Protection of RPR strict order traffic

Amund Kvalbein and Stein Gjessing
Simula Research Laboratory, Oslo, Norway
Email: {amundk, steing}@simula.no

***Abstract*— Resilient Packet Ring (RPR, IEEE 802.17) is designed with a protection mechanism aiming at restoring traffic on the ring within 50 ms in case of a link or node failure. However, the total experienced disruption time often exceeds this if strict order delivery of packets is required, since a 40 ms (configurable from 10 ms to 100 ms) topology stabilisation timer is used to avoid packet reordering. In this article, we present three alternative ways to avoid packet reordering for strict order traffic in RPR networks. The three suggested methods are discussed analytically, and simulations are used to compare their performance. Our simulation results suggest that the new methods give up to 90% reduction in packet loss during a failure situation compared to the current RPR standard.**

***Index Terms*— Packet reordering, protection, Resilient Packet Ring**

I. Introduction

Resilient Packet Ring (RPR, IEEE Standard 802.17) is a new standard for packet based, ring topology, metropolitan and wide area networks [1], [2]. Traffic in an RPR network may be sent shortest path along one of the two counter-rotating unidirectional ringlets, which both carry data and control packets. Among the important features of RPR is a fairness mechanism that secures each station a fair share of the available bandwidth, and support for three packet priority classes, named A, B and C. RPR is a buffer insertion ring, and can be implemented with a single or a dual transit buffer design. In a single buffer design, all packets flow through the same transit buffer, termed the Primary Transit Queue (PTQ). In a dual buffer design, the class A packets use the PTQ, while the class B and class C packets use the Secondary Transit Queue (STQ).

RPR is designed with a protection mechanism aiming at restoring traffic within 50 ms in case of a link or station failure. Every station on the ring is reachable through either of the ringlets, which allows one ringlet to serve as a secondary path for traffic on the other. Each station maintains a topology image, with information on the hop count to the other stations on both ringlets. The operations of the RPR protection mechanism is transparent to higher layer protocols like IP, except for the performance degradation that will be experienced following a failure.

In this work, we focus on how packet reordering can best be avoided in an RPR network after a failure situation. We analyse the mechanism used in the RPR standard, and argue that the current mechanism is not satisfactory. Instead, we suggest three different alternatives, all giving reduced packet loss and disruption time compared to the original mechanism. The three new methods and the original are compared analytically and through simulations. The simulations have been performed using a simulator developed at Simula Research Laboratory. The simulator is built on top of the J-sim framework [3].

The rest of this work is organised as follows. In section II, we briefly discuss the potential causes of packet reordering and explain the measures RPR takes to guarantee in order delivery. Our alternative mechanisms are explained in section III. Section IV gives simulation results comparing the different solutions with respect to packet loss. Finally, in section V we summarise and conclude our work.

II. Background

Reordering of packets is a common phenomenon in large networks. Load balancing and parallel links can cause packets with the same source and destination to take different paths through the network. Since these paths can have different delay characteristics, packets can arrive out of order at the destination. Similarly, the inherent local parallelism in switches can be a substantial contributor to packet reordering [4]. Considering the benefits of parallelism with respect to equipment cost, traffic engineering and resilience, it is likely that the amount of parallelism in the Internet will continue to grow.

Protection mechanisms that come into effect when a failure occurs, can also cause packet reordering. This will be the case if packets traversing the new path arrive at the destination, while packets that had already passed the point of failure keep arriving on the old path. In the case of a connectivity failure, packet reordering will usually be combined with some amount of packet loss.

The internet does not guarantee in-order delivery of packets, and higher layer protocols must hence be able to recover from any amount of packet reordering. In practice, however, many services are based on the assumption that packet reordering is sufficiently low. If this assumption is not fulfilled, the consequence with respect to performance can be severe. For example, TCP's fast retransmit optimization treats a reordering spanning more than a few packets as a loss [5]. Since TCP interprets packet loss as a sign of congestion, this has serious implications on throughput performance [6].

A. Packet reordering in RPR

By default, RPR packets are marked *strict order* by a bit in the packet header, meaning that they are guaranteed to arrive in the same order as they were sent. This guarantee is maintained even during a failure situation. Since RPR stations are connected by two counter rotating ringlets, two stations on the ring will still be able to communicate over one of the ringlets after a single link or station failure. RPR has two protection mechanisms, *wrapping* and *steering*. With wrapping, packets reaching a point of failure are wrapped over to the other ringlet, and follows this ringlet to the destination. Wrapping gives a very short response time to failures, and minimizes packet loss. However, because of the packet reordering inherent in wrapping, it is not used for protection of strict mode traffic. Hence, the focus of this work is on the steering protection mechanism.

With steering, when a failure occurs between the source and the receiver, the source station moves traffic over from the normal *primary* ringlet, to the ringlet on which the receiver can still be reached, termed the *secondary ringlet*, as seen in figure 1. This protection mechanism, named *steering*, might introduce packet reordering, if packets traversing the new path experience a shorter buffering delay in the transit nodes than the packets in transit along the old path [7]. Hence, a mechanism is needed in order to guarantee that no packets sent before the failure occurred will arrive at the destination after packets start arriving from the new ringlet.

When a failure occurs in RPR, the stations discovering the failure broadcast a topology update. The stations receiving this update frame use the information to update their topology image. The topology update frame is sent in both directions from the point of failure, using the highest priority. Importantly, the propagation delay for such topology frames is thus bounded to the link delay plus one MTU at each transit station [7].

The current RPR standard [2] defines a *context containment period* to avoid packet reordering. Context containment is declared in a station when a topology frame is received and the topology image updated, and is cleared after a *topology stabilisation timer* expires. The value of this timer is configurable from 10 to 100 ms, with a default value of 40 ms. When the topology stabilisation timer expires, the station calculates a checksum on its topology image, and sends the checksum to its immediate neighbours. If the checksum matches that of the neighbours, context containment is cleared.

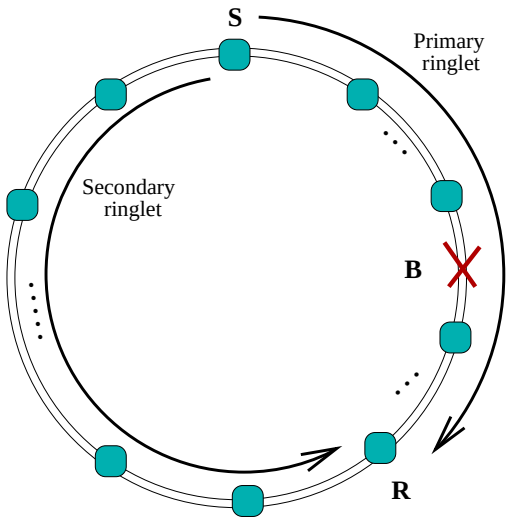

Fig. 1. A generic RPR topology with a source S, a receiver R and a break point B.

During the context containment period, all strict order packets are discarded upon entering and leaving a station, and no new strict order packets are allowed to enter the ring. This is done to make sure that no packets sent in an old topology context are left on the ring to be mixed with packets from a new topology context, potentially causing packet reordering.

The mechanism prescribed by the RPR specification has obvious weaknesses. All strict order packets in transit in the PTQ and STQ are discarded during the context containment period, even those that ideally should not be affected by the failure. Also, new strict order packets are not allowed on the ring during context containment, and are discarded. This gives an unnecessarily high drop count resulting from a protection event. With the default configuration of 40 ms, the topology stabilisation timer will in many cases make the experienced disruption time exceed the 50 ms guarantee [7].

III. Three different improvements

In this section, we present three different ways to reduce the consequences of a link or station failure for strict mode traffic. We refer to the methods as *Automatic*, *Receiver* and *Selective Discard* in the text below.

The *Automatic* method requires only very small changes to the existing specification, while the *Receiver* and *Selective Discard* methods demand some logic to be moved from the transit stations to the receiver station. The *Selective Discard* method gives the most optimal performance, at the cost of some added complexity. None of the three proposed mechanisms rely on packets to be sent shortest path before a failure. Hence, they can easily be adapted to handle network elements that are restored on the ring, as well as failing elements.

A. Automatic setting of the topology stabilisation timer

As mentioned above, the context containment period is used to make sure that no strict order packet in a particular flow sent in a new topology context, will arrive at the destination before a strict order packet from an old context. Once a station enters context containment, all strict mode packets are discarded upon reception and transmission from the station. Hence, the period with context containment must be long enough to let a station on the ring empty its transit buffers. Once all packets have left the transit buffers in a station, the station is in the clear with respect to packet reordering. The aim of the *Automatic* method is to reduce packet loss by reducing the value of the topology stabilisation timer to the minimal safe value.

In the existing RPR specification, the topology stabilisation timer is configurable in the interval 10 to 100 ms. In many cases, even the minimal value of 10 ms is too restrictive with respect to reordering. The optimal value of the topology stabilisation timer depends on the link bandwidth, the distribution of traffic in the different service classes, and the size of the STQ.

We observe that propagation delays between the source, destination and the point of failure do not influence the setting of the stabilisation timer. Remember that a topology update is broadcast on both ringlets with the highest priority after a failure. This ensures that all stations on the ring will receive notification of a failure and enter context containment before any steered traffic can reach that station. Once a station A enters context containment, no more strict order traffic is sent from that station for one context containment period. In other words, once a topology update that triggers context containment arrives on the next downstream station B, no more strict order packets will arrive from A for at least one context containment period. It is then sufficient with a context containment period that is long enough to guarantee that all strict order packets have left the transit queues in station B.

To find the minimal value for the topology stabilisation timer, we need to know the maximal time a data packet

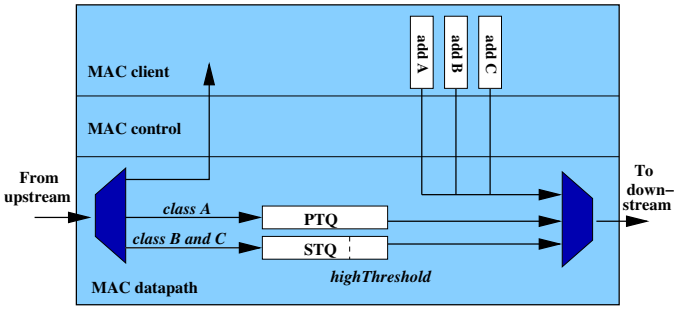

Fig. 2. Simplified view of the transit path for one ringlet in an RPR station, with dual transit queues. Each service class has its own¡ transmit buffer in the MAC client.

can be delayed in the STQ. In the worst case, the STQ might be completely filled up by transit traffic. The time it takes for the last packet in the buffer to be transmitted, depends on the available link rate assigned to the STQ. As seen in figure 2, the output from the STQ has to compete for the outlink bandwidth with the traffic from the PTQ and the add traffic from the MAC client.

The class A packets from the PTQ and class A and B packets from the MAC client has precedence over packets from the STQ in the output selector. Class A and B traffic is shaped at each source station, to make sure no station exceeds its allowed rate. In the worst case, all stations on the ring send class A traffic through this station on this ringlet. At the same time, the MAC client transmits class A and class B traffic at its maximum allowed rate.

Packets from the STQ has absolute precedence over added class C packets from the MAC client as long as the STQ is filled above a *highThreshold*, which defaults to one quarter of the STQ size. When the STQ level drops below this threshold, the available bandwidth is shared equally between the STQ transit traffic and the MAC client class C add traffic.

Equation (1) shows an estimate of the upper bound on the time T needed to completely empty the STQ. The numerator contains the size of the STQ. The part below the highThreshold is counted twice, since the bandwidth is potentially halved below this threshold. The denominator contains the outlink bandwidth[1], minus the worst case bandwidth used for traffic from the PTQ and

[1]Class A traffic is divided into subclasses A_0 and A_1. Subclass A_0 uses reserved bandwidth, which cannot be reused by other traffic classes. *unreservedRate* - $rateA_1$ would therefore be a more precise formulation than outLinkRate -rateA.

class A and B traffic added from the MAC client.

$$T = \frac{(stqSize + highThreshold)}{outLinkRate - \sum_{numStations-1} rate_A - rate_B} \quad (1)$$

In a station with a 1 Gb/s outlink, a STQ size of 256 kB, a total maximum of 10% of the bandwidth used for class A traffic, and a 10% total limit on class B traffic, the calculated maximum time required to flush the STQ, is about 3.2 ms. This is the minimum timeout value we can give the topology stabilisation timer while still fulfilling the strict ordering guarantee.

B. Discarding packets at the receiver

Reducing the topology stabilisation timer to the optimal value, gives a lower packet loss count. However, the mechanism still does not differentiate between traffic from different sources. Packets that were not affected by the failure, will still be discarded during the context containment period. This is clearly sub-optimal: packet flows that are transmitted on the same ringlet before and after the failure, can never experience reordering, and should not be discarded.

Figure 1 shows a generic RPR topology with a source S, a receiver R and a break point B. By default, RPR will send packets from the MAC client shortest path (minimum hop count) around the ring. However, the MAC client may choose to override this, by explicitly defining which ringlet should be used on a per packet basis. This could be done for instance to avoid congestion points on the ring, and thus improve the throughput.

Since the MAC client can choose which ringlet to transmit a packet on, a receiver R will possibly receive packets from a source S on both ringlets when the ring is connected. However, when a connectivity failure occurs, there is only one valid path from S to R. Packets will only be sent along this path, irrespective of the preferences of the MAC client.

The *Receiver* mechanism seeks to avoid dropping unaffected packets, by exploiting the knowledge of where the ring is broken. With the *Receiver* mechanism, transit stations no longer drop strict order packets during context containment. Instead, packets are dropped selectively by the receiving station, based on the source address of the packet. Once the first topology update is received by a source S (remember that two topology updates will be received, one on each ringlet), the topology image is updated so that the source knows which stations are reachable on each ringlet. Similarly the other way around, the topology update gives a receiver R enough information to know which sources S can send packets to R on each ringlet, without passing a break point B. For each ringlet, the receiver can then decide which stations are "valid" sources. With the *Receiver* mechanism, only packets received from stations that are not "valid" are discarded, since only these frames are potentially received out of order.

Transferring the responsibility for discarding packets to the receiver somewhat increases the traffic load on the ring in the time following the failure compared to the standard RPR and *Automatic* methods, since packets are not discarded immediately by transit stations. This also prevents the topology stabilisation time from being set as low as indicated by (1), since packets from an old topology context will exist on the ring for a longer period of time. However, increasing the topology stabilisation timer will not result in increased packet drop count in this situation, since only packets originating from beyond the break point B on the ringlet are discarded. Such packets will stop arriving as soon as the traffic is steered over on the secondary ringlet.

As shown below in section IV, the *Receiver* method reduces the packet drop count, compared to the *Automatic* method. This gain in performance comes at the cost of some added complexity in the receiver station, since the receiver must do a check on each received packet during context containment, to decide whether it should be kept or discarded. On the other hand, complexity is removed from the transit stations, since no check to decide if a strict order packet should be discarded is needed there. No extra state information is required in the stations, as only information already present in the topology image is used to perform the checks.

C. Selective packet discarding at the receiver

With the *Receiver* method, all packets that were sent from beyond a point of failure in an old topology context, are discarded. However, packets sent in an old topology context will not necessarily lead to reordering. Only if packets keep arriving on the primary ringlet even after packets start arriving on the secondary ringlet, will reordering occur. The idea with *Selective Discard*, is to accept packets from beyond a point of failure even during context containment, as long as no packets from that sender has yet arrived on the secondary ringlet.

Selective Discard requires the receiver station to maintain one bit per station indicating whether a packet has been received on the secondary ringlet after the topology was updated. Once this bit has been set, no more strict order packets are accepted from beyond the point of failure on the primary ringlet.

Remember that the MAC client in a sender node can choose which ringlet to transmit on, to achieve load balancing or avoid congestion points. If a source station sends traffic to the same receiver on both ringlets, the performance gain obtained with *Selective Discard* can be reduced, since traffic can be received on the secondary ringlet immediately after a topology update. In figure 1, if the source S sends traffic over both the primary and the secondary ringlet before a failure, packets in transit along the secondary ringlet can start arriving immediately after the topology update. The *Selective Discard* method cannot distinguish the packets that were sent along the secondary path due to MAC client preferences, from the packets that were steered over to the secondary path by the protection mechanism. Only the last category of packets can cause reordering.

One way to improve the performance of the *Selective Discard* algorithm when the source S sends packets to the receiver R along both ringlets, would be to introduce a bit in the frame header marking the packets that are sent a non-default way due to a protection event. This would allow the receiver to distinguish the potentially reordered packets from the packets that were sent along the secondary ringlet due to MAC client preferences. With this improvement, the *Selective Discard* mechanism would give the optimal achievable performance for a reorder avoidance mechanism. Only packets that were actually received out of order would be discarded.

IV. Evaluation

In this section we compare the different mechanisms described above. The measurements are performed with our simulator based on the J-sim framework [3]. We do our measurements on a ring with 64 stations, numbered 0 – 63, and a link length of 40 km. The link capacity is 1 Gb/s, and the STQ buffer size is 256 kByte where not otherwise specified.

A. Optimal topology stabilisation timer

In the *Automatic* method, (1) is used to calculate the timeout value of the topology stabilisation timer. The calculated value is dependent on several variables, among them the size of the secondary transit queue. In figure 3, the calculated timer value is shown as a function of the STQ size, along with the maximum experienced STQ transit delay in the network in our simulations.

The traffic pattern is designed to stress the occupancy level of the STQ, to achieve a high transit delay. In the simulated scenario, node 31 is a “hot receiver”. Stations 0 to 29 send class C traffic to node 31 at the maximum rate allowed by the fairness mechanism.

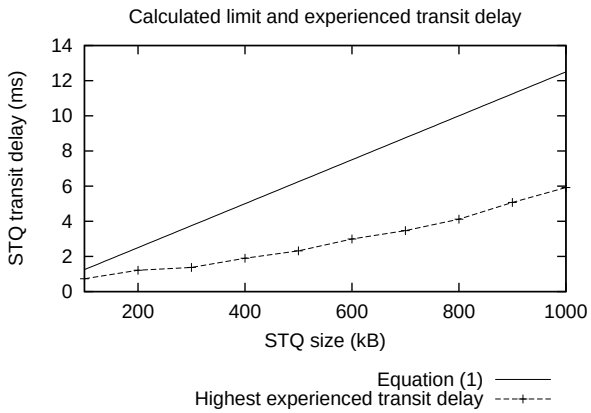

Fig. 3. The maximum experienced transit queue delay in one node for different STQ sizes is plotted, along with the calculated maximum from (1).

When the fairness mechanism has stabilised, all stations start sending class A traffic to node 31 at the maximum rate (*rateA* in (1)). 0.2 seconds later, the ring is broken between nodes 30 and 31. The simulations show that the highest STQ transit delay occurs at node 29, which is the bottleneck node in this scenario. The values plotted are the highest experienced values after running the simulation 100 times with different seeds.

The simulation results show that the experienced STQ transit delay never exceeds the value calculated in (1). In fact, the experienced delays are normally quite far from the theoretical maximum. This is because the chance of actually filling the whole STQ before the fairness mechanism reduces the sending rate of the class C sources is very small. This effect increases when the size of the STQ grows.

B. Comparison of packet loss counts

Figure 4 shows the total number of strict order packets lost in the network after a link failure. Results are given for the different packet reordering avoidance mechanisms, with increasing traffic load.

In this scenario, all stations on the ring send traffic to random receivers. The source picks a random receiver, and sends a stream of packets to that receiver for a random (Pareto distributed) time interval. Each station has ten independent traffic sources, and may send to several receivers at the same time. The traffic is modelled to have self-similar characteristics, as outlined in [8]. The traffic load is controlled by varying the load produced by the individual subsources. Traffic is always sent the default (shortest path) way around the ring.

The original RPR mechanism, using a default topology stabilisation timer value of 40 ms, is shown in the upper graph. Optimising the value of this timer gives

Method	Who discards packets?	Which packets are discarded?
RPR default	Source and transit stations	All strict order packets entering and leaving a station during the context containment period
Automatic	Source and transit stations	All strict order packets entering and leaving a station during the calculated context containment period
Receiver	Receiver station	Packets sent from a station that cannot reach this receiver on the ringlet in question
Selective Discard	Receiver station	Packets sent from a station that cannot reach this receiver on the ringlet in question, if a packet has been received on the secondary ringlet

TABLE I
OVERVIEW OF THE DIFFERENT REORDER AVOIDANCE MECHANISMS.

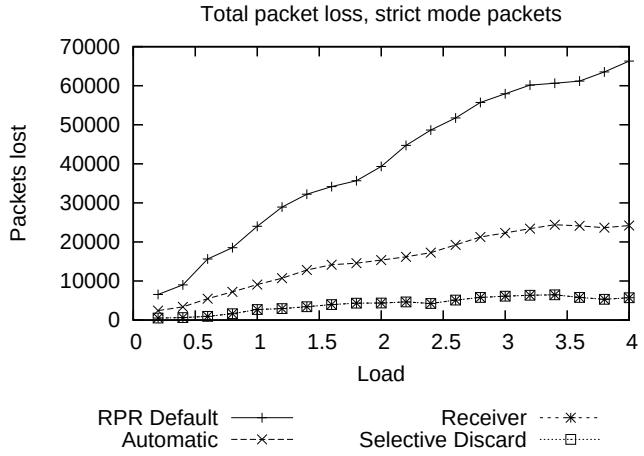

Fig. 4. The total packet loss count in the network is plotted using the original RPR method, *Automatic*, *Receiver* and *Selective Discard*.

a significant improvement, about 60% in our scenario, as seen by the second curve. *Receiver* and *Selective Discard* further reduces the experienced packet loss. The experienced reduction is between 80 and 90%. The difference between the *Receiver* and the *Selective Discard* methods is very small. In our scenario, the difference only appears with very high traffic load, and even then the difference is very limited (14 packets for load 4.0).

Note that for all mechanisms, packets in transit between the source and the failure before the topology image of the source station is updated, will be lost. These packets also contribute to the packet loss count in figure 4.

With the *Receiver* and *Selective Discard* mechanisms, the packet loss count for a specific traffic stream depends on the distance between the packet source and the point of failure. When the point of failure is close to the source, the time it takes for the topology update to reach the source is short. Thus, fewer packets are sent on the broken ringlet and lost. With increasing distance from the source to the failure, more packets are sent on the wrong ringlet before the topology update can take place at the sender. This effect is not seen with the default RPR and the *Automatic* methods, since the context containment period is the same independent of where the failure occurs.

V. DISCUSSION AND CONCLUSIONS

RPR guarantees in order delivery of strict order packets with the same {source, destination, service class} tuple, even during a failure situation. We have shown in this work that this can be done with significantly less packet loss than the method described in the current RPR standard. By transferring the responsibility for enforcing in order delivery to the receiver, fewer flows are affected by a failure.

We have presented three different improvements to the reorder avoidance mechanism. A summary of the different mechanisms is given in table I.

The *Automatic* method seeks to minimise the period when packets are discarded, without compromising on the in order guarantee. This strategy implies setting the topology stabilisation timer lower than the minimum value allowed in the RPR standard, but does not otherwise demand changes to the standard. In our simulated scenarios, packet loss is reduced by about 60%, compared to the default setting og the topology stabilisation timer.

The other two methods move some logic from the transit path in RPR to the receiver station. The *Receiver* method does not require any state information to be maintained at the receiver, while the *Selective Discard* method requires the receiver to know which sources it receives packets from after a topology update. This

is a small cost, but it requires an extra entry in the topology image of the RPR station. As shown by the simulation results above, the difference between these methods with respect to packet loss is minimal, and can only be seen in situations with very high traffic load. These methods reduce the packet loss by almost 90% in our simulated scenarios. In a practical implementation, the *Receiver* method would probably be good enough, and it is unlikely that a vendor would implement the somewhat more complex *Selective Discard* method.

ACKNOWLEDGEMENTS

We are indebted to our colleague, Olav Lysne, who first pointed out the possibility of letting the receiver discard reordered packets.

REFERENCES

[1] F. Davik, M. Yilmaz, S. Gjessing, and N. Uzun, "IEEE 802.17 Resilient Packet Ring tutorial," *IEEE Communications Magazine*, vol. 42, no. 3, pp. 112–118, March 2004.

[2] *IEEE Standard 802.17-2004*, IEEE, June 2004.

[3] H.-Y. Tyan, "Design, Realization and Evaluation of a Component-Based Compositional Software Architecture for Network Simulation," Ph.D. dissertation, Ohio State University, 2002.

[4] J. C. R. Bennett, C. Partridge, and N. Shectman, "Packet Reordering is Not Pathological Network Behavior," *IEEE/ACM Transactions on Networking*, vol. 7, no. 6, pp. 789–798, December 1999.

[5] W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms," in *IETF*, RFC 2001, Jan. 1997.

[6] E. Blanton and M. Allman, "On making TCP more robust to packet reordering," in *ACM Computer Communication Review*, vol. 32, no. 1, jan 2002.

[7] A. Kvalbein and S. Gjessing, "Analysis and improved performance of RPR protection," in *Proceedings of International Conference On Networks (ICON) 2004*. IEEE, November 2004, pp. 119–124.

[8] W. Willinger, M. Taqqu, R. Sherman, and D. Wilson, "Self-similarity through high-variability: Statistical analysis of Ethernet LAN traffic at the source level," *IEEE/ACM Transactions on Networking*, vol. 5, no. 1, pp. 71–86, February 1997.