dvjClause06TxRx02.fm

March 8, 2003 5:57 pm
Contribution to IEEE Draft P802.17-D2.1

Transmit/receive rules

This writeup clarifies proposed transmit/receive rules for known unicast, unknown unicast, multicast, and broadcast types of traffic. Only the flooding and MAC-address checking features are considered; *ttl* and error checking actions have been eliminated for clarity.

Reformatting the current receive rules, in this fashion, would eliminate bugs with documentation, implementation, and understanding.

The editor has identified through, clause numbers, the intended destination for this content. The request is to include all of the subclause 6 content. If that is found to be unacceptable, then the concerns with the current specification, that shall be addressed, include the following:

a) Correct intent, but ambiguous/erroneous details in D2.1 Row 6.16-6 (sa is context dependent)
Replace with DVJ Row 1.4-6 and Row 1.4-7, which demonstrate the need for distinct checks.

b) Incorrect precedence; an not-possible state should precede a transient discard decision.
Switch D2.1, Row 6.16-8 and Row 6.16-8.

c) Correct intent, but ambiguous/erroneous details in D2.1 Row 6.16-12.
Distinct *fcs* checks on idle/fairness and data/control are required, due to distinct coverage range.
The way this is currently phrased, idle and fairness frames are always discarded.
Replace D2.1 Row 6.16-12 with DVJ Row 1.4-13 to Row 1.4-17.

d) Data and control should be processed in the same fashion.
Discard of control frame payloads effectively outlaws the use of store&forward.
Eliminate “frame.ft==FT_DATA&&” from D2.1 D2.1 Row 6.16-13.

e) There should be an error check, when passing the source station for the second time:
Include DVJ Row 1.4-7.

f) The discard rules do not appear to handle discarding at steer-only stations.
Include DVJ Row 1.8-2.

g) The adjustment rules do not appear to handle transmitted frames.
Include DVJ Row 1.8-4 & Row 1.8-4.

1.10 Routing rules

The transmit, receive, and filter components, illustrated Figure 1.1, utilize rules designed to support bridged as well as local RPR traffic.

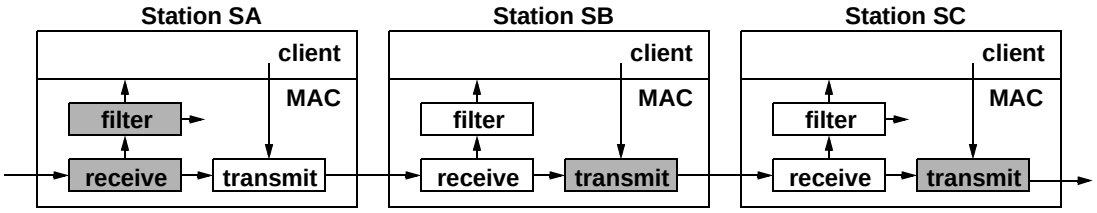

Figure 1.1—Routing components

For a station SA-to-SC transmission, the white-shaded boxes are involved: transmit rules affect the source station SA; the receive rules affect intermediate station SB; the receive and filter rules affect station SC. These rules are summarized in the following subclauses.

1.10.1 Transmit rules

The transmit rules affect the selection of flood-field values and the use of basic/extended formats, as summarized in Table 1.1. Within this context, MSID represents the station's MAC address, as reported through topology discovery.

Table 1.1—Transmit rules

Client parameters		Row	Header parameters			Payload parameters	
clientDa	clientSa		flood	da	sa	da	sa
—	!MSID	1	1	clientDa	MSID	clientDa	clientSa
local	MSID	2	0	clientDa	clientSa	-na-	-na-
—	MSID	3	1	clientDa	clientSa	-na-	-na-

Row 1.1-1: Flood the remote-source frames using the extended frame format.
NOTE—Including local station's MAC address allows the frame to be reliably source stripped, rather than accidentally learned, when unexpectedly passing by the source station.

Row 1.1-2: Direct the local-source known-unicast frame using the basic frame format.
Row 1.1-3: Flood other local-source frames using the basic frame format.
NOTE—All stations must see a broadcast, but sophisticated sources can restrict flood scoping as follows:
1) Multicast. Scoping can be reduced to those stations within the multicast group.
2) Unknown unicast. Scoping can be reduced to those bridge-capable stations.

1.10.2 Receive rules

The receive rules affect the copy and strip actions, as summarized in Table 1.2. Within this context, MSID represents the station's MAC address, as reported through topology discovery.

Table 1.2—Receive rules

Frame parameters			Row	Receive actions	
clientDa	**clientSa**	**flood**		**copy**	**strip**
!MSID	MSID	—	1	0	1
MSID	—	—	2	1	1
!MSID	—	0	3	0	0
		1	4	1	0

Row 1.2-1: Self-sourced/other-address frames are assumed to be erroneous and are stripped.
Row 1.2-2: Self-destined frames are copied and stripped.
NOTE—The copy action supports flush-frame routing; such data frames are filtered before the client (see 1.10.3).
Row 1.2-3: Others' not-flooded frames are ignored.
Row 1.2-4: Flooded frames are copied for the client.
NOTE—This copy action support bridges; others' flooded frames are filtered before reaching host clients (see 1.10.3).

1.10.3 Filter rules

The receive rules affect the discarding of data frames destined to the client, as summarized in Table 1.3. Within this context, MSID represents the station's MAC address, as reported through topology discovery. The client-configurable ***onlyMe*** bit is expected to be set to 0 and 1, for bridge and host stations respectively.

Table 1.3—Filter rules

Frame parameters			onlyMe	Row	Discard action
clientDa	**clientSa**	**flood**			
—	MSID	—	—	1	toss
MSID	—	—	—	2	pass
unicast	—	—	1	3	toss
—	—	—	—	4	pass

Row 1.3-1: Self-sourced frames are discarded before reaching the sourcing client.
Row 1.3-4: Self-addressed frames (that are not self-sourced) are passed to the client.
Row 1.3-3: If *onlyMe*==1, others' irrelevant unicast frames are discarded, rather than sent to the clients.
Row 1.3-4: Other frames are passed to the client.

6.8 Routing rules

Each station has transmit, receive, and filter components, as illustrated Figure 1.1, which utilize rules designed to support bridged as well as local RPR traffic. (See Figure 1.1 for a simplified MAC-address summary of these functions.)

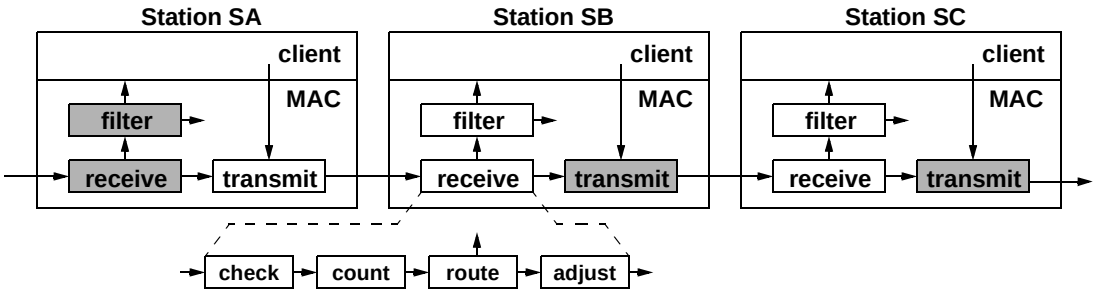

Figure 1.2—Routing components

The receive functions include multiple components, listed below and detailed in following subclauses.

a) check. Erroneous incoming frames are discarded or adjusted; MIB-related error statistics are updated.
b) count. Parameters in valid incoming frames are used to update MIB-related flow statistics.
c) route. Parameters in valid incoming frames are used to selective enable copies, as follows:
   1) receive. A copy is passed towards the receiver filter.
   2) transmit. A copy is passed towards the transmit component.
d) adjust. Parameters within the frame (time-to-live and past-status fields) are adjusted (as specified) and the header's CRC16-based *hec* field are adjusted (whenever its protected fields have changed).

6.8.1 Check rules

The check rules affect the early discarding of erroneous frames, as summarized in Table 1.4.

Table 1.4—Check rules

Current state		Row	Next state	
state	**condition**		**action**	**state**
START	(frame= Fetch())==NULL	1	—	START
	COMPACT&& frame.parity!=Parity(frame.header)	2	// Discard frame w/out MIB adjustment	
	!COMPACT&& frame.hec!=Crc16(frame.header)	3	rprSpanStatsErrorBadHecPkts+= 1;	
	frame.ttl==0	4	rprSpanStatsErrorTtlExpPkts+= 1;	
	COMPACT&&Multicast(frame.sa)	5	// Discard frame w/out MIB adjustment	
	!COMPACT&&Multicast(frame.ssid)	6	// Discard frame w/out MIB adjustment	
	frame.ri!=port.ri&& frame.we==0	7	// Discard frame w/out MIB adjustment	
	frame.ri!=port.ri && station.wrapCapable==0	8	// Discard frame w/out MIB adjustment	
	frame.ri!=port.ri&& port.protectState==IDLE	9	// Discard frame w/out MIB adjustment	
	frame.ri!=port.ri&& frame.sa==SSID&&frame.ps==1	10	// Discard frame w/out MIB adjustment	
	frame.ri==port.ri&& port.ws==1&&frame.ps==1	11	// Discard frame w/out MIB adjustment	
	frame.ri!=port.ri&& port.ws==1&&frame.ps==0	12	// Discard frame w/out MIB adjustment	
	COMPACT&& frame.fcs!=Crc32(frame.header)	13	// Discard frame w/out MIB adjustment	
	COMPACT	14	ToRoute();	
	frame.ft==FT_CONTROL&&DEP&& frame.fcs!=Crc32(frame.payload)	15	// Discard frame w/out MIB adjustment	
	!COMPACT&& frame.fcs==Stomp(frame.payload)	16	frame.extra.badData= 1;	
	!COMPACT&& frame.fcs!=Crc32(frame.payload)	17	rprSpanStatsErrorBadFcsPkts+= 1, frame.fcs=Stomp(frame.payload), frame.extra.badData= 1;	
	—	18	ToCount(frame);	

#define COMPACT ((frame.ft==FT_IDLE||frame.ft==FT_FAIRNESS) ? COMPACT:!COMPACT)
#define Stomp(x) (0XFFFFFFFF^Crc32(x))

Editors' Notes: *To be removed prior to final publication.*
Many of these error conditions have no corresponding MIB update. Should they have one?

Row 1.4-1: Wait for the next frame to arrive.
Row 1.4-2: If the header parity does not match its expected value, discard the frame.
Row 1.4-3: If the header CRC does not match its expected value, discard the frame.
Row 1.4-4: Expired frames are discarded.
Row 1.4-5: Group source addresses within idle&fairness frames are illegal and therefore discarded.
Row 1.4-6: Group source addresses within data&control frames are illegal and therefore discarded.
(Distinct source-address checks are necessary because the *sa* field location is frame-type dependent.)

Row 1.4-7: Steer-only frames are discarded when apparently wrapped.
Row 1.4-8: Wrapped frames are discarded on a steered ring.
Row 1.4-9: Wrapped frames are discarded when the ring is not apparently wrapped.
Row 1.4-10: Wrapped frames are discarded if the pass through the source station twice.
Row 1.4-11: Wrapped frames are discarded when about to be rewrapped.
Row 1.4-12: Wrapped frames are discarded at a the unwrapping point is their source has not been passed.
Row 1.4-13: A bad CRC in an idle/control frame causes the frame to be discarded.
Row 1.4-14: A valid idle/control frame is passed onward.

Row 1.4-15: A control frame with a bad data-payload CRC may be optionally discarded.
NOTE—These corrupted frames are useless, but a store-and-forward implementation can't always delete them.
Row 1.4-16: A stomped data-payload CRC is flagged for the recipient, but otherwise processed normally.
Row 1.4-17: A bad data-payload CRC is stomped, an error is logged, and passed onward.
Row 1.4-18: A good payload CRC is processed normally.

6.8.2 Count rules

The count rules affect the early discarding of erroneous frames, as summarized in Table 1.5.

Table 1.5—Attribute update rules

Current state		Row	Next state	
state	**condition**		**action**	**state**
START	(frame= Fetch())==NULL	1	—	START
	!COMPACT&&UNICAST(frame.da) &&CLASS_A(frame)	2	rprSpanStatsInUcastClassAPkts+= 1, rprSpanStatsInUcastClassBOctets+= B;	EXEC
	!COMPACT&&UNICAST(frame.da) &&CLASS_B0(frame)	3	rprSpanStatsInUcastClassBCirPkts+= 1, rprSpanStatsInUcastClassBCirOctets+= B;	
	!COMPACT&&UNICAST(frame.da) &&CLASS_B1(frame)	4	rprSpanStatsInUcastClassBEirPkts+= 1, rprSpanStatsInUcastClassEirOctets+= B;	
	!COMPACT&&UNICAST(frame.da) &&CLASS_C(frame)	5	rprSpanStatsInUcastClassCPkts+= 1, rprSpanStatsInUcastClassCOctets+= B;	
	!COMPACT&&!UNICAST(frame.da) &&CLASS_A(frame)	6	rprSpanStatsInMcastClassAPkts+= 1, rprSpanStatsInMcastClassBOctets+= B;	
	!COMPACT&&!UNICAST(frame.da) &&CLASS_B0(frame)	7	rprSpanStatsInMcastClassBCirPkts+= 1, rprSpanStatsInMcastClassBCirOctets+= B;	
	!COMPACT&&!UNICAST(frame.da) &&CLASS_B1(frame)	8	rprSpanStatsInMcastClassBEirPkts+= 1, rprSpanStatsInMcastClassEirOctets+= B;	
	!COMPACT&&!UNICAST(frame.da) &&CLASS_C(frame)	9	rprSpanStatsInMcastClassCPkts+= 1, rprSpanStatsInMcastClassCOctets+= B;	
EXEC	—	10	ToRoute(frame);	START

```
#define COMPACT ((frame.ft==FT_IDLE||frame.ft==FT_FAIRNESS) ? COMPACT:!COMPACT)
#define UNICAST(x) (((x>>32)&1)==0)
#define B SizeOfFrame(frame)
#define CLASS_A(x) (x.baseRingControl.sc==CLASS_A0||x.baseRingControl.sc==CLASS_A1)
#define CLASS_B0(x) (x.baseRingControl.sc=CLASS_B&&x.baseRingControl.fe==0)
#define CLASS_B1(x) (x.baseRingControl.sc=CLASS_B&&x.baseRingControl.fe==0)
#define CLASS_C(x) (x.baseRingControl.sc=CLASS_C)
```

Editors' Notes: *To be removed prior to final publication.*
Many of these error conditions have no corresponding MIB update. Should they have one?

Row 1.4-1: Wait for the next frame to arrive.
Row 1.4-2: If the header parity does not match its expected value, discard the frame.
Row 1.4-3: If the header CRC does not match its expected value, discard the frame.
Row 1.4-4: Expired frames are discarded.
Row 1.4-5: Group source addresses within idle&fairness frames are illegal and therefore discarded.
Row 1.4-6: Group source addresses within data&control frames are illegal and therefore discarded.
(Distinct source-address checks are necessary because the *sa* field location is frame-type dependent.)

Row 1.4-7: Steer-only frames are discarded when apparently wrapped.
Row 1.4-8: Wrapped frames are discarded on a steered ring.
Row 1.4-9: Wrapped frames are discarded when the ring is not apparently wrapped.
Row 1.4-10: Wrapped frames are discarded if the pass through the source station twice.
Row 1.4-11: Wrapped frames are discarded when about to be rewrapped.
Row 1.4-12: Wrapped frames are discarded at a the unwrapping point is their source has not been passed.
Row 1.4-13: A bad CRC in an idle/control frame causes the frame to be discarded.

Row 1.4-17: A stomped data-payload CRC is flagged for the recipient, but otherwise processed normally.
Row 1.4-16: A bad data-payload CRC is stomped and processed normally.
(A stomped CRC is a distinctive bad CRC value, as described in F.2.3.)
Row 1.4-18: A good payload CRC is processed normally.

6.8.3 Route rules

The route rules affect the frame's copy and strip actions, as summarized in Table 1.6. Within this context, MSID represents the station's MAC address, as reported through topology discovery.

Table 1.6—Route rules

Current state		Row	Next state	
state	**condition**		**action**	**state**
START	(frame= Fetch())==NULL	1	—	START
	frame.ft == FT_IDLE	2	// Discarded idles are not errors	
	frame.ft == FT_FAIRNESS	3	ToFilter(frame);	
	CENTER_WRAP&&frame.ri!=port.ri &&port.ws==0	4	// Ignore wrapped opposing-ringlet frames	
	EDGE_WRAP&&frame.ri!=port.ri	5	// Ignore wrapped opposing-ringlet frames	
	frame.da!=MSID&&frame.sa==MSID	6	rprSpanStatsSelfSrcUcastPkts+= 1	
	frame.da==MSID	7	ToFilter(frame);	
	FLOOD&& frame.da!=MSID&&frame.sa!=MSID	8	ToFilter(frame), ToAdjust(frame);	

#define EDGE_WRAP (station.capabilities.edgeWrap)
#define CENTER_WRAP (station.capabilities.centerWrap)
#define MSID (station.macAddress)
#define FLOOD ((frame.ff==FF_UNDIR||frame.ff==FT_BIDIR)

Row 1.6-1: Wait for the next frame to arrive.
Row 1.6-2: Idle frames are quietly discarded.
Row 1.6-3: Fairness frames are always accepted.
Row 1.6-4: Opposing center-wrap frames are ignored, except at the unwrap location.
Row 1.6-5: Opposing edge-wrap frames are ignored.
Row 1.6-6: Other-destination source-stripped frames are extraneous, and are therefore stripped.
Row 1.6-7: Self-destined source-stripped frames could be a flush, and are therefore stripped.
Row 1.6-8: Others' flooded frames are copied to the filter and passed through the station.

6.8.4 Filter rules

The filter rules affect the frame's copy and strip actions, as summarized in Table 1.7. Within this context, MSID represents the station's MAC address, as reported through topology discovery.

Table 1.7—Filter rules

Current state		Row	Next state	
state	**condition**		**action**	**state**
START	(frame= Fetch())==NULL	1	—	START
	frame.so&& !ConsistentHopCount(frame)	2	// Discard frame w/out MIB adjustment	
	frame.ft==FT_CONTROL	3	ToMac(frame);	
	frame.ri!=port.ri	4	// Pass through opposing run frames	
	frame.sa==MSID	5	// Discard frame w/out MIB adjustment	
	frame.da==MSID	6	ToClient(frame);	
	UNICAST(frame.sa)&&mode.onlyMe	7	// A host can discard these flooded frames	
	—	8	ToClient(frame);	

#define UNICAST(x) (((x>>32)&1)==0)

Row 1.7-1: Wait for the next frame to arrive.
Row 1.7-2: Control or data frames that fail the source-consistency check are discarded.
Row 1.7-3: Control frames are routed to the MAC control processor.
Row 1.7-4: Pass through opposing run frames.
Row 1.7-5: Self-stripped data frames are discarded.
Row 1.7-6: Self-destined data frames are passed to the client.
Row 1.7-7: Flooded unicast frames are discarded by host stations with onlyMe==1.
Row 1.7-8: Multicast and desired flooded unicasts are passed to the client.

6.8.5 Adjust rules

The adjust rules affect the frame's copy and strip actions, as summarized in Table 1.8. Within this context, MSID represents the station's MAC address, as reported through topology discovery.

Table 1.8—Adjust rules

Current state		Row	Next state	
state	**condition**		**action**	**state**
START	(frame= Fetch())==NULL	1	—	START
	STEERING&&port.ws==1	2	// Discard frame w/out MIB adjustment	
	frame.we==0&&port.ws==1	3	// Discard frame w/out MIB adjustment	
	COMPACT	4	SetParity(frame), SetHeadFcs(frame);	
	TransmittedFromHere(frame)	5	SetHec(frame), SetDataFcs(frame);	
	frame.ri!=port.ri&&port.ws!=1&& frame.sa==SSID	6	frame.ps= 1, HeadCrc(frame),ToSend(frame);	
	frame.ri!=port.ri&&port.ws!=1	7	ToSend(frame);	
	frame.ttl<=1	8	rprSpanStatsErrorTtlExpPkts+= 1;	
	frame.ri!=port.ri&&port.ws!=1&& frame.sa==SSID	9	frame.ps= 1, frame.ttl-= 1, HeadCrc(frame),ToSend(frame);	
	frame.ri!=port.ri&&frame.ttl>=1&& port.ws==1&&frame.sa==SSID	10	frame.ttl-= 1, HeadCrc(frame),ToSend(frame);	

```
#define STEERING (station.capabilities.wrapEnables==0)
#define COMPACT ((frame.ft==FT_IDLE||frame.ft==FT_FAIRNESS) ? COMPACT:!COMPACT)
#define MSID (station.macAddress)
```

Row 1.8-1: Wait for the next frame to arrive.
Row 1.8-2: Steering frames are discarded rather than wrapped.
Row 1.8-3: Discard wrap-ineligible frames when they reach the wrap point.
Row 1.8-4: Send idle/fairness frames after generating their *parity* and *fcs* fields.
Row 1.8-5: Send control/data frames after generating their *hec* and *fcs* fields.
Row 1.8-6: Set the past-source bit when passing through the opposing-ringlet source.
Row 1.8-7: Nothing changes when passing through other opposing-ringlet stations.
Row 1.8-8: Discard fully aged frames.
Row 1.8-9: Set the past-source bit and age the frame when wrapping to the originating run.
Row 1.8-10: Age the frame when passing through originating-run stations.

Editors' Notes: *To be removed prior to final publication.*

Its not clear to DVJ what naming conventions for MIB attribute components should be used (is In/Out included in the MIB attribute name, or is this implied by the grouping, etc.).

Other MIBs that need to be incorporated into this clause include:
rprSpanStatsErrorTooLongPkts—Do we agree that this is actually checked on the span?
rprSpanStatsErrorTooShortPkts—Do we agree that this is actually checked on the span?
rprSpanStatsErrorUnknownPktType—Do we agree that this is actually checked on the span?
rprSpanStatsErrorPmdAbortPkts—How would this be incorporated?