MRP State Machines

Mick Seaman

This note proposes Applicant and Registrar state machines for MRP and MVRP[a]. These provide the support necessary to limit the effect of topology changes to S-VLANs whose underlying connectivity has changed[b], for rapid point-to-point operation, and to reduce the number of PDUs for large numbers of attributes.

These machines provide efficient point-to-point and shared media operation when operPointToPointMAC is correctly configured, and correct (but noisy) operation when it is not. They accommodate more attributes than can be packed in a single frame, though all 4096 VLANs can be accommodate within 1024 octets for point-to-point operation.

Protocol Components

For each MRP Application, MRP comprises an Attribute Declaration (MAD) component for each Port, an Attribute Propagation (MAP) component for each Attribute Propagation Context, and the MRP Application components proper. These correspond to GID, GIP, GIP Context, and GARP Applications for GARP[1] as illustrated in 802.1D-2004 Figure 12-4.

This note includes two equivalent specifications of the MAD component. The first uses protocol variables (and may be an easier introduction), and the second conventional state tables (and is easier to check for completeness and correctness). Events and actions are the same for both alternatives.

Topology change signaling

To communicate topology changes for MVRP, MRP itself provides change (or more specifically new registration) signaling for attributes[2]. This signaling can be used by other applications that require an efficient mechanism for indicating the arrival of new registrants. To simplify discussion of this capability, this note may occasionally blur the distinction between MVRP and MRP.

Point-to-point operation

GARP was designed[3] principally to meet the challenges of effective operation on shared media. While an important role remains for virtual shared media over point-to-point LANS, MRP can take advantage of three major points to speed point to point operation.

First, for shared media with many participants it is necessary to jitter timers, whereas for point-to-point, one PDU can be sent immediately, with rate limiting then imposed to space subsequent transmissions as discussed below.

Second, since each point-to-point participant only has one peer, it is unnecessary for the Registrar to delay a transition to the unregistered state. Unwanted declarations can be removed immediately.

Third, which is easy to miss, the purpose of advertising the Registrar state in attribute messages is not to provide an acknowledged protocol. Once the 'channel', i.e. buffering, reception, scheduling, and transmission delays, between protocol participants is accounted for there is little gain from using the Registrar state in this way. Rather, the Registrar state is provided so that one Applicant can observe and if necessary compensate for the effects of others making and withdrawing applications. There are no 'others' on point-to-point LANs. MRP takes advantage of this, using a reduced set of messages and simpler operation for point-to-point, while avoiding the messy effects of a 'mode-switch' if another participant attaches to virtual media and 'thrashing' if the point-to-point designation is incorrect[4].

---

[a] The proposed PAR for the work can be found at
../docs2004/802-1ai-draft-par-for-30-day-rule.htm
with background information in
nfinn-vectorized-garp.zip
and earlier introductory material in
MVRP-Introduction-030.pdf
though the proposed topology change mechanism is differs.

[b] As opposed to flushing learnt addresses for a VLAN as a result of changes in part of its spanning tree not used by that VLAN.

[1] Apart from aligning with the GARP to MRP name change, the new names identify that the functionality is slightly different.

[2] Of course signaling of a new registration has to be propagated through LANs where the declaration has already been made, arriving possibly through a different bridge port, which is why it is an addition to the protocol's capabilities.

[3] In an era when widespread use of on demand video on LANs was anticipated, but most of the end stations were attached to repeaters, not bridges.

[4] Looking forward to the next "buffered repeater".

Protocol messages

Each MRP participant send 'messages', each conveying for a specific attribute:

- the Applicant state – 'Join' for declaring the attribute, and its opposite ' '
- any change in a declaration – 'New' for a change to 'Join' (either directly or propagated), 'Leave' for a change to withdraw the declaration
- the Registrar state – 'In' if the attribute is registered[5], 'Mt' (empty) otherwise.

MRP uses the following message types:

- NewIn, NewMt[6]
- JoinIn, JoinMt
- Lv[7]
- Mt
- Null – facilitates compact encoding[8]

On point-to point-media there is no need to distinguish 'Leave' from the applicant state ' ', nor to include the Registrar state, so the following reduced set of messages is used:

- New
- Join
- Lv
- Null

The Null message is not necessary for point-to-point, but provides interoperability with shared media operation, mitigating misconfiguration.

Periodically MRP participants send messages to 'garbage collect' declarations that have been withdrawn, but are still registered due to exceptional message loss, and to ensure that loss of declarations does not similarly lead to persistent lack of registration. Shared media participants send LeaveAll (LA) messages that cause new declarations to be made by all participants[9]. Point-to-point participants send LeaveMine (LM) messages. Other participant(s) registrations are aged for only a short while after receipt of a LeaveMine[10], to allow redeclaration of attributes that would not fit into the PDU.

[5] To be exact 'In' is only signalled if the Registrar is in the 'IN' state, and 'Mt' if the Registrar state is 'LV' or 'MT'.

[6] GARP does not provide new registration signaling and hence does not use NewIn and NewEmpty.

[7] LeaveIn and LeaveEmpty could be distinguished, but that serves little purpose.

[8] Not only is there little purpose to an explicit 'In' message, but when sent = 2 sending JoinIn is actually to be avoided.

[9] Subject of course to the normal mechanisms for suppressing excess messages.

[10] Use of the LeaveMine message means that a new 'prompt' like mechanism and message type does not have to be introduced for point-to-point to guard against the loss of a LeaveAll.

PDU Transmission

Transmission is organized as follows. Each Applicant (one per attribute, per MAD component) and the LeaveAll state machine (one per MAD component) indicates a need to transmit by requesting a transmit opportunity. A state machine for the entire component then provides the opportunity (finding a buffer etc.) either immediately, or imposing a time delay as required to provide jitter and rate limiting. If a future opportunity has already been scheduled, there is no need for another. Thus only one timer and few counters are required per Port.

It is possible that a single PDU is insufficient to accommodate all the messages to be transmitted. In this specification the passing of a transmit opportunity is an event, allowing participants to request a further opportunity.

Protocol events

- Begin! – initialize for this attribute
- New! – A new declaration
- Join! – without signaling new registration
- Lv! – Withdraw a declaration
- tx! – Transmission opportunity without a LeaveAll or LeaveMine[11]
- txLA! txLM! – with a LeaveAll, LeaveMine
- txF! txLAF! txLM! – with no room, F(ull)
- rNew! rNewIn! rNewMt!
  rJoin! rJoinIn! rJoinMt!
  rLv!
  rMt!
  rNull!
  rLA! rLM! rLI! – protocol message receipt

Protocol actions

- New, Join, Lv – indications to MAP and the MRP application
- ntt – need to transmit, also referred to as request transmit opportunity
- sN – send a New message (on a point-to-point LAN), or a NewIn or NewEmpty message (on shared media) as determined by the Registrar
- sJ – send a Join, JoinIn, or JoinMt
- sL, sE – send a Lv, send Mt
- [s] – send a Null if required for encoding

[11] LeaveAlls are always encoded before messages for specific attributes. GARP does not distinguish tx from txLA, so the LO state that guards against message loss for attribute redeclaration may not when the Empty 'prompt' is encoded in the same PDU.

Applicant protocol variables

Each MRP Applicant's operation can be specified using the following protocol variables:

- `member - true/false`
- `active - true/false`
- `change - true/false`
- `sent - 0, 1, 2`

together with the condition

- `p2p - true/false`

Table 1 and Table 2 show the correspondence between these variables and states suitable for a state table description for shared media and point-to-point operation respectively.

Variable	VC	AC	VA	AA	QA	VP	AP	QP	LA	LO	VO	AO	QO
member	T	T	T	T	T	T	T	T	F	F	F	F	F
active	T	T	T	T	T	F	F	F	T	F	F	F	F
change	T	T	F	F	F	F	F	F	-x-	T	F	F	F
sent	0	1	0	1	2	0	1	2	-x-	0	0	1	2

Table 1 – MAD Applicant protocol variables and shared media operation states

Variable	VC	AC	VA	AA	QA	LA	LO	VO
member	T	T	T	T	T	F	F	F
active	T	T	T	T	T	T	F	F
change	T	T	F	F	F	-x-	T	F
sent	0	1	0	1	2	-x-	0	0

Table 2 – MAD Applicant protocol variables and point-to-point operation states

Example Code

The following code implements an Applicant capable of receiving messages from participants that believe themselves attached to either point-to-point or shared media, with any number of each type participating at the same time. The Applicant's own behavior is determined by the p2p variable. Note that txLA! and txLAFull! events will not occur if p2p is true, and txLM! and txLMFull! will not if it is false. The (possibly transient) condition that requires a message to be encoded for an attribute to satisfy some aspect of compact message encoding is denoted by the variable 'encode', and sending a Null, In, or Empty by s().

State Table Specification

An equivalent state table description follows the code (Table 3). In case of any discrepancy the state table takes precedence. The differences between Table 3 and Table 12-3 of 802.1D-2004 for events they have in common are:

1. Table 12-3 contains entries for both LeaveIn and LeaveEmpty, which leads to differences for the VA, AA, QA and LA states. The most conservative of the next state selections is used in Table 3, i.e. VA for VA, AA, and QA, and LA for LA.
2. Table 12-3 LA state transitions straight to the VO state on receipt of a LeaveAll instead of going to LO first. This skips out the prompting effect of sending an "Empty" in the LO state, which is required for successful operation of an Applicant misconfigured for point-to-point operation when attached to shared media.

- Begin! member = active = change = false; sent = 0;
- New! if (!(member && change)) { member = active = change = true; sent = 0; ntt();};
- Join! if (!member) { member = true; change = false; if (sent < 2) ntt();};
- Lv! if (member) { member = false; if (active) ntt();};
- tx! if (member && change) { sN(); sent++; if (sent = = 2) change = false; else ntt();}
  else if (member) { if (sent < 2) { sJ()); active = true; sent ++; if (sent < 2) ntt();}
  else if (encode && active) sJ(); else if (encode) s();};}
  else if (active) { sL(); active = false; change = true; sent = 0; ntt();}
  else if (change) { if (p2p) sL() else sE(); change = false;}
  else { if (encode) s()};
- txLM! if (member && change) { sN(); sent++; if (sent = = 2) change = false; else ntt();}
  txLA! else if (member) { sJ()); active = true; if (sent < 2) sent ++; if (sent < 2) ntt();}
  else if (active) { if (encode) s(); active = false; change = true; sent = 0; ntt();}
  else if (change) { if (encode) s(); change = false;}
  else { if (encode) s(); if (txLA!) { change = true; ntt();};};
- txF! if ((member && (sent < 2)) || (active && !member) || change) ntt();
- txLAF! if (member) { sent = 0; ntt();}
  txLMF! else if (active) { active = false; change = true; ntt();}
  else { change = false;};
- rNew, rJoin, rNull, rLv! && p2p, rLM! && p2p, rLI! && p2p // do nothing
- rNewIn! if (member && change) { /*do nothing */ }
  rJoinIn! else if (active && !member) { /*do nothing */ }
  else { if (sent < 2)) sent++;};
- rNewMt! sent = 0;
  rJoinMt! if (member) { ntt();};
  rMt! else if (active) { if (rNewMt! || rJoinMt!) active = false;}
  rLv! && !p2p else if (change) { change = false;}
  rLM! && !p2p else if (rLv! || rLM!) { if (!p2p) { change = true; ntt();};};
- rLA! sent = 0;
  if (member && change) { /* do nothing more */
  else if (member) { active = false; change = true; ntt();}
  else { change = true;};

Event/State	VC	AC	VA	AA	QA	VP	AP	QP	LA	LO	VO	AO	QO
Begin!	VO	VO	VO	VO	VO	VO	VO	VO	VO	VO	VO	VO	VO
New!	-x-	-x-	VC	VC	VC	VC	VC	VC	VC	VC	VC	VC	VC
Join!	-x-	-x-	-x-	-x-	-x-	-x-	-x-	-x-	VA	VP	VP	AP	QP
Lv!	LA	LA	LA	LA	LA	VO	AO	QO	-x-	-x-	-x-	-x-	-x-
tx! && p2p	sN AC	sN QA	sJ AA	sJ QA	[sJ] QA	sJ AA	sJ QA	[s] QA	sL LO	sL VO	[s] -x-	[s] -x-	[s] -x-
tx! && !p2p	sN AC	sN QA	sJ AA	sJ QA	[sJ] QA	sJ AA	sJ QA	[s] QA	sL LO	sE VO	[s] -x-	[s] -x-	[s] -x-
txLM!	sN AC	sN QA	sJ AA	sJ QA	sJ QA	sJ AA	sJ QA	[s] QA	sL LO	[sL] VO	[s] -x-	[s] -x-	[s] -x-
txLA!	sN AC	sN QA	sJ AA	sJ QA	sJ QA	sJ AA	sJ QA	[s] QA	sL LO	[s] VO	[s] LO	[s] LO	[s] LO
txF!	VC	AC	VA	AA	-x-	VP	AP	-x-	LA	LO	-x-	-x-	-x-
txLAF!	VC	VC	VA	VA	VA	VP	VP	VP	LO	VO	-x-	-x-	-x-
txLMF!	VC	VC	VA	VA	VA	VP	VP	VP	LA	LO	-x-	-x-	-x-
rNewIn! rJoinIn!	-x-	-x-	AA	QA	-x-	AP	QP	-x-	-x-	AO	AO	QO	-x-
rNewMt! rJoinMt!	-x-	VC	-x-	VA	VA	-x-	VP	VP	VO	VO	VO	VO	VO
rMt!	-x-	VC	-x-	VA	VA	-x-	VP	VP	LA	VO	VO	VO	VO
rLv! rLM! && !p2p	-x-	VC	-x-	VA	VA	-x-	VP	VP	LA	VO	LO	LO	LO
rLA!	-x-	VC	VP	VP	VP	-x-	VP	VP	LO	VO	LO	LO	LO

Note – ntt on entry/re-entry to VC, AC, VA, AA, LA, VP, AP, and LO states.
Note – rNew, rJoin, rNull, rLv! && p2p, rLM! && p2p not shown, no action taken.

Table 3 – Bi-media Applicant state machine

A point-to-point subset

Table 4 is illustrates the behavior of a MAD Applicant configured for point-to-point operation and connected to another similarly configured. Equivalent code follows.

Event/State	VC	AC	VA	AA	QA	VP	AP	QP	LA	LO	VO
Begin!	VO	VO	VO	VO	VO	VO	VO	VO	VO	VO	VO
New!	-x-	-x-	VC	VC	VC	VC	VC	VC	VC	VC	VC
Join!	-x-	-x-	-x-	-x-	-x-	-x-	-x-	-x-	VA	VP	VP
Lv!	LA	LA	LA	LA	LA	VO	AO	QO	-x-	-x-	-x-
tx!	sN AC	sN QA	sJ AA	sJ QA	[sJ] QA	sJ AA	sJ QA	[s] QA	sL LO	sL VO	[s] -x-
txLM!	sN AC	sN QA	sJ AA	sJ QA	sJ QA	sJ AA	sJ QA	[s] QA	sL LO	[s] VO	[s] -x-
txF!	VC	AC	VA	AA	-x-	VP	AP	-x-	LA	LO	-x-
txLMF!	VC	VC	VA	VA	VA	VP	VP	VP	LO	VO	-x-

Table 4 – Point-to-point state machine

- Begin! member = active = change = false; sent = 0;
- New! if (!(member && change)) { member = active = change = true; sent = 0; ntt();};
- Join! if (!member) { member = true; change = false; if (sent < 2) ntt();};
- Lv! if (member) { member = false; if (active) ntt();};
- tx! if (member && change) { sN(); sent++; if (sent = = 2) change = false; else ntt();}
  else if (member) { if (sent < 2) { sJ()); active = true; sent ++; if (sent < 2) ntt();}
  else if (encode && active) sJ(); else if (encode) s();};}
  else if (active) { sL(); active = false; change = true; sent = 0; ntt();}
  else if (change) { sL(); change = false;}
  else { if (encode) s()};
- txLM! if (member && change) { sN(); sent++; if (sent = = 2) change = false; else ntt();}
  else if (member) { sJ()); active = true; if (sent < 2) sent ++; if (sent < 2) ntt();}
  else if (active) { if (encode) s(); active = false; change = true; sent = 0; ntt();}
  else if (change) { if (encode) s(); change = false;}
  else { if (encode) s();};
- txF! if ((member && (sent < 2)) || (active && !member) || change) ntt();
- txLMF! if (member) { sent = 0; ntt();}
  else if (active) { active = false; change = true; ntt();}
  else { change = false;};
- rNew, rJoin, rNull, rLv! && p2p, rLM! && p2p // do nothing

The MAD Registrar

When operPointToPointMAC is false, the Registrar operates as specified in 802.1D-2004, with the addition of new registration signaling. When attached to a point-to-point LAN, receipt of an explicit Leave message result in a Leave indication immediately. See Table 5 below.

LeaveAll and Leave Mine machines

Each MAD component (one per Port) has a single Leave All state machine, based on the specification in 802.1D-2004 Table 12-5 and extended to include sending LeaveMine messages. See Table 6 below.

Event/State	IN	LV	MT
Begin!	MT	MT	MT
rNew! rNewIn! rNewMt!	New IN	New Stop leavetimer IN	New IN
rJoin! rJoinIn! rJoinMt!	IN	Stop leavetimer IN	Join IN
rLv! && !p2p	Start leavetimer LV	-x-	-x-
rLv! && p2p	Lv MT	Lv MT	-x-
rLM! rLA! txLA!	Start leavetimer LV	-x-	MT
leavetimer!	-x-	Lv MT	MT

Table 5 – MAD Registrar state machine

Note – rMt!, rNull, txLM! not shown, no action taken.

Event/State	Active	Passive
Begin!	Start leaveall timer Passive	Start leaveall timer Passive
tx! && p2p	sLM Passive	-x-
tx! && !p2p	sLA Passive	-x-
rLA!	Start leaveall timer Passive	Start leaveall timer Passive
rLM!	-x-	-x-
leaveall timer!	Start leaveall timer Active	Start leaveall timer Active

Note – Starting the leaveall timer implies restarting it, if already running. Transmit and reception events that do not affect this machine are not specified.

Table 6 – Leave All state machine