SRP – an alternative approach
Tony Jeffree

Summary

The existing SRP draft has been a useful starting point for discussions on the reservation process and how it can be made to work. However, I believe that the current approach can be simplified – in terms not only of how SRP is documented, but also how it is implemented in L2 devices, and how the separation between L2 and higher layer functions is successfully maintained. I believe the key to this simplification is to recognize what is, and what is not, the business of the L2 devices that participate in the process, and to leverage existing technology in the form of the recently completed MMRP.

1. Introduction

My starting assumption is that as far as the SRP standard is concerned, the problem that we are trying to solve is how we associate reservation information (in the form of some kind of bandwith/traffic class specification) with stream information (in the form of MAC addresses) contained in the Filtering Databases of Bridges, in order to allow the Bridge to make meaningful forwarding decisions based on that data[1].

I don't believe that, at least in this project, we are in the business of figuring out how the end stations and stream servers might negotiate between themselves as to what streams exist, what high-level tags or names they might use to communicate about them, and how that high level data might get translated into stream identifiers that are going to be meaningful to Bridges, i.e., MAC addresses. I also don't believe that we are in the business of providing some kind of transport protocol at L2 for communicating higher level data about streams and reservations.

That is not to say that the other set of problems isn't interesting, and likely form a necessary part of the overall solution to AV; just that it fundamentally isn't the job of SRP to solve them. That being the case, there may well be one or more further projects needed in order to complete the jigsaw; alternatively, it might well be our conclusion that getting into what amounts to an Application layer protocol development isn't 802.1's job. Either way, that particular discussion hasn't (as far as I can tell) really taken place yet.

In other words, at least part of what I am trying to achieve with this proposal is to take what has proved to be an extremely successful approach over the history of 802.1 – that is, to focus on a well defined, limited scope problem that can be solved simply, and avoiding the potential pitfalls of widening the scope to the point where the project can't complete.

2. What are streams anyway?

The idea of data streams isn't new to the 802.1 Bridge standards; providing support for data (video, voice…etc.) streamed to multicast destination addresses dates back to the mid 90's, and was the driving force behind the development of GARP and GMRP (originally standardised in 802.1D:1998, now re-incarnated as MRP and MMRP in 802.1ak). The only real difference, in terms of the starting assumptions stated in 1), between what we are looking at now and what we were looking at then, is the decision to associate bandwidth reservation information with a stream, and apply traffic shaping to it in the forwarding process accordingly.

Hence, the first and most obvious simplification that can be made in SRP is simply to leverage the support for streams that already exists in our standards – namely, MMRP. It is there, and it can already be used, without modification, to create a subtree of the Spanning Tree that provides a forwarding

[1] The job of specifying those forwarding decisions is part of P802.1Qav; obviously there will be some level of interaction needed between these two projects to get it right.

path between a stream data source and any registered end stations; as can be seen later on, the detail of when a stream becomes "active", and making sure that this does not happen before the forwarding path is aware that this is a stream (and therefore, that it has to have a reservation associated with it) is something that can be handled by the data source and sink, coupled with correct specification of how the Bridges handle the forwarding path.

The first consequence of going down this route is that SRP becomes a single protocol, one that is concerned only with making and maintaining reservation decisions, rather than two protocols. There will still be some need for interaction between MMRP and the SRP reservation mechanism; however, this can successfully be achieved without affecting the specification of MMRP in any way.

The second consequence is that, as streams are (as far as SRP is concerned) identified only by a MAC address, we lose one of the optimizations of the original proposed mechanism – that stream registrations propagate only in the direction of the stream data source. In the target networks (AV networks in a home or studio environment) I don't believe that this particular optimization (or its absence) is an issue – the whole point of the 802.1ak project was to improve the scaling properties of MRP-based applications, so that their use made sense in small networks, and so they could successfully scale for use in large networks. I will grant the fact that all Bridges in an AV network (or at least, all that exist on the path between stream data sources and sinks) will need to implement the mechanism; however, given that they have to support it, the extra overhead of processing registrations on Ports that don't reach a data source is insignificant, as is the bandwidth consumed by the registration PDUs. Given that the MRP PDU structure can represent registration state information for 4K streams in a single PDU (assuming the stream addresses are consecutive)[2], the real protocol overhead, and consequently, the real optimisation opportunity, is small[3].

To put some real numbers on the protocol overhead, the way MRP is currently specified, "transmission opportunities" happen at least every 20 centiseconds, and can happen "on demand" at whatever processing rate the system can achieve, subject to a limit of no more than 3 transmissions in any 30 centisecond period.

So, the maximum sustained transmission rate for a MRP application is 10 maximum sized Ethernet frames/second[4,5], which, to put it in perspective, is about 1.5% of the available bandwidth of a 10Mbps link, or 0.15% of the bandwidth of a 100 Mbps link. Under more normal circumstances, where the rate of registration state change is low, the protocol becomes silent once all state change has been propagated (which typically requires the exchange of 2 PDUs on each link), and only "wakes up" again when the periodic "Leave All" garbage collection takes place, which happens on a 10 second timer. Again, returning the protocol to its silent state requires at most 2 PDUs on each link.

3. Registering and de-registering for a stream

A listener can register for a stream, using MMRP, at any time. There are two cases to consider:

[2] In contrast, the "MMRP-like" registration mechanism in the current draft, because it carries additional higher layer tag information, loses much of the advantage of the new "vector" packed data format used by MMRP. Each registration carries a payload of 20 octets, and because the additional information over and above the stream MAC address changes per stream, only a single stream can be packed into a "VectorAttribute". Consequently, the best that can be managed in the proposed registration protocol is packing 65 stream registrations in a max sized Ethernet frame, compared with 4K (contiguous) streams with MMRP. This sounds to me like a major scaling issue if there are applications that require more than ~60 streams, and would be a big problem if the requirement were of the order of 1000 streams.

[3] Certainly too small to justify inventing a new MRP application when one already exists that can do the job. Coupled with the scaling problem identified earlier, the justification looks even weaker.

[4] Where a full frame would potentially carry registration state for 4K streams.

[5] This also means a worst case propagation delay of registration information of 1/10th second per hop; however, propagation delays should normally be significantly smaller, comparable to the processing delay in the Bridges.

- The listener is the first in the network to register for the stream.
- The listener is not the first in the network to register for the stream.

In the first case, the talker for that stream can make use of the MMRP state information to detect the fact that there is now a listener registered for that stream where none existed before. The concept of "source pruning" is already described in MMRP; the presence/absence of a registered recipient is used to forward/filter frames destined for the registered address – the data source effectively behaving like a single port Bridge. That same mechanism can be used to trigger the reservation protocol to establish reservation information for that stream along the path from the talker to the listener. Given that we are talking about streams that are to have bandwidth reservations associated with them along the path to the listener, the talker can (and indeed, must) ensure that the reservation data required by the transmission path is transmitted before it starts to transmit stream data. So, to a first order, the talker needs to transmit the reservation information, then wait long enough before starting data transmission to ensure that the stream data cannot overtake the reservation data[6].

The degree to which the listener cares about the success or failure of the reservation process seems to be more of an application choice. If the application concerned is tolerant of degraded service, then it would seem to be a perfectly reasonable choice for the listener to completely ignore the reservation protocol. At the other extreme, an application that is sensitive to the provided QoS might listen in to reservation messages, and determine whether or not to de-register for the stream (or take some other action) depending on success or failure of the reservation. There is probably a middle ground here too, where the listener might accept degraded service on a stream, but might listen in to the reservation protocol in order to be able to report the service level to its user. In any of these situations, there is a requirement that the listener behaves correctly in the sense that when it is done listening to a stream, it requests de-registration of that stream.

From these observations, it would seem that the reservation protocol can be a one way (source to sink) transmission of data; the source can determine whether anyone is still interested in the stream, at the available QoS, simply by monitoring the registration state of the stream.

In the second case, where a new listener joins an already established stream, all that is required over and above the first case is for one or more of the intervening Bridge(s) to recognize that the stream has become registered on a new Port, and to propagate the reservation information that it already knows for that stream in that direction.

The Bridges have to be able to recognize that data transmitted on a stream address, at one of the two chosen stream priorities, can only be propagated along the registered data path if there is a reservation associated with the stream, and if the data doesn't exceed the characteristics of that reservation[7,8].

De-registration is simply the reverse of the above; as registered listeners go away, Bridges will stop propagating reservations for that stream on Ports that no longer support listeners, and eventually, when the last one goes away, the talker will stop originating the reservation information for the stream and will stop transmitting the stream data.

4. Stream reservation

As observed in 3), it looks as if this is a simple "declarative" protocol, where reservation information originates from, and is refreshed by, the stream data source, and is modified and propagated by Bridges only along the path(s) towards any registered listeners.

The protocol needs to convey:

[6] It seems that there is a discussion that needs to take place here about transmission priorities for transmitting the reservation data. Arguably, the reservation data should be considered to be "network control" (max priority) and therefore, there is no possibility of the stream data catching it up as long as it is transmitted first.

[7] I've assumed that the requirement for traffic shaping is per stream rather than per outbound Port; however, whatever the chosen enforcement mechanism, the same principle applies.

[8] This requirement is the same for both of the proposed mechanisms as far as I can tell.

- The stream ID (a MAC address);
- Reservation information, as required for establishing the parameters needed for the operation of Qav[9];
- Not a lot else. For example, given the assumption stated in 1) that some higher layer negotiation mechanism exists that allows the establishment of the right L2 stream identifiers to be used by an application, there probably isn't even a need to tie the talker address to the reservation information; the stream ID is sufficient, as the higher layer mechanisms can ensure there would be no more than one talker using any one stream ID[10].

The result is, I believe, certainly no more complex, and maybe a little simpler than the mechanism in the current draft. In particular, it doesn't carry higher layer information in the protocol that is of no concern to the L2 operation, and therefore removes any temptation for this protocol to become some kind of unspecified transport mechanism for higher layer information that it doesn't itself understand[11,12].

5. The filtering database

The purpose of the reservation protocol is to allow us to associate reservation information with a given stream; there will therefore be a need to define reservation data entries of some form as part of the filtering database, in order for the forwarding process to perform whatever filtering, metering, and de-queuing operations we may decide upon in P802.1Qav.

I've used the words "…as part of the filtering database" above deliberately. The filtering database already contains a number of different types of entry – static filtering entries, dynamic filtering entries,…etc. etc., all of which interact in ways that are clearly specified in the 802.1D and 802.1Q standard. However, it is nonetheless conceptually a single database, and adding a new type of entry doesn't change that. I.e., you might just as easily view the additional reservation information as an extension to the definition of the Group Registration Entry. If there is reservation information there, then the forwarding process uses it; if there isn't, then it behaves exactly like an existing D or Q Bridge with respect to its forwarding behaviour[13].

The reservation information that I believe we must store per stream consists of the following - call it a "Stream Reservation Entry" in the Filtering Database:

- The stream ID (MAC address);
- The VID of the VLAN in which the reservation information was registered[14];

[9] This should be equivalent to the reservation information described in the current draft. See "Reservation protocol" section later.

[10] The assumption here is that there is only ever a single talker associated with a stream. If multipoint-to-point or multipoint-to-multipoint is a requirement, then this is achieved by means of a stream per talker.

[11] Putting this more strongly: The high level tag information included in the current draft isn't needed by this L2 protocol, its use is not specified at all in the draft, and therefore, I believe it has no business being included in the specification.

[12] If the option mentioned in the current draft, of establishing the initial registration just using the high level tag information, were to be followed to its logical conclusion, then I believe there would be a considerable increase in the complexity of SRP, as at some point, the stream MAC address would have to be associated with the stream as well. I suspect that, as this would effectively change the attribute registered in SRP-reg, then the stream would actually have to be de-registered and then re-registered using the new (complete) ID information. The current draft is silent on the mechanics of this at present.

[13] I would strongly dispute the assertion in http://www.ieee802.org/1/files/public/docs2007/at-feng-SRP-MMRP-070205.pdf that the simplified approach results in a more complex forwarding engine. I believe that as far as Qav is concerned, and it is Qav where the forwarding decisions are going to be defined, this proposal and the one in the current draft should be exactly equivalent.

[14] What we are defining here is an extension to 802.1Q VLAN Bridges. The forwarding/filtering information for a given stream, and the stream data itself, will therefore be associated with a VLAN, either based on a VID contained in the Tag Header, or the Port VID (the PVID defines the default VLAN for untagged/Priority tagged frames received on a Port). Now, we could clearly choose to build simple devices that support a limited number of VLANs (could be just 1 VLAN and still be conformant), but we DO have to get to grips with the meaning of registrations/reservations that apply to a VLAN, and questions such as whether it is legitimate to make reservations for the same stream in 2 or more different VLANs, or whether reservation data applies to all VLANs (transmitted untagged).

- The resource requirement information received from the Talker[15];
- The reservation status information received from the Talker[16];
- Possibly, the inbound Port, but not clear that this is necessary.

The operation of the reservation mechanism will determine whether/how much of the reservation requirement can be met for each Port, given the set of reservations that already exist for that Port and how much bandwidth the Port has available for reservation[17]. That information has to be maintained (or at the very least, be derivable) on a per-stream basis, in order for the reservation protocol to be able to propagate the right information to the next downstream node.

Using MMRP, registering a stream results in the creation of a MAC address registration entry in the FDB consisting of:

- The MAC address specification;
- The VID on which the registration occurred; and
- A Port map, specifying forwarding or filtering for the MAC address on each outbound Port.

The obvious extension to add the actual reservation data would be to extend that last Port Map element, so that it is:

- A Port map, specifying, for each outbound Port:
    - forwarding or filtering for the MAC address specification;
    - The reservation status and resource allocation for the stream.

The natural consequence of the above would be that traffic shaping would be on a "per stream and outbound Port" basis, rather than just on a "per outbound Port" basis[18]. However, one could also envisage other ways that this "per stream" information could be appropriately glommed together to meet other requirements.

6. Reservation protocol

The reservation information carried in the protocol contains elements (hop counts,…etc) that are updated Bridge by Bridge along the route; consequently, we do not want SRP frames to be forwarded by any Bridge through the normal forwarding path – much like Spanning Tree and MRP, there is a protocol entity in each Bridge that receives incoming SRP frames from upstream (towards the Talker) Ports, processes them, updates the FDB accordingly, and transmits them out of appropriate downstream Ports[19]. Consequently, the MAC address used by this protocol will have to be one of the set of "reserved addresses" that appear in both Table 8-1 and Table 8-2 (so the PDUs don't propagate through Provider Bridges either).

The protocol needs to convey:

- The stream ID (MAC address);
- The resource requirement information[20];
- The reservation status information.

The Talker initiates transmission of reservation data for a stream when it knows that one or more listeners exist for that stream (MMRP has registered the address, and therefore, the Talker's FDB indicates that address to be "Forwarding" on its outbound Port[21]. If we go the route of a simple "declarative" protocol, then the Talker would refresh the reservation on a regular basis.[22] The talker going away would therefore allow the reservation to time out along the downstream path, and terminate the stream.

The SRP entity in a Bridge receives an SRPDU and uses the reservation information to create/update the relevant Stream Reservation Entry in its FDB, and to update the per-Port reservation information contained in the corresponding MAC address

[15] Probably very similar to what is described in the existing draft.

[16] Probably very similar to what is described in the existing draft.

[17] I seem to recall 75% of the total port throughput being mentioned as the high bar here, but is this the right number, and is this something that the user might expect to be able to manage up or down?

[18] Which of these models is what we want is one of the open questions at present.

[19] I've assumed that this is basically a declarative protocol that doesn't use any kind of backward-propagated "ack" or status PDU.

[20] This needs to include both what the Talker specified and what has been granted at this point in the data path.

[21] The description assumes that the Talker behaves like a single-port "Bridge on a stick".

[22] What frequency?

registration entry (if it exists)[23]. The entity then transmits SRPDUs on any outbound Port that is Forwarding for the stream address concerned. Hence, propagation of the SRPDUs is confined to the path(s) between the Talker and the listener(s). Any addition to the topology for a stream (a registration for a stream appearing on a Port where there wasn't one before) causes the entity to update the reservation info for that Port and to (immediately) propagate the reservation out of that Port. Similarly, if a registration goes away on a Port, the reservation information for that Port is updated, and SRPDUs no longer flow out of that Port.

7. SRP state machines

There seem to be three separate and distinct types of SRP protocol entity involved in the operation of the mechanisms described so far:

- The SRP Listener protocol entity. This is responsible for initiating and terminating stream registrations (by means of MMRP) on behalf of the application in the Listener that is making use of the stream, and reporting the result of the associated reservation (success/failure/timeout/termination etc.) back to the application;
- The SRP Bridge protocol entity. This is responsible for receiving SRPDUs from upstream ports, processing the reservation information, updating FDB entries, and propagating SRPDUs on downstream ports. This protocol entity has no need to communicate with any higher layer entities at all;
- The SRP Listener protocol entity. This is responsible for monitoring registrations and de-registrations as they occur, reporting them to the application in the Talker that is responsible for sourcing the stream, and initiating/updating/terminating reservations using the information provided by the application.

These three protocol entities will require state machine definitions that describe the detailed operation of the functions outlined above. Fortunately, I believe that these state machines are likely to be fairly simple (significantly less complex than MRP, RSTP, …etc.)

The primitives used for communication between the protocol entities and the applications in the Talker and Listener are described in the next section.

8. SRP primitives

This is a first stab at the primitives that will be needed to allow communication between the Talker and Listener applications and their corresponding SRP protocol entities. Request primitives pass from the application to the SRP entity; indication primitives pass from the SRP entity to the application.

Listener primitives:

Initiate_Registration.request (StreamID)

- Causes the SRP entity to register the stream, by means of MMRP, and start a "reservation failed" timeout for the stream.

Terminate_Registration.request (StreamID)

- Causes the SRP entity to de-register the stream, by means of MMRP, and clear the "reservation failed" timeout for the stream.

Reservation_Data.indication (StreamID, Resource_Requirement, Reservation_Status)

- Generated on receipt of a reservation PDU, and signals to the application the information that the reservation PDU contained. Receipt of the PDU also causes the "reservation failed" timeout to be re-started.

Reservation_Terminated.indication (StreamID, Reservation_Status)

- Generated on failure of a reservation; i.e., when the "reservation failed" timeout expires, or when the SRP entity has de-registered the stream (or if there are any other "failed" protocol conditions).

Talker primitives:

Registration_Received.indication (StreamID)

[23] Current assumption seems to be that the maximum reservable bandwidth on a Port is 75% of the Port's total capacity.

- Generated when the SRP entity detects that a stream has been newly registered in MMRP.

Registration_Terminated.indication (StreamID)

- Generated when the SRP entity detects that a stream has been deregistered in MMRP.

Initiate/Update_Reservation.request (StreamID, Resource_Requirement)

- Causes the SRP entity to start sending reservation PDUs for the stream, carrying the specified resource requirement data, on a regular basis.

Terminate_Reservation.request (StreamID)

- Causes the SRP entity to stop sending reservation PDUs for the stream.

9. Forwarding path behaviour

This will be specified in P802.1Qav; however, for completeness I have added some brief notes here on what will probably be needed.

SRP, as described above, establishes the forwarding path for the stream, and in each Bridge, associates with the stream:

- The resource requirement communicated from upstream; and
- The resources allocated to the stream per outbound Port (which may be less than the communicated resource requirement if the Port is handling multiple streams and has reservations that exceed the reservable bandwidth[24]).

In addition to the specification of the resources allocated to the stream per outbound Port, it will be necessary to record dynamically how much of the resource is available to each stream at a given moment in time. So there needs to be some kind of dynamic allocation register that can be interrogated to determine, for a given frame received on an inbound Port, whether it falls within or outside the stream's bandwidth allocation[25], and which is updated to show use of allocation (when a frame is queued for forwarding) and replenishment of allocation (as time progresses), in accordance with whatever algorithm(s) we specify for the traffic shaping.

There would seem to be three parts of the forwarding process that will be potentially affected by the need to perform traffic shaping:

- Q subclause 8.6.3 – Frame filtering. At present, this function determines the set of potential outbound Ports based on destination address, VID, Filtering Database information (Forward/Filter) for that address and VID, and default group filtering behaviour. This could be extended to also take account of the reservation information in the FDB for the stream and outbound Port (Filter if no reservation[26]).
- Q subclause 8.6.5 – Flow classification and metering. From the existing text of 802.1Q (including 802.1ad), this is the appropriate place to insert admission policing functions to the stream data; i.e., checking that the allocated bandwidth constraints for a stream are not exceeded by a given stream, and taking appropriate action (discarding frames, dropping priority…etc.) to enforce this[27].
- Q subclause 8.6.7 – Queue management and 8.6.8 – Transmission selection. These two subclauses would be the appropriate place to insert de-queueing functions – e.g., to ensure that stream data transmissions are properly paced, and that residual

[24] Max 75% of the Port's available bandwidth.

[25] Again, I have assumed that shaping is done per stream per outbound Port; however, similar principles will apply if the basis of the final mechanism is something else (per outbound Port, per inbound/outbound Port pair, etc.)

[26] An statement that has been made a couple of times in recent meetings is that, in an AV Bridge, all traffic that is transmitted on a stream using traffic classes 4 or 5 (Video and Voice respectively) is assumed to require a (non-zero) bandwidth reservation.

[27] There has been a certain amount of discussion in recent meetings with regard to whether or not such ingress policing is needed given that the Talkers, and any upstream Bridges, perform pacing of stream traffic on transmission. This seems to be partly a "how much do you trust the Talkers to play nice" question, and partly a question of what is required in order to maintain the right transmission characteristics of individual streams.

bandwidth is fairly allocated to the remaining lower priorities.

10. Overall architecture

To summarise the above, the overall structure that needs to be documented in the SRP standard looks like this:

- Talkers and listeners (sources and sinks of stream data) that use some higher layer mechanism (not specified in the SRP standard, but possibly the basis of some future project(s)) to determine what stream identifiers, in the form of MAC addresses, they will use;
- Stream registrations and de-registrations based on the use of MAC addresses registered by the existing MMRP mechanisms;
- A stream reservation protocol, supported by protocol entities in the talker, the Bridges, and (optionally) the listener, that carries only L2 reservation information for a given stream[28]:
    - The Talker uses MMRP registration/de-registration events to trigger the transmission of reservation information and the transmission of the stream on a regular basis, and to cease transmission when no listener exists;
    - The Bridges use the reservation information received from the Talker (or from an upstream Bridge) to update their FDBs with the information needed by P802.1Qav for each stream, and forward the reservation information (modified as appropriate) on any paths where the stream is registered;
    - The Bridges also recognize stream registration/de-registration events, and update/propagate FDB and reservation information accordingly;
    - The listeners take as much notice of the received registration information as is appropriate for the application concerned; consequently, listener support of the reservation protocol is optional. However, what is not optional is the listeners being "hygienic" about de-registering for a stream when they no longer need it.
- Extensions to the existing FDB definition that allow reservation information to be associated with a registered stream, and in a form that allows P802.1Qav to perform its job of traffic shaping for the stream.
- In P802.1Qav, the specification of how the registration and reservation data in the FDB is used to control the traffic shaping behaviour of the forwarding process.

[28] I guess we may well talk about optimizations that would allow the packing of reservation information for multiple streams into a single PDU, but that is a detail.