Title	Enhancing TCP performance in IP fast reroute
Author(s)	Mao, M; Wen, Z; Yeung, LK
Citation	The IEEE International Conference on Communications (ICC 2012), Ottawa, ON., 10-15 June 2012. In IEEE International Conference on Communications, 2012, p. 1235-1238
Issued Date	2012
URL	http://hdl.handle.net/10722/165314
Rights	IEEE International Conference on Communications. Copyright © IEEE.

Enhancing TCP Performance in IP Fast Reroute

Minjing Mao, Zheng Wen and Kwan L. Yeung
Department of Electrical and Electronic Engineering
The University of Hong Kong
Pokfulam, Hong Kong
{mjmao, wenzheng, kyeung}@eee.hku.hk

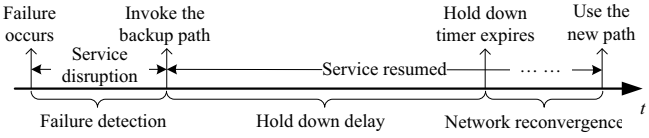
***Abstract*— IP fast reroute (IPFRR) mechanisms are efficient in providing protection against link or router failure by invoking locally determined backup paths. But in the presence of a transient network failure, the path oscillates between backup and original paths, causing frequent packet out-of-order arrivals at a TCP receiver. Duplicate acknowledgments (DUP_ACKs) generated by the receiver will then trigger unnecessary TCP congestion control at the sender. In this paper, we propose a novel yet simple algorithm called duplicate acknowledgement suppression (DAS) for enhancing the TCP performance in the presence of IPFRR. DAS only requires a minor modification to the TCP implementation at the receiver side while leaving the sender intact. The key idea is to use the time-to-live field of an out-of-order packet to infer the cause of its out-of-order arrival. If that is deemed due to a transient network failure, no DUP_ACK will be generated. Simulation results show that our DAS algorithm yields noticeable goodput improvement.**

Keywords-IP fast reroute (IPFRR); Transmission Control Protocol (TCP); packet out-of-order.

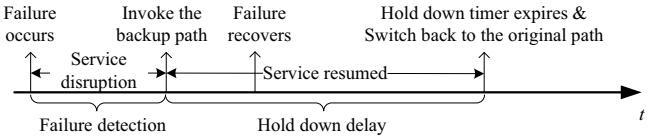
I. INTRODUCTION

Large-scale deployments of loss-sensitive Internet services require a disruption time as short as tens of milliseconds in IP networks [1]. To address this challenge, the technique known as IP fast reroute (IPFRR) was proposed [2]. The basic idea is that a router locally reroutes the affected traffic onto a pre-calculated backup path as soon as a failure is detected. Various schemes for finding efficient backup paths have been proposed [2-9]. No matter which scheme to use, a fast failure detection protocol is essential. To this end, bidirectional failure detection protocol [10] and fast hellos [11-12] have been designed.

When a router with IPFRR detects a failure, it immediately reroutes the affected traffic onto the backup path. But it does not immediately notify other routers in the network of the failure until a pre-defined hold down timer expires. If the hold down timer expires and the failure persists (as shown in Fig. 1(a)), the failure is deemed persistent and is then announced to other routers in the network. Based on the specific routing protocol used, the network re-convergence takes place for finding a new set of paths to bypass the failed component. During the re-convergence, micro-loops can be developed because of temporary inconsistencies [2] between the routers' forwarding tables. To avoid packet loss due to micro-loops, the affected traffic is rerouted to the (converged) new path only when the network re-convergence completes. If the failure is recovered before the hold down timer expires (as shown in Fig. 1(b)), the failure is deemed transient. The affected traffic is rerouted back to the original path without triggering network re-convergence. It was reported in [9] that the majority of network failures are transient, with 46% of them last less than one minute and 86% less than ten minutes. The main causes of transient failures include router crash or reboot, line-card failure or reset, routine maintenance, etc. In this paper, we are interested in enhancing the TCP performance during a transient failure.



(a) Persistent network failure.



(b) Transient network failure.

Figure 1. Procedures for handling a network failure in IPFRR.

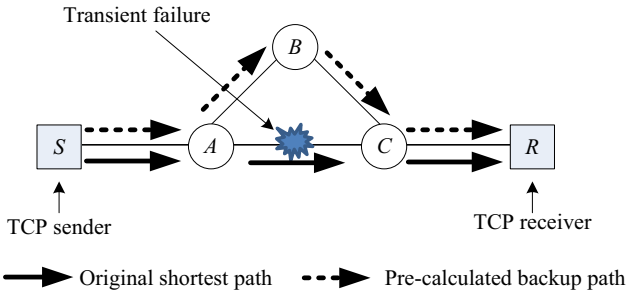


Figure 2. Packet out-of-order problem due to the path difference.

Although IPFRR improves the routing stability by not triggering network re-convergence during transient network failures, traffic rerouting can lead to packet out-of-order problem. We illustrate this using the example network in Fig. 2, where the pre-calculated backup path for link *A*-*C* is *A*-*B*-*C*. When a (transient) failure at link *A*-*C* occurs, the affected packets are rerouted from *A*-*C* to *A*-*B*-*C*. Note that this rerouting is from a shorter path to a longer one. Assume that the original path *A*-*C* is recovered before the hold down timer expires. Then the traffic on *A*-*B*-*C* will be rerouted back to *A*-*C*. This second rerouting is from a longer path to a shorter one. It

978-1-4577-2053-6/12/$31.00 ©2012 IEEE

is likely that packets sent (early) along *A-B-C* may arrive at the receiver later than packets sent (late) along *A-C*, resulting in out-of-order arrivals. Note that for each out-of-order packet, the TCP receiver generates a duplicate acknowledgment (DUP_ACK) and sends it back to the TCP sender. Upon receiving three DUP_ACKs in a row, the sender will undergo a *fast retransmit* [13], where a "lost" packet is retransmitted and the sending rate is halved. Indeed, this fast retransmit is not necessary and we call it a *false* fast retransmit. If there are many false fast retransmits in the network, the TCP performance/throughput will be lowered.

To the best of our knowledge, the above false fast retransmit problem has not been studied under the context of IPFRR. To address this problem, we propose a duplicate acknowledgement suppression (DAS) algorithm in this paper. With DAS, when an out-of-order packet arrives (as detected by the TCP sequence number gap), the TCP receiver examines the time-to-live (TTL) field in the packet (IP datagram) header, and compares it with the last-received in-order packet. If there is an increase in TTL, which serves as a strong indication of path switch from backup (longer) to original (shorter) in IPFRR, DUP_ACK is suppressed. Although packet out-of-order could also be triggered by other events, e.g. packet loss due to buffer overflow or transmission error, it is unlikely to be accompanied by a change/increase in TTL value. A distinct feature of our DAS algorithm is simplicity and ease of implementation. Indeed, only a minor modification to the TCP receiver implementation is required while the sender is kept untouched.

The rest of the paper is organized as follows. In Section II, we formally present the DAS algorithm. In Section III, the performance of the DAS algorithm is studied by simulations. In Section IV, we analyze the DAS algorithm under some specific situations. Finally, we conclude the paper in Section V.

II. Duplicate Acknowledgement Suppression

Each IP datagram contains a time-to-live (TTL) field in its header. The initial TTL value is set by the sending host (sender), and all packets sent from this host will have the same initial TTL value. At each router hop, the value of TTL will be reduced by one. If TTL is equal to zero, the router will discard the packet to prevent possible routing loops.

The TTL field in the IP header has been shown to be reliable for detecting route changes [14]. If there is no path change, packets from the same sender to the same receiver will always have the same (residual) TTL value. If there is a path change and there is a difference in path length, the TTL value of packets following different paths will be different. Note that with IPFRR, the backup path is generally longer than the original path because the original path is usually found using the shortest path routing algorithm. Accordingly, the (residual) TTL value of the packets carried by the backup path will be smaller than that of the packets carried by the original path.

To address the false fast retransmit problem illustrated by the example in Fig. 2, a duplicate acknowledgement suppression (DAS) algorithm is designed. With DAS, a TCP receiver monitors the TTL value of each packet received. When an out-of-order packet arrives (as indicated by a sequence number gap in the TCP byte stream), its TTL (denoted as current_ttl) is compared with that of the last-received in-order packet (or last_in_order_ttl). If there is an increase in TTL, the current packet has traversed a shorter distance, which is deemed due to a path switch from backup to original. Accordingly, no DUP_ACK is generated and the receiver sets its status flag DUP_ACK_SUPPRESS to 1. When an in-order packet arrives, we reset DUP_ACK_SUPPRESS to 0 and last_in_order_ttl = current_ttl.

The DAS algorithm can be summarized by the following pseudo codes.

```
1:   Initialize the status flag DUP_ACK_SUPPRESS = 0.
2:   for each packet received
3:     current_ttl = the TTL value in the packet.
4:     if (the packet is out of order)
5:       if((current_ttl> last_ in_order _ttl)
           && (DUP_ACK_SUPPRESS = 0))
6:           No DUP_ACK is generated;
             DUP_ACK_SUPPRESS = 1.
7:       else if ((current_ttl > last_ in_order _ttl)
           && (DUP_ACK_SUPPRESS = 1))
8:           No DUP_ACK is generated.
9:       else if ((current_ttl = last_ in_order _ttl)
           && (DUP_ACK_SUPPRESS = 1))
10:          DUP_ACK is generated as usual;
             DUP_ACK_SUPPRESS = 0.
11:      end if
12:    end if
13:    if (the packet is in-order)
14:      DUP_ACK_SUPPRESS = 0.
         last_ in_order _ttl = current_ttl.
15:    end if
16:  end for
```

The rest of TCP implementation at the receiver remains unchanged. In particular, DAS can properly handle the packet out-of-order arrival caused by other events. When an out-of-order packet arrives and there is no change in TTL, it implies that the packet out-of-order arrival is triggered by other events. A DUP_ACK is generated as usual, and DUP_ACK_SUPPRESS is set to 0. In Section IV, we show that our DAS algorithm can correctly handle both types of packet out-of-order events.

III. PERFORMANCE EVALUATIONS

To evaluate the proposed DAS algorithm, we implement DAS in NS2 version 2.34, and compare the performance of TCP NewReno with and without DAS. Two networks shown in Fig. 2 and Fig. 3 are simulated. The pre-determined backup paths are highlighted in the corresponding figures, which can be found by, e.g. LFA [4]. It should be noted that our DAS algorithm is independent of the backup path finding algorithm used. For simplicity, all the links in Figs. 2 & 3 are set to 100 Mb/s with a one-way propagation delay of 5ms. The router buffer size is set to be sufficiently large to avoid buffer overflow. For TCP, we assume the sender always has data to send and with a maximum segment/packet size of 512 bytes. We use goodput, the amount of successfully delivered in-order bytes, as our performance metric.

For the network in Fig. 2, we measure the performance of one single TCP connection between S and R. The simulation lasts for 11 seconds. The transient failure at link A-C occurs at t=3 second and is recovered at t=8 second. As expected, NewReno with DAS outperforms plain NewReno, achieving an improvement of up to 30.4%, where the average goodputs without and with DAS are 19931 Kb/s and 15280 Kb/s, respectively.

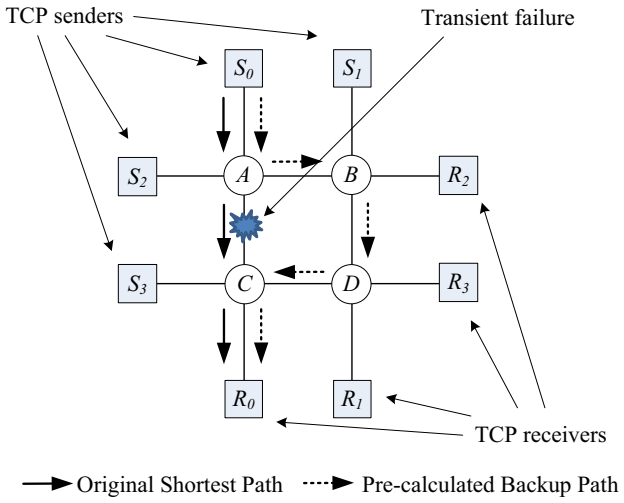


Figure 3. #-shaped network for simulation.

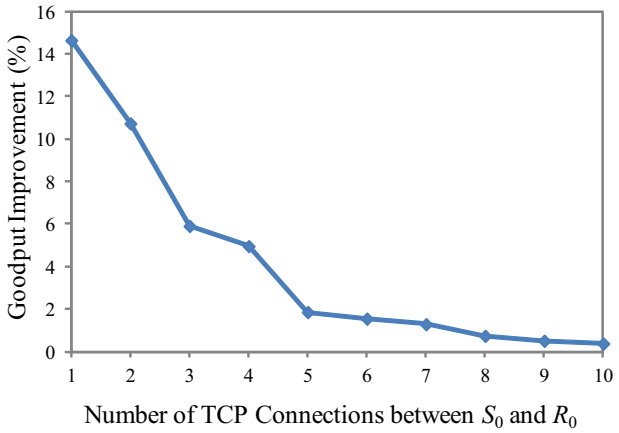


Figure 4. Goodput improvement vs. the number of observed TCP connection.

Based on the network shown in Fig. 3, a more general scenario is considered, where multiple TCP connections and background traffic are simulated. Specifically, there are four pairs of end hosts in the #-shaped network, S_0/R_0, S_1/R_1, S_2/R_2 and S_3/R_3. There are n ($1 \le n \le 10$) TCP connections between S_0 and R_0, and 10 TCP connections between the other three pairs of end hosts. The latter functions as the background traffic generators. Each simulation run lasts for 6 seconds. The transient failure at link A-C occurs at time t and lasts for 2 seconds, where t is uniformly distributed between 2 and 4 seconds. For each value of n, statistics from 15 simulation runs with different values of t are collected. For a fair comparison, the same random number seed is used for both NewReno with and without DAS algorithm.

Fig. 4 shows the percentage improvement in TCP goodput as a result of using DAS, versus the number of TCP connections (n) between S_0 and R_0. We can see that the percentage improvement decreases with the TCP connection number. This is because the fraction of the TCP connections that would suffer from false fast retransmit decreases with the total number of TCP connections.

IV. CASE BASED ANALYSIS

In our simulation study above, we only consider packet out-of-order arrivals introduced by the path switch from backup to original. In practice, packet out-of-order triggered by other events is unavoidable. In this section, we use a few designed examples to demonstrate how our DAS reacts to *real* out-of-order packet arrivals while the path is switching from backup to original.

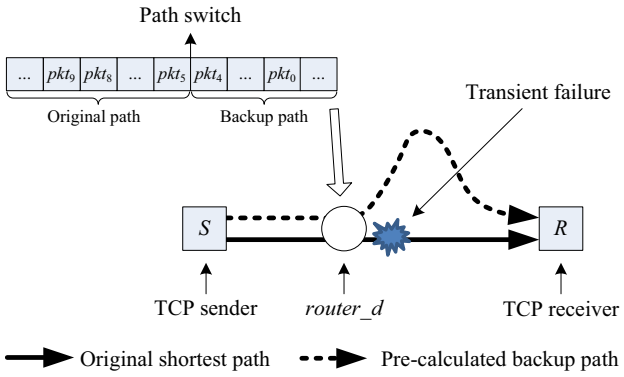
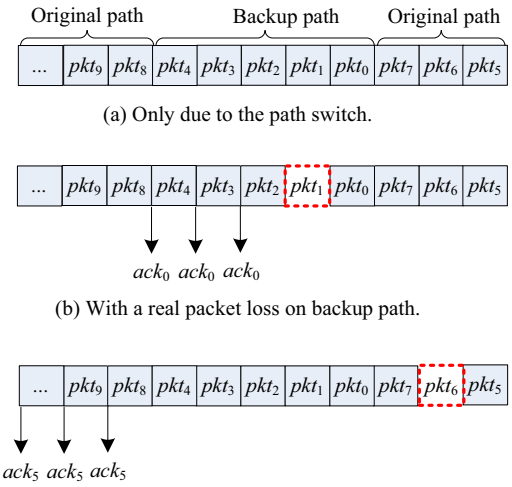


Figure 5. Example for case based analysis.

Consider the example shown in Fig. 5. Let the router that detects an adjacent failure be *router_d*. Without loss of generality, we assume the packets arrived at *router_d* are in the correct order, and the failure is recovered between the transmission of pkt_4 and pkt_5. In other words, pkt_0, pkt_1, pkt_2, pkt_3, pkt_4 and some packets transmitted before are carried by the backup path, and the packets following pkt_4 will be carried by the shorter original path.

Assume that three packets pkt_5, pkt_6 and pkt_7 arrive at the TCP receiver R out of order, as shown in Fig. 6 (a). Since the out-of-order packet arrivals are caused by the path switch in IPFRR, no DUP_ACK will be generated with our DAS.



(a) Only due to the path switch.

(b) With a real packet loss on backup path.

(c) With a real packet loss on the original path.

Figure 6. Out-of order packet arrivals at the TCP receiver.

We next study the process of DAS on a real packet out-of-order arrival with two examples, as shown in Figs. 6 (b) & (c), respectively. In Fig. 6 (b), assume that pkt_1 is lost during the transmission on the backup path. Accordingly, the packets following it arrive at *R* out of order. Although three packets received before, i.e., pkt_5, pkt_6 and pkt_7, are also out-of-order, our DAS can differentiate the reasons of their out-of-order arrivals based on the TTL values. For the packets following pkt_1, the most-recently-received in-order packet is pkt_0, and thus there is no change in TTL. With our DAS, it can be inferred that such packet out-of-order arrivals are caused by events other than the path switch. Therefore, a DUP_ACK (i.e. ack_0) should be generated as usual.

In Fig. 6 (c), we consider a lost packet, i.e. pkt_5, on the original path. The packets following pkt_5 arrive at *R* out-of-order. Since these packet out-of-order arrivals are mixed with the false ones due to the path switch, our DAS cannot distinguish them immediately. However, after all the packets delayed by the path switch are received, the implementation at the receiver will go back to normal: a DUP_ACK (i.e., ack_5) is generated upon receiving an out-of-order packet.

From the above case analysis, we can see that our DAS can process real packet out-of-order arrivals correctly.

V. CONCLUSION

IPFRR mechanisms can effectively improve the routing stability by not triggering network re-convergence during transient network failures. Nevertheless, the traffic rerouting from backup to original can result in frequent packet out-of-order arrivals at a TCP receiver. Duplicate acknowledgements (DUP_ACK) generated by the receiver will lead to false fast retransmit at the TCP sender and lower the TCP throughput. To solve the problem, we proposed a simple algorithm called duplicate acknowledgement suppression (DAS). DAS only requires a minor modification to the TCP implementation at the receiver side, with no change at the sender. In DAS algorithm, we suppressed the DUP_ACK when detecting an increase in TTL of an out-of-order packet received. As compared with plain TCP Newreno, simulation results showed that NewReno with our proposed DAS can provide considerable amount of improvement in goodput.

REFERENCES

[1] S. Rai, B. Mukherjee and O. Deshpande, "IP resilience within an autonomous system: current approaches, challenges, and future directions," *IEEE Communications Magazine*, vol. 43, no. 10, pp. 142-149, Oct. 2005.

[2] M. Shand and S. Bryant, "IP fast reroute framework," RFC5714, Jan. 2010.

[3] A. Iselt, A. Kirstdter, A. Pardigon, and T. Schwabe, "Resilient routing using ecmp and mpls," *in Proc. IEEE HPSR'04*, pp. 345-349, Apr. 2004.

[4] A. Atlas and A. Zinin, "Basic specification for IP fast reroute: loop-free alternates," RFC5286, Sep. 2008.

[5] A. Atlas, "U-turn alternates for IP/LDP fast-reroute," Internet Draft, Mar. 2006. [Online]. Available: http://tools.ietf.org/html/draft-atlas-ip-local-protect-uturn-03

[6] S. Bryant, C. Filsfils, S. Previdi, and M. Shand, "U-turn alternates for IP/LDP fast-reroute," Internet Draft, Nov. 2007. [Online]. Available: http://tools.ietf.org/html/draft-bryant-ipfrr-tunnels-03

[7] M. Shand, S. Bryant and S. Previdi, "IP fast reroute using not-via addresses," Internet Draft, Apr. 2011. [Online]. Available: http://tools.ietf.org/html/draft-ietf-rtgwg-ipfrr-notvia-addresses-07

[8] A. Kvalbein, A. F. Hansen, T. Cicic, S. Gjessing and O. Lysne, "Fast IP network recovery using multiple routing configurations," *in Proc. IEEE INFOCOM'06*, pp. 1-11, Apr. 2006.

[9] Z. Zhong *et al.*, "Failure inferencing based fast rerouting for handling transient link and node failures," *in Proc. IEEE INFOCOM'05*, pp. 2859-2863, Mar. 2005.

[10] D. Katz and D. Ward, "Bidirectional forwarding detection (BFD)," RFC5880, Jun. 2010.

[11] J. Moy, "OSPF version 2," RFC2328, Apr. 1998.

[12] M. Goyal, K.K. Ramakrishnan and Wu-chi Feng, "Achieving faster failure detection in OSPF networks," *in Proc. ICC'03*, vol.1, pp. 296- 300, May 2003.

[13] M. Allman, V. Paxson and W. Stevens, "TCP congestion control," RFC 2581 , Apr. 1999.

[14] S. Zander, G. Armitage and P. Branch, "Dynamics of the IP time to live field in Internet traffic flows," Tech. Rep. 070529A, CAIA, Technical Report, May 2007. http://caia.swin.edu.au/reports/070529A/CAIA-TR-070529A.pdf