Analysing Termination and Consistency in the AVB's Stream Reservation Protocol

Daniel Bujosa*, Inés Álvarez*, Drago Čavka†, Julián Proenza*
*Departament de Matemàtiques i Informàtica, Universitat de les Illes Balears, Spain,
{daniel.bujosa, ines.alvarez, julian.proenza}@uib.es
†Faculty of Electrical Engineering, University of Banja Luka, Bosnia and Herzegovina,
drago.cavka@etf.unibl.org

***Abstract*—The Audio Video Bridging Task Group (AVB TG) from the IEEE proposed a series of standards to provide Ethernet with soft real-time guarantees. Later on, the group was renamed to Time-Sensitive Networking and its scope was broadened to provide new services to support critical applications. The Stream Reservation Protocol (SRP) stands out among the projects developed by the groups. Nonetheless, SRP was originally designed for audio/video applications and does not take into account properties that are important for critical systems; such as termination and consistency. In this work we study the termination and consistency of SRP at different levels, using a model we developed of this protocol in UPPAAL . We see that SRP does not provide termination nor consistency, we discuss how this can impact critical applications and we propose solutions for all the issues detected.**

I. INTRODUCTION

In 2005 the IEEE Audio Video Bridging Task Group started working to provide Ethernet with soft real-time capabilities. The interest in the work of the group grew and in 2012 the group was renamed to Time-Sensitive Networking Task Group (TSN TG) and its scope broadened to provide Ethernet with hard and soft real-time communications, flexibility of the traffic and fault tolerance mechanisms for critical applications.

One of the projects developed by the groups is the Stream Reservation Protocol (SRP), which was originally standardised by the AVB TG in [1] and later extended by the TSN TG in [2]. SRP provides Ethernet with support for resource reservation between a transmitter and its receivers, e.g. reserve bandwidth in bridges' ports to forward certain traffic. SRP is a key piece to support many of the projects developed by the TSN TG. This is so because resource reservation prevents frame delays beyond predefined limits and losses due to buffer overflow. Furthermore, SRP allows modifying the traffic in run-time, providing a certain degree of flexibility to the network.

Some of the applications targeted by the TSN TG are critical. The mechanisms developed for these applications, including those that operate at the network level, must exhibit a series of properties to guarantee their correct operation. Nonetheless, AVB's distributed version of SRP was not designed considering these applications and, thus, it was not designed to fulfil these properties. Moreover, even though the revision of SRP included two new architectures, it still supports the distributed one with no modifications.

Due to the great relevance of the work carried out by the TSN TG, the community has done a significant amount of work related to the study, application and improvement of the TSN standards, such as the ones surveyed in [3] and [4]. Many others are focused on the study of SRP's efficiency, such as the work presented in [5]; while other works provide SRP with support for reserving resources in several redundant paths [6]. Nonetheless, to the best of the authors' knowledge, there are no works related to the study of the adequacy of the distributed version of SRP for its use in critical systems.

In this work we carry out a first analysis to know whether the distributed version of SRP can be deployed in critical systems as it is included in TSN's new standard. More precisely, we evaluate whether it provides termination and consistency, properties that are fundamental for critical systems. In order to evaluate these properties, we modelled the protocol (see [7] for a description of this model) using the UPPAAL model checker, thanks to which we were able to detect the issues presented in this work. We analyse the protocol assuming the absence of faults. Transient faults would not be a problem since they can be tolerated using mechanisms such as those shown in [8]; whereas permanent faults have been left for future work. Next we identify scenarios in which termination and consistency for the reservations are not achieved, we discuss the consequences derived from the absence of these properties and we provide a first overview of a series of mechanisms to enforce them.

II. SRP OVERVIEW

As anticipated in Section I, SRP [1] is key to provide real-time guarantees to Ethernet-based communications; as it allows to ensure the availability of resources during the communication. This allows to bound the end-to-end delay of frames and to prevent packet losses due to buffer overflow. Furthermore, SRP allows to modify the traffic in run-time, providing certain degree of flexibility to the network. As we said, currently there are three different architectures proposed in the SRP standard [2]. In this work we focus on the distributed architecture of SRP.

SRP enforces a publisher-subscriber model, where the publisher is called *talker* and the subscribers are called *listeners*. The data communication is carried out through *streams*. A stream is a logical communication channel which conveys traffic with specific characteristics, e.g., a certain period and

frame size. In SRP's distributed architecture, the talker is responsible for triggering the creation of the stream.

To create a stream the talker declares its intention to communicate by transmitting a special message called *talker advertise* (TA), which is transmitted in broadcast mode. Note that there are mechanisms that eliminate loops in order to prevent declarations from circulating the network indefinitely. The TA message conveys information to identify the stream and the resources it needs. This message is processed by the bridges, which check whether there are enough resources in each one of their forwarding ports to create the stream. If there are enough resources in the ports, the bridge forwards the TA message through them; otherwise, the bridge forwards a *talker failed* message (TF) through the ports with no resources. A TF message conveys the same information as the TA message plus the reason for the failure in the reservation and it is also transmitted in broadcast mode. Note that, at this point, bridges register the talker's request, but do not reserve resources.

When a listener receives a talker message, it decides whether it wants to bind to the stream. If the listener does not want to bind, it does not perform any further actions nor informs the talker about it. Conversely, if the listener wants to bind to the stream there are three options, (i) if the listener receives a TA message, it checks whether it has enough resources and, if so, it transmits a *listener ready* message (LR); (ii) if the listener does not have resources the listener forwards a *listener asking failed* message (LAF) and (iii) if a listener receives a TF message it also forwards an LAF message.

Bridges unify the responses from the listeners to transmit them towards the talker. To do so, bridges first process the responses received through each port and then generate a new response. When a bridge receives an LR message through a port, it checks if the port has enough resources. If there are resources, the LR remains unaltered and the port locks the resources; otherwise the LR becomes an LAF. On the other hand, if a bridge receives an LAF message the value remains unaltered and the port does not lock the resources.

Once the bridge has processed the responses, it must unify them. If all ports have an LR, the bridge forwards an LR to the talker; while if all ports have an LAF, the bridge forwards an LAF. Finally, if the bridge has LR in some ports and LAF in some other ports, it forwards a so called *listener ready failed* message (LRF). During the time the talker waits for a response, or if it receives an LAF, the stream is not created. Nevertheless, when the talker receives an LR or an LRF message, it creates the stream and starts transmitting.

Once the stream has been created, the talker can delete it at any time if needed. It can do so using the unadvertise stream mechanism, which consists in transmitting a message that eliminates the stream from all devices. This message is also transmitted in broadcast mode to ensure that all bridges and listeners receive the indication to eliminate the stream.

III. Evaluation of the Termination

As it was introduced in Section I, termination is an important property in critical distributed systems. Thus, TSN should provide it in order to support this kind of systems. In this work we differentiate two levels of termination.

The first level refers to termination for the application. Many critical applications require to know the result of the reservations to make important decisions. Thus, the lack of termination can cause a malfunction of those applications. At this level, SRP should ensure that all reservations are finished (creating the stream or not) within a bounded time.

The second level refers to termination for the infrastructure involved in the reservations. We understand as infrastructure the bridges and nodes of the network. Since these devices make decisions related to the reservations, it is important to provide them with termination, to prevent unforeseen and undesirable effects in future reservations.

Nevertheless, despite the importance of termination, we found some issues in both levels even in the absence of faults. It is important to note that the issues detected are mainly due to the fact that in SRP listeners do not inform the bridges nor the talker when they are not interested in binding to a stream. We next present the problems detected and some of the possible ways to solve them.

A. Termination for the Application

We first discuss termination for the application. We saw that there are scenarios where the talker does not receive any response from the listeners, even in the absence of faults, and thus it waits indefinitely, e.g., if there are no listeners interested in the stream. As said before, many critical applications require to know the result of the reservations to make important decisions. Thus, the lack of termination can cause a malfunction of those applications, such as blocking the decision process or leading to incorrect decisions due to the lack of knowledge.

A possible solution is to introduce a timeout in the talker. If the talker does not receive any listener response before the timeout expires, it should tear the stream down using the unadvertise stream mechanism available in SRP mentioned in Section II. The value assigned to the timeout should be adjusted depending on the topology, to ensure that there is enough time for the furthest listeners to communicate to the talker their intention to bind.

B. Termination for the Infrastructure

We now discuss the second level of termination, which corresponds to the termination for the infrastructure. A bridge that forwards the request of a talker waits for the responses of the listeners indefinitely. Also, bridges register talkers' attributes in all their ports, and they do so for all the talkers willing to transmit. Using the model we saw that there are scenarios where a bridge sends a talker advertise and does not receive any response, because none of the listeners connected to the bridge (directly or indirectly) want to bind to the stream. Furthermore, we saw that this can happen in scenarios where the first level of termination is actually achieved by the protocol. This can cause an unnecessary use of memory in bridges and can later prevent the creation of streams with listeners willing to bind due to a lack of resources.

We propose two different solutions to this problem. The first one consists in introducing a timeout in each bridge and listener. In a bridge, the expiration of the timeout would delete the talker registration; whereas in a listener the expiration would delete the talker registration if it does not want to bind. The value assigned to the timeout in bridges should be long enough to guarantee that listeners' responses have time to reach them before the registration is torn down; whereas in listeners it should be long enough to guarantee that they can announce their will to bind to the stream before the registration is torn down.

The second solution could consist in introducing a timeout only in bridges, and not in listeners. Instead, the declarations in listeners would be removed when receiving a special frame transmitted by bridges. Specifically, the bridge closer to the talker with no reply from the listeners would delete the stream registration from its memory and then transmit the frame to trigger the elimination of the registration in the rest of the network towards the listeners. This would be similar to what talkers do when using the unadvertise stream mechanism mentioned in Section II. As in the previous solution the timeout should be long enough to guarantee that all listeners' responses can reach the bridge before the registration is torn down on it. Furthermore, the first timer that expires should be the one in the bridge closer to the talker. Otherwise, more than one bridge might start the transmission of the frame that tears the declaration down.

The advantage of the second solution, compared to the first one, is that nodes do not need a timer for each stream they receive. Nevertheless, bridges must be able to create a frame to tear the registrations down.

IV. Consistency

As it was introduced in Section I, consistency is an important property in distributed systems for critical applications. In this work we differentiate two levels of consistency. The first level refers to consistency for the application. Some of the applications targeted by TSN require the different nodes to carry out coordinated actions. In these applications, consistency in the communications is key to guarantee the correct operation of the overall system. The first step towards achieving consistent communications is to reserve the network resources consistently. Thus, at this level, SRP should guarantee that all interested listeners have resources reserved for the communication or none of them have.

The second level refers to consistency for the infrastructure, even if the reservation itself is not consistent. We consider this second level to preserve SRP's behaviour, i.e., to allow the talker to communicate with some listeners, even if not all of them can receive. In this sense, we provide a consistent view of the reservations in all devices and leave for the application the decision to use the stream or not.

We found some consistency issues in both levels even in the absence of faults. The problems detected are mainly due to the fact that the information related to the reservations is propagated in a single direction. That is, the TA message transmitted by a talker is forwarded always towards the listeners. On the other hand, when listeners and bridges reply to a stream declaration, the information is only forwarded towards the talker. Thus, not all the devices involved in the reservation receive the same information.

A. Consistency for the Application

As said before, in this level of consistency it is important that all listeners interested in the stream can receive the same data. This is specially important for nodes that carry out coordinated actions. However, as explained in Section II, in SRP resources can be reserved for a subset of listeners, even when there are listeners willing to communicate that do not have resources to do it. In this case, the talker only communicates to a subset of listeners, generating an unnoticed inconsistency in the exchange of data. This means that actually starting a stream (with some listeners) has priority over doing it consistently (with either all or none of them).

Furthermore, even when all listeners willing to bind have enough resources to do so, there are scenarios where consistency for the application is not guaranteed all the time. This can happen for two reasons, first the paths between a talker and different listeners may differ in length and end-to-end delay and, second, the talker starts transmitting as soon as it receives the response of one listener ready to receive. Therefore, some listeners willing to bind to the stream, with enough resources throughout the whole path towards the talker, may miss the first frames transmitted by the talker.

One possible solution could be to use a timeout in the talker which should be adjusted depending on the topology, to ensure that there is enough time for the furthest listeners to communicate to the talker their intention to bind. When the timeout expires the talker checks the result of the reservation. If the reservation failed for one or more listeners (the talker receives an LRF or an LF message) or no listeners are interested in the stream (the talker does not receive response) the talker will tear the stream down. On the other hand, if the reservation was successful for all listeners (the talker receives an LR message), the talker will start communicating. In this way we ensure consistency for the application, as whether all nodes have a successful reservation or none of them have.

B. Consistency for the Infrastructure

Although the previous solution eliminates the inconsistencies for the application it also restricts some features of SRP that may be interesting for certain applications. For this reason, we now abandon the all-or-nothing approach of the previous subsection, and instead we aim at guaranteeing that all devices share the same view of the network, i.e., all devices know which listeners can bind and which ones cannot. Note that SRP does not provide this level of consistency neither; as talkers know that there are listeners willing to bind that cannot do it, but do not know which listeners they are.

Furthermore, not all devices receive the same information. This also affects bridges as they make reservations using local information mostly and provide limited information about their

results to other bridges. This can lead to inconsistencies in the reservations also in bridges, as the information two bridges receive from another bridge changes depending on whether they are on the path to the talker or to the listener.

Let us use an example to illustrate the type of problems that this can cause in bridges. Let us assume we have a talker attached to a listener through two bridges in a line topology (T-B1-B2-L). When the listener replies to a stream declaration and bridge B2 has enough resources, B2 reserves them. However, if bridge B1 does not have resources it cannot reserve them, but it does not inform bridge B2. Thus, B2 reserved resources but the stream is actually not created.

One solution could consist in introducing two lists in the listener responses, in bridges and in talkers. One of the lists would contain the IDs of listeners which can receive (hereinafter list of nodes with resources or LNR) and the other one would contain the IDs of listeners which cannot receive (hereinafter list of nodes with no resources or LNnR). Listeners interested in the stream would include their node ID in their listener response in the corresponding list, according to their resources and the talker attribute received (TA or TF).

Bridges would update their internal lists when receiving a listener response. If the response from the listener is not modified in the bridge (see Section II for details) the bridge updates its internal lists with the received node IDs accordingly, i.e., a node ID from the LNR would be saved in the bridge's LNR and one from the LNnR in the bridge's LNnR. On the other hand, if a bridge modifies the listener response due to a lack of resources in the receiving port, it would add the received node IDs in its own LNnR. Furthermore, the talker would also update its own lists upon receiving a listener response. In this way the talker would know which nodes could bind and which ones could not.

Finally, the talker would propagate the information through the network. Specifically, it would use a timeout to bound the time it waits for a response from the listeners, while guaranteeing that they have enough time to transmit it. Once the timeout expires, the talker would transmit the lists to the rest of devices so all have the same view of the network. This solves the asymmetry in the information that different devices receive. Thus, the application can now make decisions having a complete view of the network, and bridges can delete registrations when these are not going to be used.

Nevertheless, this solution can be complex and requires a significant amount of changes to SRP. Thus, if it is not required for nodes to have a consistent view of the reservations, but we want to prevent bridges from wasting resources we could use a simpler solution. Specifically, a bridge that received a listener response stating that there are enough resources in the path towards the listener, but did not have enough resources in the receiving port, would inform the rest of devices about it.

V. Conclusions and future work

The AVB Task Group started a set of projects to provide standard Ethernet with soft real-time capabilities. The interest in the work of the group reached areas with tighter constrains in terms of timing guarantees and fault-tolerance. For this reason, the group was renamed to TSN and its scope broadened to provide hard and soft real-time guarantees, flexibility of the traffic and fault tolerance mechanisms.

Among the projects carried out by the groups, we find the standardisation of the Stream Reservation Protocol. SRP is key to provide timing guarantees, as it supports the reservation of resources. SRP proposes three different architectures to carry out the resource reservation. In this work we focus on the distributed architecture of SRP proposed in AVB.

We have performed a first analysis to evaluate whether the distributed architecture is adequate for the critical applications targeted by TSN. Specifically, we focus on the termination and consistency in the reservations. We identified two different levels in both properties. The first one refers to the application level, while the second one refers to the infrastructure level. We detected problems in both levels for both properties and proposed a series of solutions based on timeouts and additional exchange of messages in order to solve them.

As future work we plan to model the proposed solutions in the mentioned Uppaal model of SRP we developed [7].

Acknowledgment

This work is supported in part by the Spanish Agencia Estatal de Investigación (AEI) and in part by FEDER funding through grant TEC2015-70313-R (AEI/FEDER, UE). Drago Čavka was supported by a scholarship of the EUROWEB+ Project, which is funded by the Erasmus Mundus Action II programme of the European Commission.

References

[1] "IEEE Standard for Local and Metropolitan Area Networks—Virtual Bridged Local Area Networks Amendment 14: Stream Reservation Protocol (SRP)," *IEEE Std 802.1Qat-2010 (Revision of IEEE Std 802.1Q-2005)*, Sept 2010.

[2] "IEEE Standard for Local and Metropolitan Area Networks–Bridges and Bridged Networks – Amendment 31: Stream Reservation Protocol (SRP) Enhancements and Performance Improvements," *IEEE Std 802.1Qcc-2018 (Amendment to IEEE Std 802.1Q-2018 as amended by IEEE Std 802.1Qcp-2018)*, pp. 1–208, Oct 2018.

[3] M. D. Johas Teener, A. N. Fredette, C. Boiger, P. Klein, C. Gunther, D. Olsen, and K. Stanton, "Heterogeneous networks for audio and video: Using ieee 802.1 audio video bridging," *Proceedings of the IEEE*, vol. 101, no. 11, pp. 2339–2354, Nov 2013.

[4] A. Nasrallah, A. S. Thyagaturu, Z. Alharbi, C. Wang, X. Shao, M. Reisslein, and H. ElBakoury, "Ultra-Low Latency (ULL) Networks: The IEEE TSN and IETF DetNet Standards and Related 5G ULL Research," *IEEE Communications Surveys Tutorials*, vol. 21, no. 1, pp. 88–145, Firstquarter 2019.

[5] D. Park, J. Lee, C. Park, and S. Park, "New Automatic De-Registration Method Utilizing a Timer in the IEEE802.1 TSN," in *2016 First IEEE International Conference on Computer Communication and the Internet (ICCCI)*, Oct 2016, pp. 47–51.

[6] O. Kleineberg, P. Fröhlich, and D. Heffernan, "Fault-Tolerant Ethernet Networks with Audio and Video Bridging," in *ETFA2011*, Sept 2011, pp. 1–8.

[7] D. Bujosa, D. Čavka, I. Álvarez, and J. Proenza, "First Analysis of the AVB's Stream Reservation Protocol in the Context of TSN," University of the Balearic Islands (UIB), Tech. Rep. A-02-2019, May 2019. [Online]. Available: http://srv.uib.es/first-analysis-of-the-avbs-stream-reservation-protocol-in-the-context-of-tsn/

[8] I. Álvarez, J. Proenza, and M. Barranco, "Towards a Time Redundancy Mechanism for Critical Frames in Time-Sensitive Networking," in *ETFA2017*, Sept 2017, pp. 1–4.