Residential Ethernet (RE)
(a working paper)

The following paper represents an initial attempt to codify the content of multiple IEEE 802.3 Residential Ethernet (RE) Study Group slide presentations. The author has also taken the liberty to expand on various slide-based proposals, with the goal of triggering/facilitating future discussions.

For the convenience of the author, this paper has been drafted using the style of IEEE standards. The quality of the figures and the consistency of the notation should not be confused with completeness of technical content.

Rather, the formality of this paper represents an attempt by the author to facilitate review by interested parties. Major changes and entire clause rewrites are expected before consensus-approved text becomes available.

JggDvj2005Apr16
2007-07-20
(July 20, 2007)

Residential Ethernet (RE) (a working paper)

Draft 0.137

Contributors:
See page 4.

Abstract: This working paper provides background and introduces possible higher level concepts for the development of Residential Ethernet (RE).
Keywords: residential, Ethernet, isochronous, real time

Version history

Version	Date	Author	Comments
0.082	2005-04-28	DVJ	Updates based on 2005Apr27 meeting discussions – Restructure document presentation order …
0.085	2005-05-11	DVJ	– Updated front-page list of contributors…
0.121	2005-06-24	DVJ	– Extensive cleanup of clock-synchronization protocols, …
0.134	2005-07-17	DVJ	– Pacing disclaimers, based on preceding meeting …
0.140	2005-07-30	DVJ	– Alternative rate-based pacing protocol provided…
—	TBD	—	—

Background

This working paper is highly preliminary and subject to changed. Comments should be sent to its editor:

David V. James
3180 South Ct
Palo Alto, CA 94306
Home: +1-650-494-0926
Cell: +1-650-954-6906
Fax: +1-360-242-5508
Email: dvj@alum.mit.edu

Formats

In many cases, readers may elect to provide contributions in the form of exact text replacements and/or additions. To simplify document maintenance, contributors are requested to use the standard formats and provide checklist reviews before submission. Relevant URLs are listed below:

General: http://grouper.ieee.org/groups/msc/WordProcessors.html
Templates: http://grouper.ieee.org/groups/msc/TemplateTools/FrameMaker/
Checklist: http://grouper.ieee.org/groups/msc/TemplateTools/Checks2004Oct18.pdf

Topics for discussion

Readers are encouraged to provide feedback in all areas, although only the following areas have been identified as specific areas of concern.

a) Terminology. Is classA an OK way to describe the traffic within an RE stream?
Alternatives:
synchronous traffic? isochronous traffic? RE traffic? quasi-synchronous traffic?

TBDs

Further definitions are needed in the following areas:

a) ClassA addressing models: review, select, and revise.
b) Pacing models: review, select, and revise.

Contents

List of figures

List of tables

Residential Ethernet (RE) (a working paper)

This document and has no official status within IEEE or alternative SDOs.
Feedback to: dvj@alum.mit.edu

1. Overview

1.1 Interoperability

RE interoperates with existing Ethernet, but the scope of RE services is limited to the RE cloud, as illustrated in Figure 1.1; normal best-effort services are available everywhere else. The scope of the RE cloud is limited by a non-RE capable bridge or a half-duplex link, neither of which can support RE services.

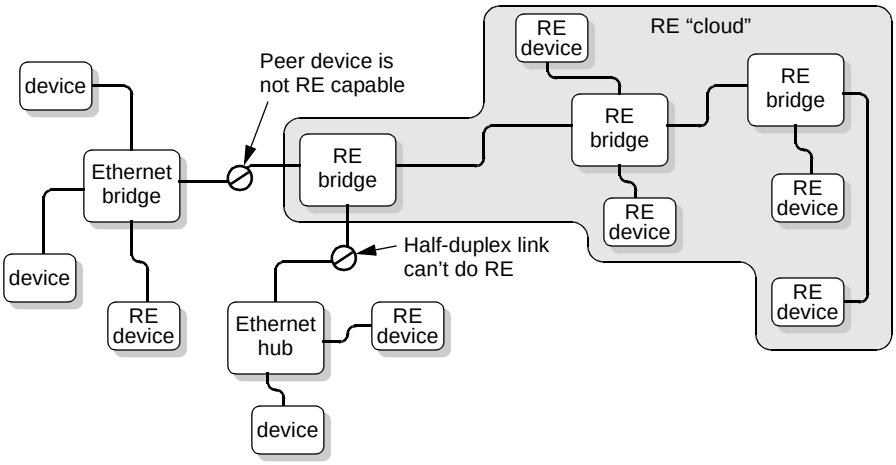

Figure 1.1—Topology and connectivity

Separation of RE devices is driven by the requirements of RE bridges to support subscription (bandwidth allocation), time-of-day clock-synchronization, and (preferably) of pacing of time-sensitive transmissions.

2. References

NOTE—This clause should be skipped on the first reading (continue with Clause 5).
This references list is highly preliminary, references will be added as this working paper evolves.

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

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

IEEE Std 802.1D-2004, IEEE Standard for Local and Metropolitan Area Networks: Media Access Control (MAC) Bridges.

[1]Replaces ANSI X3.159-1989

[2]ISO documents are available from ISO Central Secretariat, 1 Rue de Varembe, Case Postale 56, CH-1211, Geneve 20, Switzerland/Suisse; and from the Sales Department, American National Standards Institute, 11 West 42 Street, 13th Floor, New York, NY 10036-8002, USA

3. Terms, definitions, and notation

NOTE—This clause should be skipped on the first reading (continue with Clause 5).
This text has been lifted from the P802.17 draft standard, which has a relative comprehensive list.
Terms and definitions are expected to be added, revised, and/or deleted as this working paper evolves.

3.1 Conformance levels

Several key words are used to differentiate between different levels of requirements and options, as described in this subclause.

3.1.1 may: Indicates a course of action permissible within the limits of the standard with no implied preference ("may" means "is permitted to").

3.1.2 shall: Indicates mandatory requirements to be strictly followed in order to conform to the standard and from which no deviation is permitted ("shall" means "is required to").

3.1.3 should: An indication that among several possibilities, one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain course of action is deprecated but not prohibited ("should" means "is recommended to").

3.2 Terms and definitions

For the purposes of this working paper, the following terms and definitions apply. The Authoritative Dictionary of IEEE Standards Terms xx should be referenced for terms not defined in the clause.

3.2.1 audience: The set of listeners associated with a common streamID.

3.2.2 best-effort: Not associated with an explicit service guarantee.

3.2.3 bridge: A functional unit interconnecting two or more networks at the data link layer of the OSI reference model.

3.2.4 clock master: A bridge or end station that provides the link clock reference.

3.2.5 clock slave: A bridge or end station that tracks the link clock reference provided by the clock master.

3.2.6 cyclic redundancy check (CRC): A specific type of frame check sequence computed using a generator polynomial.

3.2.7 destination station: A station to which a frame is addressed.

3.2.8 frame: The MAC sublayer protocol data unit (PDU).

3.2.9 grand clock master: The clock master selected to provide the network time reference.

3.2.10 jitter: The variation in delay associated with the transfer of frames between two points.

3.2.11 latency: The time required to transfer information from one point to another.[3]

3.2.12 link: A unidirectional channel connecting adjacent stations (half of a span).

3.2.13 listener: A sink of a stream, such as a television or acoustic speaker.

3.2.14 local area network (LAN): A communications network designed for a small geographic area, typically not exceeding a few kilometers in extent, and characterized by moderate to high data transmission rates, low delay, and low bit error rates.

3.2.15 MAC client: The layer entity that invokes the MAC service interface.

3.2.16 management information base (MIB): A repository of information to describe the operation of a specific network device.

3.2.17 maximum transfer unit (MTU): The largest frame (comprising payload and all header and trailer information) that can be transferred across the network.

3.2.18 medium (plural: **media):** The material on which information signals are carried; e.g., optical fiber, coaxial cable, and twisted-wire pairs.

3.2.19 medium access control (MAC) sublayer: The portion of the data link layer that controls and mediates the access to the network medium. In this working paper, the MAC sublayer comprises the MAC datapath sublayer and the MAC control sublayer.

3.2.20 multicast: Transmission of a frame to stations specified by a group address.

3.2.21 multicast address: A group address that is not a broadcast address, i.e., is not all-ones, and identifies some subset of stations on the network.

3.2.22 network: A set of communicating stations and the media and equipment providing connectivity among the stations.

3.2.23 pacer: A credit-based entity that partions residual bandwidths between two classes of frames.

3.2.24 packet: A generic term for a PDU associated with a layer-entity above the MAC sublayer.

3.2.25 path: A logical concatenation of links and bridges over which streams flow from the talker to the listener.

3.2.26 plug-and-play: The requirement that a station perform classA transfers without operator intervention (except for any intervention needed for connection to the cable).

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

3.2.28 service discovery: The process used by listeners or controlling stations to identify, control, and configure talkers.

3.2.29 shaper: A credit-based entity that limits short-term transmission bandwidths to a specified rate.

3.2.30 simple reservation protocol (SRP): The subscription protocol used to allocate and sustain paths for streaming classA traffic.

[3]Delay and latency are synonyms for the purpose of this working paper. Delay is the preferred term.

3.2.31 span: A bidirectional channel connecting adjacent stations (two links).

3.2.32 source station: The station that originates a frame.

3.2.33 station: A device attached to a network for the purpose of transmitting and receiving information on that network.

3.2.34 stream: A sequence of frames passed from the talker to listener(s), which have the same streamID.

3.2.35 subscription: The process of establishing committed paths between the talker and one or more listeners.

3.2.36 talker: The source of a stream, such as a cable box or microphone.

3.2.37 topology: The arrangement of links and stations forming a network, together with information on station attributes.

3.2.38 transmit (transmission): The action of a station placing a frame on the medium.

3.2.39 transparent bridging: A bridging mechanism that is transparent to the end stations.

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

3.3 Service definition method and notation

The service of a layer or sublayer is the set of capabilities that it offers to a user in the next higher (sub)layer. Abstract services are specified in this working paper by describing the service primitives and parameters that characterize each service. This definition of service is independent of any particular implementation (see Figure 3.1).

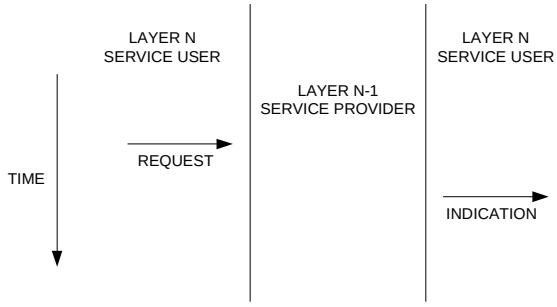

Figure 3.1—Service definitions

Specific implementations can also include provisions for interface interactions that have no direct end-to-end effects. Examples of such local interactions include interface flow control, status requests and indications, error notifications, and layer management. Specific implementation details are omitted from this service specification, because they differ from implementation to implementation and also because they do not impact the peer-to-peer protocols.

3.3.1 Classification of service primitives

Primitives are of two generic types.

a) REQUEST. The request primitive is passed from layer N to layer N-1 to request that a service be initiated.
b) INDICATION. The indication primitive is passed from layer N-1 to layer N to indicate an internal layer N-1 event that is significant to layer N. This event can be logically related to a remote service request, or can be caused by an event internal to layer N-1.

The service primitives are an abstraction of the functional specification and the user-layer interaction. The abstract definition does not contain local detail of the user/provider interaction. For instance, it does not indicate the local mechanism that allows a user to indicate that it is awaiting an incoming call. Each primitive has a set of zero or more parameters, representing data elements that are passed to qualify the functions invoked by the primitive. Parameters indicate information available in a user/provider interaction. In any particular interface, some parameters can be explicitly stated (even though not explicitly defined in the primitive) or implicitly associated with the service access point. Similarly, in any particular protocol specification, functions corresponding to a service primitive can be explicitly defined or implicitly available.

3.4 State machines

3.4.1 State machine behavior

The operation of a protocol can be described by subdividing the protocol into a number of interrelated functions. The operation of the functions can be described by state machines. Each state machine represents the domain of a function and consists of a group of connected, mutually exclusive states. Only one state of a function is active at any given time. A transition from one state to another is assumed to take place in zero time (i.e., no time period is associated with the execution of a state), based on some condition of the inputs to the state machine.

The state machines contain the authoritative statement of the functions they depict. When apparent conflicts between descriptive text and state machines arise, the order of precedence shall be formal state tables first, followed by the descriptive text, over any explanatory figures. This does not override, however, any explicit description in the text that has no parallel in the state tables.

The models presented by state machines are intended as the primary specifications of the functions to be provided. It is important to distinguish, however, between a model and a real implementation. The models are optimized for simplicity and clarity of presentation, while any realistic implementation might place heavier emphasis on efficiency and suitability to a particular implementation technology. It is the functional behavior of any unit that has to match the standard, not its internal structure. The internal details of the model are useful only to the extent that they specify the external behavior clearly and precisely.

3.4.2 State table notation

NOTE—The following state machine notation was used within 802.17, due to the exactness of C-code conditions and the simplicity of updating table entries (as opposed to 2-dimensional graphics).
Early state table descriptions can be converted (if necessary) into other formats before publication.

Each row of the table is preferably provided with a brief description of the condition and/or action for that row. The descriptions are placed after the table itself, and linked back to the rows of the table using numeric tags.

3.4.2.1 Parallel-execution state tables

State machines may be represented in tabular form. The table is organized into two columns: a left hand side representing all of the possible states of the state machine and all of the possible conditions that cause transitions out of each state, and the right hand side giving all of the permissible next states of the state machine as well as all of the actions to be performed in the various states, as illustrated in Table 3.1. The syntax of the expressions follows standard C notation (see 3.13). No time period is associated with the transition from one state to the next.

Table 3.1—State table notation example

Current		Row	Next	
state	**condition**		**action**	**state**
START	sizeOfMacControl > spaceInQueue	1	—	START
	passM == 0	2		
	—	3	TransmitFromControlQueue();	FINAL
FINAL	STATE_ENTERED	4	done = FALSE;	—
	done == TRUE	5	—	START

Row 3.1-1: Do nothing if the size of the queued MAC control frame is larger than the PTQ space.
Row 3.1-2: Do nothing in the absence of MAC control transmission credits.
Row 3.1-3: Otherwise, transmit a MAC control frame.

Row 3.1-4: When the state is entered, set *done* to FALSE.
Row 3.1-5: Wait for the *done* state to be set to TRUE (see state machine xx).

Each combination of current state, next state, and transition condition linking the two is assigned to a different row of the table. Each row of the table, read left to right, provides: the name of the current state; a condition causing a transition out of the current state; an action to perform (if the condition is satisfied); and, finally, the next state to which the state machine transitions, but only if the condition is satisfied.

The symbol "—" signifies the default condition (i.e., operative when no other condition is active) when placed in the condition column, and signifies that no action is to be performed when placed in the action column. Conditions are evaluated in order, top to bottom, and the first condition that evaluates to a result of TRUE is used to determine the transition to the next state.

The starting or initialization state of a state machine is always labeled "START" in the table (though it need not be the first state in the table); every state table has such a labeled state. The STATE_ENTERED condition is assumed to be set to TRUE on any state transition, and set to FALSE by the action of testing the condition. By convention, the STATE_ENTERED condition is only allowed in the first of (possibly multiple) rows for each specified current state value.

Each row of the table is preferably provided with a brief description of the condition and/or action for that row. The descriptions are placed after the table and linked back to the rows of the table using numeric tags.

3.4.2.2 Called state tables

A RETURN state is the terminal state of a state machine that is intended to be invoked by another state machine, as illustrated in Table 3.2. Once the RETURN state is reached, the state machine terminates execution, effectively ceasing to exist until the next invocation by the caller, at which point it begins execution again from the START state. State machines that contain a RETURN state are considered to be only instantiated when they are invoked. They do not have any persistent (static) variables.

Table 3.2—Called state table notation example

Current		Row	Next	
state	**condition**		**action**	**state**
START	sizeOfMacControl > spaceInQueue	1	—	FINAL
	passM == 0	2		
	—	3	TransmitFromControlQueue();	RETURN
FINAL	MacTransmitError();	4	errorDefect = TRUE	RETURN
	—	5	—	

Row 3.2-1: The size of the queued MAC control frame is less than the PTQ space.
Row 3.2-2: In the absence of MAC control transmission credits, no action is taken.
Row 3.2-3: MAC control transmissions have precedence over client transmissions.

Row 3.2-4: If the transmission completes with an error, set an error defect indication.
Row 3.2-5: Otherwise, no error defect is indicated.

3.5 Arithmetic and logical operators

In addition to commonly accepted notation for mathematical operators, Table 3.3 summarizes the symbols used to represent arithmetic and logical (boolean) operations. Note that the syntax of operators follows standard C notation (see 3.13).

Table 3.3—Special symbols and operators

Printed character	Meaning
&&	Boolean AND
||	Boolean OR
!	Boolean NOT (negation)
&	Bitwise AND
|	Bitwise OR
^	Bitwise XOR
<=	Less than or equal to
>=	Greater than or equal to
==	Equal to
!=	Not equal to
=	Assignment operator
//	Comment delimiter

3.6 Numerical representation

NOTE—The following notation was taken from 802.17, where it was found to have benefits:
– The subscript notation is consistent with common mathematical/logic equations.
– The subscript notation can be used consistently for all possible radix values.

Decimal, hexadecimal, and binary numbers are used within this working paper. 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”.

MAC addresses and OUI/EUI values are represented as strings of 8-bit hexadecimal numbers separated by hyphens and without a subscript, as for example “01-80-C2-00-00-15” or “AA-55-11”.

3.7 Field notations

3.7.1 Use of italics

All field names or variable names (such as *level* or *myMacAddress*), and sub-fields within variables (such as *thisState.level*) are italicized within text, figures and tables, to avoid confusion between such names and similarly spelled words without special meanings. A variable or field name that is used in a subclause heading or a figure or table caption is also italicized. Variable or field names are not italicized within C code, however, since their special meaning is implied by their context. Names used as nouns (e.g., subclassA0) are also not italicized.

3.7.2 Field conventions

This working paper describes values that are packetized or MAC-resident, such as those illustrated in Table 3.2.

Table 3.4—Names of fields and sub-fields

Name	Description
newCRC	Field within a register or frame
thisState.level	Sub-field within field *thisState*
thatState.rateC[*n*].*c*	Sub-field within array element *rateC*[*n*]

Run-together names (e.g., *thisState*) are used for fields because of their compactness when compared to equivalent underscore-separated names (e.g., *this_state*). The use of multiword names with spaces (e.g., “This State”) is avoided, to avoid confusion between commonly used capitalized key words and the capitalized word used at the start of each sentence.

A sub-field of a field is referenced by suffixing the field name with the sub-field name, separated by a period. For example, *thisState.level* refers to the sub-field *level* of the field *thisState*. This notation can be continued in order to represent sub-fields of sub-fields (e.g., *thisState.level.next* is interpreted to mean the sub-field *next* of the sub-field *level* of the field *thisState*).

Two special field names are defined for use throughout this working paper. The name *frame* is used to denote the data structure comprising the complete MAC sublayer PDU. Any valid element of the MAC sublayer PDU, can be referenced using the notation *frame.xx* (where *xx* denotes the specific element); thus, for instance, *frame.serviceDataUnit* is used to indicate the *serviceDataUnit* element of a frame.

Unless specifically specified otherwise, reserved fields are reserved for the purpose of allowing extended features to be defined in future revisions of this working paper. For devices conforming to this version of this working paper, nonzero reserved fields are not generated; values within reserved fields (whether zero or nonzero) are to be ignored.

3.7.3 Field value conventions

This working paper describes values of fields. For clarity, names can be associated with each of these defined values, as illustrated in Table 3.5. A symbolic name, consisting of upper case letters with underscore separators, allows other portions of this working paper to reference the value by its symbolic name, rather than a numerical value.

Table 3.5—*wrap* field values

Value	Name	Description
0	STANDARD	Standard processing selected
1	SPECIAL	Special processing selected
2,3	—	Reserved

Unless otherwise specified, reserved values allow extended features to be defined in future revisions of this working paper. Devices conforming to this version of this working paper do not generate nonzero reserved values, and process reserved fields as though their values were zero.

A field value of TRUE shall always be interpreted as being equivalent to a numeric value of 1 (one), unless otherwise indicated. A field value of FALSE shall always be interpreted as being equivalent to a numeric value of 0 (zero), unless otherwise indicated.

3.8 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 shown in figures using the conventions illustrated by Figure 3.2, which represents a link with four byte lanes. For multi-byte objects, the first (left-most) data byte is the most significant, and the last (right-most) data byte is the least significant.

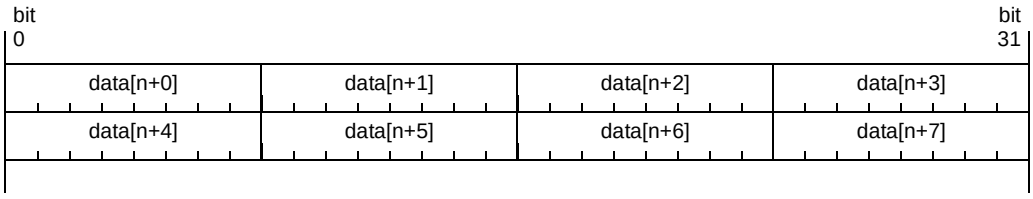

Figure 3.2—Bit numbering and ordering

Figures are drawn such that the counting order of data bytes is from left to right within each cycle, and from top to bottom between cycles. For consistency, bits and bytes are numbered in the same fashion.

NOTE—The transmission ordering of data bits and data bytes is not necessarily the same as their counting order; the translation between the counting order and the transmission order is specified by the appropriate reconciliation sublayer.

3.9 Byte sequential formats

Figure 3.3 provides an illustrative example of the conventions to be used for drawing frame formats and other byte sequential representations. These representations are drawn as fields (of arbitrary size) ordered along a vertical axis, with numbers along the left sides of the fields indicating the field sizes in bytes. Fields are drawn contiguously such that the transmission order across fields is from top to bottom. The example shows that *field1*, *field2*, and *field3* are 1-, 1- and 6-byte fields, respectively, transmitted in order starting with the *field1* field first. As illustrated on the right hand side of Figure 3.3, a multi-byte field represents a sequence of ordered bytes, where the first through last bytes correspond to the most significant through least significant portions of the multi-byte field, and the MSB of each byte is drawn to be on the left hand side.

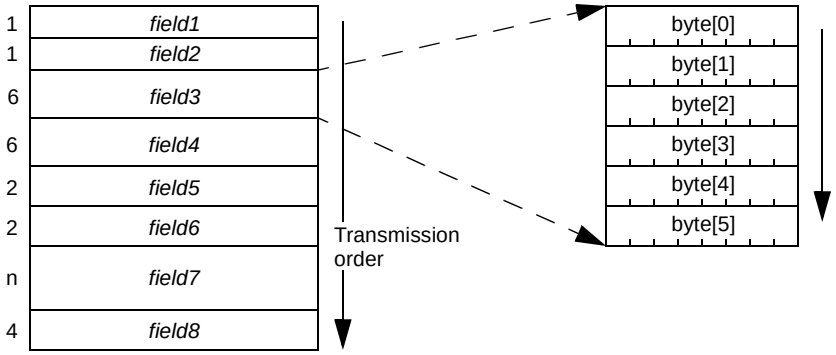

Figure 3.3—Byte sequential field format illustrations

NOTE—Only the left-hand diagram in Figure 3.3 is required for representation of byte-sequential formats. The right-hand diagram is provided in this description for explanatory purposes only, for illustrating how a multi-byte field within a byte sequential representation is expected to be ordered. The tag “Transmission order” and the associated arrows are not required to be replicated in the figures.

3.10 Ordering of multibyte fields

In many cases, bit fields within byte or multibyte objects are expanded in a horizontal fashion, as illustrated in the right side of Figure 3.4. The fields within these objects are illustrated as follows: left-to-right is the byte transmission order; the left-through-right bits are the most significant through least significant bits respectively.

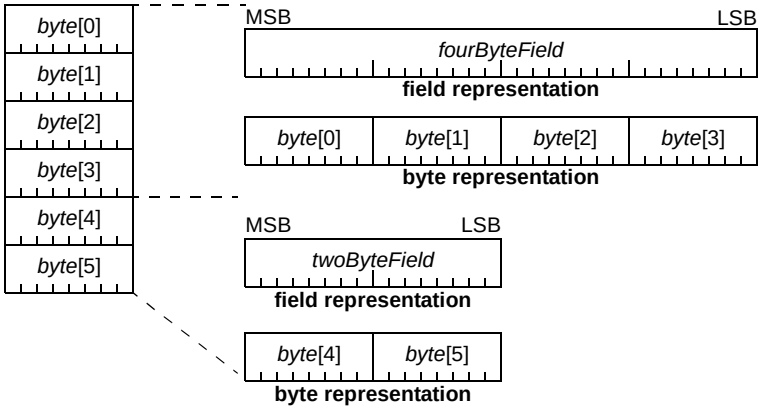

Figure 3.4—Multibyte field illustrations

The first *fourByteField* can be illustrated as a single entity or a 4-byte multibyte entity. Similarly, the second *twoByteField* can be illustrated as a single entity or a 2-byte multibyte entity.

NOTE—The following text was taken from 802.17, where it was found to have benefits:
The details should, however, be revised to illustrate fields within an RE frame header *serviceDataUnit*.

To minimize potential for confusion, four equivalent methods for illustrating frame contents are illustrated in Figure 3.5. Binary, hex, and decimal values are always shown with a left-to-right significance order, regardless of their bit-transmission order.

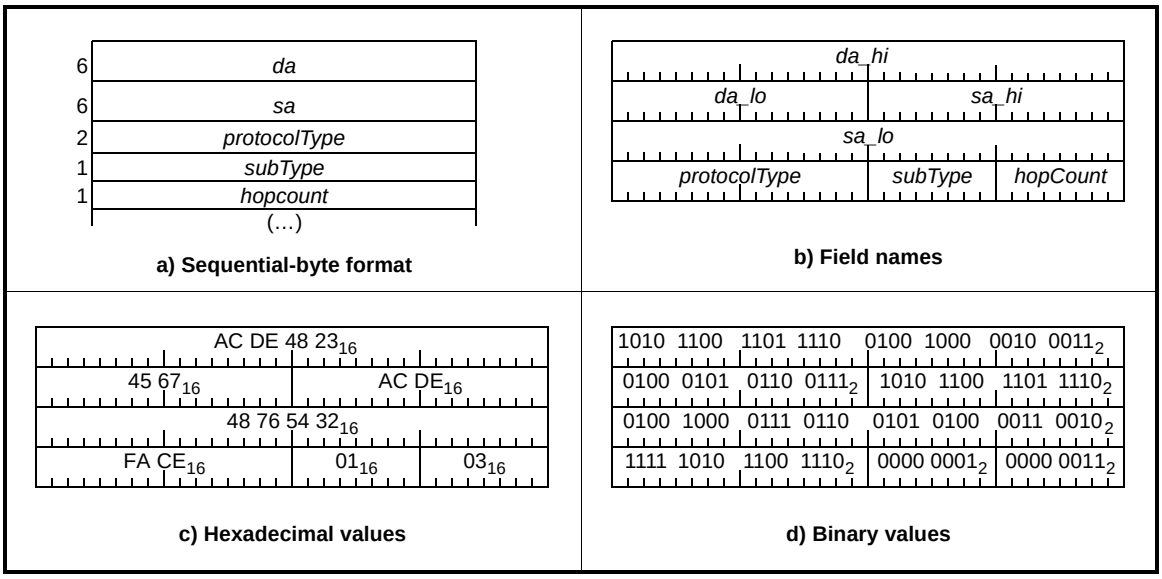

Figure 3.5—Illustration of fairness-frame structure

3.11 MAC address formats

The format of MAC address fields within frames is illustrated in Figure 3.6.

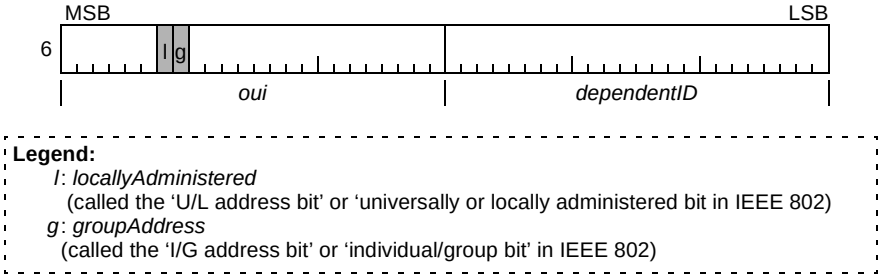

Figure 3.6—MAC address format

3.11.1 *oui*: A 24-bit organizationally unique identifier (OUI) field supplied by the IEEE/RAC for the purpose of identifying the organization supplying the (unique within the organization, for this specific context) 24-bit *dependentID*. (For clarity, the *locallyAdministered* and *groupAddress* bits are illustrated by the shaded bit locations.)

3.11.2 *dependentID*: An 24-bit field supplied by the *oui*-specified organization. The concatenation of the *oui* and *dependentID* provide a unique (within this context) identifier.

To reduce the likelihood of error, the mapping of OUI values to the *oui/dependentID* fields are illustrated in Figure 3.7. For the purposes of illustration, specific OUI and *dependentID* example values have been assumed. The two shaded bits correspond to the *locallyAdministered* and *groupAddress* bit positions illustrated in Figure 3.6.

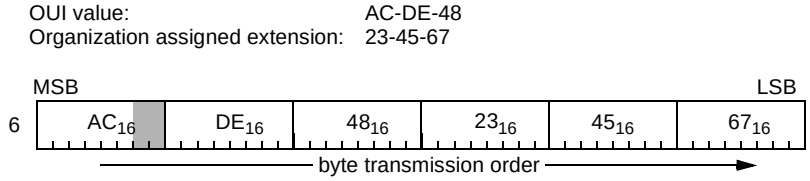

Figure 3.7—48-bit MAC address format

3.12 Informative notes

Informative notes are used in this working paper to provide guidance to implementers and also to supply useful background material. Such notes never contain normative information, and implementers are not required to adhere to any of their provisions. An example of such a note follows.

NOTE—This is an example of an informative note.

3.13 Conventions for C code used in state machines

Many of the state machines contained in this working paper utilize C code functions, operators, expressions and structures for the description of their functionality. Conventions for such C code can be found in Annex xx.

4. Transmit shapers

4.1 Credit-based shapers

4.1.1 Shaper behaviors

NOTE—These shapers are based on concepts codified during the development of 802.17.
The approved 802.17-2004 standard is available at:
http://standards.ieee.org/getieee802/802.17.html

The behavior of output-queue shapers can be characterized by a common algorithm and instance-specific parameters. The shaper's credits are adjusted down or up, based on falling-transmit and rising-idle slopes, as illustrated in Figure 4.1. The falling slope corresponds to the loss of credits-loss rate while a shaper's frame is being transmitted; the rising slope corresponds to the accumulation of credits between frame transmissions. The affiliated *transmit* and *allowed* signals are plotted in the bottom half of Figure 4.1.

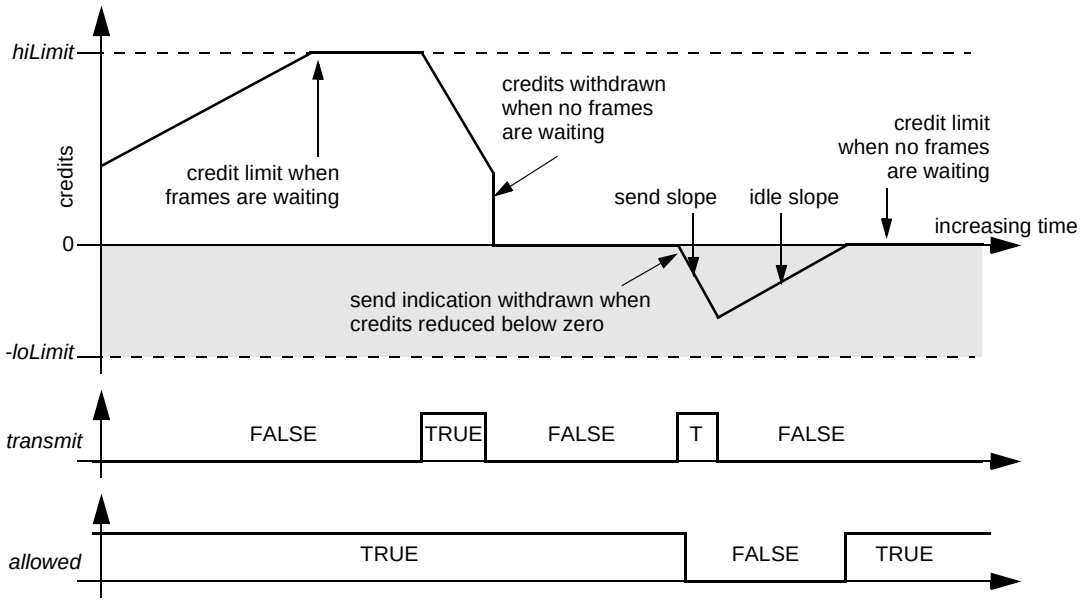

Figure 4.1—Credit-based shapers

The value of input *transmit* signal is TRUE or FALSE when the shaped transmissions are active and inactive, respectively. The value of output *allowed* signal is TRUE or FALSE when the shaped transmissions are allowed or disallowed, respectively. Because all packet transmissions are allowed to complete, the *transmit* signal can sometimes remains TRUE (for up-to a packet duration) after *allowed* has transitioned to FALSE.

In concept, the shaper consist of a token bucket. The credits in a token bucket are drained at a constant rate during each frame transmission. The credits in the token bucket are constantly replenished at the allowed transmission rate. A frame is only transmitted when the credits are positive.

Crossing below the zero threshold changes the rate-limiting *allowed* indication to FALSE, so that offered traffic can stop. By design, the credit value never goes below the *–loLimit* extreme. To bound the burst

traffic after inactive (no frames are ready for transmission) intervals, credits are reduced to zero (if currently higher than zero) and can accumulate to no more than this zero-value limit.

The *hiLimit* threshold limits the positive credits, to avoid overflow. When frames are ready for transmission (and are being blocked by conflicting traffic), credits can accumulate to no more than this *hiLimit* value.

4.1.2 Shaper parameters

Parameters for these rate shapers are specified in Table 4.1. All parameters are specified in units of bytes-per-second.

Table 4.1—ClassA and classB shaper parameters

Parameter	Value		Description
	classA	**classB**	
loLimit	classA MTU	classB MTU	Worst-case credit-loss from transmission.
hiLimit	non-classA MTU	*rateB*×*intervalB*	Worst-case credit-gain when blocked.
idleSlope	*rateA*	*rateB*	Rising slope between shaped transmissions.
sendSlope	*rateA* – *linkRate*	*rateB* – *linkRate*	Falling slope during shaped transmissions.

Notes:
rateA The negotiated rate of classA traffic, in bytes/second.
rateB The negotiated rate of classB traffic, in bytes/second.
linkRate The capacity of the link, in bytes/second.
intervalB The observation interval for classB traffic. (This value is TBD.)

Annexes

Annex A

(informative)

Bursting and bunching considerations

A.1 Topology scenarios

A.1.1 Bridge design models

The sensitivity of bridges to bursting and bunching is highly dependent on the queue management protocols within the bridge. To better understand these effects, a few bridge design models are evaluated, as illustrated in Figure A.1.

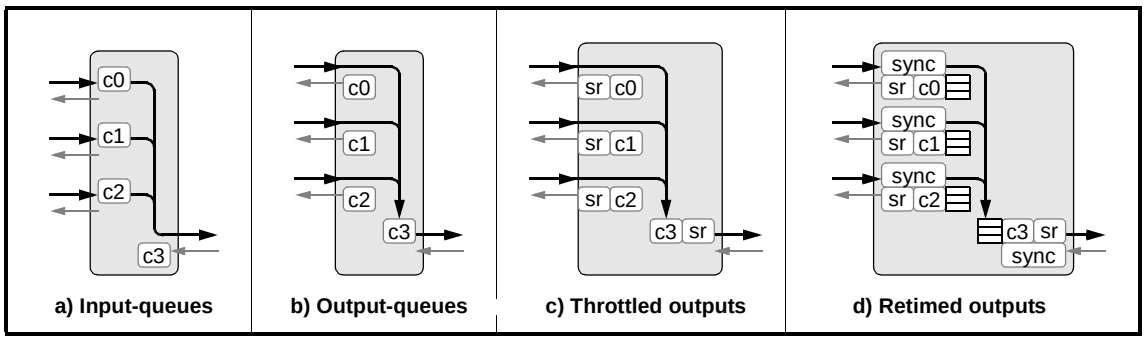

Figure A.1—Bridge design models

The input-queue design (see Figure A.1-a) assumes that frames are queued in receive buffers. The transmitter accepts frames are from the receivers, based on service-class precedence. In the case of a tie (two receivers can provide same-class frames), the lowest numbered receive port has precedence. This model best illustrates nonlinear bunching problems.

The output-queue design (see Figure A.1-b) assumes that received frames are queued in transmit buffers. Within each service class, frames are forwarded in FIFO order. This model best illustrates linear bunching problems (for steady flows), but also exhibits nonlinear bunching (for nonsteady flows).

The throttled-output design (see Figure A.1-c) is an enhanced output-queue model, with an output shaper to limit transmission rates. The purpose of the output shaper is to ensure sufficient nonreserved bandwidth for less time-sensitive control and monitoring purposes. The model illustrates how shapers can worsen the output-queue bridge's bunching behaviors.

The retimed-outputs design (see Figure A.1-d) reduces (and can eliminate) bunching problems by detecting late-arrival frames at the receivers. Several synchronous-cycle buffers are provided at the transmitters, to compensate for transmission delays in the received data.

A.1.2 Three-source hierarchical topology

A hierarchical topology best illustrate potential problems with bunching, as illustrated in Figure A.2. Traffic from talkers {a0,a1,a2} flows into bridge B. Bridge B concentrates traffic received from three talkers, with the cumulative b3 traffic sent to c3. Identical traffic flows are assumed at bridge ports {c0,c1,c3}, although only one of these sources is illustrated. Bridges {C,D,E,F,G,H} behave similarly.

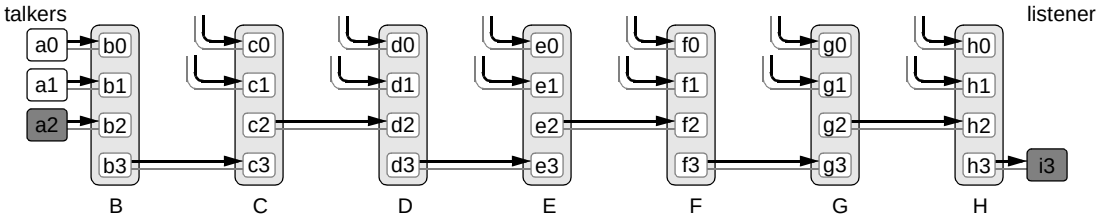

Figure A.2—Three-source topology

A.1.3 Six-source hierarchical topology

Spreading the traffic over multiple sources, as illustrated in Figure A.3, exasperates bursting and bunching problems. Traffic from talkers {a0,a1,a2,a3,a4,a5} flows into ports on bridge B. Bridge B concentrates traffic received from six talkers, with the cumulative b6 traffic sent to c6. Identical traffic flows are assumed at bridge ports {c0,c1,c3,c3,c4,c6}, although only one of these sources is illustrated. Bridges {C,D,E,F,G,H} behave similarly.

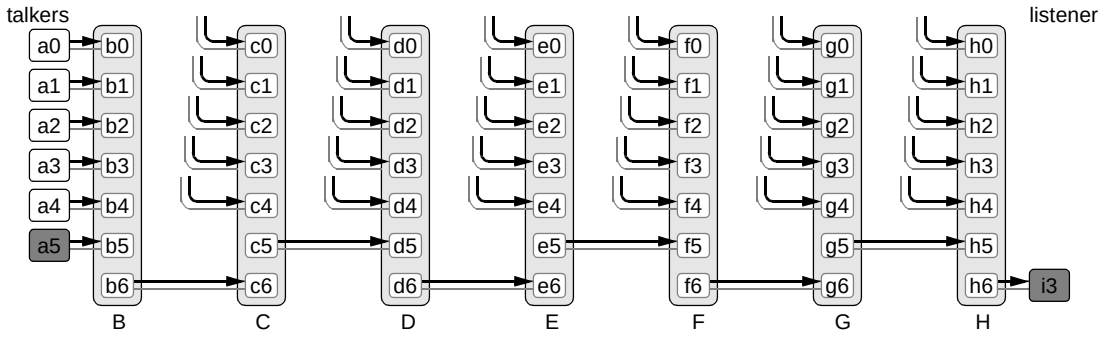

Figure A.3—Six-source topology

A.2 Bursting considerations

A.2.1 Three-source bursting scenario

A troublesome bursting scenario on a 100 Mb/s link can occur when small bandwidth streams coincidentally provide their infrequent 1500 byte frames concurrently, as illustrated in Figure A.4. Even though the cumulative bandwidths are considerably less than the capacity of the 100 Mb/s links, significant delays are incurred when passing through multiple bridges.

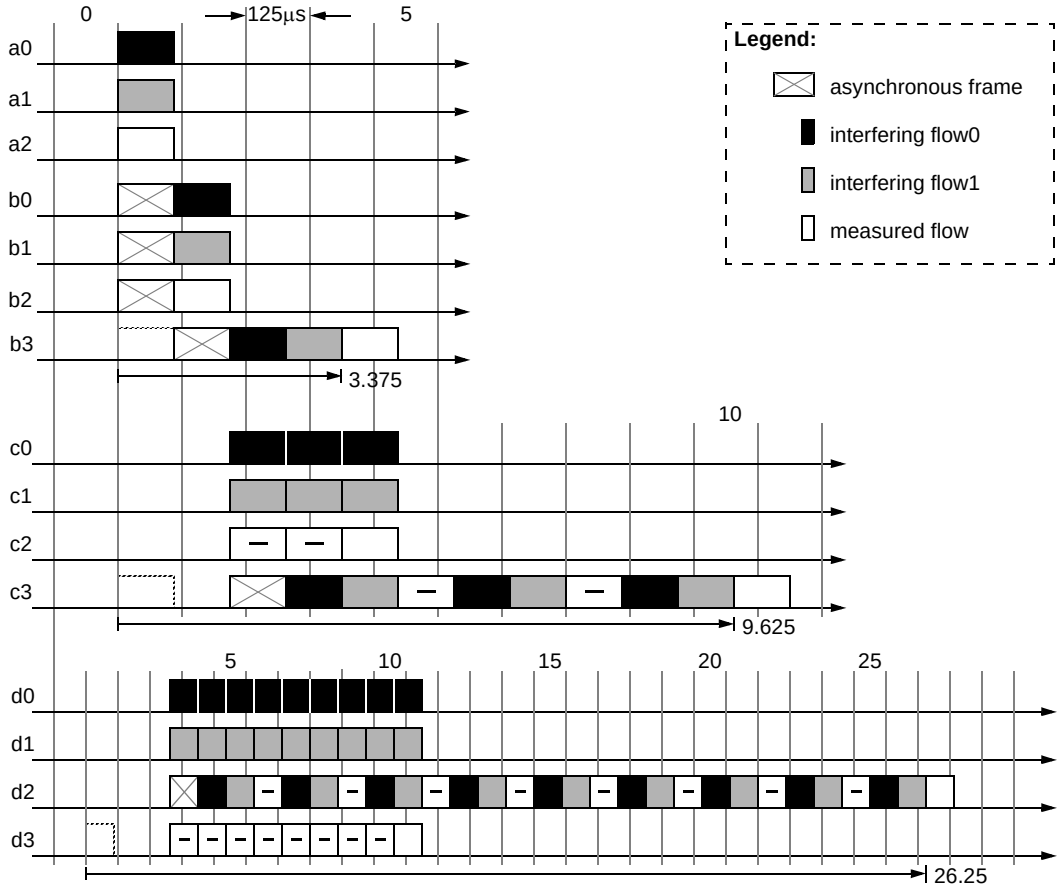

Figure A.4—Three-source bunching timing; input-queue bridges

A.2.1.1 Cumulative bunching latencies

The cumulative worst-case latencies implied by coincidental bursting are listed in Table A.1 and plotted in Figure A.5.

Table A.1—Cumulative bursting latencies

Topology	Units	Measurement point							
		A	B	C	D	E	F	G	H
3-source (see A.2.2.1)	mtu	1	4	11	30	85	248	735	2194
	ms	.120	.480	1.32	3.6	10.2	29.6	88.2	263
6-source (see A.2.2.2)	mtu	1	7	38	219	1300	7781	46662	229943
	ms	.120	.840	4.56	26.3	156	934	5600	27600

The values within this table are computed based on Equation A.1.

$$delay[n] = mtu \times (n + p^n) \quad \text{(A.1)}$$

Where:

mtu (maximum transfer unit) is the maximum frame size
n is the number of hops from the source
p is the number of receive ports in each bridge.

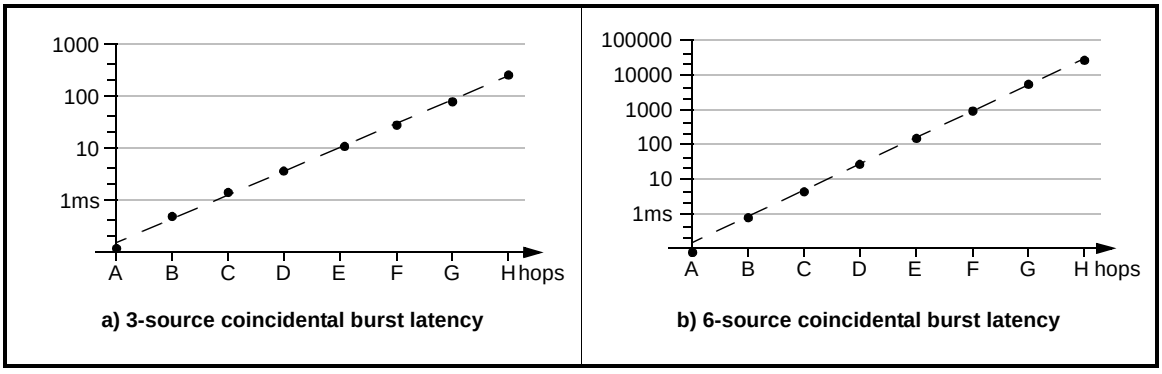

Figure A.5—Cumulative coincidental burst latencies

Conclusion: The classA traffic bandwidths should be enforced over a time interval that is on the order of an MTU size (120μs), so as to avoid excessive delays caused by coincidental back-to-back large-block transmissions.

A.2.2 Bunching scenarios; input-queue bridges

A.2.2.1 Three-source bunching; input-queue bridges

To illustrate the effects of worst case bunching on input-queue bridges, specific flows are illustrated in Figure A.6. Bridge ports {b0,b1,b2} concentrates traffic from three talkers; one third of the cumulative traffic is forwarded through b3. Each stream consumes 25% of the link bandwidth; 25% is available for asynchronous traffic.

For clarity, the traces for input traffic on ports {c0,c1,c3},…,{e0,e1,e3}, only illustrate the passing-through listener traffic; the remainder of the traffic is assumed to be routed elsewhere.

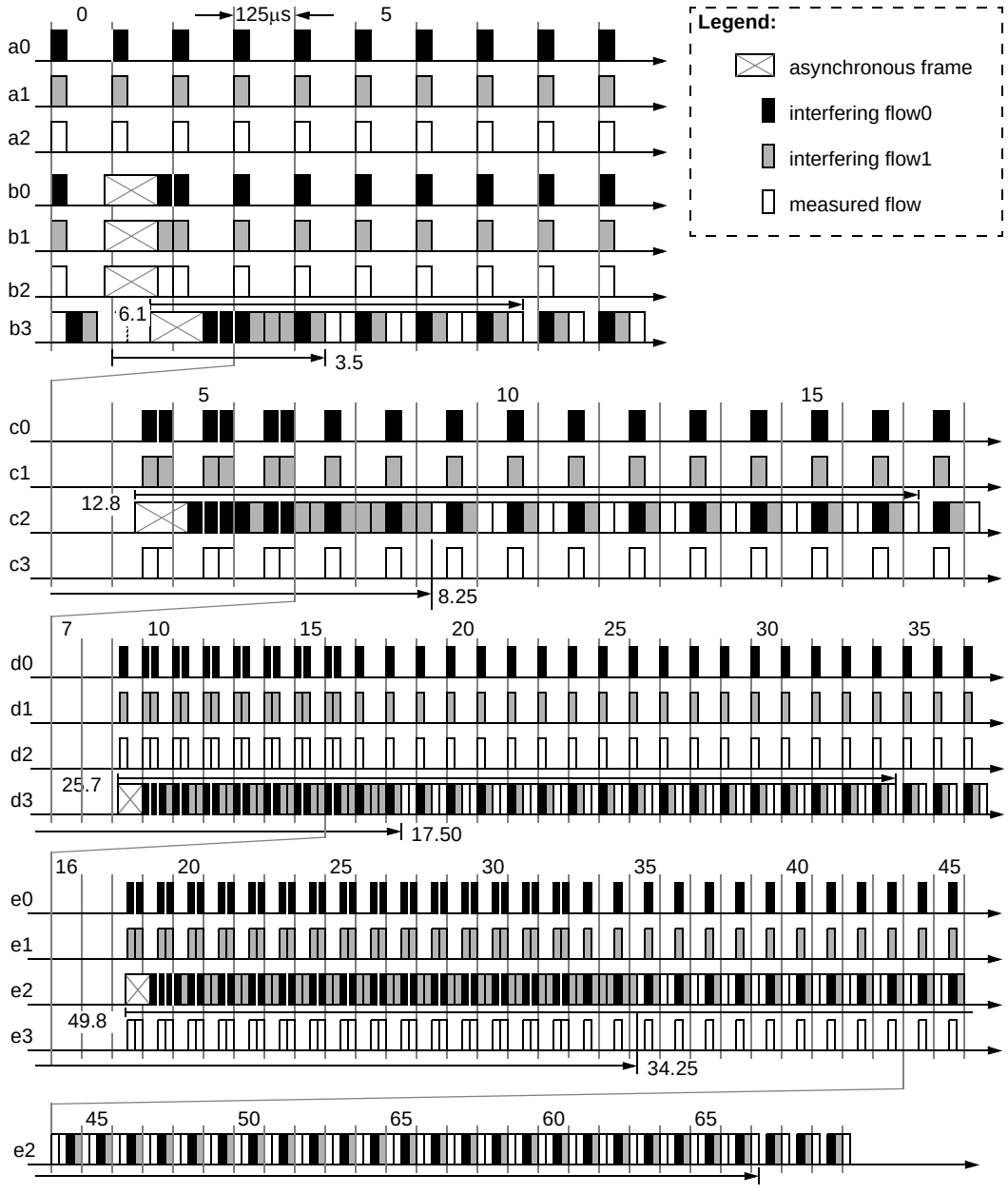

Figure A.6—Three-source bunching; input-queue bridges

A.2.2.2 Six-source bunching; input-queue bridges

To better illustrate the effects of worst case bunching on input-queue bridges, specific flows are illustrated in Figure A.7. Bridge ports {b0,b1,b2,b3,b4,b5} concentrates traffic from three talkers; one sixth of the cumulative traffic is forwarded through b6. Each of six streams consumes 12.5% of the link bandwidth, so that 25% is available for asynchronous traffic.

For clarity, the traces for input traffic on ports {c0,c1,c2,c3,c4,c6} only illustrate passing-through traffic; the remainder of the traffic is routed elsewhere.

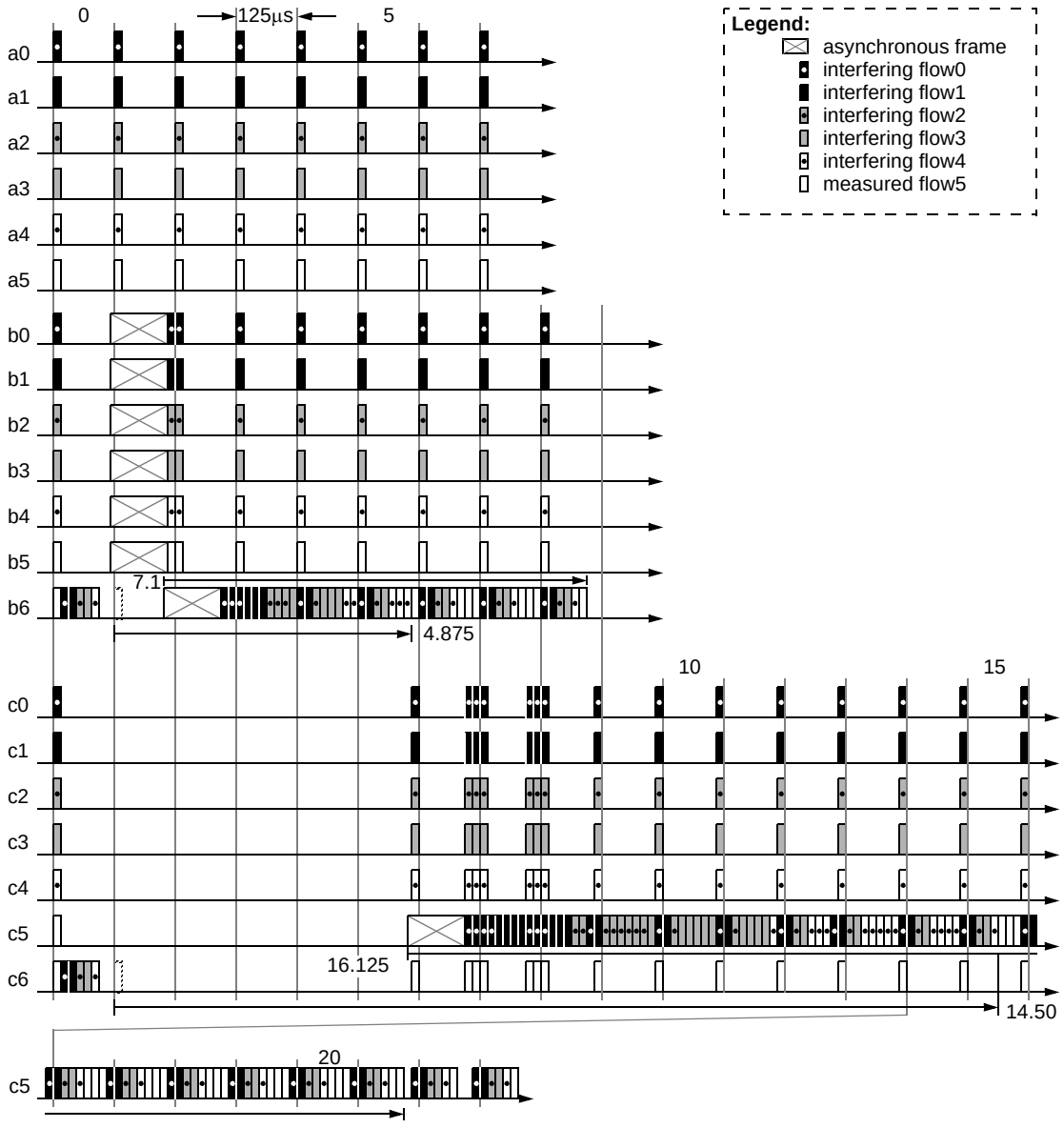

Figure A.7—Six source bunching timing; input-queue bridges

A.2.2.3 Cumulative bunching latencies, input-queue bridge

The cumulative worst-case latencies implied by coincidental bursting are listed in Table A.2 and plotted in Figure A.8.

Table A.2—Cumulative bunching latencies; input-queue bridge

Topology	Units	Measurement point							
		A	B	C	D	E	F	G	H
3-source (see A.2.2.1)	cycles	0.125	3.5	8.25	17.5	34.25	(70.75)	(143.2)	(288.2)
	ms	0.01	0.44	1.03	2.19	4.28	8.84	17.9	36.0
6-source (see A.2.2.2)	cycles	0.125	4.875	14.50	(39.33)	(107.2)	(288.2)	(771)	2058
	ms	0.01	0.61	1.81	4.92	13.4	36.0	96.4	257

The first few numbers are generated using graphical techniques, as illustrated in Figure A.2.2.2. The following numbers are estimated, based on Equation A.2.

$$delay[n+1] = (mtu + delay[n]) \times (1/(1-0.75\times(p-1)/p)) \quad \text{(A.2)}$$

Where:

mtu (maximum transfer unit) is the maximum frame size
rate is the fraction of the bandwidth reserved for class A traffic, assumed to be 0.75
n is the number of hops from the source
p is the number of receive ports in each bridge.

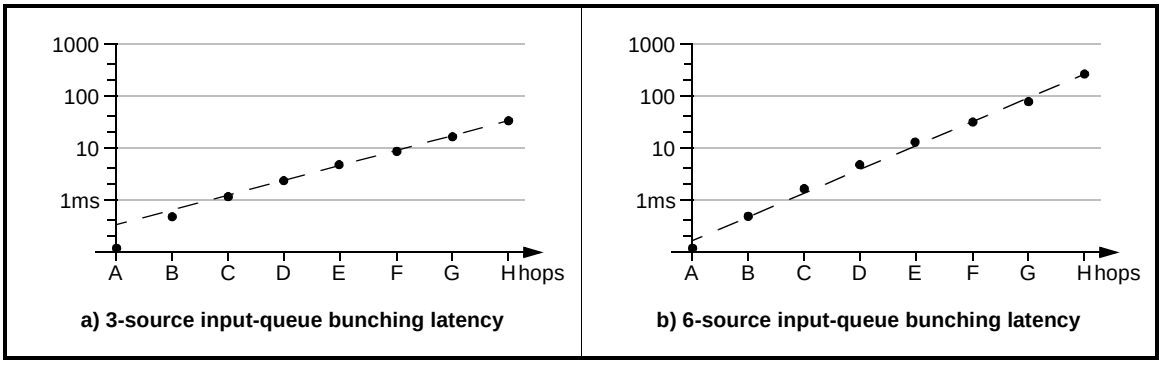

Figure A.8—Cumulative bunching latencies; input-queue bridge

Conclusion: A FIFO based output-queue bridge should be used. Alternatively (if input queuing is used), received frames should be time-stamped to ensure FIFO like forwarding.

A.2.3 Bunching topology scenarios; output-queue bridges

A.2.3.1 Three-source bunching timing; output-queue bridges

To illustrate the effects of worst case bunching, specific flows are illustrated in Figure A.9. Bridge ports {b0,b1,b2} concentrates traffic from three talkers; one third of the cumulative traffic is forwarded through b3. Each stream consumes 25% of the link bandwidth; 25% of the link bandwidth is available for asynchronous traffic.

For clarity, the traces for input traffic on ports {b0,b1,b2},…,{e0,e1,e3} only illustrate the passing-through listener traffic; the remainder of the traffic is assumed to be routed elsewhere.

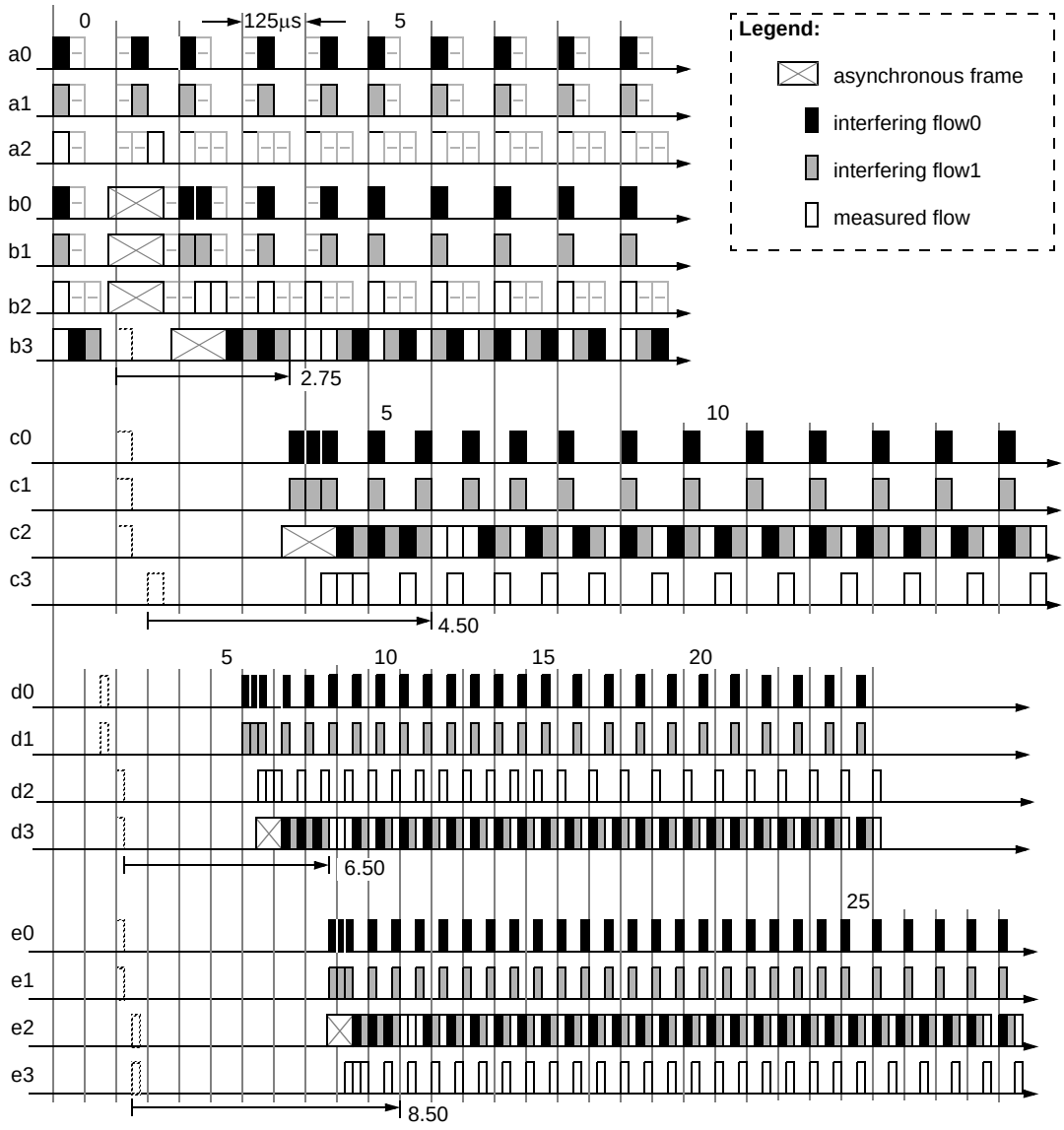

Figure A.9—Three-source bunching; output-queue bridges

A.2.3.2 Six-source bunching; output-queue bridges

To better illustrate the effects of worst case bunching, specific flows are illustrated in Figure A.10. Bridge ports {b0,b1,b2,b3,b4,b5} concentrates traffic from six talkers; one sixth of the cumulative traffic is forwarded through port b6. Each of six streams consumes 12.5% of the link bandwidth; 25% of the link bandwidth is available for asynchronous traffic.

For clarity, the traces for input traffic on ports {c0,c1,c2,c3,c4,c6} and {d0,d1,d2,d3,d4,d5} only illustrate passing-through traffic; the remainder of the traffic is routed elsewhere.

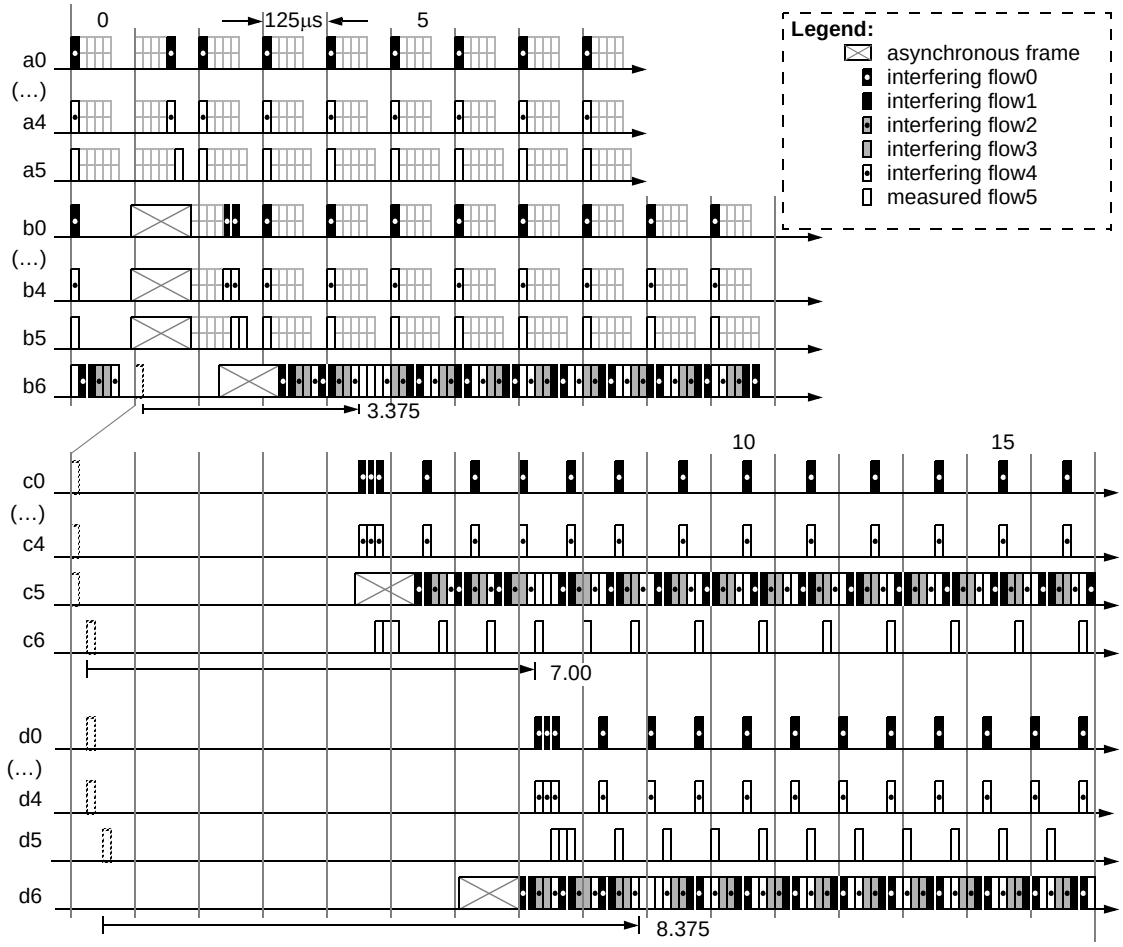

Figure A.10—Six source bunching; output-queue bridges

A.2.3.3 Cumulative bunching latencies; output-queue bridge

The cumulative worst-case latencies implied by coincidental bursting are listed in Table A.3 and plotted in Figure A.11.

Table A.3—Cumulative bunching latencies; output-queue bridge

Topology	Units	Measurement point							
		B	C	D	E	F	G	H	I
3-source (see A.2.2.1)	cycles	.875	2.75	4.5	6.5	8.5	–	–	–
	ms	0.10	0.34	0.56	0.81	1.6	–	–	–
6-source (see A.2.2.2)	cycles	.875	3.375	7.00	8.375	–	–	–	–
	ms	0.10	0.42	.875	1.05	–	–	–	–

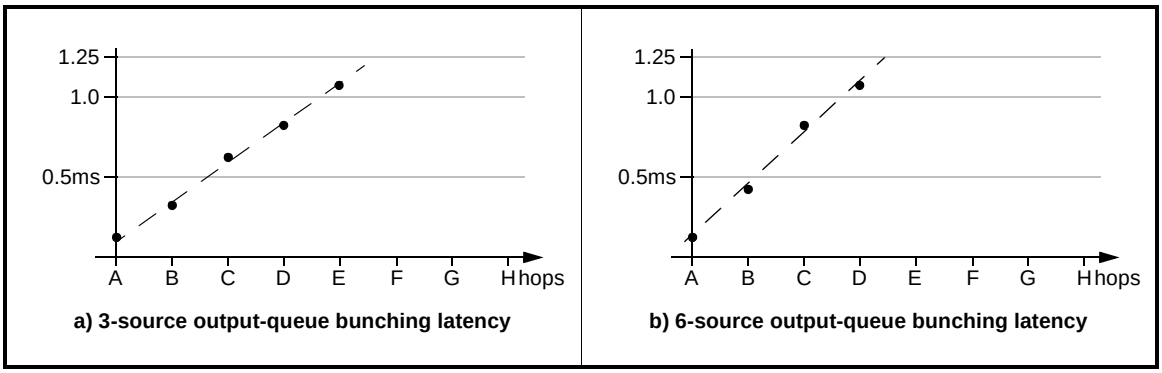

Figure A.11—Cumulative bunching latencies; output-queue bridge

Conclusion: For steady-state classA traffic, acceptably small linear latencies are introduced by output-queue bridges on 75% loaded links. Unfortunately, the nonsteady-state nature of variable-rate traffic makes this conclusion suspect (see A.2.4).

A.2.4 Bunching topology scenarios; variable-rate output-queue bridges

A.2.4.1 Three-source bunching; variable-rate output-queue bridges

To illustrate the effects of worst case bunching, specific flows are illustrated in Figure A.12. Bridge ports {b0,b1,b2} concentrates traffic from three talkers; one third of the cumulative traffic is forwarded through port b3. Each stream consumes 25% of the link bandwidth; 25% of the link bandwidth is available for asynchronous traffic.

For clarity, the traces for input traffic on ports {c0,c1,c3},…,{e0,e1,e3} only illustrate the passing-through listener traffic; the remainder of the traffic is assumed to be routed elsewhere.

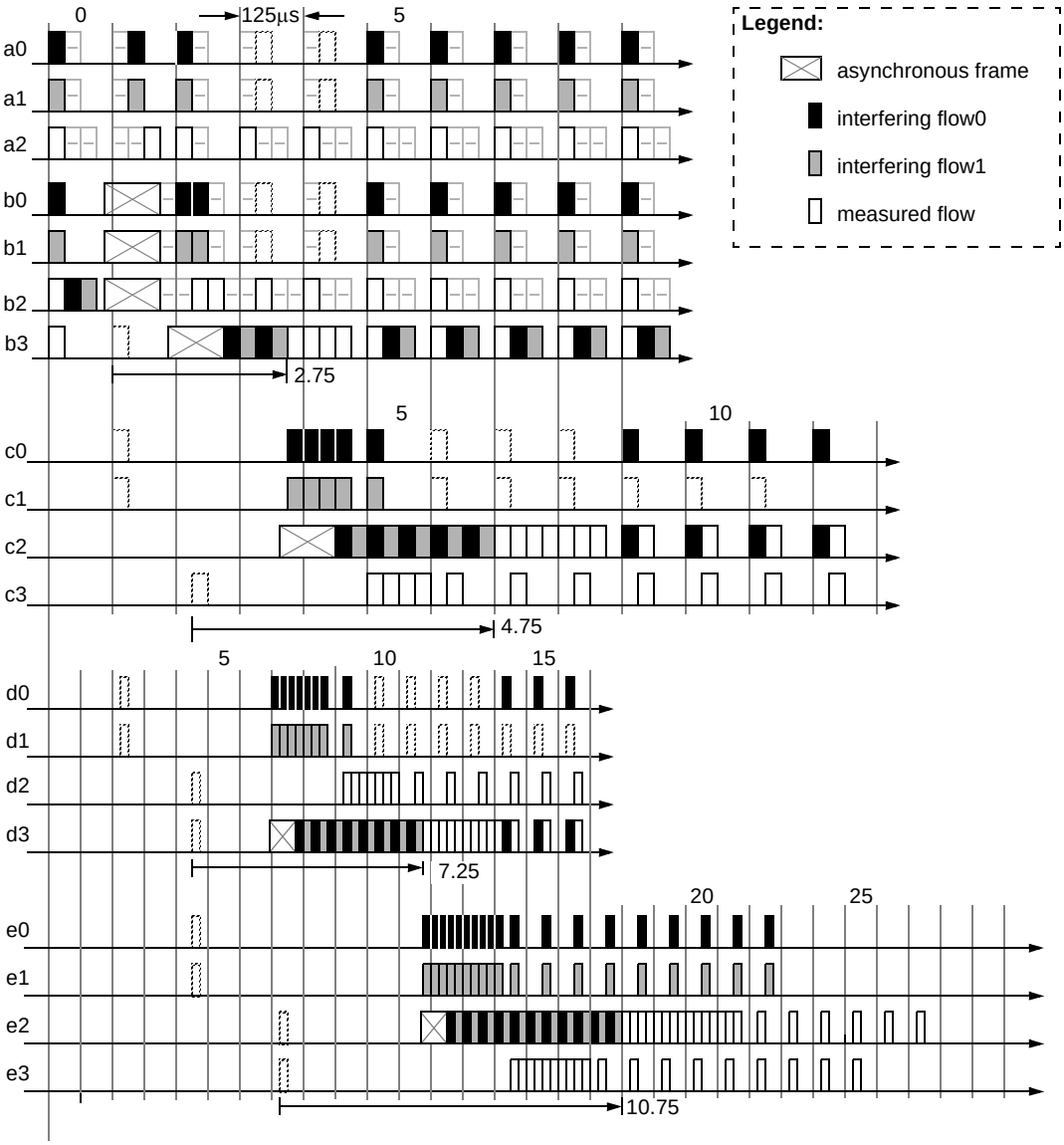

Figure A.12—Three-source bunching; variable-rate output-queue bridges

A.2.4.2 Six-source bunching; variable-rate output-queue bridges

To better illustrate the effects of worst case bunching, specific flows are illustrated in Figure A.13. Bridge ports {b0,b1,b2,b3,b4,b5} concentrates traffic from six talkers; one sixth of the cumulative traffic is forwarded through port b6. Each of six streams consumes 12.5% of the link bandwidth; 25% of the link bandwidth is available for asynchronous traffic.

For clarity, the traces for input traffic on ports {c0,c1,c2,c3,c4,c6}, {d0,d1,d2,d3,d4,d5}, and {e0,e1,e2,e3,e4,e6} only illustrate passing-through traffic; the remainder of the traffic is routed elsewhere.

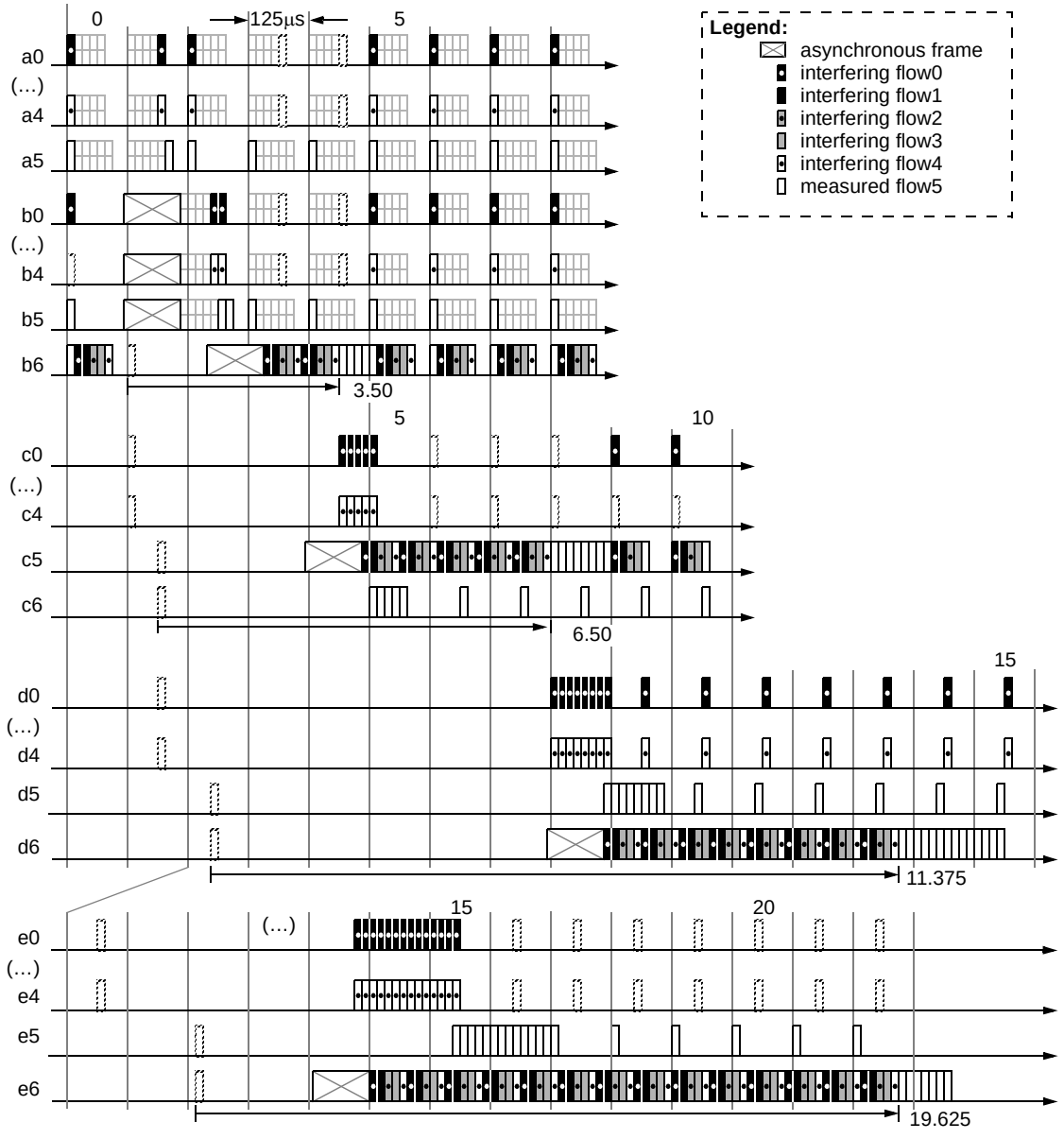

Figure A.13—Six source bunching; variable-rate output-queue bridges

Contribution from: dvj@alum.mit.edu.

A.2.4.3 Cumulative bunching latencies; variable-rate output-queue bridge

The cumulative worst-case latencies implied by coincidental bursting are listed in Table A.4 and plotted in Figure A.14.

Table A.4—Cumulative bunching latencies; variable-rate output-queue bridge

Topology	Units	Measurement point							
		A	B	C	D	E	F	G	H
3-source (see A.2.2.1)	cycles	0.75	2.75	4.75	7.25	10.75	–	–	–
	ms	0.10	0.34	0.59	0.90	1.34	–	–	–
6-source (see A.2.2.2)	cycles	0.75	3.50	6.50	11.38	19.63	–	–	–
	ms	0.10	0.44	0.81	1.42	2.45	–	–	–

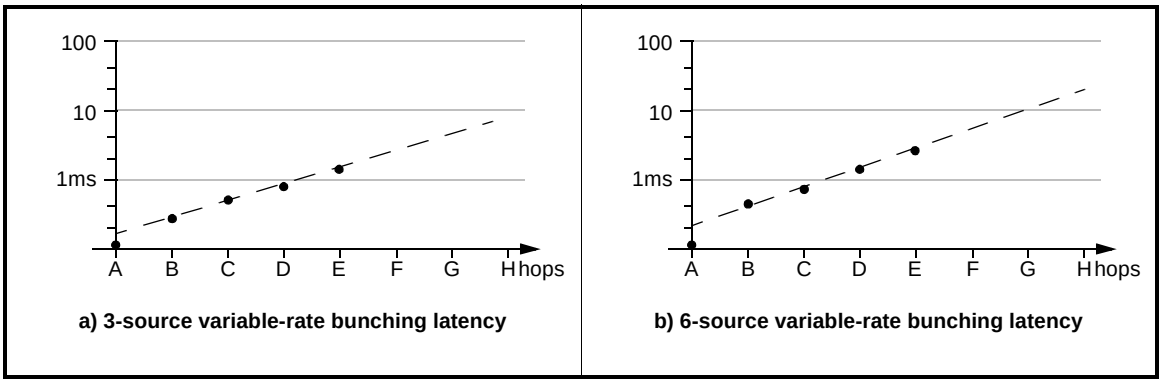

Figure A.14—Cumulative bunching latencies; variable-rate output-queue bridge

Conclusion: For nonsteady-state classA traffic, significant expediential latencies are introduced by output-queue bridges on 75% loaded links. Unfortunately, throttled outputs further exasperates this latency (see A.2.4).

A.2.5 Bunching topology scenarios; throttled-rate output-queue bridges

A.2.5.1 Three-source bunching; throttled-rate output-queue bridges

To illustrate the effects of worst case bunching, specific flows are illustrated in Figure A.15. Bridge ports {b0,b1,b2} concentrates traffic from three talkers; one third of the cumulative traffic is forwarded through port b3. Each stream consumes 25% of the link bandwidth; 25% of the link bandwidth is available for asynchronous traffic.

For clarity, the traces for input traffic on ports {c0,c1,c3}, {d0,d1,d2}, and {e0,e1,e3} only illustrate the passing-through listener traffic; the remainder of the traffic is assumed to be routed elsewhere.

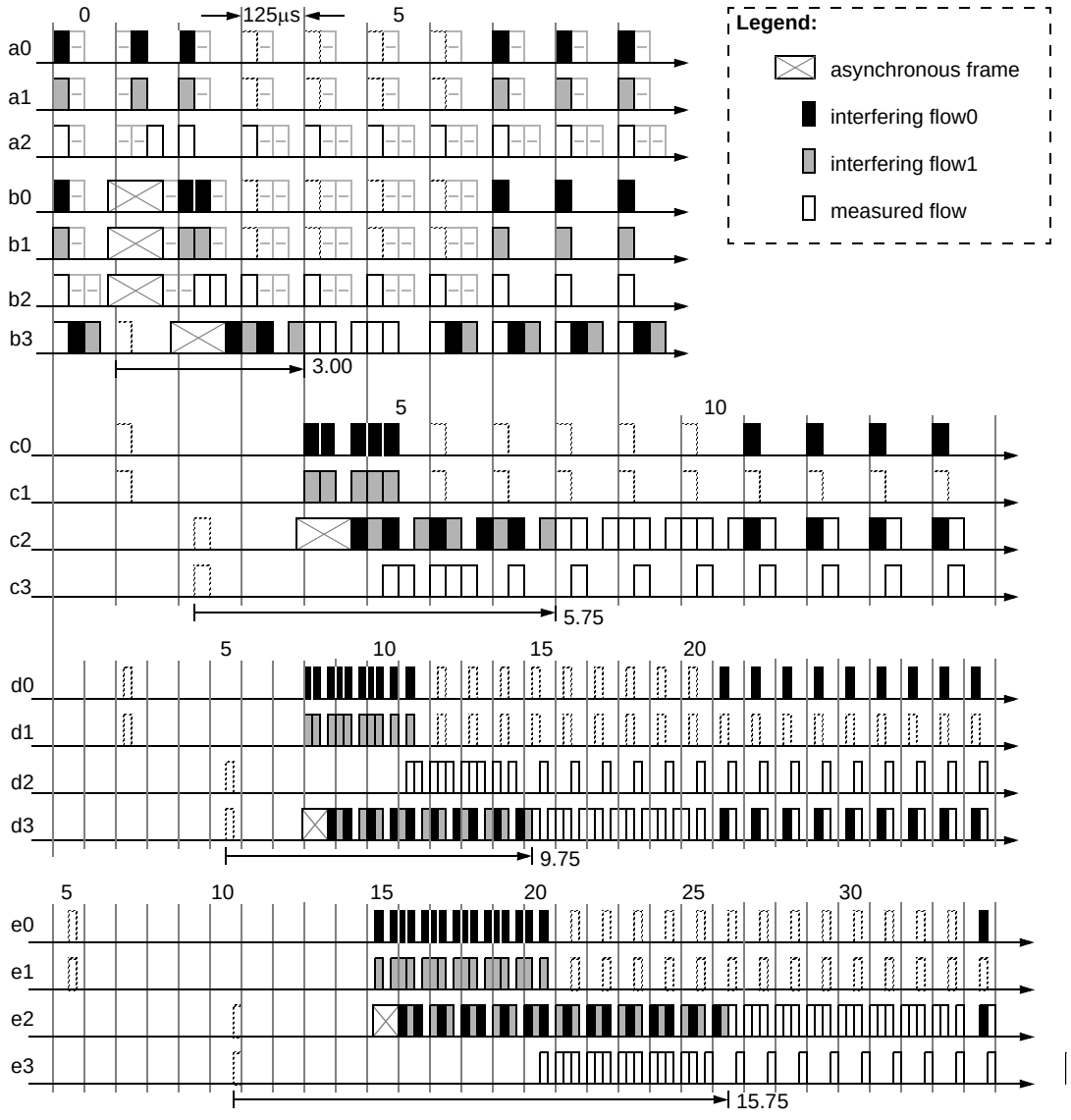

Figure A.15—Three-source bunching; throttled-rate output-queue bridges

A.2.5.2 Six-source bunching; throttled-rate output-queue bridges

To better illustrate the effects of worst case bunching, specific flows are illustrated in Figure A.16. Bridge ports {b0,b1,b2,b3,b4,b5} concentrates traffic from six talkers; one sixth of the cumulative traffic is forwarded through port b6. Each of six streams consumes 12.5% of the link bandwidth; 25% of the link bandwidth is available for asynchronous traffic.

For clarity, the traces for input traffic on ports {c0,c1,c2,c3,c4,c5},…,{e0,e1,e2,e3,e4,e6} only illustrate passing-through traffic; the remainder of the traffic is routed elsewhere.

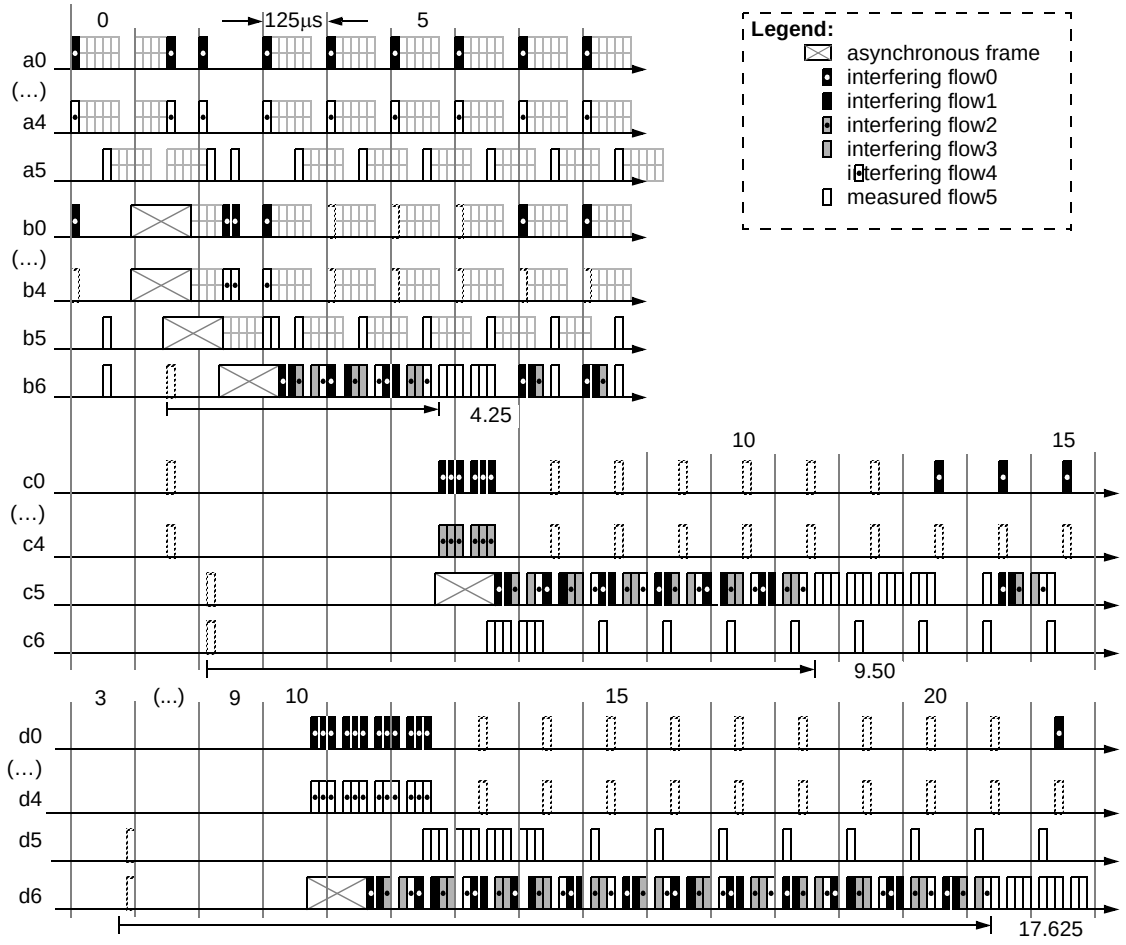

Figure A.16—Six source bunching; throttled-rate output-queue bridges

A.2.5.3 Cumulative bunching latencies; throttled-rate output-queue bridge

The cumulative worst-case latencies implied by coincidental bursting are listed in Table A.5 and plotted in Figure A.17.

Table A.5—Cumulative bunching latencies; throttled-rate output-queue bridge

Topology	Units	Measurement point							
		A	B	C	D	E	F	G	H
3-source (see A.2.2.1)	cycles	0.75	3.00	5.75	9.75	15.75	–	–	–
	ms	0.09	0.38	0.73	1.21	1.97	–	–	–
6-source (see A.2.2.2)	cycles	0.75	4.25	9.5	17.63	–	–	–	–
	ms	0.09	0.53	1.19	2.20	–	–	–	–

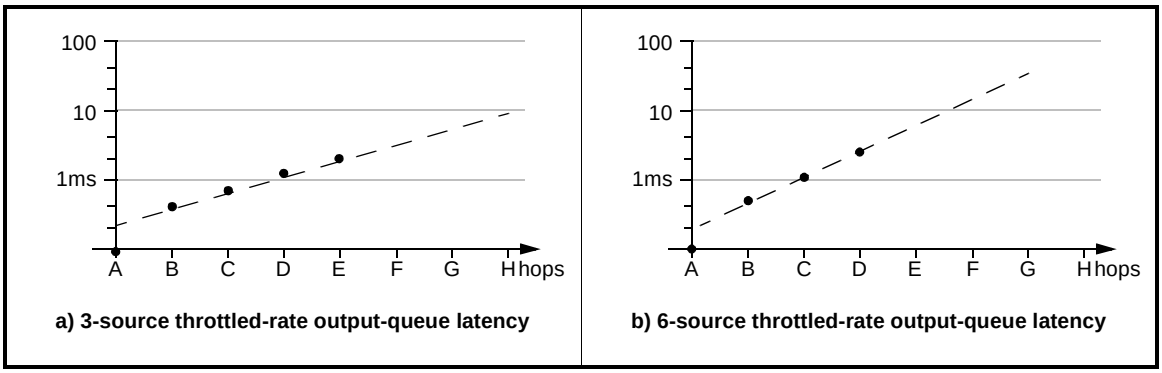

Figure A.17—Cumulative bunching latencies; throttled-rate output-queue bridge

Conclusion: On large topologies, the classA traffic latencies can accumulate beyond acceptable limits. Some form of receiver retiming may therefore be desired.

A.2.6 Bunching topology scenarios; classA throttled-rate output-queue bridges

The extent of bunching extent is worst when large classC frames are present. However, bunching can also occur in the absence of large classC frames, as described in the remainder of this subannex.

A.2.6.1 Three-source bunching; classA throttled-rate output-queue bridges

To illustrate the effects of worst case bunching, specific flows are illustrated in Figure A.18 and Figure A.19. Bridge ports {b0,b1,b2} concentrates traffic from three talkers; one third of the cumulative traffic is forwarded through port b3. Each stream consumes 25% of the link bandwidth; 25% of the link bandwidth is available for asynchronous traffic.

For clarity, the traces for input traffic on ports {c0,c1,c3}, {c0,d1,d2}, and {e0,e1,e3} only illustrate the passing-through listener traffic; the remainder of the traffic is assumed to be routed elsewhere.

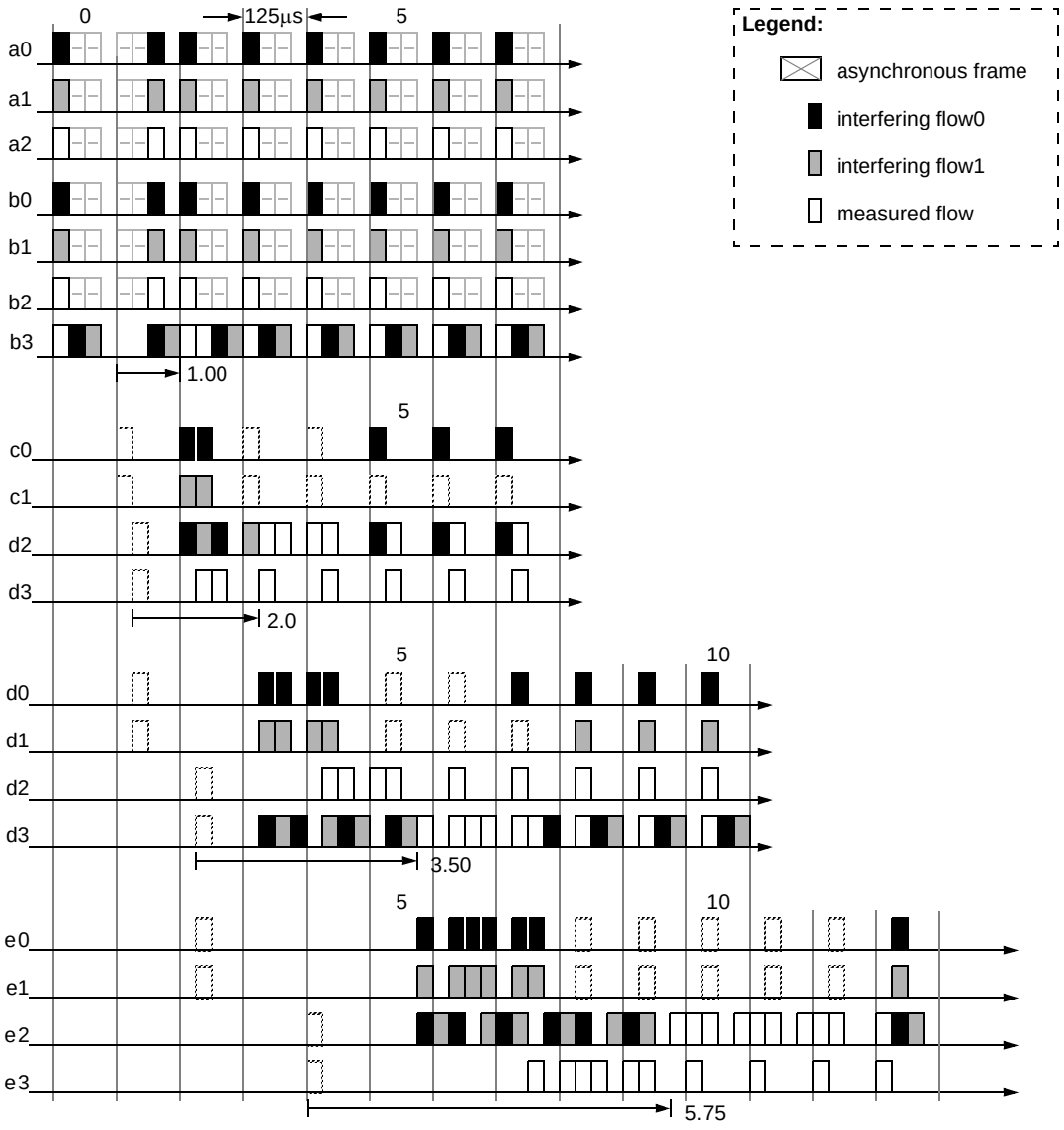

Figure A.18—Three-source bunching; throttled-rate output-queue bridges

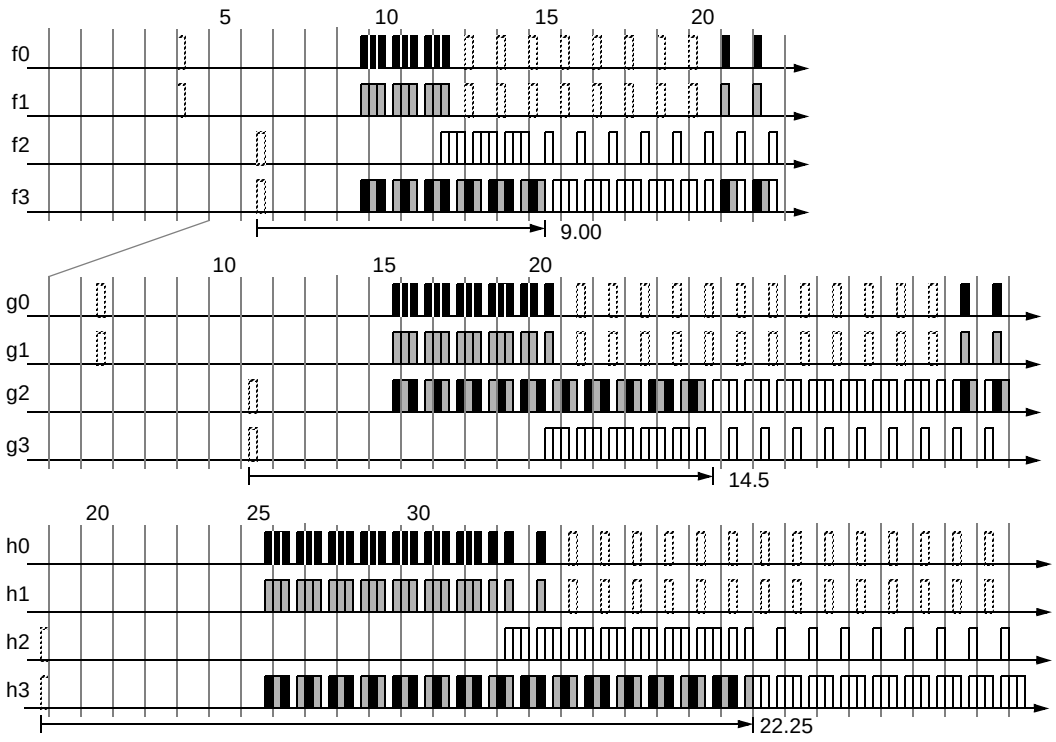

Figure A.19—Three-source bunching; throttled-rate output-queue bridges

A.2.6.2 Six-source bunching; classA throttled-rate output-queue bridges

To better illustrate the effects of worst case bunching, specific flows are illustrated in Figure A.20. Bridge ports {b0,b1,b2,b3,b4,b5} concentrates traffic from six talkers; one sixth of the cumulative traffic is forwarded through port b6. Each of six streams consumes 12.5% of the link bandwidth; 25% of the link bandwidth is available for asynchronous traffic.

For clarity, the traces for input traffic on ports {c0,c1,c2,c3,c4,c5},…,{d0,d1,d2,d3,d4,d6} only illustrate passing-through traffic; the remainder of the traffic is routed elsewhere.

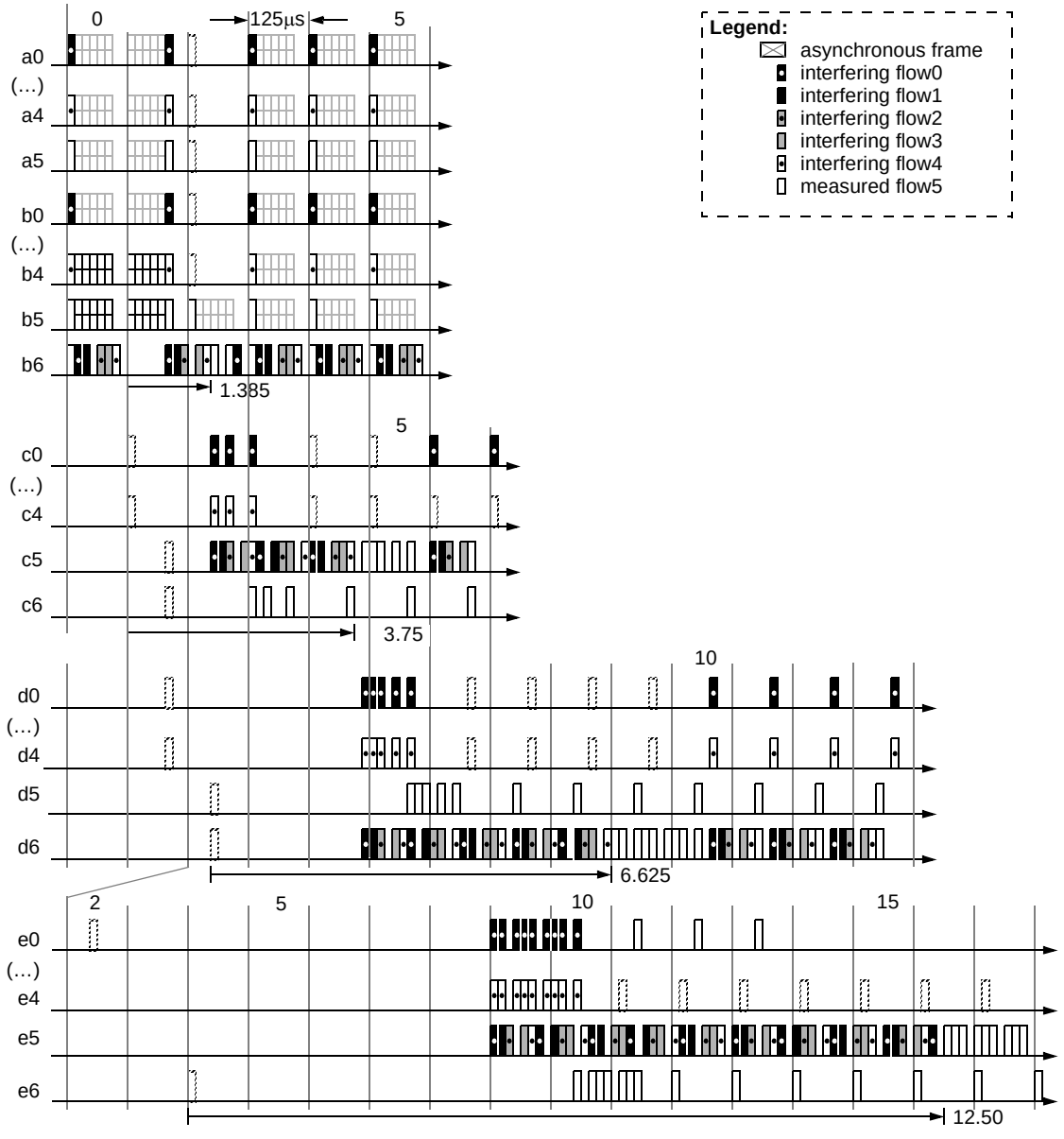

Figure A.20—Six source bunching; classA throttled-rate output-queue bridges

A.2.6.3 Cumulative bunching latencies; classA throttled-rate output-queue bridge

The cumulative worst-case latencies implied by coincidental bursting are listed in Table A.6 and plotted in Figure A.21.

Table A.6—Cumulative bunching latencies; classA throttled-rate output-queue bridge

Topology	Units	Measurement point							
		A	B	C	D	E	F	G	H
3-source (see A.2.2.1)	cycles	–	1.00	2.00	3.5	5.75	9.00	14.5	22.5
	ms	–	0.125	0.25	0.44	0.72	1.13	1.81	2.81
6-source (see A.2.2.2)	cycles	–	1.385	3.75	6.625	12.50	–	–	–
	ms	–	0.17	0.47	0.83	1.56	–	–	–

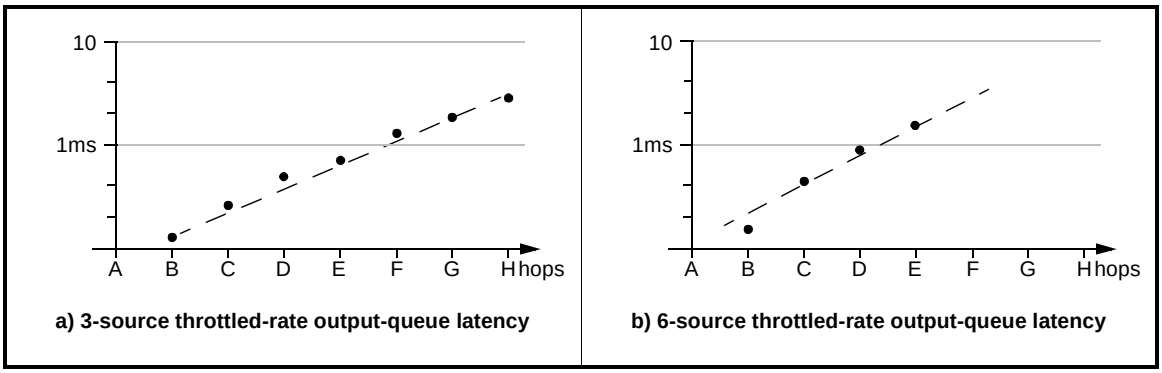

Figure A.21—Cumulative bunching latencies; classA throttled-rate output-queue bridge

Conclusion: On large topologies, the classA traffic latencies can accumulate beyond acceptable limits, even in the absence of conflicting lower-class traffic. Some form of receiver retiming may therefore be desired, even on higher speed links where the size of the MTU (in time) becomes much smaller than an assumed 8 kHz cycle time.