths for all recoverable single link failures. In general, the technique employs a series of alternate next hops pre-computed by routers. These alternate next hops are obtained based on a disjoint paths concept. However, re-routing must be assisted with a counting mechanism that allows packets to be forwarded through alternate next hops for a number of times before they can be routed normally via the primary path. The following describes the algorithms used to compute the alternate next hops and the number of recursions required to bypass a failure, and the packet forwarding mechanism.

Input: s, d, G, $E_p(s, d)$
Output: $n_s(s, d)$
1: $G' \leftarrow G$
2: **for all** $(i, j) \in E_p(s, d)$ **do**
3: $\quad w'(i, j) \leftarrow w(i, j) + W_t$
4: **end for**
5: $T'_s = ShortestPath(G', s)$
6: $n_s(s, d) \leftarrow \phi(T'_s(d))$
7: **return** $n_s(s, d)$

Fig. 1. Computing the alternate next hop.

A. Computing Alternate Next Hops

Let $G = (V, E)$ be the graph with vertices $V = \{v_1, v_2, ...\}$ and edges $E \in V \times V$ representing the network topology. Given an edge (i, j), we assign it a weight $w(i, j) \in \mathbb{R} > 0$. We define W_t as the total weight of all links in E:

$$W_t := \sum_{(i,j) \in E} w(i, j) \quad (2)$$

We seek to assign to each destination v_i a *maximally disjoint* secondary path, so that the backup path will have as few links in common with the primary path as possible.

Let $E_p(s, d)$ be the set of links used in the primary path from s to d. The primary next hop and its alternate are denoted by $n_p(s, d)$ and $n_s(s, d)$ respectively. Using W_t as a link weight re-calculation factor, the algorithm shown in Fig. 1 is run on each router for each source-destination pair. The output of the function $ShortestPath$ is the shortest path tree T_s rooted at s, with $T_s(d)$ being the shortest path from s to d excluding s, and $\phi(T_s(d))$ the first node in $T_s(d)$.

The outputs of the algorithm shown in Fig. 1 construct $S(d) = \{n_s(v_1, d), n_s(v_2, d), ...\}$, the alternate next hop nodes that every node will use to route packets to d under failure conditions. Thus, the algorithm outlined in Fig. 1 calculates an alternate next hop for each source-destination pair, which is the first hop of an alternate path that is maximally link disjoint from its corresponding normal path. However, forwarding packets to an alternate next hop in the presence of a failure does not guarantee a consistent operation as the information of the failed link is locally known to the detecting node. Thus, we need to employ a mechanism that uses these pre-computed alternate next hops to route packets to their destinations via loop-free paths.

We now introduce the algorithm used to compute the number of recursions re-routed packets must observe via a series of alternate next hops and the forwarding mechanism.

B. Computing Number of recursions

As each router in the network may have different backup paths for the same destination, forwarding must be aided with an additional mechanism that allows correct operation under failure conditions. Our technique computes and employs the number of recursions, $N_r(s, d)$, that indicate the number of alternate hops re-routed packets must go through via the

Input: s, d, $S(d)$
Output: $N_r(s,d)$
1: $N_r(s,d) \leftarrow 1$
2: $n_c \leftarrow s$
3: $C \leftarrow \emptyset$
4: **while** $n_c \neq d \wedge n_c \notin C$ **do**
5: $C \leftarrow C \cup n_c$
6: **if** $cost(n_c,d) < cost(n_c,s) + cost(s,d)$ **then**
7: **break**
8: **else**
9: $N_r(s,d) \leftarrow N_r + 1$
10: $n_c \leftarrow n_s(n_c,d)$
11: **end if**
12: **end while**
13: **return** $N_r(s,d)$

Fig. 2. Computing the number of recursions.

alternate path to ensure that no forwarding loop exists. We now illustrate how, with inconsistent information on the local alternate paths, packets can be forwarded consistently under our routing scheme.

The number of recursions $N_r(s,d)$ used by rLFAs routing is included in the packet header in order to ensure consistency. If we recall $S(d) = \{n_s(v_1,d), n_s(v_2,d), ...\}$ and define n_c as the node being determined, the $N_r(s,d)$ value can be obtained using the algorithm shown in Fig. 2.

The algorithm first initialises $N_r(s,d) = 1$ and $n_c = s$. Its alternate next hop, $n_s(n_c,d)$, is then examined using (3).

$$cost(n_c,d) < cost(n_c,s) + cost(s,d) \tag{3}$$

If $n_s(n_c,d)$ satisfies (3), the algorithm returns the current value of $N_r(s,d)$. However, if the condition is not satisfied, the algorithm increments $N_r(s,d)$ by 1 and uses $n_s(n_c,d)$ as its next n_c. This process iterates until either it finds the node that satisfies the condition, reaches the destination, or gets caught in a loop (*i.e.* no alternate path is available).

C. Packet Forwarding

Since re-routing based on rLFAs does not affect normal route calculation, packets can be forwarded to all destinations via the shortest path in the absence of failures. The operations at each node when a packet arrives are summarised in Fig. 3.

When a node s detects a failure in one of its outgoing links, it marks those packets which would be forwarded through the affected link with the number of recursions, N_r, that corresponds to the destination router of the packet. If an alternate path to that node exists, it decrements the N_r value in the packet and forwards it to its alternate next hop. When a node receives a marked packet, it determines the value of N_r. If $N_r > 0$, its value is decremented by 1 and the packet is forwarded to the node's alternate next hop. However, if $N_r = 0$, the node forwards the packet to its normal next hop until it reaches the destination.

Input: `in_pkt`
Output: `out_pkt`
1: **if** `in_pkt`.$N_r == 0$ **then**
2: **if** $(s, n_p(s, \texttt{in\_pkt}.d)) ==$ failed **then**
3: `in_pkt`.$N_r \leftarrow N_r(s, \texttt{in\_pkt}.d) - 1$
4: **return** `out_pkt` $\leftarrow$ `in_pkt`
5: **else**
6: **return** `out_pkt` $\leftarrow$ `in_pkt`
7: **end if**
8: **else**
9: **if** $(s, n_s(s, \texttt{in\_pkt}.d)) ==$ failed **then**
10: Drop(`in_pkt`)
11: **return** null
12: **else**
13: `in_pkt`.$N_r \leftarrow$ `in_pkt`.$N_r - 1$
14: **return** `out_pkt` $\leftarrow$ `in_pkt`
15: **end if**
16: **end if**

Fig. 3. Packet processing at node s.

It is important to note that, rLFAs handle only single link failures. Certain cases of multiple failures can lead to forwarding loops. However, this problem can be trivially solved. We propose the use of an extra bit to indicate a re-routed packet. That is, if a packet encounters a failure, the detecting node also marks it using this bit, in addition to the N_r. If a marked packet experiences a failure again, it will be dropped immediately.

D. Optimisation

Evidently, the calculations of alternate next hops and their corresponding N_r values imply additional computations for network elements. Since these only need to be performed for stable topology configurations to pre-compute and cache relevant values (as opposed to be carried out constantly), these can be "amortised" over longer time periods. Thus, it is feasible to perform the algorithm at practical speeds, even using commodity hardware.

If additional efficiency is required, optimised shortest path algorithms can be used. One such algorithm is the incremental shortest path first algorithm (iSPF) [8], which avoids the calculation of the whole shortest path tree and instead terminates the computation once the shortest path between the source and destination has been found. This significantly reduces the computation time of the alternate next hops.

E. Properties

The two key properties of our routing technique are: 1) full repair coverage for recoverable single link failures, and 2) loop-free forwarding. These properties are guaranteed if the routing scheme is *complete* and *correct* in the presence of *recoverable* failures. Definitions for these concepts are now given. For the remainder, we assume that equal-cost paths can be distinguished, so that all paths are essentially cost-unique, and all algorithms choose from between equal-cost

paths following a deterministic algorithm. Typical ways of achieving this include differentiating by the number of hops or on the basis of the interface ID of the first link.

Definition 3.1: A single link failure is *recoverable* for a source-destination node pair if there is at least one alternate path from the source to the destination which does not traverse the failed link.

Definition 3.2: The routing technique is *complete* if the combination of local alternate next hops and the packet forwarding mechanism guarantees a successful packet delivery in case of any single link recoverable failures.

Definition 3.3: The routing technique is *correct* if the combination of local alternate next hops and the packet forwarding mechanism can forward packets to the destination in case of any single link recoverable failures without traversing the failed links.

First, we show that our routing strategy is complete in the presence of any recoverable single link failures.

Theorem 3.1: If G is not disconnected after the removal of link $(s, n_p(s,d))$, then there exists a path from s to d via $(s, n_s(s,d))$ that does not traverse link $(s, n_p(s,d))$ under fast re-route using rLFAs.

Proof: We call $\mathcal{D}$ the set of paths from s to d that do not include $(s, n_p(s,d))$. If $\mathcal{D} = \emptyset$, the failure is non-recoverable by definition. Thus, we proceed to prove that if $\mathcal{D} \neq \emptyset$, the re-route using rLFAs algorithm will always find an alternate path from s to d in $\mathcal{D}$.

We proceed by contradiction, assuming that $\mathcal{D} \neq \emptyset$ and nonetheless the algorithm has found that $n_p(s,d) = n_s(s,d)$. This implies that the weight of all paths in $\mathcal{D}$ is strictly higher than W_t, since the algorithm adds W_t to the weight of each one of the links of the primary shortest path in order to find the alternate path from s to d, and $w(i,j) > 0; \forall (i,j)$. However, the longest path from s to d over G would be an *Eulerian path*, whose weight could be of at most W_t, and thus the weight of all paths in $\mathcal{D}$ could be of at most W_t. Hence, we have a contradiction, and $\mathcal{D} \neq \emptyset$ implies that $n_p(s,d) \neq n_s(s,d)$. ■

Second, we also show that incorporating the pre-computed alternate next hops with the alternate next hop counting mechanism can forward re-routed packets to the destination correctly under failure scenarios. Note that, the packet forwarding in normal case is based on the shortest path tree and hence, it is correct.

Theorem 3.2: If there exists a path from s to d without link $(s, n_p(s,d))$, fast re-route using rLFAs can forward packets from s to d without traversing $(s, n_p(s,d)$.

Proof: Let $T_p(d)$ be the shortest path tree rooted at d and $H_s(s,d) = \{h_1, h_2, ...\}$ be the hop sequence of the alternate path from s to d excluding s, which is locally known to s. We denote $T_p(d_s)$ as the subgraph of $T_p(d)$ below s, which includes s with a set of vertices N.

Given E is a set of vertices in N that are employed by the alternate path from s to d. Each node e_i in E has the alternate next hop, $n_s(e_i,d)$. As each node e_i shares some links in the $T_p(d)$ with s, $H_s(s,d)$ must involve $n_s(e_i,d)$.

A re-routed packet can encounter a failed link $(s, n_p(s,d))$ if and only if it traverses along $T_p(d)$ starting from any node in E. However, a node will forward a re-routed packet through $T_p(d)$ only if the N_r value is 0 - after this the packet will no longer be routed by using alternate next hops.

Since all nodes in E have alternate next hops that coincide with the alternate path from s to d, no re-routed packets arriving at e_i will have a zero value N_r. Thus, packets will not be routed along $T_p(d)$ starting from e_i. Furthermore, packets will not be routed via $T_p(d)$ starting from a node in N that does not belong to E either, since $N - E$ and $H_s(s,d)$ are disjoint sets.

Finally, routing via $T_p(d)$ from any node outside N will not cause packets to traverse $(s, n_p(s,d))$, because these nodes are not elements of $T_p(d_s)$.

Alternatively, the correctness can be proved using the condition expressed in (3). If none of the nodes satisfy (3), the routing technique fails to find an alternate paths. That is, the failure is non-recoverable.

Therefore, we conclude that a path for re-routing packets from s to d does not involve the failed link $(s, n_p(s,d))$. ■

IV. Performance Evaluation

This section presents our evaluation of fast re-route using rLFAs. We explore the incurred overheads, stretch of alternate paths and the network traffic in post-failure scenarios.

A. Method

We create our own software model to compute the alternate next hops and the number of recursions a re-routed packet needs to be forwarded along rLFAs of intermediate routers in the presence of failures. We run our simulations on a machine with a 2.16 GHz Intel Core 2 Duo processor and 2 GB memory. We use the Abilene [9] and GEANT [10] topologies, with corresponding real traffic matrices. To illustrate that our technique can perform well for arbitrary network topologies, we also use the Point-of-Presence (PoP) level topologies of Abovenet and Sprint inferred from Rocketfuel data [11].

Unfortunately, these inferred topologies do not have available traffic matrices. Therefore, we use the gravity models [12] to generate realistic traffic matrices composed of edge-to-edge flows. We use data from the U.S. Census Bureau [13] and the United Nations Statistics Division [14] to calculate the city population of each node in the network. In addition, we employ the Breadth-First Search algorithm to assign link capacity [15] in Abovenet and Sprint topologies. Furthermore, we scale the traffic matrices such that the maximum link utilisation does not exceed 100% under failure-free conditions.

Not-via addresses provide the same properties as rLFAs. Nevertheless, it requires complex mechanisms, *i.e.* IP-in-IP tunnelling. Usually not-via computation assumes a node failure; therefore, the stretch of alternate paths is obviously longer than OSPF re-route and rLFAs. Consequently, comparing stretch of alternate paths computed using not-via and rLFAs cannot yield an impartial result. Although it is possible to employ not-via addresses in combination with LFAs in order

to reduce the traffic that requires encapsulation [5], there are no strong advantages [16]. In addition, modifying not-via to cope with only link failures will yield the similar results as those of OSPF re-route. Therefore, we use OSPF re-route (the shortest paths after re-convergence) as the only benchmark throughout this evaluation.

B. Overheads

The overheads incurred by our routing strategy are evaluated as follows.

1) Computational Overhead: We evaluate the computational overhead required by computing the algorithm complexity. First, computing the alternate next hops for all source-destination node pairs require: $O(n^4 + D * n^2)$ given that D is the diameter of the network. Second, computing the number of recursions, N_r, for all soure-destination node pairs require: $O(D * n)$. Therefore, the overall computational overhead can be expressed as: $O(n^4 + D * n^2) + O(D * n^2)$.

Although the algorithm is computationally demanding, simulation results show that practical topologies have no difficulty in implementing rLFAs. For example, the backbone topology of Sprint which consists of 315 nodes and 972 symmetric links requires less than 100 ms to complete the computation of alternate next hops and their corresponding number of recursions, N_r, for all destinations. With this in mind, we conclude that our routing scheme does not incur significant computational overhead.

2) Memory Overhead: To enable re-routing via rLFAs, a router must store additional information about for each existing destination. In other words, in addition to the normal next hop, the alternate next hop and its corresponding number of recursions, N_r, must be maintained. However, no additional routing table entries are required; hence re-routing via rLFAs does not entail excessive memory overhead.

3) Packet Overhead: Our forwarding scheme requires packet space to store the number of recursions packets must be forwarded along the alternate next hops, N_r. In addition, an extra bit is needed to indicate packets that has experienced a failure to avoid possible forwarding loops resulted from multiple failures.

We conduct simulations on a large number of topologies, both real and computer generated, we find that alternate paths across all topologies have $N_r \leq 7$; therefore only 3 bits are required. However, the length of the N_r field can be adapted to suit other network topologies. Moreover, if the first alternate next hop (from the point of failure) satisfies the condition expressed in (3), we can initialise the N_r value to 0.

C. Routing Results

The characteristics of alternate paths used to re-route traffic from failures have an impact on the routing performance. We refer the ratio of alternate paths to their primary paths as the "stretch", and use it to evaluate the performance of our technique.

The main objective of fast re-route is to prevent packets from being dropped due to failures. However, it is vital to examine the stretch incurred by routing through alternate paths as a long path would degrade the quality of service of sensitive applications. Consequently, the most preferable alternate path is the path with a stretch of 1. That is, the distance between the point of failure to the destination through an alternate path is equivalent to that of its corresponding shortest path.

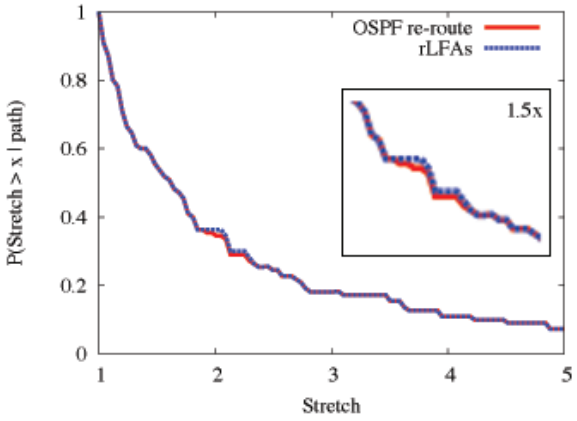

Fig. 4. Stretch comparison between OSPF re-route and rLFAs of Abilene topology.

TABLE I
AVERAGE STRETCH INCURRED BY OSPF RE-ROUTE AND RLFAS OF DIFFERENT NETWORK TOPOLOGIES.

Topology	Stretch		Equal stretch (%)
	OSPF re-route	rLFAs	
Abilene	2.413	2.417	98.182
GEANT	2.669	2.682	96.245
Abovenet	1.438	1.462	95.614
Sprint	1.313	1.319	94.812

Figure 4 illustrates the stretch incurred by alternate paths under OSPF re-route and rLFAs of Abilene network. Our simulation results show that the stretch difference between the best available paths and alternate paths computed by our algorithms is marginal regardless of network topology. As the results of OSPF re-route and rLFAs are very similar, we compare the stretch of these two strategies using the average stretch of all source-destination node pairs.

From these simulation results, we can conclude that rLFAs is capable of alleviating the problem of forwarding disruptions due to single link failures using near optimal paths (*i.e.* the shortest paths after re-convergence).

D. Traffic Results

We analyse the maximum link utilisation over different failure scenarios of each topology. The simulation result of Abilene network is illustrated in Fig. 5. The graph represents the fraction of links with MLU exceeding the value in x-axis. OSPF represents MLU under normal case while OSPF re-route and rLFAs illustrate MLU in the post-failure scenario.

In general, link utilisation is proportional to the number of links used for packet forwarding. The results show that re-routing traffic using rLFAs incurs no significant impact on the

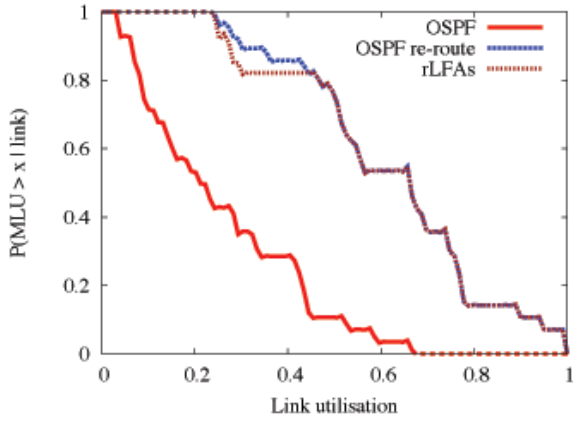

Fig. 5. MLU comparison between OSPF, OSPF re-route, and rLFAs of Abilene topology.

TABLE II
MLU OF THE LINK WITH MAXIMUM LOAD OF DIFFERENT NETWORK TOPOLOGIES.

Topology	MLU		Equal MLU (%)
	OSPF re-route	rLFAs	
Abilene	0.996	0.996	92.857
GEANT	0.993	0.993	83.784
Abovenet	0.995	0.995	88.235
Sprint	1.000	1.000	85.119

TABLE III
AVERAGE INCREASED THROUGHPUT AFTER FAILURES OF DIFFERENT NETWORK TOPOLOGIES.

Topology	ΔT (%)		Equal ΔT (%)
	OSPF re-route	rLFAs	
Abilene	5.820	5.817	96.429
GEANT	2.635	2.631	87.838
Abovenet	0.737	0.737	94.118
Sprint	0.473	0.496	84.524

MLU under different failure scenarios compared to normal re-convergence. Since the performance of OSPF re-route and that of rLFAs are very similar, we represent MLU of the link with maximum load of each network topology in Table II.

It can be seen that MLU of the link with maximum load under OSPF re-route and rLFAs are identical. Note that, the actual figures holding higher precision show negligible difference. Therefore, we also examine the increased network throughput after different failure scenarios. Table III shows the average of increased throughput, ΔT, of each network topology under OSPF re-route and rLFAs.

As can be seen, the MLU and increased network throughput after failures under rLFAs are similar to those of OSPF re-route. However, it must be noted that the performance of OSPF re-route is obtained upon the completion of re-convergence while re-routing using rLFAs provides immediate results as packets can be re-routed through rLFAs as soon as routers detect link failures.

V. CONCLUSION

Routing disruption is a major concern when deploying fault-intolerant services and applications. The routing schemes deployed in current networks do not provide sufficient mechanisms for handling failures. Several approaches such as LFAs, not-via addresses, MRC, and FCP have been proposed to alleviate the problem with considerable drawbacks.

This paper presented a novel mechanism to allow full fast re-route in the presence of transient link failures known as recursive Loop-Free Alternates (rLFAs). In addition, it was proved that the routing strategy is complete and correct. From simulation results, our technique provides not only full recovery against transient link failures, but also near optimal path and minimal impact on the network traffic after failures. We believe that fast re-route using rLFAs can greatly improve the network reliability without any expensive requirements.

REFERENCES

[1] G. Iannaccone, C. Chuah, R. Mortier, S. Bhattacharyya, and C. Diot, "Analysis of link failures in an IP backbone," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, 2002, pp. 237–242.
[2] G. Markopoulou, A. Iannaccone, S. Bhattacharyya, C. Chuah, and C. Diot, "Characterization of Failures in in an IP Backbone," in *Proc. IEEE INFOCOM*, 2004.
[3] M. Shand and S. Bryant. (2009, Feb) IP fast reroute framework. IETF Internet draft. [Online]. Available: http://tools.ietf.org/html/draft-ietf-rtgwg-ipfrr-framework-13
[4] A. Atlas and A. Zinin. (2008, Sep) Basic specification for IP fast reroute Loop-Free Alternates. RFC 5286. [Online]. Available: http://tools.ietf.org/html/rfc5286
[5] S. Bryant, M. Shand, and S. Previdi. (2009, Jul) IP fast reroute using not-via addresses. IETF Internet draft. [Online]. Available: http://tools.ietf.org/html/draft-ietf-rtgwg-ipfrr-notvia-addresses-04
[6] K. Lakshminarayanan, M. Caesar, M. Rangan, T. Anderson, S. Shenker, and I. Stoica, "Achieving convergence-free routing using failure-carrying packets," in *Proc. ACM SIGCOMM*, Kyoto, Japan, Aug 2007, pp. 241–252.
[7] A. Kvalbein, A. F. Hansen, T. Cicic, S. Gjessing, and O. Lysne, "Fast IP network recovery using multiple routing configurations," in *Proc. IEEE INFOCOM*, Barcelona, Spain, Apr 2006, pp. 23–29.
[8] J. M. McQuillan, I. Richer, and E. C. Rosen, "The new routing algorithm for the ARPANET," *IEEE Transactions on Communications*, vol. 28, no. 5, pp. 711–719, May 1980.
[9] Y. Zhang. (2004, Dec) The Abilene topology and traffic matrices. Online. [Online]. Available: http://www.cs.utexas.edu/~yzhang/research/AbileneTM/
[10] GEANT. (2004, Dec) The GEANT topology. Online. [Online]. Available: http://www.geant.net/upload/pdf/GEANT_Topology_12-2004.pdf
[11] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, "Measuring ISP topologies with Rocketfuel," *IEEE/ACM Transactions on Networking*, vol. 12, no. 1, pp. 2–16, Feb 2004.
[12] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot, "Traffic matrix estimation: Existing techniques and new directions," in *Proc. ACM SIGCOMM*, Pittsburgh, PA, Aug 2002, pp. 161–174.
[13] U.S. Census Bureau. (2000, Apr) Census 2000 gateway. Online. [Online]. Available: http://www.census.gov/main/www/cen2000.html
[14] United Nations Statistics Division. (2008, Aug) Demographic and social statistics. Online. [Online]. Available: http://unstats.un.org/unsd/demographic/
[15] W. Liu, H. T. Karaoglu, A. Gupta, M. Yuksel, and K. Kar, "Edge-to-edge bailout forward contracts for single-domain Internet services," in *Proc. IEEE IWQoS*, Enschede, The Netherlands, June 2008, pp. 259–268.
[16] R. Martin, M. Menth, M. Hartmann, T. Cicic, and A. Kvalbein, "The effect of combining loop-free alternates and not-via addresses," Institute of Computer Science, University of Würzburg, Würzburg, Germany, Research Report 432, Sep 2007.