Draft Proposal for

Information Technology—
Telecommunications and information exchange between systems—
Local and metropolitan area networks—
Specific requirements—

Part 17: Resilient packet ring access method & physical layer specifications

Submitted to IEEE 802.17 as the "DVJ Proposal"

Draft 0.20:32
November 13, 2001

Sponsored by the
LAN/MAN Standards Committee of the IEEE Computer Society

Abstract:
This document is a standard for medium-access and physical layer components that meet the functional requirements of a Resilient Packet Ring system as defined by the IEEE 802.17 Working Group. Detailed logical, electrical, and signal processing specifications are presented that enable the production of interoperable equipment.

Keywords:
Metropolitan area network standards, resilient packet rings, ring networks

IEEE Standards Activities Department
Standards Licensing and Contracts
445 Hoes Lane, P.O. Box 1331
Piscataway, NJ 08855-1331, USA

IEEE Standards documents are developed within the Technical Committees of the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Board. Members of the committees serve voluntarily and without compensation. They are not necessarily members of the Institute. The standards developed within IEEE represent a consensus of the broad expertise on the subject within the Institute as well as those activities outside of IEEE that have expressed an interest in participating in the development of the standard.

Use of an IEEE Standard is wholly voluntary. The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Every IEEE Standard is subjected to review at least every five years for revision or reaffirmation. When a document is more than five years old and has not been reaffirmed, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art.

Users are cautioned to check to determine that they have the latest edition of any IEEE Standard.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of all concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason IEEE and the members of its technical committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration.

Comments on standards and requests for interpretations should be addressed to:

Secretary, IEEE Standards Board
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331

IEEE Standards documents are adopted by the Institute of Electrical and Electronics Engineers without regard to whether their adoption may involve patents on articles, materials, or processes. Such adoption does not assume any liability to any patent owner, nor does it assume any obligation whatever to parties adopting the standards documents.

Contact summary

This proposal editor	802.17 Working group editor
David V. James, PhD Chief Architect Network Processing Solutions Data Communications Division Cypress Semiconductor 110 Nortech Parkway San Jose, CA 95134 Work: +1.408.942.2010 Cell: +1.650.954.6906 Fax: +1.408.942.2099 Work: djz@cypress.com Base: dvj@alum.mit.edu	Mike Takefman Cisco Systems, Inc. 365 March Road Kanata, Ontario Canada K2K 2C9 Phone: +1.613.271.3399 FAX: +1.613.271.3333 Email: tak@cisco.com
Fairness verification and simulation	**Technical reviewer**
Stein Gjessing University of Oslo Oslo, Norway Email: steing@ifi.uio.no	David B. Gustavson IEEE Std 1596 Chair 1946 Fallen Leaf Lane Los Altos, CA 94024-7206 Email: dbg@scizzl.com Tel: +1.650.961.0305 Fax: +1.208.475.7525
LAN bridge contributor	**Others**
Robert Castellano Jedai Broadband Networks Email: rc@jedai.com Tel: +1.732.758.9900 x236	With the current political climate, many contributors wish to remain anonymous

Prereviewers, that have directly or indirectly expressed an interest in commenting on early pre-release drafts, include the following:

dvj@alum.mit.edu, tak@cisco.com, steing@ifi.uio.no, dbg@scizzl.com, rc@jedai.com, krishna@coriolisnet.com, hpeng@nortelnetworks.com, pkj@cypress.com, rjf@cypress.com, sgu@cypress.com

Status summary

Incomplete portions:

Vast portions of this document are incomplete. Specific plans for inclusion in the near future include:

1) Bridging. An encapsulating bridging specification, that specifies where the global (as well as RPR local) addresses are placed.
2) Traffic Management. Each segment transmitter provides an indication of choke-point progress, in terms of a running transfer count. That information is sent in the reverse direction, limiting the transmissions of upstream stations to the indicated threshold.

TBDs:

The following items require further refinement:

1) Discovery & protection. Details need refinement; a unifying theme is desirable.
2) Assignment and consistency checks of client-level debts.
3) CRCs. Are these defined at the logical level, or are they physical-layer dependent.
4) Describe the source flush, sent from a node to itself, to determine when it can safely send on a different ring; i.e. when can it conclude that its previously sent frames (belonging to the same flow) have been delivered?
5) Promiscuous mode. Should this be supported? If so, it should be well defined.
6) Babbling station: should stations police their upstream stations:
    a) In case an upstream station is babbling in an uncontrolled fashion?
    b) This could be useful for measuring the receiver's flow volume.

Comment resolutions

All comments on this draft are listed here, with proposed resolution. The intent is to foster communication and quality revisions, without the delay of extensive meetings, teleconferences, and/or video conferences.

1) **Number:** DVJ-0001
Author&date: Dr. Harmen R. van As, 12Sep2001
Concern: The concept of aligned headers is good, as is the use of 64-bit MAC addresses. However, the 64-bit alignment constraint may be too coarse; perhaps 32-bit alignment is better.
Resolution: The frame alignments have been revised and are now based on 32-bit alignment.

2) **Number:** DVJ-0002
Author&date: Robert Caltellano, 02Oct2001
Concern: Sending 64-bit RPR-local destinationStationID/sourceStationID addresses, along with 48-bit destinationMacAddress/sourceMacAddress results in an excessive overhead. Perhaps 8-bit local destinationStationID/sourceStationID addresses should be used, since these semistable values are easily determined during discovery.
Resolution: Accepted.
Other point-to-point interconnects (SerialBus and SCI) have lived to regret their use of address aliases, so some caution was deemed appropriate. Further reflection noted these standards alias short 16-bit identifiers for the global destinationMacAddress/sourceMacAddress values, with nasty configuration-change implications. So, using compact aliases for local identifiers shouldn't suffer from the same difficulties these buses have experienced.

3) **Number:** DVJ-0003
Author&date: Robert Caltellano, 02Oct2001
Concern: Flow control of upstream is necessary to quickly stifle class-B and class-C traffic. Perhaps this flow control information (a measure of the passBC FIFO depth) should be passed within packet headers on the opposing run, minimizing the transmission latency.
Resolution: Accepted.
A four-bit field in the packet header communicates this congestion information.
Resolution: Modified, 10Nov2001.
Two 2-bit fields in the packet header communicates class-A and class-B congestion information.
Resolution: Modified, 11Nov2001.
A 4-bit field in the packet header communicates class-A and class-B congestion information.

4) **Number:** DVJ-0004
Author&date: Robert Caltellano, Nov2001
Concern: The packet header should contain *destinationStationID*, *sourceStationID*, *destinationMacAddress*, and *sourceMacAddress* fields. The destinationStationID fields are needed to strip multicast, remote known-station, and remote flooded frames.
Resolution: Accepted. 11Nov2001
The header now contains this information. A compact header format is also provided. This reduces the overhead of ring-local data-frame and/or control-frame transmissions.

5) **Number:** DVJ-0005
Author&date: Robert Caltellano, Nov2001
Concern: The number of pad bytes need not be specified within the packet header; the number of data bytes are more appropriately sensed by phy-specific signaling and/or encoding.
Resolution: Accepted. 11Nov2001
These bits have been reassigned for other purposes. The encoding space is now sufficient to support 7-bit *stationID* addresses, although the author feels that a 6-bit field would be sufficient.

Acknowledgments

The September2001 meeting presentation of **Dr. Harmen R. van As** reminded others of the need to coordinate bandwidth acquisition actions, so that consistent accounts can be maintained. Refinement of this concept, based on prioritized-conflict-resolution concepts from *IEEE Std 1596-1992 Standard for a Scalable Coherent Interface (SCI)* and isochronous-resource-allocation concepts from *IEEE Std 1394-1995 Standard for a High Performance Serial Bus*, led to the current bandwidth survey proposal.

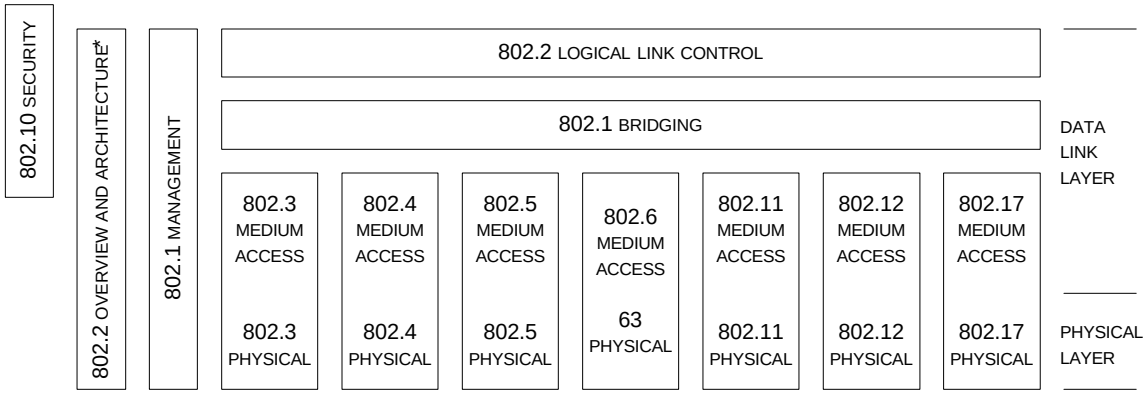
The September2001 meeting presentation of **Stein Gjessing** developed the concept of publishing everyone's congestion-progress rates. Refinement of this concept, based on run-rate concepts developed by Dr. David V. James, led to the current class-B1/class-C fairness algorithms.

The bridging annex was contributed by **Robert Castellano.** Robert was also fundamental in reducing the local source/destination identifiers to 8 bits, placing 4-bit flow control within the packet header, and formulating the ordering solution for bridged networks.

The trio of **Necdet Uzun**, **Carey Kloss**, and **Jim Kao** were responsible for authoring the initial "Cisco" proposal; portions of their text and concepts for figures have been plagiarized within this draft.

Introduction to IEEE Std 802.17

This standard is a part of a family of standards for local and metropolitan area networks. The relationship between the standard and other members of the family is shown below. (The numbers in the figure refer to IEEE standard numbers.)



* Formely IEEE Std 802.1A

This family of standards deals with the Physical and Data Link Layers as defined by the International Organization for Standardization (ISO) Open Systems Interconnection (OSI) Basic Reference Model (ISO/IEC 7498-1:1994.) The access standards define (?xxx?) types of medium access technologies and associated physical media, each appropriate for particular applications of system objectives. Other types are under investigation.

The standards defining the technologies noted above are as follows:

IEEE Std 802 Overview and Architecture. This standard provides an overview to the family of IEEE 802 Standards.

ANSI/IEEE Std 802.1B and 802.1k [ISO/IEC 15802-2]LAN/MAN Management. Defines an OSI management-compatible architecture along with services and protocol elements for use in a LAN/MAN environment for performing remote management.

ANSI/IEEE Std 802.1D Media Access Control (MAC) Bridges. Specifies an architecture and protocol for the interconnection of IEEE 802 LANs below the MAC service boundary.

ANSI/IEEE Std 802.1E[ISO/IEC 15802-4] System Load Protocol. Specifies a set of services and protocol for those aspects of management concerned with the loading of systems on IEEE 802 LANs.

ANSI/IEEE Std 802.1F Common Definitions and Procedures for IEEE 802 Management Information .

ANSI/IEEE Std 802.1G [ISO/IEC 15802-5] Remote Media Access Control (MAC)Bridging .Specifies extensions for the interconnection ,using non-LAN communication technologies, of geographically separated IEEE 802 LANs below the level of the logical link control protocol.

IEEE Std 802.1H [ISO/IEC TR 11802-5] Media Access Control (MAC) Bridging of Ethernet V2.0 in Local Area Networks .

ANSI/IEEE Std 802.2 [ISO/IEC 8802-2] Logical Link Control.

ANSI/IEEE Std 802.3CSMA/CD Access Method and Physical Layer Specifications.

ANSI/IEEE Std 802.4 [ISO/IEC 8802-4] Token Passing Bus Access Method and Physical Layer Specifications.

ANSI/IEEE Std 802.5 [ISO/IEC 8802-5] Token Ring Access Method and Physical Layer Specifications.

ANSI/IEEE Std 802.6 [ISO/IEC 8802-6] Distributed Queue Dual Bus Access Method and Physical Layer Specifications.

ANSI/IEEE Std 802.10 Interoperable LAN/MAN Security.

ANSI/IEEE Std 802.11[ISO/IEC DIS 8802-11] Wireless LAN Medium Access Control (MAC)and Physical Layer Specifications.

ANSI/IEEE Std 802.12 [ISO/IEC 8802-12]Demand Priority Access Method, Physical Layer and Repeater Specifications.

ANSI/IEEE Std 802.17 Resilient Packet Ring Access Method and Physical Layer Specifications.

In addition to the family of standards, the following is a recommended practice for a common Physical Layer technology:

IEEE Std 802.7IEEE Recommended Practice for Broadband Local Area Networks.

Table of contents

List of figures

List of tables

Change history

The following table shows the change history for this specification.

Major checkpoints

Date	Description
16Sep2001	Major changes to accommodate inputs from Sep2001 meeting: 32-bit alignment of headers and data bandwidth surveys published-rate based arbitration
05Oct2001	Major changes to accommodate feedback from limited distribution 01Oct2001 draft: local sourceStationId and destinationStationId reduced to 8 bits 4-bit priority level placed in RPR header RPR payload reduced to Ethernet packet format

1. Overview

This document, the DVJ proposal, represents an interative proposal for a Resilient Packet Ring (RPR) Access Protocol. The acronym DVJ stands for *diminutive verifiable jitter*, to emphasize the TDM nature of the proposed class-A traffic-management protocols and the bounded latency of the class-B traffic management protocols.
This document represents a consensus position of multiple participants, but has not yet been reviewed by a wide audience and definitely has not been approved by the P802.17 Working Group.

1.1 Scope and purpose

The following scope and purpose statements, as stated in the project PAR, apply to this standards activity.

Scope: Define a Resilient Packet Ring Access Protocol for use in Local, Metropolitan, and Wide Area Networks, along with appropriate Physical Layer specifications for transfer of data frames at rates scalable to multiple gigabits per second.

Purpose: The standard will define a very high-speed network protocol that is optimized for frame transmission in resilient ring topologies. Current standards are either optimized for TDM transport, or optimized for mesh topologies. There is no high-speed (greater than 1 billion bits per second) networking standard in existence that is optimized for frame transmission in ring topologies.

Distinctive properties of RPR include the following:

1) Efficient. Spatial reuse (concurrent transfers over nonoverlapping sements) allow cumulative transfer rates to exceed the capacity of any individual link.
2) Classy. Three traffic priorities (class-A, class-B, and class-C) are supported.
   a) Class-A: Low jitter provisioned bandwidth, for time division multiplexed (TDM) traffic.
   b) Class-B: Bounded-latency provisioned bandwidth, used for prioritized data transmissions.
      Subclass-B0 is provisioned guaranteed bandwidth.
      Subclass-B1 is provisioned statistical bandwidth, preferred over class-C.
   c) Class-C: Opportunistic (best effort) traffic, as used within traditional IP applications.
3) Scalable. Large numbers of stations may be attached to a single ring and rings of over 2,000 kilometers are supported.
4) Plug & play. Stations can be attached, discovered, and detached without operator intervention; the equality of nodes eliminates complexities associated with ring-master negotation protocols.
5) Protected. Packet CRC protection and wrapping ensure the correctness and completion of transfers, with levels of robustness found in SONET BLSR specifications.
6) Flexible. The MAC layer definition is applicable to a wide range of physical layers.

1.2 RPR topologies

1.2.1 RPR topology constraints

RPR protocols are highly scalable, in the sense that 1-to-63 stations can be supported on a small or large (up to 2,000 kilometer) duplex ring, as illustrated in Table 1.1. Although the RPR protocols are designed to be physical medium independent and speeds beyond 40Gbs are expected. Bandwidth allocation protocols are rely on the ringlet-nature of an RPR topology to efficiently cope with many concurrently "on-the-wire" packets.

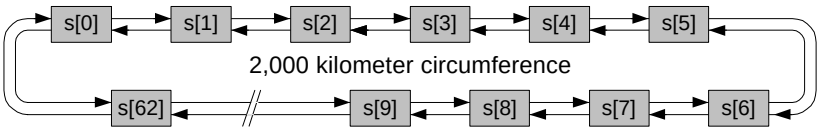


Table 1.1—RPR size constraints

RPR protocols are based on the use of duplex links, so that each ring normally consists of counter-rotating ringlets. Each of these ringlets operate in a largely independent fashions, although the flow control of data transfers involve control packets placed on the opposing ringlet.

1.2.2 Ring topologies

RPR is targeted for cable-ring topologies and maximizes bandwidth capabilities through the use of full-duplex cabling, as illustrated in Table 1.2. The full-duplex cable infrastructure normally allows concurrent transmissions on the clockwise and counter-clockwise rings, as illustrated in the left of Table 1.2. The client may choose to send data in either direction, based on shortest-distance, available bandwidth, or higher link capacity criteria.

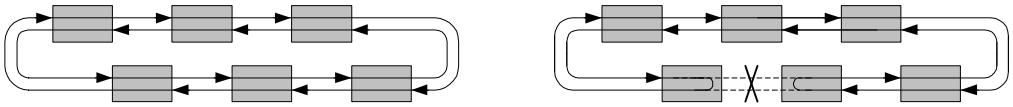


Table 1.2—Ring topologies

After a link failure, communication continues (but at a reduced rate) over the remaining ring, as illustrated in the right of Table 1.2. The clients are expected to direct data (to the right or left) based based solely on the relative physical location of the destination. (This is called steering). Although a full ring is present, steering inhibits utilization of the loop-back paths within the terminal stations.

All stations have two attachments (an attachment is a location where packets can be inserted or extracted), which allows packets to be sent in their preferred direction. On the average, assuming randomly-distributed traffic and preferred direction prediction, the average path lengths can be reduced by nearly a factor of four (when compared to an open-loop topology). Bandwidth improvements of 2.5 are more typical, due to less-than-random traffic distributions.

Similar performance enhancing techniques have also been used on serial-copper SSA, serial-fiber FDDI (Fiber Distributed Data Interface), and parallel-copper SCX interconnects.

1.2.3 Spatial reuse

Spatial reuse is a concept used on rings to increase the aggregate bandwidth beyond the capacity of an individual link. Spatial reuse occurs when concurrent data transfers occupy nonoverlapping portions of a ringlet, as illustrated in the left side of Table 1.3. In this example, traffic can be sent between stations *s[0]*-and-*s[2]* without affecting the bandwidth available between stations *s[4]*-and-*s[5]*.

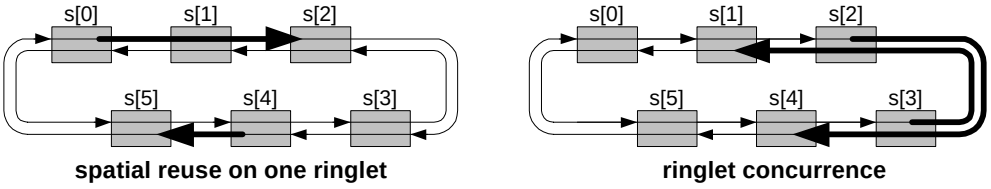


Table 1.3—Concurrent data transfers

The counter-rotating nature of the ringlets also supports concurrent transfers on overlapping segments, as illustrated on the right side of Table 1.3. In this example, traffic is being sent between stations *s[2]*-and-*s[4]* while opposing traffic is being sent between stations *s[3]*-and-*s[1]*. Concurrent transfer are possible, even though the traffic occupies opposing runs on the bidirectional link between stations *s[2]*-and-*s[3]*.

1.2.4 Traffic classes

RPR supports three classes of client traffic, although one class is partitioned into two distinct subclasses on the ring, as illustrated in Figure 1.1 and listed below. Each station is required to police its class-A and class-B traffic to avoid exceeding its provisioned limits.

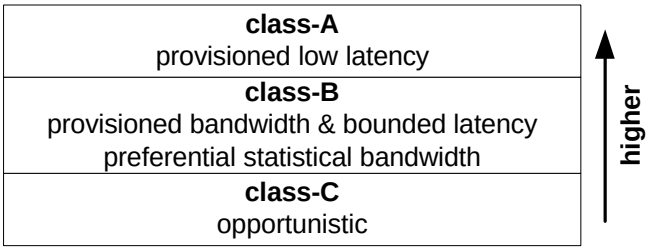


Figure 1.1—Traffic classes

1) Class-A: Provisioned guaranteed-latency bandwidth.
   This service is expected to be used for transmission of streaming audio and/or video traffic.
2) Class-B: Provisioned bandwidth, with the following properties:
   a) LevelB0: Guaranteed bandwidth with bounded-latency.
   b) LevelB1: Preferential bandwidth with statistical bandwidth guarantee.
3) Class-C: Fairly assigned unprovisioned or unused higher-priority bandwidth.
   This service is expected to be used for transmission of best-effort traffic.

The provisioning of class-B bandwidth is based on the amounts of requested provisioned class-B traffic, nicknamed levelB0 traffic. The implied preferential statistical traffic class-B traffic, nicknamed class-B1 traffic, is proportional to the negotiated levelB0 bandwidth. The constant multiplier that defines the ratio of class-B0 to class-B1 traffic is TBD.

1.2.5 RPR packet buffers

Packet transfers involve transmit queues (where packets are placed for MAC-layer processing), receive queues (when packets are placed for client-layer processing), and transit FIFOs, as illustrated in Figure 1.2. The purpose of the transitA buffer is to hold class-A packets that arrive during this station's transmissions; the purpose of the transitBC buffer is to hold class-B or class-C packets that arrive during this station's class-A transmissions.

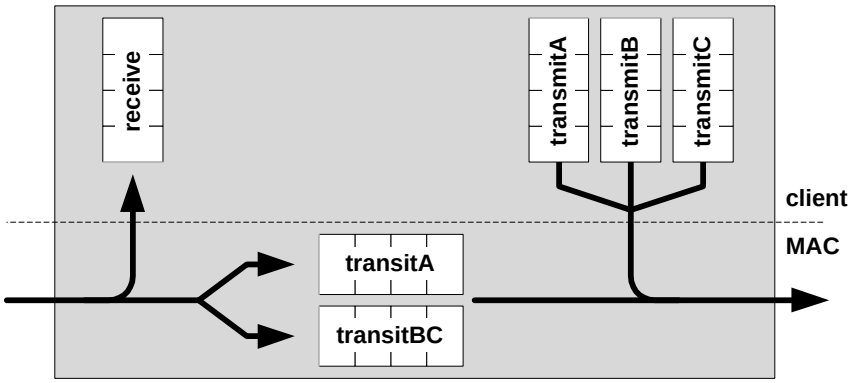


Figure 1.2—RPR packet buffers

Multiple transmit queues are provided, for class-A, class-B, and class-C traffic respectively. These transmit buffers are located in the client, which reduce the cost of the MAC while providing flexibility for vendor-dependent just-in-time scheduling protocols.

1.3 MAC interface functionality

The MAC interface definition provides interfaces to a pair of attachment points, as illustrated in Figure 1.3. The attachment interface receives data and FIFO fill-level indications from the client; the attachment transmits data and transmit-permission information.

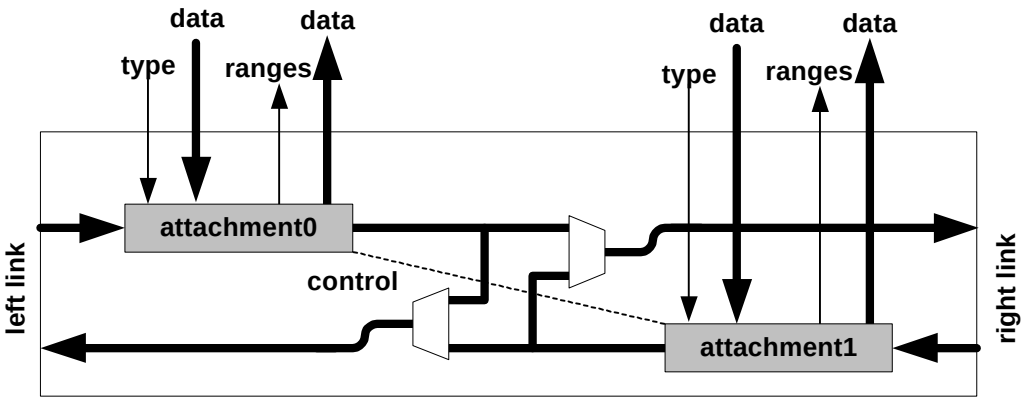


Figure 1.3—MAC interface signals

The data paths are expected to flow through two multiplexers, whose controls are semistable in that they change during protection events, rather than on a packet-by-packet basis.

This concept of an electronically-switched station is not new; a similar capability is provided by stations attached to Serial Bus. Although Serial Bus supports N-port attachments, a 2-port design is sufficient to support the common topologies and simplifies the hardware design.

1.4 Congestion management

For a shared ring topology in which the RPR MAC is used, each ring segment carries both local client traffic and the traffic from other upstream client. Unless the upstream clients control their access rates, their traffic may consume the entire ring segment bandwidth, creating congestion and hence blocking the local client from gaining access to the media.

For the higher-priority class-A and class-B0 traffic, RPR employs congestion avoidance mechanism to prevent congestion before it occurs. The congestion avoidance involves rate limiting the traffic at its source, such that the cumulative traffic across the link never exceeds the link capacity.

1.4.1 Congestion avoidance

Congestion avoidance is a two phases mechanism: bandwidth allocation & rate limiting. Bandwidth allocation is carried out to preprovision access to ring segments and more importantly to ensure the clients expectations never exceed the capacity of an individual ringlet segment.

For bandwidth allocation, each MAC limits the transmission rates of its client according to the policing protocols defined in xx. The access rate control prevent upstream MACs from gaining access more than their allocated rate, creating congestion at a down stream ring segment.

1.4.2 Virtual output queues avoid head-of-line blocking

The client of an RPR MAC may send traffic to multiple destinations traversing multiple ring segments. If the MAC does not allow an independent access rate per destination, it is possible that the MAC sets the access rate low to satisfy the bandwidth allocated by one remote congested destination and severely limits the access rates to nearby uncongested destinations.

Thus, destination insensitive congestion management protocols can cause head-of-line blocking: a frame destined to uncongested destination is forced to behind a head-of-line frame that whose destinations is congested. Until the head-of-line frame is removed from the FIFO all following frames are blocked.

A well-known solution to this head-of-line blocking problem is a virtual output queue implementation, wherein the client maintains a dedicated transmit-queue for each destination. With a per-destination queue, a frame for one destination is no longer blocked by another frame for a different destination, hence eliminating head-of-line blocking completely.

In order to allow the client to maximize a spatial reuse property of the ring, the RPR MAC implements independent access rate control for each ring segment, allowing the RPR client to provide virtual-output-queue implementations. To support virtual-output-queue implementations, the RPR MAC provides range information (number of allowed hops to the destination) for each of the congestion-level indications sent between the MAC and client.

1.4.3 Congestion control

On traditional backplane buses, arbitration protocols are used to resolve conflicts and schedule packet transmissions. To avoid transmission conflicts, arbitration protocols are invoked before every packet transmission. The arbitration protocols may invoke prioritized and opportunistic arbitration protocols to determine which packets can be sent.

In the networking environment, the overhead of invoking arbitration protocols before each packet transmission is no longer acceptable. Transmission conflicts are more efficiently resolved by transmitting during inter-packet gaps, buffering conflicting arrivals in bypass FIFO storage, and repeating the buffered packet when packet transmissions cease.

Flow control protocols are still needed to allocate bandwidth, so that the station's service level agreements (SLA) can be met, but are only invoked when heavy congestion conditions necessitate their use. A more efficient opportunistic transmission protocol is used under light loading conditions, to maximize bandwidth utilization under typical conditions.

1.4.4 Flow-control signals

For efficiency, arbitration signals and data packets normally flow in opposite directions, as illustrated in Figure 1.4. Clockwise data transmissions (solid lines) are coupled to counterclockwise arbitration signals (dotted lines), as illustrated in the left half of Figure 1.4. Counterclockwise data transmissions (solid lines) are coupled to clockwise arbitration signals (dotted lines), as illustrated in the right half of Figure 1.4.

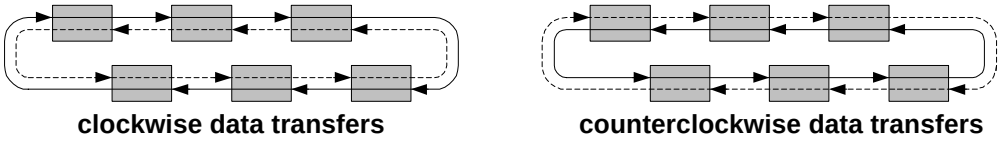


Figure 1.4—Opposing data and flow-control flows

The flow-control signals are encapsulated and sent as small class-A data packets. These packets are stripped at their downstream neighbor; their contents are algorithmically merged with that station's indications and sent further upstream.

1.4.5 Rate-limited class-A traffic

Each station is required to rate-limit its ringlet-bound class-A traffic. Distinct independent rate-limit hardware is associated with each ringlet, as illustrated in Figure 1.5. For the sake simplicity, only the top-run rate-limiting component is discussed. The rate-limit is a classic leaky-bucket protocol, described in the remainder of this subclause.

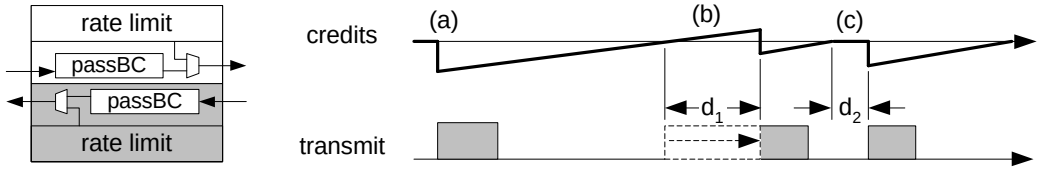


Figure 1.5—Rate-limiting class-A traffic

Rate limiting protocols limit the packet transmissions until a credits value is positive. At that point, the credits value (a) is decremented by the packet length. The negative credits value is updated, once each cycle, by the average amount of bandwidth allowed to be consumed in that cycle, so that a positive credits value is ultimately restored.

The credits value continues to increase (b) while class-A packets remain queued, even when these packet transmissions remain blocked, possibly due to a large passing-through packet. When that transmission occurs, the credits value is (once again) decremented by the packet length.

Any positive credits value is cleared (c) to zero whenever no class-A packets is queued. The intent is to avoid bunching of class-A packets, because bunching would impact the guaranteed latency of class-A traffic within other nodes.

The rate limiting of class-A traffic serves two purposes:

1) Bandwidth. Each station cannot consume more than its provisioned class-A bandwidth.
2) Latency. Bursts of class-A traffic do not effect others class-A transmit latencies.

1.4.6 Reactive class-A control

Two schemes are specified for ensuring class-A transmissions: reactive and proactive. Reactive congestion management involves signaling one's upstream neighbor (1) when its transmissions interfere with class-A traffic, as indicated by a partially full passBC FIFO, as illustrated in Figure 1.6. The duty of the upstream neighbor is to lessen (2) its returning lower-class traffic before the passBC FIFO fills.

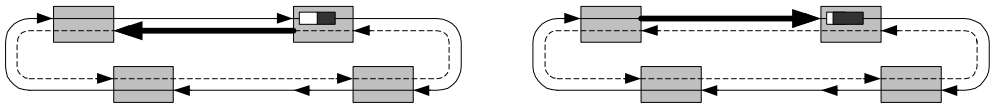


Figure 1.6—Reactive congestion management

1.4.7 Proactive class-A control

Proactive congestion management involves maintain fixed levels of class-A traffic (illustrated by black boxes) on all segments, as illustrated in Figure 1.7. Stations do not simply strip their class-A traffic, but (as necessary) propagate null class-A traffic on behalf of other nodes.

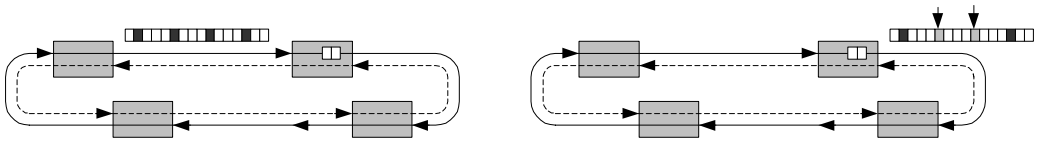


Figure 1.7—Proactive congestion management

This congestion-avoidance strategy can dramatically reduce the minimal passBC buffer size, while still utilizing the efficiencies afforded by spatial reuse. However, the class-A traffic is explicitly reserved, even when not being used, and is therefore unavailable for awaiting opportunistic transmissions.

RPR does not mandate the use of congestion-recovery or congestion-avoidance schemes, allowing this to be an implementation consideration. Both protocols are structured to interoperate, with the small costs of

interoperability borne by the congestion-recovery implementations (normally stripped class-A traffic is marked invalid but allowed to circulate).

Proactive congestion management details are under development; details are missing from this specification.

1.5 Traffic management

Traffic management involves congestion avoidance and congestion relief. Congestion avoidance involves prescheduling of provisioned traffic bandwidths, leaving the unprovisioned or unused provisioned bandwidth for opportunistic uses. Congestion relief involves throttling the opportunistic stations when the progress of provisioned traffic is threatened, due to changes in offered load or load distributions.

Different congestion management protocols are applied to class-A, class-B, and class-C traffic, as each of these traffic types has distinct sets of performance requirements. The class-A traffic is the most demanding, with strict bandwidth and latency guarantees. The class-C traffic has no guarantees, but attempts to ensure the residual best-effort bandwidth is efficiently and fairly allocated.

Management of class-A traffic involves signaling of MAC-resident transit-queue (a) depths, to maintain sufficient space for deterministic class-A transmissions, as illustrated in Figure 1.8. Management of class-B traffic involves upstream signaling of transmit-queue (b) depths. Management of class-C traffic involves upstream signaling (c) of run-rate metrics, so that an impoverished station can throttle well-to-do upstream stations. These techniques are described further in Clause 7.

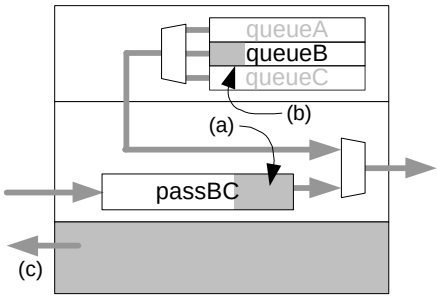


Figure 1.8—Congestion sensing and signaling

1.6 Station addressing

1.6.1 Local unicast transmissions

The RPR addressing protocol is based on the use of unique destination and source station addresses, each of which identify one-and-only-one of the directly attached stations. A local unicast packet is sent directly from station *Sa0* to *Sa5*, based on the station addresses within the RPR header, as illustrated in Figure 1.9.

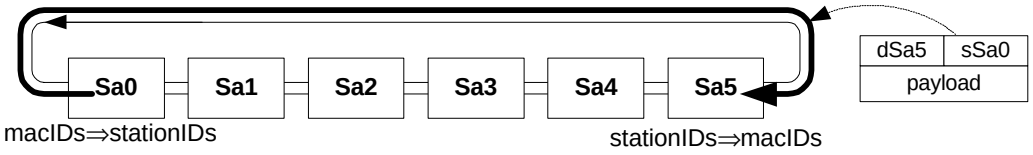


Figure 1.9—Local unicast transmission

The clients are expected to direct packets based on their *macID* addresses. During the client-to-MAC transfer, these *macID* addresses are converted to ringlet-local *stationID* addresses. During the MAC-to-client transfer, these *stationID* addresses are restored to their original *macID* addresses. Although this process may seem quite complex, the use of small stationID addresses simplifies the MAC design and reduces the packet header size.

1.6.2 Global unicast transmissions

A global unicast packet is sent directly from station *Sa0* to *Sa5*, based on local station addresses within the RPR header, as illustrated in Figure 1.11. However, the bridge prepends the payload with the globally-unique destination and source MAC addresses, *destMacSb0* and *srcMacSd1*. The intent is to have the remote bridge strip the RPR headers, leaving the destination and source MAC addresses unchanged.

Other standards have chosen to transport Ethernet packets without encapsulation. This would have complicated packet-strip decisions (many remote addresses would have to be placed in the MAC) and flow-control protocols, which require source-knowledge of the destination station to avoid head-of-line blocking conditions.

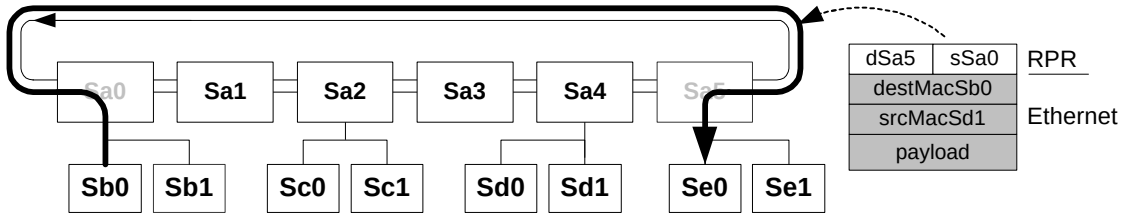


Figure 1.10—Global unicast (encapsulated Ethernet) packets

1.6.3 Broadcast packet addressing

For network bridges, a global broadcast packet is sent directly from station *Sa0* to the RPR-attached *Sa4* endpoint, based on local station addresses within the RPR header. Another global broadcast packet is concurrently sent in the opposite direction, from station *Sa0* to the station *Sa2* endpoint, as illustrated in Figure 1.11. The bridge sets the broadcast bit in the RPR header and prepends the payload with the globally-unique destination and source MAC addresses, *destMacSb0* and *srcMacSd1*. This facilitates the flooding of global packets when the destination RPR station address is unknown.

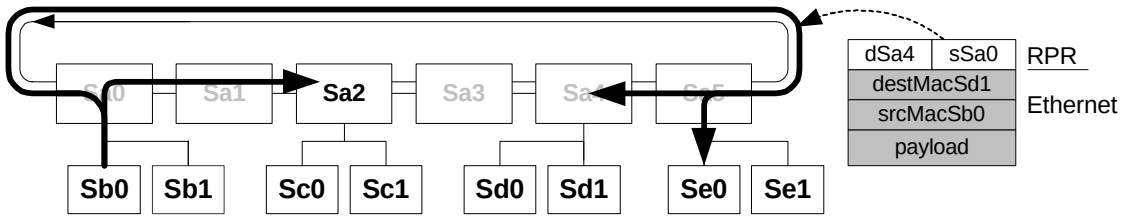


Figure 1.11—Global broadcast (encapsulated Ethernet) packets

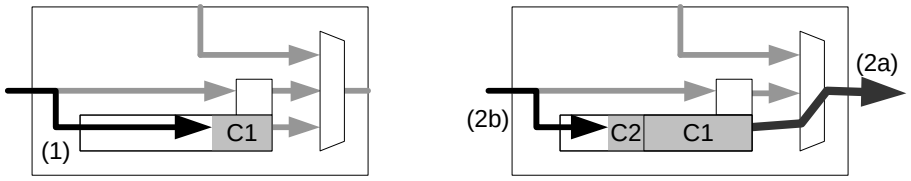
The aforementioned broadcast routing technique assumes that station Sa0 was sending to Sa2 on the shortest inside-ringlet path. Broadcasts with unknown destinations must therefore be split and following the paths of unicast packets. If broadcasts were allowed to take a different path (such as fully traversing one of the ringlets), directed and broadcast packets could be unintentionally reordered.

1.7 Queuing options

The terms cut-through and store-and-forward describe options for the processing of pass-through traffic. Implementations are use either protocols; store-and-forward is simpler but cut-through has the possibility of improved performance.

1.7.1 Store-and-forward

Store-and-forward processing delays packet forwarding until after the final portion of the packet has been received, as illustrated by the Figure 1.12 sequence. A packet cannot be retransmitted (1) before the trailing portion of the packet has been received. Once a full packet is available, that packet can be retransmitted (2a) while the following packet (2b) is being received.

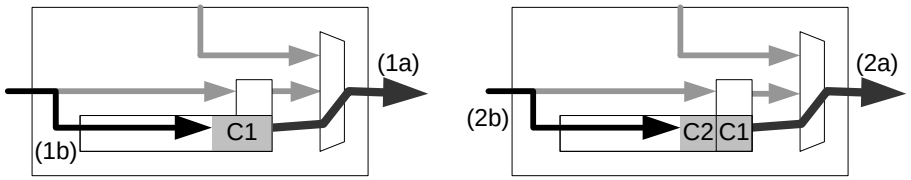


1) Receiving C1 **2) TransmitC1; receive C2**

Figure 1.12—Store&forward flows

1.7.2 Cut through

Cut-through processing allows packet forwarding before the final portion of the packet has been received, as illustrated by the Figure 1.13 sequence. The leading portion of a packet is retransmitted (1a) while the trailing portion of the packet (1b) is being received. The retransmission (2a) of the cut-through packet continues while new packets (2b) are received.



1) Transmit C1; receive C1 **2) Transmit C1; receive C2**

Figure 1.13—Cut-through flows

1.7.3 Preemption

The term preemption describes a technique for suspending the transmission (1) of pass-through lower-class traffic after class-A (2) traffic is queued, as illustrated in Figure 1.14. The suspension of lower-class traffic allows the class-A traffic (3) to be sent with lower latency; the lower-class traffic (4) can resume thereafter.

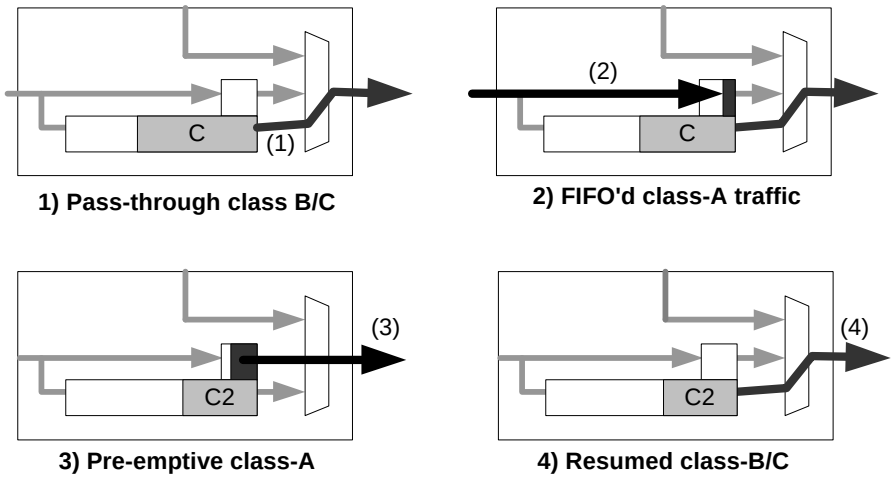


Figure 1.14—Pre-emptive traffic flows

Preemptive packet processing has the effect of splitting lower-class packets, as illustrated in the left of Figure 1.15. If packetC is stripped before packetA, a between-packet void can also be generated, as illustrated in the right of Figure 1.15.

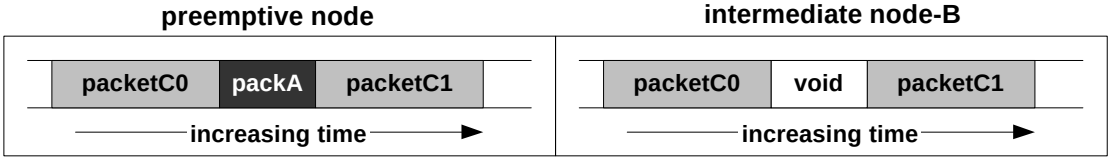


Figure 1.15—Preemptive packet processing

A station may discard these void symbols while transmitting idles or packetized symbols, with the intent of reducing the depth of packets (or packet fragments) within fifoA or fifoBC components. The void symbols are distinct from other between-packet idles, however, in that their transmissions shall not be interrupted by the transmission of other lower-class components.

If not discarded previously, the void symbols are converted into the normal between-packet idles when the preceding packet component (packet-A, in Figure 1.15) is stripped at its destination station.

Preemptive packet processing relies on distinct code symbols to identify the border between packetized lower-class data and the start of preemptive class-A packets. Because the necessary control codes may not be supported by all physical layers, this feature is physical-layer dependent and not mandated by this standard. Supportive physical layers may mandate preemption, may prohibit preemption, or may define an interoperable preemption option.

Preemption mandates additional hardware, including store-and-forward sequencing of receive queues, to isolate the application from the packet fragments generated on the interconnect. To ensure interoperability, preemptive transmitters connected to downstream non-preemptive receivers shall use store-and-forward processing techniques to merge received packet fragments before their retransmission, as illustrated in Figure 1.16.

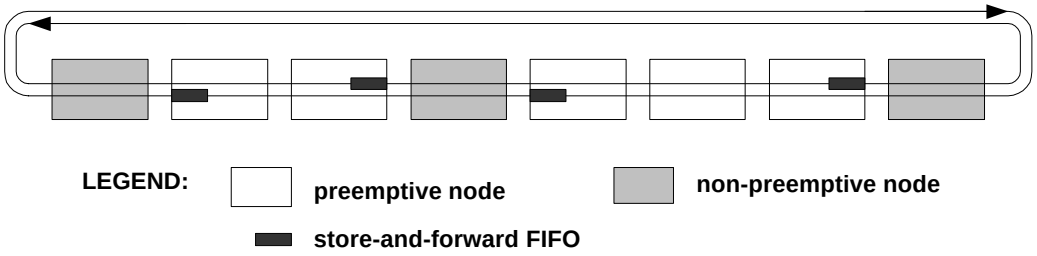


Figure 1.16—Preemptive store-and-forward boundaries

1.8 Future possibilities

1.8.1 Heterogeneous link bandwidths

This standard supports rings constructed from different-speed links, as illustrated in Figure 1.17. The goal is to burden the multispeed-capable stations, not the single-speed capable stations, with the costs of supporting a multispeed topology. This burden is not insignificant: boundary stations must supply large rate-matching transit buffers (several times the ringlet circulation time) to avoid dropping high-speed traffic while asserting backpressure through upstream higher-bandwidth stations.

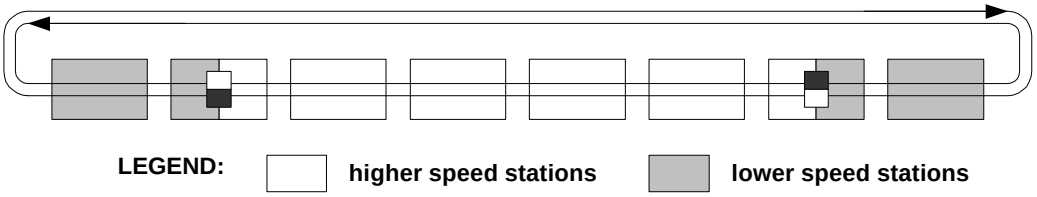


Figure 1.17—Heterogeneous link bandwidths

1.8.2 Mesh topologies

Mesh topologies can support higher bandwidths, due to the additional links and a reduced number of links in most source-to-destination paths, as illustrated in Figure 1.18. However, additional routing and buffering specifications could be required in order to support transfers of packets from one ring to another.

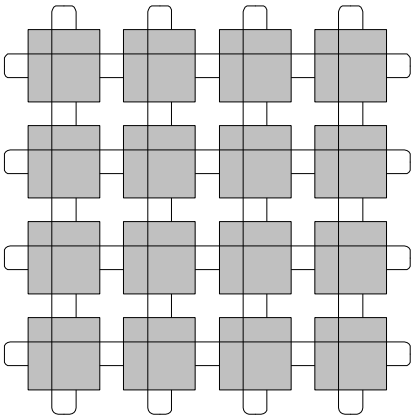


Figure 1.18—Mesh topologies

1.8.3 Hub topologies

An RPR topology can include larger multiported stations, called hubs, as illustrated in Figure 1.19. An N-ported hub can be used to connect a number of lower-cost single-ported stations.

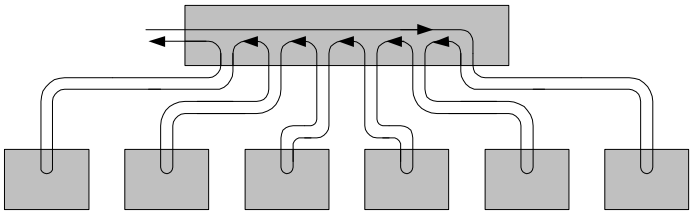


Figure 1.19—Hub topologies

1.8.4 Redundant connections

Hub-like 3-ported stations can also be used to support redundant fault tolerant connections, as illustrated in Figure 1.20.

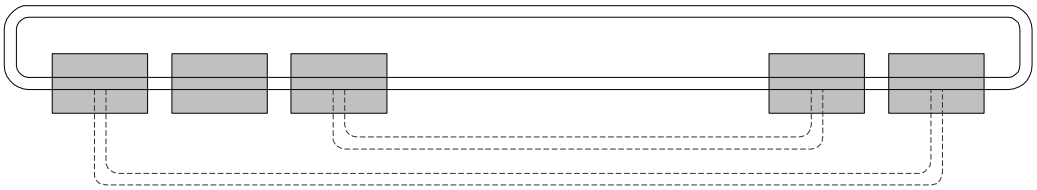


Figure 1.20—Redundant topologies

2. References

The following standards contain provisions, which through reference in this document, constitute provisions of this standard. All the standards listed are normative references. Informative references are given in Annex A. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below.

[R1] ANSI/ISO 9899-1990, Programming Language—C.[1,2]

1 Replaces ANSI X3.159-1989.

2 ISO documents are available from ISO Central Secretariat, 1 rue de Varembé, Case Postale 56, CH-1211, Genève 20, Switzer-land/Suisse; and from the Sales Department, American National Standards Institute, 11 West 42nd Street, 13th Floor, New York, NY 10036-8002, USA

3. Definitions

The definitions for 802.17 are controlled by a separate terms-and-definitions draft (T&D) and subject to change. A recent version of the T&D draft has been included in this proposal for the benefit of the author and readers .

NOTE—Footnotes will be deleted from final document unless otherwise noted.
Text appearing in angle brackets will be removed prior to ballot on the 802.17 standard. This material is maintained for discussion purposes during the standardization effort.
Each definition listed below will contain a comment at the end of the definition giving the first place of usage within this standard. The comments will be of the form "(See IEEE 802.17, Clause 10.)" or "(See IEEE 802.17, 4.5.6.7.)". All definitions not referenced to a usage in this standard shall be removed without need of vote before balloting of the standard.

3.1 Conformance levels

3.1.1 expected: A key word used to describe the behavior of the hardware or software in the design models *assumed* by this Specification. Other hardware and software design models may also be implemented.

3.1.2 ignored, ign: A term used to describe the fields within registers or frames, whose zero or last-written values shall be ignored.

3.1.3 may: A key word that indicates flexibility of choice with *no implied preference*.

3.1.4 shall: A key word indicating a mandatory requirement. Designers are *required* to implement all such mandatory requirements.

3.1.5 should: A key word indicating flexibility of choice with a strongly preferred alternative. Equivalent to the phrase *is recommended*.

3.1.6 reserved fields: A set of bits within a data structure that are defined in this specification as reserved, and are not otherwise used. Implementations of this specification shall zero these fields. Future revisions of this specification, however, may define their usage.

3.1.7 reserved values: A set of values for a field that are defined in this specification as reserved, and are not otherwise used. Implementations of this specification shall not generate these values for the field. Future revisions of this specification, however, may define their usage.

3.2 Glossary of terms

A large number of network and interconnect-related technical terms are used in this document. These terms are defined below:

NOTE—The following terms are proposed by the author of this draft:

3.2.1 aligned: A term which refers to the constraints placed on the address of the data; the address is constrained to be a multiple of the data format size.

3.2.2 big endian: A term used to describe the arithmetic significance of addressed data-bytes within a multibyte register. Within a big-endian register or register set, the data byte with the largest address is the least significant.

3.2.3 byte: An 8-bit entity. In other standards, this is also called an octet.

3.2.4 class-A: Data traffic for which the transmission bandwidth is provisioned and low latency is ensured by assignment of the maximum effective transmission priority.

3.2.5 class-B: Data traffic for which the transmission bandwidth is provisioned and latency is bounded by assignment of the high effective transmission priority.

3.2.6 class-C: Data traffic for which the transmission bandwidth is unprovisioned; this traffic class has no minimum bandwidth or maximum latency guarantees.

3.2.7 doublet: A data format or data type that is 2 bytes in size.

3.2.8 hexlet: A data format or data type that is 16 bytes in size.

3.2.9 octlet: A data format or data type that is 8 bytes in size. Not to be confused with an octet, which has been commonly used to describe 8 bits of data. In this document, the term byte, rather than octet, is used to describe 8 bits of data.

3.2.10 quadlet: A data format or data type that is 4 bytes in size.

NOTE—The following terms were obtained from the T&A draft:

3.2.11 802.17: See IEEE Std. 802.17.

3.2.12 agent: [802.3-2000 1.4.30 (modified)] A network management entity (NME) which can be used to configure the station and/or collect data describing operation of that station.

3.2.13 <all-stations MAC address[3]: TBD >

3.2.14 backpressure: The sending of a *control frame* in the *upstream* direction, to stop or slow the flow of *data traffic*.

3.2.15 <bandwidth: Note: The term *bandwidth* is applicable to the *physical layer* and should **not** be used in reference to the *MAC layer*. The term *data-rate or capacity* is used instead.>

3.2.16 best-effort service (BES): A service not providing any *QoS* guarantee.

3.2.17 bit error ratio (BER): [802.3-2000 1.4.47] The ratio of the number of bits *received* in error to the total number of bits *received*.

3.2.18 bit rate: [ISO/IEC2382-09 9.03.01 (modified)] The speed at which bits are transferred.

3.2.19 bridge: [IEC2382-25 25.01.12 (modified)] A functional unit that interconnects two networks that use the same logical link control protocol but may use different *medium access control protocols*. *Local area networks (LANs)* and *metropolitan area networks (MANs)* are example of *networks* that a *bridge* may interconnect.

3.2.20 bridged network: [(C/LM) 10038-1993, 802.1G-1996 (modified)] A concatenation of individual *networks* interconnected by *MAC bridges*.

[3] Brian Holden to investigate.

3.2.21 broadcast address: [ISO/IEC2382-25 25.01.13] A *group address* that identifies the set of all *stations* on the *network*.

3.2.22 broadcast: [802.5-1998 1.3.10] The act of sending a *frame* addressed to all *stations*.

3.2.23 buffer insertion ring (BIR): An *access technique* for ring media that gives absolute priority to passthru traffic except when transmission of an ingress frame is in progress.

3.2.24 buffer: An area of memory used for temporary storage of *frames*.

3.2.25 bursty (burstiness): Characterization of traffic as using the maximum data-rate of a channel only intermittently.

3.2.26 capacity: The maximum *data-rate* supported by a *medium* or *channel*.

3.2.27 channel: see *transmission channel*.

3.2.28 class of service (CoS): The categorization of traffic according to *relative* delivery priority.

3.2.29 closed user group (CUG): [09.08.14, 610.7-1995] A specified group of *network* users who are permitted communications among themselves but not with other *network* users.

3.2.30 committed burst size (Bc): [ITU I.233.1 A.5 (modified)] The maximum amount of data (in bits) that the *network* agrees to *transfer*, under normal conditions, during a time interval *Tc*.

3.2.31 committed information rate (CIR): [ITU I.233.1 A.8 (modified)] The information transfer rate which the network is committed to transfer under normal conditions. The rate is averaged over a minimum interval of time (*Tc*).

3.2.32 committed rate measurement interval (Tc): [ITU I.233.1 A.7 (modified)] The time interval during which the user is allowed to send only the committed amount of data (*Bc*) and the excess amount of data (*Be*).

3.2.33 congestion avoidance: [ITU I.233.1 A.11 (truncated)] Procedures initiated at or prior to the onset of mild congestion in order to prevent congestion from becoming severe.

3.2.34 congestion control: [ITU I.233.1 A.9] Real-time mechanisms to prevent and recover from congestion during periods of coincidental peak *traffic* demands or *network* overload conditions (e.g. resource failures). *Congestion control* includes both *congestion avoidance* and *congestion recovery* mechanisms.

3.2.35 congestion management: [ITU I.233.1 A.10] This includes *network* engineering, OAM procedures to detect the onset of congestion, and real-time mechanisms to prevent or recover from congestion. Congestion management includes, but is not limited to, *congestion control*, *congestion avoidance*, and *congestion recovery*.

3.2.36 congestion recovery: [ITU I.233.1 A.12 (truncated)] Procedures initiated to prevent congestion from severely degrading the end user perceived *quality of service*(s) delivered by the *network*.

3.2.37 congruent ringlets: *Ringlets* that share the same set of *stations*, but a distinct set of *links*, such that the order of station traversal via the links is identical or is exactly reversed.

3.2.38 control frame: A *frame* carrying only *MAC sublayer* control information.

3.2.39 control latency: Interval between the time that a *control frame* is sent from a *station* and the time that the effect of that *control frame* is observable at the issuing *station*.

3.2.40 conversation: [IEEE 100 (C/LM) 802.3 ad-2000] A set of *MAC frames transmitted* from one *end station* to another, where all of the *MAC frames* form an ordered sequence, and where the communicating *end stations* require the ordering to be maintained among the set of *MAC frames* exchanged.

3.2.41 copy (copying): Replication of an *inbound frame* by the *MAC sublayer* (independent of whether or not the *frame* is *stripped*).

3.2.42 customer separation: The property that data associated with one group of network users (e.g. a customer organization) is not communicated to a different group of *network* users.

3.2.43 <cut-thru: The *passthru* of a *frame* through a *station* such that the first bit of the *frame* is *retransmitted* before the last bit is *received*[4].>

3.2.44 cyclic redundancy check (CRC): A form of error check used to ensure the accuracy of *transmitting* a message. Note: The *CRC* is the result of a calculation carried out on the set of *transmitted bits* by the *transmitter*. The *CRC* is encoded into the *transmitted signal* with the data. At the *receiver*, the calculation creating the *CRC* may be repeated, and the result compared to that encoded in the signal.

3.2.45 data delivery ratio (DDR): [FRF.13 section 5 (modified)] Reports the network's effectiveness in transporting offered data. The DDR is a ratio of successful payload octets received to attempted payload octets transmitted.

3.2.46 data frame: A *MAC frame* carrying data supplied by the *MAC client*.

3.2.47 data-rate: The rate at which information is transferred, measured in bits-per-second.

3.2.48 <data-stream (stream): [(C) 610.10-1994 (modified)] A continuous stream of data elements being *transmitted*.>

3.2.49 delivered duplicated frames: [ITU I.233.1 A.15 (modified)] A *frame received* at a *destination* such that the frame was not generated by the *source station* identified by the *source address* and the *frame* is exactly the same as a *frame* that was previously delivered to that *destination*.

3.2.50 delivered errored frames: [ITU I.233.1 A.14 (modified)] The number of *frames* for which the value of one or more of the bits in the *frame* is in error, or when some, but not all, bits in the *frame* are lost bits or extra bits (i.e. bits that were not present in the original signal).

3.2.51 delivered out-of-sequence frames: [ITU I.233.1 A.16 (modified)] A frame (Ft) arriving at a *destination station* after a *frame* F_{t+1}, F_{t+2}, F_{t+3}…., F_n in a sequence of *frames* F_1, F_2, F_3, …. , F_n sent from a *source station*.

3.2.52 destination station (destination): [from IEEE 100 (C/BA) 1355-1995 (modified)] The *station(s)* on a network that is(are) the intended recipient(s) of an *802.17 frame*.

3.2.53 discard eligibility (DE): [FRF glossary (modified)] A bit indicating that a *frame* may be discarded in preference to other frames if congestion occurs, in order to maintain the committed *quality of service* within the *network*.

[4] LAN switches typically perform cut-thru of 802.3 frames after reception of the destination *MAC address* (first six bytes of the *frame*). LAN switch cut-thru is not described in 802.1D but is left as a device specific feature. The frame check is not performed for frames that are cut-thru.

3.2.54 downstream: The direction of data flow.

3.2.55 dual-ring: A *ring* composed of exactly two *congruent ringlets* having opposite orientations.

3.2.56 <dynamic bandwidth allocation: Candidate definition should be proposed. The T&D Ad Hoc noted that some people think of this as a synonym for fairness.>

3.2.57 <effective transfer rate: [ISO/IEC2382-09 9.05.22 (modified)] The number of bytes *transferred* between two points per unit time and accepted as valid at the *destination*.>

3.2.58 egress queue delay: TBD by Performance Ad Hoc

3.2.59 egress stripping: The removal of *frames* by the *egress station*.

3.2.60 egress: The direction towards the *MAC client* from the *ring* or *MAC sublayer*.

3.2.61 encapsulation: A process by which an entity places a *header* and, optionally, a *trailer* on an *SDU*.

3.2.62 <encapsulating bridge[5]**:**> (TBD)

3.2.63 end station: [802.3-2000 1.4.111 (modified)] A station attached to a *network* that is an initial source or a final destination of MAC frames transmitted across that *network*. A network layer router is, from the perspective of the *LAN*, an *end station*; a *MAC Bridge*, in its role of *forwarding MAC frames* from one *LAN* to another, is not an *end station*

3.2.64 excess burst size (Be): [ITU I.233.1 A.6 (modified)] The maximum amount of data by which a user can exceed *Bc* during a time interval *Tc*. This data (*Be*) is delivered in general with a lower probability than *Bc*.

3.2.65 fairness: The assignment of *ring ingress rates* such that available *capacity* is shared according to a specified algorithm.

3.2.66 flow control: A *congestion control* mechanism allowing one *station* to communicate to another *station* the information that *frame transmission* should be reduced or halted in order to avoid *buffer* overrun, or other conditions associated with congestion, at the *receiving station* and allowing the resumption of normal levels *frame transmission* when the condition is resolved.

3.2.67 flow: The collection of *frames* associated with a *conversation* that can be identified by one or a combination of specific values carried in the *protocol headers* at the *MAC layer* or above.

3.2.68 frame check sequence (FCS): [IEEE 100 (C/LM) 802.12-1995] *A Cyclic Redundancy Check (CRC)* used by the *transmit* and *receive* algorithms to detect errors in the bit sequence of a *MAC frame*

3.2.69 frame delivery ratio (FTR): [FRF.13 section 4] The ratio of successful *frame receptions* to attempted *frame transmissions*.

3.2.70 frame transfer delay (FTD): [FRF.13 section 3] The difference in milliseconds between the time a *frame* exits a *source station* and the time the same *frame* enters the *destination station*.

3.2.71 frame transmission time: TBD by Performance Ad Hoc

3.2.72 frame: (see *MAC frame*)

[5] Bob Castellano to investigate.

3.2.73 global fairness: TBD by working group.

3.2.74 global spatial reuse: The utilization of *ring capacity* by *stations* on the *ring* when the *station* to which the *capacity* is assigned does not utilize that *capacity*.

3.2.75 group address: [ISO/IEC2382-25 25.01.15] An *address* that identifies a group of *stations* on a *network*.

3.2.76 guaranteed-service (GS): Service that assures conformance to specific QoS parameter values.

3.2.77 IEEE Std. 802.17 (802.17): The IEEE *resilient packet ring* standard.

3.2.78 inbound: The direction of *frame* arrival at a *station* from a *ringlet*.

3.2.79 individual address: [ISO/IEC2382-25 25.01.14] An *address* that identifies a particular *station* on a *network*.

3.2.80 ingress queue delay: TBD by Performance Ad Hoc

3.2.81 ingress rate control: *Rate control* performed at the *ring ingress*.

3.2.82 ingress stripping: The removal of *frames* by the *ingress station*.

3.2.83 ingress: The direction from the *MAC client* towards the *ring* or *MAC sublayer*.

3.2.84 insert[6] (insertion): The placement of an *ingress frame* on the *ring* by a *station*.

3.2.85 interconnected rings: Non-*congruent rings* that intersect at one or more *stations*.

3.2.86 jitter: Variation in *delay* associated with the *transfer* of *frames* from one point in the *network* to another.

3.2.87 latency: The time required for information to be *transferred* between two points. Synonymous with *delay* for purposes of the 802.17 specification.

3.2.88 link aggregation group[7]: [IEEE 802.3-2000 1.4.154 (modified)] A group of *links* that appear to a *MAC client* as if they were a single *link*. All *links* in *a link aggregation group* connect the same pair of aggregation systems. One or more *conversations* may be associated with each *link* that is part of a *link aggregation group*.

3.2.89 link partner: The device at the opposite end of a *link* from the local *station*.

3.2.90 link: [IEEE 100 (C/LM) 802.5c-1991] A unidirectional physical and media connection between two *stations*.

[6] (C/LM) 11802-4-1994 is not applicable. That definition uses the term to specify device, rather than frame, insertion in the ring..
[7] 802.17 link aggregation is not necessarily identical to that specified by 802.1ae.

3.2.91 local area network (LAN): [adapted from IEEE 100 (C/DIS) 1278.2-1995, 1278.3-1996[8]] A communications network designed for a user premises, typically not exceeding a few kilometers in length, and characterized by moderate to high data transmission rates, low delay, and low bit error rates.

3.2.92 local fairness: TBD by working group.

3.2.93 local spatial reuse: The utilization of common *ring capacity* by *stations* communicating across non-overlapping *segments* of the *ring*.

3.2.94 logical link control (LLC) sublayer: [C/LM 8802-5-1992s] That part of the *data link layer* that supports *media* independent *data link* functions, and uses the services of the *MAC sublayer* to provide services to the *network layer*.

3.2.95 lost frames: [ITU I.233.1 A.17 (modified)] A *frame* not delivered to the intended *destination* user within a specified time-out period, and the *network* is responsible for the non-delivery.

3.2.96 MAC client: The *protocol layer* (or *sublayer*) immediately above the *MAC sublayer*. Generally, the *network layer* or *logical link control (LLC) sublayer*.

3.2.97 MAC end-to-end delay: TBD by Performance Ad Hoc

3.2.98 MAC frame (frame): [IEEE 100 (C/LM) 802.12-1995] The logical organization of control and data fields (e.g., addresses, data, error check sequences) defined for the *MAC sublayer*[9]. Note: The term *frame* can be prefixed with an orientation (*ingress*, *egress*, *inbound*, *outbound*) or an operation (*inserted*, *copied*, *stripped*, *passedthru*).

3.2.99 management information base (MIB): [802.3-2000 1.4.163] A repository of information to describe the operation of a specific network device.

3.2.100 maximum frame size (MFS): The maximum number of bytes in a *frame*.

3.2.101 maximum transfer unit (MTU): [IEEE 100 610.7-1995 (modified)] The largest *payload* that can be *transferred* across a given *physical network* in a single *frame*.

3.2.102 medium: See *transmission medium*.

3.2.103 medium access control (MAC) sublayer: (1) [IEEE 100 (C/LM) 8802-5-1995] The portion of the *data link sublayer* that controls and mediates the access to the *ring*. **(2)** [802.3-2000 1.4.167] The *data link sublayer* that is responsible for *transferring data* to and from the *physical layer*[10]. **(3)** [ISO/IEC 15802-1] The MAC service provider.

3.2.104 medium access delay: TBD by Performance Ad Hoc

3.2.105 <medium[11] agnostic: Denotes a *MAC sublayer* that can operate with arbitrary *physical layer* alternatives, requiring a *reconciliation sublayer* for each specific *PHY* type supported.>

[8] 'moderate sized geographic area' replaced by 'user premises, typically not exceeding a few kilometers in length,'

[9] Omitted sentence: "The MAC frame may be constructed in either ISO/IEC 8802-3 or ISO/IEC 8802-5 format.

[10] This appears as a definition of medium access control, but it is clearly a definition of medium access control sublayer.

[11] Standards documents are split on whether this is medi*a* or medi*um*. Medium, the singular, seems more appropriate and is used here.

3.2.106 <medium interface connector (MIC): [802.5-1998 1.3.36 (modified)] A connector interface at which signal transmit and receive characteristics are specified for attaching stations.>

3.2.107 metropolitan area network (MAN): [IEEE 100 (C/LM) 8802-6-1994] A *network* for connecting a group of individual *stations* and *networks* [for example, *local area networks (LANs)*] located in the same urban area. Note: A *MAN* generally operates at a higher speed than the networks interconnected, crosses network administrative boundaries, may be subject to some form of regulation, and supports several access methods.

3.2.108 misdelivered frames: [ITU I.233.1 A.16 (modified)] A *frame transferred* from a *source* to a *destination* user other than the intended *destination* user. It is considered inconsequential whether the information is correct or incorrect in content.

3.2.109 <MTU transparency: The ability to *passthru frames* without regard to *MTU* size.>

3.2.110 multicast address: [ISO/IEC2382-25 25.01.16]. A *group address* that identifies a subset of the *stations* on a *network.*

3.2.111 multicast: The act of sending a *frame* addressed to a group of *stations*.

3.2.112 multi-ring: A *ring* composed of multiple *congruent ringlets*, at least two of which are *opposing ringlets*.

3.2.113 neighbor: [BH] A *station* that is exactly one *link* away from a given *station*.

3.2.114 network control host: [802.3-2000 1.4.176] A network management central control center that is used to configure *agents*, communicate with *agents*, and display information collected from *agents*.

3.2.115 network: A generic designation for a bridged LAN, MAN, RAN, or WAN.

3.2.116 opposing ringlet: A *ringlet* whose traffic circulates in the direction opposite that of a given *ringlet*.

3.2.117 outbound: The direction of *frame* departure from a *station* to a *ringlet*.

3.2.118 packet: A *frame* to which has been added those fields that are medium dependent[12].

3.2.119 <packetization delay: TBD by Performance Ad Hoc>

3.2.120 partition: One set of communicating *stations* on a *partitioned ring*.

3.2.121 partitioned ring: A *ring* having two or more points of failure resulting in two or more non-communicating sets of *stations*.

3.2.122 passthru buffer delay: TBD by Performance Ad Hoc

3.2.123 passthru delay[13]: TBD by Performance Ad Hoc

3.2.124 passthru queuing: A method of *passthru* in which *passthru traffic* is queued to allow *passthru* or *insertion* of *traffic* of higher priority and to allow a *transmission* in progress to complete.

[12] This is the IEEE view of a packet. It is entirely different from the IEEE view of a packet as an L3 PDU.
[13] The term *station* is used to qualify the term *transit-delay* since the term *transit-delay* is used by *frame relay* to indicate *end-to-end transit delay*.

3.2.125 passthru: The passing of a *frame* through a *station* via the *ring*[14].

3.2.126 path: The specific sequence of *stations* and *links* traversed by a *frame* in *passthru* between two *stations*.

3.2.127 <pause: [802.3-2000 1.4.209 (modified)] A mechanism associated with the *IEEE 802.3 MAC* specification for providing full duplex *flow control*.>

3.2.128 <payload agnostic: Denotes a *MAC sublayer* that is not sensitive to the contents of the *payload transferred* to/from the *MAC client*.>

3.2.129 physical layer (PHY): [(C/LM) 8802-5-1995] The layer responsible for interfacing with the *transmission medium*. This includes conditioning signals received from the *MAC* for *transmitting* to the *medium* and processing *signals received* from the *medium* for sending to the *MAC*.

3.2.130 plug-and-play: The requirement that a *station* perform *passthru*, *strip*, and *ring* control activities without manual intervention except for what may be needed for connection to the *ring*. The station may additionally *copy* and *insert frames*.

3.2.131 <port: (1) The point of *ingress* for *inbound frames* and the point of *egress* for *outbound frames* with respect to the *data station*. **(2)** [IEEE 100 (C/LM) 802.1G-1996, 8802-5-1995 (modified)] A signal interface provided by stations that is generally terminated at a *medium interface connector (MIC)*.[15]>

3.2.132 <preemption: The interruption of a *frame* in *transmission* for the purpose of *transmitting* a *frame* of higher priority.>

3.2.133 propagation delay: TBD by Performance Ad Hoc

3.2.134 <protocol agnostic: Denotes a *MAC sublayer* that can operate with arbitrary *upper-layer protocol* alternatives.>

3.2.135 protocol data unit (PDU): [802.5-1998 1.3.46] Information delivered as a unit between peer entities that contains control information and, optionally, data.

3.2.136 protocol implementation conformance statement (PICS): 1.3.47: A statement of which capabilities and options have been implemented for a given Open Systems Interconnection (OSI) protocol.

3.2.137 <protocol stack delay: TBD by Performance Ad Hoc>

3.2.138 <QTag prefix: [802.3-2000 1.4.222] The first four octets of an Ethernet-encoded Tag Header. The Ethernet-encoded Tag Header is defined in IEEE P802.1Q.>

3.2.139 quality of service (QoS): One or a combination of measurable properties (parameters) defining the requirements of a given data service.

3.2.140 rate control: Limitation of the *traffic rate* in bytes over a specified time interval.

3.2.141 receive (receipt, reception): The action of a station taking a frame from the medium.

[14] Includes the case of wrapping, if supported.
[15] Removed 'Ports may or may not provide physical containment of channels' What, exactly, does this mean?

3.2.142 reconciliation sublayer (RS): [adapted from IEEE 100 (C/LM) 802.3 –1998 modified] A mapping function that reconciles the *signals* at the media independent interface (MII) to the *media access control (MAC)* – physical signaling sublayer (PLS) service definitions.

3.2.143 regional area network (RAN): (1) A *network* for connecting a group of individual *stations* and *networks* [for example, *metropolitan area networks (MANs)*] located in multiple contiguous urban areas. (2) A *MAN* spanning multiple urban areas.

3.2.144 <residual error rate: [ITU I.233.1 A.13 (modified[16])] As applied to *MAC layer* service: (1 – (total correct MAC SDUs delivered)/(total offered MAC SDUs)).>

3.2.145 resilient packet ring (RPR): (1) A *connectionless* ring-based *MAC protocol* as defined by IEEE 802.17, appropriate for *LAN*, *MAN*, or *RAN* [17]deployment[18]. (2) A collection of *stations* conforming to the *resilient packet ring protocol*, and the links forming the *ring*.

3.2.146 ring end-to-end delay: TBD by Performance Ad Hoc

3.2.147 ring latency: TBD by Performance Ad Hoc

3.2.148 ring medium: The abstraction of a *ring* as a continuous closed path *transmission medium*.

3.2.149 ring segment (segment): The portion of a *ring* bounded by two *stations* interconnected by one or more *links*.

3.2.150 ring topology (topology): [IEEE 100 610.7-1995 (modified)] The logical and/or physical arrangement of *stations* on a *ring*.

3.2.151 ring: (1) The collection of *stations* and *links* forming a *resilient packet ring*. **(2)** The set of *congruent ringlets* forming a *resilient packet ring*.

3.2.152 ringlet: A closed unidirectional *path* formed by an ordered set of *stations*, and the *links* interconnecting *stations*, such that each *station* has exactly one *link* entering the *station* and one *link* exiting the *station*.

3.2.153 <round trip propagation time: > TBD by Performance Ad Hoc

3.2.154 segment: (see *ring segment*)

3.2.155 service data unit (SDU): [802.5-1998 1.3.59] Information delivered as a unit between adjacent entities that may also contain a PDU of the *upper layer*.

3.2.156 service level agreement (SLA): Contract between a *network* service provider and a customer that specifies, in measurable terms, what services the *network* service provider will furnish.

3.2.157 shared access: The capability of two or more *stations* to share the *capacity* of the *ring medium*[19].

3.2.158 simple-fairness: A class of *fairness algorithm* that assigns *equal* shares of *ring capacity*.

[16] Need help with this one.
[17] Why did we exclude WAN?
[18] Or should this be specifically the protocol standardized by IEEE802?
[19] The definition of ring latency in ISO/IEC 2382-25 25.04.03 suggests that the ring is modeled as a shared medium even if it is not a continuous physical medium.

3.2.159 simultaneous access: The *insertion* of *traffic* onto the *ring medium* by two or more *stations* at the same instant in time.

3.2.160 source station (source): The *station* that originates an 802.17 MAC *frame* with respect to a *network.*

3.2.161 spatial reuse: The utilization of *ring capacity* by a *station* different from the *station* to which the *capacity* was nominally assigned.

3.2.162 <spatial reuse protocol (SRP): A protocol, described in IETF informational RFC 2892, August 2000.>

3.2.163 station (data station): [IEEE 100 1073.3.1-1994, 1073.4.1-1994, 8802-5-1995 (modified)] A device that may be attached to a *network* for the purpose of *transmitting* and *receiving* information on that *network.*

3.2.164 station management (SMT): [802.5-1998 1.3.66] The conceptual control element of a station that interfaces with all of the layers of the station and is responsible for the setting and resetting of control parameters, obtaining reports of error conditions, and determining if the station should be connected to or disconnected from the medium.

3.2.165 station passthru delay: TBD by Performance Ad Hoc

3.2.166 steering: The *transmission* of a *frame* on a specific *ringlet* at the *ingress station* based on knowledge of the *ring topology*.

3.2.167 store-and-forward: A method of *passthru* such that all bits of the *frame* are *received* and *buffered* before re*transmission* begins[20].

3.2.168 <stream: (see *data-stream*)>

3.2.169 strip (stripping): The removal of a *frame* from the *ring*.

3.2.170 <tagged MAC frame: [802.3-2000 1.4.269] A frame that contains a QTag Prefix.>

3.2.171 throttle: The sending of a *control frame* to a specific *station*, to stop or slow the flow of *data traffic*.

3.2.172 throughput: [ITU I.233.1 A.1(modified)] The number of data bits contained in the MAC frame payload successfully transferred from *source station* to *destination station* per unit time. A frame is successfully transferred if the *FCS* check for the frame is satisfied.

3.2.173 time-to-live (TTL): Value carried in the *protocol header* of a *frame* in order to allow the stripping of a *frame* that has *passedthru* a sufficient number of stations. The *TTL* value is generally set to an initial value at the source and decremented at each subsequent hop. The *frame* is *stripped* when the *TTL* value reaches zero.

3.2.174 topology: (see *ring topology*)

3.2.175 traffic class: A grouping of traffic that is to be processed by a distinct set of rules.

[20] The definition that appears in 09.07.13 ISO/IEC 2382-9 1995, 'A mode of operation of a *data network* in which data are temporarily stored before they are *retransmitted* toward the *destination*', is ambiguous, as it is not clear whether 'data' refers to a complete *frame* or some portion of a *frame*.

3.2.176 train: A collection of two or more contiguous *frames* on the ring.

3.2.177 transfer: [ISO/IEC2382-09 9.03.01 (modified)] The movement of an *SDU* from one layer to an adjacent layer. Also used generally to refer to any movement of information from one point to another.

3.2.178 <transfer rate: [ISO/IEC2382-09 9.05.21] The number of bytes *transferred* per unit time between two points.>

3.2.179 transmission channel (channel): [ISO/IEC2382-9 09.02.14] A means of transmission of a signal in one direction between two stations where the signals are physically or logically isolated from the signals in other channels.

3.2.180 transmission medium (medium): [IEEE 100 (C/LM) 8802-6-1994, 802.5-1998 1.3.34] The material on which *information signals* may be carried; e.g., optical fiber, coaxial cable, and twisted-wire pairs.

3.2.181 transmission: (see *transmit*)

3.2.182 transmit (transmission): [(C/LM) 802.5-1989s, 8802-5-1995 (modified)] The action of a *station* placing a *frame* on the *medium*.

3.2.183 transparent bridging: [(C/LM) 8802-5-1995] A *bridging* mechanism in a *bridged network* that is transparent to the *end stations*.

3.2.184 unicast: The act of sending a *frame* addressed to a single *station*.

3.2.185 unknown unicast: The act of sending a *frame* addressed to a single *station*, where the location in the *network* is unknown.

3.2.186 upper-layers: The collection of *protocol layers* above the *data-link layer*.

3.2.187 upstream: The direction opposite that of the *downstream* direction.

3.2.188 <upstream neighbor's address (UNA): [802.5-1998 1.3.77 (modified)] The address of the *station* immediately *upstream* from a given *station*.>

3.2.189 verified frame: [802.5-1998 1.3.79 (modified)] A valid *frame* addressed to the *station*, for which the information field has met the validity check.

3.2.190 virtual LAN (VLAN): [IEEE 100 (C/LM 802.1Q-1998)] A subset of the active *topology* of a *bridged local area network*. Associated with each *VLAN* is a *VLAN Identifier (VID).*

3.2.191 virtual medium (VMedium): A logical partition of the *network* intended to provide *customer separation*.

3.2.192 weighted-fairness: A class of *fairness algorithm* that allows the assignment of unequal shares of *ring capacity*.

3.2.193 wide area network (WAN): [IEEE 100 (C/DIS) 1278.2-1995] A communications network designed for large geographic areas. Sometimes called *long-haul* network.

3.2.194 wrapping: In the case of a *dual ring*, the *transmission* of a *frame* on the ringlet opposing the ringlet on which it was received.

3.3 Acronyms and abbreviations

The acronyms for 802.17 are controlled by a separate terms-and-definitions draft (T&D) and subject to change. A recent version of the T&D draft has been included in this proposal for the benefit of the author and readers .

802.17: IEEE Std 802.17

Bc: committed burst size

Be: excess burst size

BER: Bit error ratio

BES: Best effort service

BIR: buffer insertion ring

CIR: committed information rate

CoS: class of service

CRC: cyclic redundancy check

CUG: Closed user group

DDR: data delivery ratio

DE: discard eligibility

FCS: frame check sequence

FTD: frame transfer delay

FTR: frame delivery ratio

GS: guaranteed-service

LAN: local area network

LLC: logical link control

MAC: medium access control

MAN: metropolitan area network

MFS: maximum frame size

MIB: management information base

MTU: maximum transfer unit

PDU: protocol data unit

PHY: physical layer

PICS: protocol implementation conformance statement

QoS: quality of service

RAN: regional area network

RPR: Resilient Packet Ring

RS: reconciliation sublayer

SDU: service data unit

SLA: service level agreement

SMT: station management

Tc: committed rate measurement interval

TTL: time-to-live

VLAN: virtual LAN

VMedium: virtual medium

WAN: wide area network

3.4 Numerical values

Decimal, hexadecimal, and binary numbers are used within this document. For clarity, decimal numbers are generally used to represent counts, hexadecimal numbers are used to represent addresses, and binary numbers are used to describe bit patterns within binary fields.

Decimal numbers are represented in their usual 0, 1, 2, ... format. Hexadecimal numbers are represented by a string of one or more hexadecimal (0-9,A-F) digits followed by the subscript 16, except in C-code contexts, where they are written as `0x123EF2` etc. Binary numbers are represented by a string of one or more binary (0,1) digits, followed by the subscript 2. Thus the decimal number "26" may also be represented as "$1A_{16}$" or "11010_2".

3.5 Field names

This document describes values that are packetized or located in MAC-resident registers. For clarity, names of these values have an italic font and contain the context as well as field names, as illustrated in Table 3.1.

Table 3.1—Names of registers and fields

Name	Description
thisState.levelAB	A register within the MAC
informState.accounts[n].rateB.c	A congestion indication transported within a packet

Note that run-together names (like "*thisState*") are preferred because they are more compact than under-score-separated names (like "*this_state*"). The use of multiword names with spaces (like "This State" is avoided, to avoid confusion between commonly used capitalized key words and the capitalized word used at the start of each sentence.

3.6 Bit numbering and ordering

Data transfer sequences normally involve one or more cycles, where the number of bytes transmitted in each cycle depends on the number of byte lanes within the interconnecting link. Data byte sequences are illustrated as 4-byte groups, as illustrated in Figure 3.1. For multibyte objects, the first-through-last data bytes are the most-through-least significant respectively.

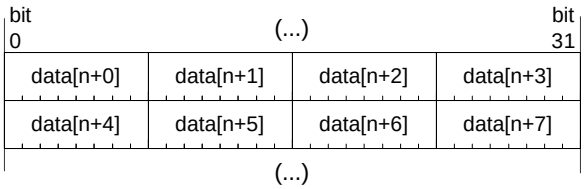


Figure 3.1—Byte and bit ordering

The data-byte transmission order is left-to-right within each cycle and top-to-bottom between cycles, as is consistent with the flow of English language documentation. For consistency, bits and bytes are numbered in the same fashion.

3.7 C code notation

The behavior of data-transfer command execution is frequently specified by C code, such as Equation 3.1. To differentiate such code from textual descriptions, such C code listings are formatted using a fixed-width Courier font. Similar C-code segments are included within some figures.

```
// Return maximum of a and b values
Max(a,b) {
  if (a<b)
    return(LT);
  if (a>b)
    return(GT);
  return(EQ);
}
```

3.1

Since the meaning of many C code operators are not obvious to the casual reader, their meanings are summarized in Table 3.2.

Table 3.2—C code expressions

Expression	Description
`~i`	Bitwise complement of integer *i*
`i^j`	Bitwise EXOR of integers *i* and *j*
`i&j`	Bitwise AND of integers *i* and *j*
`i<<j`	Left shift of bits in *i* by value of *j*
`i*j`	Arithmetic bmultiplication of integers *i* and *j*
`!i`	Logical negation of Boolean value *i*
`i&&j`	Logical AND of Boolean *i* and *j* values
`i||j`	Logical OR of Boolean *i* and *j* values
`i^= j`	Equivalent to: `i= i^j`.
`i==j`	Equality test, true if *i* equals *j*
`i!=j`	Equality test, true if *i* does not equal *j*
`i<j`	Inequality test, true if *i* is less than *j*
`i>j`	Inequality test, true if *i* is greater than *j*

4. Media access control (MAC) service specification

4.1 Scope

This clause specifies the services provided by the MAC sublayer and the MAC Control sublayer to the client of the MAC (see Figure 2.1). MAC clients may include the Logical Link Control (LLC) sublayer, Bridge Relay Entity, or other users of ISO/IEC LAN International Standard MAC services (see Figure 2.2). The services are described in an abstract way and do not imply any particular implementations any exposed interface. There is not necessarily a one-to-one correspondence between the primitives and the formal procedures and interfaces.

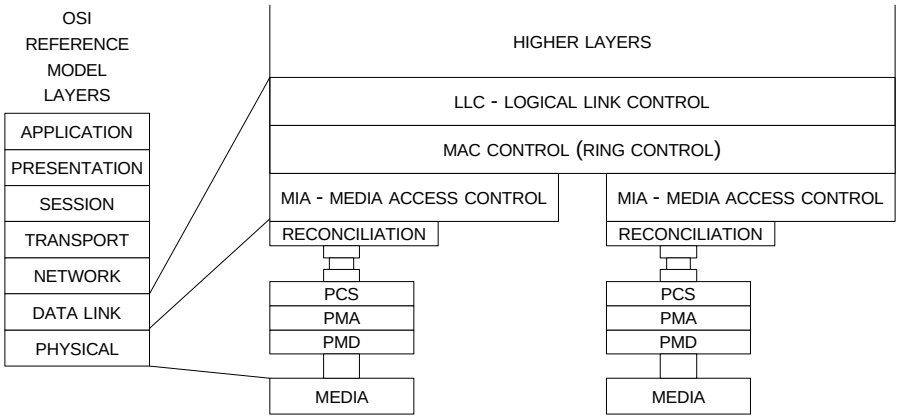


Figure 4.1—Service specification relation to the LAN model

4.2 Overview of MAC services

4.2.1 Class-A service

The MAC provides a class-A service with guaranteed bandwidth and low jitter specifications. This class is intended to allow the client to implement a synchronous traffic class. The MAC is responsible for policing class-A traffic to ensure that provisioned service parameters are not violated; therefore class-A traffic need not be shaped by the client.

The MAC provides mechanisms for provisioning class-A traffic (see clause 6) and ensures that the provisioned bandwidths never exceed link capacities. Since the levels of sustainable class-A traffic are limited, requests for class-A allotments may sometimes be rejected. The forces release of other provisioned bandwidths, to avoid continued bandwidth-request rejections, is beyond the scope of this RPR specification

The service access point provides an indication to the MAC client of the status of the underlying channel, indicating where there is dynamic backpressure from the media and traffic over this path cannot currently be accepted.

4.2.2 Class-B service

The MAC provides a class-B service with guaranteed bandwidth and bounded delays specifications. This class is intended to allow the client to implement a guaranteed traffic class (GTC). As is true for class-A

traffic, the MAC is responsible for policing class-B traffic to ensure that provisioned service parameters are not violated; therefore class-B traffic need not be shaped by the client.

The service access point provides an indication to the MAC client of the status of the underlying channel, indicating where there is dynamic backpressure from the media and traffic over this path cannot currently be accepted.

4.2.3 Class-C service

The class-C service is provided to implement a best effort traffic class. The class-C traffic passes through the lower-priority transmit-path FIFO so that, once accepted, the bounded delays for class-B and class-C traffic are the same. The MAC is responsible for enforcing weighted fairness, therefore class-C traffic need not be shaped by the client. The allocation of fairness weights is beyond the scope of this specification.

The service access point provides an indication to the MAC client of the status of the underlying channel, indicating where there is dynamic backpressure from the media and traffic over this path cannot currently be accepted.

5. Packet formats

5.1 Packet framing

The physical layer is expected to provide first-byte and final-byte framing of packets, as illustrated in Figure 5.2.

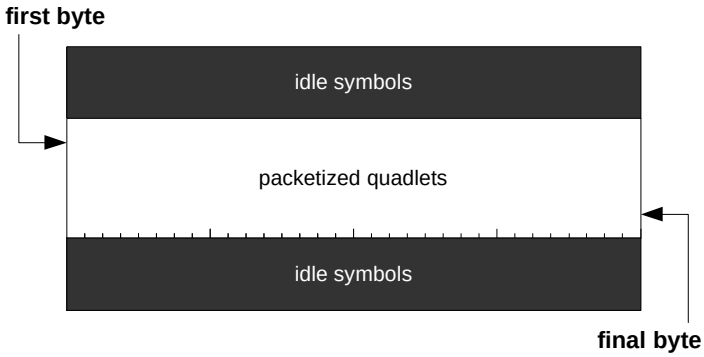


Figure 5.1—Packet framing

The physical layer may also provide other services, including the following:

1) Rate matching. Insertion and/or deletion of between-packet symbols, as necessary to compensate for drifts between transmit and receive clocks. MAC-level support (see D.1) is possible when this capability is not supplied by the PHY.
2) Timer synchronization. Primitives for maintaining accurate clock synchronization between attached clock-master and clock-slave station. MAC-level support (see D.2) is possible when this capability is not supplied by the PHY.
3) Fault monitoring. Primitives for maintaining accurate clock synchronization between attached clock-master and clock-slave station.

5.2 Frame formats

5.2.1 Compact and complete frames

A frame consists of header and payload components, both of which are CRC protected, as illustrated in Figure 5.2. A bit in the initial leader differentiates between the basic header (illustrated on the left) and extended header (illustrated on the right) formats.

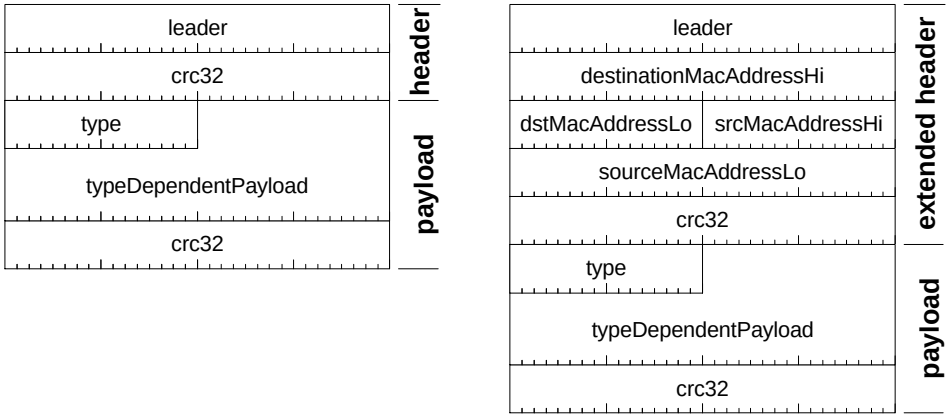


Figure 5.2—Frame formats

For both formats, the format of the following payload is implied by the first 16-bit *type* field within the payload. For the extended header: the 48-bit *destinationMacAddress* is the concatenation of the 32-bit ***destinationMacAddressHi*** and 16-bit ***dstAddressLo*** fields; the 48-bit *sourceMacAddress* is the concatenation of the 16-bit ***srcMacAddressHi*** and 32-bit ***sourceAddressLo*** fields.

5.2.2 Leader format

The frame leader provides 7-bit *targetStationID* and *sourceStationID* identifiers, as illustrated in Figure 5.3. The least-significant bit of each identifier differentiates between attachments on ring0 and ring1, for values of 0 and 1 respectively.

Figure 5.3—Packet leader format

The ***wrap*** bit values of 0 and 1 indicate the packet shall be discarded or sustained at wrap points..

The 7-bit ***destinationStationID*** field identifies the destination RPR station that strips the packet. The all-ones *targetStation* field has a special meanings; the frame shall be stripped at the next station. This address is typically used to send control messages (such as discovery frames) between stations.

The ***ring*** bit values of 0 and 1 indicate the packet was sourced on ring0 and ring1 respectively.

The 7-bit ***sourceStationID*** field identifies the source RPR station, typically for bridge-routing purposes. The all-ones *sourceStation* field also has a special meanings; it indicates the frame was not generated by a specific station, but represents cumulative contributions from multiple stations

The ***flood*** bit values of 1 indicates the station corresponding to the frame's *destinationMacAddress* field is unknown; bridges typically flood this frame to remote stations.

The ***macs*** bit values of 0 and 1 distinguish between basic and extended header formats (see 5.2.1).

The 2-bit ***class*** field values specify the class of RPR traffic, as specified in Table 5.1. The CLASS_A0 and CLASS_A1 values identify proactive class-A traffic, when passing from destination-to-source and source-to-destination respectively. The CLASS_A label identifies reactive class-A traffic; the CLASS_BC identifies lower-class CLASS_B and CLASS_C traffic.

Table 5.1—class field values

Value	Name	Description
0	CLASS_A0	Proactive class-A traffic
1	CLASS_A1	Proactive class-A residue
2	CLASS_A	Reactive class-A traffic
3	CLASS_BC	All class-B and class-C traffic

The 4-bit ***depthBC*** field communicates the class-A congestion associated with the opposing run, where values of 0 through 15 correspond to empty through nearly-full *passBC* FIFO-depth conditions respectively (see 7.3 for details).

The 8-bit ***timeToLive*** field is decremented when packets pass through stations, marking stale packets to facilitate their timely demise.

The header (like the payload) is followed by a 32-bit ***crc32*** value, which protects the aforementioned header parameters. A standard 32-bit CRC protocol is specified; see Annex E for details.

6. Bandwidth provisioning

Bandwidth provisioning involves negotiation for guaranteed transmission bandwidths, over specified source-to-destination paths, such that the cumulative provisioned bandwidths remains below the capacity of any link located between the source and destination.

Bandwidth provisioning is parameterized by the class of traffic being partitioned, with the constraint that the cumulative class-A and class-B traffic never exceeds the bandwidth guaranteed by the link:

6.1 Consistent bandwidth provisioning

6.1.1 Bandwidth accounts

Provisioned communication between source and destination stations requires allocation of link-bandwidth resources affiliated with one or more intermediate hops, as illustrated in Figure 6.1. This provisioning is performed in a distributed fashion: each station has provisioned-bandwidth accounts (one entry for each distance) and special survey messages are provided for providing per-link provisioned-bandwidth summaries.

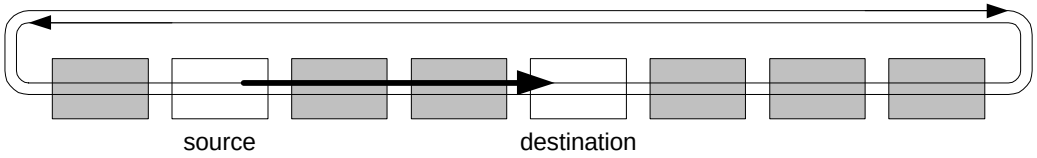


Figure 6.1—Provisioned-bandwidth segments

Each station keeps accounts of its provisioned class-A and class-B resource allocations, on a per-hop basis. This requires an array of storage entries, where each *entry[n]* specifies the bandwidth provisioned for communication through *n* stations. Each entry consists of two values, corresponding the fractional link bandwidth provisioned for class-A and class-B traffic, as illustrated in Figure 6.2.

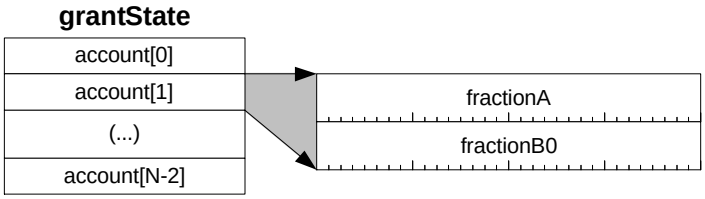


Figure 6.2—Provisioned accounts

The 32-bit ***fractionA*** value is the amount of class-A traffic provisioned to this path. The 32-bit ***fractionB0*** value is the amount of class-B0 traffic provisioned over this path.

The provisioned bandwidth numbers decrease in a monotonic fashion: the value of *account[n+1].fractionA* is equal-to or less than *account[n].fractionA*. This monotonic relationship is based on the pipelined nature of the traffic: all traffic passing through *n+i* stations also passes through *n* hops.

6.1.2 Bandwidth surveys

Each station is responsible for updating its provisioned-bandwidths accounts. These accounts can be reduced without conferring with others. However, these accounts cannot be increased without a bandwidth-survey, to verify availability of the desired bandwidths. A survey of bandwidth accounts (by *station[1]*, for example) involves sending of a bandwidth survey message through others, as illustrated in Figure 6.3.

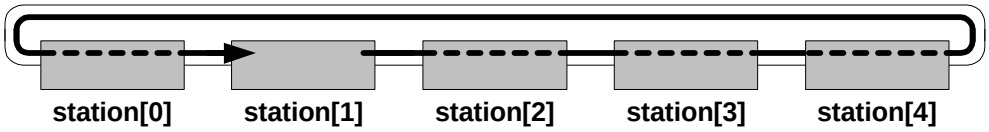


Figure 6.3—Bandwidth surveys

Rather than checking for availability of specific links and bandwidths, the bandwidth-check message determines the available bandwidth on a link-by-link basis, allowing the requester to make the most intelligent decisions on how that bandwidth should be allocated among multiple (possibly prioritized) subclients.

Multiple requesters could attempt to simultaneously sample and allocate additional bandwidths. To avoid the data inconsistencies that could be caused by such conflicts, these provisioned-bandwidth messages are serialized. Conflict resolution and serialization are handled by the same mechanism: conflicts are always resolved in favor of the requester with the highest MAC address.

The precedence of provisioning messages is based on the need to resolve circular conflicts, such as multiple stations generating messages concurrently. The intent is to break the circular deadlock by assigning asymmetric precedence values. Higher level protocols, not hardware enforced precedence rules, are expected to partition provisionable bandwidth resources.

6.1.3 Survey messages

Bandwidth survey messages are sent from one station to itself, but modified by all intermediate stations, as illustrated in Figure 6.4. The initial message has the summarizes the bandwidth accounts of the requester, listed in order of the link's distance from the source.

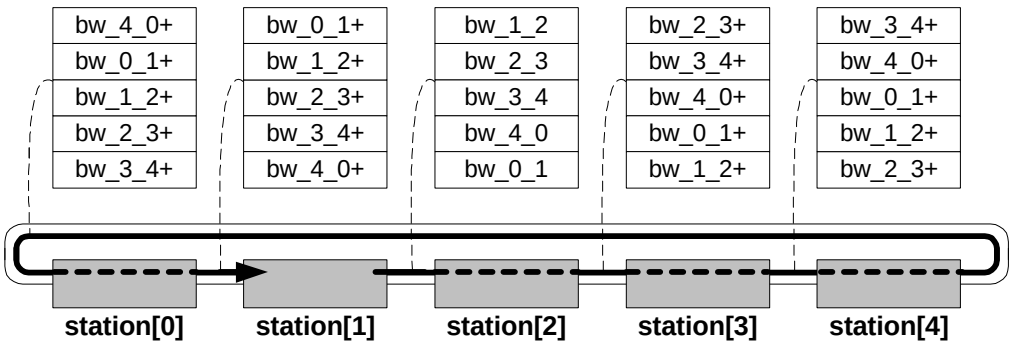


Figure 6.4—Bandwidth survey messages

6.1.4 Survey message processing

To simplify the protocols, each of the intermediate stations has the same behavior, as illustrated in Figure 7.10. The ordering of the incoming entries is first rotated. These rotated values are then added to the station-provided values, either in parallel (as illustrated) or in sequential operations (not illustrated). The cumulative effect of these actions, when performed by all stations, is the return of an accurate bandwidth survey to the requesting station, in this example, *station[0]*.

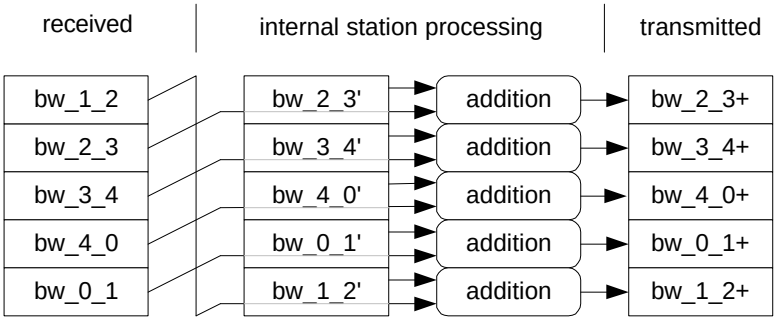


Figure 6.5—Bandwidth survey messages

7. Flow control

7.1 Flow control overview

Flow control protocols are based on the policing of offered traffic, based on the class of the traffic and the provisioned bandwidths. Higher level protocols are expected to further partition bandwidth restrictions of flows from within a station, based on per-flow service level agreements (SLAs) maintained within that station. However, the use of within-the-station per-flow restrictions is beyond the scope of this standard.

Flow control protocols involve limiting transmissions of class-A, class-B, and class-C traffic. The objectives are to achieve desired bandwidth partitioning without compromising bandwidth efficiencies or spatial-reuse opportunities. Strategies for achieving these goals are listed below:

1) Class-A. Two interoperable options for supporting class-A traffic are specified:
   a) Reactive. Each station has a large transit buffer, passBC. The upstream station provides assistance (by stifling class-B transmissions) when that queue is approximately half full.
   b) Proactive. Each station has a small transit buffer, passBC. Each station consumes its returning class-A bandwith to provide sufficient bandwidth for new class-A transmissions.
2) Class-B. Each station communicates its congestion condition to its upstream neighbor. That neighbor provides assistance (by stifling its class-B transmissions) while its class-B queue is less full.
3) Class-C. Each class-C congested station communicates its cumulative transmission count to others. Other stations stifle their class-C traffic when their transmission count exceeds that of the congested link.

7.2 Flow control components

The send queues are located in the client, as opposed to the MAC, so that packets can be conveniently shuffled, inserted, or deleted while awaiting transmission. The queue management protocols are application dependent and beyond the scope of this standard, but class-B and class-C transmit-limit information is passed across the MAC-to-client interface. These transmission-limit indications assume the presence of three client-level queues, as illustrated in Figure 7.1.

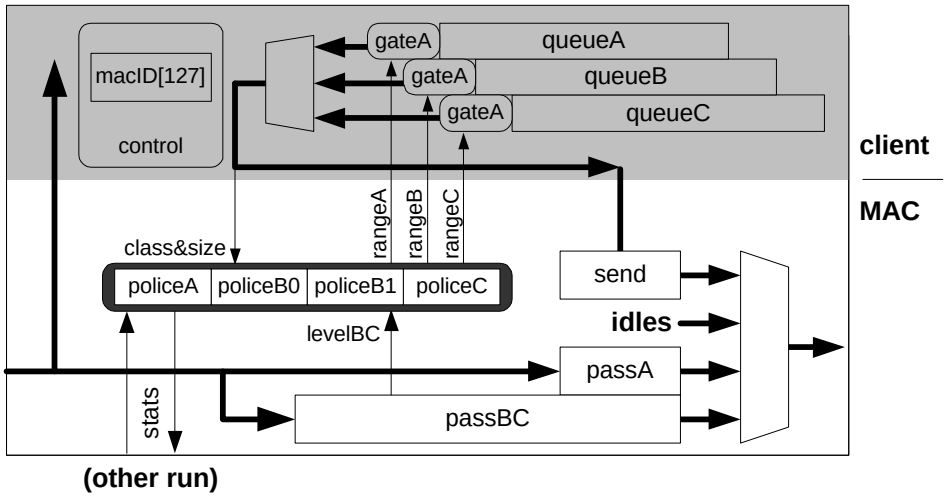


Figure 7.1—MAC data-path components

Whenever possible, the transmission queues are located in the client, as opposed to the MAC, so that packets can be conveniently resorted, inserted, or deleted while awaiting transmission. The client-level queue management protocols are application dependent and beyond the scope of this standard, although queue-gating and queue-depth indications are passed across the MAC-to-client and client-to-MAC interfaces respectively.

The arbitration indications flow in the reverse direction, with respect to the data-frame flows, starting from stations currently requesting their share of the bus bandwidth. The reverse-flow direction allows inactive stations to delay forwarding of arbitration indications while filling of their bypass FIFO generates idles; stations which cannot generate idles quickly forward arbitration indications to throttle upstream stations.

The arbitration indications are level-sensitive signals, rather than tokens or edge-sensitive values, making the protocols robust. Most importantly, from a simplicity perspective, these arbitration indications are fault tolerant, in that special fault-restoration protocols are unnecessary.

7.2.2 Client components

The client is expected to provide ***queueA***, ***queueB***, and ***queueC*** storage for holding class-A, class-B, and class-C traffic respectively. Each queue has a *gate* function, which inhibits client-to-MAC packet transmissions based on the MAC provided flow-control signals.

The client is also expected to maintain a topology table, to assist in identifying the station locations (measured in hop counts) based on their unique station identifiers.

7.2.3 Client interface

The MAC-supplied ***rangeA*** indication specifies the distance over which class-A transmissions are allowed. This information allows the client to select class-A packets eligible for transmission over few uncongested hops, while properly inhibiting transmissions over one or more congested hops.

The MAC-supplied ***rangeB*** indication specifies the distance over which class-B transmissions are allowed. This information allows the client to select class-B packets eligible for transmission over few uncongested hops, while properly inhibiting transmissions over one or more congested hops.

The MAC-supplied ***rangeC*** indication specifies the distance over which class-C transmissions are allowed. This information allows the client to select class-C packets eligible for transmission over few uncongested hops, while properly inhibiting transmissions over one or more congested hops.

7.2.4 MAC components

The ***send*** storage holds a ready-to-send packets in high-speed MAC-resident buffers, to ensure its reliable delivery independent of the client-to-MAC bandwidth and flow-control latencies. This FIFO is expected to be slightly larger than 2 MTUs in size. The presence of a send buffer reduces the critical timing requirements of the client-to-MAC interfae; the late checking of access rights (until the *send* storage is available) and the small size of the send buffer avoid significant priority inversions.

The ***passA*** storage holds class-A traffic that arrives and cannot be immediately retransmitted (typically because the station is transmitting from *sendA*, *sendB*, or *passBC*. The intent is to prepare this incoming class-A traffic for retransmission after the current transmission ends. This FIFOs is expected to be slightly larger than 1 MTU in size.

The ***passBC*** storage holds class-B and class-C traffic that and cannot be immediately retransmitted. This FIFO is expected to be approximately $2 \times Rt \times Fa$, where *Rt* is the number of bytes corresponding to the round-trip latency on the upstream link and *Fa* is the fraction of provisioned class-A traffic. The intent is to save incoming lower-priority traffic while class-A congestion indications are passed upstream.

When used within large metropolitan or small-state environments, the transit buffer *passBC* may be multiple megabytes (not kilobytes) in size. Such buffers are expected to be implemented in high-density high-bandwidth DRAM technologies, not on-chip SRAM technologies.

The MAC scheduler has policing accounts (***policeA***, ***policeB***, and ***policeC***) for this station's traffic classes, and copies of transmission accounts from other congested stations. The scheduler is responsible for controlling the station's output-selection multiplexer and providing transmit-inhibit information (*allowA*, *rangeB*, and *rangeC*) to the client. The intent is to facilitate the client's selections from the appropriate transmission queues.

7.3 Timing resources

Portions of the flow-control protocols rely on knowledge of round-trip delays, as measured between a congested station and its upstream neighbor. The round-trip time is closely associated with the assumed response time of an upstream neighbor and is therefore applicable towards the setting of class-B and class-C flow-control thresholds. These timers piggyback on the periodic congestion messages sent between stations, as illustrated in Figure 7.2.

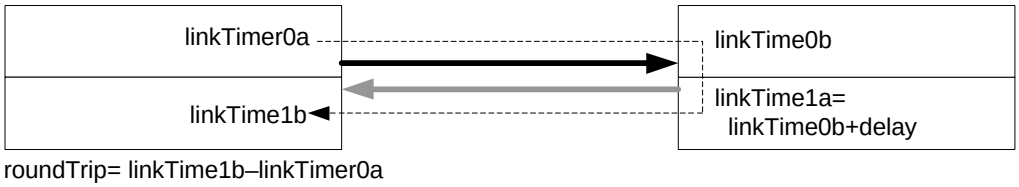


Figure 7.2—Round-trip delay calibration

7.4 Reactive class-A controls

7.4.1 Reactive class-A congestion detection

Class-A congestion relief relies on the depth of the MAC-resident *passBC* FIFO. When *passBC* is mostly empty, as illustrated in the left of Figure 7.3, the transmission of lower-class traffic is allowed. When *passBC* is mostly full, as illustrated in the right of Figure 7.3, the transmission of lower-class traffic is disallowed and assistance from the upstream station (in the form of idle-symbol transmissions) is requested.

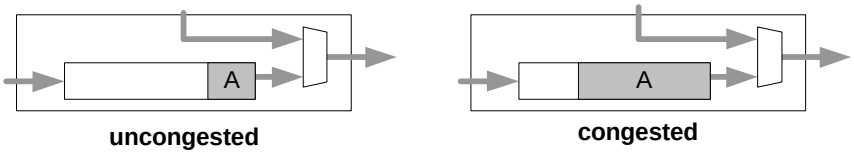


Figure 7.3—Class-A congestion conditions

The upstream node inhibits its class-B traffic when conditions are unfavorable over the downstream link, as specified below:

```
thisDepthBC= (thisDepthBC+thisDelayBC)/FIFO_BC_SIZE;
needDepthBC= (sendDepthB<8 ? sendDepth/4.0 : (sendDepth+8)/8.0);
stopClassB= (thisDepthBC<needDepthBC);
```

7.4.2 Reactive class-A assistance

To avoid *passBC* FIFO overflow, each station communicates its congestion condition to its upstream neighbor. That congestion information is used to set a target depth for the upstream station's *passBC* FIFO, as illustrated in Figure 7.4. While attempting to meet this target depth, the upstream station supplies idles rather than retransmitting *passBC*-resident packets.

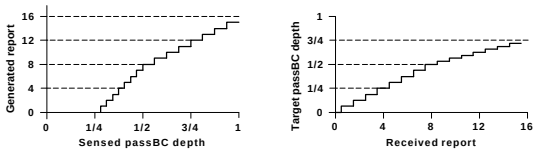


Figure 7.4—Distorted passBC depth objective

7.5 Proactive class-A controls

7.5.1 Proactive class-A congestion avoidance

Proactive class-A flow control involves periodic interleaving of class-A packets within the data stream. The amount of interleaved class-A flow control packets on each link is sufficient to sustain the provisioned class-A traffic on any of the links; e.g. all links effectively support the class-A traffic requirements of most heavily provisioned link.

With the presence of such interleaved traffic, each node receives sufficient strippable class-A traffic to support its class-A transmissions. For brief periods, some class-A traffic may be transmitted while no class-A traffic is present for consumption, as illustrated in the left of Figure 7.3. When this occurs, the passBC FIFO depth increases, as necessary to hold the incoming traffic.

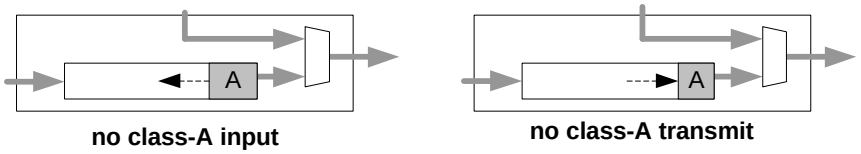


Figure 7.5—Proactive class-A conditions

However, when incoming class-A traffic arrives, the passBC FIFO can be emptied, as illustrated in the right of Figure 7.3. Since the average rate of class-A traffic is prenegotiated and maintained throughout the ringlet, there is no need to dynamically throttle upstream stations based on the depth of the passBC FIFO. However, a modest (several times the maximum frame size) *passBC* FIFO is thought to be sufficient to sustain class-A transmissions during brief pauses in incoming class-A traffic.

Support of proactive class-A controls has an effect on the packet-stripping protocols of reactive class-A stations, as listed below.

1) Proactive. A proactive station processes class-Ax packets as follows:
   a) Stripping. The class-Ap packets are stripped by relabeling them as class-Aq packets.
   b) Passing. The class-Aq packets are ignored when passing through.
2) Reactive. A reactive station processed class-Ax packets as follows:
   a) Stripping. Selected class-Ap and all class-Aq packets are stripped.
   b) Sourcing. Additional class-Aq packets are transmitted, from within the MAC, as necessary to sustain the provisioned level of class-A link traffic.

7.6 Class-B congestion

7.6.1 Class-B congestion indications

Class-B congestion relief relies on the depth of the *queueB* buffer, as illustrated in Figure 7.6. When *queueB* is mostly empty or nearly full, few or many of idle units are requested from the upstream station. The request for idle units throttles the upstream station based on this station's class-B requirements.

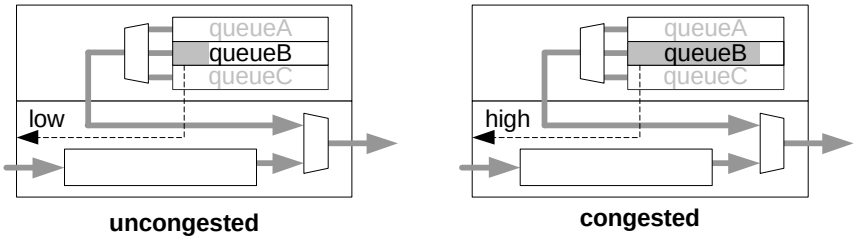


Figure 7.6—Class-B congestion conditions

The upstream node inhibits its class-B1 traffic when conditions are unfavorable over the transmitted link, as specified below:

```
thisDepthB= (thisDepthB+thisDelayB)/THRESHOLD;
needDepthB= (sendDepthB<8 ? sendDepth/8.0 : (sendDepth+8)/16.0);
stopClassB1= (thisDepthB<needDepthB);
```

7.6.2 Distributed class-B assistance

To avoid overprovisioned overflows, each station communicates its class-B congestion condition to the other stations. That congestion information is used to set a target depth for the upstream station's class-B queues, as illustrated in Figure 7.4. While attempting to meet this target depth, the conflicting stations inhibit their transmission of *class-B1* traffic.

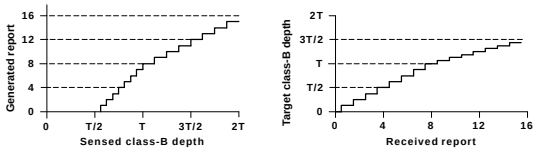


Figure 7.7—Distorted class-B depth objective

7.7 Class-C congestion

Unless throttled by others, or higher-class traffic is available, a station transmits class-C traffic, as illustrated in Figure 7.8. Perceived congestion conditions cause a station to publish its *runRate* to upstream nodes. Greedy upstream stations are prevented from further increasing their *runRate* (associated with this link) beyond the communicated level. The intent is to ensure fairness on the basis of *runRate* counts over the most congested links.

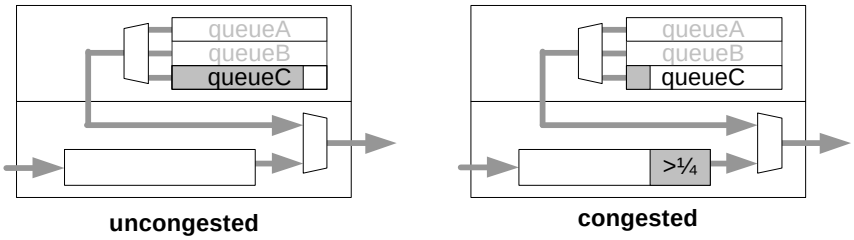


Figure 7.8—Class-C congestion conditions

The congestion indication is set based on the depth of the passBC FIFO, the amount of

```
// depthBC is the depth of the passBC FIFO
// depthC is the depth of the client's queueC buffer
// timeC is the waiting time for the first queueC entry
// timeScale is implementation-dependent, typically 1
// thresholdC is implementation-dependent, typically roundTrip time
congestion= (depthBC+depthC+timeScale*timeC)>thresholdC;
```

The intent is to defer the assertion of upstream flow-control indications, until sufficient traffic is available to consume unnecessarily sent idles. A time threshold is also involved, to avoid indefinite blocking of low-rate class-C traffic.

The current assumption of strict fairness is intended to simplify the initial writeups. Weighted fairness would be based on the same principles, with *rate* increment values are adjusted by a weighted fairness scaling factor.

7.8 Transmit selections

Transmission selections depend on the need for help by the downstream station, in the form of idle symbols, as specified in Table 7.1.

Table 7.1—Transmit selections

condition	Row	selection	Description
—	7.1.1	passA	Always prepare for class-A
	7.1.2	send	Always transmit client-supplied traffic
Sloped(informState.levelBC) ≥ thisState.levelBC	7.1.3	idles	Assist downstream neighbor
—	7.1.4	passBC	Retransmission when helpful
	7.1.5	idles	Send idles when no frames are available

Row 7.1.1: The class-A transmit FIFO is emptied first, to enable further class-A transmissions.

Row 7.1.2: All transmissions wait until the *send* FIFO has been emptied. No distinction between class-A/class-B/class-C traffic is made, since these considerations were made in selecting the appropriate frame from the client.

Row 7.1.3: When this *passBC* FIFO is relatively empty and the downstream stations's *passBC* FIFO is relatively full, idle units are supplied to assist the downstream station.

Row 7.1.4: The *passBC* FIFO is emptied in preparation for the next transmission conflict.

Row 7.1.5: Idle units are generated while no transmit frames are available.

7.9 Congestion-condition packets

An array of congestioin information represents a concatenation of range-dependent information, as illustrated in Figure 7.9.

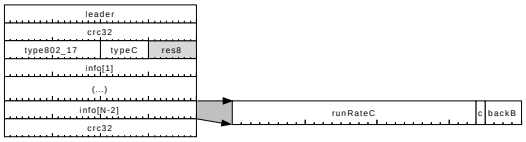


Figure 7.9—Communicated congestion information

The ***info[0]*** through ***info[N-2]*** entries communicate fairness-progress counts from congested stations to the others. Within each of these *info[n]* components, the 23-bit *runRateC* values correspond to the published rate of each station's class-C traffic. The ***c*** (congestion) bit values of 0 and 1 indicate that traffic is uncongested and congested respectively (see rows 7.3.9 and 7.3.10 of Table 7.3). The 4-bit ***backB*** values correspond to the level of retained class-B traffic.

7.10 Policing actions

7.10.1 Policing state

Each station maintains policing state, to restrict the rate of its transmissions, as illustrated in Figure 7.10.

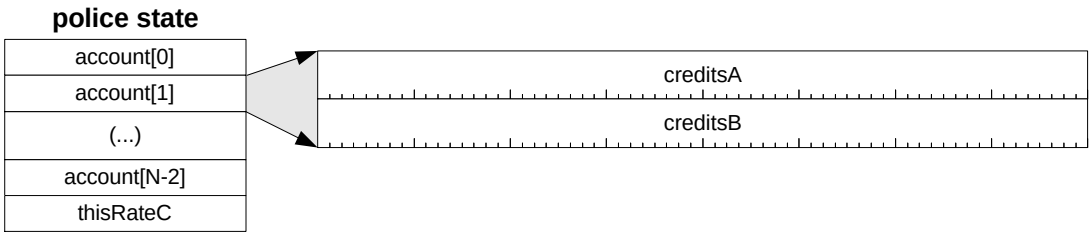


Figure 7.10—Policing state

The police state involves maintenance of N-1 accounts, where N is the number of stations attached to the ring. The *n*'th account corresponds to a link located between the source and destination, where *n* is the number of stations located between source and this link. Each account has multiple components, described in the remainder of this subclause.

The signed 64-bit ***creditsA*** value enables or disables class-A traffic on this link, for positive and negative values respectively. The signed 64-bit ***creditsB*** value enables or disables class-B traffic on this link

The more-significant portion of these credit values corresponds to 16-byte transmission units; the less-significant portion corresponds to fractions of said units. Credits are decremented by the frame size, measured in transmission units, when frames are selected for transmission. Credits are incremented every transmission unit interval, based on the provisioned fraction of available bandwidth.

A single 31-bit ***thisRateC*** value tracks the total number of class-C transmission units sent during congested conditions. The intent is to distribute this value to others, under congestion conditions, to inhibit their excessive class-C traffic transmissions.

7.10.2 Policed ranges

The policing policies restrict the selection of transmission traffic by creation of the appropriate flow-control indications for the client, as specified in Table 7.2.

Table 7.2—Transmit indications

Class-A				
Blocking	Scope	Condition	Row	Indication
blockedA==0	for all n≤m	policeState.account[n].creditsA ≥ 0	7.2.1	rangeA=m
blockedA==1	—	—	7.2.2	rangeA= –1
Class-B				
Blocking	Scope	Condition	Row	range
blockedB==0	for all n≤m	policeState.account[n].creditsB ≥ 0	7.2.3	rangeB= m, defer=0
blockedB==1	for all n≤m	policeState.account[n].creditsB ≥ 0	7.2.4	rangeB= m, defer=1
Class-C				
Blocking	Scope	Condition	Row	range
blockedB==0	for all n≤m	informState.accounts[n].runC.c==0 || (Difference(informState.accounts[n].rateC, policeState.accounts[n].rateC, 31)>0)	7.2.5	rangeC=m
blockedB==1	—	—	7.2.6	rangeC= –1

```
#define BlockedA (thisState.levelA > HALF_A)
#define BlockedB \
  (blockedA || thisState.levelBC > HALF_BC || Clamped(informState.levelBC)>thisState.levelBC))
```

Row 7.2.1: The class-A traffic is limited to its per-hop prenegotiated provisioned rate.
Row 7.2.2: Accept nothing when blocked (e.g., the send buffer could overflow).

Row 7.2.3: The class-B0 traffic is limited to its per-hop prenegotiated provisioned rate.
Row 7.2.4: The class-B0 traffic allocations continue while transmissions are blocked.

Row 7.2.5: Block class-C based by downstream station’s class-B or class-C fairness threshold.
Row 7.2.5: The class-C traffic is opportunistic, no credits are accumulated when traffic is blocked.

7.10.3 Credit adjustments

Police-state updates on frame selections depend on the frame type, as listed below. The class-A and class-B0 updates involve updates of *n* accounts, where *n* is the number of hops between source and destination stations.

Transmission selections depend on the need for help by the downstream station, in the form of idle symbols, as specified in Table 7.3.

Table 7.3—Credit adjustments

class	action	Row	Action
A	frameTransfer	7.3.1	policeState.account[n].creditsA–= frameSize
	queueEmpty	7.3.2	policeState.account[n].creditsA= MIN(policeState.account[n].creditsA, 0)
	transferUnitTick	7.3.3	MIN(policeState.account[n].creditsA+ grantState.account[n].fractionA, timeA)
B	frameTransfer	7.3.4	policeState.account[n].creditsB–= frameSize
	queueEmpty	7.3.5	policeState.account[n].creditsB= MIN(policeState.account[n].creditsB, 0)
	TransferUnitTick&& policeState.account[n].creditsB<timeB	7.3.6	MIN(policeState.account[n].creditsB+ scaleB*grantState.account[n].fractionA, timeB)
	transferUnitTick&& policeState.account[n].creditsB>=timeB	7.3.7	MIN(policeState.account[n].creditsB+ grantState.account[n].fractionA, timeC)
C	frameTransfer	7.3.8	thisRateC+= frameSize
	queue not empty	7.3.9	thisRateC.c= 1
	queueEmpty	7.3.10	thisRateC.c= 0

Row 7.3.1: Credits are reduced by the packet-transmission size.
Row 7.3.2: Positive credits are discarded (rather than continually accumulated) if nothing is ready to send.
Row 7.3.3: Credits are accumulated while waiting to send. Although the credit limit is bounded by design, a maximum bound is provided to recover from erroneous-state conditions.

Row 7.3.4: Credits are reduced by the packet-transmission size.
Row 7.3.5: Positive credits are discarded (rather than continually accumulated) if nothing is ready to send.
Row 7.3.7: Credits are accumulated while waiting to send, based on a scaling factor *scaleB* (see xx). Although the credit limit is bounded by design, a maximum bound is provided to recover from erroneous-state conditions.

Row 7.3.8: Each frame transmission increased this node's class-C run-rate value by the transmission size.
Row 7.3.9: A congestion indication is provided when class-C traffic is blocked.
Row 7.3.10: An uncongested indication is provided no class-C traffic is being blocked. The intent is to allow other stations to ignore the associated run-rate information when no congestion condition is present.

8. Topology discovery

8.1 Topology discovery protocol

The implementation of the discovery mechanism is a periodic activity, but can also be initiated on “a need to know” basis. For the need-to-know protocols, each node generates a topology discovery packet whenever they one is needed e.g., on first entering a ring, when a protection switch request message is received, or when the station detects a fiber failure condition.

Topology discovery involves passing of identifiers from each station to its adjacent neighbor, with a cumulative effect of identifying the ring topology. The discovery-packet header indicates the packet is to be stripped and regenerated when passing though nodes. Regeneration involves having each station places its identifier at the start of the cumulative identifier array, while stripping redundant entries from end of the identifier array.

If there is a wrap on the ring, the discovery packet is forwarded across the wrap. The length of these wrapped packets is twice as large, since each station has two distinct identifiers, one for each ring attachment.

The topology information can be used to determine the shortest path to a station (since there are two attach points). The topology information is also used to generate compact stationID addresses for directly attached stations.

Note that the topology map only contains the reachable nodes, and (in the case of wrap) some of the nodes are reached twice. It cannot identify unreachable stations present on other ring segments. The topology information is not required to support the protection mechanism.

TBD—This is probably not good enough.
A topology map is changed only after receiving two topology packets which indicate the same new topology (to prevent topology changes on transient conditions).

8.2 Discovery messages

8.2.1 Created discovery messages

Discovery messages are used to accumulate topology information, by having each station prepend its identifier to passing through discovery messages, as illustrated in Figure 8.1.

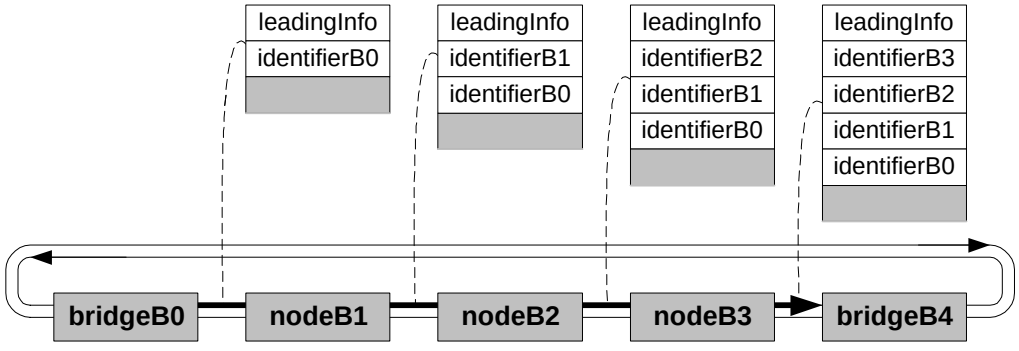


Figure 8.1—Prepending discovery messages

8.2.2 Truncating discovery messages

Each station is also responsible for truncating the discovery message before the second instance of its identifier, as illustrated in Figure 8.2.

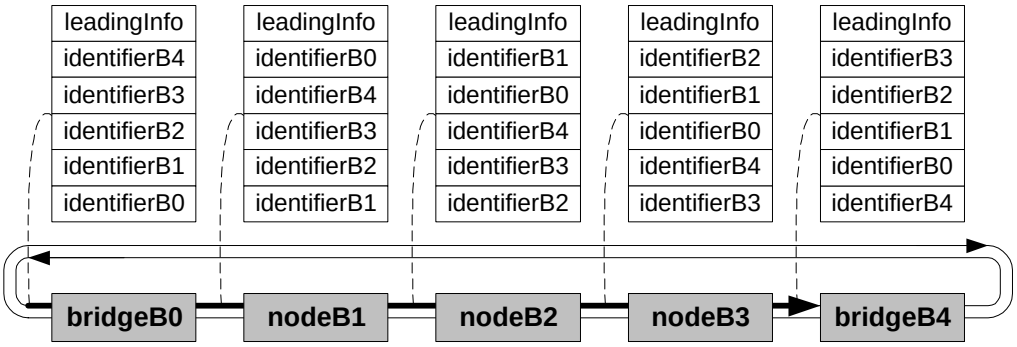


Figure 8.2—Truncated discovery messages

8.3 Discovery formats

The discovery frame provides topology information, as illustrated in Figure 8.3.

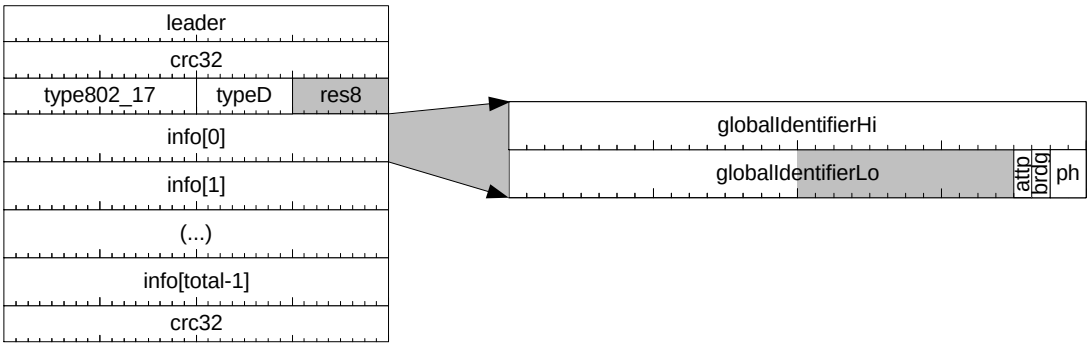


Figure 8.3—Discovery frame formats

Distinct ***attp*** (attach point) bit values are assigned to each of the attach points, allowing them to be distinctively identified.

There is no requirement that the stations *attp* identifier correspond to the attached ring identifier, except on the station with the largest *globalIdentifier*, where the *ring* identifier is derived from the *attp* identifier.

The ***brdg*** (bridge attachment) bit is 1 if the station can respond to more than one unicast destinationMacAddress, typically for the purposes for communicating with remotely located stations. Otherwise, the *brdg* bit shall be 0. The intent is to easily identify attached bridge stations, so that other MACs can properly address frames to pass by them.

The 2-bit ***ph*** (discovery phase) field ensures the integrity of passing through discovery packets.
Details are TBD.

9. Transit processing

9.1 Incoming frame processing

The processing of incoming frames depends on the validity of the packet and its contents. If the ***headerCrc*** is invalid, the frame shall be immediately discarded. Otherwise, the *timeToLive*, *destinationStationID*, *sourceStationID*, *destinationMacAddress*, and *sourceMacAddress* fields affect the frame processing, as specified in Table 9.1 and following row-by-row descriptions.

Table 9.1—Incoming frame processing

TTY	destination StationID	source StationID	flood	destination StationID	source MacAddress	Row	Action	Route
—	—	—	—	—	—	9.1.1	discard*	strip
≥1	—	—	—	matchMac	anyMac	9.1.2	copied	
	matchID	—	—	NONE	NONE	9.1.3	mapped	
		—	—	multicastMac	anyMac	9.1.4	multicheck	
		—	—	unicastMac	anyMac	9.1.5	unicheck#	
—	—	matchID	—	—	—	9.1.6	discard*	
—	—	—	—	anyMac	matchMac	9.1.7		
≤1	—	—	—	—	—	9.1.8		
>1	—	—	—	multicastMac	anyMac	9.1.9	multicheck	pass
	—	—	1	unicastMac	anyMac	9.1.10	unicheck#	
	—	—	—	—	—	9.1.11	ignored	

Notes:
* Error condition should be logged
# Non-bridge nodes discard these frames

Row 9.1.1: A corrupted frame, with an invalid header-CRC is invalid, shall be discarded.

Row 9.1.2: Any frame with matching destinationMacAddress is stripped and copied.
Row 9.1.3: A *destinationStationID*-matching frame is stripped; a mapped MAC addresses is assigned.
Row 9.1.4: A *destinationStationID*-matching multicast is stripped & checked for multicast matches.
Row 9.1.5: A *destinationStationID*-matching unicast frame is stripped.
Bridges check these frames for possible forwarding to remote locations; nonbridges discard these frames.

Row 9.1.6: A sourceID-matching frame is stripped&discarded at its source station; an error is logged.
Row 9.1.7: A sourceID-matching frame is stripped&discarded at its source MAC; an error is logged.
Row 9.1.8 : A will-become-zero *timeToLive* field is stripped&discarded; an error is logged.

Row 9.1.9: A different-ID multicast frame is copied and checked for multicast matches.
The time-to-live field is decremented as the packet passes through the station.
Row 9.1.10: A different-ID unicast frame is copied by bridges and checked for unicast matches.
Bridges check these frames for possible forwarding to remote locations; nonbridges discard these frames.
The time-to-live field is decremented as the packet passes through the station.
Row 9.1.11: The time-to-live field is decremented as the packet passes through the station.

The integrity of the payload-CRC value has no effect on its routing decision, but affects error logging and stomped-CRC processing, as further described in 9.2.1.

9.2 CRC processing

9.2.1 Data CRC stomping

Cut-through frame processing allows frame payload retransmissions to begin before the frame's CRC has been verified. Store-and-forward processing may confirm a valid header-CRC but detect an invalid data-CRC. In both cases, the verified header continues circulating and the payload is marked invalid.

With this invalidation strategy, a payload transmission error causes an error to be logged and the frame's CRC set to a well defined "stomped" value. That stomped value is also an invalid CRC value, but further logging of the error condition is inhibited. These erroneous-CRC processing steps are illustrated in Figure 9.2.

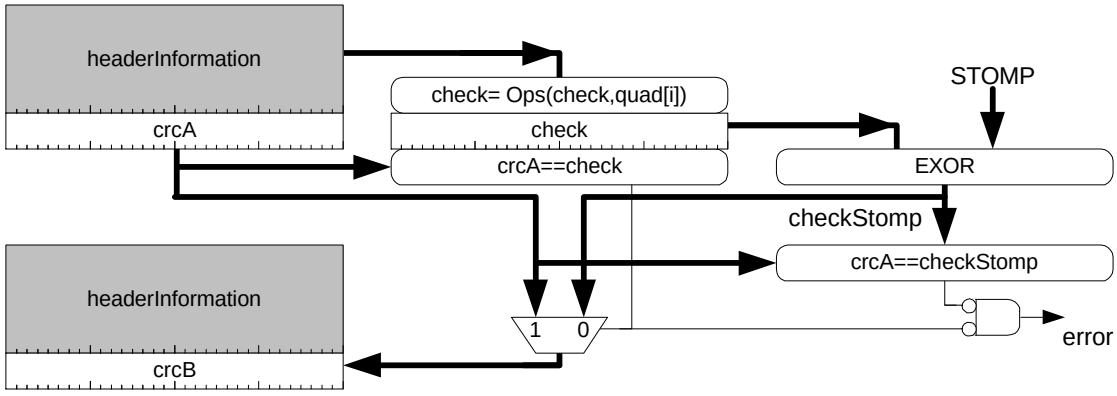


Figure 9.1—Data CRC stomping

A new CRC value, called check, is computed based on the frame's contents. The *checkStomp* value is computed by EXOR'ing the check value with a STOMP value (STOMP is a 32-bit constant). If the frame's *crcA* differs from the computed *check* value, the revised *crcB* value is set to the *checkStomp* value. An error condition is flagged if the frame's CRC value is incorrect (*crcA*!=*check*) and the error has apparently not been previously flagged (*crcA*!=*checkStomp*).

9.2.2 Protected time-to-live adjustments

The time-to-live field is normally decremented when frames pass through stations, so that corrupted frames can be discarded when the destination station is no longer present or is incorrectly identified. Decrementing the TTL field involves adjusting the following CRC field, as illustrated in Figure 9.2.

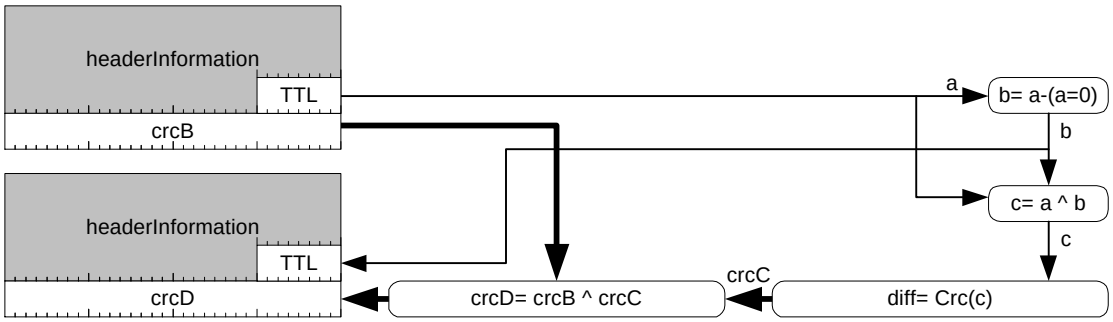


Figure 9.2—Protected time-to-live adjustments

An incremental update of the CRC shall be used to maintain CRC coverage when the TTL field is adjusted. This involves computing the new TTL field value *b* and the difference *c* between new and old TTL values. The difference value *c* generates an incremental CRC value *crcC*, which is EXOR'd with the old *crcB* value to generate the new *crcD* value. The data is never left unprotected: an error in *crcB* or the computation of internal CRC values will (nearly) always be reflected as an error in check value *crcD*.

10. Protection

This clause is preliminary; further specification details are required.

10.1 Severed link effects

10.1.1 Link failures

A link failure effects the routing (1) of packets that would normally pass over the affected link. A station may elect to wrap quickly, returning packets (2a) on the opposing run rather than discarding them at the filed link, as illustrated in the left of Figure 10.1. Although packet loss is minimized, excessive link bandwidth is consumed.

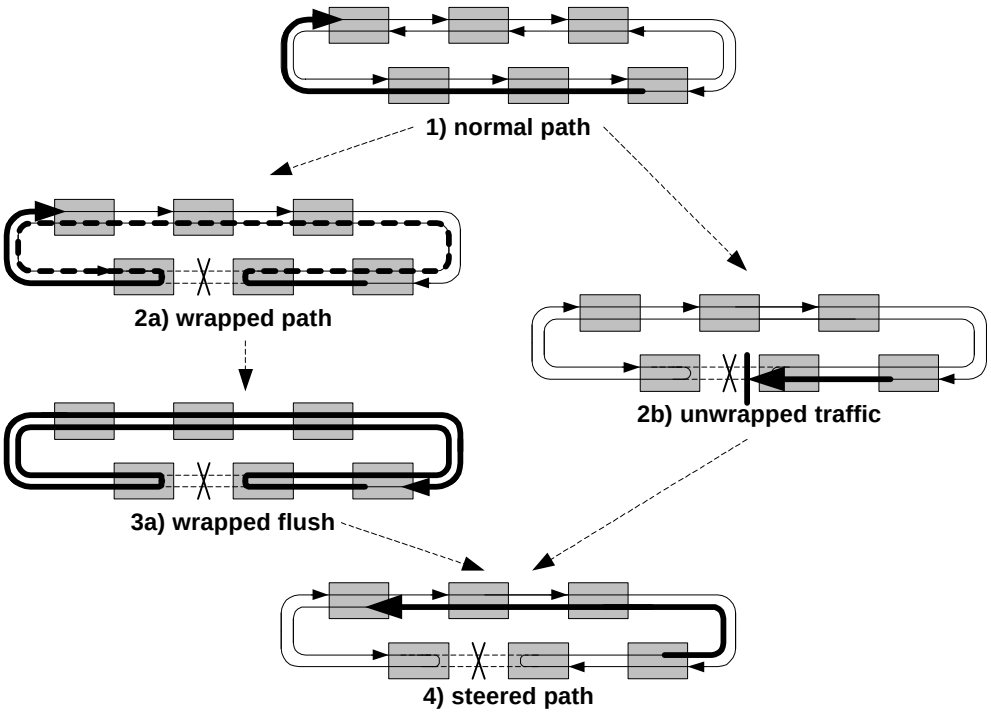


Figure 10.1—Protection steering

Wrapping is expected to be a transient state, as steering is more efficient and part of the failed-link recover protocols (which are ultimately invoked when the failed link operation is restored). To maintain packet ordering when switching between wrapping and steering modes, outstanding traffic must be flushed before the change occurs. Packet flushing involves sending a non-class-A packet to one's self, along the wrapping path. All packets are known to be delivered, and switching between wrapping and steering is therefore safe, when this packets returns to its source.

If only steering is employed, some traffic will continue to be lost (2b) until intermediate nodes become aware of the ring failure, and begin transmitting traffic on both ringlets. However, because the data packets are discarded at the failed link, no flush operation is required before steering (4) is invoked.

10.1.2 Link recovery

A severed link (1) is expected to be recovered, whereupon dual ringlet operations (2) become possible. Efficient utilization of these ringlets involves flushing outstanding traffic (3) before redirecting traffic (4) in the preferred direction, as illustrated in Figure 10.2.

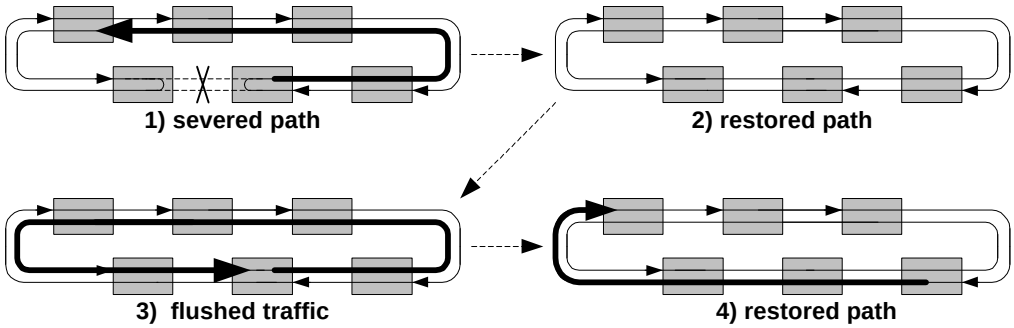


Figure 10.2—Protection steering

Link recovery from a wrapped mode is not supported. Instead, wrapped rings are converted to steered rings (see 10.1.1), whereupon the aforementioned link-recovery techniques can be used.

Annexes

Annex A: Bibliography (informative)

The following publications are recommended as background material for understanding the objectives behind this standard:

[B1] IEEE Std 1596-1992, Scalable Coherent Interface.[21]

[B2] IEEE Std 1394-1995, High Performance Serial Bus.[4]

[21] ANSI/IEEE publications are available from the Institute of Electrical and Electronics Engineers, Service Center, 445 Hoes Lane, P. O. Box 1331, Piscataway, NJ 08855-1331, USA.

Annex B: 802 LAN bridging (normative)

B.1 Bridging overview

All types of IEEE 802 Local Area Networks (or LANs) can be interconnected using MAC bridges. Each individual LAN consists of devices attached to the LAN having the same MAC type. The bridged LAN created allows for the inter-connection of stations attached to separate LANs as if they were attached to a single LAN, although they are in fact attached to separate LANs. A transparent MAC bridge operates below the MAC service boundary, and is transparent to protocols operating above this boundary, in the logical link control (LLC) sublayer or network layer (ISO/IEC 7498-1: 1994 1). The presence of one or more MAC bridges can lead to differences in the quality of service (QOS) provided by the MAC sublayer; it is only because of such differences that MAC bridge operation may not be fully transparent.

A bridged LAN can provide for

1) The interconnection of stations attached to LANs of different MAC types;
2) An effective increase in the physical extent, the number of permissible attachments, or the total performance of a LAN;
3) Partitioning of the physical LAN for administrative or maintenance reasons.

The MAC bridge standard IEEE Std 802.1D-1990 (subsequently republished as ISO/IEC 10038:1993 [IEEE Std 802.1D, 1993 Edition]) specifies an architecture and protocol for the interconnection of IEEE 802 LANs below the MAC service boundary. Within this context, the RPR network defines a ring topology forming a broadcast media where specific access control mechanisms are employed by the MAC in order to achieve frame delivery and spatial reuse on the ring media. The RPR MAC entity shall provide optional functions within the MAC which optimize bridging of 802 traffic across the ring medium in order to maintain spatial reuse of unicast traffic, as illustrated in bridging reference model of Figure B.1.

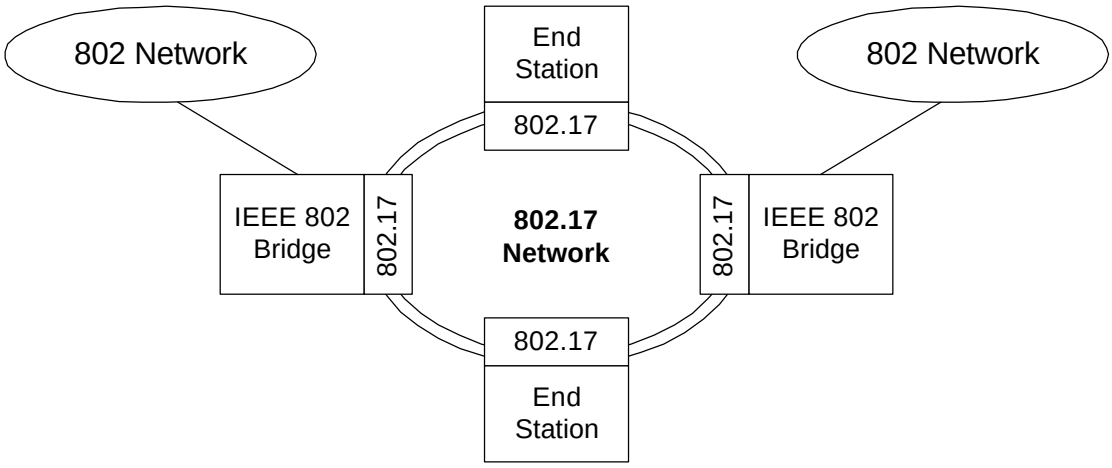


Figure B.1—Bridging reference model for an 802.17 network

In order to support transparent bridging of 802 traffic and maintain the spatial reuse property of the ring, the RPR MAC service interface performs the following functions : simple mapping of 802 traffic to the RPR frame format, transport of 802 traffic across the RPR physical medium and delivery of 802 traffic to either the MAC relay or intended 802.17 client at the RPR MAC service interface. The mapping function performed by the RPR MAC service interface shall conform to the interface between a MAC entity and MAC relay, and preserve the filtering services and other requirements for bridged LANs as specified in

ISO/IEC 10038 [IEEE Std 802.1D, 1998 Edition], and ISO/IEC [IEEE Std 802.1q, 1998 Edition]. These services include:

1) Maintaining the bridge architecture;
2) Maintaining the nature of filtering services in bridged LANs;
3) Maintaining the extensions specified by IEEE P802.1Q to allow MAC bridges to support the definition and management of virtual LANS (VLANs);
4) Maintaining the provision of filtering services that support the dynamic definition and establishment of groups in a LAN environment, and the filtering of frames by Bridges such that frames addressed to a given group are forwarded only on those LAN segments that are required in order to reach the members of that group;
5) Supporting the registration protocol that is required in order to provide dynamic multicast filtering services;
6) Supporting management services and operations that are required in order to support administration of dynamic multicast filtering services;
7) Maintaining the provision of expedited traffic capabilities, to support the transmission of time-critical information in a LAN environment;
8) Maintaining the concept of traffic classes and the effect on the operation of the forwarding process of supporting multiple traffic classes in bridges;
9) Maintaining the spanning tree algorithm and protocol;
10) Maintaining the generic attribute registration protocol (GARP);
11) Maintaining the GARP multicast registration protocol (GMRP);

B.2 Architectural model of a bridge

The RPR MAC conforms to the architectural model of a bridge as defined by IEEE 802.1D. The component LANs are interconnected by means of MAC bridges; each port of a MAC bridge connects to a single LAN. Figure B.2 illustrates the architecture of such a bridge.

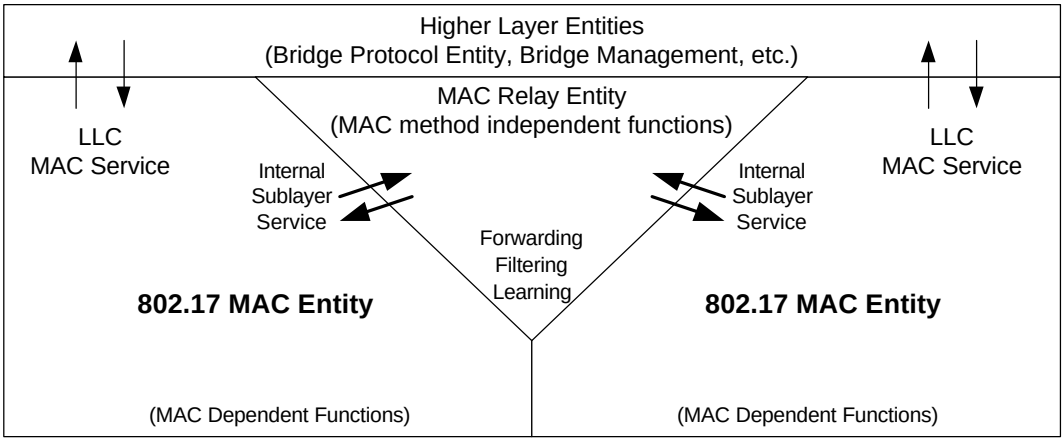


Figure B.2—Bridge architecture model

A bridge consists of:

1) A MAC relay entity that interconnects the bridge's ports;
2) At least two ports;

3) Higher layer entities, including at least a bridge protocol entity.

B.2.2 MAC relay entity

The MAC relay entity handles the MAC method independent functions of relaying frames between bridge ports, filtering frames, and learning filtering information. It uses the internal sublayer service provided by the separate MAC entities for each port. Frames are relayed between ports attached to different LANs.

B.2.3 Ports

Each bridge port transmits and receives frames to and from the LAN to which it is attached. An individual MAC entity permanently associated with the port provides the internal sublayer service used for frame transmission and reception. The MAC entity handles all the MAC method dependent functions (MAC protocol and procedures) as specified in the relevant standard for that IEEE 802 LAN MAC technology.

B.2.4 Higher layer entities

The bridge protocol entity handles calculation and configuration of bridged LAN topology.

The bridge protocol entity and other higher layer protocol users, such as bridge management (7.1.3) and GARP application entities including GARP participants (Clause 12), make use of logical link control procedures. These procedures are provided separately for each port, and use the MAC service provided by the individual MAC Entities.

B.3 RPR MAC bridging reference model

The MAC reference model is illustrated in Figure B.3. The RPR MAC consists of a MAC entity which provides the media access control functions to the pair of ringlets (ringlet0/ringlet1) comprising the RPR ring. The pass-through function within the RPR MAC entity processes frames which are intended for other RPR stations on the ring. The pass-through function takes frames from the receive side of the ringlet and presents them to the transmit side of the ringlet. Traffic received from either ringlet_0 or ringlet_1, intended for this RPR station, is passed up to the RPR internal sublayer service which in turn passes ingress traffic to the 802 MAC relay entity. The 802 MAC relay performs forwarding, filtering, learning functions between this RPR interface and other 802 type interfaces within the bridge. Traffic from the 802 MAC Relay destined to the ring is presented to the RPR internal sublayer service which in turn determines whether to transmit the traffic on either ringlet_0, ringlet_1, or in some cases both. The RPR internal sublayer service performs the mapping between client MAC addresses provided by the 802 MAC relay, and RPR station addresses in the RPR frame header.

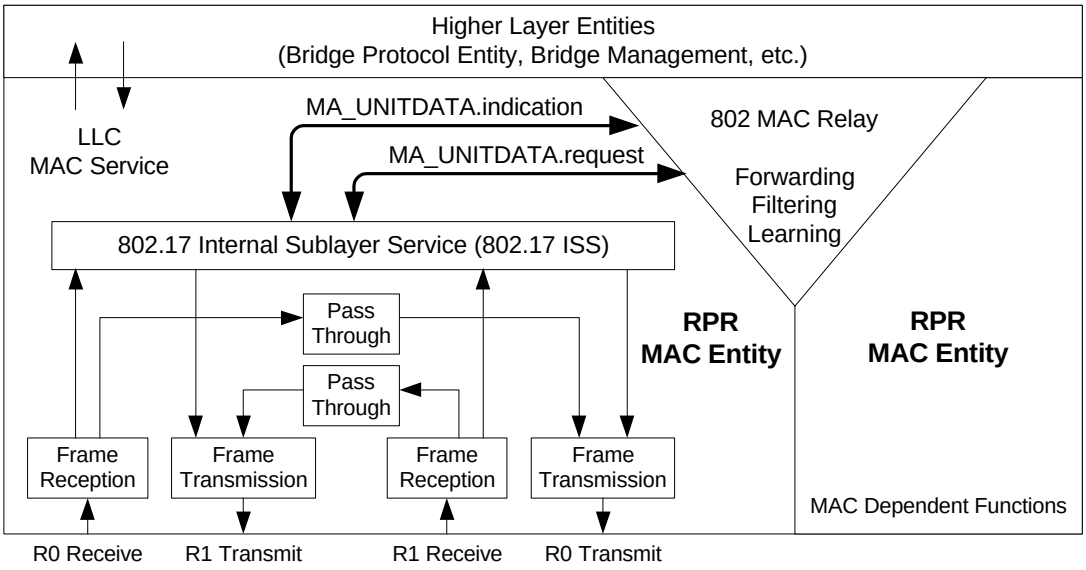


Figure B.3—RPR MAC reference model

The RPR MAC entity appears as a single interface to the 802 MAC relay. This means the RPR ring media and the collection of stations which attach to the ring appears to the 802 MAC relay as a single loop free broadcast media. The RPR MAC ensures that a frame is delivered to the intended RPR station (in the case of a known unicast) or is delivered to all stations (in the cast of a multicast, broadcast or unknown frame). The RPR MAC also ensures that only a single copy of a frame is delivered to the RPR MAC internal sublayer service within each station. RPR MAC procedures ensure that duplicate copies of a frame are not transferred to the RPR internal sublayer service (ISS). This includes scenarios where the ring is in a normal operating configuration, or frames are being wrapped or steered during a ring failure. Since the RPR ring behaves as a loop free broadcast media, spanning tree protocol is not required for networks where a collection of 802 bridges attach to a single RPR ring and do not create a loop via another network connection. spanning tree protocol can be enabled over an RPR ring for the purpose of maintaining a loop free bridged network topology when 802 bridges attach to an RPR ring and are multiply interconnected via another RPR ring or 802 type network. The RPR MAC entity provides LLC services to support the bridge protocol entity and other higher layer protocol users.

The RPR MAC entity appends RPR source/destination station identifier's (DSID, SSID) to the RPR frame for the purpose of performing destination and source stripping of frames from the ring. Destination stripping allows a frame to be stripped from the ring when arriving at the intended destination without having to traverse the entire ring. Subsequent spans, following the span where the packet was stripped, can be reused by other stations for transmitting new traffic onto the ring thereby providing spatial reuse of the ring. Source stripping ensures that a frame which traverses the entire ring is not read a second time by stations, thus maintaining a loop free behavior. RPR destination station ID is appended to the RPR frame via a mapping function as part of the 802.17 MAC Entity support of the ISS. This mapping function maps the 802 *destinationMacAddress* to the RPR *destinationStationID* in the RPR frame header. The RPR ISS also appends the RPR *sourceStationID* in the RPR frame header with the transmitter's source station identifier.

B.4 Model of operation

The model of operation is simply a basis for describing the functionality of the MAC bridge. It is in no way intended to constrain real implementations of a MAC bridge; these may adopt any internal model of operation compatible with the externally visible behavior that this standard specifies. Conformance of equipment to this standard is purely in respect of observable protocol.

B.4.1 802.17 Support of the Internal Sublayer Service

The following figure illustrates the mapping of the MA-UNITDATA.request / MA-UNITDATA.indication primitives to the 802.17 frame format.

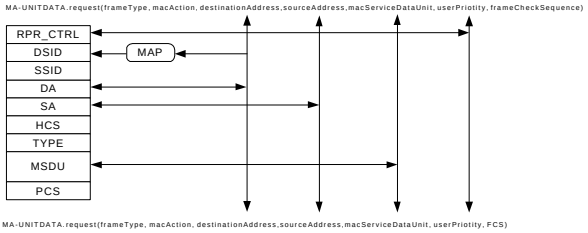


Figure B.4—Mapping of MA-UNITDATA primitives to 802.17 frame format

On receipt of an M_UNITDATA.request primitive, the local MAC Entity performs Transmit Data Encapsulation, assembling a frame using the parameters supplied as specified below. On receipt of a MAC frame by Receive Media Access Management, the MAC frame is passed to Receive Data Decapsulation, which validates the FCS and disassembles the frame, as specified below, into the parameters that are supplied with an M_UNITDATA.indication primitive.

The ***frameType*** parameter takes only the value user_data_frame and is not explicitly encoded in MAC frames. The ***macAction*** parameter takes only the value request_with_no_response and is not explicitly encoded in MAC frames.

The ***destinationAddress*** parameter is encoded in the *destinationMacAddress* field of the MAC frame (see 5.2.1). The ***sourceAddress*** parameter is encoded in the *sourceMacAddress* field of the MAC frame (see 5.2.1).

The number of octets in the ***macServiceDataUnit*** parameter is encoded in the length field of the MAC frame (IEEE Std 802.17 ?.?.?), and the octets of data are encoded in the data field (see 5.2.1).

The ***userPriority*** parameter provided in a data request primitive is encoded in corresponding priority bits of the RPR control header of the transmitted frame. The *userPriority* parameter provided in a data indication primitive takes the value of the corresponding priority bits of the RPR control header of the received frame.

The headerCheckSequence (***HCS***) of the MAC frame is computed as a function of the *destinationStationID, sourceStationID*, *destinationMacAddress*, *sourceMacAddress*, and RPR header control fields of the transmitted frame.

The *payloadCheckSequence* (***PCS***) of the MAC frame is re-computed as a function of the MacServiceDataUnit (see xx).

The ***frameCheckSequence*** parameter in the MA_UNITDATA.request is defined as an unspecified value, signaling the underlying 802.17 MAC to regenerate the frame FCS. The FCS in the

MA_UNITDATA.indication is set to either valid or invalid based on whether the FCS of the receive frame is valid/invalid.

NOTE 1—IEEE Std 802.3, 1998 Edition, describes the use of either a Length or an Ethernet protocol type in its frame format; however, the text of this subclause has yet to be revised to describe the use of Ethernet protocol types.

B.4.2 Frame transmission

The individual MAC entity associated with each bridge port transmits frames submitted to it by the MAC relay entity.

Relayed frames for transmission by the forwarding process are submitted to the RPR ISS. The M_UNITDATA.request primitive associated with such frames conveys the values of the client MAC source and destination address fields received in the corresponding M_UNITDATA.indication primitive.

LLC protocol data units (PDUs) are submitted by LLC as a user of the MAC service provided by the bridge port. Frames transmitted to convey such PDUs carry the individual client MAC address of the port in the source address field. All LLC PDUs are submitted to the RPR ISS. The RPR ISS in turn performs the same client MAC destination address to RPR *destinationStationID* and *destinationAddress* mapping as described for frames submitted to the RPR ISS from the MAC relay entity.

Each frame is transmitted subject to the following procedure associated with the RPR MAC technology. The values of the frameType and macAction parameters of the corresponding M_UNIT-DATA. request primitive shall be user_data_frame and request_with_no_response, respectively (6.5).

The client MAC destination address is used by the RPR ISS mapping function to determine the RPR *destinationStationID* (DSID) and *destinationAddress* used in the RPR frame header of the transmitted frame.

1) If the client MAC destination address is found in the RPR ISS mapping table, the associated RPR *destinationStationID* and *ringletID* are extracted from the table; these provide for destination stripping of the unicast frame and shortest-path routing. This station's *sourceStationID* is included in the header. The RPR *destinationMacAddress* and *sourceMacAddress* fields are copies of the client MAC *destinationAddress* and *sourceAddress* fields respectively.
2) If the client MAC destination address is not found in the mapping table, two frames are created. Within these frames, this station's *sourceStationID* is included in the header. The RPR *destinationMacAddress* and *sourceMacAddress* fields are copies of the client MAC *destinationAddress* and *sourceAddress* fields respectively. Other parameters are different within each of these frames, as follows:
    a) The *destinationStationID* is set to identify bridge0 and ringletID is set to 0 (bridge0 may be any bridge station located on ring0).
    b) The *destinationStationID* is set to identify bridge1 and ringletID is set to 1 (bridge1 shall be the last bridge before bridge0).

    There are several acceptable degenerate cases where only one frame is sent, as follows:
    The frame is sent on ring0 and *destinationStationID* equals *sourceStationID*.
    The frame is sent on ring1 and *destinationStationID* equals *sourceStationID*.
    The frame is sent on ring0 and *destinationStationID* identifies the last reachable bridge.
    The frame is sent on ring1 and *destinationStationID* identifies the last reachable bridge.
3) All broadcast and multicast type frames set the *flood* bit in the RPR header. The frame contents are otherwise the same as specified in (2).

The RPR *sourceStationID* (SSID) in the transmitted frame shall always be set to the transmitting station's source station ID. This parameter is used to invoke source stripping at the receiver, which allows the

receiver to learn the association of *sourceStationID* with client MAC *sourceAddress* in received frames. This knowledge should then be used to efficiently direct the expected unicast response frames to the client.

Frames transmitted following a request by the LLC user of the MAC service provided by the bridge port shall also be submitted to the MAC relay entity.

NOTE—Maintaining ordering sometimes mandates flushing of in-flight packets during protection events; see Clause 10 for details.

B.4.3 Frame reception

The individual MAC entity associated with each bridge port examines all frames received on the RPR ringlet to which it is attached. The RPR *destinationStationID* and *sourceStationID* affect where the packet is stripped; the *destinationMacAddress* and *sourceMacAddress* affect how packets are processed; see 9.1 for details.

All error-free received frames are passed to the RPR ISS give rise to M_UNITDATA indication primitives which shall be handled as follows:

A frame that is in error, as defined by the relevant MAC specification, is discarded by the MAC entity without giving rise to any M_UNITDATA indication; see 6.4.

The receiving station's receive procedure updates its mapping table with the client MAC source address, its associated VID (if available), and the RPR *sourceStationID* address from the RPR frame header. The RPR ISS provides the M_UNITDATA indication primitive, *frameType* and *macAction* parameter values of user_data_frame and request_with_no_response respectively to the learning and forwarding processes in the MAC relay entity.

Frames with other values of *frameType* and *macAction* parameters (e.g., request_with_response and response frames), shall not be submitted to the forwarding process. They may be submitted to the learning process.

Frames with a *frameType* of user_data_frame and addressed to the bridge port as an end station shall be submitted to LLC. Such frames carry either the individual MAC address of the port or a group address associated with the port (7.12) in the destination address field. Frames submitted to LLC can also be submitted to the learning and forwarding processes, as specified above.

Frames addressed to a bridge port as an end station, and relayed to that bridge port from other bridge ports in the same bridge by the forwarding process, shall also be submitted to LLC.

No other frames shall be submitted to LLC.

Annex C: Client-to-MAC interface (normative)

C.1 Interface topologies

Both split and unified MAC implementation models are supported, as illustrated in the left and right sides of Figure C.1 respectively. Assuming 40Gbs data paths, 10Gbs status paths, and a signal-pin capacity of 1Gbs, this implies 250 and 340 signal pins for the split-MAC and unified-MAC implementations respectively.

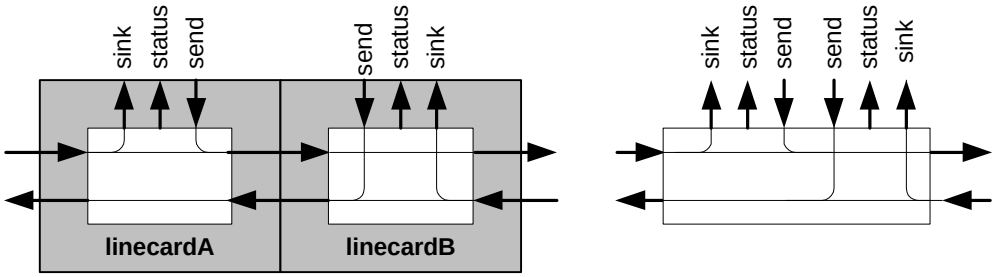


Figure C.1—MAC partitioning models

These examples illustrate the need for a high-speed, low-power, low-cost communication pipe between components. This subannex describes LiteLink, an implementation of a 10Gbs byte lane that meets these objectives. LiteLink is a byte-wide pseudo-differential signaling scheme based on a parellel-signal DC-free signal coding, developed by Cypress for connecting high-speed networking components.

Key properties of the link, when compared to existing parallel data-transfer standards include: increased speed, supply-independent voltages, and pseudo differential signaling.

C.1.2 Limited distances

The intent of this LiteLink interface is to support chip-to-chip connections on the motherboard, as illustrated in the left of Figure C.2. To maintain signal integrity, the length of the connection is limited to 20cm and no more than two connectors are assumed.

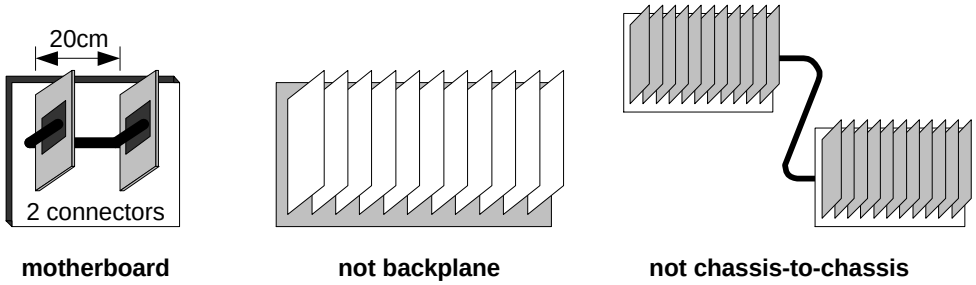


Figure C.2—Types of connections

This LiteLink proposal is not intended to support board-to-board motherboard connections or chassis-to-chassis connections, as illustrated in the center and right of Figure C.2. Such connections are expected to have distinct (and typically more expensive) driver-supply voltage and/or fiber-optic signaling requirements.

Transceivers are expected to be used in such longer-link environments. These transceivers may be responsible for voltage-level shifting and/or electro-optic conversions. Standardizing on 10Gbs unidirectional links is expected to reduce the costs of such links, although their detailed specifications are beyond the scope of this specification.

C.2 Features

The LiteLink communication pipe is based on the use of balanced code terminated logic (BCTL), a recently approved JEDEC 0.8V signaling standard. Use of a lower-voltage signaling scheme eliminates the need to support higher-voltage I/O signal levels, with their associated circuitry and gate-oxide thickness complications.

This BCTL technology is based on the transmission of 8-bit data bytes over an 11-bit wide link. Approximately half (5-or-6) of these signals are one and the remainder are zero, allowing the receiver reference voltage to be derived from the average of the observed signal values. BCTL transceivers have the following properties, which facilitates their incorporation into a wide range of VLSI technologies:

1) Balanced. Signal encoding results in nearly DC-balanced signals; the ratio of ones to zeros is either 5/6 or 6/5, depending on the encoded value, with the following benefits:
    a) Ground bounce reduction. The number of simultaneous 0-to-1 and 1-to-0 I/O signal transitions is reduced, with the effect of reducing the chip's ground bounce noise.
    b) Ground bounce insensitivity. A differential-receiver reference is implied by the averaged input signals; differential receivers using this implied reference are less sensitive to the effects of ground bounce noise.
2) Efficient. The source-synchronous pseudo-differential link is efficient:
    a) Pins. Only 11 signal pins are needed to transmit 8-bit data, clock, and control.
    b) Codes. The presence of many control codes allows packets to be efficiently framed, with only a 4-bit (as opposed to 8-bit) reduced-packet-size overhead.
3) Practical. The link is practical for a wide range of interconnect applications:
    a) Extensible. The modest 1.25Gb/s (8ns per bit) clock frequency can be doubled in the future.
    b) Tolerant. A source-synchronous strobe reduces sensitivities to data-to-clock skews.
    c) Compatible. Low-swing (0-to- 0.8V) transceivers are compatible with current and forseen VLSI design methodologies.
    d) Die size. The digital nature of the interface reduces the silicon size of pad designs.
    e) Power. Each driver dissipates less than 2.5mA per 1Gb/s transfer rate.
4) Robust. The link has built-in robustness:
    a) Self-testing. Distinct code patterns are provided for signal integrity testing.
    b) Partial parity enables detection of common transmission errors.

The transmitter encoding is stateless, in that the previously encoded data value has no effect on the encoding of the current data value, so multiple data bytes can be encoded concurrently.

C.3 LiteLink signaling

The link driver reduces the drive voltage from the chip supply the nominal .8V SLVS-400 technology-independent value; the link receiver uses a virtual center-tap voltage as a differential voltage reference, as illustrated below. The transmitter is fully specified by the SLVS-400 specification; the receiver input voltage ranges are being described in a companion SLVS-400, Class-1-BCTL addendum.

The BCTL signaling assumes 10 data signals (labeled code0 through code9) and a strobe. The strobe is complemented each cycle; data transitions occur on the rising and falling edges of the strobe.

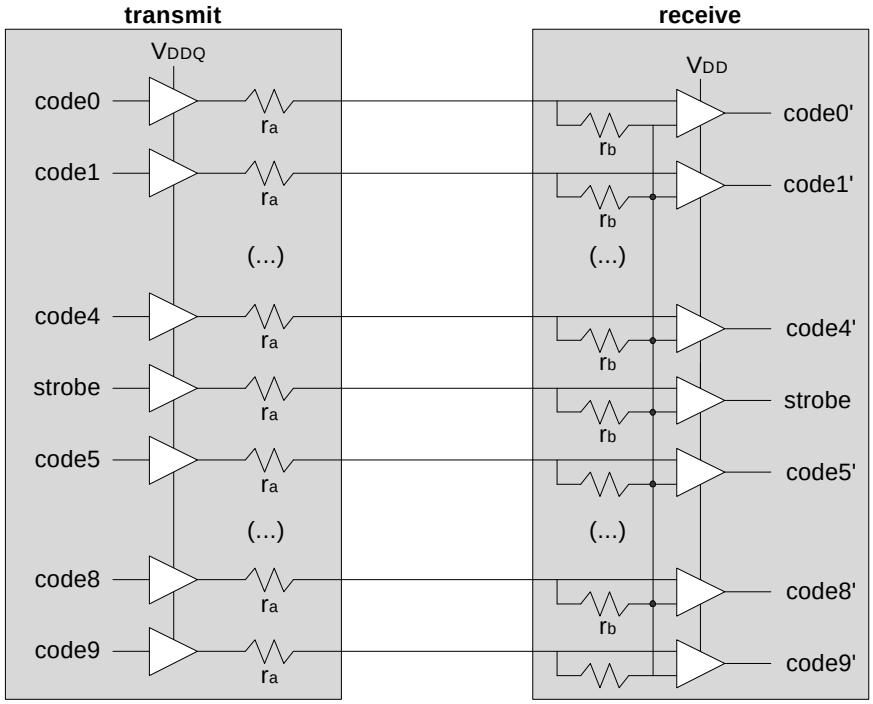


Figure C.3—Transceiver designs

C.3.2 Encoding properties

The termination scheme and signal integrity assume the transmission of 10-bit code values, where code values correspond to 8-bit data values or control codes. The data encoding has the following properties:

1) Full balanced. Signals are nearly DC balanced when measured over the 11 data and strobe pins.
2) Half balanced. Signals are nearly DC balanced when measured over either group of 5 data pins.
3) Monotonic. If *a* is greater than *b*, then *Code(a)* is greater than *Code(b)*.

C.3.3 Signal levels

LiteLink signaling is based on a .8V drive voltage, so that separate core and I/O supplies are unnecessary. To preserve signal integrity, the driver and receiver are both assumed to have a trace-matching impedances, as illustrated in the top of Figure C.4. The .8V swing of the driver is dissipated in the transmit and receive terminations, so that only a .4V swing is observed at the receiver, as illustrated in the bottom of Figure C.4.

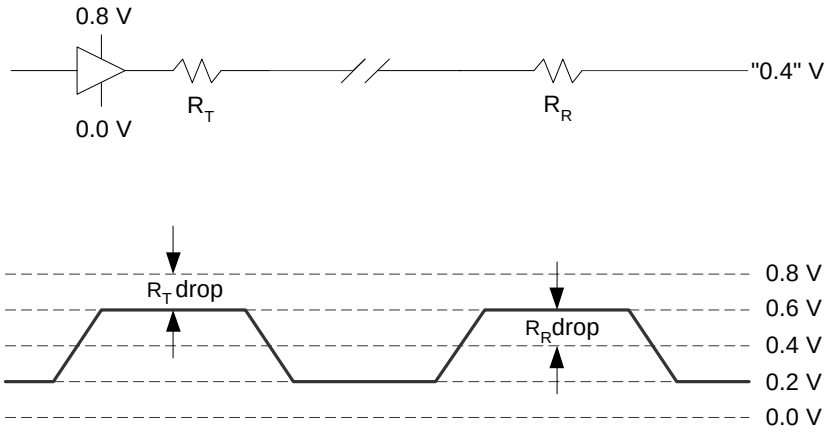


Figure C.4—Transceiver designs

The receiver's termination is approximately 0.4V, but changes by ±5% based on the transmitted signal codes, which are sometimes slightly positive (6/11 are ones) and sometimes slightly negative (5/11 are ones).

C.3.4 Data signal codings

The data encoding specification is specified by two tables, *Code0* and *Code1*, as specified in Table C.2 and in Table C.3, and applies to input data values between 0 and 255 inclusive. The remaining balanced codes are available for control, as specified in 2.3

Table C.1—Encoding summary

input	strobe==0	strobe==1
0-127	Code0(data)	~Code0(data)
128-255	Code1(data)	~Code1(data)

The '~' character signifies a complement operation, where the bit-wise boolean complement of the encoded value is sent. Note that the same coding algorithm is used in both strobe cycles, but the code values are selectively complemented in every other cycle.

C.3.5 LiteLink data coding

The Code0 table specifies the encoding of the lower valued data-bytes, as specified in Table C.2.

Table C.2—Code0 data coding

data	code	data	code	data	code	data	code
00000000	00101-1-00011	00100000	00110-1-10011	01000000	01001-1-10101	01100000	01011-1-10001
00000001	00101-1-00101	00100001	00110-1-10100	01000001	01001-1-10110	01100001	01011-1-10010
00000010	00101-1-00110	00100010	00110-1-10101	01000010	01001-1-11000	01100010	01011-1-10100
00000011	00101-1-00111	00100011	00110-1-10110	01000011	01001-1-11001	01100011	01011-1-11000
00000100	00101-1-01001	00100100	00110-1-11000	01000100	01001-1-11010	01100100	01100-1-00011
00000101	00101-1-01010	00100101	00110-1-11001	01000101	01001-1-11100	01100101	01100-1-00101
00000110	00101-1-01011	00100110	00110-1-11010	01000110	01010-1-00011	01100110	01100-1-00110
00000111	00101-1-01100	00100111	00110-1-11100	01000111	01010-1-00101	01100111	01100-1-00111
00001000	00101-1-01101	00101000	00111-1-00011	01001000	01010-1-00110	01101000	01100-1-01001
00001001	00101-1-01110	00101001	00111-1-00101	01001001	01010-1-00111	01101001	01100-1-01010
00001010	00101-1-10001	00101010	00111-1-00110	01001010	01010-1-01001	01101010	01100-1-01011
00001011	00101-1-10010	00101011	00111-1-01001	01001011	01010-1-01010	01101011	01100-1-01100
00001100	00101-1-10011	00101100	00111-1-01010	01001100	01010-1-01011	01101100	01100-1-01101
00001101	00101-1-10100	00101101	00111-1-01100	01001101	01010-1-01100	01101101	01100-1-01110
00001110	00101-1-10101	00101110	00111-1-10001	01001110	01010-1-01101	01101110	01100-1-10001
00001111	00101-1-10110	00101111	00111-1-10010	01001111	01010-1-01110	01101111	01100-1-10010
00010000	00101-1-11000	00110000	00111-1-10100	01010000	01010-1-10001	01110000	01100-1-10011
00010001	00101-1-11001	00110001	00111-1-11000	01010001	01010-1-10010	01110001	01100-1-10100
00010010	00101-1-11010	00110010	01001-1-00011	01010010	01010-1-10011	01110010	01100-1-10101
00010011	00101-1-11100	00110011	01001-1-00101	01010011	01010-1-10100	01110011	01100-1-10110
00010100	00110-1-00011	00110100	01001-1-00110	01010100	01010-1-10101	01110100	01100-1-11000
00010101	00110-1-00101	00110101	01001-1-00111	01010101	01010-1-10110	01110101	01100-1-11001
00010110	00110-1-00110	00110110	01001-1-01001	01010110	01010-1-11000	01110110	01100-1-11010
00010111	00110-1-00111	00110111	01001-1-01010	01010111	01010-1-11001	01110111	01100-1-11100
00011000	00110-1-01001	00111000	01001-1-01011	01011000	01010-1-11010	01111000	01101-1-00011
00011001	00110-1-01010	00111001	01001-1-01100	01011001	01010-1-11100	01111001	01101-1-00101
00011010	00110-1-01011	00111010	01001-1-01101	01011010	01011-1-00011	01111010	01101-1-00110
00011011	00110-1-01100	00111011	01001-1-01110	01011011	01011-1-00101	01111011	01101-1-01001
00001100	00110-1-01101	00111100	01001-1-10001	01001100	01011-1-00110	01101100	01101-1-01010
00011101	00110-1-01110	00111101	01001-1-10010	01011101	01011-1-01001	01111101	01101-1-01100
00011110	00110-1-10001	00111110	01001-1-10011	01011110	01011-1-01010	01111110	01101-1-10001
00011111	00110-1-10010	00111111	01001-1-10100	01011111	01011-1-01100	01111111	01101-1-10010

Table C.3—Code1 data coding (continued)

data	code	data	code	data	code	data	code
10000000	01101-1-10100	10100000	10010-1-00011	11000000	10100-1-00110	11100000	10110-1-01010
10000001	01101-1-11000	10100001	10010-1-00101	11000001	10100-1-00111	11100001	10110-1-01100
10000010	01110-1-00011	10100010	10010-1-00110	11000010	10100-1-01001	11100010	10110-1-10001
10000011	01110-1-00101	10100011	10010-1-00111	11000011	10100-1-01010	11100011	10110-1-10010
10000100	01110-1-00110	10100100	10010-1-01001	11000100	10100-1-01011	11100100	10110-1-10100
10000101	01110-1-01001	10100101	10010-1-01010	11000101	10100-1-01100	11100101	10110-1-11000
10000110	01110-1-01010	10100110	10010-1-01011	11000110	10100-1-01101	11100110	11000-1-00011
10000111	01110-1-01100	10100111	10010-1-01100	11000111	10100-1-01110	11100111	11000-1-00101
10001000	01110-1-10001	10101000	10010-1-01101	11001000	10100-1-10001	11101000	11000-1-00110
10001001	01110-1-10010	10101001	10010-1-01110	11001001	10100-1-10010	11101001	11000-1-00111
10001010	01110-1-10100	10101010	10010-1-10001	11001010	10100-1-10011	11101010	11000-1-01001
10001011	01110-1-11000	10101011	10010-1-10010	11001011	10100-1-10100	11101011	11000-1-01010
10001100	10001-1-00011	10101100	10010-1-10011	11001100	10100-1-10101	11101100	11000-1-01011
10001101	10001-1-00101	10101101	10010-1-10100	11001101	10100-1-10110	11101101	11000-1-01100
10001110	10001-1-00110	10101110	10010-1-10101	11001110	10100-1-11000	11101110	11000-1-01101
10001111	10001-1-00111	10101111	10010-1-10110	11001111	10100-1-11001	11101111	11000-1-01110
10010000	10001-1-01001	10110000	10010-1-11000	11010000	10100-1-11010	11110000	11000-1-10001
10010001	10001-1-01010	10110001	10010-1-11001	11010001	10100-1-11100	11110001	11000-1-10010
10010010	10001-1-01011	10110010	10010-1-11010	11010010	10101-1-00011	11110010	11000-1-10011
10010011	10001-1-01100	10110011	10010-1-11100	11010011	10101-1-00101	11110011	11000-1-10100
10010100	10001-1-01101	10110100	10011-1-00011	11010100	10101-1-00110	11110100	11000-1-10101
10010101	10001-1-01110	10110101	10011-1-00101	11010101	10101-1-01001	11110101	11000-1-10110
10010110	10001-1-10001	10110110	10011-1-00110	11010110	10101-1-01010	11110110	11000-1-11000
10010111	10001-1-10010	10110111	10011-1-01001	11010111	10101-1-01100	11110111	11000-1-11001
10011000	10001-1-10011	10111000	10011-1-01010	11011000	10101-1-10001	11111000	11000-1-11010
10011001	10001-1-10100	10111001	10011-1-01100	11011001	10101-1-10010	11111001	11000-1-11100
10011010	10001-1-10101	10111010	10011-1-10001	11011010	10101-1-10100	11111010	11001-1-00011
10011011	10001-1-10110	10111011	10011-1-10010	11011011	10101-1-11000	11111011	11001-1-00101
10001100	10001-1-11000	10111100	10011-1-10100	11001100	10110-1-00011	11101100	11001-1-00110
10011101	10001-1-11001	10111101	10011-1-11000	11011101	10110-1-00101	11111101	11001-1-01001
10011110	10001-1-11010	10111110	10100-1-00011	11011110	10110-1-00110	11111110	11001-1-01010
10011111	10001-1-11100	10111111	10100-1-00101	11011111	10110-1-01001	11111111	11001-1-01100

C.3.6 LiteLink control coding

In addition to the data codes, 44 balanced codes are available for control, as specified in Table C.4. Within this and previous tables, the '~' character signifies a complement operation

Table C.4—Control code summary

control	strobe==0	strobe==1
256-299	~Code2(data)	Code2(data)

The Code2 table specifies the encoding of control-code values, as specified in Table C.5.

Table C.5—Code2 control coding

control	code	control	code	control	code
100000000	11001-1-10001	100010000	00011-1-00110	100100000	00011-1-11010
100000001	11001-1-10010	100010001	00011-1-00111	100100001	00011-1-11100
100000010	11001-1-10100	100010010	00011-1-01001	100100010	11100-1-00011
100000011	11001-1-11000	100010011	00011-1-01010	100100011	11100-1-00101
100000100	11010-1-00011	100010100	00011-1-01011	100100100	11100-1-00110
100000101	11010-1-00101	100010101	00011-1-01100	100100101	11100-1-01001
100000110	11010-1-00110	100010110	00011-1-01101	100100110	11100-1-01010
100000111	11010-1-01001	100010111	00011-1-01110	100100111	11100-1-01100
100001000	11010-1-01010	100011000	00011-1-10001	100101000	11100-1-10001
100001001	11010-1-01100	100011001	00011-1-10010	100101001	11100-1-10010
100001010	11010-1-10001	100011010	00011-1-10011	100101010	11100-1-10100
100001011	11010-1-10010	100011011	00011-1-10100	100101011	11100-1-11000
100001100	11010-1-10100	100011100	00011-1-10101		
100001101	11010-1-11000	100011101	00011-1-10110		
100001110	00011-1-00011	100011110	00011-1-11000		
100001111	00011-1-00101	100011111	00011-1-11001		

Within this table, the following control code assignments apply: The allocation of 16 control codes to flags allows a 4-bit *flag* indication and a 4-bit *code* value to be collocated within the same control byte. The allocation of 16 control codes to marks allows a 4-bit *mark* indication and a 4-bit *addressHi* component to be collocated within the same control byte.

Table C.6—Special control coding

control	code	Name
100000000-100001111	—	flags
100010000-100011111	—	marks
100100000	00011-1-11010	idleA0
100100001	00011-1-11100	idleB0
100100010	11100-1-00011	idleB1
100100011	11100-1-00101	idleA1
	(others)	reserved

C.4 Transmission widths

C.4.1 Narrow-width transmissions

Although specified as a 4-byte-lane connection, LiteLink can be configured to transmit as 1-byte or 2-byte sequences, as illustrated in Figure C.5. The use of distinct idle symbols (idleA0/A1 or idleB0/B1) within each cycle allows a 4-byte-lane link to be readily identified as two 2-byte-wide ports or four 1-byte-wide ports.

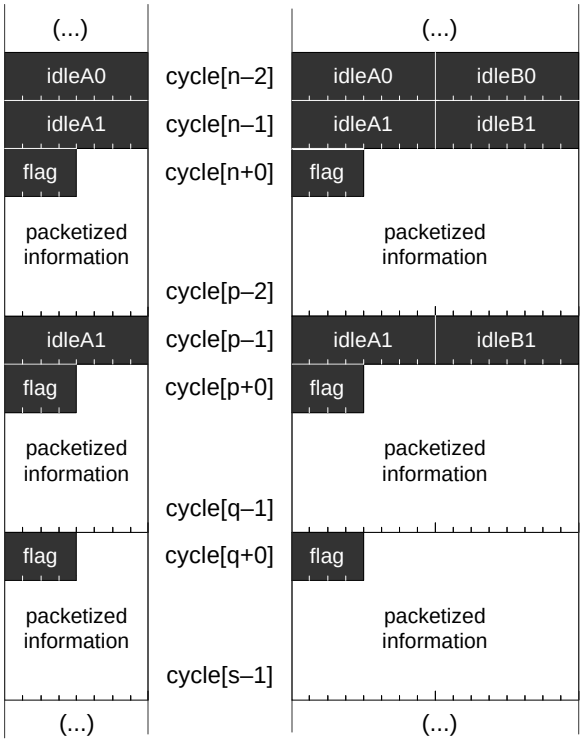


Figure C.5—Narrow-width transmissions

The 4-bit *flag* is located at the start of each packet. The remaining fields within the packet are TBD.

C.4.2 Packet framing

A nominal LiteLink connection transmits data as sequences of 4-byte words, as illustrated in Figure C.6. In even (*stobe* is 0) cycles, idles consist of ***idle0*** characters;s in odd (*stobe* is 1) cycles, idles consist of ***idle1*** characters. The assignment of {*idleA0,idleA1*} and {*idleB0,idleB1*} values is such that only the strobe line changes between successive transmissions of these paired idle-control-symbol characters.

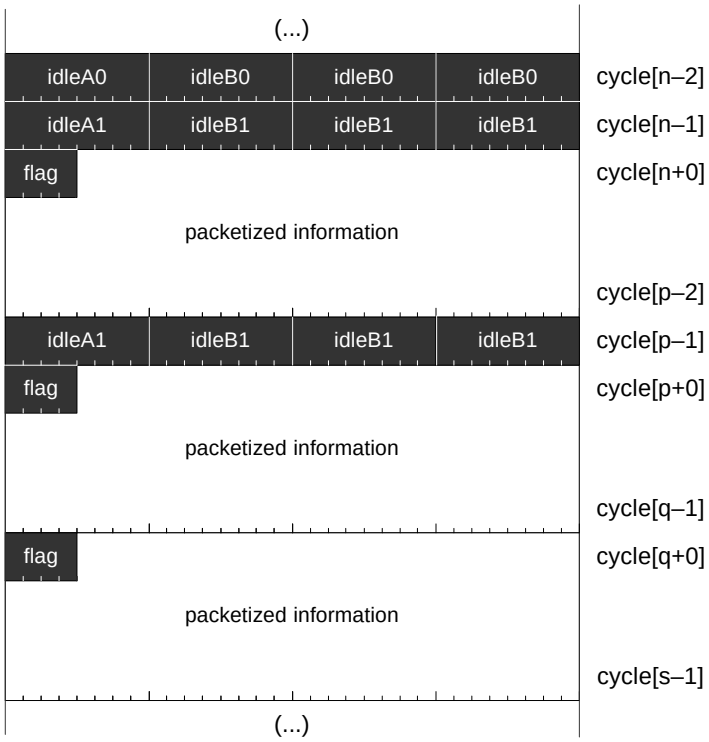


Figure C.6—Nominal packet-width framing

The 4-bit *flag* is located at the start of each packet. The remaining fields within the packet are TBD.

C.4.3 Wide transmission widths

Although specified as a 4-byte-lane connection, LiteLink can be configured to transmit sequences of 8-byte words, as illustrated in Figure C.7. The use of distinct idle symbols (idleA0/A1 or idleB0/B1) within each cycle allows an 8-byte-lane link to be readily distinguished from collections of 4-byte-wide, 2-byte-wide, and 1-byte-wide lanes.

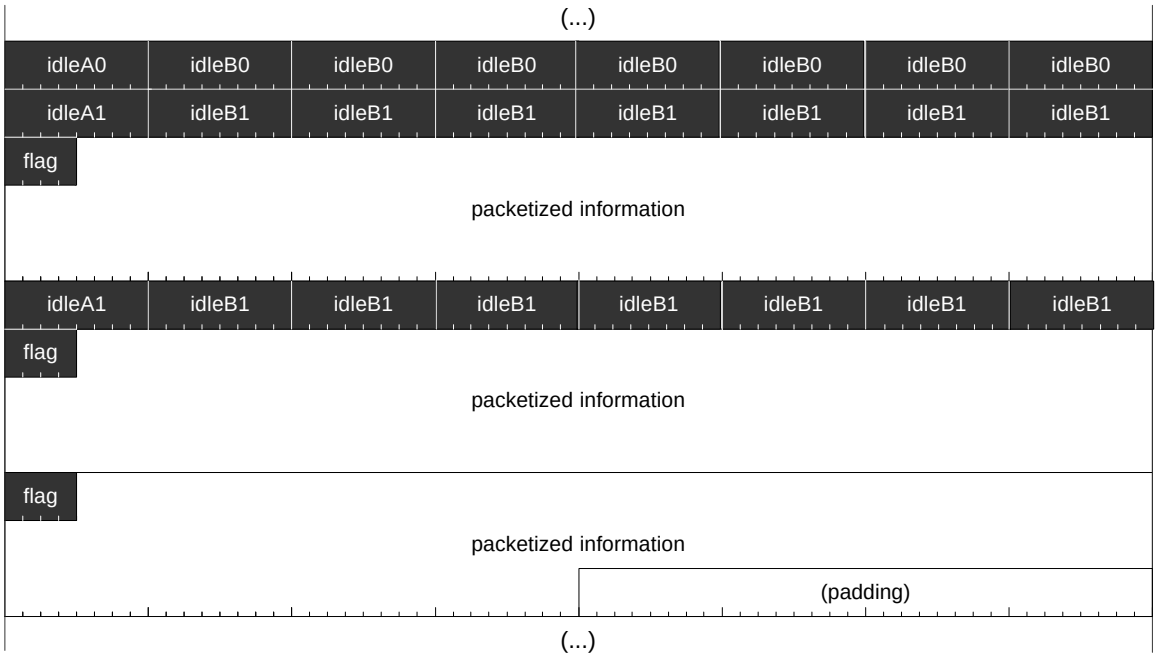


Figure C.7—Wide transmission formats

The 4-bit *flag* is located at the start of each packet. When necessary, packets are padded to maintain 8-byte alignment over the eight byte lanes. The remaining fields within the packet are TBD.

Annex D: Physical layer services (normative)

D.1 Idle symbol pools

Each station is expected to have its own clock, with ±100ppm or less accuracy (as specified by the physical layer). This ensures approximately equal rates between one station's transmitter and another node's receiver. However, small frequency differences accumulate and symbols must eventually be dropped (if the transmitter runs faster than the receiver) or inserted (if the transmitter runs slower than the receiver).

Symbol insertion and deletion is delayed during packet transmissions, so that only idle between-packet symbols are affected. Idle-symbol insertion and deletion occurs in what is commonly referred to as an elasticity buffer, a small FIFO structure with external insert/delete controls. The elasticity buffer must be sufficiently sized to avoid overrun or underrun during large packet transmissions.

Because of its influence of packet-transfer efficiencies, only a small number of between-packet idles are guaranteed by the MAC-level transmission protocol. Management of these between-packet idles is required to avoid empty elasticity buffers when near simultaneous idle deletions are performed by consecutive stations. This subannex describes the management of idle-symbol pools, with these effects in mind.

D.1.1 Elasticity buffer usage

Each station has elasticity buffers that may insert or delete passing-through idle symbols, based on the differences between their clock frequencies and the clock frequencies of their transmitting neighbors. To ensure successful operation of these elasticity buffers, idle symbols are managed in several ways, listed below:

1) Sending. When transmitting frames, a replicated idle (in addition to the frame's EOF idle) is periodically sent (this effectively extends the length of some frame by an additional idle symbol).
2) Recovering. While emptying a downstream bypass FIFO, a periodic stream of deletable idles passes through.
3) Adjustments. Elasticity buffers attempt to insert or delete idles, to maintain an approximately constant distance between deletable idles.

The default distances between idles can increase if elasticity buffers in multiple stations concurrently delete passing-through idles. This has the temporary undesirable effect of temporarily starving other elasticity buffer from desired deletable idles, and could (if allowed) cause elasticity buffer failures.

D.1.2 Elasticity buffer pacing

To minimize the worst-case idle-frame starvation conditions, the dynamics of the elasticity buffer are constrained, as illustrated in Figure D.1. Desired deletions are normally deferred until consecutive deletable idles are encountered, or after a significant increase from the nominal elasticity depth. Similarly, desired insertions are normally deferred until an isolated nondeletable idle is encountered, or after a significant decrease from the nominal elasticity depth.

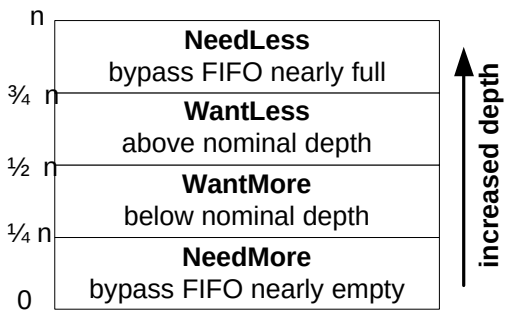


Figure D.1—Elasticity delay-adjustment ranges

The elasticity buffer behaves as a variable-delay storage element. Its idle-insertion and idle-deletion decisions are based on the current elastic-buffer delay, as follows:

1) **NeedMore.** When the delays are short, idles are inserted at the next opportunity. The intent is to quickly restore the nominal elasticity delay.
2) **WantMore.** When the delays are shorter, idle insertions are performed if IDLE_SPACING symbols have been received since the last passed-through deletable idle. The intent is to create an evenly-spaced set of deletable idles.
3) **WantLess.** When delays are longer, idle deletions are performed if IDLE_SPACING symbols have been received since the last passing-through deletable idle. The intent is to maintain an evenly-spaced set of deletable idles.
4) **NeedLess.** When the delays are very long, the next deletable idle is deleted. The intent is to quickly restore the nominal elasticity delay.

The IDLE_SPACING constant value is physical-layer dependent and beyond the scope of this standard.

D.2 Wallclock synchronization

Wallclock synchronization involves the tight synchronization of timers maintained on clock-master and clock-slave stations. The intent is to provide uniform “stratum” clocks, to enable synchronization of source and destination devices, to avoid data slips or gaps during the distribution and/or presentation of real-time information, such as telephony traffic.

Some physical layers, such as SONET provide stratum clock services. This subannex describes how these services may be provided at higher layers within other less-supportive physical layers.

D.2.1 Wallclock calibration

With bidirectional cables, the clockSync transmissions can account for the constant cable-induced delays, by measuring round-trip cable delays. Using such techniques, the accuracy of these wallclock synchronization protocols is dependent on the delay differences between incoming and outgoing links, not the overall delay of either. Implementation of these wallclock synchronization protocols involves monitoring the arrival and departure time of specialized class-A frames, called clockSync frames, as described in this subclause.

The root station is responsible for generation of clockSync frames. All stations (root as well as nonroot) are responsible for measuring the clockSync propagation time through themselves. Clock deviations are sampled in cycle N and calibrations are performed in cycle N+1. Clock sampling involves through-station delays measurements and sampling of the station’s *clockTime* value at its transmitter, as illustrated in Figure D.2.

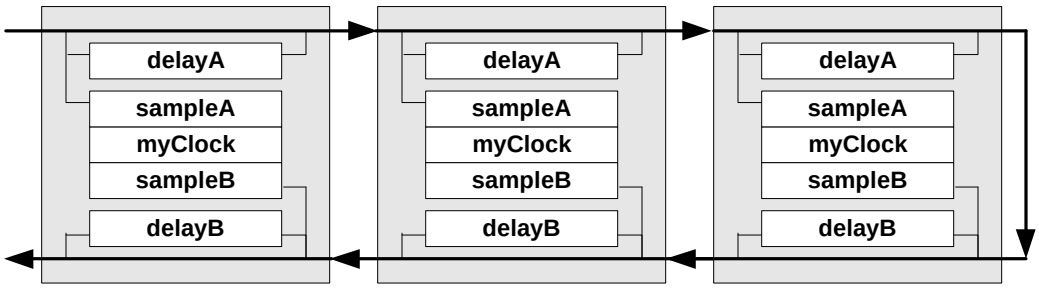


Figure D.2—Clock and delay measurements

The behavior on synchronization protocols is as follows:

1) Deviation. The station computes its clock deviation, as follows:
timeSink= (sampleA+sampleB+delayB)/2;
timeDiff= timeSend–timeSink;

2) Core. The core sends the average of the observed right-side times, as follows:
timeSendOut= timeSendIn+(*delayA–delayB*)/2

D.2.2 Wallclock adjustments

The wallclock measurements in cycle N are used to adjust the clock-slave wallclock values in cycle N+1, as specified in equation 2. Initial synchronization involves setting the station's *clockTime* value, to minimize the clock-value lock-up delays. Maintaining synchronization involves clock-rate adjustments, to avoid *clockTime* discontinuities.

```
#define THRESHOLD ONE_SECOND/8000       // Adjust after 8KHz interval
#define TICK (CLOCK_NOMINAL/5000)       // 200PPM overcomes 100PPM
inaccuracy
delta= timeSink-timeSend;
if (Magnitude(delta)>(THRESHOLD/2))
  clockTime+= delta;
else
  clockRate+= difference>0 ? TICK:-TICK;
```

Annex E: Parallel CRC calculations (informative)

E.1 Cyclic redundancy check (CRC)

E.1.1 Algorithmic definition

There is a 32-bit check symbol at the end of the packet header and payload. For good error coverage, a cyclic redundancy code (CRC) is used. The CRC efficiently detects errors but does not correct errors. Error recovery is performed at a higher level.

The CRCs that is used is the same CRC used by IEEE 802 LANs and FDDI. The CRC uses the generator polynomial of equation xx. Which is performed on the most significant bits first:

$$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{2}+x+1$$

E.1.2 Serial CRC calculation

The serial implementation of the CRC-32 polynomial, as applied to the most- through least-significant bits, is specified by the C-code calculations of Table E.1 and the hardware implementation illustrated in figurexx.

Table E.1—Serial CRC-32 computations

```
// c00-through-c31 are the most- through least-significant bits of check.
// d00 is the input value, sum is an intermediate value.
Sum= c00^d00;
c00= c01;        c01= c02;         c02= c03;        c03= c04;
c04= c05;        c05= c06^sum;     c06= c07;        c07= c08;
c08= c09^sum;    c09= c10^sum;     c10= c11;        c11= c12;
c12= c13;        c13= c12;         c14= c15;        c15= c16^sum;
c16= c17;        c17= c18;         c18= c19;        c19= c20^sum;
c20= c21^sum;    c21= c22^sum;     c22= c23;        d23= c24^sum;
c24= c25^sum;    c25= c26;         c26= c27^sum;    c27= c28^sum;
c28= c29;        c29= c30^sum;     c30= c31^sum;    c31= sum;
```

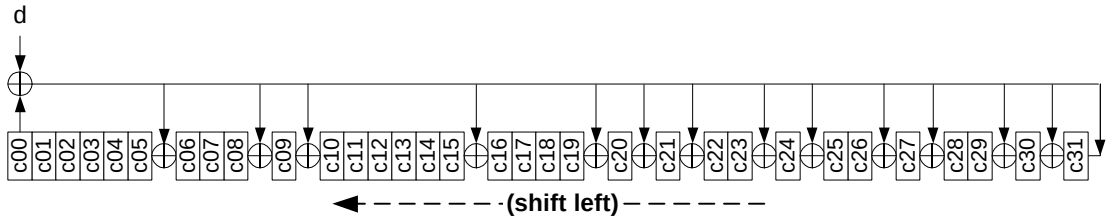


Figure E.1—Serial crc32 reference model

The CRC calculation has several somtimes subtle characteristics, in addition to the basic polynomial-based CRC calculations, that could produce non-standard results if implemented differently. For the benefit of the casual reader, and to reduce the possibility of creating non-standard implementations, these characteristics are summarized below:

1) Startup. The CRC calculations start with an all-ones crcSum value.
2) Complete. The computed CRC value is complemented before being appended to the packet.

E.2 Arranged-CRC calculations

Several interconnects send bits in a most- through least-significant ordering. For such interconnects, the CRC calculation is based on this ordering assumption and described in the remainder of this subclause.

If the CRC becomes a MAC level definition, this would be the proposed algorithm. If the CRC becomes a PHY level definition, then this would be one of the physical layer definitions.

E.2.1 Arranged ExorSum calculations

The generation and checking of 32-bit CRC values, optimized for bit-sequential transmission, is illustrated in Figure E.2.

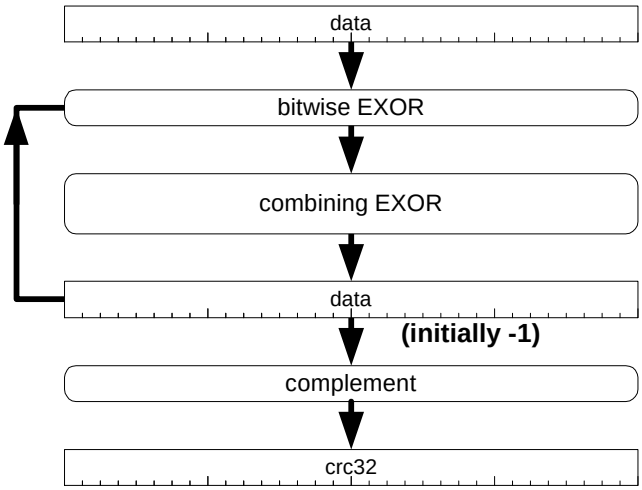


Figure E.2—Arranged ExorSum calculations

The CRC-generation code of B.1.2 can be called to generate CRC-computation tables, using the C program documented in Annex E. This program supports the creation of tables for performing parallel CRC checks, where 1, 2, 4, 8, 16, or 32 data bits are processed in parallel. Computer generation of the CRC-table text, rather than their values, minimized the possibility of introducing errors in the documentation process.

E.2.2 Arranged ExorSum32 equations

Although the CRC is specified as a bit-serial computation, the CRC value can be computed in parallel. This is important for RPR, because CRCs have to be checked and regenerated at full RPR speed. Parallelizing the serial specification, to process 32 data bits in parallel, generates the equations shown in Table E.2.

Table E.2—Arranged ExorSumCrc32 equations

```
// C00-through-c31 are the most- through least-significant bits of check.
// d00-through-d31 are the most- through least-significant bits of input.
// "a".."t""A".." " are intermediate bit values.
a= c00^d00;   b= c01^d01;   c= c02^d02;   d= c03^d03;
e= c04^d04;   f= c05^d05;   g= c06^d06;   h= c07^d07;
j= c08^d08;   k= c09^d09;   m= c10^d10;   n= c11^d11;
p= c12^d12;   r= c13^d13;   s= c14^d14;   t= c15^d15;
A= c16^d16;   B= c17^d17;   C= c18^d18;   D= c19^d19;
E= c20^d20;   F= c21^d21;   G= c22^d22;   H= c23^d23;
J= c24^d24;   K= c25^d25;   M= c26^d26;   N= c27^d27;
P= c28^d28;   R= c29^d29;   S= c30^d30;   T= c31^d31;
//                      1                   2                   3
//   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
//   a b c d e f g h j k m n p r s t A B C D E F G H J K M N P R S T
c00= a^b^c^d^e^  g^h^j^             A^      E^  G^H^    M          ;
c01=   b^c^d^e^f^  h^j^k^             B^      F^  H^J^    N        ;
c02= a^  c^d^e^f^g^  j^k^m^             C^      G^  J^K^    P      ;
c03=   b^  d^e^f^g^h^  k^m^n^             D^      H^  K^M^    R    ;
c04=     c^  e^f^g^h^j^  m^n^p^             E^      J^  M^N^    S  ;
c05= a^    d^  f^g^h^j^k^  n^p^r^             F^      K^  N^P^    T;
c06= a^  c^d^          k^m^  p^r^s^  A^      E^    H^        P^R    ;
c07=   b^  d^e^          m^n^  r^s^t^  B^      F^    J^        R^S  ;
c08= a^  c^  e^f^          n^p^  s^t^A^  C^      G^    K^        S^T;
c09= a^  c^  e^f^  h^j^      p^r^  t^  B^  D^E^  G^                T;
c10= a^  c^  e^f^  h^  k^      r^s^      C^    F^G^      M          ;
c11=   b^  d^  f^g^  j^  m^      s^t^      D^    G^H^      N        ;
c12=     c^  e^  g^h^  k^  n^      t^A^      E^    H^J^      P      ;
c13= a^    d^  f^  h^j^  m^  p^      A^B^      F^    J^K^      R    ;
c14= a^b^    e^  g^  j^k^  n^  r^      B^C^      G^    K^M^      S  ;
c15=   b^c^    f^  h^  k^m^  p^  s^      C^D^      H^    M^N^      T;
c16=   b^    e^    h^    m^n^  r^  t^A^    D^    G^H^J^  M^N^P      ;
c17=     c^    f^    j^    n^p^  s^  A^B^    E^    H^J^K^  N^P^R    ;
c18= a^    d^    g^    k^    p^r^  t^  B^C^    F^    J^K^M^  P^R^S  ;
c19= a^b^    e^    h^    m^    r^s^  A^  C^D^    G^    K^M^N^  R^S^T;
c20= a^    d^e^f^g^h^      n^    s^t^A^B^  D^    G^        N^P^  S^T;
c21= a^  c^d^  f^            p^    t^  B^C^      G^      M^  P^R^  T;
c22=     c^        h^j^        r^        C^D^E^  G^      M^N^  R^S  ;
c23= a^    d^        j^k^        s^        D^E^F^  H^      N^P^  S^T;
c24=     c^d^    g^h^j^k^m^        t^A^        F^  H^J^  M^  P^R^  T;
c25=   b^c^      g^    k^m^n^          B^    E^    H^J^K^M^N^  R^S  ;
c26=     c^d^      h^    m^n^p^          C^    F^    J^K^M^N^P^  S^T;
c27= a^b^c^      g^h^      n^p^r^    A^    D^E^    H^  K^  N^P^R^  T;
c28= a^      e^  g^          p^r^s^  A^B^      F^G^H^J^      P^R^S  ;
c29= a^b^      f^  h^          r^s^t^  B^C^      G^H^J^K^      R^S^T;
c30=       d^e^    h^            s^t^    C^D^E^  G^  J^K^        S^T;
c31= a^b^c^d^  f^g^h^              t^      D^  F^G^    K^          T;
```

E.2.3 Arranged ExorSumCrc16 equations

Although the CRC computation is expected to be performed in a 32-bit parallel fashion, less logic is required if 16 data bits can be processed at twice the 32-bit-symbol clock rate. Parallelizing the serial specification, to process 16 data bits in parallel, generates the equations shown in Table E.3.

Table E.3—Arranged ExorSumCrc16 equations

```
// C00-through-c31 are the most- through least-significant bits of check.
// d00-through-d15 are the most- through least-significant bits of input.
// "a".."t""A".." " are intermediate bit values.
a= c00^d00;   b= c01^d01;   c= c02^d02;   d= c03^d03;
e= c04^d04;   f= c05^d05;   g= c06^d06;   h= c07^d07;
j= c08^d08;   k= c09^d09;   m= c10^d10;   n= c11^d11;
p= c12^d12;   r= c13^d13;   s= c14^d14;   t= c15^d15;
A= c16;       B= c17;       C= c18;       D= c19;
E= c20;       F= c21;       G= c22;       H= c23;
J= c24;       K= c25;       M= c26;       N= c27;
P= c28;       R= c29;       S= c30;       T= c31;
//                      1                   2                   3
//   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
//   a b c d e f g h j k m n p r s t A B C D E F G H J K M N P R S T
c00= a^      e^  g^h^    m^          A                               ;
c01=   b^      f^  h^j^    n^          B                             ;
c02=     c^      g^  j^k^    p^          C                           ;
c03=       d^      h^  k^m^    r^          D                         ;
c04=         e^      j^  m^n^    s^          E                       ;
c05=           f^      k^  n^p^    t^          F                     ;
c06= a^      e^    h^        p^r^              G                   ;
c07=   b^      f^    j^        r^s^                H               ;
c08= a^  c^      g^    k^        s^t^                J             ;
c09=   b^  d^e^  g^                t^                  K           ;
c10=     c^    f^g^      m^                            M          ;
c11=       d^    g^h^      n^                            N        ;
c12= a^      e^    h^j^      p^                            P      ;
c13= a^b^      f^    j^k^      r^                            R    ;
c14=   b^c^      g^    k^m^      s^                            S  ;
c15=     c^d^      h^    m^n^      t^                            T;
c16= a^    d^    g^h^j^  m^n^p                                   ;
c17= a^b^    e^    h^j^k^  n^p^r                                 ;
c18=   b^c^    f^    j^k^m^  p^r^s                               ;
c19= a^  c^d^    g^    k^m^n^  r^s^t                             ;
c20= a^b^  d^    g^        n^p^  s^t                             ;
c21=   b^c^      g^      m^  p^r^  t                             ;
c22=     c^d^e^  g^      m^n^  r^s                               ;
c23=       d^e^f^  h^      n^p^  s^t                             ;
c24= a^        f^  h^j^  m^  p^r^  t                             ;
c25=   b^    e^    h^j^k^m^n^  r^s                               ;
c26=     c^    f^    j^k^m^n^p^  s^t                             ;
c27= a^    d^e^    h^  k^  n^p^r^  t                             ;
c28= a^b^      f^g^h^j^      p^r^s                               ;
c29=   b^c^      g^h^j^k^      r^s^t                             ;
c30=     c^d^e^  g^  j^k^        s^t                             ;
c31=       d^  f^g^    k^          t                             ;
```

E.2.4 Arranged ExorSumCrc8 equations

The CRC computation logic can be further reduced if 8 data bits can be processed at four times the 32-bit-symbol clock rate. Parallelizing the serial specification, to process 8 data bits in parallel, generates the equations shown in Table E.4.

Table E.4—Arranged ExorSumCrc8 equations

```
// c00-through-c31 are the most- through least-significant bits of check.
// d00-through-d07 are the most- through least-significant bits of input.
// "a".."t""A".." " are intermediate bit values.
a= c00^d00;   b= c01^d01;   c= c02^d02;   d= c03^d03;
e= c04^d04;   f= c05^d05;   g= c06^d06;   h= c07^d07;
j= c08;       k= c09;       m= c10;       n= c11;
p= c12;       r= c13;       s= c14;       t= c15;
A= c16;       B= c17;       C= c18;       D= c19;
E= c20;       F= c21;       G= c22;       H= c23;
J= c24;       K= c25;       M= c26;       N= c27;
P= c28;       R= c29;       S= c30;       T= c31;
//   00                  10                  20                  30
//   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
//   a b c d e f g h j k m n p r s t A B C D E F G H J K M N P R S T
c00=     c^          j                                             ;
c01= a^    d^          k                                           ;
c02= a^b^    e^          m                                         ;
c03=   b^c^    f^          n                                       ;
c04= a^  c^d^    g^          p                                     ;
c05=   b^  d^e^    h^          r                                   ;
c06=         e^f^                s                               ;
c07= a^        f^g^                t                             ;
c08=   b^        g^h^                A                           ;
c09=               h^                  B                         ;
c10=     c^                              C                       ;
c11=       d^                              D                     ;
c12= a^      e^                              E                   ;
c13= a^b^      f^                              F                 ;
c14=   b^c^      g^                              G               ;
c15=     c^d^      h^                              H             ;
c16= a^  c^d^e^                                    J            ;
c17= a^b^  d^e^f^                                    K          ;
c18= a^b^c^  e^f^g^                                    M        ;
c19=   b^c^d^  f^g^h^                                    N      ;
c20=       d^e^  g^h^                                      P    ;
c21=     c^  e^f^  h^                                        R  ;
c22=     c^d^  f^g^                                            S;
c23=       d^e^  g^h^                                            T
c24= a^  c^  e^f^  h                                             ;
c25= a^b^c^d^  f^g                                               ;
c26= a^b^c^d^e^  g^h                                             ;
c27=   b^  d^e^f^  h                                             ;
c28= a^      e^f^g                                               ;
c29= a^b^      f^g^h                                             ;
c30= a^b^        g^h                                             ;
c31=   b^          h                                             ;
```

E.3 Exchanged CRC calculations

Several interconnects send bits in a least- through most-significant bit ordering. For such interconnects, the CRC calculation is based on this ordering assumption and described in the remainder of this subclause.

If the CRC becomes a MAC level definition, this proposed algorithm would be abandoned. If the CRC becomes a PHY level definition, then this would be one of the two physical layer definitions.

E.3.1 Exchanged ExorSum calculations

The generation and checking of 32-bit CRC values, optimized for bit-reversed transmission, is illustrated in Figure E.3. The complexity of the implementation is smaller than at first seems, since the bitSwap operations involve not circuitry and can be eliminated by encorporating their functionality within the combining-EXOR circuitry.

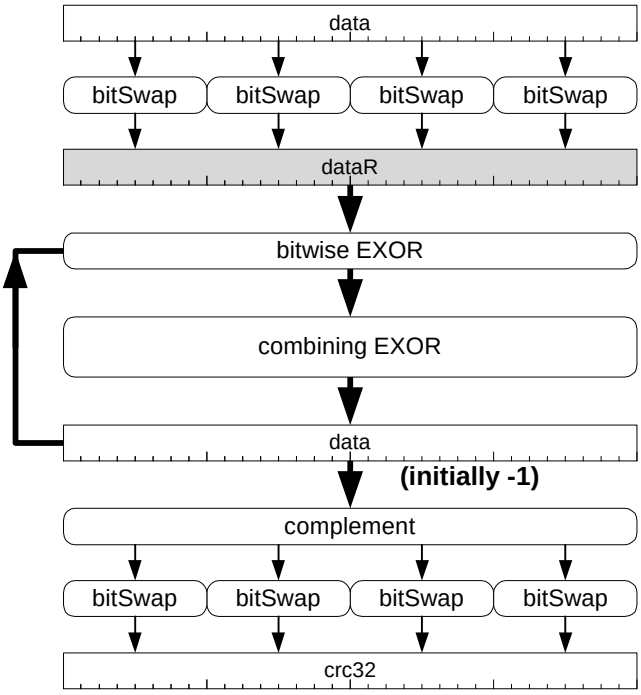


Figure E.3—exchangedExorSum calculations

The CRC-generation code of B.1.2 can be called to generate CRC-computation tables, using the C program documented in Annex E. This program supports the creation of tables for performing parallel CRC checks, where 1, 2, 4, 8, 16, or 32 data bits are processed in parallel. Computer generation of the CRC-table text, rather than their values, minimized the possibility of introducing errors in the documentation process.

E.3.2 Exchanged ExorSumCrc32 equations

Although the CRC is specified as a bit-serial computation, the CRC value can be computed in parallel. This is important for RPR, because CRCs have to be checked and regenerated at full RPR speed. Combining the bit-within-byte reversals and parallelizing the serial specification, to process 32 data bits in parallel, generates the equations shown in table B.1.

Table E.5—Exchanged ExorSumCrc32 equations

```
// C00-through-c31 are the most- through least-significant bits of check.
// d00-through-d31 are the most- through least-significant bits of input.
// "a".."t""A".." " are intermediate bit values.
a= c00^d00;   b= c01^d01;   c= c02^d02;   d= c03^d03;
e= c04^d04;   f= c05^d05;   g= c06^d06;   h= c07^d07;
j= c08^d08;   k= c09^d09;   m= c10^d10;   n= c11^d11;
p= c12^d12;   r= c13^d13;   s= c14^d14;   t= c15^d15;
A= c16^d16;   B= c17^d17;   C= c18^d18;   D= c19^d19;
E= c20^d20;   F= c21^d21;   G= c22^d22;   H= c23^d23;
J= c24^d24;   K= c25^d25;   M= c26^d26;   N= c27^d27;
P= c28^d28;   R= c29^d29;   S= c30^d30;   T= c31^d31;
//   00                  10                  20                  30
//   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
//   a b c d e f g h j k m n p r s t A B C D E F G H J K M N P R S T
c00=       d^e^  g^  j^k^m^  p^r^        C^      G^    K^M^        T;
c01=         e^f^  h^  k^m^n^  r^s^  A^    D^      H^    M^N         ;
c02= a^b^c^  e^    h^    m^n^p^  s^t^    C^          J^    N^P^  S  ;
c03= a^b^c^d^  f^          n^p^r^  t^      D^          K^    P^R^  T;
c04= a^b^c^d^e^  g^          p^r^s^  A^      E^          M^    R^S  ;
c05=   b^c^d^e^f^  h^          r^s^t^  B^      F^          N^    S^T;
c06= a^  c^d^e^f^g^              s^t^A^  C^      G^          P^    T;
c07= a^b^  d^e^f^g^h^              t^A^B^  D^      H^          R    ;
c08= a^  c^    f^g^    k^  n^  r^s^  A^      E^F^    J^      P^R    ;
c09=   b^  d^    g^h^    m^  p^  s^t^  B^      F^G^    K^      R^S  ;
c10= a^  c^  e^    h^      n^  r^  t^    C^      G^H^    M^      S^T;
c11= a^b^  d^  f^    j^      p^  s^  A^    D^      H^      N^      T;
c12=   b^c^  e^  g^  j^k^      r^  t^A^B^    E^              P      ;
c13= a^  c^d^  f^  h^  k^m^      s^    B^C^    F^              R    ;
c14= a^  c^d^  f^  h^j^  m^n^      t^  B^  D^E^  G^  J             ;
c15=     c^d^  f^  h^j^k^  n^p^        B^      F^  H^J^K^        S  ;
c16=         e^    h^  k^        s^t^A^  C^D^E^      J^K^  N^P      ;
c17= a^        f^        m^        t^  B^  D^E^F^      K^M^  P^R    ;
c18=     c^  e^f^  h^j^    n^          B^      F^G^  J^  M^N^  R    ;
c19= a^b^c^d^e^    h^j^k^    p^        B^    E^  G^H^J^K^  N^P      ;
c20= a^    d^    g^h^  k^m^    r^      B^    E^F^  H^J^K^M^  P^R^S  ;
c21=   b^    e^    h^j^  m^n^    s^      C^    F^G^    K^M^N^  R^S^T;
c22=     c^    f^      k^  n^p^    t^A^    D^    G^H^    M^N^P^  S^T;
c23= a^    d^    g^  j^  m^  p^r^    A^B^    E^    H^      N^P^R^  T;
c24= a^b^c^  e^f^g^h^j^              B^C^  E^      J^        S  ;
c25= a^    d^e^      j^k^              B^  D^E^F^    J^K^        S^T;
c26= a^  c^      g^h^j^k^m^          A^B^      F^G^  J^K^M^      S^T;
c27=   b^  d^      h^  k^m^n^        A^B^C^      G^H^  K^M^N^      T;
c28= a^b^      f^g^h^    m^n^p^      A^    D^E^    H^J^  M^N^P^  S  ;
c29= a^      e^f^          n^p^r^        C^    F^    J^K^  N^P^R^S^T;
c30=   b^      f^g^          p^r^s^  A^    D^    G^    K^M^  P^R^S^T;
c31= a^b^    e^f^    j^        r^s^t^A^  C^        H^J^  M^N^  R^  T;
```

E.3.3 Exchanged ExorSumCrc16 equations

Although the CRC computation is expected to be performed in a 32-bit parallel fashion, less logic is required if 16 data bits can be processed at twice the 32-bit-symbol clock rate. Combining the bit-within-byte reversals and parallelizing the serial specification, to process 16 data bits in parallel, generates the equations shown in table B.2.

Table E.6—Exchanged ExorSumCrc16 equations

```
// C00-through-c31 are the most- through least-significant bits of check.
// d00-through-d15 are the most- through least-significant bits of input.
// "a".."t""A".." " are intermediate bit values.
a= c00^d00;   b= c01^d01;   c= c02^d02;   d= c03^d03;
e= c04^d04;   f= c05^d05;   g= c06^d06;   h= c07^d07;
j= c08^d08;   k= c09^d09;   m= c10^d10;   n= c11^d11;
p= c12^d12;   r= c13^d13;   s= c14^d14;   t= c15^d15;
A= c16;       B= c17;       C= c18;       D= c19;
E= c20;       F= c21;       G= c22;       H= c23;
J= c24;       K= c25;       M= c26;       N= c27;
P= c28;       R= c29;       S= c30;       T= c31;
//   00                  10                  20                  30
//   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
//   a b c d e f g h j k m n p r s t A B C D E F G H J K M N P R S T
c00=     c^      g^    k^m^        t^A                               ;
c01= a^    d^      h^    m^n^          B                             ;
c02=     c^          j^    n^p^  s^      C                           ;
c03=       d^          k^    p^r^  t^      D                         ;
c04= a^      e^          m^    r^s^          E                       ;
c05=   b^      f^          n^    s^t^          F                     ;
c06= a^  c^      g^          p^    t^            G                   ;
c07= a^b^  d^      h^          r^                  H                 ;
c08= a^      e^f^    j^      p^r^                    J               ;
c09=   b^      f^g^    k^      r^s^                    K             ;
c10=     c^      g^h^    m^      s^t^                    M           ;
c11= a^    d^      h^      n^      t^                      N         ;
c12= a^b^    e^              p^                            P       ;
c13=   b^c^    f^              r^                            R     ;
c14=   b^  d^e^  g^  j^                                      S   ;
c15=   b^      f^  h^j^k^        s^                            T ;
c16= a^  c^d^e^      j^k^  n^p                                   ;
c17=   b^  d^e^f^      k^m^  p^r                                 ;
c18=   b^      f^g^  j^  m^n^  r                                 ;
c19=   b^    e^  g^h^j^k^  n^p                                   ;
c20=   b^    e^f^  h^j^k^m^  p^r^s                               ;
c21=     c^    f^g^    k^m^n^  r^s^t                             ;
c22= a^    d^    g^h^    m^n^p^  s^t                             ;
c23= a^b^    e^    h^      n^p^r^  t                             ;
c24=   b^c^  e^      j^          s                               ;
c25=   b^  d^e^f^    j^k^        s^t                             ;
c26= a^b^      f^g^  j^k^m^      s^t                             ;
c27= a^b^c^      g^h^  k^m^n^      t                             ;
c28= a^    d^e^    h^j^  m^n^p^  s                               ;
c29=     c^    f^    j^k^  n^p^r^s^t                             ;
c30= a^    d^    g^    k^m^  p^r^s^t                             ;
c31= a^  c^        h^j^  m^n^  r^  t                             ;
```

E.3.4 Exchanged ExorSumCrc8 equations

The CRC computation logic can be further reduced if 8 data bits can be processed at four times the 32-bit-symbol clock rate. Combining the bit-within-byte reversals and parallelizing the serial specification, to process 8 data bits in parallel, generates the equations shown in table B.3.

Table E.7—Exchanged ExorSumCrc8 equations

```
// c00-through-c31 are the most- through least-significant bits of check.
// d00-through-d07 are the most- through least-significant bits of input.
// "a".."t""A".." " are intermediate bit values.
a= c00^d00;   b= c01^d01;   c= c02^d02;   d= c03^d03;
e= c04^d04;   f= c05^d05;   g= c06^d06;   h= c07^d07;
j= c08;       k= c09;       m= c10;       n= c11;
p= c12;       r= c13;       s= c14;       t= c15;
A= c16;       B= c17;       C= c18;       D= c19;
E= c20;       F= c21;       G= c22;       H= c23;
J= c24;       K= c25;       M= c26;       N= c27;
P= c28;       R= c29;       S= c30;       T= c31;
//   00                  10                  20                  30
//   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
//   a b c d e f g h j k m n p r s t A B C D E F G H J K M N P R S T
c00=   b^c^        h^j                                             ;
c01=     c^d^          k                                           ;
c02= a^    d^e^  g^      m                                         ;
c03=   b^    e^f^  h^      n                                       ;
c04=     c^    f^g^          p                                     ;
c05=       d^    g^h^          r                                   ;
c06=         e^    h^            s                                 ;
c07=           f^                  t                               ;
c08= a^      e^f^                    A                             ;
c09=   b^      f^g^                    B                           ;
c10=     c^      g^h^                    C                         ;
c11=       d^      h^                      D                       ;
c12=         e^                              E                     ;
c13=           f^                              F                   ;
c14= a^                                          G                 ;
c15= a^b^        g^                                H               ;
c16= a^b^  d^e^                                      J             ;
c17=   b^c^  e^f^                                      K           ;
c18= a^  c^d^  f^                                        M         ;
c19= a^b^  d^e^                                            N       ;
c20= a^b^c^  e^f^g^                                          P     ;
c21=   b^c^d^  f^g^h^                                          R   ;
c22=     c^d^e^  g^h^                                            S ;
c23=       d^e^f^  h^                                              T;
c24= a^          g                                                 ;
c25= a^b^        g^h                                               ;
c26= a^b^c^      g^h                                               ;
c27=   b^c^d^      h                                               ;
c28= a^  c^d^e^  g                                                 ;
c29= a^b^  d^e^f^g^h                                               ;
c30=   b^c^  e^f^g^h                                               ;
c31= a^  c^d^  f^  h                                               ;
```

Annex F: Background information (informative)

F.1 Objectives summary

The primary objective of RPR is to provide enhanced services for the transmission of Ethernet frames over a ring-based interconnect topology. Secondary objectives include the following.

1) Flexible. Flexibility features, that increase the usefulness of the standard, include the following:.
   a) Partitions. RPR is easily partitioned to support distinct service level agreements (SLAs).
   b) Synchronized. Stations can be accurately synchronized to standard global timers.
   c) Ordered. Ordered delivery is maintained within each flow.
   d) Extensible. The protocols can be readily extended, in standard or vendor-dependent ways.
      i) Standard headers. Efficient standardized extended header are provided.
      ii) Dependent headers. Self-administered vendor-dependent extended headers are provided.
      iii) Payload. Larger (but less than 18 kbyte) frames are allowed.
2) Classified. Multiple classes of transport services are provided, including:
   a) Class-A. minimum latency prenegotiated bandwidth.
      Expected uses include telephony and video transfers.
   b) Class-B. bounded latency prenegotiated bandwidth.
      Subclass B0. Expected uses include deterministic SLA bandwidth partitioning.
      Subclass B1. Expected uses include statistical SLA bandwidth partitioning.
   c) Class-C. unprovisioned and unused-provisioned bandwidth.
      This residual traffic is partitioned in an approximately fair fashion.
3) Robust. The interconnect operates well in the presence of transient transmission errors:
   a) Errors. The interconnect operates well in the presence of transient transmission errors:
      i) Checked. Error-checked header and extended headers
      ii) Covered. Encapsulated Ethernet frame (including FCS)
   b) Faults. The interconnect operates well in the presence of persistent of cable failures:
      i) Recovery. Fast topology-change recovery (under 50ms)
      ii) Resilient. Ring survives in presence of single-fiber failure
   c) Plug-and-play. The interconnect operates well in the presence of cable topology changes:
      i) Changes. Nondisruptive insertion and deletion reduces the impact of on-line upgrades.
      ii) Topology. An arbitrary physical-cable topology is allowed
      (only a ring subset of the physical topology is actively enabled).
4) Scalable. The protocols should be scalable in multiple ways:
   a) Distance. RPR is applicable between within-the-home and within-the-planet distances.
   b) Bandwidth. RPR is applicable to low-rate as well as multiple gigabit/second rate media.

F.2 Objectives, requirements, and strategies

F.2.1 Compatible

Objective: Ethernet, ATM, and other protocol frames should be sent in an unmodified fashion.
Requirement: Shall be capable of encapsulating Ethernet frames.
Strategy: The 48-bit MAC addresses are encapsulated within 64-bit MAC addresses.
Standard and vendor-dependent headers enable identification of encapsulated formats.

Objective: Should be physical layer independent, allowing use of any byte transfer media.
Requirement: Shall be capable of using any byte and control transfer media.
Strategy: Control information is sent within small class-A frames.

F.2.2 Quality of service

Objective: Should support eight or more quality-of-service levels.
Requirement: Shall support three/four class-of-service categories:
1) Class-A: minimum latency guaranteed bandwidth.
2) Class-B: bounded latency guaranteed bandwidth.
3) Class-C: unprovisioned and unused provisioned bandwidth.
Strategy: Separate bypass FIFOs for class-A and class-B/class-C traffic.
Allow class-A mid-packet preemption of class-B or class-C traffic.

Objective: Should support SLA (service level agreements) of bandwidth and quality of service.
Requirement: Shall support SLA (service level agreements) of guaranteed bandwidth and latency.
Strategy: Bandwidth guarantees for provisioned low-latency and bounded-latency traffic.
Provide 1 ms latency guarantees for low-latency traffic.

Objective: Non-negotiated class-C bandwidth should be fairly allocated.
Requirement: Non-negotiable class-C traffic shall have bounded delivery delays.
Strategy: Signaling throttles station with consumption count higher than congested class-C stations.

Objective: Within each flow, all packets are transmitted and received in the same order.
Requirement: On each ringlet, packets within each flow are transmitted and received in the same order.
Strategy: Packets are transmitted and received in the same order, if they are sent over the same ringlet and have the same A/B/C class identifiers.

F.2.3 Wallclock synchronization

Objective: The clock-slave station timers can be accurately synchronized to the clock-master station.
Requirement: The clock-slave station timers do not drift from the timer in the clock-master station.
Strategy: If not otherwise supported by the physical layer, periodic time-reference frames are sent over duplex links, allowing attached stations to monitor the arrival and departure times of these frames. The clock-slave stations compensate their timers based on the differences between expected and observed times.

F.2.4 Scalable

Objective: The link should be applicable to within-the-home through within-the-planet applications.
Requirement: The link shall be applicable to within-the-city through within-the-nation applications.
Strategy: All timeouts are based on self-calibrating ring-circulation timers, so the protocols readily adapt to changed in ring diameters. Synchronous traffic preempts class-BC traffic, so the length of class-BC frames doesn't effect class-A frame latencies.

F.2.5 Resilient

Objective: The link should recover from topology changes within 50 ms.
Requirement: The link shall recover from topology changes within several ringlet-circulation latencies.
Strategy: Link changes involve a topology rediscovery, to handle all join/sever combinations.
Merging and dividing of two segments are optimized special cases.
Miss-addressed packets are stripped within several ringlet-circulation times.

Objective: The link should operate over arbitrary cable topologies.
Requirement: The link shall operate over daisy-chain or loop cable topologies.
Strategy: A cable topology invokes topology exploration.
That exploration attempts to form a spanning tree by selectively disabling paths.
The spanning-tree protocols detect and enables (rather than disables) a loop topology.

Objective: The location of failed or marginal links should be easily identifiable.
Requirement: The location of failed links should be easily identifiable.
Strategy: Data CRC checks are performed on a hop-by-hop basis.
Bad-CRC packets are distinctively marked, to avoid incrementing spurious error counts.
Miss-addressed frames are aged and quickly discarded on their second ringlet circulation.

F.2.6 Efficient

Objective: The unicast destination or final multicast destination strips the returning packet.
Requirement: The packet is stripped when its destination is reached.
Strategy: The unicast packets are stripped based when they reach their destination.
The multicast packets are stripped when they return to their source.

Objective: Packets should be transmitted in the most-optimal direction.
Requirement: Packets shall be allowed to be sent on the more optimal counter-rotating ringlet.
Strategy: Allow a time-stamped acknowledge indication to be returned.
The transmitter has the option of changing ringlets, based on acknowledge information.

Annex G: C code illustrations

```
//                                                                                                                   1         1         1         1
//       1         2         3         4         5         6         7         8         9         0         1         2         3
//23456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012
//
//  Basic frame format:
//  +-------------------------------------------------------------+
//  |   targetID  |t|   sourceID  |s|cls|afl|bfl|w|m|   timeToLive  |
//  +-'-'-'-'-'-'-+-'-'-'-'-'-'-'-+-'-'-'-'-'-'-'-+-'-'-'-'-'-'-'-|
//  |                             crc32                           |
//  +-'-'-'-'-'-'-+-'-'-'-'-'-'-'-+-'-'-'-'-'-'-'-+-'-'-'-'-'-'-'-|
//  |             type            |                               |
//  +-'-'-'-'-'-'-+-'-'-'-'-'-'-'-+                               +
//  |                                                             |
//  |                         typeDependent                       |
//  |                                                             |
//  +-'-'-'-'-'-'-+-'-'-'-'-'-'-'-+-'-'-'-'-'-'-'-+-'-'-'-'-'-'-'-+
//  |                             crc32                           |
//  +-'-'-'-'-'-'-+-'-'-'-'-'-'-'-+-'-'-'-'-'-'-'-+-'-'-'-'-'-'-'-+

//  Extended frame format:
//  +-------------------------------------------------------------+
//  |   targetID  |t|   sourceID  |s|cls|afl|bfl|w|m|   timeToLive  |
//  +-'-'-'-'-'-'-+-'-'-'-'-'-'-'-+-'-'-'-'-'-'-'-+-'-'-'-'-'-'-'-|
//  |                   destinationMacAddressHi                   |
//  +-'-'-'-'-'-'-+-'-'-'-'-'-'-'-+-'-'-'-'-'-'-'-+-'-'-'-'-'-'-'-|
//  |    destinationMacAddressLo    |       sourceMacAddressLo      |
//  +-'-'-'-'-'-'-+-'-'-'-'-'-'-'-+-'-'-'-'-'-'-'-+-'-'-'-'-'-'-'-|
//  |                     sourceMacAddressLo                      |
//  +-'-'-'-'-'-'-+-'-'-'-'-'-'-'-+-'-'-'-'-'-'-'-+-'-'-'-'-'-'-'-|
//  |                             crc32                           |
//  +-'-'-'-'-'-'-+-'-'-'-'-'-'-'-+-'-'-'-'-'-'-'-+-'-'-'-'-'-'-'-|
//  |             type            |                               |
//  +-'-'-'-'-'-'-+-'-'-'-'-'-'-'-+                               +
//  |                                                             |
//  |                         typeDependent                       |
//  |                                                             |
//  +-'-'-'-'-'-'-+-'-'-'-'-'-'-'-+-'-'-'-'-'-'-'-+-'-'-'-'-'-'-'-+
//  |                             crc32                           |
//  +-'-'-'-'-'-'-+-'-'-'-'-'-'-'-+-'-'-'-'-'-'-'-+-'-'-'-'-'-'-'-+
```

```
typedef unsigned long Quadlet;          /* assuming 'long' is a 32-bit value */

void PrintTable(int, int);
Quadlet GenerateCrc(int, Quadlet *, int);
int ValidateCrc(int, Quadlet *, int);
Quadlet CalculateCrc(int, Quadlet *, int);
Quadlet CrcBits(Quadlet, Quadlet, int);
Quadlet BitReverse(Quadlet);
void Error(char *);
void assert(int);

#define MSB32  ((unsigned)1<<31)
#define ONES32 0XFFFFFFFF
#define CRC_COMPUTE ((Quadlet)0X04C11DB7)
#define CRC_RESULTS ((Quadlet)0XC704DD7B)

int
main(int argc, char **argv) {
    int i;
    int size=32, reverse=1, table=1;
    int setRev= 0, setHow=0;
    char *argPtr;

    // Command line specifies number of bits computed in parallel
    for (i= 1; i<argc; i+= 1) {
        argPtr= argv[i];
        if (*argPtr!='-')
            Error("Illegal argument, use: -n -r -tdd -c");
        argPtr+= 1;

        switch (*argPtr) {
        case 'n':
        case 'r':
            if (setRev)
                Error("Mutually exclusive options: -n -r");
            reverse= (*argPtr=='r');
            setRev= 1;
            break;

        case 't':
            if (setHow)
                Error("Mutually exclusive options: -tdd -c");
            size = atoi(argPtr+1);
            if (size != 1 && size != 2 && size != 4 &&
             size != 8 && size != 16 && size != 32)
                Error("Incorrect width; -t1 -t2 -t4 -t8 -t16 or -t32\n");
            table= 1;
            break;
```

```
        case 'c':
            if (setHow)
                Error("Mutually exclusive options: -wdd -c");
            table= 0;
            setHow= 1;
            break;

        default:
            Error("Arguments: -n -r -t[1,2,4,8,16,32] -c\n");
            break;
        }
    }
    assert(table);
    PrintTable(reverse, size);
    return(0);
}

void
Error(char *string)
{
    printf(string);
    exit(1);
}

void
assert(int test)
{
    if (test==0)
        exit(1);
}
```

```
char keys[]= {'a','b','c','d','e','f','g','h','j','k','m','n','p','r','s','t'};
void
PrintTable(int reverse, int size)
{
    Quadlet last, next, select, mask, sum;
    int i, j, numb;
    for (i=0; i<32; i+= 1) {
        /* Calculate contributing-input values */
        select= 1 << (31-i);
        mask= reverse ? BitReverse(select) : select;
        printf("c%02d= ", i);
        for (j= sum= 0; j < 32; j+= 1) {
            last= 1<<(31-j);
            next = CrcBits(last, (Quadlet)0, size);
            if (next&mask)
                sum|= last;
        }
        for (j= 0; j<32; j+= 1) {
            select= 1<<(31-j);
            mask= reverse ? BitReverse(select) : select;
            numb= j<16 ? keys[j] : keys[j-16]+'A'-'a';
            if ((sum & mask) != 0) {
                sum&= ~mask;
                printf("%c", numb);
                if (j != 31)
                printf(sum!=0 ? "^" : " ");
            } else {
                printf(j!= 31 ? "  " : " ");
            }
        }
      printf(";\n");
    }
}

// Generate the CRC in packet containing "sizeInQuads" quadlet data values
Quadlet
GenerateCrc(int reverse, Quadlet *inputs, int sizeInQuads)
{
    Quadlet crcSum;

    assert(sizeInQuads >= 1);                   // Packet size including CRC

    crcSum = CalculateCrc(reverse, inputs, sizeInQuads - 1);
    // compute CRC on just the data
    return (~crcSum);
}
```

```
// Validate the CRC for a packet containing "size" quadlet data values
int
ValidateCrc(int reverse, Quadlet *inputs, int sizeInQuads)
{
    Quadlet crcSum, check;

    assert(sizeInQuads >= 1);                  // Packet size including CRC

    crcSum = CalculateCrc(reverse, inputs, sizeInQuads);
    /* compute CRC on the data * and * the received CRC */
    check = reverse ? BitReverse(crcSum) : crcSum;
    return (check != CRC_RESULTS);
}

// The GenerateCrc() function points to protected values,
// it checks these values and return a final 32 - bit result
Quadlet
CalculateCrc(int reverse, Quadlet *inputs, int sizeInQuads)
{
    Quadlet inQuad, crcSum, sum;
    int i;

    // The crcSum value is initialized to all ones
    crcSum= (Quadlet) 0XFFFFFFFF;

    // Process each of the quadlets covered by the CRC value
    for (i = 0; i < sizeInQuads; i += 1) {
        inQuad= reverse ? inputs[i] : BitReverse(inputs[i]);
        crcSum= CrcBits(crcSum, inQuad, 32);
    }
    sum= reverse ? BitReverse(crcSum) : crcSum;
    return (sum);
}

Quadlet
CrcBits(Quadlet last, Quadlet input, int size)
{
    Quadlet crcSum, newMask;
    int i, oldBit, newBit, sumBit;

    // Process each of the bits within the input quadlet value
    crcSum= last;
    for (newMask = MSB32, i= 0; i < size; newMask >>= 1, i+= 1) {
        newBit = ((input & newMask) != 0);      // The next input bit
        oldBit = ((crcSum & MSB32) != 0);       // and MSB of crcSum
        sumBit = oldBit ^ newBit;               // are EXOR'd together

        // Shift the old crcSum left and exclusive - OR the new newBit values
        crcSum = ((crcSum << 1) & ONES32) ^ (sumBit ? CRC_COMPUTE : 0);
    }
    return(crcSum);
}
```

```
// Reverse the order of bits within bytes
Quadlet
BitReverse(Quadlet old)
{
    Quadlet new, oldMask, newMask;
    int i, j;

    for (i= new= 0; i < 4; i+= 1) {
        for (j= 0; j < 8; j+= 1) {
          oldMask= 1 << (8*i + 7 - j);
          newMask= 1 << (8*i + j);
          new|= (old & oldMask) ? newMask : 0;
        }
    }
    return(new);
}
```