Handling Transient Link Failures Using Alternate Next Hop Counters

Suksant Sae Lor, Raul Landa, Redouane Ali, and Miguel Rio

Network & Services Research Laboratory
Department of Electronic & Electrical Engineering
University College London, UK
{s.lor,r.landa,r.ali,m.rio}@ee.ucl.ac.uk

Abstract. In this paper, we propose a routing technique to alleviate packet loss due to transient link failures, which are major causes of disruption in the Internet. The proposed technique based on Alternate Next Hop Counters (ANHC) allows routers to calculate backup paths and re-route packets accordingly, thereby bypassing transient failures. This technique guarantees full repair coverage for single link failures, without significantly changing the way traditional routing works and with minimal impact on the computation and memory requirements for routers. We evaluate the performance of our proposed ANHC approach through extensive simulations and show that the stretch of its pre-computed alternate paths, its failure-state link load increase, and its computational and memory overheads are minimal.

Keywords: Internet routing, transient failures, fast re-routing.

1 Introduction

The development of error sensitive and critical applications over the Internet, such as health services and military has demanded a rethinking of the traditional best effort service approach. Link failures, and in particular transient link failures, represent major obstacles in guaranteeing reliability. There have been many proposals that handle failures. A well-known framework for minimising packet loss rates due to link and/or node failures is IP Fast Re-Route (IPFRR) [1]. In [2], analysis shows that the duration of forwarding disruptions must incorporate the times taken to detect the failure, react to it, propagate the failure notification, re-compute forwarding tables, and the time taken to actually install the changes into the Forwarding Information Base (FIB). IPFRR eliminates the time required by all of these except for the failure detection time, which can be shortened by adjusting protocol parameters [3]. This significantly reduces disruption time, as it allows packets to be re-routed via pre-computed backup paths as soon as the failure is detected. In this paper, we use the terms "backup" and "alternate" interchangeably to refer to these paths. The idea of using fast re-route to handle failures is not new. Several techniques such as Loop-Free Alternates (LFAs) [4] and not-via addresses [5] have been proposed and implemented in

M. Crovella et al. (Eds.): NETWORKING 2010, LNCS 6091, pp. 186–197, 2010.

real networks. However, these techniques are not sufficient, since LFAs cannot guarantee full protection against recoverable failures and the use of not-via addresses can lead to router performance degradation due to IP-in-IP tunnelling. On the other hand, a technique such as Failure-Insensitive Routing (FIR) [6] does not require a mechanism such as tunnelling to provide a 100% repair coverage against single link failures. Nevertheless, a router requires an interface-specific forwarding table which makes it more difficult to implement in hardware.

Other proposed approaches that deal with link and/or node failures include Multiple Routing Configurations (MRC) [7], Resilient Overlay Network (RON) [8] and Failure-Carrying Packets (FCP) [9]. MRC computes a number of configurations for various failure scenarios such that for a given failure, one or more configurations can be used for bypassing. However, this technique requires excessive computational and memory overheads. RON identifies a subset of nodes called the RON nodes that monitor the network and decide whether or not to send packets directly or between themselves, thereby bypassing failures if detected. A drawback of RON is that it requires continual probing of the alternate paths, hence adding overhead. FCP embeds the failed link information in the packet header. Based on this information, a router receiving FCP can calculate the latest shortest path available. Nonetheless, this increases the packet overhead and requires a very expensive dynamic computation.

In this paper, we propose a combination of an algorithm for computing backup paths and a fast packet re-routing mechanism based on Alternate Next Hop Counters (ANHC) which guarantees that packets can be fully recovered in the presence of single link failures without any of the disadvantages of the existing solutions.

The rest of the paper is organised as follows: we introduce our resilient routing technique as an alternative to IPFRR mechanisms in Sect 2. To guarantee full repair coverage and loop-free forwarding, we prove the properties of our routing strategy. In Sect. 3, we evaluate the performance of the routing technique with respect to various characteristics of the alternate paths, its impacts on network traffic, and the overheads and conclude the paper in Sect. 4.

2 Alternate Next Hop Counting

If source routing is available, the use of backup paths is greatly simplified. In contrast, it has been proven difficult to design hop-by-hop resilient routing protocols that can recover from failure quickly while keeping complexity to a minimum. The main cause of this difficulty lies in topology inconsistency among routers during re-convergence, which often leads to packet loss and forwarding loops.

Alternate next hop counting is a novel mechanism used in conjunction with specific backup path computation to provide fast re-route in traditional IP networks. Re-routing in this technique relies on the Alternate Next Hop Counter (ANHC) in the packet header to ensure correct forwarding. After the normal shortest paths are calculated, a router first determines the sum of all link weights W_t. For each node pair (s, d), a router adds W_t to the weights of the links present

in the shortest path between s and d and re-calculates the shortest path from s to d. This *secondary path* is used to calculate the *alternate next hop* that s will use to send packets to d if the shortest path is unavailable. To allow consistent routing only on the basis of alternate next hops, each router sets the ANHC value for every destination equal to the number of times the packet needs to be forwarded using the alternate next hop. If the ANHC holds a positive number, a router decreases its value by 1 and forwards the packet to the alternate next hop corresponding to its own secondary path to d. If the ANHC value equals to 0, the router forwards the packet via its normal path to d (the next hop according to the shortest path route).

2.1 Computing Backup Paths

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:

$$W_t := \sum_{(i,j)\in E} w(i, j) \tag{1}$$

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 normal path as possible.

Let $E_p(s, d)$ be the set of links used in a normal path from s to d. The primary next hop and its alternate are denoted as $n_p(s, d)$ and $n_s(s, d)$. Using W_t as a link weight re-calculation factor, Algorithm 1 is run on each router for each source-destination pair. In Algorithm 1, the output of $ShortestPath$ is a 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 $T_s(d)$.

Algorithm 1. Computing the alternate next hop

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: $H_s(s, d) \leftarrow T'_s(d)$
7: $n_s(s, d) \leftarrow \phi(T'_s(d))$
8: **return** $n_s(s, d)$

The outputs of Algorithm 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, and $H_s(s, d) = \{h_1, h_2, ...\}$, where $H_s(s, d)$ is the backup path

from s to d excluding s, and h_i represents the i-th next hop in the backup path from s to d. Thus, Algorithm 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. Of course, each node $s \in G$ will calculate a different $n_s(s, d)$ for a given destination d: alternate paths for a destination d are not consistent throughout the network. Our technique is different from some other resilient mechanisms precisely because the alternate path locally known to each router does not have to be consistent: there is no guarantee that a path formed by concatenating the alternate next hops to a particular destination will be loop-free. 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 propose a mechanism called *alternate next hop counting* to eliminate this path inconsistency problem.

2.2 Computing ANHC Values

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 uses alternate next hop counting to ensure that no forwarding loop is possible in the presence of a single link failure. We now illustrate how, with inconsistent information on the local alternate paths, packets can be forwarded consistently under our routing scheme.

Alternate next hop counting makes use of an Alternate Next Hop Counter (ANHC) stored in the packet header. A few bits in the Type of Service (ToS) field of IPv4 or the Traffic Class field of IPv6 are sufficient to store the ANHC.

If we recall $S(d) = \{n_s(v_1, d), n_s(v_2, d), ...\}$ and $H_s(s, d) = \{h_1, h_2, ...\}$, the $AHNC(s, d)$ value can be obtained using Algorithm 2.

Algorithm 2. Computing the ANHC value

Input: s, d, $H_s(s, d)$, $S(d)$
Output: $ANHC(s, d)$

```
1: ANHC ← 0
2: current_node ← s
3: i ← 1
4: while h_i ≠ d do
5:    if h_i == n_s(current_node, d) then
6:       ANHC ← ANHC + 1
7:       current_node ← h_i
8:       i ← i + 1
9:    else
10:      break
11:   end if
12: end while
13: return ANHC
```

The algorithm first considers the *current_node* which is initialised with s. The first hop in the alternate path from s to d is then compared with the alternate next hop of *current_node* to d. If they are the same node, $ANHC(s, d)$ is incremented by 1 and the alternate next hop of *current_node* to d becomes *current_node*. After that, the second hop in the alternate path from s to d is used for comparison. This process iterates until either it reaches d or the condition fails, implying that it is no longer necessary to route via the alternate next hop. From this point, the packet can be forwarded normally. When the algorithm is terminated, $ANHC(s, d)$ is obtained.

2.3 Packet Forwarding

Since the alternate next hop counting mechanism does not affect normal route calculation, packets can be forwarded to all destinations via the shortest paths in the absence of failures. When a node v detects a failure in one of its outgoing links, it marks those packets which would be forwarded through the affected link with the ANHC value corresponding to the destination router of the packet. If an alternate path to that node exists, it decrements the ANHC value in the packet and forwards it to its alternate next hop. When a node receives a marked packet, it determines the value of ANHC. If ANHC holds a positive number, its value is decremented by 1 and the packet is forwarded to the node's alternate next hop. However, if the ANHC value is 0, the node forwards the packet to its normal next hop until it reaches the destination. Algorithm 3 summarises the operations when a packet arrives at each node.

Algorithm 3. Packet processing at node s

Input: `in_pkt`
Output: `out_pkt`

if `in_pkt`.$ANHC == 0$ **then**
  if $(s, n_p(s, \texttt{in\_pkt}.d)) ==$ failed **then**
    `in_pkt`.$ANHC \leftarrow ANHC(s, \texttt{in\_pkt}.d) - 1$
    return `out_pkt` $\leftarrow$ `in_pkt`
  else
    return `out_pkt` $\leftarrow$ `in_pkt`
  end if
else
  if $(s, n_s(s, \texttt{in\_pkt}.d)) ==$ failed **then**
    Drop(`in_pkt`)
    return null
  else
    `in_pkt`.$ANHC \leftarrow$ `in_pkt`.$ANHC - 1$
    return `out_pkt` $\leftarrow$ `in_pkt`
  end if
end if

It is important to note that, our routing technique handles only **single** link failures. Certain cases of multiple failures can lead to forwarding loops. However, this problem can be trivially corrected. 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 ANHC. If a marked packet experiences a failure again, it will be dropped immediately as the routing technique does not handle multiple failures.

Intuitively, the resilient routing technique described will, in case of failure, route packets along the first hops of maximally disjoint paths terminating on their destination node until, after a number of hops equal to the ANHC, a node is reached where they can be routed using the normal shortest path tree of the network. The shortest path route from this node need not be equal to the backup path calculated by the failure-detecting node - in fact, it is frequently shorter, as it does not need to be maximally disjoint. Thus, the length of the actual path that the packets will traverse under failure scenarios is no greater than the length of the backup path to their destination starting from their failure-detecting node.

Furthermore, since all failure-avoiding packets will traverse a number of hops equal to their ANHC before they are shortest-path routed, the length of the actual path that the packets will traverse under failure scenarios is no shorter than the ANHC to their destination starting from their source.

Evidently, the computation of alternate next hops and their corresponding ANHC values implies 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) [10], 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.

2.4 Properties

The two key properties of our routing technique are *a)* full repair coverage for recoverable single link failures, and *b)* 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 involve differentiating by the number of hops on each path, or choosing on the basis of the interface ID for the first link.

Definition 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 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. *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 through the failed links.*

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

Theorem 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 with ANHC.*

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 ANHC 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 2. *If there exists a path from s to d without link $(s, n_p(s,d))$, fast re-route using ANHC 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 ANHC 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 ANHC. 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 $T_p(d_s)$.

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

It is important to note that, alternate path locally known to each router is used for ANHC value calculation only and the routing technique does not hinder the network operator from knowing the actual alternate path used for packet re-routing.

3 Performance Evaluation

This section presents our evaluation of fast re-route resiliency using ANHC. The broad areas we investigate include protocol overhead, characteristics of the alternate paths, the impact of failures and associated recovery mechanisms on link load.

3.1 Method

We create our own software model to compute alternate next hops and their corresponding ANHC values. 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 [11] and GEANT [12] topologies, with corresponding real traffic matrices [13]. To illustrate that our technique can perform well for arbitrary network topologies, we also use the Abovenet and Sprintlink topologies inferred from Rocketfuel data [14], and synthetic topologies generated by BRITE [15] based on Waxman and Barabasi-Albert (BA) models.

We use the gravity models [16] to generate traffic matrices composed of edge-to-edge flows. To use realistic inputs to the gravity models, we use data from the U.S. Census Bureau [17] and the United Nations Statistics Division [18] to calculate the city population of each node in the network. Furthermore, we use a technique based on the Breadth-First Search (BFS) algorithm to assign link capacity [19]. To examine the traffic characteristics, we scale the traffic matrices such that the maximum link utilisation does not exceed 100% under normal routing or after re-convergence.

We use the normal shortest path routing as a benchmark comparison. The path used upon completion of the re-convergence process is denoted as *OSPF re-route*.

Of course, ANHC achieves the performance level reported immediately after a failure is detected, whereas OSPF re-route normally requires several seconds of re-convergence stabilisation before it can achieve its reported performance.

3.2 Overheads

We evaluate the computational overhead of our routing strategy by reporting the time required for computation of the alternate next hops and their corresponding ANHC values as evaluation metrics. The largest topology (*i.e.* Sprintlink) requires less than 100 ms for alternate next hop computation. ANHC value computation time is negligible for all topologies. Table 1 summarises the simulation results based on different topologies. Overall, results show that the algorithms used to compute essential parameters (*i.e.* alternate next hops and ANHC values) of our routing scheme do not incur any significant computational overhead.

Table 1. Summary of topologies used in simulations and the corresponding computational overhead introduced by fast re-route using ANHC

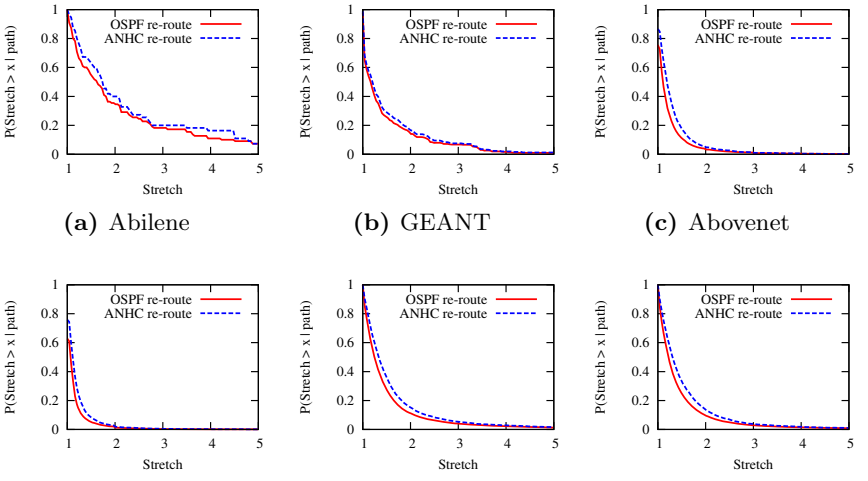
Topology	Type	Number of nodes	Number of links	In-/Out-degree	Computation time (ms)		
					ANHs	ANHC	Total
Abilene	Real	11	28	1.273	0.108	0.006	0.114
GEANT	Real	23	74	1.609	0.120	0.008	0.128
Abovenet	Inferred	138	744	2.906	11.920	0.020	11.940
Sprintlink	Inferred	315	1944	3.086	77.091	0.051	77.142
Waxman	Random	100	400	2.000	3.091	0.015	3.106
Barabasi-Albert	Random	100	394	1.970	3.383	0.016	3.339

In order to enable re-routing with alternate next hop counting, a router must store additional information for each existing destination. That is, apart from the normal next hop, its alternate and the corresponding ANHC value must be maintained. However, no additional routing table entries are required and, hence, our routing technique does not entail excessive memory overhead.

Our forwarding scheme requires a few bits to store the ANHC value. We suggest that a part of the ToS field can be used for IPv4, and a part of the Traffic Class field can be used for IPv6. From simulation results, more than 90% of the alternate paths have an ANHC value of less than 3. In practice, a maximum ANHC of 7 is needed in the packet, as the failure-detecting node will decrement the ANHC value before it forwards the packet to its alternate next hop. To prevent forwarding loops in the presence of multiple failures, we propose the use of an extra bit to indicate a re-routed packet. With this in mind, our routing technique needs no more than 4 bits for an optimal packet re-routing in practical topologies.

3.3 Path Stretch

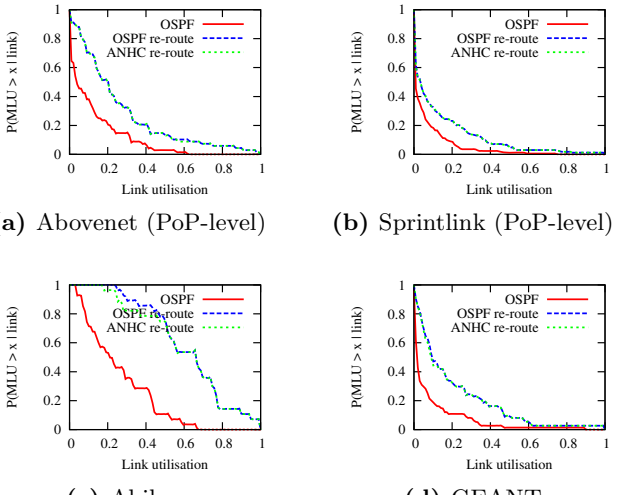
We measure the excess latency or weight required for delivery via alternate paths using *stretch*. Although the main objective of the IPFRR framework is to prevent packets from being dropped in case of failures, it is important



(a) Abilene **(b)** GEANT **(c)** Abovenet

(d) Sprintlink **(e)** Waxman **(f)** Barabasi-Albert

Fig. 1. Stretch comparison between OSPF re-route and IPFRR using ANHC



(a) Abovenet (PoP-level) **(b)** Sprintlink (PoP-level)

(c) Abilene **(d)** GEANT

Fig. 2. Maximum link utilisation before and after single link failures of different topologies

that packet delivery via alternate paths does not cause too high delay so that end users of sensitive applications can perceive a poorer Quality of Service (QoS).

As can be seen in Fig. 1, our fast re-route technique offers recovery via alternate paths that are near optimal regardless of the underlying network topology. The average of optimal stretch across all topologies is 1.221 while the average of stretch provided by our proposed technique is 1.305. In most cases, the cost of an alternate path is close to that of the shortest path.

3.4 Maximum Link Utilisation

We analyse the maximum link utilisation over different failure scenarios of each topology. Figure 2 illustrates the fraction of links in the network with maximum link utilisation exceeding the value in x-axes. Interestingly, unlike other IPFRR techniques [20], none of the links in all topologies are overloaded under our routing strategy. Fast re-route with ANHC also performs as well as normal re-convergence in any arbitrary network.

4 Conclusion

Packet dropping due to transient failures is a major cause of disruption in today's Internet and will become more critical as more demanding applications are developed. Several approaches such as IPFRR, multi-homing and multi-path routing, and overlay networks have been proposed to alleviate the problem. Although IPFRR provides better modularity, none of its techniques can guarantee full protection against single failures without using mechanisms (*e.g.* tunnelling) that degrade the performance of a router.

We proposed a new IPFRR technique based on the Alternate Next Hop Counter (ANHC) to handle transient link failures. In the normal scenario, packets are forwarded along the shortest path calculated similarly as in traditional IP routing. When a link fails, the detecting router is responsible for setting the pre-computed ANHC value in the packet header. This value is used by intermediate routers to determine the packet's next hop. We presented two algorithms for computing the alternate paths and their corresponding ANHC values. Furthermore, we proved that our technique is complete and correct.

As the path provided by OSPF re-route is the shortest path after a failure, we used it as a benchmark throughout our evaluation. From simulation results, we concluded that fast re-route using ANHC requires no significant overheads and can be easily deployed without major modifications. Moreover, it can fully protects single link failures using low stretch alternate paths and has minimal impact on network traffic. We greatly believe that fast re-route using ANHC can greatly enhance the network reliability without any expensive requirements.

References

1. Shand, M., Bryant, S.: IP fast reroute framework. RFC 5714 (January 2010), http://tools.ietf.org/html/rfc5714
2. Li, A., Francois, P., Yang, X.: On improving the efficiency and manageability of notvia. In: Proc. ACM CoNEXT, New York, December 2007, pp. 1–12 (2007)
3. Goyal, M.R., Feng, K.K.W.: Achieving faster failure detection in OSPF networks. In: Proc. IEEE ICC, Anchorage, AK, May 2003, pp. 296–300 (2003)
4. Atlas, A., Zinin, A.: Basic specification for IP fast reroute Loop-Free Alternates. RFC 5286 (September 2008), http://tools.ietf.org/html/rfc5286
5. Bryant, S., Shand, M., Previdi, S.: IP fast reroute using not-via addresses. IETF Internet draft (July 2009), http://tools.ietf.org/html/draft-ietf-rtgwg-ipfrr-notvia-addresses-04
6. Nelakuditi, S., Lee, S., Yu, Y., Zhang, Z.L., Chuah, C.N.: Fast local rerouting for handling transient link failures. IEEE/ACM Transactions on Networking 15(2), 359–372 (2007)
7. Kvalbein, A., Hansen, A.F., Cicic, T., Gjessing, S., Lysne, O.: Fast IP network recovery using multiple routing configurations. In: Proc. IEEE INFOCOM, Barcelona, Spain, April 2006, pp. 23–29 (2006)
8. Andersen, D.G., Balakrishnan, H., Kaashoek, M.F., Morris, R.: Resilient Overlay Network. In: Proc. ACM SOSP, Banff, Canada, October 2001, pp. 131–145 (2001)
9. Lakshminarayanan, K., Caesar, M., Rangan, M., Anderson, T., Shenker, S., Stoica, I.: Achieving convergence-free routing using failure-carrying packets. In: Proc. ACM SIGCOMM, Kyoto, Japan, August 2007, pp. 241–252 (2007)
10. McQuillan, J.M., Richer, I., Rosen, E.C.: The new routing algorithm for the ARPANET. IEEE Transactions on Communications 28(5), 711–719 (1980)
11. Zhang, Y.: The Abilene topology and traffic matrices (December 2004), http://www.cs.utexas.edu/~yzhang/research/AbileneTM/
12. GEANT: The GEANT topology (December 2004), http://www.geant.net/upload/pdf/GEANT_Topology_12-2004.pdf
13. Uhlig, S., Quoitin, B., Balon, S., Lepropre, J.: Providing public intradomain traffic matrices to the research community. ACM SIGCOMM Computer Communication Review 36(1), 83–86 (2006)
14. Spring, N., Mahajan, R., Wetherall, D., Anderson, T.: Measuring ISP topologies with Rocketfuel. IEEE/ACM Transactions on Networking 12(1), 2–16 (2004)
15. Medina, A., Lakhina, A., Matta, I., Byers, J.: BRITE: an approach to universal topology generation. In: Proc. IEEE MASCOTS, Cincinnati, OH, August 2001, pp. 346–353 (2001)
16. Medina, A., Taft, N., Salamatian, K., Bhattacharyya, S., Diot, C.: Traffic matrix estimation: Existing techniques and new directions. In: Proc. ACM SIGCOMM, Pittsburgh, PA, August 2002, pp. 161–174 (2002)
17. U.S. Census Bureau: Census 2000 gateway (April 2000), http://www.census.gov/main/www/cen2000.html
18. United Nations Statistics Division: Demographic and social statistics (August 2008), http://unstats.un.org/unsd/demographic/
19. Liu, W., Karaoglu, H.T., Gupta, A., Yuksel, M., Kar, K.: Edge-to-edge bailout forward contracts for single-domain Internet services. In: Proc. IEEE IWQoS, Enschede, The Netherlands, June 2008, pp. 259–268 (2008)
20. Menth, M., Hartman, M., Martin, R., Cicic, T., Kvalbein, A.: Loop-free alternates and not-via addresses: A proper combination for IP fast reroute? Computer Networks (2009)