RRR: Rapid Ring Recovery

Sub-millisecond Decentralized Recovery for Ethernet Ring

Minh Huynh and Prasant Mohapatra
Computer Science Department
University of California, Davis
Davis, CA. USA.
{mahuynh, pmohapatra}@ucdavis.edu

Stuart Goose and Raymond Liao
Technology To Business
Siemens Corporation
Berkeley, CA. USA
{stuart.goose, raymond.liao}@siemens.com

***Abstract*—Ethernet is the indisputable de-facto technology for local area networks due to its simplicity, low cost, and wide-scale adoption. In recent times, Ethernet has entered new networking areas such as Metro Area Network (MAN) and Industrial Area Network (IAN) where specialized protocols dominate the market. In addition to the well known advantages, Ethernet acts as the common platform to integrate multiple protocols. However, Ethernet falls short of the stringent resilience requirements mandated by applications in MEN and IAN, despite progress made by the community on additional standardization. We describe a new approach for swift failure detection and recovery in Ethernet ring topologies called Rapid Ring Recovery (RRR). RRR is based on the novel usage of multiple virtual rings. Our implementation augmenting an off-the-shelf Ethernet switch shows that RRR reconverges after a fault in 294 microseconds while sustaining the loss of only 8 large frames at 95% traffic load.**

Keywords-Resilience; Ethernet; Metro Ethernet Network; Industrial Ethernet Network; rapid recovery; ring; VLAN; tunnel; MAC-in-MAC

I. INTRODUCTION

Quintessentially, Ethernet is a simple networking technology to connect two endpoints at the data link layer. Using Ethernet, a local area network (LAN) can be built and configured in a short amount of time. Gradually, Ethernet is emerging as a significant player in new territories, such as Metro Area Network (MAN) and Industrial Area Network (IAN), where it is predicted to replace the legacy technologies that are currently the major players. MAN has the default recovery time of 50ms as the result of the recovery in legacy optical networks such as SONET/SDH. For Ethernet that operates in MAN, or Carrier Ethernet, it is competitively necessary to follow the performance specification standard defined in the Metro Ethernet Forum (MEF). Industrial Ethernet Network has the most stringent QoS requirements because of the high precision required to perform measurements and to control the plant reliably and safely. Many industrial machines necessitate a real-time synchronization between the master node and the slave nodes. This synchronization constrains the request and response cycle time to a few milliseconds time, and in some cases microseconds range. An operational plant can tolerate a failure in the automation system only for a short amount of time, called grace period. For the plants to be in continuous operation, the recovery time has to be shorter than the grace period. TABLE I. shows the typical recovery time from ranges of applications.

TABLE I. APPLICATIONS TYPICAL RECOVERY TIME

Applications	Grace Time
Web browsing, e-commerce, emails [13]	4s
Gaming, telnet [13]	250ms
Video streaming, video conference [13]	100ms
Audio streaming, VoIP [13]	150ms
Metropolitan Area Network [12]	50ms
Automation management [1]	2s
General automation: process automation [1]	200ms
Time-critical automation: synchronized drives [1]	20ms
Drive control [14]	5-10ms
Motion control [14]	Sub-1ms

RRR answers the need for a rapid recovery for applications in MAN and IAN on native Ethernet platform. Our results demonstrate that RRR can recover from a fault in 294µs with minimal loss ranging from 0 to 36 frames depending on the traffic loads and frame sizes. In addition, RRR prevents TCP timeout from expiring and thus preserves the constant throughput in flows.

The paper is organized as follows. Section II describes the related work, while section III outlines the motivation and advantages of a sub-millisecond recovery scheme for an Ethernet network. Section IV explains in detail the RRR concept and architecture together with an example and theoretical analysis. Experimental results from the RRR prototype are presented in section V, while the concluding remarks are presented in section VI.

II. Related Work

Resilience protocols focused on ring topologies are popular because of the deterministic behavior of the ring. Ethernet Automation Protection Switching (EAPS) [4] in IETF RFC 3619, Metro Ring Protocol [3], and Rapid Ring Spanning Tree Protocol [2] focus on a ring topology for Ethernet based networks. In general, these protocols are similar in that each ring has a root node to manage the ring topology and monitor the ring health. Initially, the root blocks the secondary port on the ring and forwards traffic only on the primary port. The root sends a test packet periodically around the ring. A loss of three consecutive test packets constitutes a failure on the ring. All nodes on the ring must be synchronized to flush the forwarding database before the nodes can resume forwarding. This delay incurs a cost of at most one trip around the ring.

For a no-frame-loss-recovery protocol or "*bumpless*" recovery, a redundant ring network on standby is established as in the case of Parallel Redundancy Protocol (PRP) [1]. An end host is connected to two disjoint networks running in parallel. Both networks do not have any connection between them in order to isolate the fault into one while continuing to forward traffic on the other. An end host having the same MAC address on both interfaces sends a frame onto one network and a duplicate frame onto the other simultaneously. When two frames arrive at the destination, ideally at the same time, one frame is forwarded to the upper layer while the duplicate frame is discarded.

Resilient Ethernet Protocol (REP) [16], constructs a network topology out of REP segment. REP can manage each separate segment, or ring, where each ring can recover by itself in 50ms. Using the Spanning Tree Protocol, REP connects multiple segments into a large topology. Resilient Packet Ring (RPR) is a ring-based protocol that is being standardized in IEEE 802.17 [15]. RPR consists of two counter-rotating rings and each ring is built by point-to-point connections. Unlike FDDI, the destination in RPR removes the frame leaving the remaining part of the ring reusable. This is known as spatial reuse. There is also a time-to-live (TTL) in case no station can recognize the destination address. Unlike SONET where a router wastefully sends a filler packet if it cannot fill up the connection with data, RPR is designed for statistical multiplexing. Resilience in RPR is sub-50ms independent of the physical layer.

Failure Handling Protocol (FHP) [17] broadcasts beacon messages to the selected arbitrator(s) to detect failure in the network based on the principle that these messages will traverse through all the links of the topology. Any missing beacon messages within the detection interval will trigger the recovery mechanism. The global message is used as a means for global recovergence. In contrast, the RRR recovery mechanism occurs locally between the switch and its adjacent neighbors. The remote nodes are completely oblivious to failures. In addition, FHP monitors the network condition via the messages exchange between a group of beacon senders and a group of arbitrators. Each group may comprise many members. The paradigm is similar to a master-slave scheme, as opposed to RRR which is fully decentralized.

Topology Adaptation Network Management Protocol (TANMP) [18] employs failure detection at the end-hosts via a customized dual Ethernet port module before broadcasting the topology change notification to the remainder of the network. Unlike RRR, TANMP does not have any switching/bridging nodes. All end-hosts are linked directly together in a ring fashion. In contrast to RRR, TANMP broadcasts the topology change notification message whenever there is a failure to synchronize the reconvergence.

III. Motivation

A. Recovery Time in Traditional Ethernet

In a conventional Ethernet network, the Spanning Tree Protocol (STP) or IEEE 802.1d [5] has been the de facto protocol for switching frames inside of bridges and switches at the link layer for more than ten years. The STP family includes Spanning Tree Protocol, Rapid Spanning Tree Protocol (RSTP), and Multiple Spanning Tree Protocol (MSTP). STP forms a spanning tree on top of a physical mesh topology using the shortest path to the root node approach. Since there is no time-to-live field in layer2 frames, STP blocks the redundant links to prevent loops. In effect, the loads concentrate on the bottleneck links around the root leaving the network at risk of failures and without load balancing. RSTP and MSTP were designed to mitigate these problems. However, their effective recovery times are 30s-60s for STP and 1-3s for RSTP and MSTP. The performance of these protocols do not meet the applications' requirements in MAN and IAN. As a result, switch vendors invented proprietary protocols to meet their customers' requirement. Often, for a swift recovery of less than 100ms, the network topology is a specialized network with multiple constraints, or another technology other than Ethernet is used. With specialized networks, scalability is the underlying drawback; as for a different link layer technology, it will not be compatible with the customers' existing networks thus forcing an expensive migration to a single vendor domain, or a cumbersome network management nightmare due to the complexity of protocol translation between different platforms and systems.

B. Benefits of RRR

Rapid Ring Recovery (RRR) is a fully distributed protocol, having eliminated global control messages and a central control system. The advantages are:

- no single point of failure, staying true to the spirit of Ethernet
- eliminating the significant long delay required for global control messages to be propagated to every node
- recovery can be achieved in less than 1ms

- easily integrated with the current hardware via software implementation on IEEE802.1Q compatible switches
- does not require duplicate cables or additional hardware for added resilience

IV. RAPID RING RECOVERY

Rapid Ring Recovery (RRR) is a decentralized recovery scheme (no arbitrator or root node) designed for a ring topology in an Ethernet network. The goal is that *only* the two ring nodes adjacent to the fault are involved in detecting and healing the ring, thus reducing the recovery time to less than 1ms and enabling flows to continue with little or no disruption. Healing the ring *locally* obviates the need for global coordination among the ring nodes and the associated long recovery times.

A. RRR Philosophy

In an Ethernet network, the working topology must not have any loop to prevent broadcast storm cause by flooding unknown MAC addresses and infinite looping. Therefore, in an Ethernet ring, a working topology allows all links to participate in frame forwarding except one. Since only one link is blocked to create a working topology, there exist n possible different configurations for a ring topology with n physical links. Each configuration is called a Virtual LAN (VLAN). In Fig. 1, there are four physical links in the ring creating four different possible VLANs. RRR utilizes all n possible VLANs for forwarding traffic. Initially, RRR will choose one VLAN to be the primary forwarding topology. When it is detected that a link on the primary VLAN fails, the RRR-enabled switches on either side of the failure forward the traffic on an alternative VLAN for the remainder of its journey.

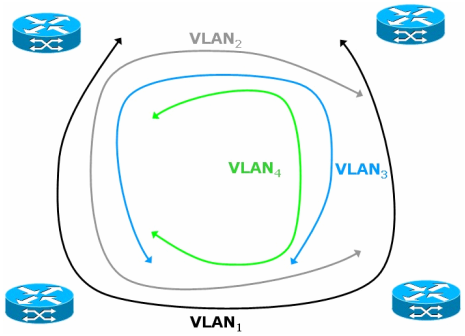

Figure 1. RRR topology management

The key to the rapid recovery of RRR is the local detection and management of the fault. The switches adjacent to the fault can detect and act faster than having to wait for notifications from the root before taking any actions. Autonomously, these switches know the correct VLAN to switch to based on a RRR VLAN switching rule. The rerouted traffic may temporarily traverse on a suboptimal path, but once the reply traffic of the same flow has been received the switches update their forwarding tables and transparently migrates the flows to an optimal path. RRR introduces no additional flooding beyond that of the normal Ethernet behavior. During the recovery phase, it is possible that frames belonging to the same flow exist across two different VLANs en-route to its destination. This is discussed in section V.C.3).

B. Example

Fig. 2, Fig. 3, and Fig. 4 demonstrate an example of the RRR approach when a fault occurs. In this example, A1 is sending to E1 via the path A-B-C-E, as shown in Fig. 2. When a fault occurs on the link E-C, the failure is detected and known only by switch C and E. Frames arriving at A from A1 are fowarded normally until they reach C. When C receives the frames on its ring port, it changes the default VLAN ID on the frames to that of the VLAN ID where the blocked link of that VLAN is that of the failed link in the physical topology. C is the start of a tunnel and marks a frame as a tunneled frame and forwards to all the local ports and back onto the ring port on which it arrived. Arriving at B, the switch checks if the frame is in the tunnel, and if so, simply forwards from one ring port to the other. Address learning is disabled whilst a frame is in a tunnel. Switch A detects that the frame has reached the end of the tunnel because the frame originally entered the ring at this switch. The switch removes the tunnel marker, and forwards the frame onto the other ring port but still ignoring address learning. When D receives the frame, it realizes that this frame is on a backup VLAN. D records the new VLAN ID with the corresponding source MAC address, A1, into a separate table called MAC2VID table. This entry is later used if D forwards to this destination. D enables address learning thus recording that A1 arrived at a ring port on the new VLAN ID. Since the destination address, E1, is not in D's forwarding database for this new VLAN ID, the frame is flooded to all ports except the one on which it arrived – as would be expected with Ethernet. Similarly, E behaves the same as D and floods the frame, whereupon it is received at E1. During this phase of the recovery, the frame traverses a suboptimal path A-B-C-B-A-D-E, as shown in Fig. 3. When E1 replies to A1, switch E recieves an VLAN-untagged frame from E1. E looks for A1 in its MAC2VID table to identify on which VLAN should be used and also on the backup VLAN to determine on which port the frame should be forwarded. At this time, E learns that E1 on the backup VLAN arrived on one of the local ports. Similarly, when the frame arrives at D, D finds a match for the A1 destination. D learns E1 on the new backup VLAN. Finally, the reply frame arrives at A and is forwarded to A1. The return frame educated every switch on the way to establish a new optimal path. Therefore every subsequent frame has been transparently migrated from a temporary suboptimal path to an optimal path for the flow, A-D-E as shown in Fig. 4.

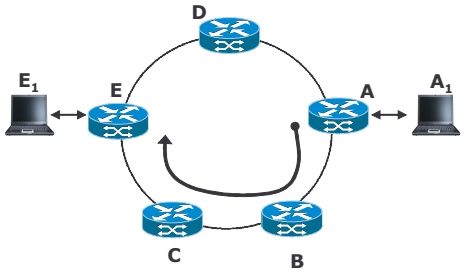

Figure 2. Initial forwarding path

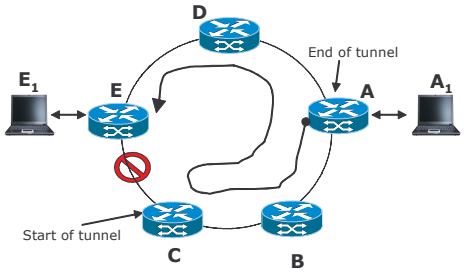

Figure 3. Suboptimal path after the fault

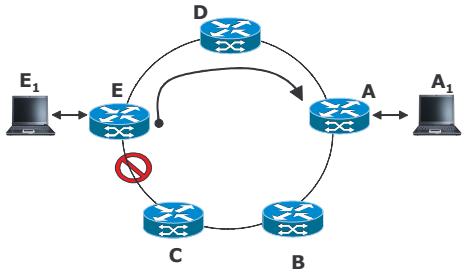

Figure 4. Reconverged path

C. *Algorithm*

Fig. 5 shows the RRR recovery algorithm for when a frame arrives at a switch. The algorithm is performed on a per frame basis. Depending on whether the frame originates from local or ring ports, and whether the switch is adjacent to a fault or not, each frame arriving has four VLAN ID (VID) selections:

- Default VID
- VID from MAC2VID table
- VID from the frame tag
- Backup VID

The backup VID of the switch adjacent to the fault is used whenever a frame passes through that switch. If the MAC address of the end-host is found in the MAC2VID table, the VID associated with that MAC address is used instead. In normal cases, the default VID is used if a frame arrives from a local port; else the VID from the frame tag is used if a frame arrives from a ring port. During a fault when a frame is turned around, each switch inside the tunnel must not perform address learning to prevent poisoning of the MAC address forwarding table. The tunnel bit is used to indicate whether a frame is still inside the tunnel or not. The flooding proceeds as normal, except when the Bridge-Source address is the same as the current Bridge address. Finally, loopback is an important feature that allows a frame to reroute by retracing part of the path in the face of failure.

```
Let M be the set of all end host MAC addresses in the
    MAC2VID table
Let B-SA be the Bridge Source Address where the frame
    ingress the ring
Let VID be the current VLAN ID of the frame
Let backup_VID be the blocked VLAN ID on the faulty link
Let f be the end host source MAC address
Let MAC2VID(f) return the VID associated with f MAC address

if frame arrive from local port
     encapsulate Mac-in-Mac header
          B-SA <= switch MAC address
       if switch adjacent to fault
               VID <= backup_VID
          else
               if f ϵ M
                    VID <= MAC2VID(f)
               else
                    VID <= default VID
else frame arrive from ring port
     if switch adjacent to fault
          Mark tunnel bit on
          VID <= backup_VID
     else
          VID <= Bridge VID from frame tag

if tunnel bit on
    Address learning <= false
    if switch at end of tunnel
          turn off tunnel bit
else
    learn address <= true
    if VID == backup_VID
         record f and VID to MAC2VID table

  if VID == backup_VID
       If f ϵ FWD
            forward to FWD(f)
       Else:
            If f ϵ FWD and FWD(f) == local port
                 forward to local port
            Else:
                 If B-SA == switch MAC address
                      Flood on ring ports only
                 Else
                      Flood ring and local
  else
      If f ϵ FWD of default VLAN
           forward to FWD(f)
      Else
           If B-SA == switch MAC address
               Flood on ring ports only
           Else
               Flood ring and local

  if VID changed
      loopback on ring port <= allowed
  else
      loopback on ring port <= disallowed
```

Figure 5. Recovery algorithm

D. *Analytical Performance*

During a fault, the RRR recovery time depends on the location of fault with respect to the VLAN. Fig. 6 and Fig. 7 show the worst and best case scenario, respectively. A worst case recovery requires the traffic to traverse suboptimally over more than half of the ring, while a best case delivers the frame to the adjacent neighbor. Generally, the recovery time of RRR is the *[processing_time] + [propagation_time] + link_down_detection_time*. The equations for the RRR recovery time is as follows:

Best Time = [switch time *2] + [Ring Perimeter*cable propagation delay*1] + link down detection (1)

Worst Time = [switch time *(2*number of ring nodes – 2)] + [(ring perimeter*cable propagation delay)*(2*number of ring nodes -3)] + link down detection (2)

Average Time = [switch time * number of ring nodes] + [ring perimeter*cable propagation delay*(number of ring nodes - 1)] + link down detection (3)

where:

- *switch time* is the time for a chipset to push a frame through its switching fabric.
- *ring perimeter* is the total length of the cables forming the ring.

For example, given the following parameters values the best, worst and averages case results can be seen:

- Number of ring nodes = 10
- Ring perimeter = 0.5km
- Cable propagation delay = 5µs/km
- Link down detection = 230µs
- Switching time = 5µs for a small frame size

Best = 5*2 + 0.5*5*1 + 230 = **242.5µs**

Worst = 5*(2*10-2) + 0.5*5*(2*10-3) + 230 = **362.5µs**

Average =5*(10) + 0.5*5*(10-1) + 230 = **302.5µs**

E. System Architecture

The RRR algorithm was implemented on an Altera Cyclone III FPGA board. The FPGA board is connected to a MoreThanIP QuadPhy daughter card, through which the FPGA to can talk to a commodity Gb Ethernet switch via a regular Ethernet cable. This can be appreciated in Figure 8. The advantage of this configuration is that RRR does not impinge on the switch architecture but serves simply as an extension. Another advantage is that RRR can be used with any 802.1Q compatible switches. All traffic originating from the ring ports and all traffic destined to the ring are forwarded automatically to the FPGA to be processed by RRR while leaving local LAN traffic untouched. This reduced unnecessary load upon the FPGA. Following the RRR processing, a frame, it is returned to the switch for transmission to the destination ports. As such, RRR is transparent to the switch leaving it the freedom to run other features and services such as QoS, LLDP, and others. RRR processing does introduce additional latency, but as shown in section V.B the delay is negligible and does not impact network operation.

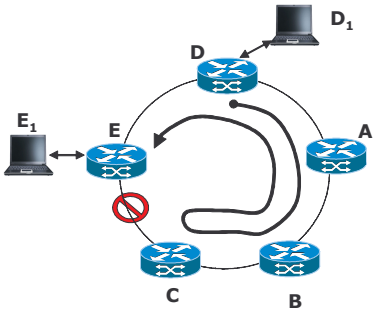

Figure 6. Worst Cast Recovery Scenario

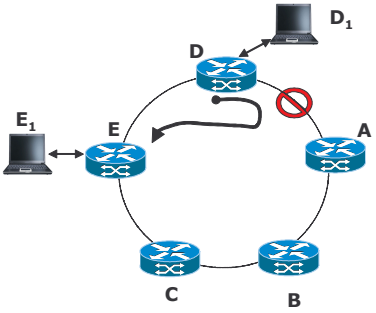

Figure 7. Best Case Recovery Scenario

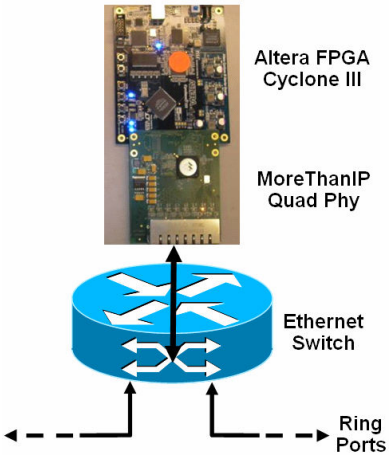

Figure 8. RRR setup

F. Technical Details

1) Initialization.

The VLAN configuration is a key element of a robust topology. The assignment during RRR initialization is automated in a distributed manner by following a simple rule called "*to-the-left*." Since there are only two ring ports on a switch, let the lower ring port number be the 'left" side and the other ring port be the "right." During the connection of the physical cables between the ports to form the ring, the "right" side port of a switch connect to the "left" side port of its neighbor switch, and so on around the ring. The auto-assignment of the blocked VLAN on a link is performed by a switch only on its left ring port. Once a switch has determined the VLAN ID for its left ring port, it passes on this information to the adjacent switches. Then the

information is propagated through the ring. In case there is a collision in the VLAN ID chosen by neighboring switches, the switch with the higher priority retains its chosen number. The switch priority is determined in the same way as the IEEE 802.1D Spanning Tree Protocol. Any switch with a lesser MAC address has higher priority. In addition, there is a 12bit priority bit concatenated to the beginning of the MAC address for flexibility if switch priorities must be specified.

2) Link Down Detection

For a duplex copper gigabit physical interface with error correction, it is complex to detect a link failure. Since it is always receiving both the data sent by the other end as well as its own transmission, when a link is disconnected the link is not receiving data from the neighbor but it still receives its own transmit signal. Therefore, relying on signal detection is not enough for fault detection. The IEEE 802.3 Clause 40 specified the maximum wait time for a physical interface failure detection of a duplex copper gigabits link is 750ms±10ms [6], which is an unacceptably long delay for critical operations in some Industrial Networks. Therefore, a custom Link Down Detection (LDD) mechanism was developed for RRR.

As the LDD time is the dominant component of the recovery time, the LDD mechanism must be fine tuned to achieve the fastest recovery without compromising the stability of the system. In addition, it is important that the LDD mechanism does not consume bandwidth needed for application traffic when the network is healthy. The RRR LDD mechanism requires each switch to maintain communication with its two neighbors. As long as there is traffic being forwarded between the switches above a threshold rate, no explicit LDD mechanism traffic is required as the network is evidently functioning properly. However, if no traffic is present, or the rate drops below the threshold, then the RRR LDD mechanism is activated.

Each switch monitors the activity on its two ring ports. After receiving a frame on a ring port, a switch starts its *Receive_Timer_Portx* where x is the port number of the ring port in question. Whenever a frame is received on this port, this *Receive_Timer_Portx* is reset. If the *Receive_Timer_Portx* expires before a frame is received, then the link is declared to be down. Concurrently, the switch keeps track of another *Receive_Timer_Port* for the other ring port. Each ring port also has a *Send_Timer_Portx*. After transmitting a frame out on a ring port, the *Send_Timer_Portx* for this ring port is started. Whenever there is a frame destined to go out this port_*x*, the *Send_Timer_Portx* is reset. If this *Send_Timer_Portx* expires before the switch sends a subsequent frame, a *Keep_Alive* frame is sent to the corresponding ring port to notify its adjacent neighbor that the link is up to prevent its neighbor's *Receive_Timer_Portx* from expiring. The smaller the *Receive_Timer_Port* value, the faster the switch can detect the fault. However, there exists a tradeoff between a fast fault detection and a false positive caused by a transient error on the link.

3) Per-link VLAN for failover

As mentioned in the previous section, RRR chooses the per-link-VLAN approach for failover. Each link is associated with a VLAN for which traffic carried on this VLAN is blocked from using this link. When a fault occurs, the switch makes the decision to switch the frames, originally intended for the faulty link, to the VLAN ID that is blocked on the faulty link. Healing the ring locally eliminates the need for the conventional global coordination among the nodes and the associated long recovery times. This scheme necessitates the setting up of *n* VLANs where *n* is the number of links in the physical topology. The local detection and dynamic switching to backup VLANs enables a rapid recovery. In addition, this approach enables more than one VLAN to be traversed while a flow is en-route to its destination in the face of failure. Once the source receives the reply traffic, all switches along the new path will have gracefully converged to carry subsequent traffic optimally.

4) Tunneling for Loop back traffic

When traffic is rerouted during a fault, a loopback is formed on the path where traffic is sent back on the port from which it arrived. In traditional Ethernet, this behavior is disallowed to prevent poisoning of the forwarding table. A scenario of the forwarding table being poisoned is when the adjacent switch sees the loopback traffic, it will have learned these MAC addresses arrived on this ring port which is the incorrect information. If subsequent frames destined for these MAC addresses arrive at this switch, it will attempt to send out on the wrong port resulting in dropped traffic. However, RRR enables this rogue behavior to support the swift recovery. To resolve this conflict, RRR prohibits address learning at switches when the traffic is in the loopback mode. This concept is similar to tunneling. The tunnel starts at the switch that is adjacent to the fault where the traffic is initially turned around and it ends at the switch that originally introduced this flow of traffic into the ring.

To keep track of the loopback activities without imposing additional burden on the switch, RRR utilizes the upcoming standard IEEE 802.1ah Provider Backbone [7], also known as MAC-in-MAC. Inside the tunnel, the intermediate switches simply forward frames from one ring port to the next ring port. They do not flood to any local ports. The switch at the start of the tunnel encapsulates the necessary header onto the frames, marks the frame as inside the tunnel, and rewrites the VLAN ID field to the new VLAN ID - the VLAN ID that is blocked on the failed link. The switch also floods these frames to its local ports. The switch at the end of the tunnel removes the encapsulated header and tunnel marker, and forwards the traffic to the other ring port. A switch determines that it is the penultimate switch of a tunnel for a frame when its MAC address matches the source MAC address in the MAC-in-MAC header. While a frame is inside the tunnel, switches are prohibited from learning the source MAC address of the frame to prevent forwarding table poisoning. The address learning resumes as normal once the frame emerges from the tunnel.

Fig. 9 shows the header of an Ethernet frame with the MAC-in-MAC header encapsulated as it is entering the traffic in the ring. There are four different scenarios for a frame inside a switch:

- ingress from the ring – egress to the ring,
- ingress from the ring – egress to LAN,
- ingress from LAN – egress to the ring,
- ingress from LAN – egress to LAN.

All frames that enter the ring are encapsulated with the MAC-in-MAC header regardless of the ring health. The ring ingress switch inserts its MAC address to the B-SA field so that if it sees any frame matching the B-SA with its own MAC address, it knows that it is the switch at the end of the tunnel; it can then remove the tunnel bit which is the most significant bit in the I-SID. The VID ID for RRR topology management purpose is stored in the B-VID, leaving the C-VID untouched for the user's own VID configuration. The MAC-in-MAC header is removed before forwarding to a LAN.

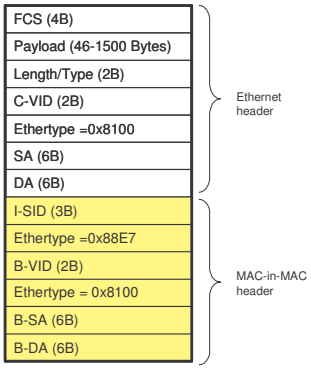

Figure 9. MAC-in-MAC header

5) MAC2VID Table and the graceful reconvergence

In addition to the usual forwarding table in every switch, RRR introduces an additional MAC2VID table in order to divert traffic to the correct VLAN after a failure for a graceful reconvergence, thus avoiding any global messaging and unnecessary flooding. After exiting the tunnel, every subsequent switch until the ring egress switch will see that the frame is traversing on a backup VLAN. They each record the backup VLAN ID associated with the end host source MAC address on the frame into the MAC2VID table. Using this table, subsequent frames destined for this MAC address will use the newly recorded backup VLAN ID. Intuitively, the MAC2VID table is a way to record that the path to a MAC address has changed due to a fault along the original path; and to reach this destination the switch has to send on a new VLAN that all the switches along this new path have established. The table ensures that after traffic has flowed from node A to node Z, the reply traffic from node Z to node A can reach node A on a new optimal established route without having to flow on the loopback or discovering the new path via flooding.

6) Addition and Removal of nodes

The addition of a new switch requires a new VLAN to be instantiated to protect the newly added link. The ring will enter a process similar to the initialization phase to obtain a VLAN ID for the newly added link. During the process, one of the existing VLANs is affected because one end of an existing link must be disconnected and reattached to the new switch. The new switch and its two adjacent neighbors will negotiate to see which ring port on the new switch will use the existing VLAN based on the priority of the port. The remaining port on the new switch will assume the new VLAN. The VLAN information is then propagated on to the rest of the ring. Meanwhile, the traffic on the ring in the default active VLAN will continue to be forwarded.

The removal of a switch is managed identically to that of a failed switch or two links failing. In this case, two VLANs will be affected, but only one VLAN is removed from the active topology. The port priorities on the adjacent neighboring switch of the removed node are examined to determine which VLAN can stay and which must be removed. Following the removal, the VLAN information is propagated throughout the ring and all entries relating to that VLAN are flushed from the forwarding table and MAC2VID table.

V. EVALUATIONS

A. Experimental Setup

The test bed comprised six off-the-shelf Gb Ethernet switches each augmented with a RRR FPGA board (as illustrated in figure 8) connected in a ring topology. Each ring port is 1Gbps; there are 8 local ports on a switch, each having capacity of 100Mbps. All traffic flows are bidirectional with a single source-destination pattern, and a multiple source-destination pattern. In the single source-destination pattern, there are four pairs of source and destination nodes sending in bidirectional traffic, as shown in Fig. 10. This scenario concentrates high throughput on the same switches to evaluate the correctness of the algorithm, since the traffic flows are more deterministic. In the multiple source-destination scenario, the source-destination pairs are as follows (A1, D1), (C1, F1), (B1, E1), and (B2, E2) as shown in Fig. 11. This scenario is to emulate more typical balanced traffic flowing across the network. Each flow is UDP traffic sent with the following traffic loads on the local ports: 10%, 25%, 50%, 80%, and 95%. Each test runs for 20 seconds. The traffic generator and data collection is performed via the Spirent Smartbits 600/6000 [8] machine, an industry standard for network performance analysis.

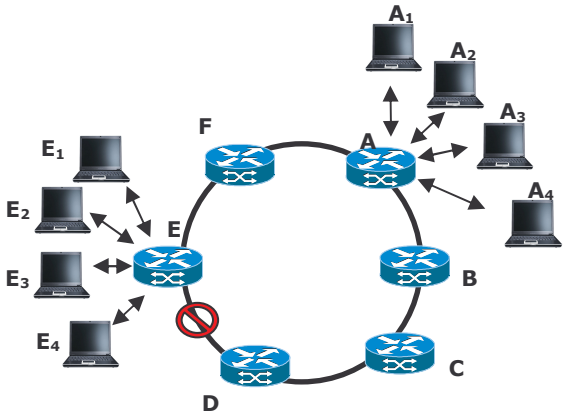

Figure 10. single source-destination scenario, highly concentrated throughput in a deterministic enviroment

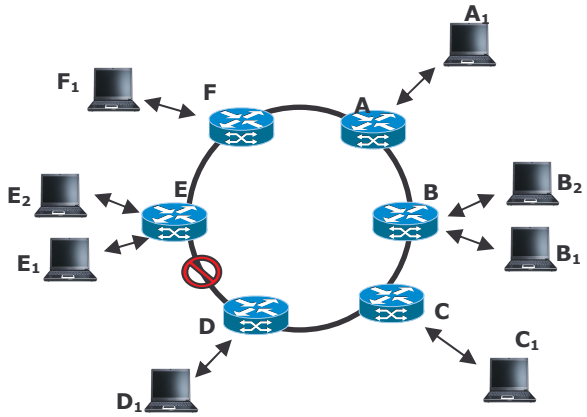

Figure 11. Multiple source-destination scenario, a general balanced traffic network

B. RRR Normal Behavior

Fig. 12 shows the latency of the traffic flows during normal no fault conditions for both small frame size of 132B and a large frame size of 1460B. The graph shows the overhead latency added by RRR compared with RSTP, the existing Ethernet ring protocol running on the Ethernet switches. The additional latency is expected because frames are forwarded by the switch to the RRR FPGA board for processing and returned to the switch for forwarding. In effect, each frame passes through a switch twice. Naturally, this additional latency would be eliminated if RRR were to be implemented within the switching ASIC. Nevertheless, the added latency is constant in all traffic loads, 15µs for small frame size and 80µs for large frame size. Since the additional delay is minimal, it does not impact the traffic flow.

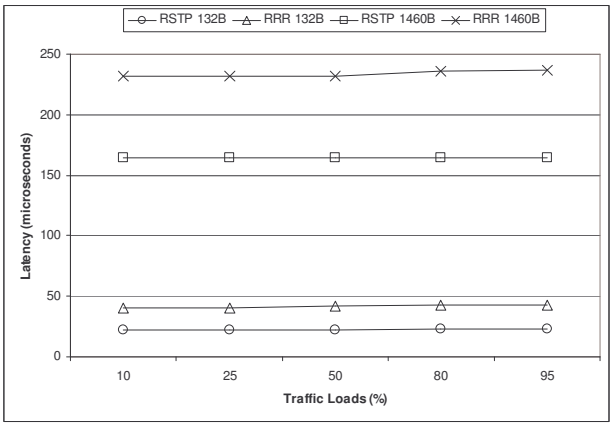

Figure 12. Latency under no fault condition

C. Failure Scenarios

1) Recovery time

TABLE II shows the recovery time between different Ethernet protocols, where the time is divided into link down detection and recovery. The recovery time interval starts immediately after the switch detects a fault until the first frame is received at the destination. The IEEE 802.1s, Rapid Spanning Tree Protocol (RSTP) [10], is the standard protocol to manage a topology in an Ethernet network. The performance of RSTP is documented to reconverge after a fault between 1s and 3s. In our experiment, RSTP reconverged after 10s.

Resilient Packet Ring (RPR) [15] inherits characteristics from SONET/SDH network with a recovery time of 50ms. For every path in RPR, there is a standby backup path where both paths are configured via MPLS. Resilient Ethernet Protocol (REP) [16] constructs a network topology out of an REP segment. REP is documented to reconverge each separate segment in ~50ms.

Viking [9] has a client-server architecture to detect network conditions and react to failures. Viking pre-calculates the forwarding topology to avoid highly congested links. The server instructs the nodes to converge to the selected topology. Viking is slower with a total recovery time between 300ms to 400ms.

Failure Handling Protocol (FHP) [17] broadcasts beacon messages to the selected arbitrator(s) to detect failure in the network based on the principle that the messages will traverse through all the links of the topology. Any missing beacon messages within an interval will trigger the recovery mechanism, which was shown to be between 7ms to 29 ms experimentally.

Topology Adaptation Network Management Protocol (TANMP) [18] employs failure detection at the end-hosts via a customized module before broadcasting the topology change notification to the remainder of the network. The recovery time is approximately 1ms.

Finally, RRR detects the fault locally using the scheme described in section IV.F.2). The long latency of the previously mentioned protocols is due to the waiting period for all of the nodes in the network to receive the global failure notification messages. In contrast, RRR is a distributed protocol where each node reacts on-the-fly to the failure reducing the recovery time significantly. A link is declared as failed after missing three hello packets from its neighboring switch, a period of 0.225ms. Following this timeout, all frames destined for the failed link arriving at this switch adjacent to the fault will be forwarded onto the backup VLAN towards their destination. The first frame arriving at the destination took 0.069ms for a total reconvergence time of 0.294ms. During this time, the frames still traverse on the suboptimal loopback path until the sender receives the first frame from the receiver whereupon traffic migrates to the optimal path.

2) Frame loss

TABLE III. and TABLE IV. compares the frame loss incurred by a failure between RRR and RSTP. Since RRR recovers much quicker than RSTP, the frames lost in a RRR network is significantly lower than that of RSTP. The frame loss in RRR occurs during the link down detection period which is 0.225ms. As the data rate increases, the higher the frame loss. However, the percentage of frame loss stays relatively constant for both RRR and RSTP.

TABLE II. COMPARISON OF RECOVERY TIME

Protocols	Link Down Detection	Recovery	Total
RRR	0.225ms	0.069ms	0.294ms
RPR	N/A	N/A	50ms
Cisco REP	N/A	N/A	50ms-250ms
Viking	N/A	N/A	300ms - 400ms
RSTP	750ms	N/A	10s (experiment)
TANMP	N/A	N/A	~1ms
FHP	5ms-10ms	N/A	7ms-29ms

TABLE III. COMPARISON OF FRAME LOSS FOR SINGLE SRC-DST

Protocols	Traffic loads (%)	Frame size (Bytes)	Frame loss count	Bytes loss (Mbps)	Loss percentage (%)
RSTP	10	132	63728	67.30	4.84
		1460	6830	79.77	5.05
	25	132	163313	172.46	4.96
		1460	16347	190.93	4.84
	50	132	322966	341.05	4.91
		1460	32994	385.37	4.88
	80	1460	52313	611.02	4.84
	95	1460	62728	732.66	4.89
RRR	10	132	13	0.014	0.0020
		1460	0	0	0
	25	132	36	0.038	0.0022
		1460	6	0.070	.00036
	50	132	76	0.080	0.0023
		1460	7	0.082	0.0021
	80	1460	11	0.128	0.0020
	95	1460	16	0.187	0.0025

TABLE IV. COMPARISON OF FRAME LOSS FOR MULTIPLE SRC-DST

Protocols	Traffic loads (%)	Frame size (Bytes)	Frame loss count	Bytes loss (Mbps)	Loss percentage (%)
RSTP	10	132	32599	34.42	2.48
		1460	3341	39.02	2.47
	25	132	81987	86.58	2.49
		1460	8266	96.55	2.45
	50	132	163953	173.13	2.49
		1460	16654	194.52	2.46
	80	1460	26912	314.33	2.49
	95	1460	31946	373.13	2.49
RRR	10	132	4	0.004	0.0006
		1460	0	0	0
	25	132	17	0.018	0.0010
		1460	2	0.023	0.0012
	50	132	37	0.039	0.0011
		1460	4	0.047	0.0012
	80	1460	5	0.058	0.0009
	95	1460	8	0.093	0.0012

3) Impact on TCP traffic

As mentioned, RRR can forward frames to the destination via two different VLANs during a fault. As a result, frames from the same flow might take different paths to the destination and arrive out-of-order. Certain applications or protocols might be affected by out-of-order arrival. For example, out-of-order frames in TCP could trigger the fast retransmission mechanism and reduce the effective goodputs by cutting the transmission window size. However, the RRR results shown in Fig. 13 show that the number out-of-order frames caused by RRR are insignificant and remain relatively small as the load increases, eventually reaching a plateau in the multiple src-dst scenario.

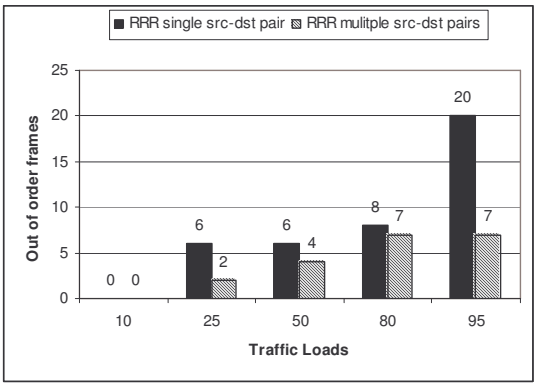

Figure 13. Out of order frames

While Fig. 13 shows that out-of-order frames are introduced during a fault, these frames do not affect the TCP performance by falsely triggering the fast retransmission mechanism. Fig. 14 shows that RRR remains at a constant goodput level during the failure period, which means that the TCP timeout never expired. The timeout for TCP is derived dynamically based on the Round Trip Time (RTT) of the received packets as follow [11]:

$$\begin{aligned} EstRTT &= (1-\alpha)*EstRTT + \alpha*SampleRTT \\ &= (1-0.875)*331.76 + 0.125*331.4 \\ &= 331.715\mu s \end{aligned} \quad (4)$$

$$\begin{aligned} DevRTT &= (1-\beta)*DevRTT + \beta*|SampleRTT - EstRTT| \\ &= (1-0.75)*0.314 + 0.25*|331.4 - 331.715| \\ &= 0.31425 \end{aligned} \quad (5)$$

$$\begin{aligned} Timeout &= EstRTT + 4*DevRTT \\ &= 331.715 + 4*0.31425 \\ &= 332.972\ \mu s \end{aligned} \quad (6)$$

Since RRR recovers in 294µs, the TCP timeout never expires. Therefore, the transmission window size was never reduced and TCP never entered the slow start phase. As RSTP recovers in 10s, the TCP timeout expires and the flow entered the TCP slow start phase. As a result, Fig. 14 shows RSTP from the start of the fault detection until TCP pickup the sending rate again for ~10s, from 3s to 13s.

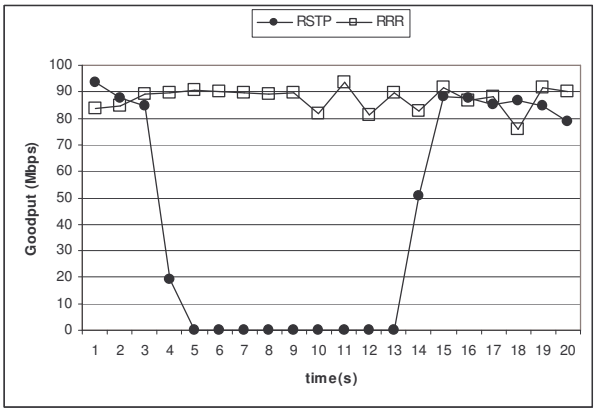

Figure 14. RSTP vs. RRR failover with TCP traffic

VI. CONCLUSION

This paper reports a novel usage of multiple virtual rings, called Rapid Ring Recovery (RRR), to improve the performance of fault recovery in Ethernet networks. The RRR architecture is implemented on a FPGA board and integrated with a commodity Gb Ethernet switch supporting 802.1Q. RRR does not impinge on the native switch architecture, but serves simply as an extension. The results reported demonstrate that RRR has a fault recovery performance of 294µs, frame loss approaching zero, and the ability to maintain a constant TCP throughput. With a fully distributed protocol, RRR does not have single point of failure.

REFERENCES

[1] IEC 62439 "High Availability automation Networks" Feb 2008

[2] G. Holland. "Carrier Class Metro Networking: The High Availability Features of Riverstone's RS Metro Routers" Riverstone Networks Technology whitepaper #135. http://www.riverstonenet.com

[3] Foundry Networks "Foundry Switch and Router Installation and Basic Configuration Guide - Chapter 13 - Configuring Metro Features" http://www.foundrynet.com/services/documentation/sribcg/Metro.html

[4] S. Shah, M. Yip "Extreme Networks' Ethernet Automatic Protection Switching EAPS" RFC 3619

[5] IEEE Information Technology, Part3: Media Access Control (MAC) bridges, ISO/IEC 15802-3, ANSI/IEEE Std 802.1D, 1998

[6] IEEE 802.3 LAN/MAN CSMA/CD (Ethernet) Access Method

[7] IEEE 802.1ah Provider Backbone Bridges June 2008

[8] Spirent Smartbits http://www.smartechconsulting.com

[9] S. Sharma, K. Gopalan, S. Nanda, T. Chiueh, Viking: a multi-spanning-tree Ethernet architecture for metropolitan area and cluster networks, in: Proceedings of IEEE INFOCOM 2004

[10] IEEE Standard for Local and metropolitan area networks - common specifications. Part 3: Media Access Control (MAC) bridges - Amendment 2: Rapid Reconfiguration Amendment to IEEE Std 802.1D, 1998 Edition, IEEE Std 802.1w-2001.

[11] J. Kurose and K. Ross "Computer Networking: A Top-Down Approach Featureing the Internet" 2nd Edition 2003. Pearson Education, Inc.

[12] Metro Ethernet Forum (MEF). "Requirements and Framework for Ethernet Service Protection in Metro Ehternet Networks" Technical Specification MEF2. www.metroethernetforum.org Feb 2004

[13] ITU-T Recommendation G.1010: "End-user multimedia QoS categories" www.itu-t.org

[14] M. Rostan "Industrial Ethernet Technologies: Overview" ETG Industrial Ethernet Seminar Series, Nuremberg, Nov. 2008

[15] F. Davik, M. Yilmaz, S. Gjessing, N. Uzun, "IEEE 802.17 Resilient Packet Ring Tutorial" IEEE Communications Magazine. March 2004

[16] Cisco "Cisco Resilient Ethernet Protocol" http://www.cisco.com/en/US/prod/collateral/switches/ps6568/ps6580/prod_white_paper0900aecd806ec6fa.pdf 2007

[17] J. Farkas, C. Antal, L. Westberg, A. Paradisi, T. Tronco, V. de Oliveira "Fast Failure Handling in Ethernet Networks" IEEE ICC 2006

[18] G. Yoon, D. Kwon, S. Kwon, Y. Park, Y. Lee. "Ring Topology-based Redundancy Ethernet for Industrial Network" SICE-ICASE International Joint Conference 2006

Minh Huynh received his B.S. degree in Computer Science from the University of California at Davis in 2002, and currently is a PhD candidate. His research interest is on Metro Ethernet Network with focus on resilience, network load balancing, QoS, and security.

Between degrees, Minh Huynh was a scholar researcher at Lawrence Livermore National Lab and innovator at Siemens Technology-to-Business. He was on the student volunteer committee for Globecom 2006 in San Francisco.

Stuart Goose has B.Sc. (1993) and Ph.D.(1997) degrees in computer science both from the University of Southampton, United Kingdom.Following a postdoctoral position at the University of Southampton, he joined Siemens Corporate Research Inc. in Princeton, New Jersey, USA. He held various positions in the Multimedia Technology Department, leading a research group exploring and applying various aspects of Internet, mobility, multimedia, speech, and audio technologies. His current position is Director of Venture Technology at Siemens Technology-To-Business Center in Berkeley, California, USA. This involves scouting for disruptive technologies from universities and startups, running projects to validate the technical and business merit of technologies, and, if successful, transferring the technologies to the business units within Siemens for commercialization.

Prasant Mohapatra received his doctoral degree from Penn State University in 1993, and received an Outstanding Engineering Alumni Award in 2008. Dr. Mohapatra is a Fellow of the IEEE and is currently the Tim Bucher Family Endowed Chair Professor and the Chairman of the Department of Computer Science at the University of California, Davis. In the past, he has been on the faculty at Iowa State University and Michigan State University. He has also held Visiting Scientist positions at Intel Corporation, Panasonic Technologies, Institute of Infocomm Research (I2R), Singapore, and National ICT Australia (NICTA). He has been a Visiting Professor at the University of Padova, Italy and Yonsei University, South Korea. He was/is on the editorial board of the IEEE Transactions on Computers, IEEE Transactions on Mobile Computing, IEEE Transaction on Parallel and Distributed Systems, ACM WINET, and Ad Hoc Networks. He has been on the program/organizational committees of several international conferences. He served as the Program Vice-Chair of INFOCOM 2004 and the Program Chair of SECON 2004, QShine 2006, and WoWMoM 2009. He has been a Guest Editor for IEEE Network, IEEE Transactions on Mobile Computing, IEEE Communications, IEEE Wireless Communications, and the IEEE Computer.

Raymond Liao has M.A.Sc. (1993) from University of Toronto and Ph.D. (2002) from Columbia University in Electrical Engineering. Between degrees, Dr. Liao worked as a Network Performance Analyst at Newbridge Network. As an innovator at Siemens Technology-To-Business Center, he commercialized his Ph.D. research into two Siemens industrial networking products. Then, as Director of Venture Technology, he managed the lifecycle of innovation projects: from scouting innovation outside of Siemens, funding seed-stage startups, to exiting from our incubation center.
His responsibilities as Director of Technology & Innovation (CTO) at Siemens Industrial Automation/Drive Technology include advanced technology development, technology roadmap and research cooperation for the industrial networking product lines.