SRP Scaling: Refresh Timer

Andre Fredette

IEEE 802 Plenary
July 16, 2013

Outline

- MRP LeaveAll Processing Overview
- Review the MRP Timer issue for MSRP.
  - Previously discussed in:
    - at-cgunther-mrp-timers-0310-v02.pdf (March 2010)
    - avb-dolsen-srp-limitations-v2.pdf (November 2011)
- Propose a modification to MRP that solves the problem.
  - Introduced in:
    - new-tsn-cgunther-SRP-next-gen-0313-v01.pdf (March 2013)

The LeaveAll

- The LeaveAll process performs a refresh and garbage collection function for MRP applications.
  - If a Leave was missed, the attribute will get aged out.
  - If a Join was missed, the attribute will be added.
    - Note: Not a likely case since MRP ensures that two join messages are sent for each attribute or peer has reported the attribute as registered.

- Each time a leaveall (LA) event is sent or received,
  - All attributes (Talkers, Listeners, Domain) must be “refreshed” before the Leavetimer expires,
  - Otherwise, they age out.

How it Works (Registrar State Machine)

- If (S==IN && r/txLA)
  - Start leavetimer
  - -> LV
- If (if S==LV & rJoin*!)
  - Stop leavetimer
  - -> IN
- If (S==LV && leavetimer!)
  - Lv
  - -> MT

Table 10-4—Registrar state table

		STATE		
		IN	**LV**	**MT**
EVENT	**Begin!**	MT	MT	MT
	rNew!	New IN	New Stop leavetimer IN	New IN
	rJoinIn! || rJoinMt!	IN	Stop leavetimer IN	Join IN
	rLv! || rLA! || txLA! || Re-declare!	Start leavetimer LV	-x-	-x-
	Flush!	MT	Lv MT	MT
	leavetimer!	-x-	Lv MT	MT

How it Works (Applicant State Machine)

- When a LA event is sent or received:
  - Applicant state machine causes the participant to send a Join event for each attribute being declaring.

Table 10-3—Applicant state table

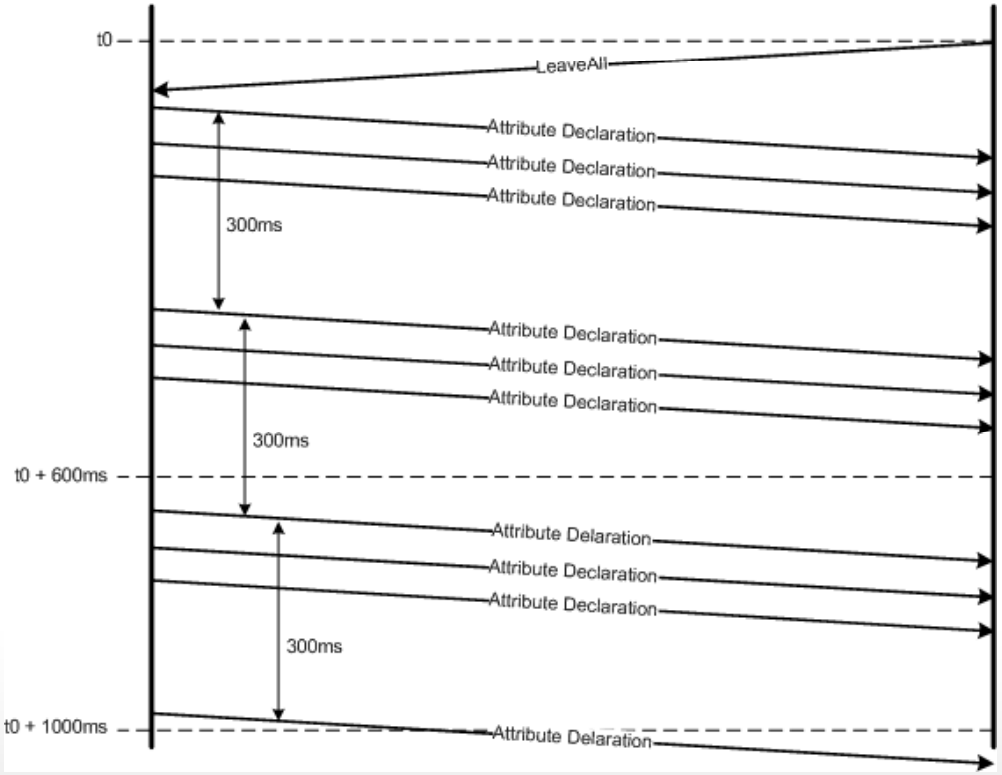
		STATE											
		VO[11]	VP[6]	VN[6]	AN[6]	AA[6]	QA	LA[6]	AO[3,11]	QO[3,11]	AP[3,6]	QP[3]	LO[6]
EVENT	Begin!	—	VO	VO	VO	VO	VO	VO	VO	VO	VO	VO	VO
	New!	VN	VN	—	—	VN	VN	VN	VN	VN	VN	VN	VN
	Join!	VP	—	—	—	—	—	AA	AP	QP	—	—	VP
	Lv!	—	VO	LA	LA	LA	LA	—	—	—	AO	QO	—
	rNew!	—	—	—	—	—	—	—	—	—	—	—	—
	rJoinIn!	AO[4]	AP[4]	—	—	QA	—	—	QO	—	QP	—	—
	rIn!	—	—	—	—	QA[5]	—	—	—	—	—	—	—
	rJoinMt! || rMt!	—	—	—	—	—	AA	—	—	AO	—	AP	VO
	rLv! || rLA! || Re-declare!	LO[1]	—	—	VN	VP[9]	VP[9]	—[10]	LO[1]	LO[1]	VP	VP	—
	periodic!	—	—	—	—	—	AA	—	—	—	—	AP	—
	tx![7]	[s] —	sJ AA	sN AN	sN QA[8]	sJ QA	[sJ] —	sL VO	[s] —	[s] —	sJ QA	[s] —	s VO
	txLA![2]	[s] LO	s AA	sN AN	sN QA	sJ QA	sJ —	[s] LO	[s] LO	[s] LO	sJ QA	sJ QA	[s] —
	txLAF![2]	LO	VP	VN	VN	VP	VP	LO	LO	LO	VP	VP	—

The Timer Issue

- The MRP Protocol limits the rate at which PDUs may be transmitted.
  - 3 PDUs per 1.5 * Jointime
- Number of attributes that can fit in a PDU is limited.
- Therefore, this PDU rate-limiting limits the number of attributes that can be refreshed before timers expire and attributes age out.

LeaveTime Operation (point to point)

- Point to Point operation allows 3 packets every 1.5 JoinTime
- JoinTime is defined as 200 milliseconds
- No way to discover if the entire network is point to point
- See IEEE Std. 802.1Q-2011 Table 10-7



Worst Case Max Streams

- Assuming worst case of 1 attribute per vector

- Max Streams Calculations
  - Vector_Size = Vector_Hdr(2) + First_Value_Len
    + Ceiling(Values_In_Vector/3)
    {+ Ceiling(Values_In_Vector/4)} /* If Listner */
  - Vectors_Per_PDU =
    floor((Max_PDU_Size - Message_Overhead(9))/Vector_Size)
  - PDUs_Per_Leavetime = Leavetime / (JoinTime/2)
  - Max_Streams = PDUs_Per_Leavetime * Vectors_Per_PDU

The Limits

- MRP allows for efficient packing of sequential attributes
- But, attributes are not always efficiently packable.
- Examples of max attributes with default timer values (JoinTime = 200ms, LeaveTime = 600ms).

Attributes per Vector	Talker Advertise	Talker Fail	Listener
1	318	240	810
8	2352	1824	5472
Max	26,352	26,190	15,228

- Other Applications
  - Not an issue for MVRP because all 4K attributes always fit in one PDU.
  - Similar issue for MMRP.

Near “Solutions”

- Increase Leavetime
  - This is the only solution available in the current version of SRP.
  - Problem: Increases the time to explicitly tear down an existing stream.
- Don’t use the Leavetimer (just refresh before the next LA is sent or recieved:
  - Problems:
    - if both peers send a LA at approximately the same time, they will both
      - Start Leavetimer when txLA!, and then
      - See the rxLA! as the “next LA”.
    - If one device misses the LA from it’s peer, it will send a LA of it’s own

Root Cause of Problem

- This is a problem because the same Leavetimer is used for two different purposes:
  1. As a refresh timer after the LeaveAll event is sent or received, and
  2. To support multiple devices in a shared medium
     - When a device on a shared medium sends a Leave event for an attribute,
     - And another device on that same shared medium is still declaring the attribute,
     - The other device needs to send a Join event to prevent the attribute from aging out,
     - and the Leavetimer allows time for this to happen.
  - Note: The Leavetimer is not needed for purpose #2 on a P2P link – it only serves to lengthen the stream tear-down time.
- It worked for MVRP because all possible attributes fit in one PDU.
- Would be a problem for MMRP.
- Conclusion: We need a different timer for each of the above purposes.

Proposed Solution

- Add a new timer called the RefreshTimer to be used in the LeaveAll process.
  - State: IN_R – In Refresh
  - Event: refreshTimer!
  - Actions: Start/Stop RefreshTimer
  - Send event rules: Same as when in the LV state
- Update the Registrar State Machine to handle the RefreshTimer as shown on the following slide.
- No changes are needed to the Applicant State Machine.
- The recommended default value for Refreshtime is 9 seconds (slightly less than the default Leavetime of 10 seconds).

Updated Registrar State Machine

		STATE			
		IN	IN_R	LV	MT
EVENT	Begin!	MT	Stop refreshtimer MT	Stop leavetimer MT	MT
	rNew!	New IN	New Stop refreshtimer IN	New Stop leavetimer IN	New IN
	rJoinIn! || rJoinMt!	IN	Stop refreshtimer IN	Stop leavetimer IN	Join IN
	rLv!	Start leavetimer LV	Stop refreshtimer Start leavetimer LV	-x-	-x-
	rLA! || txLA! || Re-declare!	Start refreshtimer IN_R	-x-	-x-	-x-
	Flush!	Lv MT	Stop refreshtimer Lv MT	Stop leavetimer Lv MT	MT
	leavetimer!	-x-	-x-	Lv MT	-x-
	refreshTimer!	-x-	Start leavetimer LV	-x-	-x-

Discussion

- The Refresh Timer proposal is interoperable with existing implementations
  - Protocol behaves as if it is in the LV state when in the IN_R state.
  - Loos like a longer LeaveTimer.
- Needed even when IS-IS is used because MRP-based MSRP will likely be used when talking to end points.
- Should still consider other MSRP optimizations such as summary messages and timer negotiation.
- Recommend that the RefreshTimer and associated Registrar State Machine changes be included with the next version of SRP.