University of Oslo
Department of Informatics

Bridging in RPR networks

Amund Kvalbein

Evaluation of an enhanced bridging algorithm

May 1, 2003

Preface

The hardest thing in life to learn
is which bridge to cross and which to burn.

—David Russell

This thesis is presented for the degree of Cand. Scient. at the Department of Informatics, University of Oslo.

Most of the work has been done at the Simula Research Laboratory at Fornebu, Oslo. I wish to thank the Simula community for providing good working conditions and many nice lunches during the project period.

Fredrik Davik and professor Stein Gjessing have been my supervisors in this work. I never found their office doors closed. A warm thank goes to them for their cooperation and help during this period.

Also, thanks to professor Ørnulf Borgan for help with statistical issues in this work.

Oslo, May 2003

Amund Kvalbein

Contents

Chapter 1

Introduction

1.1 Motivation

Resilient Packet Ring (RPR) is a new standard for a packet based ring network currently under development. RPR is primarily intended to be a technology for the metro environment, supporting link capacities up to multiple gigabits per second. One of the important characteristics of RPR is *spatial reuse*, which allows data frames in the network to traverse only the shortest path between the source and destination nodes on the ring.

The main topic of this work is how several such RPR rings can be interconnected by *bridges*. Bridges are devices with two or more network interfaces, that forward data frames from one interface to one or more of the others. Figure 1.1 shows a situation where two RPR rings are interconnected by a bridge.

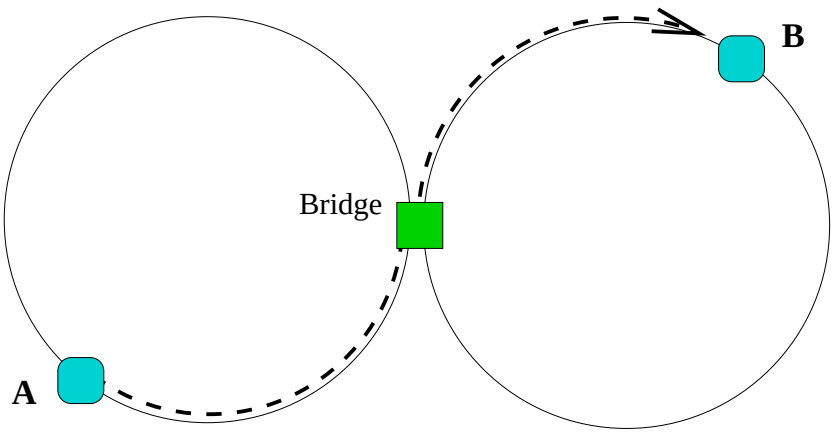


Figure 1.1: Two RPR rings interconnected by a bridge.

As will be explained in the sequel, bridging as specified in the current RPR draft standard leads to loss of the spatial reuse properties on the ring. Frames from node A to node B in figure 1.1 must be broadcasted on both rings. This broadcasting consumes extra bandwidth resources, and leads to a poor utilisation of the network resources.

Ideally, frames from node A to node B would only consume bandwidth resources along the shortest path between the two nodes, as illustrated in the figure. This work discusses an enhanced bridging algorithm for RPR networks, which allows spatial reuse also for bridged traffic. This will be achieved by keeping tables in the RPR nodes. These tables will be used to direct traffic destined for nodes on a remote network to a specified bridge.

1.2 Problem statement

The goal of this work is to evaluate an enhanced bridging algorithm for RPR. The differences between the enhanced bridging approach and the algorithm in the current RPR draft standard are discussed, and the performance of the enhanced bridging algorithm is evaluated through simulations.

The purpose of the enhanced bridging algorithm is to give better resource utilisation in a bridged RPR network by allowing spatial reuse for bridged traffic. This work tries to answer how such improvements can be made with respect to bandwidth utilisation, and at what cost this can be achieved. The tradeoffs that must be made are discussed. The enhanced bridging algorithm relies on mapping tables in the RPR nodes, demanding some amount of available memory. The effect of limiting the size of these tables is investigated. Furthermore, reducing the amount of traffic in a network means that the probability that a link gets congested declines. This in turn has a positive effect on the latency and jitter characteristics of all the traffic. This work tries to shed some light on what effect the choice of bridging algorithm has for the local traffic on an RPR ring.

1.3 Simulation model

A simulation model of RPR networks written in the Java programming language is used to evaluate the enhanced bridging algorithm. The model used in my work is an extension of a model previously developed by professor Stein Gjessing at Simula Research Laboratory. The work with this model has been a substantial part of the work in this project. The simulation model has been used to investigate the above problems, through constructing and simulating

three different traffic scenarios.

The source code for the simulation model used in this work is available at

`http://www.simula.no/download/nd/rpr/rprbridging/`

Three configurations of the simulation model are given, corresponding to the three scenarios described in chapter 6. By following the steps described in the `readme` file associated with each scenario, it should be possible to recreate the simulation results. An electronic version of this document is also provided at the web site.

1.4 The structure of this paper

This paper is organised in seven chapters. After this introduction, chapters 2 and 3 give the reader the background needed to follow the discussion of bridging in RPR. Chapter 2 discusses the emerging RPR standard, while chapter 3 treats bridging in general, with emphasis on how different bridging strategies are used in existing network technologies. Then, chapter 4 introduces the problems that arise when bridging is used in RPR networks. The bridging algorithm used in the existing RPR draft standard is discussed, together with an enhanced bridging algorithm which allows spatial reuse of bridged traffic. Chapter 5 describes the simulation model used in the performance evaluations of the bridging algorithms. The three simulated scenarios are presented in chapter 6, and the simulation results are discussed. Finally, chapter 7 summarises the work, and discusses whether the goals have been reached and what has been learnt through this work. Some ideas for further research topics are also presented.

At the end, appendix A gives an explanation of some of the terms and acronyms used in this work.

Chapter 2

Resilient Packet Ring

A basic knowledge of the RPR technology is needed to fully appreciate the discussion of bridging algorithms which is the main topic of this work. As will be explained in chapter 4, some adaptations are needed before bridging of RPR networks is possible. The reason for this can be found in some of the properties of the RPR technology, and the main purpose of this chapter is to provide the reader with the necessary understanding of RPR. The latest RPR draft standard consists of over 600 pages. This chapter is thus not a full description of all RPR functionality. However, most of the important RPR characteristics are mentioned.

This survey of the RPR technology is based on the February 2003 draft of the RPR standard [18]. This document is not an official standard, only a working document for the RPR working group. It is only available through the working group's password-protected web site.

The purpose of section 2.1 is to place RPR in a broader context. Different network topologies are described, and some existing ring network technologies are mentioned. Some of the motivating factors for developing RPR is highlighted, and IEEE 802 is discussed as the formal framework in which RPR is developed. Then, section 2.2 explains some of the key features the designers of RPR had in mind, and what makes RPR distinct from formerly developed ring technologies. Section 2.3 describes the data path in RPR, which contains functionality for transmitting, forwarding and removing frames on the ring. Finally, section 2.4 gives an overview of the MAC control sublayer, describing various control functions in RPR.

2.1 RPR in context

2.1.1 Network topologies

Computers can be connected in a wide variety of topologies. The most intuitive topology is perhaps to have a direct link between each of the nodes in the network. This topology, known as a *mesh*, has its clear advantage in providing each node with direct connection to all other nodes. The obvious drawback is scalability; the number of links and network interfaces needed is proportional to the square of nodes. A more general variant of this topology is a *partial mesh*, in which not all nodes are directly connected. This kind of general network is perhaps the most usual of all network topologies.

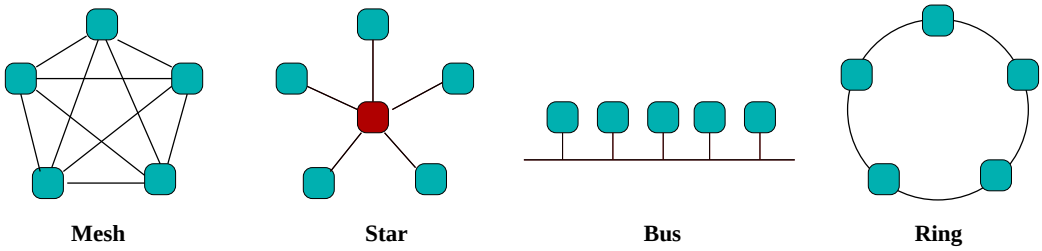


Figure 2.1: Possible network topologies

Another possible topology is *star* networks. Here, all nodes are connected to one central node. The cables radiate from this central node much like the spokes in a wheel. This topology demands less cables and network interfaces than does the mesh topology. However, the scalability can be limited by the central node, through which all traffic in the network must pass. Another issue is reliability - if the central node fails, all the other nodes become unreachable.

In a *bus* topology, all nodes are connected to the same cable in a linear fashion. Data transmitted by one node on this common cable is available to all other nodes. This topology is easy to manage, since nodes can be added and removed without affecting the other nodes in the network. In a heavy loaded network, performance may be limited by the fact that all nodes have to share the available capacity on the cable.

Finally, nodes in a network can be connected in a *ring*. In this topology, each node is linked to its two neighbours. Data is passed from node to node around the ring until it reaches its destination. A ring topology allows nodes to be connected using a modest number of links and network interfaces, without the possible bottleneck of a central node. A drawback with a ring topology, is that the loss of a single node or link can possibly disable the

entire network. This problem can be overcome in a *dual ring* network, where each node is connected to two counter rotating rings. A dual ring allows the construction of protection mechanisms that can restore connectivity in case of a node or link failure, so that if a node or link breaks, all nodes are still connected.

2.1.2 Existing ring technologies

The idea of organising computer networks as rings is far from new. For the last three decades, which is the main part of the history of computer networking, different kinds of ring network technologies have been developed. RPR has a number of interesting older relatives.

The first ring network technology was the Cambridge Ring [23], developed at the University of Cambridge in the early seventies. This single ring technology came into use in 1975, and was one of the first LANs in use. The Cambridge Ring regulated access to the medium by passing a token from node to node around the ring. The owner of the token was allowed to insert two bytes of data into it, and send it to the desired recipient. Using the Basic Block Protocol [24] for communication, the total size of the token with its two bytes of payload was 36 bits, giving an overhead of 125%.

A few years later, IBM developed Token Ring as their main LAN technology. This technology has also been standardised by the IEEE as the 802.5 standard [2]. Like the Cambridge Ring, Token Ring is a single ring technology. As the name suggests, Token Ring kept the idea of circulating a token to regulate access to the medium. Each station is allowed to hold the token for a limited period of time, and has the exclusive right to transmit on the ring during this period. The frame size in a token ring is not fixed, as it was in Cambridge ring. Instead, the maximum frame size is limited by the maximum token holding time. Token Ring originally operated over 4 Mbps copper wires, with a later extension allowing 16 Mbps links. Being IBM's main LAN technology for a long time, Token Ring has been one of the most widely deployed ring technologies.

The need for higher bandwidth led to the development of Fiber Distributed Data Interface (FDDI) [29]. This technology operates in much the same way as Token Ring. The most striking difference is with respect to bandwidth, where FDDI operates at a speed of 100 Mbps. Also, FDDI is a *dual* ring network, meaning that it consists of two separate fibre rings known as the primary and secondary ring. The secondary ring is not used during normal operation, but it allows the implementation of a protection mechanism. In case of a network failure, traffic is wrapped over on the secondary ring, thus maintaining a ring topology even after a link or station failure.

Another interesting single ring network, is Scalable Coherent Interface (SCI) [14]. This high speed network offers transfer speeds of up to 8 Gbps. The origin of SCI was an attempt to come up with a more efficient bus architecture for multiprocessor computers. The limitations of a bus architecture led the inventors to choose a ring topology [10]. SCI is a single ring technology, consisting of uni-directional links connecting up to a theoretical maximum of 64K nodes. Unlike the technologies discussed above, SCI has no token controlling access to the medium. Instead, SCI is a *buffer insertion* network, where each node makes a decision on when to send based on the traffic passing on the ring. SCI uses destination stripping, meaning that the destination node is responsible for stripping a received frame off the ring. When a receiver strips a frame, an ack or a nack message continues around the ring back to the sender, indicating whether the frame could be received. The sender node buffers the transmitted frames, and resends them if the destination node could not receive it. The strategy of destination stripping is different from Token Ring and FDDI, where the sender node strips the frames when they have traversed the whole ring. As we will see later, destination stripping is also used in RPR.

2.1.3 Why RPR?

Much of the fibre found in todays metro networking environment is organised in ring topologies. Data traffic over such fibre rings is typically sent using the SONET [1] protocol. SONET is well suited for a ring topology, with a fast (sub 50 ms) protection mechanism that can restore connectivity using an alternate path around the ring in case of a network failure. However, SONETs circuit switched nature has clear disadvantages for data traffic. The need to reserve bandwidth for each connection between nodes is a bad match for the often bursty data traffic. Also, multicast in SONET demands a separate circuit to be set up for each recipient. A separate copy is then sent to each of them, resulting in multiple copies of multicast frames travelling around the ring. Point-to-point Ethernet on the other hand, is better suited for data traffic, making more efficient use of the available bandwidth. But it does not take advantage of a ring topology to implement a protection mechanism when this is possible. Point-to-point Ethernet also lacks a way to provide a fair sharing of the ring-wide available bandwidth between the nodes. One of the motivations for developing the RPR ring network is to come up with a technology that combines the strengths of the SONET and Ethernet protocols used over ring topologies today. RPR is packet based, which allows for a more dynamic use of the available bandwidth than a circuit switched SONET ring. At the same time RPR exploits the ring topology to implement

a protection mechanism and a ring-wide fairness algorithm, as discussed in section 2.4.

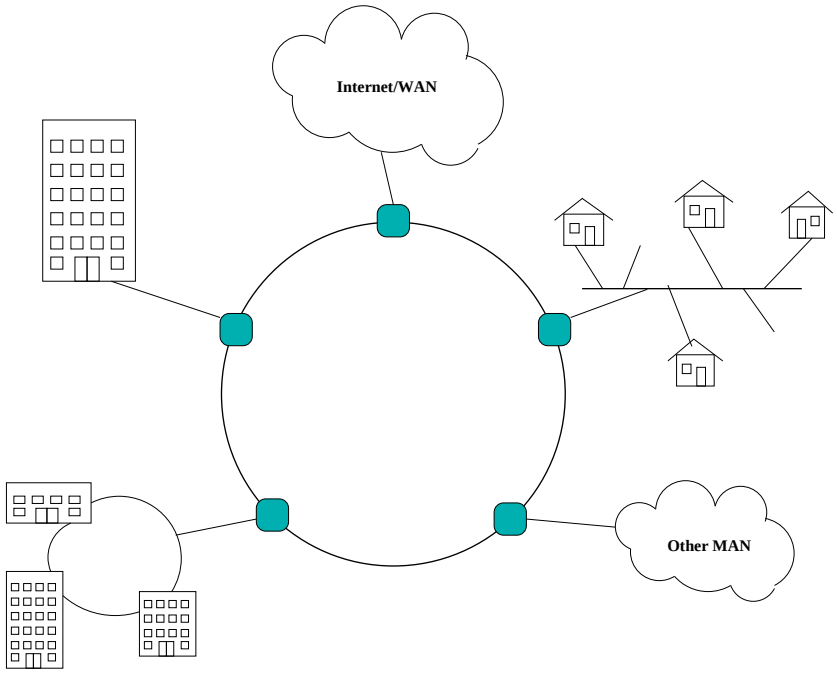


Figure 2.2: An RPR ring can typically be used in a metro area to interconnect several local networks.

2.1.4 Access scenarios

RPR is primarily designed to be a metropolitan or wide area (MAN/WAN) technology, but its use in a local area (LAN) context is also discussed. A typical scenario could involve an RPR ring spanning a campus, a city or a larger region. Each node on the ring could aggregate traffic from an office building, a campus, or a rural area, depending on the scale of the network. Figure 2.2 shows a possible scenario, where RPR is used to interconnect different local networks.

An interesting initiative in this context is called Ethernet in the First Mile (EFM)[32]. The first mile here means the infrastructure that connects a subscriber to the service provider. This infrastructure can often be a bottleneck between the business or residential subscriber's local network and the high capacity metro networks. Various kinds of technologies such as ISDN and xDSL are in use in this first mile today. The EFM task force argues that the well-known Ethernet technology would give clear performance and

manageability advantage over these. The work with Ethernet in the First mile is today taking place in the IEEE 802.3ah working group[1].

The EFM is interesting in an RPR/bridging context. Ethernet and RPR both use the same kind of MAC addresses for frame routing. One could thus imagine using bridges to interconnect an EFM network and an RPR ring, without the need for a higher layer protocol such as IP.

2.1.5 A word on IEEE 802

The RPR standard is being developed by the Institute of Electrical and Electronics Engineers (IEEE). IEEE is an organisation doing work in a wide range of engineering fields, from biomedical technology to electric power. In the field of local- and metropolitan area computer networking, their efforts are collected in the 802 series. The 802 family of specifications is traditionally often placed in the link layer (layer two) of the Open Systems Interconnection (OSI) reference model [19]. This is because the 802 protocols act like a link layer seen from a network protocol like IP. However, the 802 protocols also perform many functions that usually would belong in the network layer, like addressing and some routing functionality. It is therefore not possible to give an accurate description of the relationship between the 802 specifications and the OSI reference model. The placement of the 802 standards in the link layer given in figure 2.3 is thus not the whole truth. Still, it gives an idea of what kind of protocols the 802 specifications define, at least as seen from IP's perspective.

The RPR technology can be placed in what is known in IEEE terminology as the *Medium Access Control (MAC)* layer, and is said to define a specific MAC protocol. The MAC layer defined by RPR can be split in a data path and a control sublayer. RPR also defines a service interface to the above, logical link control sublayer, and the underlying physical layer. RPR does not itself define a physical layer. Instead, RPR is designed to run over existing layer one technologies. In the current draft standard, reconciliation layers to the Ethernet and SONET/SDH physical layers are defined.

The origin of what is today known as Resilient Packet Ring, was a technology called Spatial Reuse Protocol [33], developed by Cisco Systems around 1998. Cisco took their work to the IEEE, to see if it could form the basis for a new standard for ring networks. Here, a study group was formed, which in March 2001 was given status as a working group. The project got the name Resilient Packet Ring, and was designated the IEEE number 802.17. The

[1]The proceedings of the 802.3ah standardisation efforts can be followed at http://grouper.ieee.org/groups/802/3/efm/

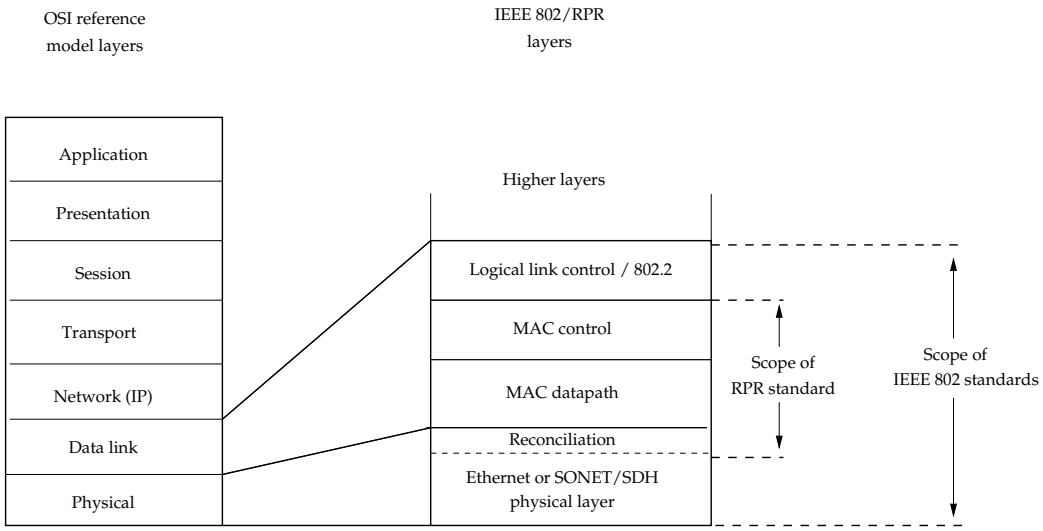


Figure 2.3: The IEEE 802 family of specifications can be placed in the lower part of the OSI reference model.

working group consists of representatives from many of the major actors in the networking industry and from several academic communities. The group has produced several draft versions of the new standard. An official version is expected in the second half of 2003.

Being an IEEE 802 standard, RPR falls into line with other well known networking standards standards, such as 802.3 [4] (Ethernet), 802.5 [2] (Token Ring) and 802.11 [3] (Wireless LAN). The 802 family also contains a specification of bridges as a means for interconnecting 802 networks of various kinds. These standards, known as 802.1D [16] and 802.1Q [15], must be supported by all 802 networks. A closer look upon this kind of bridging is given in section 3.2.

2.2 Major design issues

One of the important characteristics of RPR is its *dual ring* topology. An RPR network consists of two counter rotating rings, giving each node a full duplex connection to each of its two neighbors. This double ring can be seen as two independent *ringlets*. The ringlets are called counter rotating, because the traffic on each flows in opposite directions. Traffic on the *outer* ringlet is defined to flow clockwise, while the traffic on the *inner* ringlet flows counter-clockwise, as shown in figure 2.4. At each ringlet, a node receives frames from its *upstream* neighbour, while it transmits to its *downstream*

neighbour.

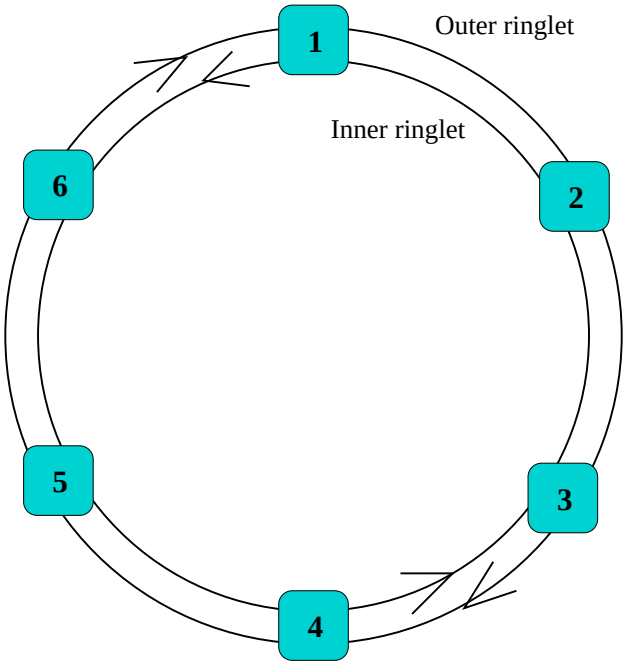


Figure 2.4: The traffic flows in opposite directions on the inner and the outer ringlet

The concept of a dual ring topology has several advantages. Like in FDDI mentioned above, it is used to implement a protection mechanism that restores connectivity in case of a node or link failure. The key difference here is that while the secondary ring in FDDI is idle during normal operation, both RPR ringlets take part in the packet transport. RPR chooses which of the two available ringlets to transmit on. This makes possible a better link utilisation, allowing RPR to transport twice as much traffic as FDDI over the same amount of fibre. The protection mechanisms in RPR are discussed in section 2.4.

Unlike Token Ring and FDDI, RPR has no token or other synchronisation mechanism to control access to the medium. Instead, RPR is a *buffer insertion* ring, like SCI. Each node makes its own decisions on when to transmit, in correspondence with the fairness algorithm described in section 2.4. To avoid collision when the node is pulsing out a frame from its own transmit buffer, a buffer is needed to store traffic arriving from the upstream neighbour, as shown in figure 2.5. This buffer is called a *transit buffer* in RPR, or sometimes also an *insertion buffer*. If the transit buffer is filled above a configurable threshold, the node is prevented from adding traffic to the ring by the fairness algorithm.

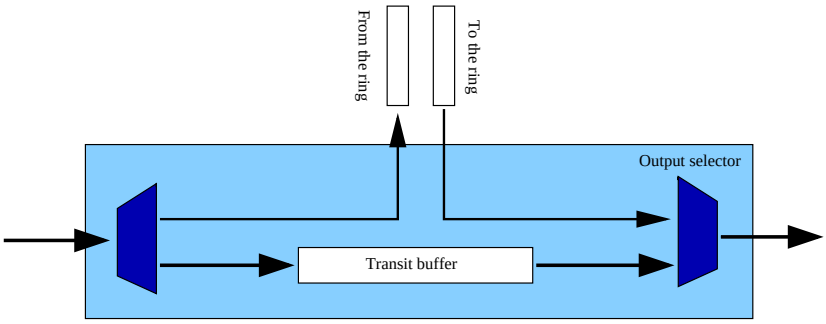


Figure 2.5: Buffer insertion rings have a transit buffer to hold back frames in transit while transmitting frames on the ring.

Another important characteristic of RPR is *spatial reuse* of the available bandwidth. This allows several frames to be in transit on different parts of the same ringlet at the same time, as illustrated in figure 2.6. Spatial reuse is made possible by the use of *destination stripping*. This means that a frame is stripped from the ring when it reaches its destination node, as in SCI mentioned above. The bandwidth that would otherwise be consumed by the frame traversing the whole ring back to the source, is this way made available for other traffic. The reuse of this freed bandwidth, is termed spatial reuse. Destination stripping and spatial reuse allows each unicast frame to travel only the shortest path between the sender and destination nodes. As we shall see later, destination stripping makes it more complicated to do bridging in an RPR network. Maintaining the spatial reuse properties on the ring becomes one of the key challenges in the context of bridging.

The dual ring and spatial reuse properties of RPR can give significant performance advantages. By choosing the ideal ringlet for transmission, no frame has to travel more than half way around the ring. On average, each frame needs only to traverse one quarter of the ring, with uniformly distributed traffic. Compared to Token Ring and FDDI, where a frame always travels the full length of the ring, this can significantly improve bandwidth utilisation.

2.2.1 RPR frame format

The RPR frame format has had many different forms, before it reached the design here described. This clause treats the RPR frame format as defined in draft 2.1 of the RPR standard [18].

Four different types of frames are transmitted on an RPR ring; idle frames, control frames, fairness frames and data frames. Idle frames are used for synchronisation purposes when no other traffic is sent on a link.

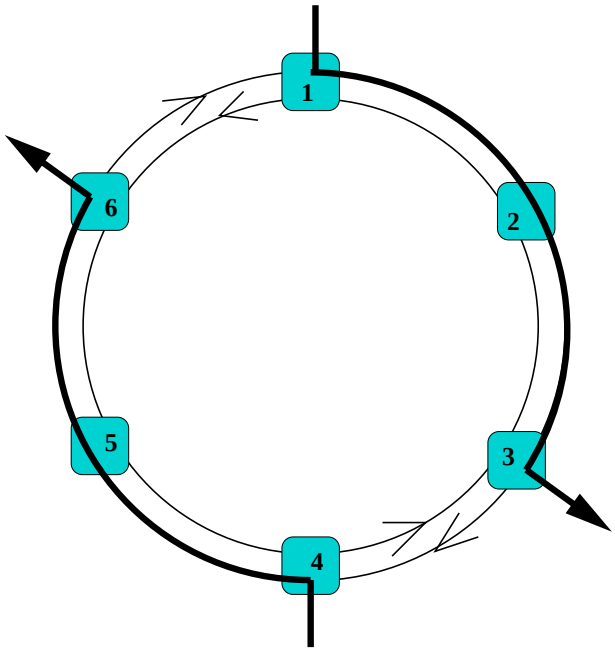


Figure 2.6: Spatial reuse allows several packets to be sent on different spans of the same ringlet simultaneously. Node 1 can transmit to node 3 at the same time as node 4 transmits to node 6.

Control frames are used to propagate topology, maintenance and error recovery information. Fairness frames are small fixed-size frames used by the fairness algorithm described below in its communication. This discussion will focus on the data frames used in the normal exchange of traffic between the nodes on the ring.

The minimum RPR data frame size is 24 bytes, which is the combined size of the header and trailer fields, along with a minimum two byte payload. These two bytes contain the protocolType field described below. The maximum frame size is 1522 bytes, allowing a 1500 byte payload for Ethernet compatibility. An RPR ring can optionally support jumbo frames, which can be up to 9216 bytes long. Figure 2.7 shows the layout of an RPR data frame, with its different fields.

- The time-to-live (ttl) field is used to limit the number of hops a frame can travel before reaching its destination. It can be set to the expected number of hops for the frame, or a higher number. The use of this field in a bridging algorithm is discussed in section 4.3. The ttl is an eight bit value, limiting the maximum number of nodes on an RPR ring to 256. [2]

[2]Due to other considerations, the maximum number of nodes allowed on an RPR ring

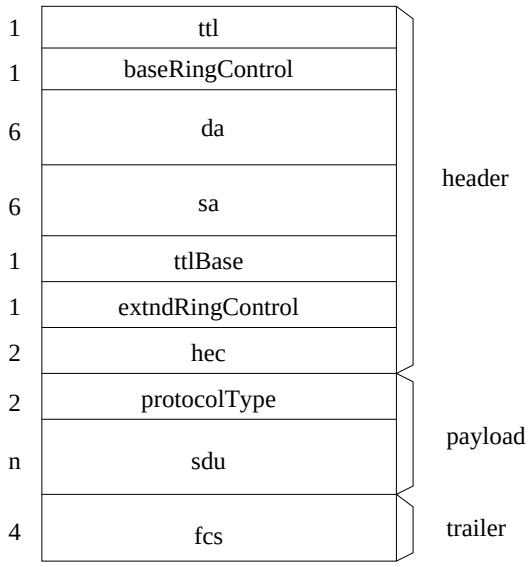


Figure 2.7: RPR data frame format

- The baseRingControl field in the header contains several fields with information that is used to control the frame's flow around the ring. This includes information such as what type of frame it is, what service class it belongs to, and various information needed in case of a protection event.

- The destination and sender addresses are kept in the da and sa fields respectively. These are globally unique 48 bit MAC addresses, used to identify the nodes on the ring.

- By setting the value of the ttlBase field to the ttl value at the time of transmission, the number of hops a frame has travelled can easily be calculated at the destination. This is useful in the context of protection, to prevent packet reordering and duplication. As will be shown in section 4.3, this information can also be helpful when implementing an enhanced bridging algorithm.

- The extndRingControl field contains more option subfields to control the frame behaviour. This includes information that is needed if the source or destination node of the frame is not on the local ring.

- The hec field contains a checksum on the header.

is 255, not 256

- The two bytes protocolType field in the payload of the frame are interpreted either as a specifier of the MAC client protocol, or as the length of the payload. The maximum RPR payload (MTU) is 1500 bytes, allowing a protocolType value greater than this to specify the nature of the MAC client protocol.
- The sdu can be of variable length, and contains the service data unit provided by the MAC client.
- Finally, the 32 bit frame check sequence in the trailer is used to discover bit errors in the payload of the frame.

An extended frame format is used for frames whose source or destination address is not local on the ring. These frames must contain an extra local source station identifier to ensure reliable delivery with respect to packet reordering and duplication. This extra header information becomes very important in the context of bridging. A full treatment of this topic is postponed to section 4.3, where an improved bridging algorithm is discussed.

2.3 The MAC data path

The medium access control layer defined by RPR, can be naturally split into a MAC data path and a MAC control sublayer, as shown in figure 2.3 on page 11. The MAC data path sublayer is responsible for transmitting frames on the ring, forwarding traffic from the upstream to the downstream node, and stripping frames from the ring.

Figure 2.8 shows a simplified model of the MAC data path. An RPR node contains a separate MAC data path for each of the ringlets. Each data path contains logic to strip frames from the ring, store the frames in transit in a buffer if needed, and select the next frame to be transmitted. The transit buffer is needed to store frames that cannot be passed on to the downstream neighbour immediately, when the node is already transmitting on the outbound link. The RPR standard does not specify the size of this buffer, but it needs to be at least as big as the RPR maximum transfer unit. Two different options are allowed with respect to the transit buffer. A single-queue design (as on figure 2.8) has only one transit buffer, while a dual-queue (figure 2.9) has two. The use of these buffers is explained below.

2.3.1 Frame transmission

RPR supports three different service classes for frames transmitted on the ring; high, medium and low priority. A limited fraction of the frames are

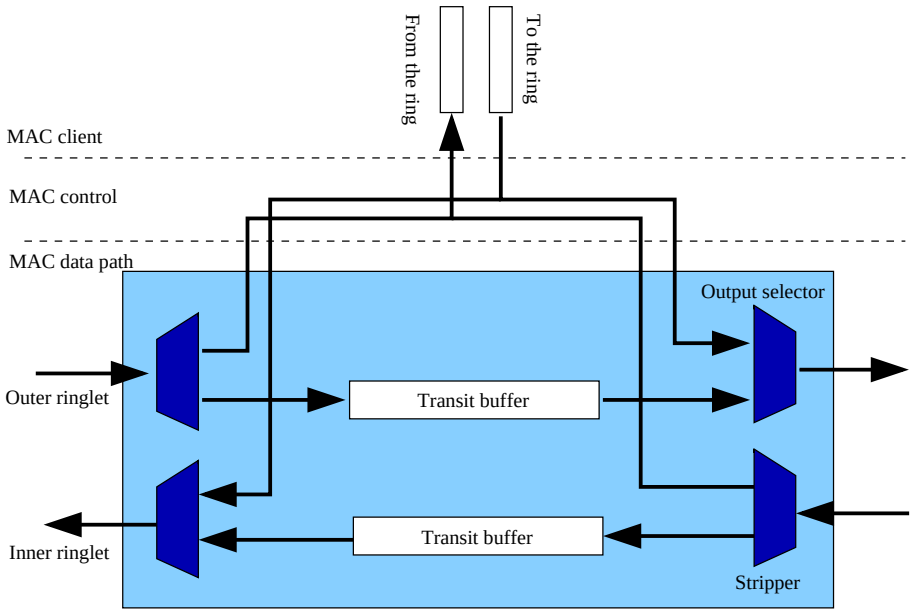


Figure 2.8: There are two separate data paths in the MAC data path layer, one for each ringlet.

high priority, known as class A traffic. This service class is subdivided into class A0 and class A1, the difference being that A0 uses reserved bandwidth that cannot be re-used by the other traffic classes. The class A service is used by applications with strict demands on latency and jitter, such as interactive audio or video applications. In addition, RPR control frames are usually transmitted using this traffic class. The class B service offers a bounded delay on a part of the traffic, which does not exceed that nodes *fair rate*. This traffic is known as *in-profile* traffic, and is not subject to rate limiting by the fairness algorithm. Class B traffic that exceeds the fair rate is deemed *out-of-profile*. Together with the low priority class C service, this traffic has no guarantees on delay, and is rate limited by the fairness algorithm.

The MAC client has a separate transmission buffer for each traffic class. Frames are taken from this buffer and put into a stage buffer, and then onto the ring, as shown in the simplified model in figure 2.9. Associated with each traffic class is a *shaper*, which controls how much traffic is added on the ring. The shapers provide each traffic class with *credits*, that are accumulated over time in a token bucket fashion. The MAC client must pay one token from this token bucket for each byte of traffic it adds to the ring.

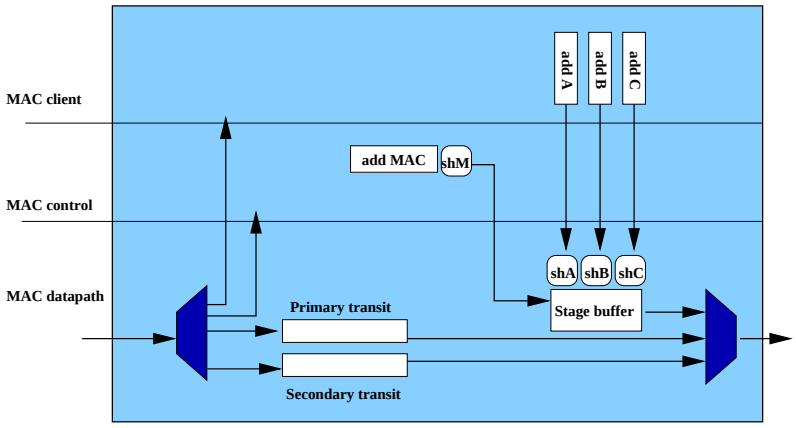


Figure 2.9: Each traffic class has its own transmit buffer, from which frames are put onto the ring. In addition, MAC control frames have a separate transmit buffer. shA, shB and shC are the shapers associated with traffic classes A, B and C respectively. shM is the shaper for the MAC control frames.

2.3.2 Frame reception

A node on an RPR ring must inspect every frame received from its upstream neighbour, and make a decision on how to treat it. The node first inspects the destination address to see if the frame is destined for this node. "Frames destined for this node" includes unicast frames for this node, broadcast frames, or multicast frames meant for this node. If any of these is true, the frame is copied to the MAC control sublayer. From here, data frames are passed on to the MAC client, based on the frame type field discussed in section 2.2.

The node then decrements the ttl of the frame, and frames with a ttl of zero are discarded. Frames with a destination address different from this nodes MAC address are put in a transit buffer, waiting to be passed on to the downstream node. The transit buffer acts like a fifo queue, and is sometimes referred to as the transit queue. As mentioned above, RPR allows one or two such buffers. In a dual queue design, the primary buffer is reserved for class A traffic, while both class B and C use the secondary transit buffer. Since the primary buffer has higher priority when frames are selected for transmission, it is possible for class A traffic to overtake other traffic in the nodes. Note that class B traffic can never overtake class C traffic on the ring, even with a dual queue implementation. The difference between these classes lies exclusively in how they are treated in the ingress node. The RPR standard puts no constraints on whether the transit buffers should be "cut through", where transmission can start before the whole frame has entered

the buffer, or "store and forward".

2.3.3 Stripping frames from the ring

As described in section 2.2, spatial reuse is made possible in RPR by making the destination node responsible for stripping a frame from the ring. Multicast and broadcast frames are exceptions from this rule. These are not stripped by their destination, but continue around the ring and are not removed until they return to their source node. Obviously, no spatial reuse for multicast or broadcast traffic is possible with this strategy. Source stripping is also used as a safety mechanism for unicast traffic; unicast frames that for some reason were not stripped by the destination, are removed when they return to the source. Specifically, frames with a unicast destination address that is not found on the ring will be source stripped.

If the source node disappears from the ring while a multicast frame is in transit (or *both* the source and destination node for a unicast frame), no node would ever remove the frame. It would seemingly keep circulating on the ring forever. This is prevented by the ttl field in the RPR header. The ttl works as a maximum hop count, and is decremented every time the frame reaches a new node. If the ttl reaches zero, the frame is discarded.

To sum up, there are four different conditions that leads to the stripping of a frame from the ring.

1. If a bit error is detected in the received frame, the frame is discarded. For data frames, only bit errors in the header results in stripping.
2. A unicast frame reaches its destination node.
3. A frame returns to its source node, after travelling around the ring.
4. The ttl value of the frame reaches zero.

Frames that do not meet any of these conditions, are passed on to the next downstream node.

2.4 The MAC control sublayer

While the MAC data path sublayer is responsible for the actual sending of frames from one node, the MAC control sublayer is concerned with ring-wide functionality that demands cooperation between several nodes. Here lies the logic that monitors the traffic on the ring, and takes appropriate action when an exception occurs. Figure 2.10 shows an overview of the MAC

control sublayer and the functions it perform. This section will discuss each of these functions.

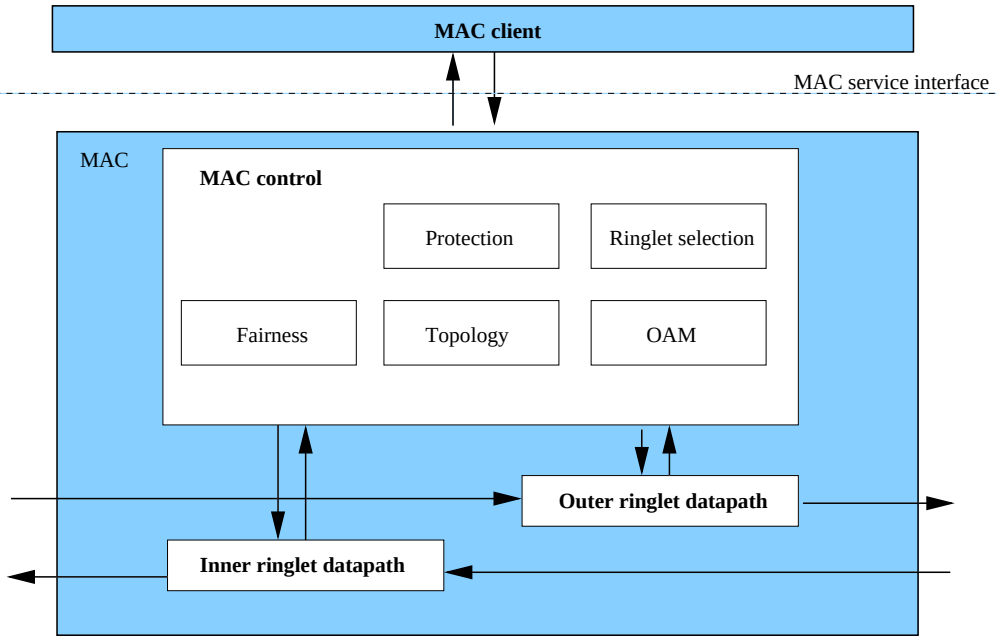


Figure 2.10: This figure shows an overview of the medium access control layer defined by RPR, with the placement of some of the important functionality.

2.4.1 Protection

RPR will be used to carry different kinds of traffic, including data with strict demands on latency, such as real time interactive audio. A goal is therefore to be able to detect and repair a node crash or a link failure within 50 ms, which is considered the time constraint for audio. A connectivity failure, either in a node or on a link, is termed a *protection event*. RPR offers two separate protection mechanisms that can be used to restore connectivity when such an event occurs, *steering* and *wrapping*. Both of them are made possible by the dual ring nature of RPR.

Steering is the normal protection mechanism, and must be supported by all nodes. The basic idea is to make the sender nodes responsible for avoiding points of error when frames are transmitted. The node(s) that first detects a protection event will send an error report around the ring. This is used by the other nodes to update their topology image, using the topology discovery mechanism described below. A new host-to-ringlet mapping is built in all the nodes. When this new image is complete, the point of failure can be avoided

by choosing the correct ringlet to send on. If steering is the only protection mechanism in use on the ring, all frames that are sent in the time from the failure occurs to the topology image is rebuilt, are lost.

The advantage of the other protection mechanism in RPR, wrapping, is that it reduces the number of lost frames during a protection event. An example of wrapping is shown in figure 2.11. In case of a node failure, frames destined for a node beyond the point of error are looped and placed on the opposite ringlet. This way, frames that are in transit when an error is detected can still reach their destination on the other ringlet. The support of this protection mechanism is optional in the RPR standard.

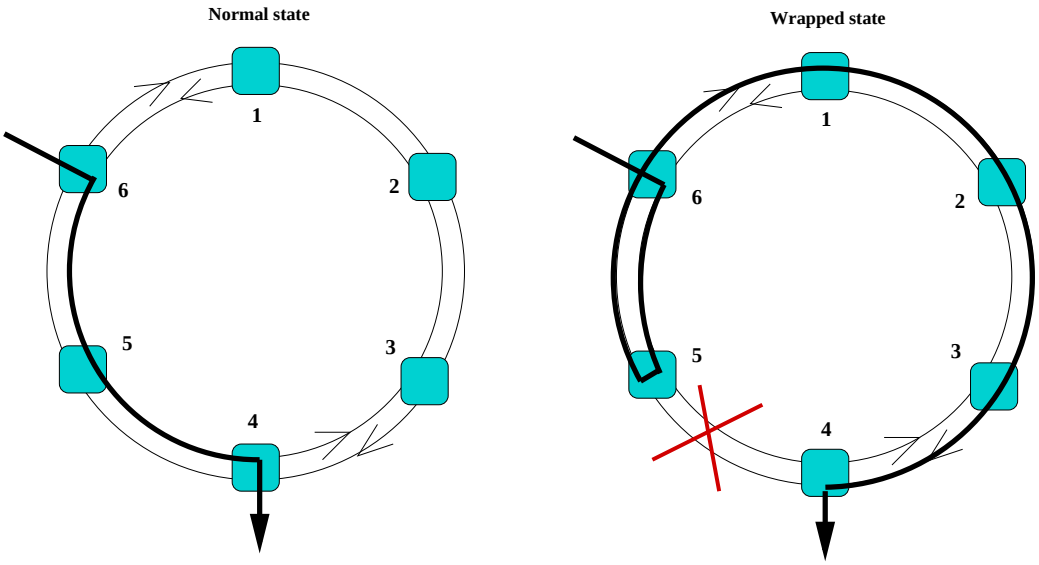


Figure 2.11: Frames reaching a point of error are wrapped and sent back on the opposite ringlet. During normal operation, node 6 sends to node 4 via the path 6-5-4. In a wrapped state, the frames are sent 6-5-6-1-2-3-4.

Wrapping increases the complexity of the ring, by introducing some new problems regarding frame duplication and reordering. In figure 2.11, consider a case when the ring returns from wrapped to normal state, or when steering causes frames from node 6 to node 4 to be transmitted on the outer ringlet. This transition suddenly reduces the number of hops the frames must travel, leading to the possibility that frames reach their destination in a different order than they were sent. Such frame reordering is not acceptable to some applications. RPR therefore allows traffic to be sent in two different modes, known as *strict mode* and *relaxed mode*. Frames in strict mode are never wrapped. This guarantees that no frame reordering will occur, but increases the number of frames lost during a protection event. Strict mode frames will

be discarded at the point of error, and all such frames will be lost until the steering mechanism restores the connectivity.

If a link fails or a node suddenly crashes, loss of some frames is inevitable without buffering. Frames in transit in a node that dies, for instance because of a power failure, will certainly be lost. The above protection mechanisms are used to restore connectivity in case of such an event. The use of wrapping minimises the number of frames lost. Note that during normal operation, RPR guarantees that no frames are lost or reordered on the ring. Only in case of a protection event might this occur.

2.4.2 Fairness

The nodes on an RPR ring must compete for the available bandwidth. If the nodes' aggregate need for bandwidth exceed the capacity of the ring, a mechanism to provide each node with its fair share is needed. The algorithm used to achieve this is called RPR fairness.

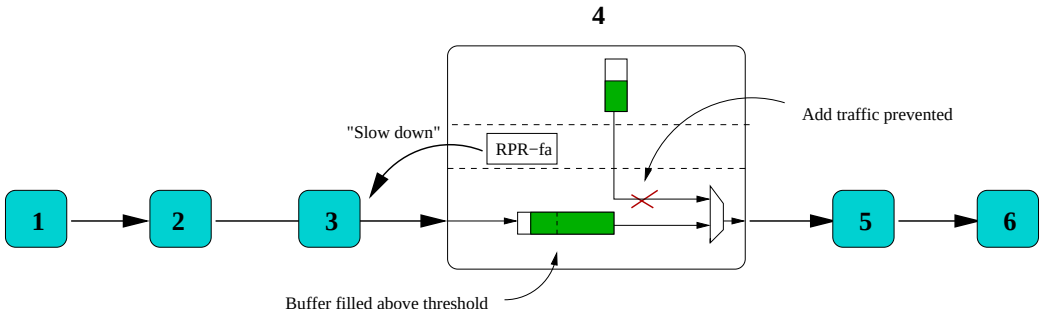


Figure 2.12: Node four is prevented from sending, because the transit buffer is filled up with frames arriving from node 3. It sends a fairness frame to its upstream node to slow it down.

A node is prevented from adding traffic on a ringlet when its transit buffer is filled above a configured threshold. If a node continuously receives frames from an aggressive upstream neighbour, this can lead to *starvation*. Figure 2.12 shows a situation where node four is prevented from transmitting on a ringlet, because node two is sending a continuous flow of frames to node five. If node four had kept adding traffic in this situation, the result would be buffer overflow and loss of frames in the transit buffer. This would not be acceptable, since RPR is designed to be a lossless network technology, meaning that no frames are lost on a single ring during normal operation. To end the starvation in node four, the fairness functionality generates small fairness frames with information on how much traffic this node (node four) is adding to the ring. This information, called the node's *allowedRate*, is

sent to its upstream neighbour. Fairness frames concerning the data flow on one ringlet, are sent using the opposite ringlet. When the upstream node receives the fairness message, it adjusts its send rate so that it transmits no more than the received allowedRate. The idea of returning fairness messages to the upstream node is called *backpressure*, and is shown in figure 2.12.

Each node advertises its allowedRate periodically. The fairness frames are forwarded around the ring and processed by all upstream nodes, so each of them know where the ring is most congested. The upstream nodes then adjust their sending rates accordingly. This is called "single-choke", since each node keeps track of the single most congested link on the ring. A more advanced version of the fairness algorithm allows every node to monitor the congestion at each of the downstream links. By using this technique, known as "multi-choke", a node knows which links are congested, and which are not. This information can be used to send traffic to nodes beyond a congested link at a slower rate than to other nodes, as shown in figure 2.13. To avoid so-called head-of-line blocking, the nodes must implement Virtual Output Queueing (VOQ). Head-of-line blocking occurs when the first frame in the output buffer is held back because it is destined for a node beyond a congested link, thus blocking the way for the frames behind, destined for nodes before the congested link. By maintaining a separate output queue for each of the destination nodes, other frames can be transmitted before such a blocking frame. By allowing traffic destined for nodes near the sender to ignore far-away points of congestion, the use of multi-choke and VOQ can improve throughput on the ring [9].

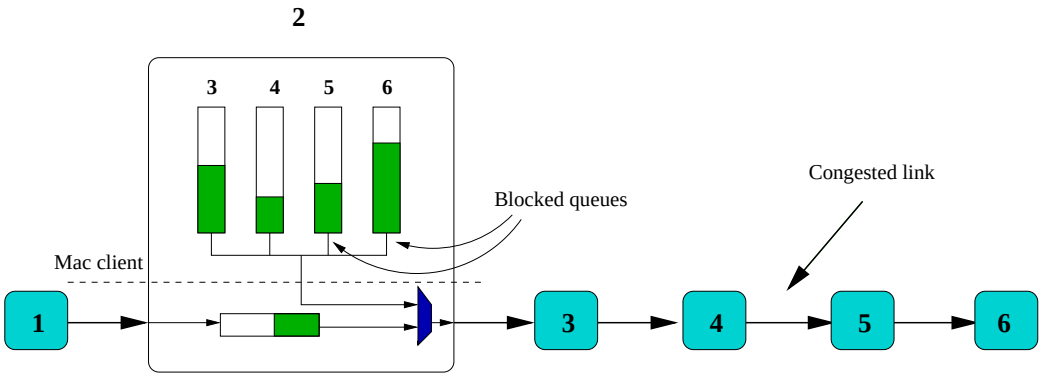


Figure 2.13: Node two keeps a separate output queue for each of the other nodes. When the link between nodes 4 and 5 gets congested, the frames destined beyond this link are held back.

2.4.3 Ringlet selection

Before a frame can be transmitted on the ring, it must be decided which ringlet to transmit on. When this decision has been made, the frame can be put in the appropriate add queue. RPR allows this decision to be taken by the MAC client. If the MAC client has no preferences, a default ringlet will be chosen by the MAC control sublayer. The algorithm used to choose the default ringlet is implementation specific. The only constraint imposed by the standard is that the same ringlet must be chosen for all frames with the same [source address, destination address, traffic class] tuple.

2.4.4 Topology discovery

RPR contains a topology discovery mechanism in the MAC control sublayer. This mechanism is responsible for providing each node with an overview of the number and placement of nodes on the ring. Every node keeps and maintains such an overview, called the *topology image*. The topology image consists of one record for each node on the ring. These records contain a hop count to that node on each of the ringlets. The records also contain additional information concerning each node, like protection status and reserved bandwidth status.

The topology information is distributed to the nodes by broadcasting protection messages on the ring. This distribution is not governed by a master node, each node is responsible for initiating the protocol as needed. Initially, a node's topology image contains only one record, with information about the node itself. It then triggers the topology algorithm by broadcasting a protection message, and continually listens for received protection messages. Each node that receive this protection message, responds by broadcasting a protection message on all ringlets. By looking at the received protection messages, each node gets information about the distance to every other node, and information about its protection status. This information is used to build the topology image.

The calculation of the topology image is repeated in case of a protection event, that is when a node discovers a failure in the connection to one of its neighbours. Note particularly that such a protection event will occur when a node is added to or removed from the ring. The topology calculation is also repeated periodically. The time between each periodic topology image update is configurable, the default period being 10 ms.

2.4.5 Operations, Administration and Maintenance (OAM)

The OAM functionality in the MAC control sublayer offers a set of control functions to support configuration management, fault management and performance management. This includes functions for monitoring the connections between stations, and for reporting abnormal behaviour to a managing system. This information is distributed using special OAM control frames. The RPR draft standard allows some of these frames to be vendor specific, meaning that each individual vendor can define their own OAM frames and functionality.

2.5 Summary

This section has given a description of the RPR technology. RPR is a new buffer insertion ring network technology under development by the IEEE. It is a dual ring, with traffic flowing in opposite directions on the outer and the inner ringlet. Among RPR's most important properties are destination stripping, spatial reuse, fairness and protection.

Chapter 3

Bridging

While the previous chapter introduced the RPR technology, this chapter turns to the topic of bridging. Bridging is a well established technology in some traditional networks. The purpose of this section is to explain the idea behind bridging as a way of interconnecting networks. A basic understanding of how a bridge works is necessary in the discussion of different bridging solutions for RPR in chapter 4.

First, section 3.1 takes a look at the basic idea of bridging, and what distinguishes bridging from the related terms of routing and switching. Then, sections 3.2 and 3.3 gives a description of two different approaches to bridging; transparent bridging and source route bridging. Transparent bridging is offered most attention, since it is the most widespread and the most important technology in the context of RPR. A comparison of the two strategies is given in section 3.4, and some of the limitations of bridging as a way of interconnecting networks are discussed.

3.1 Bridging in context

To enable hosts to communicate with hosts on a different LAN or MAN, a way to interconnect such networks is needed. *Routers*, *switches* and *bridges* are all devices that perform forwarding of traffic between networks or segments of networks.

The terms router and routing is normally used in connection with IP. A router is a device with several inputs and several outputs. A router receives an IP packet on one of the input ports, and forwards it on the correct output, based on the destination address found in the IP header. The correct output port is selected based on a table with mappings between IP addresses and output ports. This table can be set manually by a system manager, or

maintained by a routing protocol like OSPF [22]. The routing process often involves modifying the routed packets, by decreasing the time-to-live field in the IP header.

A switch performs much the same tasks as a router, but this term is normally used in a different context. Like a router, a switch can be defined as a multiple-input multiple-output device that forwards frames from an input to one or more outputs [27]. But in contrast to routers, switches usually do not operate on IP packets. The term switching is normally used to describe forwarding of frames *below* the IP layer. In an IEEE 802 context, switching is used to forward MAC frames between the inputs and the outputs. Switching is also used to denote the forwarding of cells in an ATM network.

The above definition of a switch also suits a bridge, so a bridge can be looked upon as a special kind of switch. A bridge is sometimes referred to as a LAN switch. Bridges are most common in an 802 context, but they are also used with other network technologies, such as IEEE 1596 Firewire [17]. Traditionally, a bridge interconnects *shared medium* networks such as Ethernets, where all transmitted frames can be seen by all the nodes. This is, however, not always the case. RPR is an example of a network that supports bridging without being a shared medium network[1]. Bridges are also distinct from other switches in that they sometimes can interconnect networks using different MAC protocols. For example, a bridge can be used to interconnect an 802.3 (Ethernet) network to an 802.17 (RPR) network. This is possible with transparent bridges described in section 3.2, but it is not true for source route bridging described in section 3.3.

Bridges are said to operate *below the MAC service boundary*. This means that the precence of a bridge is invisible to a higher layer protocol. A network that consists of several LAN segments interconnected by bridges, is commonly referred to as an *extended* LAN. The only difference that a higher layer protocol operating over an extended LAN encounter, is a possible degradation of service quality. Note here that a bridge may be forced to discard data frames. If data from several inputs are destined for the same output, this can lead to buffer overflow in the bridge. In the context of RPR, this means that a bridged network cannot give the same guarantees with respect to packet loss as a single RPR ring.

Forwarding frames below the MAC service boundary can give an efficiency gain compared to using IP routing. In a router, every MAC frame must be unpacked and the IP address inspected, before the packet is re-packed into

[1]In RPR, the ring itself is often called a shared medium. I would argue that this is at least inaccurate. It is rather so that the ring can be made to imitate shared medium behaviour, if this is required by higher layer protocols. This is what is done in the basic bridging algorithm described in section 4.2.

a new MAC frame for transmission. Bridges base their forwarding decisions on MAC addresses, and thus avoids this unpacking and re-packing of frames. This way, bridges can be kept simpler and cheaper.

As mentioned above, bridging is always invisible to a higher layer protocol. However, the way bridging is implemented may vary. The following two sections will describe two different bridging strategies. Transparent Bridging hides from the end nodes the fact that they belong to an extended LAN. Source Route Bridging on the other hand, distributes most of the bridging logic in the end nodes.

3.2 Transparent bridging

Transparent bridging allows nodes in a network to communicate as if attached to a single LAN, while in fact they are attached to different LANs - possibly with different MAC methods. However, it is required that LANs interconnected by bridges use the same addressing scheme, e.g. 48 bit IEEE addresses (MAC addresses).

The concept of transparent bridging was first developed by Digital Equipment Corporation in the early eighties [26], primarily to interconnect several Ethernet segments. It was later submitted to IEEE, and developed into the IEEE 802.1D standard. This standard was first published in 1990, and later revised in 1993 and 1998 with descriptions of support for different traffic priorities in bridges. Several additions to this standard has been published, among the more important is IEEE 802.1Q, which defines bridge support for Virtual LANs (VLANs).

The presence of bridges in a network is always invisible to a higher layer protocol working over the network. *Transparent* bridging, in its traditional fields of application, is special in the sense that the bridges are also invisible to the end nodes in the network. The end nodes communicate as if they were all part of the same LAN segment. This is a key difference between transparent and source route bridging, described in section 3.3.

3.2.1 Conceptual model of a transparent bridge

A bridge consists of an interface to each of the network segments it connects, called a *port*, and a relay unit. The relay forwards frames between the ports, and implements the learning and filtering processes described below. Bridges implementing these learning- and filtering processes are known as *learning bridges*. Earlier bridges did not always have this functionality. Instead, all

frames reaching the bridge was forwarded on all ports except the incoming port.

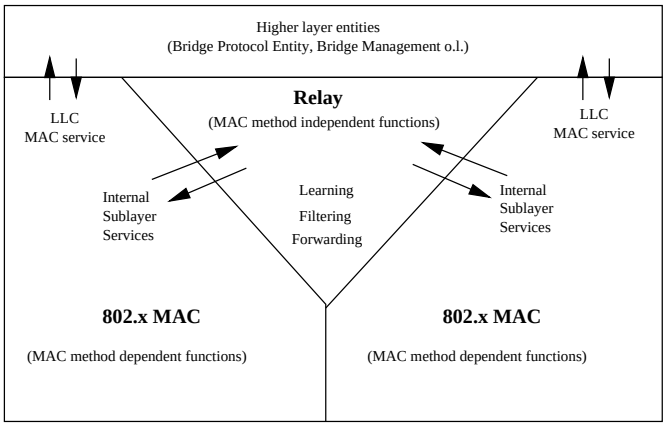


Figure 3.1: The bridge consists of a relay entity, with an interface towards each of the connected LANs

As mentioned above, a bridge can interconnect 802 LANs, each having their own way to control access to the media. The different MAC protocols communicate with the bridge using an interface known as Internal Sublayer Services (ISS). When an error-free frame from a LAN reaches a bridge, it will under certain conditions be sent through this interface to the relay part of the bridge. The relay will then decide whether the frame should be forwarded, and forward it on the correct port using the ISS interface again.

The learning-, filtering- and forwarding processes take part in the relay part of the bridge. These processes are independent of the different MAC protocols used in the attached LANs. The relay contains a table with information on which nodes are attached to the different ports, known as the *filtering database*. The filtering database can consist of a mix of static and dynamic entries. Static entries are set by an administrator, while the dynamic entries are governed by the learning process. By inspecting the source addresses of the frames passing the relay, the learning process builds up an image of which stations are reachable through the different ports. This information is used to make the bridge more efficient, by only passing unicast frames to the LAN where the destination node is known to be.

3.2.2 Transparent bridging example

In figure 3.2, LAN 1, LAN 2 and LAN 3 are connected by a bridge. The bridge maintains a filtering database, which initially is empty. Nodes A and

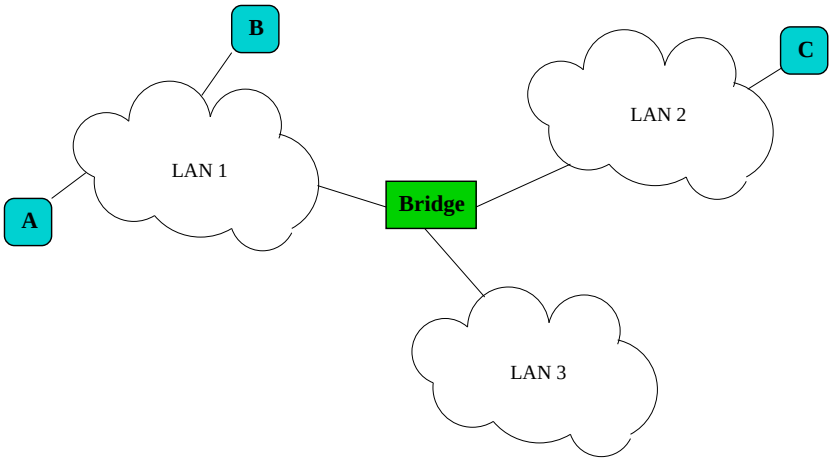


Figure 3.2: The bridge inspects the source address of each frame to learn which station is attached to which port.

B are attached to LAN 1, while node C resides at LAN 2. Consider a case where host A first wants to transmit a frame to host C. When the frame reaches the bridge, it consults it (initially empty) forwarding table. It finds no record for host C, and therefore forwards the frame on all ports, except the one it arrived on. The bridge inspects the source address of the frame, and saves a record that node A belongs to LAN 1. A response from C to A will not be forwarded to LAN 3, since the bridge now knows that node A resides at LAN 1. The bridge inspects the source address field of this response frame, and learns that node C resides on LAN 2. Note that since node B still has not transmitted a frame, any traffic destined for this node will be forwarded on all LANs. This situation will not end before the bridge has seen a frame from node B, and added an entry for node B in its filtering database.

The important thing to note from this example is that before the filtering database is built complete, some "unnecessary" forwarding will occur. This means that the bridge will forward some frames to LANs other than where the destination nodes reside. But when the bridge has built a complete image of the network topology, eg. has learnt which LAN each of the nodes belong to, frames will only be forwarded on the correct ports. The entries in the filtering database are equipped with a time stamp. This way, the bridge can discover if a node is removed from the network, or moved from one LAN segment to another. When the entry for a node gets too old, it is removed. A new entry for the node is created once a frame from that node reaches the bridge.

3.2.3 The Spanning Tree Protocol

In an extended LAN, there will usually be several bridges interconnecting the different segments. This leads to the possibility that there are loops in the network, meaning that there is more than one path from one node in the network to another, as shown in figure 3.3. To avoid having several copies of a frame reaching a node using different paths, the bridges exchange topology information using the *Spanning Tree Protocol*. This protocol was originally developed by Digital Equipment Corporation [26], and later adopted by the 802.1D standard. The purpose of the spanning tree protocol is to extract a loop-free subset of the network topology, while still maintaining full connectivity between all nodes. This is achieved through putting some of the bridge ports in a *blocking state* if necessary. A port in a blocking state does not receive or forward any frames.

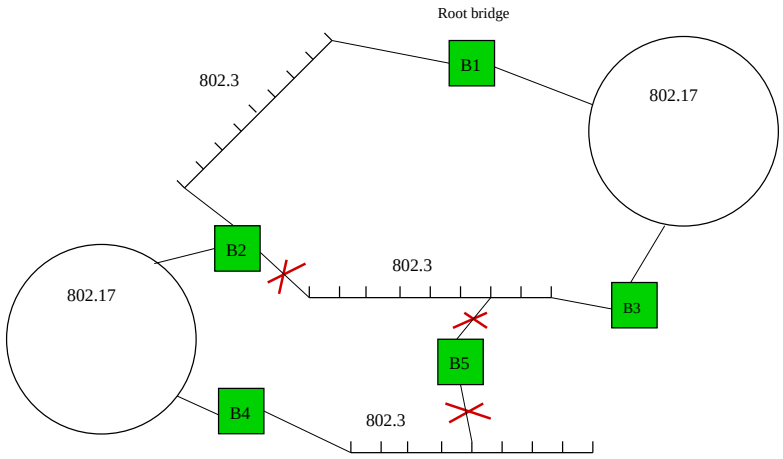


Figure 3.3: The spanning tree protocol makes the network cycle-free, by blocking one of B2s ports and both of B5s ports

The spanning tree protocol starts by selecting a *Root Bridge* in the network, based on bridge identifiers. Each bridge must have a unique identifier, typically based on the MAC address of one of its ports. The bridge with the lowest bridge identifier is selected as the Root Bridge. For each LAN segment, one of the connected bridges is then selected as a *Designated Bridge*. The designated bridge is responsible for forwarding frames from the direction of the Root to that LAN, and from that LAN in the direction of the Root. When every LAN is connected to the root through a designated bridge, all the bridge ports that are not in use are redundant. These ports are put in a blocking state, and do not participate in the forwarding of traffic. Such

ports are named *Alternate Ports*, and may be activated in case of a topology change. A topology change in this context occurs if a bridge or one of its ports goes down, or a new bridge is added to the network. Every time a topology change is detected, the spanning tree calculation must be repeated.

Figure 3.3 shows a bridged network consisting of different kinds of IEEE 802 networks. To make the network loop-free, the spanning tree protocol has put some of the bridge ports in a blocking state. Bridge B1 has taken on the role as root, and the bridges with the least path-cost to the root have been appointed designated bridges for the various LAN segments.

For some time now, almost all bridging software has used a newer version of the spanning tree protocol called the Rapid Spanning Tree Protocol (RSTP), which allows faster reconfiguration of the network in the event of a topology change [31]. This algorithm is now optional, but will probably be the only valid choice in the next revision of the IEEE 802.1D standard, expected some time in 2003 [13].

3.3 Source Route Bridging

As seen above, in transparent bridging, all the logic needed to bridge several LAN segments is kept in the bridges. The attached stations do not have to worry about bridging, they behave the same as when connected through a regular LAN. *Source route bridging* utilises a different strategy. Here, most of the logic is stripped from the bridges, and put into the normal end nodes. The sending node is responsible for calculating the complete path to the destination, and include this in the transmitted frames. Source route bridging has no equivalent to the spanning tree protocol used in transparent bridging. It allows multiple paths between two nodes to exist, and leaves it to the end nodes to choose between these.

IBM developed Token Ring in the seventies, as their main LAN technology. It was later submitted to IEEE, and standardised as IEEE 802.5. To bridge Token Rings, IBM developed source route bridging. This algorithm was originally proposed to IEEE as a standard for bridging between all LANs. This did not gain enough support, and instead transparent bridging emerged as the standard bridging method in LANs. However, source route bridging has played an important role as a technology for bridging of Token Ring networks. The source route bridging functionality is explained through an example below.

3.3.1 Source route bridging example

Consider a situation where an end node in a source route bridged network wants to transmit to another end node residing on a different LAN segment. The first thing it must do, is to compute a path for the frame to travel. In figure 3.4, assume that host A wants to transmit to host B. Initially, it does not know whether host B resides on the same LAN as itself. To find out, it first sends out a test frame. If the test frame returns without indication that host B has seen it, host A concludes that it belongs to a remote LAN.

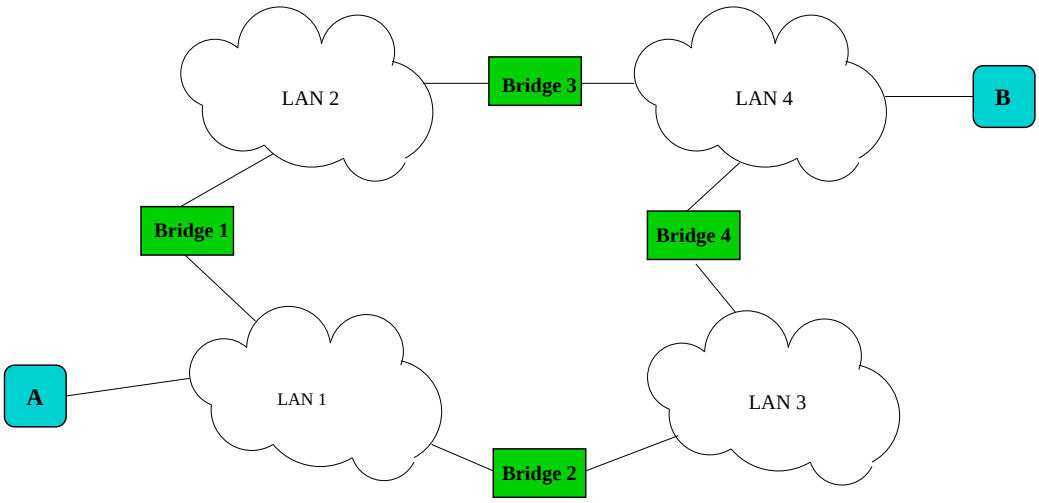


Figure 3.4: A source route bridged network consisting of four LAN segments and four bridges

To determine the exact location of host B, host A sends out an explorer frame. The explorer frame is forwarded by all bridges on the LAN (bridges 1 and 2 in this example), and routing information is added to the explorer frame as it passes. The explorer frame finally reaches host B (through bridges 3 and 4). Host B responds to each of the two frames received individually, using the accumulated routing information in the frames.

Host A thus receives two answers to the explorer frame, each representing a different path to host B:

- LAN 1 to bridge 1 to LAN 2 to bridge 3 to LAN 4
- LAN 1 to bridge 2 to LAN 3 to bridge 4 to LAN 4

Host A then has to choose which of these paths it wants to use for the transmission. IEEE 802.5 does not specify how this choice should be made.

Normally, host A would use the path contained in the first frame received. This is reasonable - the frame probably returned first because it travelled the shortest or least congested path.

When a path is chosen, it is inserted into a *Routing Information Field* in all frames sent from host A to host B. Only frames destined for a remote LAN segment contain such a routing information field. The presence of this field is marked by setting a bit in the frame header called the *Routing Information Identifier*. The bridges in the network inspect this bit, and extract routing information from the Routing Information Field if it is set.

Today, source route bridging has been replaced by a newer algorithm called *source route transparent* bridging (SRT). This technology provides a mean to interconnect ring networks using source route bridging as their bridging technology to other LANs using transparent bridging. Source route transparent was first developed by IBM around 1990, and is today specified in the 802.1D standard. In such a mixed bridging environment, stations depending on source route bridging as their bridging mechanism, may only communicate with other source route bridging stations. This is evident, since other stations will not be able to respond to an explorer frame as described above. Even with the emergence of this new algorithm, source route bridging is still often used in older systems.

3.4 The limitations of bridging

Transparent and source route bridging both belong to the IEEE 802 family, yet they still have different fields of application. Ethernet-like network technologies typically use transparent bridging, while source route bridging in the form described above is, to my knowledge, exclusively found in Token Ring environments. Each of these strategies have their strengths and weaknesses.

Transparent bridging has the most general approach, and is used to bridge all kinds of 802 LANs. Its major advantage is contained in its name - it is transparent. The end nodes do not have to worry about bridging, all the logic needed for the interconnection is contained in the bridge itself. This makes it easy to change the topology of the network, by inserting/moving/removing nodes or bridges. One of the drawbacks of transparent bridging, is that a network reconfiguration might trigger a so-called *broadcast storm*. When the spanning tree is changed by the spanning tree protocol, this forces all bridges to clear their filtering databases. The potentially vast number of broadcasts needed to rebuild these tables, can significantly limit network performance in the learning period that follows. Hence, transparent bridging is not well suited in networks with frequent topology changes. Also, in a

bridged network containing loops, some bridges will be put in a blocked state by the spanning tree protocol. These ports are then unable to help spreading the traffic over all available paths. This gives a sub-optimal utilisation of the network resources.

With source route bridging, no spanning tree exists, and the traffic may be balanced between the available bridges. There is also no filtering database in the bridges, so topology reconfigurations do not produce the same kinds of broadcast storms. However, every node has to broadcast explorer frames before they can transmit to a remote host. Hence, source route bridging generally produces more broadcasting than does transparent bridging. With source route bridging, each end node has to maintain a path to each remote station that it wants to transmit to. This increases the demands on the end nodes in the network. It also creates much redundant information in the network as a whole, since the same paths are learned by each host individually.

In general, both of these bridging technologies have an issue with scale. Bridging scales linearly, meaning that it provides no means to impose a hierarchy on the extended LAN. Also, all broadcast frames are forwarded by bridges. Broadcast frames meant only for the local LAN segment are thus propagated through the whole network, possibly wasting network resources. Due to these considerations, bridges are seldom used to interconnect more than a few LANs, where in practice "a few" typically means "tens of" [28].

The scaling properties of bridged LANs can be improved by the use of so-called *virtual* LANs (VLAN). The basic idea is to split a single extended LAN into several seemingly independent LANs. It is, however, outside the scope of this work to further discuss this.

3.5 Summary

In this section, we have looked at the two major strategies for bridging at the MAC level; transparent and source route bridging.

In transparent bridging, the end nodes behave as in a normal LAN, all the bridging logic is placed in the bridges. The bridges monitor the passing frames, and builds an image of the network topology.

In source route bridging, the end nodes are responsible for calculating a path to remote stations. It does this by sending out explorer frames that gather an image of the topology. Source route bridging is today rendered obsolete by the new source route transparent technology.

Chapter 4

Bridging RPR networks

With the background on RPR and bridging given in the previous chapters, it is now time to combine the two into the main topic of this work; bridging in RPR networks. This chapter explains the issues that must be addressed to make this possible, and discusses two different solutions.

First, section 4.1 explains how bridging is complicated by the destination stripping and spatial reuse properties of RPR. Then, section 4.2 explains how these issues are resolved in the existing RPR draft standard, and which other similar solutions have been discussed. These solutions have apparent weaknesses, which are also pointed at. The enhanced bridging algorithm described in section 4.3 is a response to these shortcomings. Several variations of the enhanced bridging strategy are possible. These are discussed, with emphasis on the algorithm that is implemented in the simulation model described in chapter 5.

4.1 The ideal case

Since IEEE started working on the RPR standard, it has been clear that a way to do bridging must be provided [30]. Being developed within the IEEE 802 framework, RPR is required to be compliant with transparent bridging as defined in 802.1D and 802.1Q. However, the ring nature of RPR makes this a non-trivial task.

Figure 4.1 shows a topology of three RPR rings interconnected by bridges. Consider a case where node X wants to send a frame to a remote node Y. Ideally, the frame should follow the path shown in the figure. This path secures the shortest travel for the frame, and, perhaps more importantly, the frame does not waste any unnecessary bandwidth.

Transparent bridging is designed to work over a *shared medium* network.

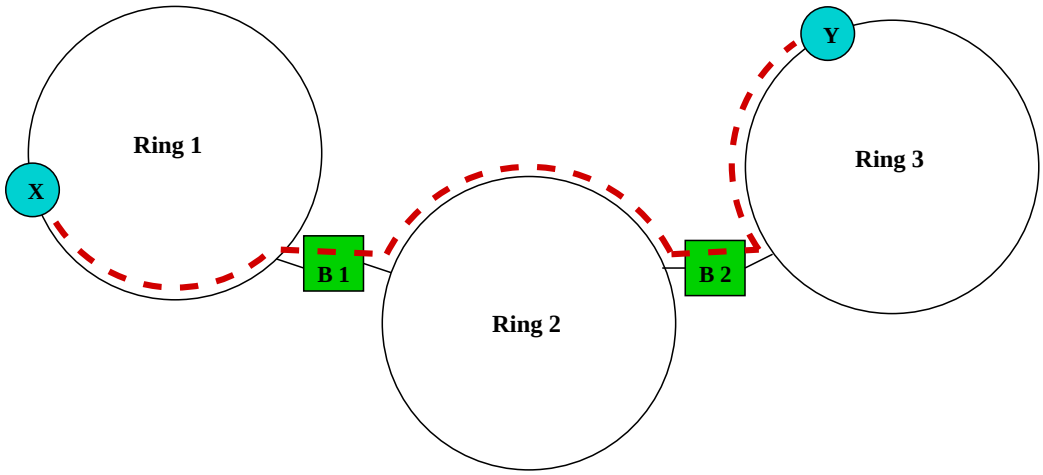


Figure 4.1: Frames from node X to node Y should ideally follow the shortest path, and never be broadcast on any ring

A shared medium network is a network in which all the attached nodes have access to all the frames that are put onto the medium that connects them. The most obvious examples of a shared medium network technology is Ethernet/802.3. In Ethernet, frames are transmitted on a single cable, shared by all the attached nodes. A bridge in such a network has access to all transmitted frames, just like every other node. All that is needed to implement bridging in such a network is the learning, filtering and forwarding functionality in the bridge itself, and the spanning tree protocol to avoid loops.

Things are not quite so easy in RPR. Since RPR is not a shared medium network, a bridge does not automatically see all transmitted frames. On the contrary, one of the important features in RPR is its spatial reuse properties, as explained in section 2.2. To achieve maximum bandwidth utilisation, every frame only traverses part of the ring before it is stripped by its destination. Frames either destined for or transmitted by a node that is not local on the ring, are referred to as *remote frames*. The destination stripping in RPR makes it necessary to give these frames some kind of special treatment. A way is needed to make sure a frame that should be bridged actually reaches the bridge. Ideally, this should be done without losing the spatial reuse properties of RPR.

Two fundamental strategies are possible to make sure that all remote frames reach the intended bridge. One possibility is to make sure that *all* remote frames are seen by *all* bridges. This is the strategy adopted in the RPR draft standard, and will be called *basic* bridging in the following text.

Another option is to send remote frames directly to the bridge that is responsible for forwarding it towards its destination. This alternative strategy is adopted by the *enhanced* bridging algorithm[1].

The next two sections will discuss these two bridging strategies.

4.2 Basic bridging in RPR

Frames are normally stripped at their destination in RPR. This strategy can obviously not be used for remote frames, since the destination node is not on the ring. Remote frames must be picked up by a bridge and forwarded to a remote network. As mentioned above, a way is needed to make sure that these frames actually reach the bridge. In basic bridging, this is solved by broadcasting all remote frames on the ring, so that they are seen by all bridges. This way RPR imitates a shared medium network, where all frames are visible to all nodes.

4.2.1 Promiscuous mode and its discontents

In one of the initial proposals to bridging in RPR, one imagined bridges to be operating in a *promiscuous mode* [11]. Bridges would, according to this proposal, pick up all passing frames, both local and remote, and copy them to their bridging relay. As discussed in section 2.3, unicast frames that are not destined for any of the local nodes on the ring, will continue around the ring until it is stripped at its source. A bridge operating in promiscuous mode would thus see all remote frames. Bridges operating in promiscuous mode would clearly be quite busy, as they would have to process every single frame passing by on the ring. This is, however, just what is done in a normal Ethernet LAN. In an Ethernet, all frames passing a bridge is presented to the bridge relay, which in turn makes a decision on whether to forward the frame based on the information in the filtering database. The main reason why the promiscuous mode method was not adopted, is the problems it encounters with respect to the learning process in the bridge relay.

This problem becomes evident in the situation illustrated in figure 4.2. With the strategy described above, frames from node A to node B would be picked up correctly by bridge 1. Bridge 1 would then send these frames directly to node B. Hence, bridges 2 and 3 would never see these frames, and never learn the location of node A. If node B responds to node A, this response would be picked up by bridges 2 and 3. Node A is unknown to these

[1]The terms basic and enhanced are adapted from a presentation given at the July 2002 meeting of the RPR working group [7].

bridges, and they would relay this response on all their ports. We would then have a situation where frames from node B to node A are persistently broadcast over several networks, clearly an unwanted situation.

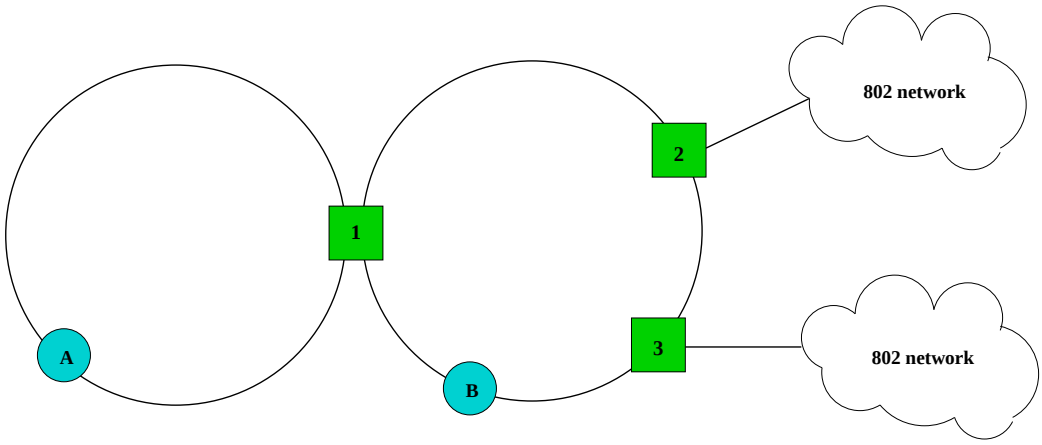


Figure 4.2: Frames that are sent directly from bridge 1 to node B are never seen by bridges 2 and 3. This prevents correct learning in the relay parts of bridges 2 and 3, and leads to persistent forwarding of frames from B to A by bridges 2 and 3.

4.2.2 Flooding

The problem above occurred when a remote frame was not seen by all the bridges on the egress ring, because it was destination stripped at the receiver. This illustrates the need for a special mechanism to make sure that remote frames are always seen by all nodes on *all* rings, including the egress ring. This mechanism is used in the current RPR draft standard (draft 2.1 [18]), and is called *flooding*.

Flooding can be looked upon as a special kind of broadcast. A flooded frame traverses the whole ring, and is thus visible to every node. It is not stripped by the destination node, instead the ttl field in the header (discussed in section 2.2) is used to limit the scope of a flooded frame.

A special bit in the RPR header is set to mark the frames as flooded. Flooding can be either *unidirectional* or *bidirectional*. With unidirectional flooding, the frame traverses the whole ring on one of the ringlets. In the bidirectional case, shown in figure 4.3, separate copies are transmitted on each ringlet. This helps balance the traffic load on the two ringlets, and reduces the maximum time from a frame is transmitted until it has reached

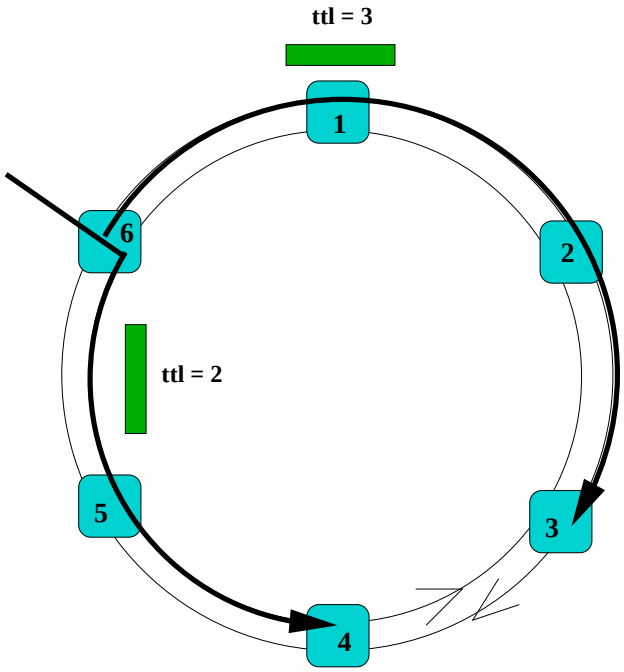


Figure 4.3: The ttl of a flooded frame is set so that it reaches each node exactly once. With bidirectional flooding, a separate copy of the frame is transmitted on each ringlet.

all nodes on the ring. The ttl is in both cases set so that the frame reach each node exactly once.

Bridges on an RPR ring inspect the flooding bit in the frame header. If the flooding bit is set, the frame is copied and presented to the relay part of the bridge. The relay then decides whether to forward the frame on some or all of its ports.

Flooding remote frames on the ring solves the problem with the bridge relay learning described above. To achieve this, remote frames must be flooded on *all* rings they pass, including the egress ring. In addition, it eases the task of the bridges, compared to the promiscuous mode solution. Instead of having to treat every passing node on the ring, only flooded frames are now presented to the bridge relay.

Note that the strategy of marking flooded frames with a special bit to make the receiving nodes give them a special treatment, is also used in Source Route Bridging. Recall from the discussion in section 3.3, that frames eligible for bridging is marked with a special Routing Information Identifier bit. This bit marks the presence of the Routing Information Field, which contains the path information for the frame.

The flooding approach is the bridging method that is referred to as basic bridging in the following text. Basic bridging is at the time of writing the only valid choice for bridging in RPR. It is possible that another bridging strategy, more similar to the one described in section 4.3, will be allowed as an option in a later version of the standard.

To sum it up, the main strategy of basic bridging is to make sure that all remote frames are seen by all bridges. This is achieved by flooding. With basic bridging, RPR can be said to act like a shared medium network for remote traffic, while keeping its spatial reuse properties for local traffic. By flooding all remote traffic on the ring, RPR makes sure that the frames reach the bridges, and that the necessary learning processes can take place in the bridge relays. However, flooding prevents spatial reuse for remote traffic, and consumes much bandwidth resources. This leads to the desire to find more bandwidth-efficient ways to do bridging.

4.3 An improved bridging algorithm

The efforts to make transparent bridging more efficient in RPR, has led to a strategy called *enhanced bridging*. The basic ideas behind enhanced bridging have been developed through discussions in the RPR working group and its subcommittees. However, enhanced bridging has been ruled out of scope of the current standardisation efforts. It is therefore not part of any current draft standard. The algorithm described in this section are mainly based on presentations given at the RPR working group summits [12, 6, 7].

The intuitive idea behind the enhanced bridging algorithm is to enable a frame to travel the shortest path from the source to the destination, as shown in figure 4.1 on page 38. The relaying bridges on the path is responsible for stripping the frames from the ring, thus securing the spatial reuse properties.

4.3.1 SRCS tables

The purpose of enhanced bridging is to allow spatial reuse for remote traffic. This is achieved by letting the end nodes address remote frames directly to the bridge that is responsible for forwarding traffic to the specific receiver. The end nodes then obviously need some way to keep an association between a remote host and its designated bridge. Every node must keep and maintain a table that maps a remote address to a local (bridge) address. The functionality that builds, maintains and uses these tables is known as the

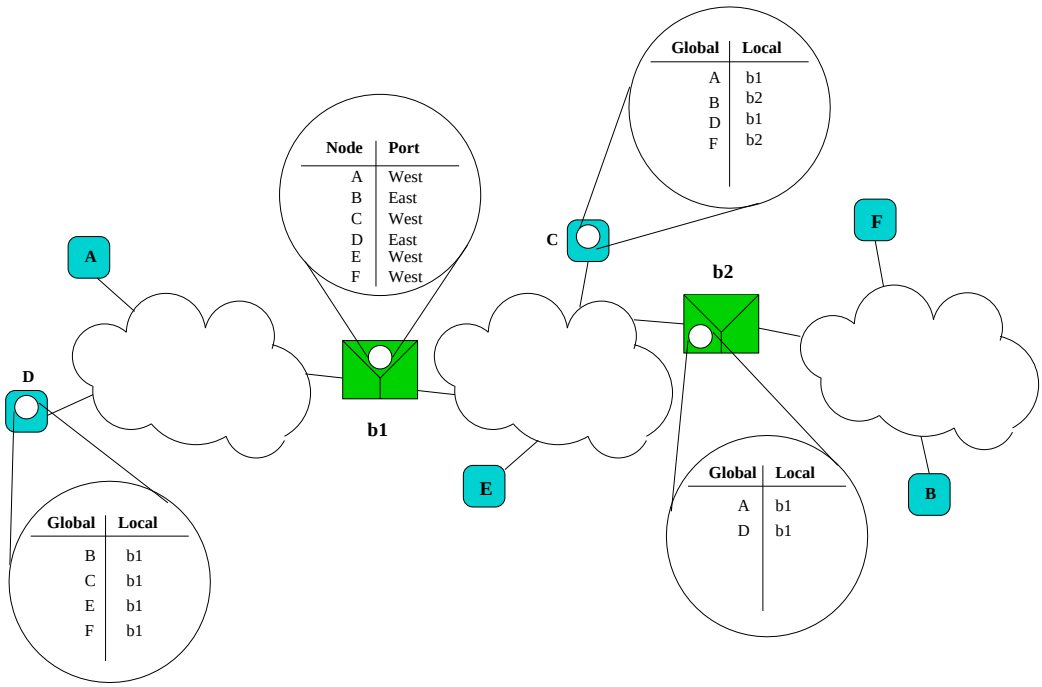


Figure 4.4: With enhanced bridging, all nodes are equipped with tables that map global remote node addresses to the local bridge that is responsible for forwarding frames towards that node

Spatial Reuse Control Sublayer (SRCS)[2]. The global-to-local mapping tables are therefore termed SRCS tables in the following.

Figure 4.4 shows an example network with SRCS tables in the end nodes. Recall from section 3.2 that a transparent bridge consists of an interface to each of the networks it interconnects, called a port, and a relay unit that forwards frames between the ports. The bridge ports can be seen as nodes in their respective networks. Hence, each bridge port that belongs to an enhanced bridging network contains a SRCS table, just like any other node in the network. In addition, the bridges contain the normal filtering database in their relay unit.

The idea of keeping SRCS tables in the end nodes gives enhanced bridging some similarities with source route bridging as discussed in section 3.3. The bridging functionality is no longer only held in the bridges, some logic to support bridging is required in the end nodes. Furthermore, with the enhanced bridging algorithm, the end node provides the first step of a remote frame's

[2]The term SRCS is adapted from a presentation given by the RPR working group's bridging ad-hoc subcommittee [7]

path, by identifying the local bridge that is responsible for forwarding the frame. With source route bridging, the sender is responsible for calculating the complete path to the receiver. Enhanced bridging thus takes one step in the direction of source route bridging, away from the transparent bridging idea of "no bridging awareness in the end nodes".

The SRCS tables are built dynamically, based on the traffic received by the end nodes. A discussion of this learning process is given below, but first a look at the enhanced bridging frame format.

4.3.2 Frame format

The enhanced bridging algorithm demands some changes in the RPR frame format discussed in section 2.2. In addition to the original source and destination MAC addresses, some way to identify the *local* sender and destination nodes on the ring is needed. This can be achieved in several different ways.

One possibility would be a tunnelling-like approach, where the original RPR frame is encapsulated in a new frame. The outer frame is given the bridge as its destination address. When the frame reaches the bridge, the original frame is restored. The bridge will also have to insert its own address as the source address in the frame header when the frame is relayed to the new network. This strategy will not demand any changes in the frame format for local traffic. Only remote frames will get the additional overhead of an extra RPR header.

Another option is to insert a pair of extra addresses in the frame header of all frames, to identify the local source and destination. This is the strategy that is used in the simulation model described in section 5. Figure 4.5 shows the RPR frame format with these extra fields inserted.

The local source address, called the Sending Station ID (SSID), is set to the MAC address of the node that actually transmits the frame on the ring. This can be either the original sender, in which case the SSID is equal to the source address, or it can be the address of a bridge that relays the frame onto the ring. The local destination address, called the Destination Station ID (DSID), is set to the MAC address of the node that is responsible for removing the frame from the ring. This can be either the destination node, or a bridge. The DSID is used by bridges to determine whether they should strip the frame or not.

In figure 4.5, 48 bit MAC addresses are used to identify the local sender and receiver. This is convenient, since every node is already equipped with this identifier. However, notice that the local station identifiers only have to be unique within the ring. This means that the 48 bit SSID and DSID could be replaced with 8 bit identifiers, to reduce the 12 bytes of extra overhead

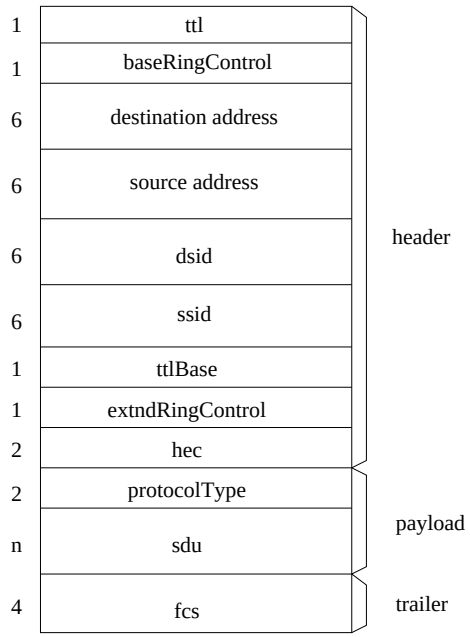


Figure 4.5: Enhanced bridging demands some way of identifying the local sender and destination nodes. This figure shows a possible way to achieve this.

put on every frame.

Note that local frames do not use the SSID and DSID fields in the header. These fields thus give an unnecessary overhead on local traffic. This can be avoided by always letting the source and destination addresses in the frame header be local addresses. The global source and destination addresses of remote frames are instead put at the beginning of the frame payload. A bit in the extndRingControl field of the header can be used to mark the presence of these extra address fields in remote frames. When a remote frame arrives at a bridge, the global addresses found in the payload are used to give the new source and destination addresses to the frame when it is forwarded to a new network. This strategy implies that there are in fact two different RPR frame formats - one for local and one for remote frames.

The purpose of the DSID field, is to allow remote frames to be unicasted to the correct bridge, when this is known in advance. The SSID field is needed for learning purposes, as described below. The important thing here is that *some way* to identify the local sender and receiver is needed. An interesting proposal provides a way to derive the information needed about the local sender and receiver, while minimising the header overhead and sticking to a single frame format [8]. In this proposal, the 48 bit source and destination

MAC addresses used in the frame header are always global node addresses. In addition, an eight bit sending station identifier is used to determine the local sender. The main purpose of this field is to protect against frame duplication and reordering in case of a protection event, as discussed in section 2.4. Each node on the ring is assigned such a local station identifier during topology discovery. There is no field in the header for the local destination with this proposal. Instead the local destination is decided by using the ttl and ttl_base fields discussed in section 2.2. Bridges pick up frames where the ttl has reached zero, and determine if they should be forwarded to a remote network. With this algorithm, the nodes no longer maintain a local-station-to-remote-station address mapping in their SRCS tables, as described above. Instead, they keep a mapping between remote hosts and the number of hops to the local destination. When a frame is transmitted to a remote receiver, the ttl and ttl_base parameters are set so that the frame exactly makes it to the bridge. The bridge, noticing that the frame has a ttl of zero, copies the frame to its relay, and makes a forwarding decision based on the relay forwarding table.

4.3.3 The learning process

The SRCS tables described above are built dynamically, based on the traffic received in the end nodes. This section describes the learning process based on a frame format with SSID and DSID fields in the header, as described above. This is the way learning is performed in the simulation model described in section 5. A similar learning process would be needed with the other frame formats discussed above.

The basic idea in the learning process is to compare the global source address field of the incoming frames to the SSID field. A bridge that forwards a frame, inserts its own MAC address as the SSID. The node that strips the frame from the ring, will thus learn that frames transmitted by this specific remote node, comes through that specific local bridge. Before the SRCS tables are built, the nodes rely on flooding when transmitting remote frames.

For an example of how this learning process takes place, return now to the scenario in figure 4.1 on page 38. Initially, end nodes X and Y both have empty SRCS tables. When X wants to transmit to Y, it will find no entry for Y in its SRCS table. It will therefore flood the frame over ring 1, just as it would do with the basic bridging algorithm. The source address and the SSID of the frame is set to X's MAC address. Both the destination address and the DSID is set to Y's MAC address. Note that the DSID is not used in this case, since the frame is flooded. Bridge B1, as well as any other bridge on ring 1, will see that the frame is flooded, copy it to its relay, and forward

it on all ports. Since the ring 2-port of B1 does not have Y in its SRCS table, it will flood the frame on ring 2. As it does so, B1 sets the SSID of the frame to its own MAC address. Note here again that enhanced bridges keep two separate kinds of tables. The filtering database resides in the relay part of the bridge, and keeps the association between end nodes and the port to which it belongs. This table is present in any 802.1D/Q compliant bridge. The SRCS table resides in the MAC dependent part of the bridge, and keeps the association between global and local node identifiers. Every node in an RPR network that supports enhanced bridging has such a SRCS table.

The frame flooded by B1 is picked up by B2, which learns that frames from end node X is sent through B1. B2 then inserts its own address as the SSID, and sends the frame directly to node Y. Node Y learns that frames from X goes through B2. Note that so far, no efficiency gain is achieved by enhanced bridging compared to basic bridging, except that the frame was not flooded on ring 3. The main gain comes only after the nodes on the path have learned the mapping between the remote address and the local node identifiers. When Y now wants to respond to X, no flooding will be necessary. Node Y now has node X in its SRCS table, and so inserts the address of X as the destination address, and the address of B2 as the DSID. The frame is thus sent directly to B2, which forwards the frame to ring 2, inserts its own address as the SSID, and inserts B1 as the DSID. The frame can now be sent directly to bridge B1. B1 does the same as B2 did; it changes the SSID and DSID to the address of B1 and X respectively, and it also learns that frames from node Y goes through B2. Finally, B1 can send the frame directly to X. When node X now receives the frame and learns that frames from Y goes through B1, the learning process is completed. All subsequent communication between nodes X and Y can now take place without any flooding.

An important thing to understand, is that there are two separate learning processes going on simultaneously in the above example. The one described, is the learning of the mapping between global end station addresses, and local station identifiers. This learning takes place in all the nodes supporting enhanced bridging, both the end nodes and the bridges, and results in entries in the SRCS tables. At the same time the bridges perform their learning of host-to-port mappings as described in section 3.2, which enables them to decide which frames are to be forwarded on which ports. This learning results in entries in the filtering database in the bridge relay.

As seen from this example, some flooding is still necessary with the enhanced bridging algorithm before the SRCS tables have been built across the network. The amount of flooding needed is investigated in scenario 1 in section 6.1. Some steps can be taken that would possibly reduce the amount

of flooding. First, it is reasonable to believe that many rings only contain one or at least very few active bridges. An active bridge here means a bridge whose port to this ring has not been blocked by the spanning tree algorithm as discussed in section 3.2. The SRCS tables could then be equipped with a default bridge address, decided during topology discovery. Frames destined for remote nodes that did not match any of the other entries in the SRCS table, would be sent to this default bridge. The bridge would then be responsible for the further handling of the frame. If the bridge knew the location of the destination node, the frame would be forwarded as normal. Else, the bridge would flood the frame on all ports, including the one it arrived on.

Another possibility is to run a "global" topology discovery algorithm when several RPR rings are connected by bridges. This way, the SRCS tables in the end nodes could be built complete before any traffic was sent in the network. This strategy would, however, generate more traffic in the network when building and maintaining the global topology image.

4.4 Summary

This section has introduced the problems that arise when bridges are used to interconnect RPR rings. The destination stripping used in RPR makes some special treatment of remote traffic necessary, to ensure that these frames reach the designated bridge.

The basic bridging algorithm currently in the RPR draft standard, solves this by flooding all remote frames on the ring. Enhanced bridging avoids the flooding after a learning period, by equipping the end nodes with SRCS tables and sending remote frames directly to the bridges.

Chapter 5

The Java simulation model

The previous chapter discussed an enhanced bridging algorithm for RPR networks. The next major goal of this work is to evaluate the performance of this algorithm, and compare it to the basic bridging algorithm. This is done using a simulation model of an RPR network, written in the Java programming language. The development of this model has been a substantial part of the work in this project.

The simulation model described in this section was originally developed by professor Stein Gjessing at Simula Research Laboratory. His model was built to simulate traffic with different priorities on a single RPR ring. Basic functionality like buffer handling and event scheduling is kept unchanged from this original model. I have later extended this model to handle several rings connected by bridges. A more detailed outline of which parts of the original model have been kept unchanged and which parts have been altered or added, is given towards the end of this chapter.

This chapter will give a detailed description of the simulation model. First, section 5.1 briefly discusses some of the principles used in discrete event simulation models. Then, section 5.2 gives an overview of how the Java model is built, and how some of the different classes interact. This section also explains how the implementation of enhanced bridging differs from the basic bridging model. Section 5.3 provides a more detailed look upon some of the most important classes in the model, with a description of how they are related to the RPR standard. Finally, section 5.4 is given to make clarify the relationship between this simulation model and the model that existed before my work began.

5.1 Discrete event simulators

A *simulator* is a device used to reproduce under test conditions phenomena that are likely to occur in the real world. Simulators are useful as tools in many different settings; from modelling flows in oil pipes to simulating factory assembly lines or training aircraft pilots. Such simulators give the advantage of being able to predict a system's behaviour without having to work on the actual system. This can give considerable savings in terms of both money and development time. Simulators can be either real world models such as wind tunnels, or computerised models.

The simulator used in this work is a computer program written in the Java programming language, used to simulate the behaviour of an RPR network. This model is a *discrete event* simulator, where the changes in the system are represented as separate events. This as opposed to continuous models, where the time advances in a continuous fashion, based on a differential function.

Three essential parts of a computer simulation model are the *entities* that are modelled, the *relationship* between these, and the *simulation executive*. Entities are the real world elements that are the subject of investigation. For the model here described, these include nodes, links and data frames. Note that while nodes and links are permanent entities, frames are temporary elements that are created and deleted during the simulation. The objective of a simulation is often to observe the behaviour of the temporary entities as they flow through the permanent ones.

The entities in a model are linked together by logical relationships and statements. These relationships define the behaviour of the model. An example of a logical statement can be "start transmitting if outgoing link is free".

Finally, a simulation executive is needed. The executive is responsible for controlling the time advance of the simulation. Two basic approaches are possible for controlling the time advance; *time slicing* and *next event* [5]. With time slicing, the system moves forward in fixed time intervals. The model is moved from one time step to the next, regardless of whether anything will happen between the two. The next event approach is generally more efficient, and is the one used in this work. Here, the model is advanced from one significant event to the next, no matter how long time has elapsed between the two. The simulation executive maintains a sorted queue of the scheduled events, as shown in figure 5.1. Each event contains the time at which it is to be executed, and a reference to the logic that should be performed.

The executive goes in a loop, removing the event at the head of the queue, and performing the associated logic. This event might in turn generate new

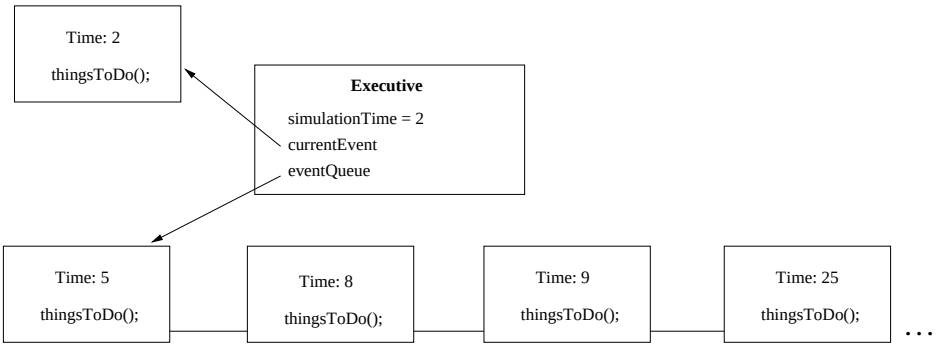


Figure 5.1: The simulation executive in a discrete event simulation model maintains an ordered queue of events. It also keeps track of the simulation time.

events, which must be inserted into the queue. When one event has been handled, the executive moves to the next event in the queue, and advances the simulation clock to the time of this event.

In addition to the entities, the relations between these, and the executive, a simulation model is also dependent on random number generators and a mechanism for gathering the results of the simulation.

5.2 About this simulation model

The simulation model used in this work is a discrete vent simulator as described above, using the next event approach for advancing the simulator clock. The model is used to simulate traffic in RPR networks with different topologies. The results from these simulations are then used to evaluate and compare the two bridging strategies described in section 4. This is done by modelling a number of RPR rings consisting of nodes connected with links. These rings are then interconnected by bridges, and statistics are gathered as frames are sent through the network.

The RPR nodes in the model can be of two different kinds. The first type supports only basic bridging, and is referred to as a *basic node* in the following text. The second type supports enhanced bridging, and is thus named an *enhanced node*. The model allows these two kinds of nodes to coexist on the ring. An RPR ring in this model thus consists of either basic nodes, enhanced nodes, or a combination of the two. Figure 5.2 shows how the two kinds of nodes are composed of several java classes. The major difference is the Spatial Reuse Control Sublayer in the enhanced nodes. A more detailed description of the various classes is given in the next section.

Bridges are used to interconnect the RPR rings in the model. The model

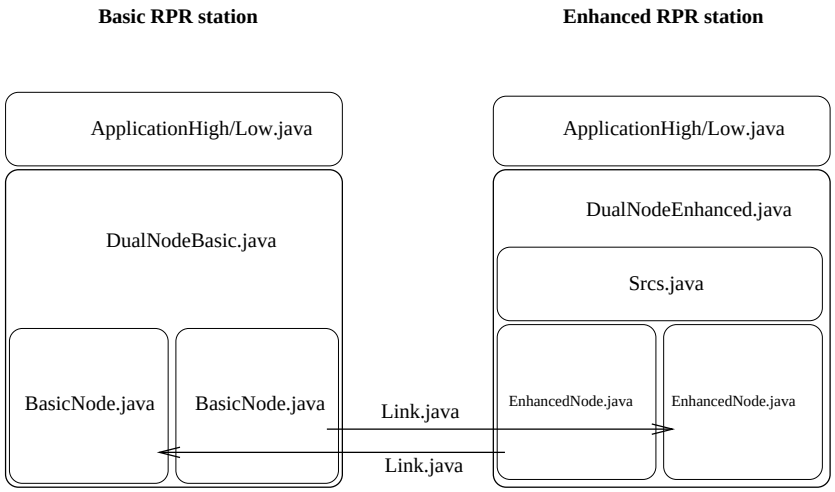


Figure 5.2: The two different kinds of RPR nodes are built of several classes. The most significant difference between the two is the class Srcs.java in the enhanced nodes.

allows up to four bridges to exist on a ring, and each bridge can interconnect either two or three rings. These limitations are of a purely practical nature, and can easily be overcome if a more exotic topology is wanted. A bridge in the model consists of one node from each ring it interconnects, and a relay unit that performs the forwarding of frames from one ring to another.

The simulation model described in this work is a Java program of over four thousand code lines; hence it contains errors. There are probably shortcomings in the way the RPR specification has been transformed into a simulation model. And there are certainly programming errors distributed around the code, that might cause the model to misbehave. More extensive testing and verification of the model would have been advantageous. However, the model has been run with quite a few different inputs and configurations during the work with this thesis, producing meaningful results. Single frames have been followed through the model, and are seen to behave as expected and according to the specification. Tests have been made to check that frames arrive at their correct destination and in correct order. Furthermore, the simulation results presented in section 6 seems to be reasonable, at least explainable. It is therefore my opinion that this model is a quite credible basis for drawing conclusions about the performance of the enhanced bridging algorithm.

In total, the simulator model consists of about thirty java classes. Many of these describe different entities found in a real world computer network, such as nodes, links, buffers, packets etc. The methods in these classes also make up the logical relationship between the entities. Other classes take

care of the execution of the simulation itself, and the gathering of statistics. The next section will give an overview of how the most important classes are constructed and related.

5.3 The important classes

The goal of this section is to give a detailed description of some of the central classes in the simulator. Helper-classes and other classes that are not central to understand the implementation of the model are omitted. UML notation is used to show the relations between the different classes. Readers who are interested in further implementation details, please refer to the source code available at the URL given in chapter 1.

The class descriptions start with a look at the simulation executive. It then takes on the main classes describing the entities in the network and the relationships between these. Finally, the reporting functionality of the simulation model is discussed.

5.3.1 Kernel.java

The simulation executive is contained in the class called Kernel. This class maintains a queue of scheduled events, and the simulator clock that shows the current time. The event queue is arranged as a binary tree, the root being the first event to take place. Examples of events can be that a new frame is ready for transmission from the MAC client, a frame has arrived at the end of a link, or a frame has completely entered or left a buffer. Every schedulable class must extend the abstract class Unit, and contain a method named *act*. When the simulator clock reaches the time at which an event is scheduled, the act method of the class in question is called.

Kernel also contains the method *simulate*, which starts the simulation. It goes through the event queue, calling the act method of each scheduled event until a given stop time is reached.

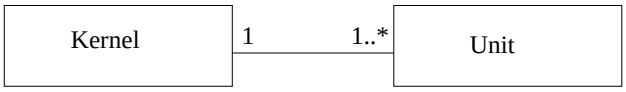


Figure 5.3: The Kernel class maintains a queue of schedulable objects, which all must be subclasses of the abstract class Unit.

5.3.2 Packet.java

An RPR frame is modelled in the class *Packet*. A Packet is the only temporary entity in the model. Packets are created by the application classes described below, and spends their life flowing between the permanent entities in the model. The Packet class is not in direct correspondence with the RPR frame format described in section 2.2. It only contains information that is needed for the simulations described in chapter 6.

Frames eligible for enhanced bridging will need extra fields in the RPR header to decide the local sender and destination addresses, as discussed in section 4.3. In the Packet class, a boolean value is used to decide whether a frame is sent from an enhanced node, and hence contains these fields. The equivalent of this boolean value is found in the extRingControl field of the RPR frame. Packet also contains values to represent other relevant fields of the RPR header, such as the size of the frame, which service class it belongs to, the timeToLive value of the frame, and whether the frame is flooded or not.

In addition, the Packet class contains information that is not part of the RPR frame format, but is used in the simulator to gather statistics. These include the time of creation and information on when the frame enters and leaves buffers. Since Packet is a schedulable class, it also contains an act method.

5.3.3 ApplicationHigh.java and ApplicationLow.java

As mentioned in section 2.3, RPR allows three different classes of traffic, with different priority characteristics. In the simulator model, only two priority levels are implemented. The classes *ApplicationHigh* and *ApplicationLow* play the roles of traffic generators, and generate frames with high and low priority respectively. Each RPR node can have any number of such applications associated with it.

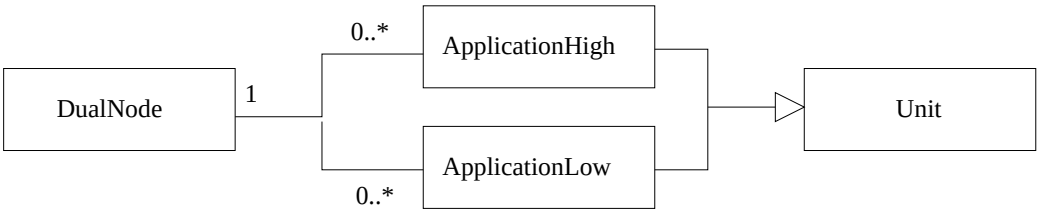


Figure 5.4: Each instance of ApplicationHigh or ApplicationLow belongs to a specific DualNode object. Since they are schedulable, they must extend Unit.

When the act method is called in an ApplicationHigh or ApplicationLow object, a new Packet object is initialised. Sender and receiver addresses and other parameters are set, and the frame is transmitted. The class then reschedules itself to the time of the next transmission. The size of the frames and the time between each transmission can either be static, or given by a chosen random distribution.

5.3.4 DualNode.java

When a Packet is generated, it is sent to the MAC layer for transmission. The MAC layer of a station is represented by the *DualNode* class. Much of the functionality in the MAC data path described in section 2.3 is separate for each ringlet. This functionality is collected in the *Node* class described below. A DualNode consists of two such Node classes, one for the inner and one for the outer ringlet. In addition, DualNode has functionality for choosing which ringlet to transmit on, and for flooding frames over the ring. One of the other MAC control sublayer functions described in section 2.4, fairness, is mainly implemented in the class *FlowControl*. The flow control functionality was part of Stein Gjessing's original model, and has been left almost unchanged. Other functions found in the MAC control sublayer, such as protection and topology discovery, are not part of the model.

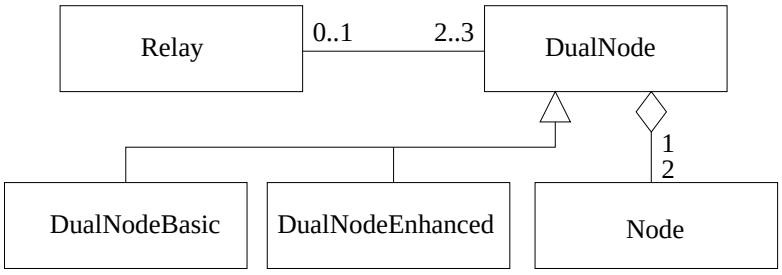


Figure 5.5: DualNode has two subclasses, to represent basic and enhanced RPR stations respectively. It also contains two Node objects, one for each ringlet. Finally, DualNodes that are part of a bridge are associated with a Relay.

The simulator is used to simulate traffic in a network implementing enhanced bridging, basic bridging, or a combination of the two. The functionality of DualNode is slightly different in the enhanced and basic case. DualNode is hence implemented as an abstract class, an instance of the class being either a *DualNodeBasic* or a *DualNodeEnhanced*. The main difference between these is that an enhanced station contains a Spatial Reuse Control Sublayer (SRCS), described below.

A DualNode object can be turned into a port in a bridge at initialisation by calling the method *turnIntoBridge*. The DualNode will then be associated with a Relay object, as described below.

5.3.5 Node.java

After DualNode has decided which ringlet to transmit the Packet on, it is passed on to a *Node*. The Node class implements MAC data path functionality, dealing with frame reception, transit, and transmission at one of the ringlets. The Node class implements a dual transit buffer data path, as discussed in section 2.3.

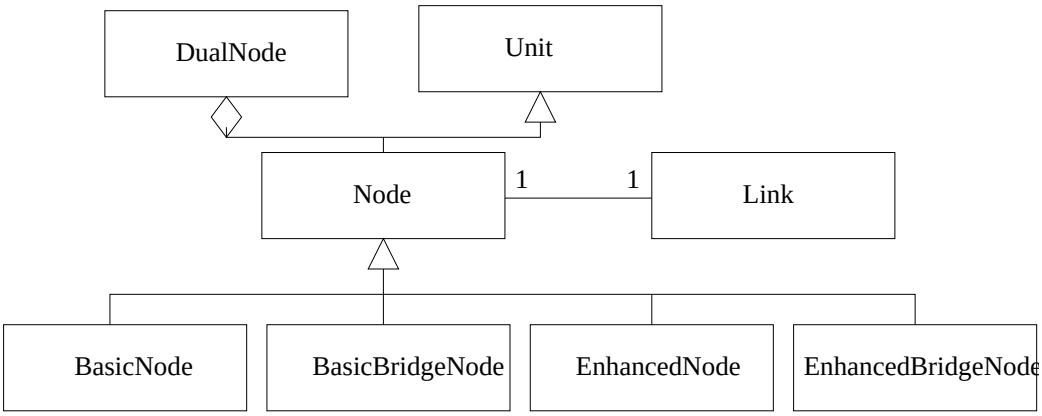


Figure 5.6: The abstract class Node has four subclasses. It is also associated with a Link, onto which it transmits frames.

As with DualNode, the functionality in Node is dependent on whether or not it supports enhanced bridging. In addition, a Node should behave differently if it is part of a bridge. This leads to a structure where the abstract class Node is extended by the four classes *BasicNode*, *EnhancedNode*, *BasicBridgeNode*, and *EnhancedBridgeNode*. The difference between these classes is the *stripper* method, which is called when a Packet arrives at a node. Bridge nodes might pass a copy of the Packet to the bridge relay, while enhanced nodes pass Packets destined for this node to the SRCS described below for learning.

The rest of the Node functionality is the same for all the four subclasses. They all contain the method *outputSelector*, which is responsible for picking the next frame to be sent from the node. This can be a high or low priority frame from one of the transit queues, or a frame added by this node.

The goal of the simulations described in this work is not to test the error recovery capabilities of RPR. The Node class does therefore not contain any

error recovery functionality. It does, however, implement Virtual Output Queueing as described in section 2.4. Each Node thus contains one high priority transmit queue, plus one low priority queue for each other node on the ring.

5.3.6 Link.java

Each Node object is associated with an outgoing *Link*. The links are used to connect the nodes on the ring. Link is a passive component in the model, meaning that it has no act method, and can not be scheduled in the simulator kernel. The most important characteristic of a Link is its length, which decides how much time a Packet needs to traverse it.

5.3.7 Srcs.java

All enhanced nodes contain a Spatial Reuse Control Sublayer (SRCS). All traffic to and from the MAC client of an enhanced node must pass through this sublayer. The purpose of the SRCS is to make possible spatial reuse of bridged traffic, through inserting source- and destination station identifiers as explained in section 4.3. The SRCS also keeps the table that maps remote MAC addresses to station identifiers that are local to the ring.

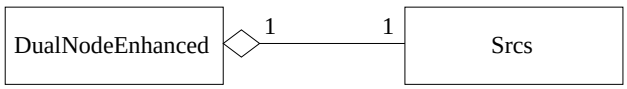


Figure 5.7: Srcs is a part of the DualNodes that support enhanced bridging

Srcs receives Packets from the ring through the *receive* method. This is where the learning operation takes place. The method inspects the global and local sender ids of the frame, and updates the mapping table if necessary. The frame is then passed to the MAC client represented by an Application class described above, or to the bridge relay if this node is a bridge.

Add traffic from this node to the ring passes through the SRCS via the *put* method. Here, the mapping table is used to insert a local destination id if needed. The Packets are also marked as coming from an enhanced sender, to make other nodes aware of the presence of local sender and destination fields.

5.3.8 Relay.java

When a Packet reaches a bridge, it is passed from the MAC layer and over to the relay part of the bridge. As mentioned in section 3.2, the functions

performed in the bridge relay are independent of the MAC technology. So, the Relay makes no distinction between enhanced and basic Packets.

Relay offers a *put* method, that receives Packet objects from one of the attached nodes. If the Packet comes from a hitherto unknown node, the Relay updates its forwarding table. Based on this forwarding table and the destination address kept in the Packet object, the Relay makes a decision about which ports the frame should be forwarded to.

A bridge as defined in IEEE 802.1D consists of a Relay and any number of ports, at least two. In my model, the number of ports is limited to three; a Relay can be attached to two or three DualNode objects.

5.3.9 Reporter.java

Different kinds of statistics are wanted as output from the simulations. Depending on the purpose of the simulation, one might be interested in registering that a frame has reached its destination, a frame is flooded, a frame has traversed a link, a table is full, or any other event. The role of the Reporter class is to gather such information, and to write it to files that can later be analysed. To achieve this, Reporter offers a variety of methods that can be called at any point during the simulation. Examples of such methods are *reportP*, registering that a frame has traversed a link, and *reportFlood*, registering that a frame has been flooded on a ring. Reporter writes different statistics to files, which later can be further processed or plotted in graphs.

5.4 About the development of this model

As stated in the introduction to this chapter, parts of this simulation model existed before my work on this project began. The original model was developed by professor Stein Gjessing at Simula Research Laboratory in Oslo. His original model was built to simulate traffic on a single RPR ring. Major parts of this model has been kept virtually unchanged. This includes the simulation executive in Kernel.java, the classes that handle the buffers in the various nodes, and the fairness functionality.

My own contribution to the model has been to extend it to allow scenarios with multiple RPR rings connected by bridges. This new functionality has demanded changes and additions in many of the classes, and the splitting of the Node and DualNode classes into several subclasses. Furthermore, some new classes have been needed to represent entities such as rings, bridges, relays and SRCS sublayers.

Even if most of the classes have been changed to some extent, the basic structure with Node-, DualNode-, Link- and Application objects have been kept. This basic structure, and the existence of a simulation executive, has made the development of the existing model much easier, and has made it possible to finish this work within the intended time limit.

5.5 Summary

The model described in this chapter is used as a tool for evaluating the enhanced bridging algorithm. The model is a discrete event simulation model, with an executive that maintains an ordered event queue.

In addition to the simulation executive, the model consists of a number of Java classes representing the real world entities found in an RPR network. All schedulable classes contain an *act* method, which is called by the simulation executive when the scheduled time is reached. The model also contains classes for gathering statistics from the simulations, and write these data to files.

Chapter 6

Simulations and results

Chapter 4 gave a description of two different strategies for doing bridging in RPR networks. *Basic* bridging is the straightforward approach. It stays almost true to the transparent bridging principle that all the bridging logic should be contained in the bridges. The end nodes always flood remote frames over the ring, to make sure they are seen by all bridges.

Enhanced bridging goes further in moving bridging functionality from the bridges themselves and into the end nodes. The end nodes use their Spatial Reuse Control Sublayer to find the address of the first bridge on a remote frame's path. The frame can then be sent directly to that bridge. This way, enhanced bridging moves one step in the direction of source route bridging, as discussed in section 3.3. With source route bridging, the sender node is responsible for calculating the whole path to the destination, and include it in each remote frame.

The simulations described in this section are performed to evaluate the different behaviour of these two bridging strategies. Specifically, there are three aspects that this work tries to shed some light on:

- The enhanced bridging algorithm reduces the amount of traffic in a network by maintaining the spatial reuse properties for remote traffic. How significant is this traffic reduction? And how long is the learning period before this reduction is achieved?

- To allow spatial reuse of remote traffic, the enhanced bridging algorithm depends on remote-to-local mapping tables in the end nodes. How is the traffic in the network affected if these tables are of limited size?

- The reduced traffic on an RPR ring achieved by enhanced bridging means that the probability that a link gets congested declines. This in turn has a positive effect on the latency and jitter characteristics of

the traffic. How is the local traffic on the ring affected by the choice of bridging algorithm?

To investigate these questions, three different network scenarios are created and simulated. The performance metrics that are extracted from these simulations, will be helpful to provide some possible answers.

Note that the simulation model does not contain an implementation of the spanning tree protocol discussed in section 3.2. This algorithm is used to make a bridged network free of cycles. The exchange of information between bridges to accomplish this is independent of which MAC method is used in the individual LANs. Whether the basic or the enhanced bridging algorithm is used, does not affect the spanning tree protocol. All the scenarios discussed in this section use cycle-free topologies. This is a reasonable choice - cycles would have been removed by the spanning tree protocol as soon as redundant paths were discovered.

6.1 Scenario 1: A simple flow

The apparent advantage of enhanced bridging, is reducing the amount of flooding in an RPR network. Enhanced bridging allows frames to be sent directly to the bridge, after an initial period of learning. The purpose of this simulation is to compare the amount of traffic generated by a simple data flow, when the basic and enhanced bridging algorithms are used. The simulation is run twice. In the first case, all the nodes involved support enhanced bridging, while in the second case, basic bridging is the only strategy supported.

The simulation and results described here, are almost identical to the ones given in a presentation at the September 2002 meeting of the RPR working group in New Orleans [20].

6.1.1 Topology

The topology used in this simulation is shown in figure 6.1. Four identical rings are connected by bridges. The rings each have eight nodes, including the bridges. The links that connect the nodes in the simulations are about 500 meters, giving a total ring span of about 4 kilometres. The simulations have also been run with link lengths of 5 kilometres, giving very similar results. The available link capacity is about 8 Gbps, almost the speed of an OC-192 optical fibre without the SONET overhead.

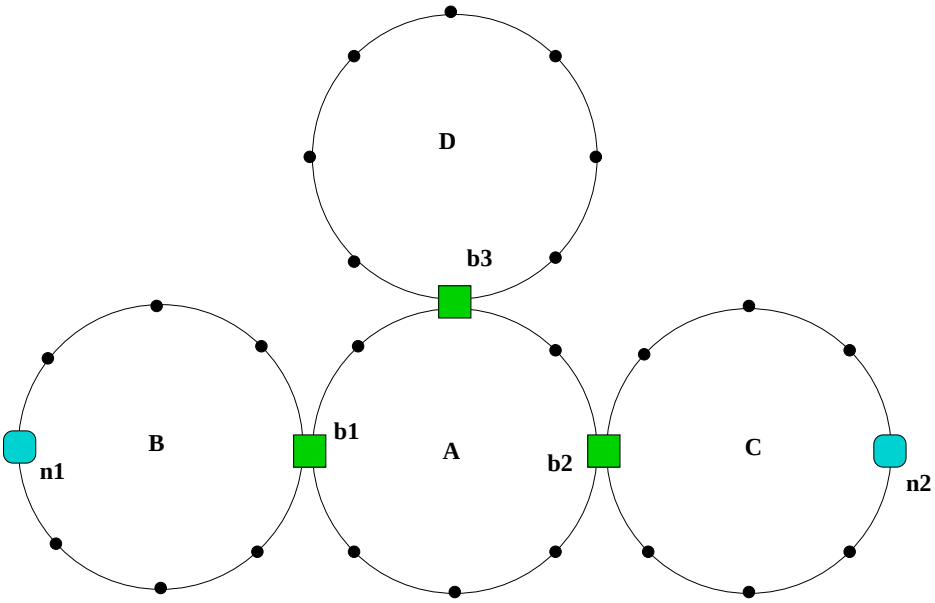


Figure 6.1: Two nodes communicate over a bridged network consisting of four RPR rings connected by three bridges.

6.1.2 Traffic pattern

The traffic flow in this simulation is deliberately kept very simple, so that the resulting statistics should be easy to read. The purpose is not to provide a realistic real-life traffic scenario, but to isolate a single data exchange between a pair of nodes. Only two nodes are active during the simulation, all the other nodes simply participate in the forwarding of frames. On a real ring, there would necessarily be a small amount of background traffic, due to the operations of the MAC control sublayer. These are ignored in this simulation, leaving only the data flow of interest.

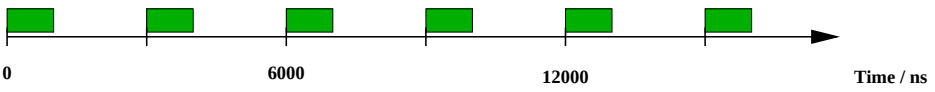


Figure 6.2: Node n1 sends 1kB data frames to node n2 at a constant rate equal to one third of the link capacity.

In the simulation, node n1 sends a continuous flow of low priority data to the remote node n2. The frames sent have a size of 1kB, and are sent at a rate that equals one third of the link capacity. They are sent at equal

intervals, as illustrated in figure 6.2. Upon receipt of each frame from node n1, node n2 returns a small "ack" frame. The size of these frames are 80B. Thus, we really have two flows in this simulation, a flow of 1kB "data frames" from n1 to n2, and a flow of 80B "acks" in the opposite direction. This gives a TCP-like traffic pattern.

Note that no random numbers are used to generate the traffic in this scenario, so the simulation generates exactly the same output each time it is re-run. Hence, there is no statistical error in the results, and it is therefore not necessary to run the simulation several times to increase the statistical validity of the output.

6.1.3 The metrics

The goal of this simulation is to measure the difference in traffic load with basic and enhanced bridging. Two different metrics are chosen to illustrate this.

Figure 6.3 shows the total number of *floods* on any of the four rings, in intervals of 10 μs. A flood is performed either by the source node, or by a bridge forwarding the frame onto the next ring. That is, the figure illustrates how many times *any* of the nodes, bridge or end node alike, floods a frame on *any* of the rings during the given time interval. The relatively short time interval of 10 μs gives a curve that is not very smooth, and even small variations are visible. This may not be ideal - a stable situation can be made to appear more unstable than it really is. However, the short time intervals are needed to show how the traffic varies in the short period of time while the network is still in a learning state, before the SRCS tables in the bridges and end nodes are built complete.

The other performance metric chosen, illustrated in figure 6.4, says something about the total throughput in the network. The graph shows the number of frames that traverse a link somewhere in the network. A count is made every time any node on any of the four rings receives a frame from its upstream neighbour. The goal is to give a measure of the overall throughput in the network, caused by the single traffic flow. The time interval used is the same as for the flooding plot, 10 μs.

6.1.4 Analysis of the results

Consider the graph in figure 6.3, showing the number of floods in the network. Following the time line from the left to the right, we can see how enhanced bridging reduces the number of floods in the network, after an initial period

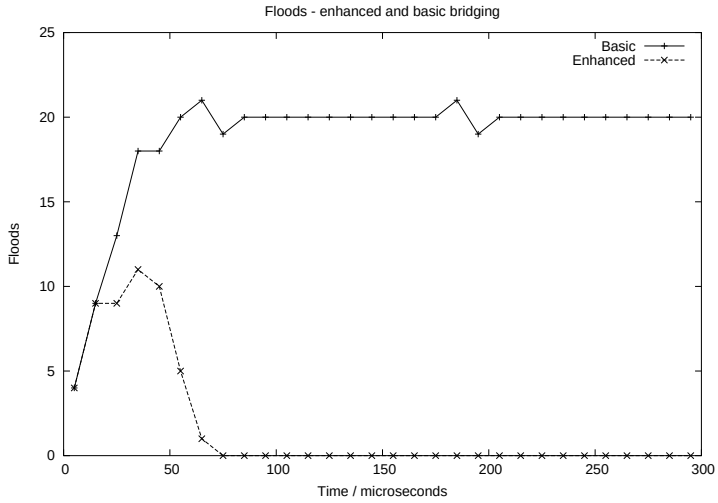


Figure 6.3: The graph shows how the number of floods per 10 μs interval for the basic and enhanced bridging algorithms.

of learning. At the beginning of the simulation, all the SRCS tables in the bridges and the enhanced nodes are empty.

Node n1 starts transmitting frames to node n2 at time zero. These frames are flooded on ring B, forwarded by bridge b1, and flooded on ring A. This is the same for basic and enhanced bridging, since no learning has yet taken place in the end nodes. When the frames reach b3, they are flooded on ring D, since b3 has no information on the position of n2. During this initial period, we see the number of floods rise sharply, as the pipe fills up with more and more frames from n1. At time 20, the first frame reaches bridge b2. From here, we can see a difference between the two bridging strategies. Basic bridging will cause b2 to flood the frames on ring C, for the reasons discussed in section 4.2. Enhanced bridging allows b2 to send the frames directly to n2, giving a lower flooding count.

Focus first on the graph for the enhanced bridging algorithm. At time 30, the first frame from n1 reaches n2, and n2 starts returning acks. The enhanced bridging algorithm allows these frames to be addressed directly to bridge b2, so no increase in the flooding count is experienced. The acks from n2 are never flooded on any of the rings with enhanced bridging, since the location of n1 is now known all along the path. At time 50, the first ack reaches b1. b1 then learns that frames for n2 should be addressed to b2, and can stop flooding these frames on ring A. This gives a sharp fall in the flooding count for enhanced bridging. When the flooding of frames from

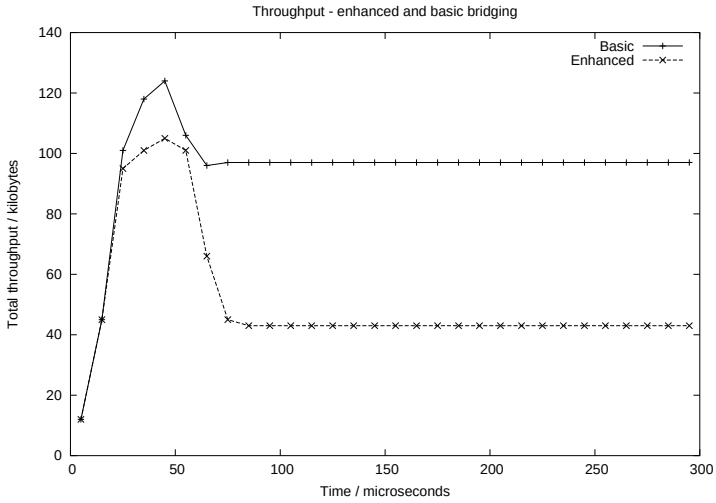


Figure 6.4: The graph shows how the number of bytes traversing a link in the network per 10 μs interval.

n1 on ring A stops, these frames will no longer reach b3. Consequently, no more frames are flooded on ring D. At time 60, the first ack from n2 reaches n1. This completes the learning process for enhanced bridging, and no more flooding occurs.

Turn now to the graph for the basic algorithm. When the first frame from n1 reaches n2 at time 30, the returned acks must be flooded on ring C, A and B. They are never flooded on ring D, because the filtering database in the relay part of bridge b3 has previously learnt the location of n1. Once the acks reach b3 at time 45, it also learns the location of n2, and stops forwarding frames from n1. Note that this happens earlier with basic than with enhanced bridging. Since enhanced bridging does not flood the acks on ring A, they will never reach b3. Bridge b3 will therefore continue to forward frames from n1 to n2 as long as these are flooded on ring A. As seen in the graph, the flooding count does not decrease with basic bridging when the flooding on ring D stops. This is because the acks from n2 to n1 start being flooded on ring B at time 50. These two events - the end of flooding on ring D and the start of flooding on ring B - makes the total flooding count stay relatively stable. At time 60 the first ack from node n2 reaches n1. The n1-n2-n1 pipe is now filled with frames, and no more learning occurs in the bridge relays. The flooding count stays the same for basic bridging until the simulation stops.

The flooding count makes no distinction between 1000 byte “data” frames

and 80 byte "ack" frames. It does therefore not give a true picture of the load on the network. This is better illustrated in figure 6.4. This figure shows the total throughput in the network. The figure indicates that there is a close relation between the throughput and the number of floods in the network. The difference between the two bridging algorithms becomes apparent when the traffic reaches ring C at time 30. The increase in throughput with basic bridging ends when b3 stops forwarding frames for n2 at time 45. The fall in the throughput as no more frames are flooded on ring D, is not equalised by the acks flooded on ring B, since these acks are so much smaller.

For the enhanced algorithm, the throughput takes a dive when the first ack reach bridge b1 at time 50, and data frames stop being flooded on rings A and D.

The number of floods stabilises for both bridging algorithms at time 60. Hence, the load also stabilises once the flooded frames in transit reach their final destinations.

6.1.5 Discussion

The graphs in figures 6.3 and 6.4 show that the potential gain in bandwidth efficiency for remote traffic is significant with enhanced bridging. In this scenario, enhanced bridging gives a 56% reduction in the total throughput in the network, after the initial learning period.

A period of learning is needed before the enhanced bridging algorithm achieves spatial reuse for remote traffic. This learning period ends after one round-trip-time in the communication between nodes n1 and n2. When a frame from n1 has reached n2, and the first reply message has returned to n1, all the necessary learning in bridges as well as in end nodes is finished.

The active nodes and relaying bridges in this scenario are placed at the opposite ends of the respective rings. This means that a frame that travels the shortest path from n1 to n2 still has to traverse half of ring B, then half of ring A, and finally half of ring C. This is a worst-case scenario for enhanced bridging, giving a total node hop count of twelve for each frame. If the active nodes were placed otherwise on the rings, the difference in throughput between basic and enhanced bridging would further increase. Thus, the potential efficiency gain of a single flow through a bridged network is even larger than what appears in figure 6.4. When traffic can be sent directly to the bridges without flooding, no frame will normally[1] traverse more than half the circumference on any ring. The overall bandwidth consumed by a

[1]The RPR draft standard allows the MAC client to choose the "longest" path around a ring when transmitting on the ring. This is, however, not normally the case.

bridged traffic flow is thus always more than halved by the enhanced bridging algorithm.

6.2 Scenario 2: Filling the SRCS tables

The previous scenario illustrated the advantage of the enhanced bridging algorithm - the reduced amount of flooding that gives a better utilisation of the available bandwidth. The price to pay for this advantage is first of all the need for more RAM in the end nodes. To avoid flooding and allow spatial reuse, every end node must keep a mapping between remote node addresses and a local bridge address. In the simulator model discussed in section 5, these tables reside in the Spatial Reuse Control Sublayer (SRCS) in the nodes that support enhanced bridging. The tables containing the remote-to-local node addresses are thus referred to as SRCS tables.

The SRCS table contains one record for each remote host the node has received traffic from. In other words, the size of this table potentially grows linearly with the total number of nodes in the bridged network, if the node receives traffic from all other nodes. The number of nodes in a bridged network can grow to several thousand. These tables must be stored in each node and will demand a certain amount of available fast memory.

The goal of this simulation is to investigate the traffic behaviour in a network where the SRCS tables in the end nodes are of limited size. The idea is to let the SRCS tables work as a kind of a cache. A least recently used (LRU) algorithm can be used to select the oldest entry in the table for removal when the table is full. If the entries are organised in a circular list, this LRU algorithm can be implemented by providing each entry with a recently used bit. This bit is set to one each time the node receives a frame from the node corresponding to this entry, or when a frame is sent to this node. When an entry must be removed from the table, a pointer is moved around the circular list, setting the recently used bit to zero as it passes. The first entry whose recently used bit is already zero, is discarded. When a remote receiver is not found in the sender's SRCS table, the frame must be flooded. This flooding must continue until a reply is received from the node in question, and a new entry can be made in the SRCS table.

6.2.1 Topology

In the previous scenario, the topology consisted of four rings with eight nodes in each ring. With such a topology, no node has to keep more than 24 entries in its SRCS table - one entry for each remote node. The topology

in this scenario contains a large number of nodes, spread over eight rings as shown in figure 6.5. Rings A through F are thought of as large access rings, with 200 end nodes connected to each of them. The link lengths in these rings are approximately 200 metres, giving a total ring length of about 40 kilometres. Rings M1 and M2 connect the access rings. They contain no active end nodes themselves, only bridges whose only purpose is forwarding frames between the access rings. The link lengths of these rings are greater, about 20 kilometres, giving a total ring span of about 80 kilometres. The link capacity on the rings is the same as in the previous scenario, 8 Gbps.

With this topology, each end node on the access rings needs an SRCS table size of 1000 to save an entry for each remote frames.

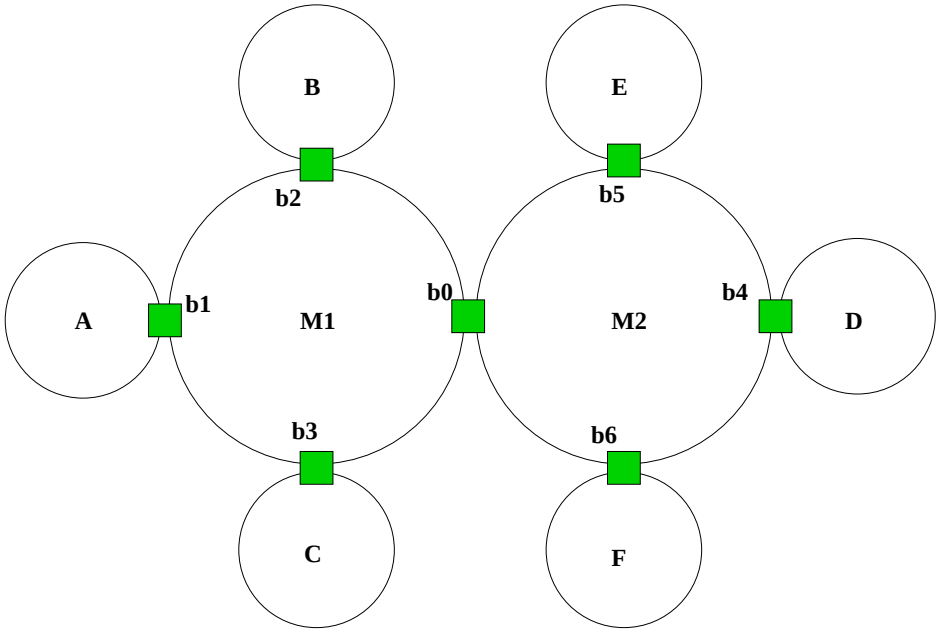


Figure 6.5: Six smaller rings A through F are attached to two main rings M1 and M2. Each access ring contains 200 nodes.

6.2.2 Traffic pattern

The purpose of this simulation is to analyse the behaviour of the enhanced bridging algorithm with limited SRCS table sizes in the end nodes. The hypothesis is that the size of these tables becomes a performance limiting factor when a node sends and receives traffic from a large number of other nodes. This scenario thus contains a large number of active nodes. Every

node at the access rings sends frames to any other node in a random fashion. It does this by first randomly drawing a node to send to, and then send between one and three frames to this node. It then draws a new receiver. The random receiver sending pattern is a worst case scenario with respect to the SRCS tables. If a node communicates with all other nodes, it needs an entry for all other nodes in this table.

In this scenario, all nodes send 500 byte low priority frames. The time interval between each transmission is a random number, uniformly distributed between 0 and 300 μs. On average, each node transmits a 500 byte frame every 150 μs, giving an effective send rate of approximately 3.3 Mbyte/s for each node.

Every end node in the network must keep and maintain an SRCS table. A bridge acts like a node in all the networks it interconnects. It must therefore keep an SRCS table for each of its ports. In addition, the relay part of the bridge keeps its usual filtering database with host-to-port mappings. Figure 6.6 shows the placement of the different tables kept in bridge b0 in this scenario.

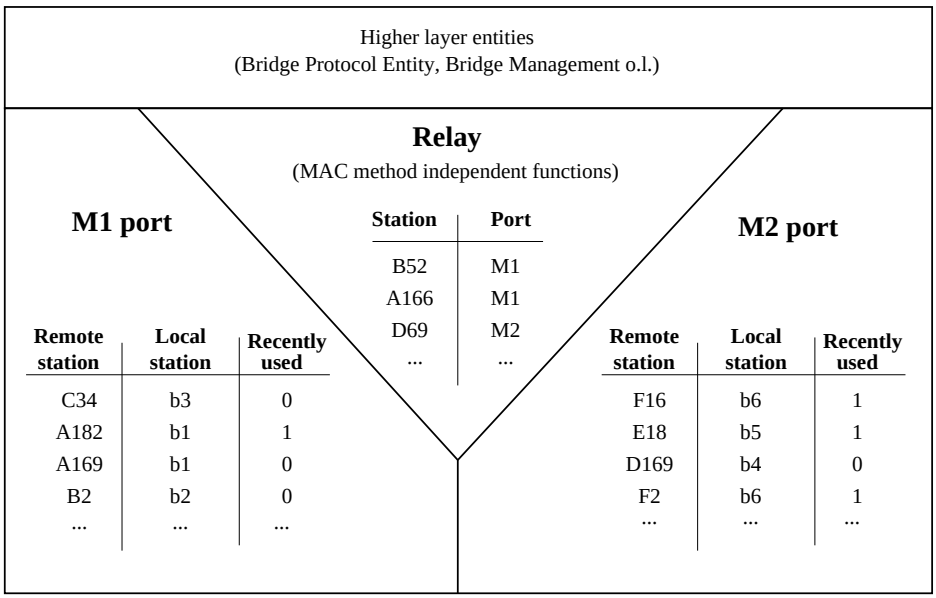


Figure 6.6: Bridge b0 maintains a separate SRCS table for M1 and M2. Note that all entries in the M1 table must come from rings A-C, while all entries in the M2 table must come from rings D-F. In addition, the relay part of the bridge contains the normal filtering database.

Normally, the SRCS tables of all nodes would initially be empty. As seen

in the previous simulation, a number of floods is necessary with the enhanced bridging algorithm even without limited SRCS table sizes. This flooding occurs before the learning processes in the end nodes have converged, and the SRCS tables have been built. With 1200 active nodes, as in this scenario, the time before this stable state is reached becomes very long. The purpose of this simulation is not to see how the number of floods is reduced as the SRCS tables are built in the end nodes. This was illustrated in the previous scenario. Instead, this simulation seeks to investigate at which level the number of floods stabilises, depending on the size of the SRCS tables. Hence, the traffic characteristics in the initial learning period is not of interest. In this simulation, the end nodes are therefore equipped with non-empty SRCS tables at the start of the simulation time. The tables are filled up with entries for a randomly chosen set of remote nodes. Thus, the extra flooding needed in the initial learning period is eliminated in this scenario.

As emphasised in section 4.3, there are two learning processes taking place in an enhanced bridging network. In addition to the learning in all the end nodes, there is also the normal learning process in the bridge relays. For the same reasons as stated above, the filtering database in the bridge relays are also pre-filled in this scenario. The bridge relays have a complete picture of the network at simulation startup, and will thus never perform any unnecessary frame forwarding.

6.2.3 Collected data

Two key metrics are chosen to illustrate how the network performance is affected by the SRCS table sizes. First, the flooding count is used in the same way as in scenario 1. The main objective of the enhanced bridging algorithm is to eliminate unnecessary flooding in the network, and the flooding count therefore becomes a good measure for how well this algorithm works with reduced SRCS table sizes. The flooded frames gives an increase in the traffic load in the network. To show the significance of this increase, the second metric used in this scenario is the traffic throughput in two selected nodes. The flooding count and the measured throughput will be discussed separately below, before an overall discussion is provided at the end of this section.

6.2.4 Flooding

Figure 6.7 shows how the number of floods varies over time in a network with nodes that can keep 350 or 700 entries in their SRCS tables. As in scenario 1, the "number of floods" means that every time a frame is flooded over any one of the rings, a count is made. The time period over which the number of

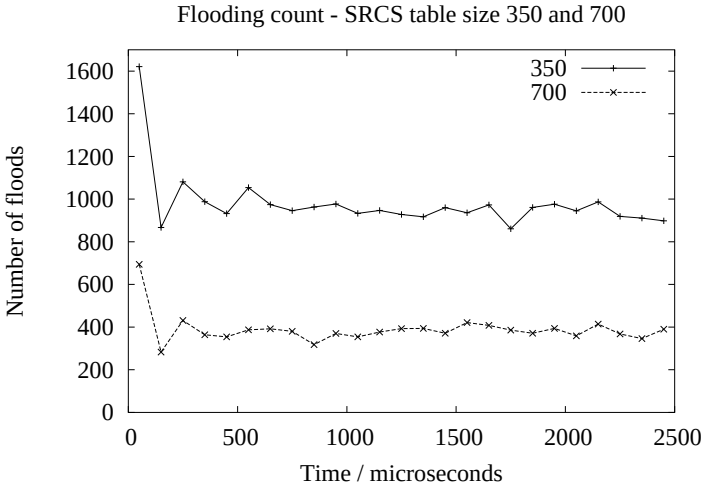


Figure 6.7: After the initial learning period, the amount of flooding is relatively stable over time. The graph shows the flooding count over time when the SRCS tables can keep 350 and 700 entries.

floods is counted, is 100000 simulator time units, equalling 100 μs. As seen from the graph, the number of floods is relatively stable over time. This is as expected. No initial learning period is needed, and the probability that a remote destination node is not in the sending node's SRCS table is constant.

The peak in the flooding count at the very beginning of the time period, is due to the sending pattern of the active end nodes. When the simulation starts, every node transmits its first frame almost immediately. After that, the time before the next frame is transmitted is randomly drawn as explained above. It thus takes some time before the transmit times of all the nodes are uniformly spread over the time line.

In figure 6.8, the average number of floods per 100 μs time period is shown for different SRCS table sizes, ranging from 0 to 1200. This average is calculated for each table size as the arithmetic mean of the flooding counts measured for that particular table size. For each table size, the number of floods is measured the same way as for the examples shown in figure 6.7. The average value in this series of measures is then calculated, and results in one value in figure 6.8. To avoid the initial flooding peak seen in figure 6.7, the first three values are omitted in this calculation.

The two graphs in figure 6.8 illustrates two different possibilities in the scenario. In the upper graph (6.8a), the SRCS tables in the bridges are variable, just as in the normal end nodes. In the lower graph (6.8b), these

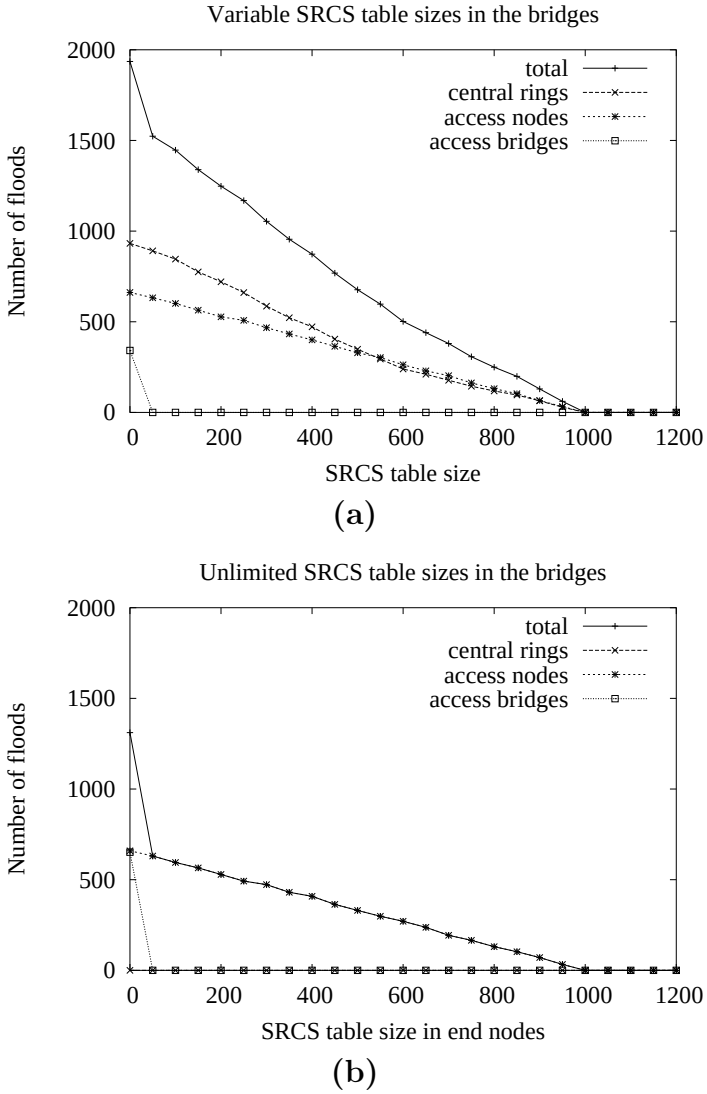


Figure 6.8: The plots show how the number of floods in the network is reduced with growing SRCS table size. The right plot shows that when the size of the SRCS tables in the bridges is unlimited, no flooding takes place on rings M1 and M2

tables are of "unlimited" size, meaning large enough to keep entries for all the remote nodes in the network. The idea here is that the bridges are more important than the other nodes in this context, because they flood frames sourced by many other nodes. The SRCS table sizes in the bridges will thus be more important to the overall network performance.

Figure 6.8 shows the total number of floods in the network, but also

the flooding count for different kinds of nodes. The flooding count is given separately for the end nodes on the access rings (access nodes), the bridge interfaces on the access rings (access bridges), and the bridge interfaces on rings M1 and M2 (central rings).

Figure 6.8a, shows the number of floods in the network when the SRCS table size in the bridges vary in the same way as in the end nodes. The total number of floods in the network is the sum of the floods from the end nodes in rings A to F, and the frames flooded by the bridges on rings M1 and M2. The bridges only flood frames onto the access rings when the SRCS table size is zero, as will be explained below. The flooding count decreases steadily with increasing SRCS table size. When the table size reaches 1000, no more flooding will occur in the network, as all remote nodes are kept in the SRCS tables. Note that the graph for the number of floods made on rings M1 and M2 (central rings) makes a break when the SRCS table size reaches 600. This is due to the learning process in bridge b0. b0 receives traffic from 600 nodes on each of its ports, and the learning process thus completes when the SRCS table can keep 600 entries.

In figure 6.8b, the bridges are equipped with SRCS tables that are large enough to keep an entry for all the nodes in the network. The plot shows that the bridges no longer flood frames on rings M1 and M2. Only the end nodes on rings A to F now flood frames. Hence, the "total" graph in figure 6.8b is the same as the "access nodes" graph. The exception again is SRCS table size 0, when traffic must be flooded on the access rings.

The flooding count is noticeably higher for a SRCS table size of 0 than for one of 50. This is the case in both of the plots in figure 6.8, and comes from an increased flooding from the bridge nodes. To find an explanation for this, recall from the discussion in section 4.2 that bridges using the basic bridging strategy always must flood frames on all rings to ensure correct learning in the bridge relays. This is also true for "enhanced" bridging with SRCS table size 0. Enhanced bridge nodes normally send the frames directly to the local receiver on the egress ring, without any flooding. But since the receiver node cannot store the address of the local sender (bridge) in the SRCS table, it will have to flood the reply. This would lead to the same bridge learning problems as described for basic bridging in section 4.2. A SRCS table size of zero thus gives the same flooding and throughput characteristics as would basic bridging. Once the SRCS tables in the end nodes get a size greater than zero, this problem does no longer occur, and bridges may send the frames directly to the local receiver. Even if the size of the SRCS table is small, the node will be able to reply without flooding to the nodes that it most recently received traffic from. There can exist traffic scenarios where this strategy can lead to additional flooding, and thus give a poorer utilisation of the network

resources. This is, however, not the case with the topology described in this scenario.

We note that the flooding count for the access bridges equals the count for the end nodes for SRCS table size 0 in figure 6.8b. This is as expected; a remote frame that is flooded by the sender on the ingress ring, must also be flooded by the relaying bridge on the egress ring. Figure 6.8a, however, shows a lower flooding count for the access bridges than for the end nodes. The huge amount of flooding on rings M1 and M2 leads to buffer overflow in bridge b0, and frames are discarded. If no frames were discarded this value would, as in the plot to the right, be equal to the end node flooding count.

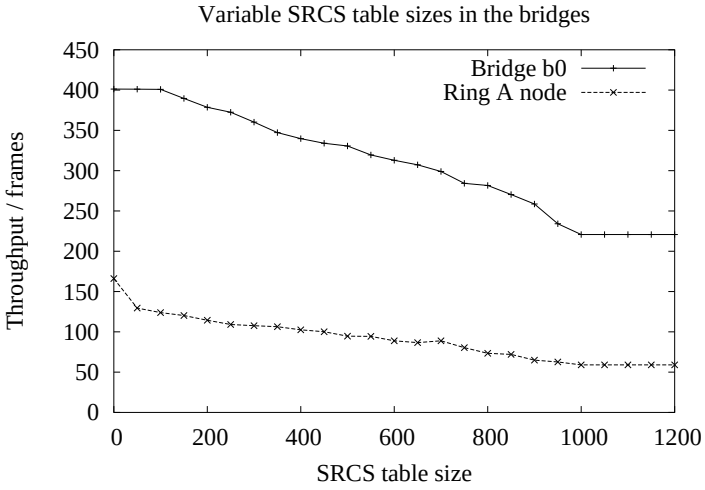
6.2.5 Throughput

As seen in scenario 1, there is a close relation between the amount of flooding in the network and the traffic load. Flooding causes more frames to be sent over each link in the network on average. Figure 6.9 shows the throughput for two selected nodes. The throughput in a node means the combined throughput of the two incoming links in this node. As with the flooding above, figure 6.9a shows the case with variable SRCS table sizes in the bridges, while figure 6.9b shows the case where the bridges have "unlimited" SRCS table sizes.

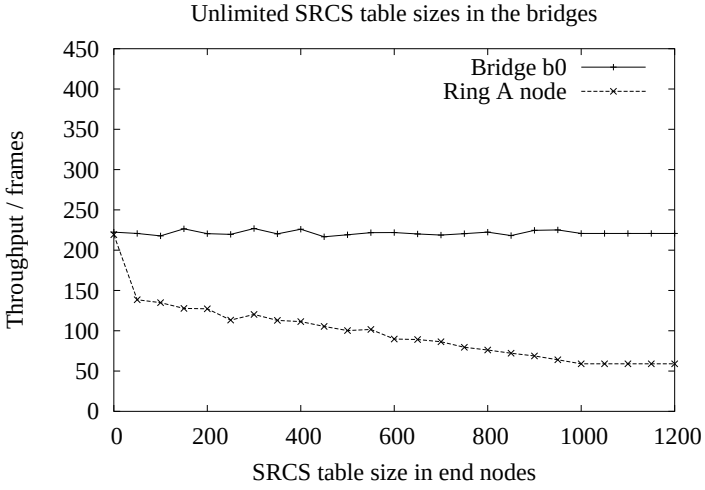
The first measured node resides on ring A, the second node is the ring M1 interface of bridge b0. The node on ring A is situated 50 hops from the bridge. This way, remote frames from a quarter of the nodes on the ring will pass through it in the case of enhanced bridging with "unlimited" SRCS table sizes. The node thus experiences a traffic throughput that is the average of all the nodes on the access rings.

Looking first at the throughput in the ring A node, we see that this is the same in the two plots. The intuition behind this is that only end nodes flood frames on the access rings when the SRCS table size is greater than zero, as discussed above. The throughput on the access rings are thus not affected by the size of the SRCS table size in the bridges. The exception again, is for table size zero. In this situation, frames must be flooded onto the access rings by the relaying bridges, and causes increased traffic. Just like for the flooding count, the throughput on the access rings in the zero-table case is not the same in the two plots. As discussed above, the huge amount of traffic when the bridges are forced to flood frames on rings M1 and M2 causes buffer overflow in bridge b0. This leads to packet loss, and reduces the number of frames flooded onto the access rings.

Turning now to the throughput in bridge b0, we see that this value is far from the same in the two plots. When the bridges have unlimited SRCS table sizes, the throughput in b0 is stable, independent of the SRCS table



(a)



(b)

Figure 6.9: The plots show that the throughput on the access rings decreases with increasing SRCS table size. This decrease is the same whether the SRCS table size in the bridges are limited or not. With infinite SRCS table sizes in the bridges, no flooding occurs on rings M1 and M2, so the throughput in b0 is stable.

sizes in the end nodes. No flooding takes place on rings M1 and M2, since only bridges with unlimited SRCS table sizes are attached to these rings. This is not the case in the graph to the left, where the SRCS table size in the bridges vary along with the tables in the end nodes. In this case, the throughput in bridge b0 decreases with increasing SRCS table size, as less

flooding is needed on ring M1. When the SRCS table size reaches 1000, all nodes that the bridges on rings M1 and M2 receive traffic from are kept in the tables. No more flooding is needed on these rings, and further increasing the SRCS table size does not reduce the amount of traffic. Note that there is no reduction in the throughput in b0 when the SCRS table size is increased from 0 to 100. This is again caused by overflow in one of the bridges. 400 frames of 500 bytes passing through node b0 each 100 μs interval, means that the throughput in each of the ringlets is 1Gbyte/s. This is the link speed in this scenario, and no further increase in throughput is possible.

6.2.6 Discussion

This scenario has illustrated how the traffic in a bridged RPR network is affected by the size of the SRCS table size in the bridges and end nodes. We have seen that reducing the size of these tables leads to more flooding, and thus an increased traffic load in the network. A few further observations are worth noting.

The situation with zero-sized SRCS tables in both the bridges and the end nodes, means that we have basic bridging on all rings. In this case, of course, all remote traffic is flooded on all rings. In this situation, the traffic load becomes too heavy in this scenario, and forces frames to be discarded in bridge b0.

The situation with infinite SRCS table sizes in the bridges and zero-sized tables in the end nodes, corresponds to having basic bridging in the access rings, and enhanced bridging on rings M1 and M2. This scenario thus illustrates how these two bridging strategies can co-exist in the same network. In this case, the traffic throughput is reduced by 45% in bridge b0, compared to the all-basic situation.

Finally, unlimited SRCS table sizes in both the bridges and the end nodes, means that we have "pure" enhanced bridging on all the rings. In this scenario, this gives a reduction in the throughput of 45% in bridge b0, and 73% in the measured node on ring A compared to the all-basic situation. Note that the reduction in throughput is the same in bridge b0, regardless of which bridging algorithm is used in the access rings.

The motivation for reducing the SRCS table sizes in the first place, was to reduce the amount of extra memory needed in the nodes in an RPR network. The size of the SRCS tables increases linearly with the number of nodes in a bridged network. The amount of data stored in each entry does not have to be large. Enough data must be stored to keep the MAC address of the remote node, and identify the bridge that is the local destination for traffic for that node. Some control information associated with each entry might

also be wanted, in addition to the mentioned recently used bit which allows old values to be discarded when the SRCS table is full. For instance, a time stamp on each entry could be needed, so that erroneous entries will eventually be removed in case of a topology change. With six byte MAC addresses used to identify both local and remote nodes, and eight bytes reserved for control information, each entry in the SRCS tables would demand 20 bytes of memory. Bridged networks can be of considerable size, but they encounter some scaling problems when they become very large, as mentioned in section 3.4. For all practical purposes, a bridged network consisting of more than 10000 nodes would seem like a bad idea - the practical limit is probably well below this number. In a network consisting of 10000 nodes, using 20 byte entries in the SRCS tables, each node would need 200 kilobytes of memory to keep these tables.

The simulated scenario confirms that reducing the SRCS table sizes to save memory makes a bridged network less effective with respect to bandwidth utilisation. The graphs in figure 6.9 suggest that there is a linear connection between the SRCS table size and the traffic load produced by a node on the ring.

More stable traffic sessions between pairs of nodes could probably make better use of the cache-like SRCS tables than in this scenario. Running this simulation with a more realistic traffic pattern would probably give a better "cache hit rate", and thus better bandwidth utilisation. Still, there is no doubt that the network performance would be affected by reduced SRCS table sizes. The relatively small amount of memory needed would normally seem like a price well worth paying for avoiding this efficiency reduction.

6.3 Scenario 3: Bridging impact on service level

The first and second scenarios both described situations that are probably far from a real-life traffic scenario. This last scenario moves more in the direction of a realistic setting. An effort is made to make the traffic on the ring look like the traffic found in a typical network today.

The goal of this scenario is to show how the choice of bridging strategy affects the local traffic on an RPR ring. The simulations performed indicate that the reduced amount of traffic produced with the enhanced bridging algorithm gives lower latency for the best effort traffic on the ring.

6.3.1 Self similar traffic

Studies of traffic traces in an Ethernet environment show that this traffic is statistically *self-similar* [21]. The same is true for wide-area/Internet traffic [25]. The intuitive characteristic of this traffic is that there is no natural length of a "burst". Independent of the time scale of the measures, similar-looking traffic bursts in the traffic traces are observed. Both local and remote (bridged) Ethernet traffic shows self-similar characteristics, as well as external TCP traffic. Mathematically, the degree of self-similarity in a traffic trace is expressed by the *Hurst* parameter. The Hurst parameter is a value between 0.5 and 1.0, and the degree of self-similarity increases as the Hurst parameter approaches 1.0. Through analysing a huge amount of Ethernet traffic, Leland et al. [21] estimated the Hurst parameter for such traffic to be about 0.9.

The self-similar nature of network traffic has consequences for the modelling of networks [25]. Traditional formal models used in simulations are often based on Poisson probability distributions, which produce traffic that is very different from the self-similar traffic found in actual network traces. A way to model traffic with a more self-similar nature is discussed below.

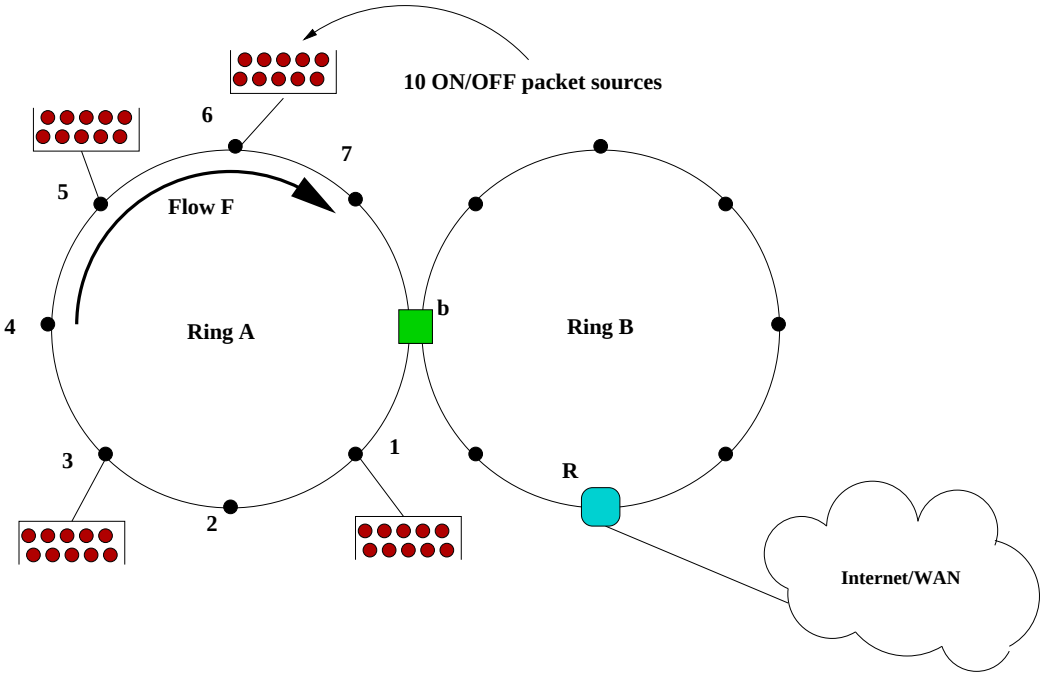


Figure 6.10: The active nodes in this scenario are connected to the outside world through a router placed on the other side of a bridge.

6.3.2 Topology

The network topology used in this scenario is shown in figure 6.10. The active nodes in this configuration are placed on ring A. These are connected to the outside world through node R, which is placed on ring B. The two rings are connected by a bridge, named b. This way, all traffic between the nodes on ring A and the Internet must pass through bridge b.

Each of the rings consist of eight nodes. These are linked with cables of about 1000 metres, giving a total ring span of about 8 km. The links have a capacity of 8 Gbps, like in both the previous scenarios.

6.3.3 Traffic pattern

The main objective of this scenario is to illustrate how the choice of bridging strategy influences the local traffic on an RPR ring. This is done by letting nodes 1, 3, 5 and 6 on ring A produce self-similar traffic destined for node R on ring B. At the same time, node 4 sends a continuous flow of 500 byte frames at 20% of the link capacity to node 7. This constant stream of frames is named flow F. All the traffic in this scenario is low priority traffic using the best effort class C service class discussed in section 2.3.

Self-similar traffic can be modelled by superpositioning several ON/OFF packet sources, if the length of the ON and OFF periods of these sources have infinite variance [34]. Probability distribution functions with high or infinite variance are often called *heavy-tailed* distributions, due to the shape of their graph. One such heavy-tailed distribution is the *Pareto* distribution, shown in equation (6.1). In this function α is known as the shape parameter and decides the thickness of the "tail", while the location parameter b decides the scale of the x-axis. The expected value of a Pareto-distributed random variable is given in (6.2). Note that an α value smaller than 1.0 gives an undefined (infinite) expected value. A nice feature of the Pareto distribution is that there is a simple relation between the α parameter and the Hurst parameter describing the degree of self-similarity of the generated traffic ($H = (3 - \alpha)/2$)

$$P(x) = \frac{\alpha b^{\alpha}}{x^{\alpha+1}} \qquad x \geq b \tag{6.1}$$

$$E(X) = \frac{\alpha b}{\alpha - 1} \qquad \alpha > 1 \tag{6.2}$$

Returning to this scenario, each of the active nodes on ring A have ten independent ON/OFF sources. The length of the ON/OFF periods of these sources are Pareto distributed with an α value of 1.2. This choice of α gives

a Hurst parameter of 0.9, which is the value estimated for Ethernet traffic in [21]. The b parameter for the ON periods is set to 40. This gives a minimum frame size of 40 bytes, chosen because it is equivalent to the minimum TCP packet size. The long-tailed nature of the Pareto distribution will sometimes result in ON periods giving frames that are much longer than 1522 bytes, which is the maximum frame size in RPR networks not supporting jumbo frames as discussed in section 2.2. Such frames are split into several 1500 bytes frames.

The b parameter for the OFF period is used to regulate the load in the network. A load of 0.5 means that the forty ON/OFF sources in the four active nodes in total transmit frames half of the time. By using the expected values of the ON and OFF periods, it can be shown that the b parameter corresponding to a desired load L is given by (6.3). Note here that due to the dual ring and spatial reuse properties of RPR, a load of 0.5 does not mean that a link is busy half of the time, unless all nodes send traffic over the same span of the same ringlet at the same time.

$$b_{OFF} = b_{ON}(\frac{1}{L_i} - 1) \qquad \text{where} \qquad L = \sum_{i=1}^{N} L_i \qquad \text{and} \qquad \alpha_{OFF} = \alpha_{ON} \tag{6.3}$$

Figure 6.11 shows the self-similar traffic produced by the ON/OFF senders in nodes 1, 3, 5 and 6. The plots show the number of bytes passing node 7 in both directions per time unit. The measures are made with a load of 0.3, and using the basic bridging algorithm. Figure 6.11 shows how self-similar traffic is bursty on several time scales. In contrast, Poisson distributed traffic would give smoother plots when the time scale was increased.

In this scenario, nodes 1, 3, 5 and 6 send traffic to router R, which acts as a gateway to the outside world. However, no traffic is returned from router R, except for a response to the first frame received from each of the senders to allow the initial learning. This does not give a very realistic traffic scenario - one would expect at least as much traffic returning from the Internet to the end nodes. The reason why this traffic is left out of the simulation, is that this traffic would never affect the frames in flow F. Using bidirectional flooding as discussed in section 4.2, traffic forwarded on ring A, by bridge b, will always use the opposite ringlet on this span of the ring. This traffic could therefore not cause congestion that would have impact on the latency from node 4 to node 7.

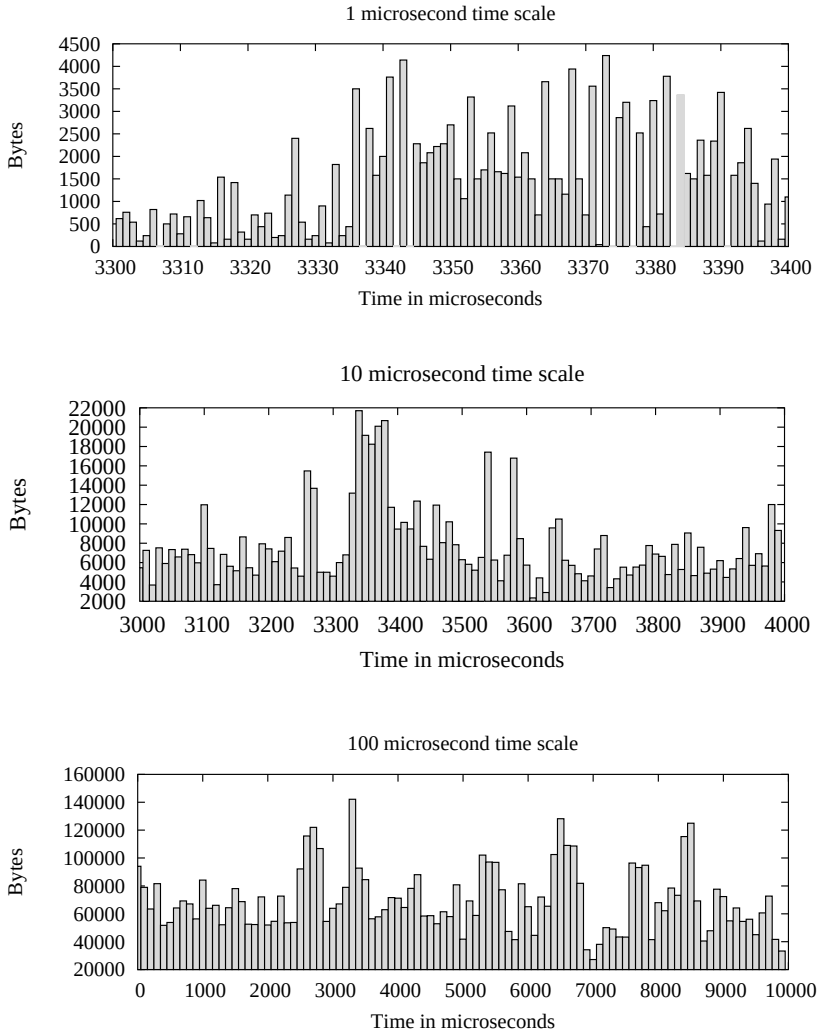


Figure 6.11: These plots show how self-similar traffic produced by nodes 1, 3, 5 and 6 is bursty on different time scales.

6.3.4 Collected data

In this simulation, the latency is measured for the frames in flow F. The goal is to find out how this latency is influenced by the traffic load in the network and the choice of bridging strategy. Latency in this scenario is measured as the time it takes from a frame is at the head of the sender node's MAC client add queue, to the frame is received by the MAC client at the destination. This time will vary depending on how long the frames have to wait for transmission in the ingress buffer in node 4, and the transit buffers in nodes 5 and 6.

Node 4 sends a constant stream of 500 byte frames to node 7 at 20% of the link capacity. This sending goes on for about 4 ms. The latency of each frame is recorded when it arrives at node 7. Figure 6.12 shows how the latency varies for the frames with different traffic loads and bridging strategies. Most of the frames are gathered in the low end of the scale, but the number of frames with a significantly higher latency is not negligible. The arrow in figure 6.12 points out the 90th percentile (also called the 0.9 quantile) for the data set. The 90th percentile denotes the value on the x axis so that 90% of the measures are smaller than this value. This value gives a good indication of the latency properties for the data stream in question, since a few extreme values will not inflict on the result. As the collected data shows, the latency can never drop below 15 μs. This is the aggregate propagation delay from node 4 to 7. Frames that are not held back in any buffers along the way will achieve this minimum latency.

The six plots in figure 6.12 shows that the latency and jitter characteristics of the traffic depends on the traffic load in the network, and the choice of bridging strategy. A higher traffic load results in more periods with congestion and thus higher latency for some of the traffic. In the upper plot in figure 6.13, the 90th percentile for the latency is plotted for different network loads, with both basic and enhanced bridging. Each point on the graphs represents the average value for the 90th percentile found by repeating the simulation 20 times with a different seed in the random generator.

The 20 measured values for the 90th percentile do not seem to follow a normal distribution. Hence, it is not a simple task to calculate a confidence interval for the 90th percentile using the arithmetic mean of the 20 values as an estimator. However, using instead the median as an estimator, a confidence interval can easily be found. The lower plot in figure 6.13 shows the 90th percentile estimated this way, with a 95% confidence interval. The size of the confidence intervals show that the variance in the measured 90th percentile is high for some of the loads.

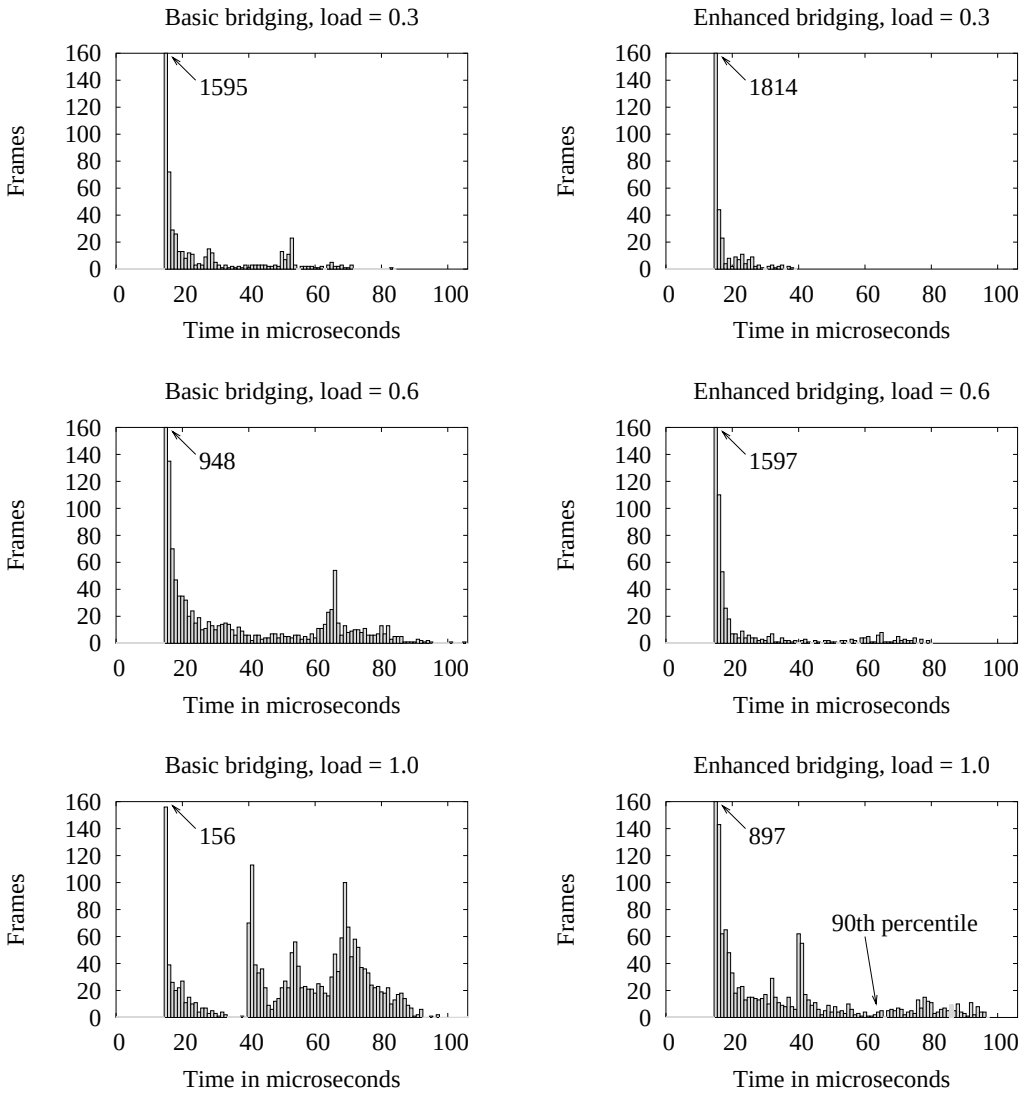


Figure 6.12: The fraction of frames that experience a low latency drops with increasing traffic load. Each bar in these plots represent a 100 ns time interval, and a total of 1960 frames was measured.

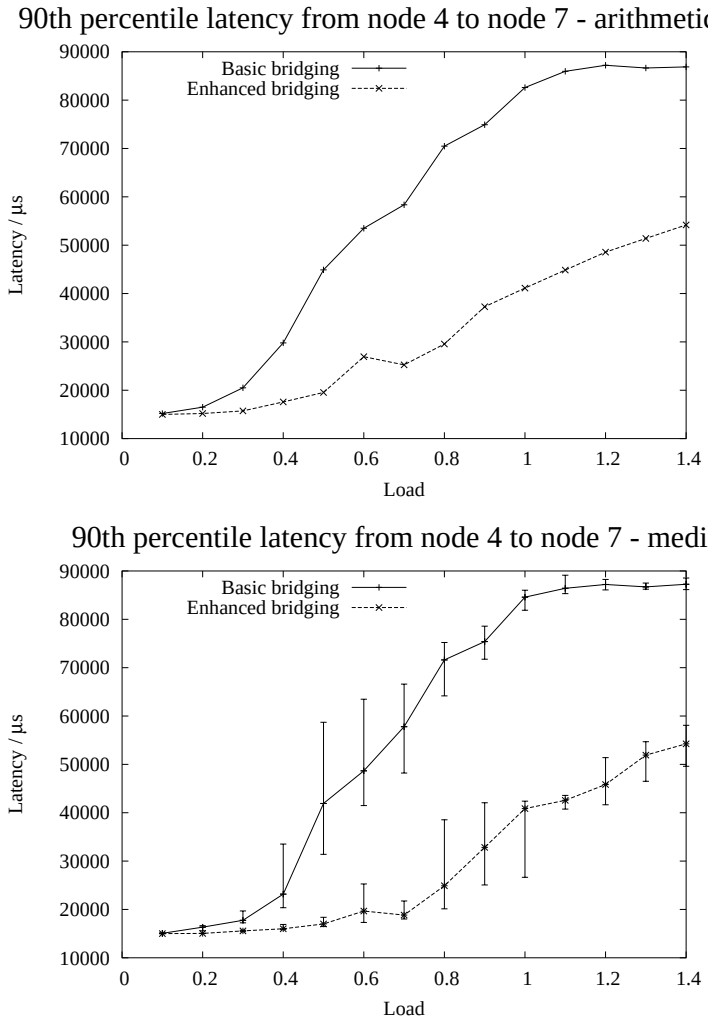


Figure 6.13: The latency from node 4 to node 7 increases when the load on the network grows. The enhanced bridging algorithm gives noticeably lower latency than the basic algorithm. The upper plot shows the 90th percentile as the arithmetic mean of the 20 simulated values. In the lower plot, the median is used as an estimator for the 90th percentile. This allows the calculation of a 95% confidence interval.

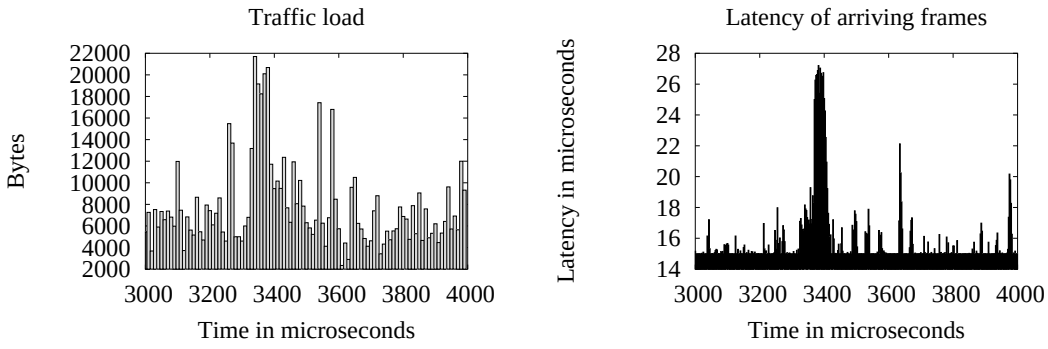


Figure 6.14: There seems to be a clear connection between the number of bytes arriving at node 7, and the latency of the frames it receives from node 4. These measures are made over the same time period, with a load of 0.4 using basic bridging.

6.3.5 Analysis and discussion

Not surprisingly, the results presented in figure 6.13 suggests that there is a clear connection between the traffic load in an RPR network and the latency characteristics. This connection is perhaps even more visible in figure 6.14. Here, the latency of the arriving frames are plotted on the same time scale as the traffic load as seen from node 7. This figure shows that the latency in flow F increases in periods when the traffic load is high.

With increasing traffic load in the network, the difference between the basic and the enhanced bridging algorithm becomes significant. As shown in scenario 1, enhanced bridging reduces the amount of traffic on the ring. In this scenario, flow F goes on the outer ringlet. It has to compete for the bandwidth on this particular ringlet span with different other traffic flows. With enhanced bridging, only frames from nodes 5 and 6 will utilise the same bandwidth. Traffic from nodes 1 and 3 will use the inner ringlet to reach bridge b. In the basic case, all remote traffic from nodes 1, 3, 5 and 6 will be bidirectionally flooded on the ring. This means that frames from all four senders will pass over some or all of the outer ringlet between nodes 4 and 7. This explains why basic bridging gives a higher traffic load on these links, and thus higher latency.

Note here that the placement of the active senders will influence the simulation results. This is true for both the active node in flow F, and the four other traffic-generating nodes. The fact that frames in flow F always move *towards* the bridge, allows us to ignore traffic from the outside world forwarded onto ring A by bridge b. As explained above, this traffic will

normally use the opposite ringlet between nodes 4 and 7. Moving the active sender in flow F to another location on the ring, could change this situation. Also, moving the other active nodes would change the amount of traffic competing with flow F for the bandwidth from node 4 to 7.

The simulations here described, show that there is a connection between the choice of bridging mode and the latency properties of best effort traffic on the ring. Enhanced bridging allows more traffic to be sent over the ring before the increase in latency and jitter becomes significant. With the traffic pattern used in this scenario, figure 6.13 shows that encanced bridging allows roughly twice as much remote traffic to be sent through the network before the 90th percentile latency for flow F becomes the same as with basic bridging.

Thus, enhanced bridging can implicitly give better quality of service properties to the best effort traffic class. As discussed in section 2.3, RPR offers the class A and class B services to traffic with special quality of service requirements. The time-critical class A traffic may use reserved bandwidth, and its own high-priority transit queue in the nodes. This traffic is therefore less likely to experience any difference in latency depending on the choice of bridging mechanism.

6.4 Summary

This section has described three simulation scenarios where bridges are used to interconnect RPR rings.

The first scenario focus on the learning process that takes place in the bridges and end nodes with the enhanced bridging algorithm. Using a simple exchange between two nodes in a bridged network, it is shown how an initial period of flooding is needed before the learning process converges.

The second scenario focus on the effects of reducing the SRCS table sizes in the enhanced end nodes. This simulation shows that reducing the SRCS table size in the end nodes reduces the efficiency gain provided by enhanced bridging.

Finally, the last scenario focus on how the choice of bridging algorithm affects the local traffic on the ring. It is shown that the reduction in traffic load achieved by enhanced bridging gives better jitter and latency characteristics for the local traffic on the ring.

Chapter 7

Conclusions and further work

The topic of this work was bridging in RPR networks. Two different bridging algorithms for RPR was discussed in chapter 4. The basic bridging algorithm is used in the current RPR draft standard, and relies on flooding to make sure that remote frames are seen by the bridges on a ring. The enhanced bridging algorithm maintains SRCS tables in the bridges and end nodes, so that remote frames can be addressed directly to the bridges. This is done to reduce the amount of flooding in the network, and thus give better bandwidth utilisation.

The Java simulation model described in chapter 5 was used to compare and evaluate the performance of the two bridging strategies. Three traffic scenarios was constructed and simulated. Chapter 6 described each of these three scenarios, and discussed the results.

7.1 Results

The three scenarios in chapter 6 focused on different aspects of the enhanced bridging algorithm.

Scenario 1 confirmed that enhanced bridging achieves spatial reuse for remote traffic, after an initial period of learning. When the learning process converges, no more flooding is seen in the communication between a pair of nodes. The learning process in a traffic session between a pair of nodes A and B is complete once a frame from node A has reached node B, and a reply has returned to node A. By eliminating the need for flooding, we have shown that enhanced bridging at least halves the traffic load produced by a bridged traffic flow.

The enhanced bridging algorithm relies on SRCS tables in the end nodes and bridges. Scenario 2 demonstrated that these tables can be used as a

cache, where only the newest entries are kept when the table is full. The simulations showed that reducing the SRCS table sizes gives a higher amount of flooding in the network and thus reduces the advantages of enhanced bridging. With the uniform sending pattern used in this scenario, the simulations indicated a linear connection between the SRCS table size and the traffic load produced by a node on the ring. This sending pattern is a worst case scenario for enhanced bridging with reduced SRCS table sizes. A more realistic traffic pattern would probably give better results for such cache style tables. This scenario also illustrated that basic and enhanced bridging can be used on adjacent rings, without reducing the efficiency gain on the enhanced bridging ring.

The simulations performed in scenario 3 suggested that the choice of bridging algorithm also affects the local traffic on the ring. The reduced traffic load in the network when the enhanced bridging algorithm was used, had a positive effect on the latency and jitter characteristics of the traffic on the ring. Using simulations of self-similar remote traffic, we have shown that with enhanced bridging, a higher traffic load can be put on the network before the the latency and jitter of the local traffic increases substantially. The traffic pattern used in scenario 3 allowed about twice as much traffic to be transmitted on the network before the basic bridging latency levels were reached.

7.2 Problem statement revisited

The goal stated in the introduction of this work was to discuss an enhanced bridging algorithm for RPR, and to evaluate the performance of this algorithm. This has at least partially been achieved.

The simulation results presented in chapter 6 clearly show that the enhanced bridging algorithm has performance advantages over basic bridging. The reduced amount of flooding gives a lower traffic load in the network, and thus more efficient use of the available bandwidth.

However, the simulations presented in this work do not give a sufficient basis to draw general conclusions about the size of the efficiency gains given by enhanced bridging. The effect of the enhanced bridging algorithm will depend on the traffic pattern in the network in question. All the scenarios described in chapter 6 rely heavily on bridges. In a scenario where only a small fraction of the traffic is bridged, the increased demands on the nodes in the network may be judged too steep a price to pay for the resulting efficiency gain. More simulation results and analysis are needed before the costs can be weighed against the benefits in such scenarios.

What can be said, is that enhanced bridging allows spatial reuse of remote traffic, and thus gives a lower traffic load on all the RPR rings involved in a unicast traffic session. The price to pay for this is first of all the need for more fast memory in the end nodes and bridges to keep the SRCS tables.

7.3 Further work

Scenario 2 describes the effect of reducing the SRCS table sizes in the end nodes and bridges in an RPR network. However, the traffic pattern used in this scenario is not well suited to take advantage of such cache-like SRCS tables. An interesting experiment would be to set up a more realistic traffic scenario to evaluate the idea of reduced SRCS tables. In the real world, some servers would certainly be more popular than others, because they contain a popular service or acts as a gateway to a larger network. Such popular nodes would need large SRCS tables, while less loaded nodes could do with smaller tables. The self-similar nature of network traffic should also be taken into account in such a traffic scenario. This will probably affect the rate at which the entries in the SRCS tables are discarded, and thus the cache hit rate.

The enhanced bridging strategy described in this work demands an initial period of learning before spatial reuse for remote traffic is possible. During this period, frames are flooded through the network, just as they are with basic bridging. This could be avoided by running a global topology discovery algorithm in scenarios where several RPR rings are connected by bridges. In this case, each node would maintain a topology image of all other nodes in the network, both the local and the remote ones. This strategy would lead to more traffic being sent when the topology image is built, and when the image is updated after a protection event or a timeout. The advantage would be that once the image is complete, the data traffic in the network would never have to be flooded. A node would learn the location of remote nodes at startup, and would not be dependent on receiving traffic before flooding can be avoided. The bridge nodes would probably get a special role in such a global topology discovery, to avoid each node having to broadcast its topology information through the entire network. Developing, implementing and evaluating a global discovery algorithm for RPR could be a topic for further research.

A discussion on the side of this work, is whether transparent bridging is at all an appropriate way to interconnect RPR and other 802 networks. Chapter 4 mentions that transparent bridging was originally designed to operate in shared medium networks like Ethernet. As demonstrated in this work, things become more complicated in networks such as RPR, where spatial reuse is an

important characteristic. Being an IEEE 802 standard, RPR is required to support transparent bridging. But it is not a priori given that this is a wise choice of interconnection mechanism between RPR networks. Perhaps could a layer three router perform the same task at a lower cost. In this case, each remote frame would demand more processing in the router, since it operates on IP addresses. A router would also require setting and maintaining routing tables, thus increasing the management complexity of the network. On the other hand, spatial reuse would be secured without extra demands on the nodes on the RPR ring, and local traffic would not be affected. It has not been the aim of this work to analyse the costs of routing traffic between RPR networks. I can therefore not draw any conclusions on the advantages of routing versus bridging in an RPR setting. This could be an interesting subject for further studies.

Bibliography

[1] ANSI. *ANSI T1.105.06-1996 Synchronous Optical Network (SONET) - Physical Layer Specification (Revision of ANSI T1.106-1988)*, 1996.

[2] ANSI/IEEE. *ANSI/IEEE Std. 802.5, 1989 Edition*, 1989. IEEE standard for Token Ring.

[3] ANSI/IEEE. *ANSI/IEEE Std. 802.11, 2000 Edition*, 2000. IEEE standard for Wireless LAN.

[4] ANSI/IEEE. *ANSI/IEEE Std. 802.3, 2000 Edition*, 2000. IEEE standard for Ethernet.

[5] Peter Ball. Introduction to discrete event simulation. In *Proceedings of the 2nd DYCOMANS workshop on "Management and Control : Tools in Action"*, pages 367–376, Algarve, Portugal, May 1996.

[6] Robert Castellano. Basic bridging compliance requirements for draft 1.2. Presentation at RPR WG meeting in Hawaii, November 2002. Available at http://grouper.ieee.org/groups/802/17/proceedings.htm.

[7] Robert Castellano. Enhanced bridging - spatial reuse of 802.17 bridge traffic. Presentation at RPR WG meeting in Vancouver, July 2002. Available at http://grouper.ieee.org/groups/802/17/proceedings.htm.

[8] Robert Castellano. Frame control bit consolidation - brave proposal. Presentation at RPR WG meeting in Atlanta, January 2003. Available at http://grouper.ieee.org/groups/802/17/proceedings.htm.

[9] Stein Gjessing and Fredrik Davik. Avoiding head of line blocking using an enhanced fairness algorithm. *Proceedings IEEE International Conference on Telecommunication (ICT 2002)*, June 2002.

[10] David B. Gustavson. Scalable coherent interface. In *COMPCON Spring '89. Thirty-Fourth IEEE Computer Society International Conference: Intellectual Leverage, Digest of Papers*, pages 536–538, Feb 1989.

[11] Marc Holness. Bridging on 802.17 lan with 802.1d/q compliance. Presentation at RPR WG meeting in Ottawa, May 2002. Available at http://grouper.ieee.org/groups/802/17/proceedings.htm.

[12] Marc Holness. Rpr frame transmission proposal. Presentation at RPR WG meeting in Hawaii, November 2002. Available at http://grouper.ieee.org/groups/802/17/proceedings.htm.

[13] IEEE. P802.1d project authorization request. http://www.ieee802.org/1/mirror/8021/docs2002/P802.1D revision PAR-revised.pdf.

[14] IEEE. *IEEE Std 1596*, 1992. IEEE standard for Scalable Coherent Interface.

[15] IEEE. *IEEE Std. 802.1Q-1998, Draft Standard for Local and Metropolitan Area Networks: Virtual Bridged Local Area Networks*, 1998. IEEE standard for transparent bridges in virtual LANs.

[16] IEEE. *ISO/IEC 15802-3: 1998 [ANSI/IEEE Std. 802.1D, 1998 Edition], Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Common specifications - Part 3: Media Acess Control (MAC) bridges*, 1998. IEEE standard for transparent bridges.

[17] IEEE. *IEEE Std 1394a*, 2000. IEEE standard for Firewire 400.

[18] IEEE. *IEEE draft P802.17/D2.1*, February 2003. Draft 2.1 of the RPR standard.

[19] ISO. *ISO/IEC 7498-1 1994, Information technology - Open System Interconnection - Basic reference model: The BasicModel*, 1994.

[20] Amund Kvalbein and Fredrik Davik. Simulation results for enhanced vs. basic bridging. Presentation at RPR WG meeting in New Orleans, Sept 2002. Available at http://grouper.ieee.org/groups/802/17/proceedings.htm.

[21] Will E. Leland, Murad S. Taqqu, Walter Willinger, and Daniel V. Wilson. On the self-similar nature of ethernet traffic. In Deepinder P. Sidhu, editor, *ACM SIGCOMM*, pages 183–193, San Francisco, California, 1993.

[22] J. Moy. Ospf version 2. Technical Report RFC 2328, IETF, April 1998.

[23] R.M. Needham and A.J. Herbert. *The Cambridge Distributed Computing System.* Addison Wesley, London, 1982.

[24] Fabio Panziere and Santosh K. Shrivastava. A structured description of protocols for the cambridge ring. Technical report, University of Newcastle upon Tyne, Dec 1981.

[25] Vern Paxson and Sally Floyd. Wide-area traffic: The failure of poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244, June 1995.

[26] Radia Perlman. A protocol for distributed computation of a spanning tree in an extended lan. *Ninth Data Communications Symposium*, 1985.

[27] Larry L. Peterson and Bruce S. Davies. *Computer Networks*, pages 172–174. Morgan Kaufmann Publishers, San Francisco, 2000.

[28] Larry L. Peterson and Bruce S. Davies. *Computer Networks*, pages 196–197. Morgan Kaufmann Publishers, San Francisco, 2000.

[29] Floyd E Ross. An overview of fddi: the fiber distributed data interface. *IEEE Journal on Selected Areas in Communications*, 7(7):1043–1051, Sept 1989.

[30] RPRWG. Resilient packet ring 5 criteria. Available at http://www.ieee802.org/17/documents/standards/5criteria.pdf.

[31] Mike Seaman. Loop cutting in the original and rapid spanning tree algorithms, 1999. Available at http://www.ieee802.org/1/mirror/8021/docs99/loop_cutting08.pdf.

[32] Bruce Tolley. Ethernet in the first mile setting the standard for fast broadband access. Cisco White paper. Available at http://www.cisco.com/warp/public/cc/so/neso/efmsol/efm_wp.pdf.

[33] D. Tsiang and G. Suwala. The cisco srp mac layer protocol. Technical Report RFC 2892, IETF, August 2000.

[34] Walter Willinger, Murad Taqqu, Robert Sherman, and Daniel Wilson. Self-similarity through high-variability: Statistical analysis of ethernet lan traffic at the source level. *IEEE/ACM Transactions on Networking*, 5(1):71–86, February 1997.

Appendix A

Dictionary

This dictionary contains selected terms and acronyms used in this work. Some of the terms may have a different meaning in a different context.

Basic bridging Denotes the bridging strategy used in the current RPR draft standard. Relies on flooding to make sure that remote frames are seen by all bridges.

Buffer insertion Strategy used to regulate transmission in ring networks. Frames in transit are inserted into a buffer while the node transmits on the link.

Destination stripping Strategy where the destination node is responsible for removing frames from the transmission medium.

EFM (Ethernet in the First Mile). Initiative in the IEEE to promote Ethernet as a network technology used between a subscriber and the service provider.

Enhanced bridging Denotes an improved bridging algorithm for RPR, which allows spatial reuse of remote traffic.

Filtering database Table kept in the relay part of a transparent bridge. Maps MAC addresses of end nodes to one of the bridge ports.

Flooding Special kind of broadcast used in RPR to make sure that a frame reaches every node on the ring. Flooding can be either *unidirectional*, using only one ringlet, or *bidirectional*, using both.

IEEE (Institute of Electrical and Electronics Engineers). Organisation doing work in a wide range of engineering fields. IEEE is maintaining

several important standards in the field of local and metropolitan area networks. These are gathered in the family of 802 standards.

IETF (Internet Engineering Task Force).

ISS (Internal Sublayer Services). Interface used for communication between the MAC layer and a bridge relay..

LAN (Local Area Network).

LRU (Least Recently Used). Algorithm used to select the oldest entry in a caching or paging system.

MAC address (Media Access Control). A globally unique address given to a piece of hardware. Issued by the IEEE. Used for communication at the MAC layer.

MAC layer Denotes the protocol that controls access to the transmission medium in an 802 network. MAC addresses are used to distinguish the nodes.

MAN (Metropolitan Area Network).

MTU (Maximum Transmission Unit). The size of the largest packet that can be sent over a physical network.

OSI (Open Systems Interconnect). Model that describes the layering structure in a communications system.

Promiscuous mode Bridge operating mode used in early RPR draft standard proposals. Bridges operating in promiscuous mode would present every frame passing on the ring to its relay unit.

Remote frame Frame either transmitted by or destined for a node that is not in the local network segment.

RIF (Routing Information Field). Field in Token Ring frames to used to describe the path of the frame.

RII (Routing Information Identifier). Bit in the Token Ring frame header used to mark the presence of a RIF.

Ringlet Denotes one of the logical rings in a ring network with several links between the nodes.

RSTP (Rapid Spanning Tree Protocol). A later version of STP that enables faster reconfiguration of the network.

SDU (Service Data Unit). The part of a data frame that is used to carry client data.

SRB (Source Route Bridging). An alternative bridging technology used in Token Rings. The source is responsible for calculating the whole path to the remote destination, and include it in every transmitted frame.

Spatial reuse When frames are stripped from the transmission medium by the destination node in a ring network, this allows the remaining part of the ringlet to be re-used by other traffic. This is termed spatial reuse.

SRCS (Spatial Reuse Control Sublayer). Functionality in an RPR node supporting enhanced bridging. Responsible for maintaining a table of mappings between global and local node identifiers, and use this table to allow spatial reuse.

SRP (Spatial Reuse Protocol). Protocol used to achieve a fair sharing of the bandwidth in a ring network. The precursor of RPR, developed by Cisco Systems in the late nineties.

SRT (Source Route Transparent). The successor of SRB, which allows the co-existence of the transparent and source route bridging strategies in a network.

Steering Protection mechanism used in RPR. The sending node uses its topology image to choose the right ringlet for transmission, so that a point of error can be avoided.

STP (Spanning Tree Protocol). Algorithm used by the Bridging Protocol Entity to ensure that a bridged LAN is cycle-free.

VLAN (Virtual LAN). Traffic in a network can be made to appear to belong to a private LAN, by marking it with identifying tags.

VOQ (Virtual Output Queueing). Technique used to avoid head-of-line blocking in queues with multiple outputs. Used in RPR to improve bandwidth utilisation.

WAN (Wide Area Network).

Wrapping Protection mechanism used in RPR and other dual ring networks. Frames reaching a point of failure are wrapped back on the opposite ringlet.