Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Memory-efficient and Safe Sequence Recovery Algorithm for IEEE 802.1CB in Time-Sensitive Networks

Arnaud Dion
arnaud.dion@isae-supaero.fr
ISAE-SUPAERO
Jérôme Lacan
ISAE-SUPAERO
Ahlem Mifdaoui
ISAE-SUPAERO

Research Article

Keywords: FRER, TSN, 802.1CB, Safety, Memory footprint, Sequence Recovery Algorithm, FPGA

Posted Date: April 21st, 2026

DOI: https://doi.org/10.21203/rs.3.rs-9246656/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License. Read Full License

Additional Declarations: No competing interests reported.

Memory-efficient and Safe Sequence Recovery Algorithm for IEEE 802.1CB in Time-Sensitive Networks

Abstract

Time-Sensitive Networking (TSN) has become a key enabler for deterministic and reliable communications for safety-critical systems. Among its mechanisms, Frame Replication and Elimination for Reliability (FRER) (IEEE 802.1CB) enhances reliability by replicating frames over multiple paths and eliminating replicates at the receiver. However, the correct configuration of FRER—particularly its sequence recovery function—remains a major challenge, as improper parameter tuning can lead to unsafe behavior, including the acceptance of replicate frames or the loss of valid ones. In this paper, we address these limitations by proposing a novel recovery algorithm, Sequence-Intervals Vector Recovery Algorithm (SI-VRA), which extends the standard FRER while remaining compliant with existing specifications and reducing configuration complexity. We evaluate the proposed algorithm and conduct extensive comparisons with the standard FRER recovery algorithm across various network scenarios. Results demonstrate that SI-VRA outperforms the standard algorithm in terms of safety, while reducing memory requirements. Additionally, we present a hardware implementation on FPGA, providing insights into its practical feasibility, latency, and memory footprint.

Keywords: FRER, TSN, 802.1CB, Safety, Memory footprint, Sequence Recovery Algorithm, FPGA

1 Introduction

The rapid expansion of time-sensitive applications across various domains, including industrial automation, automotive systems, and aerospace, has necessitated stringent guarantees for determinism and reliability in network communications.

Time-Sensitive Network (TSN) standards [1] have emerged as an attractive solution to meet these requirements, providing mechanisms to ensure low latency, bounded jitter, and high reliability. Among these mechanisms, Frame Replication and Elimination for Reliability (FRER) [2] plays a central role in enhancing reliability and decreasing data loss by replicating packets over multiple paths and eliminating replicates at specific nodes in the network.

Despite its importance, the FRER mechanism faces several challenges in practical deployments, as identified in [3, 4], to guarantee safe behavior, i.e., to avoid accepting replicates or discarding new packets. In this paper, we focus on one of the most important functions in FRER, the Sequence Recovery Function (Sec. 7.4 in [2]) that examines the sequence number of each frame and discards replicates through two standardized recovery algorithms.

One key challenge is the need for an adequate configuration of the recovery algorithms [4], which significantly impacts FRER behavior's safety. Specifically, this consists in: (i) selecting the appropriate recovery algorithm according to the traffic profile, i.e., Constant Bit Rate (CBR) or bursty; (ii) managing the choice of replicates paths to accommodate the CBR traffic profile and tuning the sequence history size to correctly eliminate replicates for bursty traffic; (iii) tuning a timeout to cover latent error detection. Moreover, the correct setting of these parameters becomes even more complex under scenarios with varying levels of replicates paths jitter, i.e., delay deviation between replicates paths.

Currently, there are no clear recommendations in the standard concerning this matter, which may lead to FRER's misconfiguration and consequently potential failures for safety-critical systems.

On the other hand, this fact has encouraged research into FRER configuration and analysis, particularly the replicates paths management [5], adequate tuning of FRER parameters [4] and analysis [6–8]. Furthermore, some implementations at the hardware level have been proposed in the literature to validate the FRER behavior's safety [9–12].

However, these existing approaches often result in tremendous configuration efforts and substantial memory footprint, making FRER deployment highly complex and inefficient for resource-constrained environments. Additionally, there is currently no comprehensive solution guaranteeing FRER safety, while taking into account realistic network behavior with potential bursty packet losses and variable replicates paths jitters.

Our main objective in this paper is to bridge these gaps through extending the standard recovery sequence function of FRER with a novel yet compliant recovery algorithm, called Sequence-Intervals Vector Recovery Algorithm (SI-VRA). Our proposed algorithm has the following features that fulfill the main challenges of FRER: (i) support of both traffic profiles, i.e., CBR and bursty; (ii) minimization of the configuration effort where one needs to tune only one parameter even under latent errors, i.e., no need of timeout configuration to cover latent error detection; (iii) safe and memory-efficient deployment under realistic network behavior, especially when accommodating bursty packet losses and high replicates paths jitters.

Hence, our main contributions in this paper are:

- Specification of a novel recovery algorithm, Sequence-Intervals Vector Recovery Algorithm, guaranteeing safe behavior of FRER while minimizing the configuration effort and memory footprint;
- Probability of failure and memory requirement analyses of SI-VRA based on Markov chain model;

- Evaluation of our proposal in comparison to the standard recovery algorithm under varying different network parameters, such as the packet loss rate and paths jitter. The results show that our proposal outperforms the existing algorithm in terms of safety (the probability of failure) as well as memory costs;
- Hardware implementation of our algorithm on FPGA and measurements of its memory footprint and latency.

The remainder of this paper is organized as follows: Section 2 presents the main concepts of FRER mechanism as well as its main limitations. An overview of related work is presented in Section 3. Section 4 details the proposed SI-VRA. Sections 5, 6 and 7 detail respectively the system model and metrics, the safety and scalability analyses and evaluation of our proposal under various network conditions and scenarios, with reference to the standard recovery algorithm. Finally, Section 8 details the hardware implementation of such a proposal and some measurements of its memory footprint as well as its latency.

2 Background: IEEE 802.1CB concepts and Limitations

In this section, we first introduce the FRER mechanism as standardized in IEEE TSN [2]. Then we discuss the main limitations and warning points to implement FRER.

2.1 Main concepts

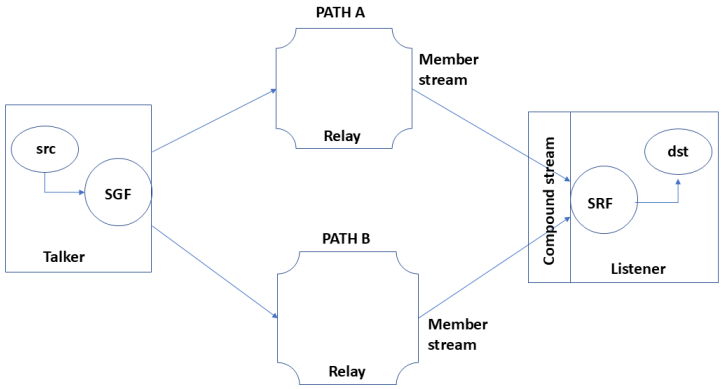


Fig. 1: *An example of FRER use with two replicates paths and main functions*

The TSN FRER mechanism is a seamless redundancy mechanism that allows redundant transmissions and the elimination of replicates packets. This mechanism is used to decrease the end-to-end packet loss ratio by distributing the content of streams on multiple paths, so that the loss of some of the paths does not cause the loss of streams. To do so, a talker or a relay system (e.g., switch) with FRER capability generates a sequence number encoded on 2 bytes for each transmitted frame and forwards the multiple frame copies over multiple paths. Afterwards, the replicated frames are eliminated either at the listener (e.g, final destination) or an intermediate relay system. As illustrated in Figure 1, each replicate flow on a disjoint path is called member stream. All member streams of the same flow represent a compound stream.

In FRER, there is a distinction between: (i) *intermittent* streams, i.e., there is no more than one packet in transit on any given member stream (or path) than on any other member stream (or path); (ii) *Bulk* streams, i.e., there is at least one member stream (or path) where there are many more packets in transit in comparison to other member streams (paths).

One of the most important functions in FRER is the Sequencing Function (Sec. 7.4 in [2]), that acts differently depending on the packet processing direction. This function assigns a sequence number for every frame going from the upper layer to the lower layer, based on a Sequence Generation Function (SGF), illustrated in the talker in Fig. 1.

In addition, it examines the sequence number of each frame coming from the lower layer to the upper layer and discards the replicates, based on Sequence Recovery Function (SRF), illustrated in the listener in Fig. 1. The Sequence Recovery Function implements two possible recovery algorithms.

1. Match Recovery Algorithm (MRA) is used for *intermittent* streams. At the initialization, the MRA accepts the first packet and stores its sequence number. The replicates having the same sequence number are eliminated whereas the others are accepted. The highest sequence number among the accepted packets is stored and considered for the next step of MRA to either accept or discard packets. It is worth noting that MRA fits only intermittent streams to avoid passing replicates.
2. Vector Recovery Algorithm (VRA) is suitable for *Bulk* streams. Similarly to MRA, at the initialization, the VRA accepts the first packet and stores its sequence number as the highest sequence number accepted using a variable RecovSeqNum (defined in 7.4.3.2.3 in [2]). Then, VRA defines a range of sequence numbers as follows:

   $$RecovSeqNum \pm frerSeqRcvyHistoryLengh$$

   , where frerSeqRcvyHistoryLengh (defined in 10.4.1.6 in [2]) defines the history length of recently received packets with a sequence number lower than RecovSeqNum and the default value in the standard is 2. All the new packets with sequence numbers in this range are accepted and replicates are discarded. Received packets with sequence numbers outside this range are discarded even if they are seen for the first time and marked as rogue. Accepted packets with sequence number higher than RecovSeqNum lead to update RecovSeqNum variable with the highest sequence number among those, and consequently the range is shifted to be centered on the new RecovSeqNum value. This recovery algorithm relies on a bit-vector

representation to track the reception status of packets within a sliding window of recent sequence numbers. Such a structure enables deterministic replicates discard, but may introduce a high-memory requirement.

Finally, the Sequencing Function implements latent error detection function using a timeout to monitor periods of time during which no packets have been accepted by the SRF. This timeout is defined by the variable FrerSeqRcvyResetMSec (10.4.1.7 in [2]) in milliseconds and it is handled by a timer. Each time a new packet is accepted, the timer restarts; else the timer is decremented. If the timer achieves zero, SRF is reset.

2.2 Limitations

We detail herein some issues when implementing FRER, and particularly the Sequence Recovery Function, that have been identified after analyzing the standard [2] and the literature [3] [4].

First, one needs to correctly select the recovery algorithm for the SRF. There are two possible recovery algorithms to discard replicates: MRA and VRA. The use of these algorithms depends on the difference in terms of packets between member streams. If the difference is at most one and this is generally the case for intermittent streams, it is better to use MRA; else VRA is recommended and in this case we talk about bulk streams. However, there is no clear guidance in the standard on how to enable the correct choice of the recovery algorithm within SRF. For MRA, we need to guarantee that the difference between any two paths is at most one packet for intermittent streams. Indeed, if this requirement is not fulfilled MRA can accept replicates. In that case, the only possible choice is VRA to correctly discard packets.

Second, one needs to correctly configure the network to enable the correct use of each recovery algorithm. For MRA, we need to select the paths for replicates in a way that we guarantee that there is no more than one packet of difference between replicates paths. This implies selecting the paths for replicates in a way that this constraint is guaranteed, which is equivalent to solving a routing problem under constraints. This detail is not covered by the standard and the network designer has to be aware that it is part of the MRA configuration.

To enable the correct use of VRA, we need to tune the sequence history length to correctly define the interval range that will condition the acceptance or the discard of packets. If this history length is not well configured, the sequence recovery function may lead to accepting replicates and even worse discarding new packets. This detail has been arisen in the standard in Note 1 of p.39 in [2]: *"... the sequence recovery function need only to support a minimum difference in path length for the correct elimination of duplicate in bulk streams. If the actual path difference in a given network exceeds the capability of a sequence recovery function, then duplicate packets will be delivered. It is up to the user to see that the capabilities of the systems match the needs of the particular network"*. Hence, the sequence history size scales with both the stream transmission rate and the replicates paths jitter. In practical deployments, correctly dimensioning this parameter is difficult, especially in networks with variable jitter or highly heterogeneous path delays [4], [3]. As a result, an undersized bit-vector may

lead to the acceptance of replicates or the discard of new packets, while an oversized one increases hardware memory requirements.

Third, the latent error detection function that is part of SRF is based on a timeout that needs to be configured by the designer. If this timeout is too short, there is a risk to accept replicates because the sequence number vector has been reset too early. On the other hand, if the timeout is too long there is a risk to discard new packets transmitted after a connection failure for instance, since this event has not been detected during the timeout. This issue has been identified in the standard in Note p.46 in [2]. So this timeout has to be well configured by the network designer.

Hence, as one can notice there is a tremendous effort to configure the sequence recovery function, specified in the standard [2], to guarantee safe behavior with no risk of accepting replicates or discarding new packets. However, there is no guidelines in the standard for the correct configuration, which leads to increasing the design complexity and introducing potential failures for safety-critical systems.

Our main objective in this paper is to overcome these limitations through introducing a compliant yet efficient recovery algorithm, called Sequence-Intervals Vector Recovery algorithm (SI-VRA) that will be detailed in Section 4.

3 Related Work

The existing approaches in the literature leveraging the FRER usage are mainly based on the idea of configuring its parameters correctly using analysis or hardware implementation.

The most relevant works concerning FRER configuration based on analysis are conducted at different levels. In [3], the authors have identified several challenges of the IEEE 802.1CB standard at the qualitative level but no solutions have been proposed to solve them. More recently, in [4], in addition to highlighting further challenges when using FRER, the authors have detailed configuration guidelines of various FRER parameters to guarantee safe behavior and conducted quantitative analysis based on simulation. Concerning more particularly the path redundancy configuration, [5] utilizes the OMNeT++ simulation environment to model FRER mechanisms, focusing on orchestration and path redundancy. Similarly, [6] explores redundancy management in safety-critical TSN applications, emphasizing reliability through temporal fault injection techniques. In the area of simulation, there are also interesting approaches to analyze the performance of FRER [6,7] to measure paths' latencies, where a simulation framework based on OMNeT++ has been developed for TSN mechanisms, including FRER. However, using simulations does not provide deterministic guarantees. Some existing works have provided such guarantees based on Network Calculus [13] for particular FRER configuration options, as stated in [2, AnnexB] considering replication and elimination only at end-devices, such as Parallel Redundancy Protocol (PRP) [14] and High Availability Seamless Redundancy Protocol (HSR) [15], and more recently in the general use-case [8],i.e., at end systems and/or intermediate nodes.

On the other hand, there exist only few seminal works in the literature considering the FRER implementation at the hardware level.

Public implementations mainly concern HSR and PRP. In [9], the authors describe the principles of HSR and PRP and provide software implementations and a "basic HSR implementation" on a Xilinx Spartan-6 Field Programmable Gate Array (FPGA) (XC6SLX45T). The validation is very basic and consists simply in checking that both paths work. [10] proposes an implementation of HSR/PRP on an Altera Cyclone IV FPGA and presents a demonstration for a profinet network. [11] acknowledges the performance problem of maintaining a simple table of frame identifiers, and discusses the improvements of either using a hash algorithm to speed up the search, or using a sliding window. [12] provides only a software implementation of HSR based on a multiplexer.

As stated above, existing works attempt to overcome the highlighted limitations of FRER through configuring correctly its parameters based on analysis or hardware implementation. In this paper, our primary goal is to bridge these gaps and provide a new recovery algorithm extending the sequence recovery function for IEEE 802.1CB to decrease the configuration effort while guaranteeing safe behavior. To the best of our knowledge, this is the first approach proposing a new recovery algorithm compliant with the standard and tailored for hardware deployment with memory-efficient implementation.

4 Sequence-Intervals Vector Recovery Algorithm

In this section, we first explain the main design principles of our proposal SI-VRA. Then, we detail the underneath algorithm as well as an illustrative example.

4.1 Design Principles

The main principle of our algorithm lies in considering real networks conditions that often deviate from the implicit assumption of independent packet losses. Measurements and modeling studies have shown that packet losses are frequently bursty, leading to correlated packet loss patterns rather than uniformly distributed missing packets [16]. This behavior can be captured using Markov-based channel models such as the Gilbert or Gilbert–Elliott models [17–20], which explicitly describe alternating periods of successful reception and loss of packets.

These observations motivate a more compact representation of the sequence history. Instead of storing the reception status of each individual sequence number using a bit-vector representation as proposed in the standardized VRA [2], the proposed approach represents the sequence history as a vector of **intervals of sequence numbers** of consecutive correctly received packets. This abstraction leverages the bursty nature of losses: when frames are successfully received over several consecutive transmissions, their sequence numbers are naturally contiguous and can be represented by a single interval.

The resulting Sequence-Intervals vector representation preserves the functional semantics of the FRER sequence recovery function while significantly reducing memory usage. In particular, the memory requirement depends on the number of packets reception gaps rather than on the delay deviation between replicates paths, i.e., replicates paths jitter. This property makes the approach especially attractive for FRER

```
   /* Upon arrival of frame F with sequence number s: F_s                    */
1  if ∃i s.t s ∈ [l_i, h_i[ then
       /* A replicate has already been received                              */
2      Delete F_s
3  else
       /* This is the first replicate of F_s                                 */
4      Transfer F_s
5      if ∃i s.t s = h_{i-1} and s + 1 = l_i then
           /* Intervals I_{i-1} and I_i are consecutive                      */
6          merge intervals I_{i-1} and I_i
7      else if ∃i s.t s = h_i then
           /* Extend interval upward                                         */
8          Increment h_i: h_i = s + 1
9          if h_{U-1} − l_0 > 2^{n-1} − 1 then
               /* Adjust the max span across intervals                       */
10             Increase l_0: l_0 = h_{U-1} − 2^{n-1} + 1
11     else if ∃i s.t s + 1 = l_i then
           /* Extend interval downward                                       */
12         Decrement l_i: l_i = s
13     else
           /* Create a new interval                                          */
14         if U is the maximum number of intervals then
               /* Delete the oldest interval                                 */
15             Merge intervals I_0 and I_1
16         Create a new interval [l_i = s, h_i = s + 1[
```

Algorithm 1: Management of Sequence-interval slots upon arrival of a frame in SI-VRA

hardware implementation in time-sensitive networks, where deterministic latency and bounded resource usage are critical design objectives. Furthermore, unlike existing hardware implementations [9–12] tailored to particular redundancy solutions like HSR or PRP, the proposed approach covers the general case of FRER mechanism and its hardware implementation remains scalable as replicates paths jitters increase as detailed in Section 8.

Because only a limited number of intervals can be stored, the algorithm also incorporates a memory-efficient policy. When all sequence-interval slots are occupied and a new gap appears, the two oldest sequence-interval slots are merged. Sequence numbers located between them are then assumed to have been received. Although this approximation may cause discarding very late replicates due to the packets out-of-order, its impact on reliability is minor when varying packet losses as shown in Section 7.

4.2 SI-VRA Algorithm

The Algorithm 1 underneath our proposed approach SI-VRA is described herein.

One of the main input parameters of the algorithm is parameter U, which represents the maximum number of sequence-intervals slots to manage the sequence-numbers history of a given stream. The sequence history is represented by at most U ordered sequence-intervals slots as follows:

$$[l_0, h_0[, [l_1, h_1[, \ldots, [l_{U-1}, h_{U-1}[$$

Where each interval $I_i = [l_i, h_i[$ denotes that all sequence numbers from l_i to $h_i - 1$ have already been received. An incoming frame with sequence number s is therefore considered as already processed if and only if there exists $i \in [0, U-1]$ such that $s \in I_i$.

This interval abstraction compresses consecutive correctly received sequence numbers into a compact representation based solely on sequence-interval bounds, thereby reducing memory requirements in comparison to the bit-vector sequence history used for standardized VRA [2].

Upon reception of a frame, our recovery algorithm updates the sequence-interval slots according to the following situations.

Replicate detection.

If the received sequence number belongs to one of the stored intervals, the frame is classified as a replicate and immediately discarded. This corresponds to the same decision specified in the standard [2]. See line 1 of Algorithm 1.

Merging two consecutive intervals

If the received sequence number bridges two previously disjoint intervals, meaning that it is adjacent to the upper bound of one interval and simultaneously to the lower bound of the next interval, the two intervals are merged into a single larger interval. This operation reflects the closure of a previously missing gap in the reception history. The associated frame is considered as a new frame and forwarded. See line 5 of Algorithm 1

Extension of an existing interval.

If the sequence number s is immediately adjacent to one bound of an interval i, the interval is extended. In particular, if $s = h_i$, then the interval i becomes $[l_i, s+1[$; or if $s + 1 = l_i$, then the interval i becomes $[s, h_i[$. These operations maintain the invariant that each interval represents a maximal contiguous run of correctly received sequence numbers. The associated frame is considered as a new frame and forwarded. See line 7 to 11 of Algorithm 1.

It is worth noting here that to maintain the correctness of the sequence recovery function, our algorithm enforces the following constraint (see line 9 of Algorithm 1):

$$h_{U-1} - l_0 < 2^{n-1} \tag{1}$$

Where n is the number of bits needed to encode each sequence number. Since sequence numbers belong to a modulo 2^n space, then ordering sequence-intervals slots must account for wrap-around effects. The algorithm assumes that two consecutively transmitted packets belonging to the same stream differ by at most $2^{n-1} - 1$.

Therefore, to guarantee consistent sequence-intervals comparisons even when sequence numbers cross the modulus boundary, we consider the following convention: temporal ordering between two sequence numbers s and s' is determined using signed modular subtraction; if $s' - s > 0$, then s' is considered newer; otherwise, it is considered older.

In addition to ensuring that the active sequence history remains unambiguously ordered in the modulus space, the condition in Eq. 1 allows our recovery algorithm to tolerate replicates-paths jitters of up to $2^{n-1} - 1$ packets. This improves our algorithm scalability with respect to replicates-paths jitters while preserving compatibility with the FRER standard [2].

Creation of a new interval.

If none of the previous conditions applies, the received sequence number starts a new interval of length one, namely $[s, s+1[$. The associated frame is considered as a new frame and forwarded. It is worth noting that since the number of sequence-interval slots is bounded by U, the algorithm must handle the case where all sequence-interval slots are already occupied. In this situation, the two oldest slots are merged before inserting the new one. Sequence numbers located between the merged slots are then assumed already received. This bounded-memory policy ensures deterministic storage requirement and constant update complexity, which are essential properties for hardware implementations of redundancy mechanisms in TSN devices [3], [9], [10] (see line 13 of Algorithm 1).

Finally, the memory cost of the sequence-intervals slots representation is proportional to $2U$, since only sequence-interval bounds must be stored. For each incoming frame, replicate detection requires at most U comparisons with interval limits, making the processing complexity compatible with deterministic hardware pipelines and large-scale TSN deployments.

4.3 Illustrative Example

To illustrate our recovery algorithm, consider the example shown in Table 1, where the system is configured with 4 sequence-interval slots and the sequence numbers are encoded on 8 bits. Therefore, the system must ensure that the maximum difference between the oldest and latest sequence numbers does not exceed 127, i.e., 2^7. We detail herein how sequence-intervals slots evolve as frames are received. Recall that $[i, j[$ indicates that frames F_i, F_{i+1}, ..., F_{j-1} have been received and already forwarded.

Table 1: Example of evolution of sequence-interval slots

Frame arrival	**decision**	I_0	I_1	I_2	I_3
Initial state		$[33, 35[$	$[37, 49[$	$[50, 120[$	$[125, 160[$
F_{110}	Discard	$[33, 35[$	$[37, 49[$	$[50, 120[$	$[125, 160[$
F_{160}	Forward	$[33, 35[$	$[37, 49[$	$[50, 120[$	$[125, 161[$
F_{161}	Forward	$[34, 35[$	$[37, 49[$	$[50, 120[$	$[125, 162[$
F_{49}	Forward	$[34, 35[$	$[37, 120[$	$[125, 162[$	$\emptyset$
F_{123}	Forward	$[34, 35[$	$[37, 120[$	$[123, 124[$	$[125, 162[$

As one can notice, considering the initial state of sequence-interval slots in Table 1, for each frame arrival event, our algorithm leads to an associated decision, particularly:

1. The frame with sequence number 110 is discarded because it falls within an existing interval $[50, 120[$ (see line 1 of Alg. 1);
2. The frame with sequence number 160 is forwarded and extends the upper bound of interval I_3 (see line 7 of Alg. 1);
3. When the frame with sequence number 161 arrives, it is forwarded and extends the interval further (see line 7 of Alg. 1), but since the total range exceeds 128, the lower bound of I_0 is adjusted (see line 9 of Alg. 1);
4. The frame with sequence number 49 is forwarded and merges two adjacent intervals I_1 and I_2 (see line 5 of Alg. 1);
5. The frame with sequence number 123 is forwarded and leads to the creation of new sequence-interval slot (see line 13 of Alg. 1).

5 System Model and Metrics

In this section, we first present the system model including the network and flows models. Afterwards, we introduce metrics to analyze the safety as well as the scalability of the standardized VRA as well as our proposed SI-VRA. The main notations used throughout this paper are summarized in Table 2.

Table 2: Summary of notations

CMI^f	Period (resp. the min. inter-arrival time) of periodic (resp. sporadic) flow f
MFS^f	Maximum frame size of f
MIF^f	Maximum size of the burst of f expressed in packets number during CMI^f
J^f	Maximum jitter at source f
$\mathcal{I}^f$	Set of member flows of a compound flow$f$
ϕ^f	Path of flow f
J^f_{repl}	Max. jitter between replicates paths of f
Δ^f_{repl}	Max. differential number of packets between replicates paths of f
PLR	Packet Loss Rate
r	Probability of transition from G to B
P_{fail}	Probability of failure of a recovery algorithm
M_{min}	Minimum required memory for a safe recovery algorithm
H	Sequence History Length for VRA
U	Sequence-Intervals slots number for SI-VRA

5.1 System Model

We consider a time-sensitive network (TSN) composed of devices, e.g., end-systems and switches, that is packet-switching full-duplex and transports packets between applications. The considered network implements the FRER mechanism that can be enabled in one or many redundant sections. A redundant section starts with a SGF (for

replication) and ends by a SRF (for elimination). These replication and elimination functions can be implemented on any device within the network. Figure 1 illustrates an example of a redundant section between the initial source (talker) and the final destination (listener).

A TSN flow is a coherent sequence of packets that originates from a unique source to reach one or several destinations. In addition, we consider four main characteristics for each TSN flow as described in [21]: (i) Class Measurement Interval (CMI) for the period for periodic flow and for the minimum inter-arrival time for sporadic one; (ii) Maximum Frame Size (MFS) for the maximum frame size of any packet within the flow; (iii) Maximum Interval Frames (MIF) the size of the burst during CMI, e.g., $MIF = 1$ for non bursty traffic; (iv) Maximum jitter at the source (J).

In this paper, we focus on flows submitted to FRER mechanism. Along each redundant section for a given *compound* flow f, we consider the associated set of *member* flows $\mathcal{I}^f = \{f_0, f_1, \ldots, f_k\}$, where $k > 0$. All member flows in $\mathcal{I}^f$ have the same characteristics of the compound flow f, i.e., The tuple $(CMI^f, MFS^f, MIF^f, J^f)$.

Each compound flow f has different replicates paths that may have different end-to-end delays. We consider $\forall f_i \in \mathcal{I}^f$ the associated path ϕ^{f_i} within a redundant section and that the end-to-end delay along ϕ^{f_i} is bounded within $[d^{f_i}_{min}, d^{f_i}_{max}]$. Hence, we define the maximum jitter between the different replicates paths of the compound flow f, J^f_{repl}, as follows:

$$J^f_{repl} = \left(\max_{f_i \in \mathcal{I}^f} d^{f_i}_{\max} - \min_{f_j \in \mathcal{I}^f} d^{f_j}_{min} \right) \tag{2}$$

In our model, to handle different flows profiles during the evaluation process, we convert for each flow f the temporal maximum replicates-paths jitter J^f_{repl} in the maximum differential number of packets between replicates paths, denoted Δ^f_{repl} as follows:

$$\Delta^f_{repl} = MIF^f * \lceil \frac{J^f_{repl} + J^f}{CMI^f} \rceil \tag{3}$$

This maximum differential number of packets between replicates paths accounts for: (i) the maximum number of transmission occurrences (the ceil function) during the maximum replicates paths jitter, J^f_{Repl}, when taking into account the source jitter J^f and CMI^f; (ii) The maximum burst MIF^f during CMI^f.

On the other hand, we consider that packet losses may occur in such a TSN network. To model packet loss behavior, we adopt the simple 2-state Markov chain Gilbert model [17] as defined in [19] and depicted in Fig. 2.

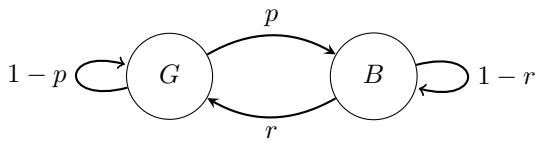


Fig. 2: The 2-state Markov Chain Gilbert loss model

This model is based on the following assumptions:

- Frames are received when the state X is $X = G$;
- Frames are lost when $X = B$.

The state transition probabilities are:

$$\mathbf{P}(X_n = B \mid X_{n-1} = G) = p \tag{4}$$
$$\mathbf{P}(X_n = G \mid X_{n-1} = B) = r \tag{5}$$

Hence, a burst of received (resp. lost) packets occurs when consecutive packets maintain the same state G (resp. B).

The transition matrix of this simplified model is:

$$\begin{array}{c} \\ G \\ B \end{array} \begin{array}{c} \begin{array}{cc} G & B \end{array} \\ \begin{bmatrix} 1-p & p \\ 1-r & r \end{bmatrix} \end{array}$$

Under these assumptions, we have well known results [22] concerning: (i) the Packet Loss Rate (PLR), PLR which is the probability of being in the state B; (ii) the mean burst length of losses b_{loss}, which is the average number of consecutive steps in state B; (iii) the state transitions probabilities r and p. These are as follows:

$$PLR = \frac{p}{p+r}$$
$$b_{loss} = 1/r$$
$$r = \frac{p(1-PLR)}{PLR}$$
$$p = \frac{r.PLR}{1-PLR}$$

A particular case of this packet loss model is the *Bernoulli* memoryless process, when the transitions between both states are independent. This is achieved when $p = 1-r$; thus $PLR = p = 1-r$ and the mean burst size of losses reaches its minimum.

5.2 Metrics

To evaluate and compare the safety and scalability of our proposed SI-VRA to standardized VRA, we introduce two metrics.

The first metric is the **probability of failure** P_{fail}. This probability captures the risk that the recovery algorithm behaves unsafely, i.e., by discarding new frames or accepting replicates.

The second metric is the minimum **memory size** M_{min} required for the recovery algorithm to satisfy a given safety level. This metric is critical in hardware-oriented

TSN deployments, where the amount of available on-chip memory directly impacts implementation cost and scalability.

6 Safety and Scalability Analyses

In this paper, we focus on the Vector Recovery Algorithm specified in the standard [2] since it fits the bursty traffic profile, commonly used in TSN networks for video, audio and monitoring applications. This algorithm is considered as a baseline to evaluate the safety of our proposed recovery algorithm SI-VRA. In this section, we detail the analyses of both metrics the probability of failure P_{fail} and minimum required memory M_{min} to assess respectively the safety and scalability for the standardized VRA as well as our proposed SI-VRA.

6.1 Vector Recovery Algorithm Analysis

First, we explicit the necessary condition to correctly configure the sequence history length H of the standardized VRA [21], as well as the timeout for the latent error detection to overcome the limitations explained in Section 2.2. These results are proved in [4].

To safely use VRA, the network designer needs to select the adequate value for the sequence history length H. This parameter should guarantee that only the packets that cannot arrive anymore are dropped from the sequence vector range. For that, we need to take into account for each compound flow f the maximum replicates-paths jitter J^f_{repl} as well the number of packets that can arrive during this duration. This fact leads to the following necessary condition for safe behavior of VRA:

$$\begin{aligned} &\text{For each flow } f, \\ &\quad H \geq MIF^f * \lceil \frac{J^f_{repl} + J^f}{CMI^f} \rceil \\ &\quad \geq \Delta^f_{repl} \end{aligned} \tag{6}$$

Furthermore, the designer has to set the correct value for the timeout T to reset the sequence recovery function after an interruption, e.g., when the talker loses its connection. This value has to be safe to trigger the timeout when no more replicates can arrive. For a nominal case, if the packets of flow f are sent at the end of CMI^f, then we need to count CMI^f and source jitter J^f, in addition to the maximum jitter between replicates paths. If no packet is received after this duration, this means that there is an interruption on the link and we need to reset the SRF to handle new sequence numbers. This is equivalent to the following necessary condition:

$$\begin{aligned} &\text{For each flow } f, \\ &\quad T \geq J^f_{repl} + J^f + CMI^f \end{aligned} \tag{7}$$

Both conditions leads to the computation of probability of failure P_{fail} as follows:

If necessary conditions (6) and (7) are verified, then:

$$\begin{aligned} &P_{fail} = 0 \\ &\text{else,} \\ &P_{fail} = PLR \end{aligned} \tag{8}$$

When the necessary conditions are not verified, we consider that the probability of failure of VRA is equivalent to the failure of the network channel, PLR. Since all replicates arriving too late in this case are tagged as "rogue", this is similar to not having replicates paths anymore, which is like having only one used path guaranteeing its own probability of failure, PLR.

To compute the minimum required memory M_{min}, we need to measure the required size of the bit-vector representation of VRA, that is centered on the maximum received sequence number within a range H. This leads to the following:

$$M_{min}(bits) \geq 2 * H \tag{9}$$

6.2 Sequence-Intervals Vector Recovery Algorithm Analysis

To analyze the P_{fail} of our approach SI-VRA, we consider the following assumptions for any flow f submitted to FRER mechanism:

- There are at least 2 replicates paths between a sender and a receiver (see Fig. 1);
- If a packet is lost on one path, it is received on the other;
- Only a maximum number of sequence intervals U can be stored.

Consider for $k \in [0, U-1]$ that the interval $I_{(k+1)}$ is used to store the $(k+1)$-th received sequence of contiguous and in-order sequence numbers. To optimize interval management, we assume that only the sequence numbers of the first and last packets in an interval are stored:

$$I_k = [F_n, F_{n+m}[$$

Where packet F_n and the subsequent $m-2$ packets have been received, whereas packets F_{n-1} and F_{n+m} are missing.

The failure model of our approach SI-VRA is based on a $(2U+1)$-states Markov chain Gilbert model [17], as illustrated in Fig. 3.

This model is based on the following assumptions:

- The sequence numbers are correctly received in order and contiguous manner when the state is $X_k = G_k$;
- The sequence numbers are lost when the state is $X_k = B_k$;
- The creation of the $(k+1)$-th interval starts when the model transitions from B_k to G_{k+1}. This transition is triggered when we start receiving new packets after the $(k+1)$-th bursty packet loss has occurred.
- The creation of the $(k+1)$-th interval ends when the model transitions from G_{k+1} to B_{k+1}. This transition is triggered when the $(k+2)$-th bursty packet loss occurs;

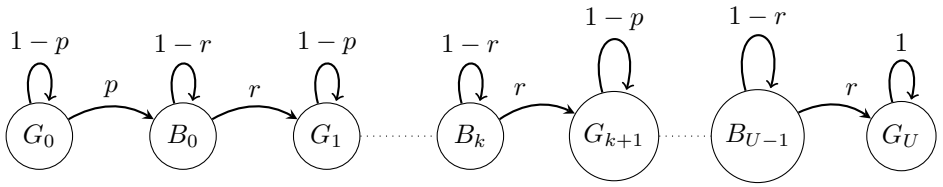


Fig. 3: The $(2U + 1)$-states Markov Chain for SI-VRA failure model

- G_0 is the initial state of the Markov chain where we have already initialized one sequence interval;
- G_U is the last state of the Markov chain and is achieved when a failure of SI-VRA occurs.

Let us consider the maximum differential number of packets between replicates paths Δ^f_{repl} for any flow f submitted to FRER. A failure of our approach SI-VRA occurs when the state transitions from G_0 to G_U occur within fewer than Δ^f_{repl} packets. The probability of failure of SI-VRA for a flow f is then:

$$P_{fail} = \mathbf{P}(X_{\Delta^f_{repl}} = G_U \mid X_0 = G_0)$$

The transition matrix of this model is:

$$\begin{array}{c} \\ G_0 \\ B_0 \\ G_1 \\ \vdots \\ B_{U-1} \\ G_U \end{array} \begin{array}{c} \begin{array}{cccccc} G_0 & B_0 & G_1 & \dots & B_{U-1} & G_U \end{array} \\ \begin{bmatrix} 1-p & p & 0 & \dots & 0 & 0 \\ 0 & 1-r & r & \dots & 0 & 0 \\ 0 & 0 & 1-p & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1-r & r \\ 0 & 0 & 0 & \dots & 0 & 1 \end{bmatrix} \end{array}$$

The probability P_{fail} of this model can be computed with the Chapman-Kolmogorov equation for n-step transition probabilities [23].

On the other hand, the minimum memory required for SI-VRA to behave safely depends only on the number of sequence intervals U. Since we assume that to identify an interval we only store the sequence numbers of the first and last received packets of the associated contiguous and in-order packets sequence, and that each sequence number is encoded on 2 Bytes, then we have:

$$M_{min}(bits) \geq 32 * U \tag{10}$$

7 Safety and Scalability Evaluation

In this section, we evaluate the safety and scalability of the proposed Sequence-Intervals Vector Recovery Algorithm, in comparison with the standardized Vector Recovery Algorithm. First, we describe the considered setup and scenarios. Then, we report the obtained results and discuss the trade-off safety versus scalability of both algorithms.

7.1 Setup and Scenarios

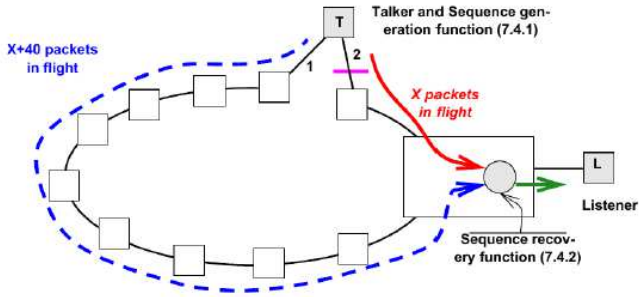


Fig. 4: An example of a time-sensitive network based on a ring topology enabling two replicates paths [2]

We consider a representative setup of a time sensitive network based on a ring topology, as illustrated in Fig 4 extracted from the standard [2]. We focus on a given flow of interest, submitted to the FRER mechanism, that has two replicates paths; and we vary the differential number of packets between its replicates paths, denoted Δ_{repl} for short.

This parameter directly captures the effect of paths asymmetry on recovery algorithms performance. In the considered setup, Δ_{repl} is progressively increased to determine the maximum range that can be tolerated by SI-VRA and VRA before the probability of failure and/or required memory become significant.

First, we compute both metrics the probability of failure P_{fail} as well as the minimum memory M_{min} of SI-VRA and VRA, when varying Δ_{repl}. This will highlight the impact of replicates-paths jitter on safety and scalability. The idea is to compute one metric while fixing the other to measure independently the safety and the scalability of SI-VRA, with reference to the standardized VRA. Afterwards, we compute P_{fail} versus M_{min} to measure the trade-off safety-scalability.

We conduct such evaluation under various packet-loss scenarios. First, two PLR values are considered in the experiments:

- $PLR = 10^{-3}$, representing a relatively unreliable channel where redundancy mechanisms are essential to maintain acceptable delivery guarantees;
- $PLR = 10^{-5}$, corresponding to a higher-quality channel typically targeted in safety-critical TSN deployments.

These values are chosen to highlight how the proposed algorithm behaves under both stressed and favorable network conditions. In addition, while maintaining the same overall Packet Loss Rate (PLR), three representative packet losses are considered to cover a wide spectrum of temporal correlation levels and investigate the impact of loss correlation on our recovery algorithm:

- **Independent losses (Bernoulli process):** This configuration corresponds to the special case $p = 1 - r$, where consecutive packet losses are statistically independent. This situation represents the most unfavorable condition for SI-VRA because losses are uniformly distributed rather than grouped into bursts; thus, leads to an increasing number of sequence intervals. This scenario establishes a conservative lower bound on the achievable safety level for our approach SI-VRA.
- **Moderately bursty losses:** In this case, the transition probability from the loss state to the good state in Fig. 2 is set to $r = 0.5$, resulting in an average loss-burst length of two packets. This scenario reflects network environments where short congestion episodes or transient interference cause temporally correlated losses.
- **Highly bursty losses.** To represent strongly correlated loss conditions, the parameter is set to $r = 0.1$, corresponding to an average burst size of ten consecutive lost packets. Such behavior may occur in wireless links, industrial environments with electromagnetic disturbances, or heavily loaded deterministic networks where contention periods are localized in time.

Therefore, considering that a network channel is defined by the pair (PLR, r), we have in total six network-channel configurations to conduct the safety and scalability evaluation of both recovery algorithms, SI-VRA and VRA.

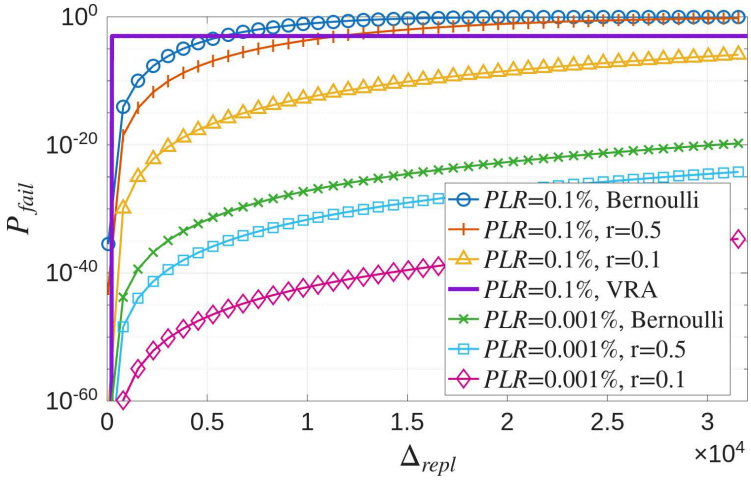
In the following figures, the performance of SI-VRA is bounded where the minimum is achieved under the most unfavorable case of network channel, i.e., a Bernoulli process and a high $PLR = 10^{-3}$, while the maximum is observed under the most favorable network channel, i.e., highly bursty losses with $r = 0.1$ and low $PLR = 10^{-5}$.

7.2 Results and Discussion

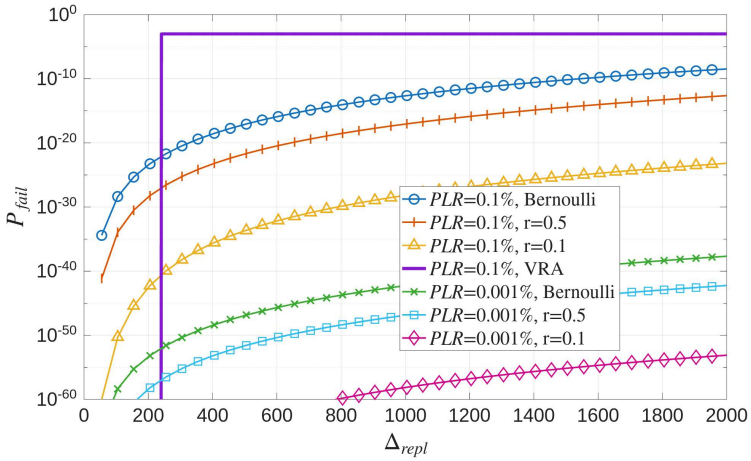
7.2.1 Safety versus maximum replicates-paths jitter

Figure 5 shows the probability of failure P_{fail} of our recovery algorithm SI-VRA and the standardized VRA as a function of the maximum differential packets number between replicates paths Δ_{repl}, under six network-channel configurations defined by the pair (PLR, r) and a fixed memory budget of 480 bits. The latter corresponds to a *Sequence History* length $H = 240$ bits for the standardized VRA and to $U = 15$ sequence intervals for our algorithm SI-VRA.

For both algorithms, P_{fail} increases with Δ_{repl}, since larger paths jitters require maintaining a longer reception history. However, both algorithms bahaviors are fundamentally different. For VRA, the admissible paths jitter is directly bounded by the length of the *Sequence History* $H = 240$. Once the effective bit-vector size is exceeded, late replicates are discarded, and the probability of failure rapidly becomes unacceptable. It is worth noting that the behavior of VRA depends only on the PLR; thus we have plot only one curve for instance when $PLR = 10^{-3}$.



(a) Full scale view



(b) Low-jitter region (Δ_{repl}<2000 frames)

Fig. 5: Probability of failure P_{fail} versus Δ_{repl} of SI-VRA and VRA under a fixed memory size of 480 bits.

On the other hand, the proposed SI-VRA exhibits a much more gradual degradation. Since reception history is represented through sequence intervals of consecutive successfully received frames, the amount of information that must be stored depends mainly on the number of reception gaps rather than on the maximum paths jitter. This enables the proposed algorithm to remain safe over significantly larger values of Δ_{repl} for the same memory budget.

The influence of loss correlation is also clearly visible in Figure 5. The most unfavorable case for SI-VRA is under a Bernoulli loss process with $PLR = 10^{-3}$. Under the

Bernoulli process, losses are independent and therefore maximize the number of disjoint reception gaps, making this configuration the worst case for SI-VRA. Even under these conditions, the proposed algorithm achieves $P_{fail} = 1.4 \times 10^{-22}$ for $\Delta_{repl} = 240$ frames. Assuming a very constrained differential paths jitter of one millisecond, this corresponds to a Mean Time Between Failure (MTBF) around $\times 10^9$ years, which is far beyond any practical operating horizon. Even for a much larger paths jitter of $\Delta_{repl} = 2000$ frames, SI-VRA still yields $P_{fail} = 3 \times 10^{-9}$. This illustrates the ability of the sequence-intervals representation to tolerate extreme network conditions, while still maintaining meaningful safety guarantees.

As loss burstiness increases, longer contiguous runs of correctly received frames are formed; thus leading to a smaller number of stored sequence intervals. This directly benefits the interval-based representation of our recovery algorithm and decreases the probability of failure under much larger values of Δ_{repl}. The gain is especially pronounced for $PLR = 10^{-5}$, where the proposed algorithm retains extremely low failure probabilities ($< 10^{-30}$) even for very large jitter values ($> 3.10^4$).

Overall, Figure 5 shows that, under a fixed memory budget, SI-VRA provides a much higher safety level than the standardized VRA, especially under moderately bursty losses.

7.2.2 Scalability versus maximum replicates-paths jitter

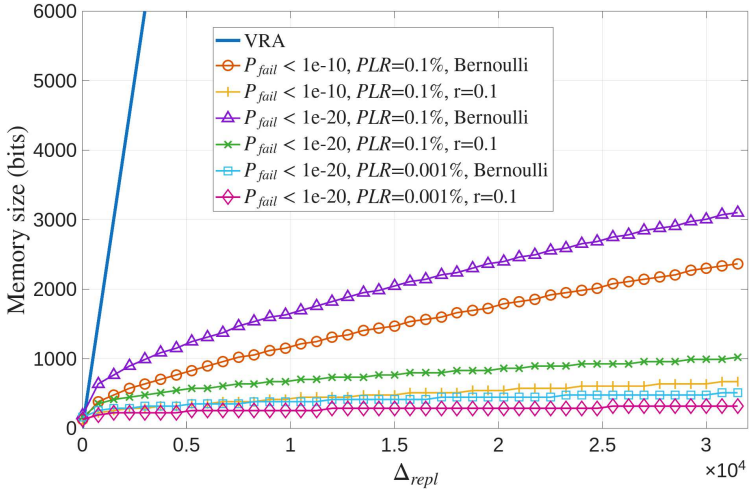
Figure 6 highlights the fundamentally different memory-scaling behaviors of both algorithms. The figure shows the minimum memory required for safe behaviors of our recovery algorithm SI-VRA and the standardized VRA as a function of the maximum differential packets number between replicates paths Δ_{repl}, under six network-channel configurations defined by the pair (PLR, r) and two safety levels with a high probability of failure 10^{-10} or a low one 10^{-20}.

For the standardized VRA, the required memory increases proportionally to the maximum paths jitter Δ_{repl}, because the *Sequence History* structure must explicitly represent all sequence numbers that may correspond to delayed replicates. This results in a linear growth of the memory footprint, independently of the network conditions.

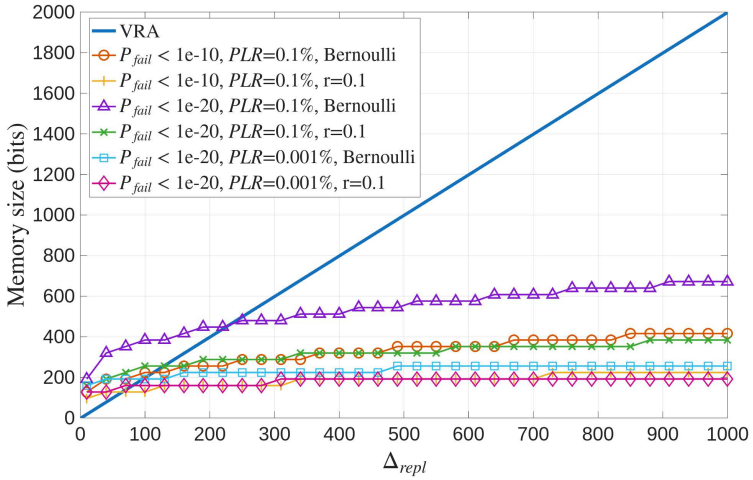
On the other hand, SI-VRA follows a markedly different scaling law. Since only the boundaries of correctly received sequence-numbers intervals are stored, the required memory depends primarily on the number of reception gaps, which is governed by the network channel characteristics. As a result, the memory requirement remains limited even for large paths jitters, especially under highly bursty losses.

This difference is already visible for small paths jitter values. For example, for $\Delta_{repl} = 100$ frames and a safety level target $P_{fail} < 10^{-20}$, our proposed algorithm requires 384 bits of memory, whereas the standard algorithm needs only 200 bits. However, this apparent advantage of the standard algorithm disappears as soon as the actual paths jitter increases until exceeding the configured memory size. In contrast, SI-VRA continues to operate safely when increasing paths jitter. For instance, with 384 bits, it still achieves $P_{fail} = 4 \times 10^{-16}$ for $\Delta_{repl} = 400$ frames, corresponding to a MTBF around 10^2 years.

Therefore, the main benefit of SI-VRA is that it scales pretty well under increasing paths jitter while maintaining a high safety level. In the particular case of very small



(a) Full scale view

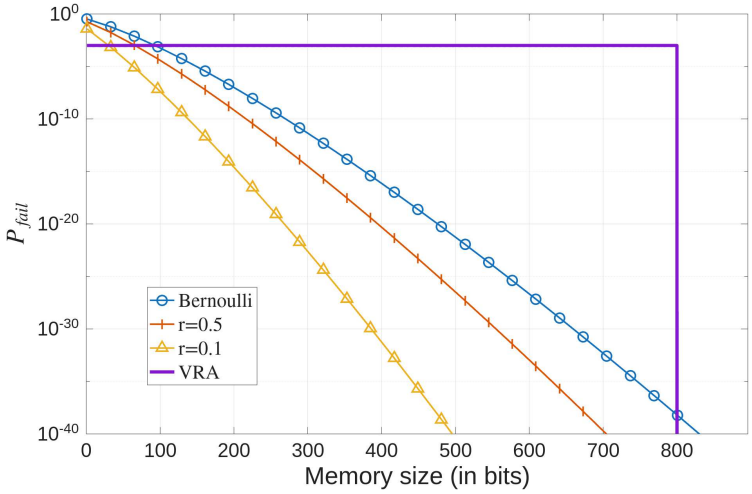


(b) Low-jitter region (Δ_{repl}<1000 frames)

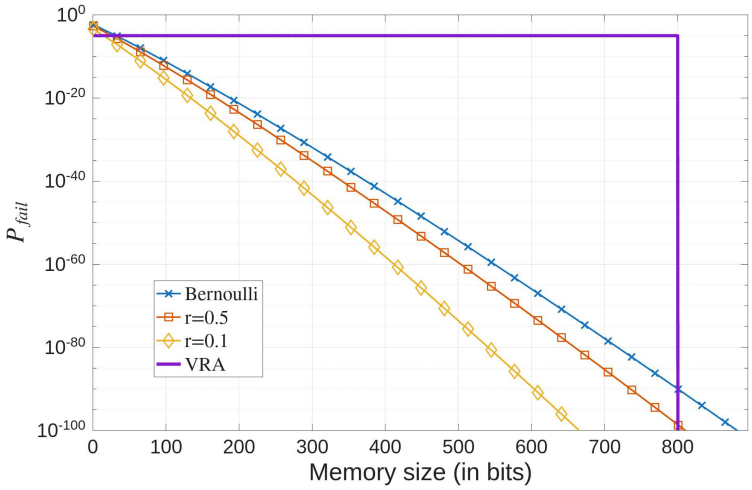
Fig. 6: Memory requirement as a function of Δ_{repl} for VRA and SI-VRA

paths jitter, its underneath implementation based on sequence intervals **relaxes the need for paths jitter estimation** and avoids drastic probability of failure when rare but extreme paths delay deviations occur. This property is particularly attractive in practice, since determining a strict worst-case Δ_{repl} is difficult in heterogeneous or dynamically changing TSN deployments.

Another important consequence of the analytical model developed in Section 6 is that memory dimensioning can be formulated as a **safety-driven design problem**. If the network channel characteristics are known — in particular the packet loss rate and the transition probabilities p or r of the underlying Markov model — then the



(a) $PLR = 10^{-3}$



(b) $PLR = 10^{-5}$

Fig. 7: P_{fail} as a function of memory size for $\Delta_{repl} = 400$ frames.

minimum number of sequence intervals, thus the required memory size, can be computed to satisfy a target P_{fail}. This provides a systematic and quantitative method to the worst-case memory dimensioning required by our algorithm SI-VRA.

7.2.3 Safety-Scalability Trade-Offs

Figures 7 and 8 complement the previous analysis by computing the probability of failure when varying the available memory and considering different network channel configurations, under paths jitter of 400 and 2000 frames, respectively.

Figure 7 considers a moderate paths jitter value $\Delta_{repl} = 400$ frames. As expected, the standardized VRA fails as soon as the available memory becomes smaller than the maximum paths jitter, since delayed replicates fall outside the *Sequence History* window. In contrast, SI-VRA continues to operate even under constrained memory budgets. Although P_{fail} increases when fewer sequence intervals are available, the resulting safety level remains far better than the one guaranteed by the standard VRA.

For $PLR = 10^{-3}$, shown in Figure 7a, even the Bernoulli process case remains highly favorable to SI-VRA compared with the standard VRA. With 384 bits of memory, corresponding to $U = 12$ intervals, our proposed algorithm achieves $P_{fail} = 4 \times 10^{-16}$. As losses become moderately bursty $r = 0.5$ or strongly bursty $r = 0.1$, the probability of failure decreases drastically when increasing the memory, since fewer intervals are needed to store the reception history.

Figure 7b shows the same experiment for a more reliable network channel with $PLR = 10^{-5}$, which corresponds to the upper safety regime of SI-VRA. In this case, extremely small failure probabilities are obtained even with reduced memory. With 384 bits ($U = 12$), the proposed algorithm reaches $P_{fail} = 5.7 \times 10^{-42}$. Even with only 288 bits of memory ($U = 9$), it still achieves $P_{fail} = 2 \times 10^{-31}$. For a moderately bursty losses network channel $r = 0.5$, the probability of failure further decreases to 4.5×10^{-46}. These results clearly show that SI-VRA safety degrades gracefully under constrained memory, whereas the standard VRA exhibits an unacceptable failure threshold.

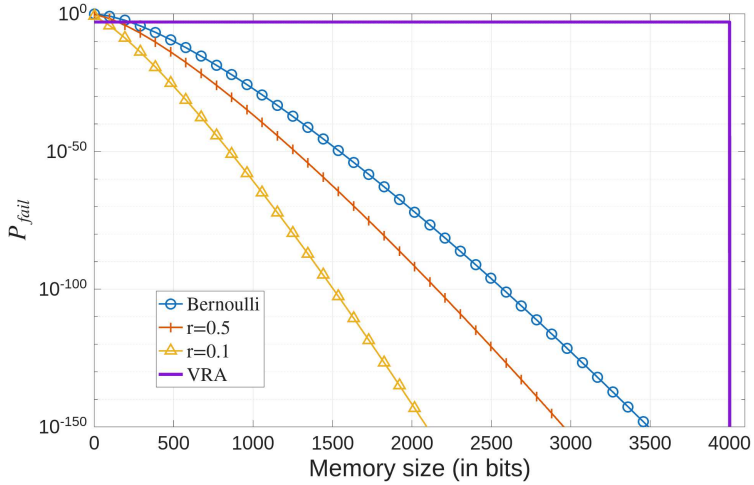
Figure 8 extends this analysis to a larger paths jitter value $\Delta_{repl} = 2000$ frames, where the benefit of the sequence-intervals representation becomes even more pronounced. In the unfavorable Bernoulli process case with $PLR = 10^{-3}$, SI-VRA achieves $P_{fail} = 1.15 \times 10^{-27}$ with 992 bits, corresponding to $U = 31$, and $P_{fail} = 4 \times 10^{-11}$ with only 512 bits, corresponding to $U = 16$. These values are obtained with memory budgets respectively about four times and eight times smaller than those required by the standard algorithm for the same paths jitter. For a moderately bursty losses network channel $r = 0.5$, the safety level further improves, with $P_{fail} = 7.9 \times 10^{-16}$.

These results confirm that the safety level of SI-VRA is much higher than the one guaranteed by the standard VRA even under constrained memory, high paths jitter and degradated network conditions.

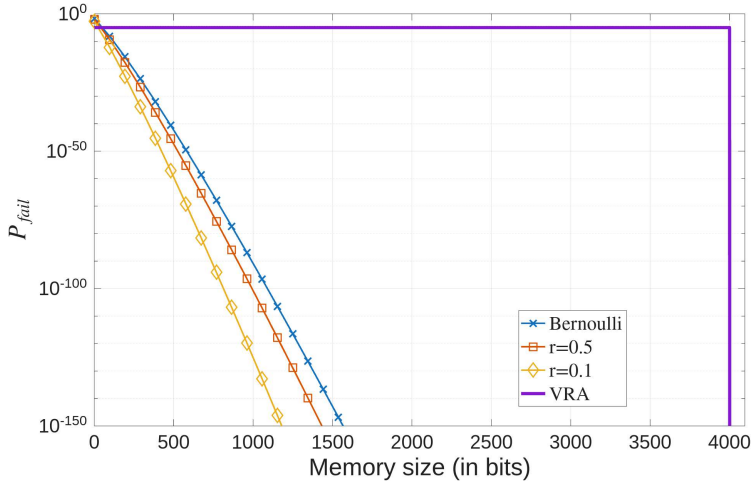
When the paths jitter becomes large, the standard bit-vector history incurs a prohibitive memory requirement to maintain high safety level, whereas the proposed sequence-interval vector representation continues to provide a beneficial trade-off between scalability (memory costs) and safety (probability of failure).

7.2.4 Main outcome

Taken together, Figures 5 to 8 show that the proposed SI-VRA fundamentally changes the design trade-off of FRER sequence recovery function. Unlike the standard VRA that requires a memory proportional to the worst-case paths jitter, SI-VRA leverages the statistical model of packet losses to maintain high safety levels with a bounded and often much smaller memory footprint. This enables the dimensioning of the sequence



(a) $PLR = 10^{-3}$



(b) $PLR = 10^{-5}$

Fig. 8: P_{fail} as a function of memory size for $\Delta_{repl} = 2000$ frames.

recovery function memory from a target failure probability and channel characteristics, rather than from an overly conservative worst-case paths jitter estimate.

8 Implementation and Measurements

The proposed algorithm has been implemented as part of a TSN gateway, referred to as the *T Gateway*, which connects an Ethernet end system to a TSN-compatible ring network (see Fig. 9). This gateway was developed on an AMD Arty7 XC7A100T FPGA to handle a TSN-Ethernet link with a bandwidth of $1\,\mathrm{Gbit\,s^{-1}}$. The data frequency

for Ethernet is 125 MHz, but the internal frequency of the gateway was increased to 175 MHz to reduce latency. Given that the algorithm's memory footprint is smaller than the standard implementation (discussed in Section 7.2.2), the primary objectives of the implementation were to achieve low and deterministic latency.

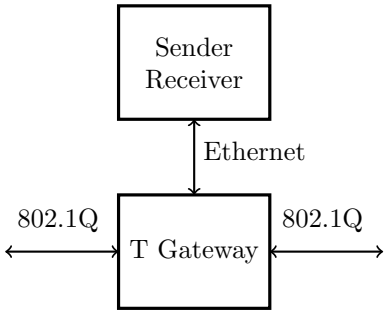


Fig. 9: TSN-Ethernet Gateway

8.1 Stream identification function

The first step in the algorithm involves identifying the stream. In this implementation, a stream is identified using the source and destination Media Access Control (MAC) addresses. Although the IEEE 802.1CB standard suggests using Virtual Local Area Network (VLAN) identifiers for stream identification, this implementation did not adopt VLANs. However, VLANs can be easily incorporated using the same approach.

Since the recovery function operates at the receiver, the destination MAC address is already known. Identifying the sender's address is achieved using Look-Up Tables (LUTs) configured as programmable decoders (see Fig. 10). A 48-bit MAC address is split into ten 5-bit vectors (with 2 zero-padding bits), and each vector is stored in a separate LUT. If all ten LUTs match the corresponding fragments of the received MAC address, the corresponding stream is identified. The result is encoded as a one-hot binary word, where the bit set to 1 indicates the address hit. A one-hot-to-binary decoder then generates the storage address for the stream's intervals.

For example, if the MAC address corresponds to the second bank of LUTs (denoted as @1), the one-hot word will be 00...010, and the generated address will be 0x02. If no match is found (all bits are 0), the received MAC address is considered unknown, and a Finite State Machine (FSM) programs it into an empty LUT bank.

This approach uses 10 LUTs per MAC address. For 128 unique streams, the implementation requires 1280 LUTs. The decoder operates at 175 MHz, achieving a latency of only two clock cycles (11.43 ns) to process a new MAC address and identify the corresponding stream address. Programming a new MAC address takes 16 clock cycles due to the use of programmable LUTs, but this latency is negligible because it occurs during frame reception.

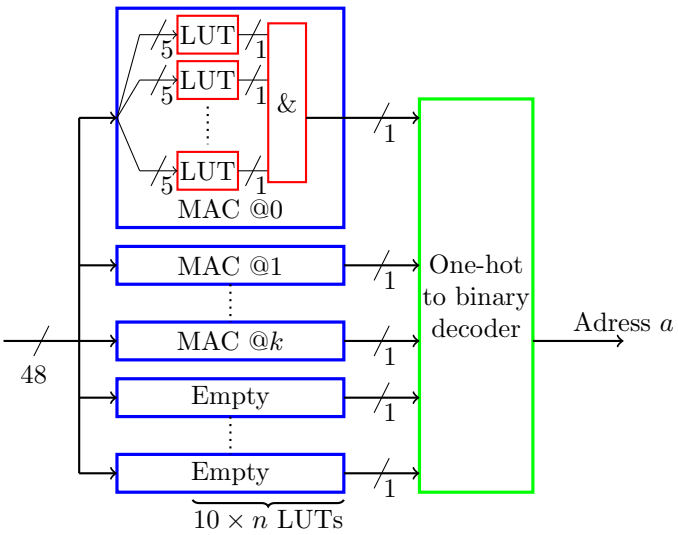


Fig. 10: stream identifier

8.2 Hardware organization of the interval-based recovery function

Once the stream identifier is obtained, the corresponding set of sequence-intervals is selected and processed by the recovery logic. As illustrated in Figure 11, the interval bounds are stored in dedicated memory arrays, each stream being associated with a fixed number U of interval slots. This bounded structure guarantees deterministic storage requirements and constant processing complexity, which are key properties for hardware deployment in TSN devices.

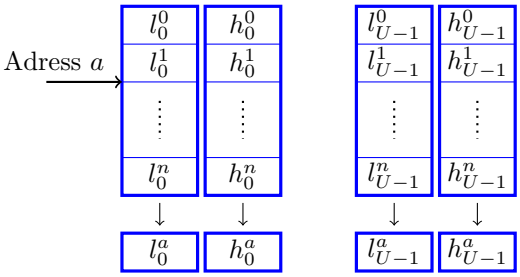


Fig. 11: Arrays of intervals

In the presented implementation, interval bounds are stored in LUT-based distributed memories. Each interval requires storing its lower and upper sequence limits,

leading to a memory cost proportional to $2U$ per stream. Depending on the targeted scalability and resource availability, the same architecture can be implemented using BlockRAM to further increase the number of supported streams or intervals.

Duplicate detection and interval management rely on a small set of relational operators implemented using modular subtractions and sign-bit extraction. In particular, interval membership testing requires only two main comparators and two equality detectors per interval slot. These operators generate compact binary decision vectors that enable fully parallel evaluation of interval conditions, ensuring deterministic latency independent of the reception history length.

With the insertion of pipeline registers required for timing closure at 175 MHz, the latency to determine whether an incoming frame has already been received is four clock cycles ≈ 22.9 ns after stream identification. Interval update operations such as merging, extension, or creation are performed through simple memory write operations and can be completed within one additional clock cycle. This results in a bounded and predictable processing time compatible with high-throughput TSN gateways.

Overall, the interval-based organization preserves the functional behavior of the recovery algorithm while significantly reducing memory footprint and maintaining a low hardware complexity, which is essential for scalable multi-stream implementations.

8.3 Hardware implementation results

To evaluate the practical feasibility of the proposed Sequence-Interval recovery algorithm, the architecture was synthesized on an FPGA target representative of embedded TSN gateway implementations. The resulting resource utilization and timing performance are summarized in Table 3.

Table 3: Sizes of implementation

Nb of streams	**Nb of Intervals**	**Flip-flops**	**LUTs**	
			logic	***memory (bits)***
128	71	5993	7774	5824
128	16	1503	2117	2304
64	16	1419	1951	1152
64	8	792	1151	896
16	16	1355	1675	672
16	8	735	935	416

The recovery pipeline is designed to ensure deterministic processing latency while sustaining high throughput. In the implemented architecture, the total latency of the elimination function is **six clock cycles**, including both stream identification and interval processing stages. This low latency is achieved through the **parallelization of stream lookup operations and interval comparison logic**, allowing duplicate detection and interval management conditions to be evaluated concurrently.

In particular, the identification of the corresponding stream context and the comparison of the incoming sequence number with all stored interval bounds are performed in parallel data paths. This organization minimizes critical-path depth and enables

efficient pipelining. As a result, the architecture can operate at a clock frequency of 175 MHz while maintaining a bounded decision time compatible with real-time forwarding constraints.

The memory footprint scales linearly with the number of supported streams and with the configured number of intervals per stream. Since only interval bounds are stored, the required storage remains significantly lower than that of a vector-based reception history dimensioned for large differential paths delays. This property makes the proposed architecture particularly suitable for multi-stream deployments where memory resources are a primary limiting factor.

Table 3 shows that the logic utilization remains moderate, reflecting the simplicity of the interval comparison operators and update logic. Combined with the deterministic latency and reduced memory requirements, these results confirm the suitability of the proposed approach for hardware integration in TSN switches or gateways requiring predictable processing behavior.

9 Conclusions and future work

This work introduces a highly safe and memory-efficient sequence recovery algorithm compliant with the FRER sequence recovery function, as specified in the IEEE 802.1CB standard [2]. By leveraging a more realistic network channel with bursty packet losses and adopting a sequence-intervals vector representation of received packets, the proposed algorithm significantly reduces memory usage thus improving sysstem scalability, in comparison with the standard recovery algorithm. Moreover, this innovative approach enables high safety level, particularly in scenarios characterized by high replicates paths jitter.

Theoretical analysis using the Gilbert model demonstrated that the algorithm can effectively handle larger replicates paths jitters compared to the standard implementation. Furthermore, hardware implementation on an FPGA confirmed its practical feasibility, achieving deterministic performance with a significantly reduced memory footprint. These results highlight the algorithm's suitability for high-reliability and low-latency applications in TSN.

In addition to its memory efficiency, the proposed algorithm offers greater flexibility and higher safety level than the standard solution. Unlike VRA, where failure probabilities increase sharply when replicates paths jitters exceed the available memory size, our proposal SI-VRA maintains low probability of failure for any replicates paths jitter. This ensures high safety level and a beneficial safety-scalability trade-off even in networks with unpredictable delays or varying traffic patterns.

Future research could explore extending the applicability of our SI-VRA in FRER-enabled networks others than in TSN context, such as in 5G/6G context where the packet losses and paths jitters are inherently high. Since our approach has shown a stable behavior and high safety level under degradated network conditions, this may trigger a significant step forward in advancing reliability in modern networking systems through a memory-efficient FRER deployment.

10 Funding

This work has received no funding.

References

[1] "Time-Sensitive Networking (TSN) Task Group," https://1.ieee802.org/tsn/.

[2] "IEEE standard for local and metropolitan area networks - frame replication and elimination for reliability," *IEEE Std 802.1CB-2017*, 2017. [Online]. Available: https://ieeexplore.ieee.org/document/8091139

[3] R. Hofmann, B. Nikolić, and R. Ernst, "Challenges and limitations of IEEE 802.1CB-2017," *IEEE Embedded Systems Letters*, vol. 12, no. 4, pp. 105–108, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/8936415

[4] L. Maile, D. Voitlein, K.-S. Hielscher, and R. German, "Ensuring reliable and predictable behavior of IEEE 802.1CB frame replication and elimination," in *ICC 2022 - IEEE International Conference on Communications*, 2022, pp. 2706–2712. [Online]. Available: https://ieeexplore.ieee.org/document/9838905

[5] D. Ergenç and M. Fischer, "Implementation and orchestration of IEEE 802.1CB FRER in OMNET++," in *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2021, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9473722

[6] M. Pahlevan and R. Obermaisser, "Redundancy management for safety-critical applications with time sensitive networking," in *2018 28th International Telecommunication Networks and Applications Conference (ITNAC)*, 2018, pp. 1–7. [Online]. Available: https://ieeexplore.ieee.org/document/8615374

[7] P. Heise, F. Geyer, and R. Obermaisser, "TSimNet: An Industrial Time Sensitive Networking Simulation Framework Based on OMNeT++," in *2016 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, Nov. 2016, pp. 1–5.

[8] L. Thomas, A. Mifdaoui, and J.-Y. Le Boudec, "Worst-Case Delay Bounds in Time-Sensitive Networks With Packet Replication and Elimination," *IEEE/ACM Transactions on Networking*, vol. 30, no. 6, pp. 2701–2715, 2022.

[9] J. Araujo, J. Lázaro, A. Astarloa, A. Zuloaga, and A. García, "High availability automation networks: PRP and HSR ring implementations," in *2012 IEEE International Symposium on Industrial Electronics*, 2012, pp. 1197–1202. [Online]. Available: https://ieeexplore.ieee.org/document/6237259

[10] H. Flatt, J. Jasperneite, D. Dennstedt, and T. Dinh Hung, "Mapping of PRP/HSR redundancy protocols onto a configurable fpga/cpu based architecture," in *2013 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS)*, 2013, pp. 121–128. [Online]. Available: https://ieeexplore.ieee.org/document/6621114

[11] H. Heine and O. Kleineberg, "The high-availability seamless redundancy protocol (HSR): Robust fault-tolerant networking and loop prevention through duplicate discard," in *9th IEEE International Workshop on Factory Communication Systems*. IEEE, 2012, pp. 213–222. [Online]. Available: https://ieeexplore.ieee.org/document/6242569

[12] J. T. Yu, "A practical and effective approach to implementing high availability seamless redundancy (HSR)," in *2017 IEEE Conference on Dependable and Secure Computing*, 2017, pp. 392–399. [Online]. Available: https://ieeexplore.ieee.org/document/8073824

[13] J.-Y. Le Boudec and P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*, ser. Lecture Notes in Computer Science, Lect.Notes Computer. Tutorial. Berlin Heidelberg: Springer-Verlag, 2001, https://www.springer.com/us/book/9783540421849.

[14] M. Li, G. Zhu, Y. Savaria, and M. Lauer, "Reliability Enhancement of Redundancy Management in AFDX Networks," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 5, pp. 2118–2129, Oct. 2017.

[15] P. Heise, N. Tobeck, O. Hanka, and S. Schneele, "SAFDX: Deterministic high-availability ring for industrial low-cost networks," in *2014 7th International Workshop on Communication Technologies for Vehicles (Nets4Cars-Fall)*, Oct. 2014, pp. 40–44.

[16] J.-C. Bolot, "End-to-end packet delay and loss behavior in the internet," *SIGCOMM Comput. Commun. Rev.*, vol. 23, no. 4, p. 289–298, Oct. 1993. [Online]. Available: https://dl.acm.org/doi/10.1145/167954.166265

[17] E. N. Gilbert, "Capacity of a burst-noise channel," *The Bell System Technical Journal*, vol. 39, no. 5, pp. 1253–1265, 1960. [Online]. Available: https://ieeexplore.ieee.org/document/6768434

[18] E. O. Elliott, "Estimates of error rates for codes on burst-noise channels," *The Bell System Technical Journal*, vol. 42, no. 5, pp. 1977–1997, 1963. [Online]. Available: https://ieeexplore.ieee.org/document/6769369

[19] G. Hasslinger and O. Hohlfeld, "The gilbert-elliott model for packet loss in real time services on the internet," in *14th GI/ITG Conference - Measurement, Modelling and Evalutation of Computer and Communication Systems*, 2008, pp. 1–15. [Online]. Available: https://ieeexplore.ieee.org/document/5755057

[20] N. Becerra Yoma, C. Busso, , and I. Soto, "Packet-loss modelling in IP networks with state-duration constraints," in *IEE Proceedings - Communications*, vol. 152, Issue 1, 2005. [Online]. Available: https://digital-library.theiet.org/doi/10.1049/ip-com%3A20040774

[21] "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)*, pp. 1–119, 2010.

[22] E. N. Gilbert, "Capacity of a burst-noise channel," *Bell system technical journal*, vol. 39, no. 5, pp. 1253–1265, 1960.

[23] D. Bertsekas and J. N. Tsitsiklis, *Introduction to Probability*. Athena Scientific, 2008.