On Packet Reordering in Time-Sensitive Networks

Ehsan Mohammadpour, Jean-Yves Le Boudec
École Polytechnique Fédérale de Lausanne, Switzerland
{firstname.lastname}@epfl.ch

Abstract—Time-sensitive networks (IEEE TSN or IETF DetNet) may tolerate some packet reordering. Re-sequencing buffers are then used to provide in-order delivery, the parameters of which (timeout, buffer size) may affect worst-case delay and delay jitter. We apply the analysis to evaluate the performance in a lossy network, re-sequencing increases the worst-case delay (but non order-preserving) network, re-sequencing is “for free”, sequencing buffers on worst-case delay, jitter and propagation and order-preserving elements. We also analyse the effect of re-sequencing buffers on worst-case delay, jitter and propagation of arrival curves. We show in particular that, in a lossless (but non order-preserving) network, re-sequencing is “for free”, namely, it does not increase worst-case delay nor jitter, whereas in a lossy network, re-sequencing increases the worst-case delay and jitter. We apply the analysis to evaluate the performance impact of placing re-sequencing buffers at intermediate points and illustrate the results on two industrial test cases.

I. INTRODUCTION

Time-sensitive networks provide real-guarantees for applications in the automobile [1], automation [2], space [3], avionics [4], [5] and video [6] industries. Standardization is taking place at the IEEE Time Sensitive Networking (TSN) working group and at the IETF Deterministic Networking (DetNet) working group. In such networks, the aim is to provide flows with hard bounds on worst-case delay and on delay jitter (defined as the difference between worst-case and best-case delays), together with zero congestion loss and seamless redundancy [7].

Time-sensitive networks may allow some limited amount of packet reordering. This may occur due to parallelism in network elements like switches and routers, routing of packets via different paths, or packet duplication [8]–[10]. The IETF states in [11] that the amount of reordering is a key quality-of-service attribute of a flow; but neither IETF nor TSN specify what it means in detail. If a time-sensitive flow is subject to possible reordering and the application requires in-order packet delivery, a re-sequencing buffer is used to restore packet order. It is typically placed at the final destination, but it is also proposed in [11] to place at intermediate points inside the network, for example if the network path between the re-sequencing buffer and the destination preserves order, or simply to reduce the amount of reordering. A re-sequencing buffer uses the assumption that the source increments a sequence number field by 1 for every packet of the flow. Early packets are stored until all packets with smaller sequence numbers arrive [12]–[13]. A timer is used to limit the waiting time of a packet in the re-sequencing buffer, as otherwise the loss of a packet in the network would cause indefinite holding of packets with larger sequence numbers.

Packet reordering is well understood in the context of best-effort networks where it is shown to be detrimental to the performance of TCP connections (see Section [12]). Several metrics were proposed to capture the amount of reordering in [12] and in RFC4737 [13]. In these references, the aim is to define reordering metrics that can be measured on a flow and can be correlated to the performance of reliable transfers over a best-effort network.

In time-sensitive networks, flows require a guarantee on worst-case delay and delay jitter, together with zero congestion-loss (no packet is discarded due to buffer overflow). To obtain such guarantees, a time-sensitive flow must conform to an arrival curve constraint at the source, which can be seen as a formal specification of a rate and burstiness constraint (see Section [12]). Then, using some forms of Network Calculus [16], the control or management plane computes worst-case delay and jitter bounds, together with the buffer sizes required for zero congestion-loss. Surprisingly, such computations are currently done without taking into account the impact of re-sequencing buffers.

The main goal of this paper is to bridge this gap and provide a theory to compute worst-case performance guarantees in presence of packet reordering and with re-sequencing buffers. Specifically, a first issue is to find appropriate per-flow information that enables proper setting of timeout value and buffer size at a re-sequencing buffer, such that deterministic guarantees hold for a time-sensitive flow, namely, no packet is lost due to spurious timeout or buffer overflow and packets are delivered in-order. A second issue is how to compute such per-flow information. A third issue is the effect of the re-sequencing buffers on worst-case delay and on delay jitter. Furthermore, an intriguing topic is to study the aforementioned issues in the interconnection of various network elements and intermediate re-sequencing buffers. A first challenge is how to systematically compute the propagation of per-flow information in a sequence of network elements that affect the re-sequencing parameters. The second challenge is to evaluate the impact of intermediate re-sequencing buffers on the end-to-end worst-case delay and delay jitter.

Our contributions are as follows. To address the first and second issues, we need appropriate metrics for per-flow re-ordering. We show in Theorem [2] that one of the metrics
in RFC4737 [15], the reordering late time offset (RTO, the definition of which is recalled in Section III-C), equals the minimal timeout value. Furthermore, combined with other information on the flow (namely arrival curve at source and delay jitter), the RTO can be used to derive the required buffer size (Theorem 3). In-line with the operation mode of time-sensitive networks, such a metric, or an upper bound on it, must be computed by the control or management plane before a flow is set up. This differs from the intended use of the metrics in RFC4737, which focus on ex-post measurements. Therefore, we propose a theory to compute tight upper bounds on RTO for flows, given the information that is otherwise available to the time-sensitive network control or management plane (Section V). Such information includes bounds on delay jitter, arrival curve constraints of flows at their sources, and whether a network element is guaranteed to preserve per-flow order or not.

Another metric in RFC4737 is the reordering byte offset (RBO). We show in Theorem 4 that it is equal to the required size of the re-sequencing buffer when the network can be assumed to be lossless. Otherwise, if packet losses cannot be ignored, we show that the RBO underestimates the required buffer size, for which we give a formula that involves the RTO. This closes the first issue.

Concerning the third issue, observe that re-sequencing buffers may delay packets until they can be delivered in-order, therefore, they may increase the worst-case delay and the delay jitter. However, we show in Theorem 5 that, if a flow is lossless between its source and a re-sequencing buffer, the worst-case delay and the delay jitter are not increased by the re-sequencing buffer (i.e. re-sequencing is for free in terms of delay in the lossless case). In contrast, if the flow may be subject to packet losses on its path from source to the re-sequencing buffer, then the worst-case delay may be increased by an amount up to the timeout value of the re-sequencing buffer, which must be at least as large as the RTO between the source and the input of the re-sequencing buffer. These results are based on a novel input-output characterization of the re-sequencing buffer.

Finally, our theory also allows to evaluate the value of re-sequencing buffers at intermediate points, in addition to the destination, in time-sensitive networks. Regarding the first challenge, our formulas in Section V-B capture in particular the pattern of RTO amplification by downstream jitter: if a non order-preserving element (typically a switching fabric) has very small RTO but is followed by a per-flow order-preserving element (typically the queuing system on an output port) with large delay jitter, then the concatenation of the two produces a large RTO. This motivates some vendors to perform per-flow re-sequencing after every switching fabric. With respect to the second challenge, we find that such intermediate re-sequencing buffers do not improve the worst-case delay nor delay jitter if the network is lossless; but they do reduce the worst-case delay, delay jitter and RTO at destination in presence of network losses. To quantify the effect of intermediate re-sequencing buffers, we also need to evaluate how arrival curves of flows are modified by re-sequencing, since such arrival curves are required to compute delay and jitter bounds (Section V-C). We illustrate the application of our theory to two industrial test cases.

The rest of the paper is organized as follows. The state-of-the-art is presented in Section II. Common assumptions, including a formal description of the RTO and RBO metrics, are given in Section III together with background results on network calculus in non-FIFO networks, and a notation list. In Section IV we provide a formal input-output characterization of the re-sequencing buffer, which is then used to establish the link between RTO and its required parameters, and to establish its effect on worst-case delay, delay jitter and output arrival curve. In Section V we show how RTO and RBO can be computed, as required to establish performance guarantees for time-sensitive flows; the method is in two parts: first, we develop formulas for an individual network element, given delay jitter and an arrival curve of the flow; then we develop a calculus to concatenate network elements. In Section VI we apply the results to analyze the performance of intermediate re-sequencing in two industrial case studies. Section VII concludes the paper. Proofs of theorems and details of computations are in appendix.

II. RELATED WORK

Kleinrock et al obtain the average re-sequencing delay in [17], assuming Poisson arrival of messages and a number of other simplifying assumptions. A more complete analysis is then performed in [18], where the distribution of the end-to-end response time, including re-sequencing delay, is obtained.

Later studies mainly focus on the statistical measurement of the occurrence of reordering in a communication network; in [8], [19], the authors indicate that the rate of packet reordering is high inside the network. Later, other works focus on the real-time techniques to measure packet reordering [20]–[22]. In [20], the authors provide a collection of measurement techniques that can estimate end-to-end reordering rates in TCP connections. In [21], the authors propose and implement an algorithm to measure reordering at a TCP receiver. The authors in [22] provide the probability density function for the amount of reordering of an arbitrary packet, based on received packets.

All the aforementioned works focus on the techniques to capture statistical information on packet reordering inside the network. Few works study the sizing of re-sequencing buffers: [13], [14] provide probability distribution of the re-sequencing buffer size. To the best of our knowledge, there is no prior work that computes the size of re-sequencing buffer and its timeout value in the context of worst-case performance (as required with time-sensitive networks) nor the effect of re-sequencing on worst-case delay and delay jitter.

III. BACKGROUND INFORMATION

A. Network Assumptions

We consider a network that contains a set of nodes, a set of hosts, and a set of links with fixed capacity. Nodes are switches or routers. A node consists of elements that can be order-preserving for the flow of interest (e.g. output port FIFO queues) or non order-preserving (e.g. switching fabric).
Every flow follows a fixed path, has a finite lifetime and emits a finite, but arbitrary, number of packets. We consider unicast flows (extension to single-source multicast flows is straightforward) with known arrival curves at their sources (i.e. there are known bounds on the number of bits or packets that can be emitted by a flow within any period of time, see Section III-E for a formal definition of arrival curve). A node may also implement a re-sequencing buffer to provide in-order packet delivery for one or several flows of interest. If a flow requires in-order packet delivery and if there is at least one non-order-preserving element on its path, then one re-sequencing buffer is required, and can be placed anywhere after the last non order-preserving element on the path. In some configurations, we will also consider that some additional intermediate re-sequencing buffers are placed inside the network.

Hosts are sources or destinations of flows. Packet sequence numbers are written at the source, starting with number 1 for the first packet sent by the flow. The sequence number is incremented by 1 for every packet of the flow, i.e., sequence numbering is per-packet per-flow. If a packet is lost in the network, with most time-sensitive applications, there is no packet retransmission; instead, the application hides the loss using some application-specific robustness mechanism (see e.g. [23]). If the source happens to retransmit the missing data, the resulting packets are assumed to have a new sequence numbering is per-packet per-flow. If a packet is lost in the network, then one re-sequencing buffer is required, and can be placed anywhere after the last non order-preserving element on the path. In some configurations, we will also consider that some additional intermediate re-sequencing buffers are placed inside the network.

This section provides a set of general assumptions to present a high-level view of the considered network. Further details, e.g., scheduling policy inside the nodes, are not required for understanding the theory presented in this paper. In Section VII, we describe two case studies; there, we describe the network and flows with all details.

B. Delay and Jitter

For a given flow, call \( d_n \) the delay of the packet with sequence number \( n \), measured from source to destination. The “worst-case delay” of the flow is \( \max_n \{d_n\} \) where the max is over all packets sent by the flow during its lifetime. Similarly, the “best-case delay” of the flow is \( \min_n \{d_n\} \). The “delay jitter” is the difference, i.e.,

\[
V = \max_n \{d_n\} - \min_m \{d_m\},
\]

so that \( d_m - d_n \leq V \) for any \( m, n \). In the above, only packets that reach their destination should be considered (lost packets are excluded). Delay jitter is called IP Packet Delay Variation in RFC 3393 [24].

Times are assumed to be measured according to the true time, i.e. the international atomic time (temps atomique international, TAI). In reality, times are measured with local clocks, which may or may not be synchronized. Some small corrections may need to be applied to delay and jitter bounds [25]; the details are for further study.

C. Packet Reordering Metrics

RFC 4737 defines a number of packet reordering metrics, two of which are of interest in the context of time-sensitive networks: the reordering late time offset (RTO) and the reordering byte offset (RBO), which we now formally define. Both metrics are defined for a flow and between an input and an output observation points. When the input observation point is not specified, it is implicitly assumed that it is the source of the flow.

We call packet with index \( n \) the \( n \)th packet observed at the input observation point, in chronological order. If the input observation point is the source, then packet \( n \) is the packet with sequence number \( n \). Let \( E_n \) be the time at which packet \( n \) is observed at the output observation point. If this packet is lost between the two observation points, we take \( E_n = +\infty \). Note that, since some network elements may be non order-preserving, \( E_n \) cannot be assumed to be a monotonic sequence. Simultaneous packet observations might be possible in some cases (e.g. if the observation of a packet depends on the realization of a software condition) and the system must use some tie-breaking rule to determine a processing order for packets; if this happens, we assume that we modify the timestamps \( E_n \) by some small amounts to reflect the tie-breaking rule i.e. we assume that if \( j \neq n \) then \( E_j \neq E_n \). If packet \( n \) is not lost, i.e. if \( E_n < +\infty \), its reordering late time offset is

\[
\lambda_n = E_n - \min_{j \geq n, E_j \leq E_n} E_j
\]

i.e. \( \lambda_n \) is the largest amount of time by which a packet with index larger than \( n \) arrives earlier than packet \( n \), i.e. the maximum amount of “overtaking” undergone by packet \( n \); if there is no reordering after packet \( n \), then \( \lambda_n = 0 \). The reordering late time \( \lambda_n \) is undefined if packet \( n \) is lost.

The reordering late time offset, RTO, of the flow between the two observation points is \( \lambda = \max_n |E_n| < +\infty \lambda_n \). It follows that, if packet \( n \) is not lost, then for any packet index \( p \geq n \):

\[
E_p \geq E_n - \lambda.
\]

We always have \( \lambda \geq 0 \) and it is easy to see that \( \lambda = 0 \) if and only if the network path between the two observation points preserves the order of packets for this flow.

For the second metric, we need to count misordered bytes; observe that packet \( n \) is misordered between the two observation points if there exists some \( j > n \) such that \( E_j < E_n \). Then, if packet \( n \) is not lost, its reordering byte offset is defined by

\[
\pi_n = \sum_{j > n, E_j < E_n} l_j,
\]

where \( l_j \) is the size, in bytes, of packet \( j \). Thus \( \pi_n \) is the cumulated number of bytes of packets with index larger than \( n \) that arrive earlier than packet \( n \); if there is no reordered packet after \( n \), then the sum is empty and \( \pi_n = 0 \). The reordering byte offset \( \pi_n \) is undefined if packet \( n \) is lost. The reordering byte offset, RBO, of the flow between the two observation points is

\[
\pi = \max_n |E_n| < +\infty \pi_n.
\]

This definition of RBO is in bytes and not in bits as is often done for buffer and packet lengths in the context of time-sensitive networks; this is to be consistent with the terminology in RFC 4737. Also observe that a similar definition could be given by counting packets rather than bytes, as is done in [12].
D. Re-sequencing Buffer

A re-sequencing buffer stores the packets of a flow until the packets with smaller sequence numbers arrive; then it delivers them in the increasing order of their sequence numbers. A re-sequencing buffer has two parameters, a size in bytes, $B$, and a timeout value, $T$. For any individual packet that is stored in the buffer, a timer is set that expires after $T$ seconds. Then, if a timer for a packet expires, all the stored packets with smaller or equal sequence number are released in-order. Hence, a packet is released if any one of the following conditions holds: 1) all packets with smaller sequence numbers are received, 2) its timer expires or 3) the timer of a received packet with a larger sequence number expires. A detailed description of the re-sequencing buffer algorithm is provided in Appendix A.

By construction, the re-sequencing buffer delivers the packets that it does not discard in increasing sequence numbers. Furthermore, a packet is discarded by the re-sequencing buffer either when the buffer is full or when the sequence number of the arriving packet is less than the largest sequence number that was already released. The latter occurs when the timeouts of packets are too early, compared to the lateness of misordered packets. Therefore, to avoid discarding packets, the re-sequencing buffer size and the timeout value should be large enough. In Section IV, we analyze how to set the parameters such that these conditions hold.

E. Arrival Curves

In time-sensitive networks, in order to provide guarantees to flows in terms of delay, jitter and zero-congestion-loss, flow rates and burstinesses must be limited at the source. This is done precisely by imposing an “arrival curve” constraint, also called T-SPEC (traffic specification) at the source of the flow. Formally, let $\alpha$ be some wide-sense increasing function $[0, +\infty) \rightarrow [0, +\infty]$; the flow is said to satisfy the arrival curve constraint $\alpha$ at some observation point if the number of bytes (or bits) observed on the flow at this observation point is upper-bounded by $\alpha(t-s)$. Without loss of generality [16], the arrival curve $\alpha$ can be assumed to be sub-additive ($\alpha(s+t) \leq \alpha(s) + \alpha(t)$ for all $s, t \geq 0$), left-continuous and such that $\alpha(0) = 0$. In this paper we assume that entire packets are observed at the observation point, which imposes that $\alpha(0^+) \geq L_{\text{max}}$, where $L_{\text{max}}$ is the maximal packet size, otherwise no packet of maximal size can be sent by the flow ($\alpha(0^+)$ stands for the right-limit of $\alpha$ at 0). An arrival curve $\alpha$ is “achievable” if, for any sequence of packet sizes between $L_{\text{min}}$ and $L_{\text{max}}$, there is a source with these packet sizes that achieves equality in the arrival curve constraint, or more specifically, if the sequence of packets obtained by packetizing the fluid source $R(t) = \alpha(t)$ satisfies the arrival curve constraint $\alpha$. Note that the fluid source $R(t) = \alpha(t)$ always satisfies the arrival curve constraint $\alpha$ since we can always assume that $\alpha$ is sub-additive; however packetization may introduce some violations [26]. Any concave arrival curve that satisfies $\alpha(0) = 0$ and $\alpha(0^+) \geq L_{\text{max}}$ is achievable [16, 26, Thm 1.7.3]. For a flow with packets of constant size, any arrival curve whose values are integer multiples of the packet size is achievable as soon as it is sub-additive, left-continuous and satisfies $\alpha(0) = 0$.

A commonly used arrival curve is the “leaky bucket” arrival curve with rate $r$ and burst $b$, defined by $\alpha(t) = rt + b$, $t > 0$ and $\alpha(0) = 0$, which expresses that the rate of the flow is limited to $r$, with a burst tolerance equal to $b \geq L_{\text{max}}$; it is always achievable. Another commonly used arrival curve is the staircase arrival curve with period $\tau$ and burst $b$ defined by $\alpha(t) = b \left\lfloor \frac{t}{\tau} \right\rfloor$, which applies to periodic flows that send $b$ bytes every $\tau$ time units. It is achievable if all packets are of size $b$. If for example, in contrast, $b = 1500$ bytes but the packets emitted by the source all have a size equal to $1200$ bytes, then this arrival curve is not achievable: $b$ should be set to $1200$, i.e. $\alpha$ should be replaced by a smaller arrival curve, which is then achievable.

Instead of counting bytes, constraints can be expressed in number of packets. Formally, let $\alpha_{\text{pkt}}$ be some wide-sense increasing function $[0, +\infty) \rightarrow [0, +\infty]$; the flow is said to satisfy the arrival curve constraint $\alpha_{\text{pkt}}$ at some observation point if the number of packets observed on the flow at this observation point on any interval $(s, t]$ is upper-bounded by $\alpha_{\text{pkt}}(t-s)$. A packet-level arrival curve used by IEEE TSN is the staircase one, with period $\tau$ and number of packets $K$, defined by $\alpha_{\text{pkt}}(t) = K \left\lfloor \frac{t}{\tau} \right\rfloor$, which expresses that the flow sends at most $K$ packets every $\tau$ time units. Packet-level arrival curves can always be replaced by an integer-valued sub-additive, left continuous function that vanishes at zero; it is then always achievable.

Last, we also need two technical assumptions. First, we assume that arrival curves are not bounded from above, i.e. $\lim_{t \rightarrow +\infty} \alpha(t) = +\infty$. This holds for all arrival curves of interest. Second, we assume that every flow has a maximum and minimum packet size $L_{\max}$ and $L_{\min}$; then the number of bytes observed on any time interval must be an element of $\mathcal{L}$, the set of all possible sums of a finite number of packet sizes. If $L_{\max} \geq 2L_{\min}$, then $\mathcal{L}$ is made of all numbers $\geq L_{\min}$; if $L_{\max} = L_{\min}$ then $\mathcal{L}$ is made of all multiples of $L_{\max} = L_{\min}$. Unless otherwise specified, we assume either of these conditions holds, as otherwise $\mathcal{L}$ is cumbersome and tightness results would become very complex. Appendix A-A provides an alternative characterization of arrival curves that is used in the proofs of the theorems.

F. Network Calculus Results in Non-FIFO Networks

In a time-sensitive network, the burstiness of a flow may increase at every node, due to multiplexing and random delays. Thus an arrival curve constraint at the source is usually no longer valid inside the network. Analysis of time-sensitive networks uses bounds on the propagation of arrival curves [27]. Such results are based on network calculus theorems that were derived for order-preserving networks [16, Section 1.4.1], but which can be extended to non-order-preserving networks, as we show next. Specifically, we will use the following two results, the proofs of which are in appendix.

**Lemma 1.** Assume a flow has arrival curve $\alpha$ at the input of some system $S$, which needs not preserve the order of packets of the flow. Assume the delay jitter of the flow through $S$ is
upper bounded by some quantity $V$. At the output of $S$, the flow has arrival curve $\alpha'$ given by $\alpha'(t) = \alpha(t+V)$. The same holds, mutatis mutandii, for packet-level arrival curves.

**Lemma 2.** Assume a flow has arrival curve $\alpha$ at the input of some system $S$, which needs not preserve the order of packets of the flow. Assume the worst-case delay of the flow through $S$ is upper bounded by some quantity $U$. At any point in time, the amount of data of the flow that is present in $S$ is upper bounded by $\alpha(U)$. The same holds, mutatis mutandii, for packet-level arrival curves.

**G. Lower Pseudo-inverse**

Let $F$ be a wide-sense increasing function $[0, +\infty) \to [0, +\infty)$. Its lower pseudo-inverse, $F^\downarrow$, is the wide-sense increasing function $[0, +\infty) \to [0, +\infty)$ defined by [28]:

$$F^\downarrow(x) = \inf \{ s \geq 0 \mid F(s) \geq x \}. \quad (5)$$

Some of the common functions and their lower-pseudo inverses are:

\[
\begin{align*}
F(t) &= rt + b, t > 0; F(0) = 0 \quad \Rightarrow \quad F^\downarrow(x) = \left\lfloor \frac{x - b}{r} \right\rfloor, \\
F(t) &= b \left\lfloor \frac{t}{\tau} \right\rfloor \quad \Rightarrow \quad F^\downarrow(x) = \tau \left\lfloor \frac{x - b}{b} \right\rfloor, x > 0; F^\downarrow(0) = 0. \quad (6)
\end{align*}
\]

It follows immediately from [28] Property P7, Section 10.1] that

$$\forall x, y \in [0, +\infty) : \quad F^\downarrow(y) < x \quad \Rightarrow \quad y \leq F(x). \quad (7)$$

**H. Notation List**

Packet $n$ is the packet with sequence number $n$ if the input point of observation is the source of the flow, otherwise it is the $n$th packet in chronological order at the input of the system of interest.

- $l_n$: length of packet $n$, in bytes.
- $L_{\min}$: minimum packet length of the flow of interest, in bytes.
- $L_{\max}$: maximum packet length of the flow of interest, in bytes.
- $A_n$: time at which packet $n$ is released by its source.
- $D_n$: departure time of packet $n$ from a re-sequencing buffer.
- $E_n$: the exit time of packet $n$ from a non order-preserving or order-preserving element.
- $T$: timeout value of re-sequencing buffer.
- $B$: size of re-sequencing buffer.
- $\pi$: reordering late time offset (RTO) of the flow of interest.
- the maximum of an empty set is $-\infty$.
- $[x]^+ = \max(x, 0)$.
- $F^\downarrow$: lower pseudo-inverse of function $F$.

**IV. Properties of the Re-sequencing Buffer**

In this Section we first provide a formal input-output characterization of the re-sequencing buffer. Then we use it to analyze the optimal parameter setting, assuming that bounds on RTO and RBO of the flow are known. Last, we characterize the performance effect of a re-sequencing buffer in terms of delay, delay jitter and arrival curve propagation.

**A. Input-output Characterization of the Re-sequencing Buffer**

We use the notation in Figure 2. Recall that a packet may be lost in the network or discarded by the re-sequencing buffer. The latter may occur either (1) when the packet arrives after a packet with smaller sequence number was released; this is due to the timeout value $T$ being too small; or (2) if the buffer size $B$ is too small. In this subsection, we assume the buffer is large enough, and in Section IV-B we compute the maximum buffer occupancy, which will give the required buffer size for a given flow. The following theorem characterizes the departure times from the re-sequencing buffer and is the basis from which the results in the rest of this section are derived.

**Theorem 1.** Consider the re-sequencing buffer described in Section III with timeout value equal to $T$ and with infinite buffer capacity $B = +\infty$. See Figure 2 for the notation.
1) The packet with sequence number \( n \) leaves the re-sequencing buffer at time \( D_n \) given by

\[
D_n = \begin{cases} 
I_n & \text{if } n = 1, \\
\max\{G_n, I_n\} & \text{if } n > 1 
\end{cases}
\]

with \( I_n = \begin{cases} 
+\infty & \text{if } E_n > \min_{j \geq n} \{E_j\} + T, \\
E_n & \text{otherwise} 
\end{cases} \)

and, for \( n \geq 2 : \ G_n = \min\left( D_{n-1}, T + \min_{j \geq n} \{E_j\} \right). \)

2) Let \( \lambda \) be the RTO of this flow between the source and the input of the re-sequencing buffer. If \( T \geq \lambda \) and packet \( n \) is not lost in the network (i.e. \( E_n < +\infty \)) then it also holds that

\[
D_n = \begin{cases} 
E_1 & \text{if } n = 1, \\
\max\{G_n, E_n\} & \text{if } n > 1 
\end{cases}
\]

where \( G_n \) is defined in (10).

3) If \( T \geq \lambda \) and the network is lossless (i.e. \( E_n < +\infty \) for all \( n \)), then it also holds that \( D_n = \max_{k \leq n} \{E_k\} \).

The proof Theorem 1 is in Appendix B. It is based on induction and uses the description of re-sequencing buffer presented in Section III.

B. Optimal Dimensioning of the Parameters of the Re-sequencing Buffer

Recall that, by construction, the re-sequencing buffer always delivers packets in order. However, it may do so by discarding late packets. We can now use the previous theorem to derive the minimal values of the timeout \( T \) and the size \( B \) of the re-sequencing buffer, such that it never discards any packet. We start with the timeout value.

**Theorem 2.** Consider the re-sequencing buffer described in Section III and Figure 2 with timeout value of \( T \) and infinite buffer size \( B = +\infty \). Let \( \lambda \) be the RTO of the flow of interest between the source and the input of the re-sequencing buffer. The minimum value of \( T \) that guarantees that the re-sequencing buffer never discards packets of this flow is \( T = \lambda \).

The proof of Theorem 2 is in Appendix B. It consists in two steps. First, using Theorem 1 item 2, we show that, if \( T \geq \lambda \), there is no packet discarded due to spurious timeout. Second, using Theorem 1 item 1, we show that, for any \( \lambda > 0 \), if \( T < \lambda \) we can construct an execution trace with RTO \( \lambda \) such that a packet is discarded due to spurious timeout.

Theorem 2 thus establishes the central role of the RTO metric as far as the timeout value is concerned. For the required buffer size, the results are more complex, as shown in the next theorem.

**Theorem 3.** Consider the re-sequencing buffer described in Section III and Fig. 2 with timeout value of \( T \) and buffer size \( B \). Let \( \lambda, \pi \) and \( V \) be the RTO, RBO and delay jitter of the flow of interest between the source and the input of the re-sequencing buffer. Assume that \( T \geq \lambda \). Also assume that the flow has arrival curve \( \alpha \) at its source. The minimal size of the re-sequencing buffer required to avoid buffer overflow is

1) \( B = \pi \), if the network in Fig. 2 is lossless for the flow;
2) \( B = \alpha(V + T) \), if the network in Fig. 2 is not lossless for the flow.

The proof of Theorem 3 is in Appendix B. It consists in four steps. First, assuming the network in Figure 2 is lossless for this flow and using Theorem 1 we show that the actual buffer content is upper bounded by \( \pi \), which shows that a buffer of size \( B = \pi \) is sufficient. Second, we show that for any \( \lambda > 0 \) and any valid RBO value \( \pi \) (a valid RBO value is a number that can be decomposed as the sum of an arbitrary number of packet sizes) there always exists one execution trace of a flow with RTO \( \lambda \) and RBO \( \pi \), that achieves a buffer content equal to \( \pi \); therefore the minimal size cannot be less than \( \pi \). If \( \lambda = 0 \), the network preserves packet order for this flow and thus \( \pi = 0 \) as well and the result is clear. This shows item 1. Third, using Lemmas 1 and 2 in Section III-E we show that, if the network is not lossless for this flow, the actual buffer content is upper bounded by \( \alpha(V + T) \). Fourth, we show that, for any achievable arrival curve \( \alpha \), RTO \( \lambda \), jitter \( V \) and timeout value \( T \), we can construct an execution trace with RTO \( \lambda \) in which the buffer content can become arbitrarily close to \( \alpha(V + T) \). This shows item 2.

**Remark.** It follows from Theorem 3 that the bound, \( \pi \), in item 1, is always less than the bound, \( \alpha(V + T) \), in item 2, as expected.

**Remark.** Loss-free operation is often considered as the normal case in time-sensitive networks, since congestion losses are avoided and transmission losses are very rare: a packet loss might then be viewed as a normal event, treated by exception-handling routines. If such an assumption can be made, Theorem 3 shows that the required buffer content is only dependent on the RBO of the flow.

However, such an interpretation should be taken with care. Indeed, the loss of a single packet before the input to the re-sequencing buffer may delay a number of other packets; the first arriving packet with sequence number larger than the lost packet is delayed at the re-sequencing buffer by \( T \), and, depending on the scenario, following packets may be delayed as well. Thus, the loss of a single packet, even rarely, may impose a delay increase to many more subsequent packets and may lead to the violation of the bound in item 1. Hence, if the bound in item 1 is used for dimensioning the re-sequencing buffer, then the loss of a single packet in the network may cause the loss of many more packets at the re-sequencing buffer due to an insufficient buffer size (since the bound in item 2 is always larger than in item 1). Quantifying this in detail is left to further study.

**Remark.** The RBO and buffer size \( B \) in Theorem 3 are expressed in bytes. Obviously, a similar result holds if we count in packets: if the flow is constrained at the source by a packet-level arrival curve \( \alpha_{\text{pkt}} \), then the size of the re-sequencing buffer, counted in packets, is upper-bounded by \( \alpha_{\text{pkt}}(V + T) \).
C. Effect of Re-sequencing on Worst-case Delay, Jitter and Arrival Curve

When re-sequencing buffers are used, they may affect packet delay. In this section, we quantify this effect in the sense of worst-case delay and delay jitter, as required in time-sensitive networks.

**Theorem 4.** Consider a flow as in Figure 2 and let \( \lambda \) be the RTO of the flow between the source and the input of the re-sequencing buffer. Assume that the timeout value \( T \) of the re-sequencing buffer satisfies \( T \geq \lambda \). The worst-case delay and the delay jitter of the flow

1. are not increased, if the network is lossless;
2. are increased by up to \( T \), if the network is not lossless.

Formally, with the notation in Figure 2, the theorem means that, if the network is lossless (i.e. \( E_n < +\infty \) for every \( n \)), then

\[
\max_n (D_n - A_n) = \max_n (E_n - A_n),
\]

\[
\max_n (D_n - A_n) - \min_n (D_n - A_n) = \max_n (E_n - A_n) - \min_n (E_n - A_n)
\]

as the former is the worst-case delay and the latter is the delay jitter.

In contrast, if there are some losses in the network, the theorem means that

\[
\max_n (D_n - A_n) \leq \max_n (E_n - A_n) + T,
\]

\[
\max_n (D_n - A_n) - \min_n (D_n - A_n) \leq \max_n (E_n - A_n) - \min_n (E_n - A_n) + T
\]

The proof of Theorem 4 is in Appendix B. It uses the input-output characterization of re-sequencing buffers in Theorem 1.

**Remark.** Item (2) of Theorem 4 is tight. Consider a packet \( n \) that experiences maximum delay \( \delta^{\max} \) while packet \( n - 1 \) is lost in the network. Therefore, packet \( n \) should wait in the re-sequencing buffer until its timer expires after \( T \) seconds. Then, packet \( n \) is delayed by \( \delta^{\max} + T \).

**Remark.** Item (2) quantifies the price of misordering under lossy operation: the re-sequencing buffer, which is caused by the presence of misordering, increases the worst-case delay and the delay jitter of the flow by an amount \( (T) \) that is at least equal to the RTO.

**Remark.** The same remark about loss-free operation holds as in Section IV-B. Specifically, the loss of a single packet may impose a delay increase to many more subsequent packets (e.g. to all packets that arrive before timeout). For example, if the flow has packet-level arrival curve \( \alpha_{\text{pkt}} \) at the source and jitter \( V \) between the source and the input of the re-sequencing buffer, it can be easily seen, using the same arguments as in the proof of Theorem 3, that the loss of a single packet may cause the delay bound in item 1 to be violated for a number of packets equal to \( \alpha_{\text{pkt}}(V + T) \). This stresses again that the result in item 1 should be taken with care, and that the delay bounds in item (2) are more realistic.

We can apply Lemma 1 to the previous theorem and quantify the propagation of arrival curves through a re-sequencing buffer:

**Corollary 1.** Consider a flow as in Figure 2 assume that it satisfies the arrival curve \( \alpha \) at the source and that the delay jitter between source and input to the re-sequencing buffer is \( V \). Also assume that the timeout value \( T \) of the re-sequencing buffer satisfies \( T \geq \lambda \), where \( \lambda \) is the RTO of the flow between the source and the input of the re-sequencing buffer. At the output of the re-sequencing buffer, the flow has arrival curve \( \alpha' \) defined by

1. \( \alpha'(t) = \alpha(t + V) \), if the network is lossless;
2. \( \alpha'(t) = \alpha(t + V + T) \), if the network is not lossless.

The same applies, mutatis mutandis, to packet-level arrival curves.

**Remark.** Part (2) of the corollary is tight. This can be shown using the same arguments as in step (4) of the proof of Theorem 3. Specifically, the bound is achieved in a scenario where an isolated packet loss occurs, followed by a burst of in-order packets.

V. Computing RTO and RBO

In the previous section we saw how to dimension a resequencing buffer in the context of time-sensitive networks, assuming that we know the RTO of the flow and, to the extent that lossless metrics are of interest, its RBO. It remains to see how the RTO/RBO, or bounds on them, can be estimated by the control or management plane in order to setup the flow. To this end, we decompose a network path into elements that are either per-flow order-preserving or not. Examples of the former are the IEEE TSN class-based queuing subsystems [29]; examples of the latter are some switching fabrics which use parallel paths to improve throughput [30]. In Section V-A we give tight RTO and RBO bounds for network elements; in Section V-B we show how to concatenate them.

A. RTO and RBO for Network Elements

**Theorem 5.** Consider a flow that traverses a system, with delay jitter upper-bounded by \( V \). Then an upper bound on the RTO of this flow between the input and the output of the system is:

1. \( V - \alpha_{\text{pkt}}(2L_{\text{min}})^+ \), if the flow has packet level arrival curve \( \alpha_{\text{pkt}} \) at the input of the system;
2. \( V - \alpha_{\text{pk}}(2L_{\text{min}})^+ \), if the flow has packet level arrival curve \( \alpha_{\text{pk}} \) at the input of the system.

The bounds are tight, i.e. for every achievable arrival curve and every value of \( L_{\text{min}} \) and \( V \) there is a system and an execution trace that attains the bound.

The proof is in Appendix B. It consists in two steps. First, based on the definition of jitter in Section II-B and the alternative characterization of arrival curves in Appendix A, we show that the RTO is not larger than the bound in Theorem 5. Second, we show that there always exists one execution trace of a flow that its RTO is equal to the bound. Specifically,
the bound is achieved in a scenario where the two packets enter a system in a greedy manner and the first one experiences the worse-case delay and the second one experiences the best-case delay.

**Remark.** If no arrival curve is known for the flow, we can always take $\alpha(t) = +\infty$ for $t > 0$ and then $\alpha'(x) = 0$; this gives the RTO bound $\Lambda = V$, i.e., jitter is a valid RTO bound for any system and any flow.

**Remark.** If $\alpha'((2L_{\min})^+ \geq V$ or $\alpha'_\text{pkt}(2) \geq V$ then the RTO bound given by the theorem is 0, i.e., there is no reordering for this flow. Thus, the theorem captures the cases where the packets sent by the flow are rare and the delay jitter of the non order-preserving system is small, so that reordering is impossible. See the next two examples.

**Remark.** If the RTO of the flow is larger than zero, according to the theorem $\alpha'((2L_{\min})^+ < V$ or $\alpha'_\text{pkt}(2) < V$. Then by (7), we have $\alpha(V) \geq 2L_{\min}$ or $\alpha_{\text{pkt}}(V) \geq 2$. This implies that if a flow has reordering, the input generates at least two packets within the duration of $V$.

Hereafter, we provide examples on computation of RTO for multi-path connections and common forms of arrival curves, i.e., staircase and leaky bucket.

**Example.** Consider an interconnection system with $K$ paths. Every path $k$ has a worst-case delay $d^\text{max}_k$ and a best-case delay $d^\text{min}_k$. The delay jitter of this interconnection system is $V = \max_{k=1, \ldots, K} d^\text{max}_k - \min_{k=1, \ldots, K} d^\text{min}_k$, the RTO for a flow at the output of this interconnection is then given by Theorem 5. Non order-preserving switching fabrics fall into this category; here, the delay jitter, and hence the RTO, are typically very small.

**Example.** Consider a flow that has packet-level arrival curve $\alpha_{\text{pkt}}(t) = K\lceil \frac{t}{T} \rceil$, expressing that at most $K$ packets are allowed in any time window of $\tau$ seconds. Here we have $\alpha'_\text{pkt}(x) = \tau \lceil x - K \rceil$, $x > 0$ and Theorem 5 gives an RTO bound equal to $\lambda = [V - \tau \lceil 2 - K \rceil ]^+$. Applying this with $K = 1$ gives that, for a flow that generates at most one packet every $\tau$ seconds:

1) if $\tau \geq V$, the flow experiences no reordering, i.e., $\lambda = 0$.
2) if $\tau < V$, the flow may experience reordering, and $\lambda = V - \tau$.

**Example.** Consider a flow that has a leaky bucket arrival curve, i.e., $\alpha(t) = rt + b$, $t > 0$, with rate $r$ and burstiness $b \geq L_{\max}$. Then, we have $\alpha'(x) = \tau \lceil x - \frac{b}{r} \rceil$ and the RTO bound given by Theorem 5 is $\lambda = [V - \tau \lceil 2L_{\min}^\text{max} - \frac{b}{r} \rceil ]^+$. It follows that:

1) if $b < 2L_{\min}^\text{max}$ and $V \leq 2L_{\min}^\text{max} - \frac{b}{r}$, the flow experiences no reordering, i.e., $\lambda = 0$;
2) if $b < 2L_{\min}^\text{max}$ and $V > 2L_{\min}^\text{max} - \frac{b}{r}$, the flow may experience reordering, and the RTO is bounded by $\lambda = V - 2L_{\min}^\text{max} - \frac{b}{r}$;
3) if $b \geq 2L_{\min}^\text{max}$ the flow may experience reordering, and the RTO is bounded by $\lambda = \frac{L_{\max}}{2L_{\min}}$.

The first case requires $L_{\max} < 2L_{\min}^\text{max}$, which we excluded when $L_{\max} \geq 2L_{\min}^\text{max}$, but it may occur when packets are all of the same size; then reordering is impossible if the delay jitter is $\leq \frac{2L_{\min}^\text{max} - b}{r}$.

**Theorem 6.** Consider a flow that traverses a system, with delay jitter upper-bounded by $V$ and with RTO upper-bounded by $\lambda > 0$. Then a bound on RBO of the flow between the input and the output of this system is:

1) $\alpha(V) - L_{\min}$, if the flow has arrival curve $\alpha$ at the input of the system and $\alpha(V) \geq 2L_{\min}$, and 0 if $\alpha(V) < 2L_{\min}$;
2) $L_{\max} (\alpha_{\text{pkt}}(V) - 1)$, if the flow has packet-level arrival curve $\alpha_{\text{pkt}}$ at the input of the system and $\alpha_{\text{pkt}}(V) \geq 2$, and 0 if $\alpha_{\text{pkt}}(V) < 2$.

The bounds are tight, i.e. for every achievable arrival curve and every value of $L_{\min}$, $L_{\max}$, $V$ and $\lambda$ there is a system and an execution trace such that the RBO of the flow is arbitrarily close to the bound.

The proof is in Appendix B. It consists in two steps. First, based on the definition of jitter in Section III-B and the alternative characterization of arrival curves in Appendix A, we show that the RBO is not larger than the bound in Theorem 6. Second, we show that there always exists one execution trace of a flow that its RBO is arbitrary close to the bound. Specifically, it is achieved in a scenario where a sequence of packets enter a system in a greedy manner and the first packet experiences the worst-case delay while the other packets already left the system.

Observe that we must have $\alpha(0^+) \geq L_{\max}$ and $\alpha_{\text{pkt}}(0^+) \geq 1$, therefore the expressions in items (1) and (2) are always non-negative. Also notice that the RBO bounds do not depend on $\lambda$ but require that $\lambda > 0$; otherwise, namely if $\lambda = 0$, there is no reordering and the RBO is 0. Last, observe that the tightness result implies that the RBO can be extremely large if the arrival curve can also be large. In other words, it is not possible to bound the RBO solely by constraining the delay jitter; for example, a non order-preserving switching fabric can have a very large RBO, limited only by the speed of the input ports, if the flows are not otherwise constrained.

**Example.** Consider a flow that has a packet-level arrival curve $\alpha_{\text{pkt}}(t) = K\lceil \frac{t}{T} \rceil$, expressing that $K$ packets are observed in any time window of $\tau$ seconds. An RBO bound for this flow is $\pi = KL_{\max}^\text{max} - L_{\max}$ except if $K = 1$ and $V \leq \tau$ in which case it is 0.

**Example.** Consider a flow that has a leaky bucket arrival curve, i.e., $\alpha(t) = rt + b$, $t > 0$, with rate $r$ and burstiness $b \geq L_{\max}$. Applying Theorem 6 for a system with jitter bound $V$, an RBO bound for this flow is $\pi = rV + b - L_{\min}$.

**Remark.** If we count reordering offset in packets instead of bytes as in [12], the bound in item (2) should be replaced by $\alpha_{\text{pkt}}(V) - 1$.

**B. Concatenation Results**

So far, we are able to compute RTO and RBO of a flow with known arrival curve for any system with known delay jitter. In practice, a flow typically traverses a sequence of network elements, of which some cause packet reordering and the rest
preserve order. We are interested to compute RTO and RBO of the flow under such situation. To tackle this problem, a trivial method is to concatenate all the elements as a single system with a delay jitter equal to the sum of delay jitters of each network element; then by applying theorems 5 and 6 we compute RBO and RTO of the combination. However, we can do better, in two ways. First, as shown in Corollary 2 we can ignore all prefix order-preserving elements when computing RTO and all suffix order-preserving elements when computing the RBO of the sequence. Second, we can use extra information about the RTO of every network element, as shown in Theorem 7.

Corollary 2. Consider the notation in Fig. 3. An upper bound on the RTO of the flow between the input and the output of the sequence \(S_1, S_2, ..., S_K\), denoted as \(\Lambda'(K)\), is obtained by applying Theorem 5 only to \(S_1, S_2, ..., S_K\). Specifically:

\[
\Lambda'(K) = \sum_{h=1}^{K} V_h - \alpha(K)(2L_{min}),
\]

where \(\alpha(K)\) is an arrival curve of the flow at the entrance of \(S_h\). Moreover, an upper bound on the RBO of the flow between the input and the output of the sequence \(S_1, S_2, ..., S_K\) is obtained by applying Theorem 6 and setting the jitter term as the arrival curve term as the one at the entrance of \(S_1\).

The proof of the Corollary is immediate: the RTO bound is justified by the definition of RTO and \(S_1, S_2, ..., S_{K-1}\) being order preserving. Similarly, since \(S_{K-1}, ..., S_K\) are order preserving, they do not affect the RBO of the whole sequence. In (16), note that since \(S_h\) is non order-preserving, \(V_h - \alpha(K)(2L_{min})\) is non order-preserving since otherwise, by Theorem 5 we can replace \(\lambda_h\) by \(V_h - \alpha(K)(2L_{min})\). It follows that the RTO bound given by Theorem 7 is always at least as good as that of Theorem 5 and improves on it whenever \(\lambda_h < V_h - \alpha(K)(2L_{min})\), i.e. whenever the RTO bound exported by \(S_h\) does improve over the knowledge of its jitter alone.

Remark. Theorem 7 indicates that the only RTO that matters is that of the first non order-preserving element in the sequence (i.e. \(S_1\)). For subsequent network elements, it is their delay jitter \(V_h\), not their RTO \(\lambda_h\), that matters. Observe that RTO is always upper bounded by delay jitter (see remark after Theorem 5). Therefore, the RTO of the flow through a sequence of nodes may be larger than the sum of the RTOs of the flow through each individual node. This can be explained as follows: If packet 2 overtakes packet 1 at node \(S_h\) by a small amount up to \(\lambda_s\), it may still happen that the delay of packet 2 through the subsequent nodes is much less than the delay of packet 1, by an amount up to the delay jitter \(\sum_{h=s+1}^{K} V_h\). Thus, as destination, we observe that packet 2 overtakes packet 1 by an amount up to \(\lambda_s + \sum_{h=s+1}^{K} V_h\). Imagine that all non-order-preserving network elements are switching fabrics, for which the RTO is tiny (sub-microseconds) and that other network elements are class-based queuing subsystems, which preserve packet order for every flow, but may have a much larger delay jitter (milliseconds or more). The end-to-end RTO is then of the order of milliseconds, orders of magnitude larger than the amount of reordering introduced by any single network element. This is the pattern of “RTO amplification by downstream jitter”.

Similarly, we can ask whether the information on RTO bound of every element can improve the RBO bound presented in Corollary 2. The following theorem shows that the answer is no.
Theorem 8. The RBO bound in Corollary 1 is tight, i.e., for every pair of sequences \( V_h, \lambda_h \) and every achievable arrival curve there exists a system and an execution trace that comes arbitrarily close to the bound.

The proof of tightness is similar to the proof of tightness in Theorem 6.

VI. APPLICATION TO PERFORMANCE OF INTERMEDIATE RE-SEQUENCING

In this section we illustrate how the results in the previous sections can be combined in order to evaluate the performance impact of intermediate re-sequencing.

A. Methodology

For a flow that requires in-order delivery but traverses a network where some elements do not preserve packet order, re-sequencing can be performed at the destination, but it is also possible to insert re-sequencing buffers at intermediate points. For example if one is placed for every flow at the output of every non order-preserving switching fabric, then the network becomes order-preserving and the end-system is relieved from the need to re-sequence. Every choice obviously comes with a different implementation cost; here we do not address such a cost. Instead, we focus on the performance impact, primarily in terms of end-to-end worst-case delay and delay jitter. In this illustration, we consider networks that do not perform flow re-shaping §16, Section 1.5], inside the network.

If losses in the network are rare enough to be ignored for standard operation, the conclusion is straightforward. Indeed, we know from Theorem 4 that, under such an assumption, re-sequencing does not increase the worst-case delay and the delay jitter.

In contrast, if lossy operation cannot be ignored, we also know from Theorem 4 that re-sequencing adds a penalty to worst-case delay and jitter that is at least equal to the upstream RTO. Furthermore, the pattern of RTO amplification due to downstream jitter may mean that the RTO at the destination is very large, even though RTOs at non order-preserving elements are minuscule. This suggests that intermediate re-sequencing may be beneficial. However, intermediate re-sequencing also introduces a delay penalty and modifies the propagated arrival curves, which must be accounted for.

![Diagram](image)

Fig. 4: A prototypical scenario used to analyze the performance impact of intermediate re-sequencing. Potential placements of re-sequencing buffers are at points 1, 2, or 3.

To fix ideas, we consider a prototypical scenario as in Figure 4 where the path of the flow of interest goes through three subnetworks, with jitters \( \{V_i\}_{i=1}^3 \) and RTOs \( \{\lambda_i\}_{i=1}^3 \). The first subnetwork may for example represent the source output queuing, a transmission link and the switching fabric of the next node. The second may represent the output queuing that follows this switching fabric, plus transmission links and the switching fabric of the following node. The third may represent the output queuing that follows the second switching fabric, plus transmission links to the final destination. By Theorem 1 we would have \( \lambda_1 << V_1 \) and \( \lambda_2 << V_2 \). We consider the following possible placements of re-sequencing buffers:

- Only at 3 (at destination end-system).
- Only at 2 (at destination edge-switch). This is the case where the last edge-switch performs re-sequencing on behalf of the destination, just after the last non order-preserving element.
- At 1 and 2 (at every switch). This occurs when the network wants to guarantee that all switches preserve per-flow order, as some vendors do.
- At 1 and 3 (at the first switch and the destination end-system). This occurs when the network wants to guarantee that all switches except the destination edge-switch, preserve per-flow order. Then the destination performs re-sequencing with smaller timeout value.

We take as baseline the case where no re-sequencing is applied and compute the increase in worst-case delay and delay jitter with respect to the baseline, for each of the placements. We give the details for delay jitter, the computations are similar for the worst-case delay. We assume the timeout values are optimal, as given by Theorem 5. The delay jitter of the baseline is \( V_1 + V_2 + V_3 \).

For the first placement (only at 3), the re-sequencing buffer at destination increases the delay jitter in the lossy case by \( T_3 \), the timeout value of the re-sequencing buffer at point 3, which is equal to the RTO between the source and point 3. By Theorem 7, it is equal to \( \lambda_1 + V_2 + V_3 \). By Theorem 4, the increase on jitter is also \( \lambda_1 + V_2 + V_3 \).

For the last placement (at 1 and 3), the re-sequencing buffer at 1 modifies the arrival curve of the flow. This affects, in general, the downstream worst-case delay and jitter. We call \( \Delta V_2 \) and \( \Delta V_3 \) the increase on delay jitter at subnetworks 2 and 3 with respect to the baseline. In the following subsection we estimate these increases numerically on two industrial cases. The re-sequencing buffer at 1 has timeout \( T_1 = \lambda_1 \), given by the RTO of subnetwork 1. For the RTO at point 3, observe that the subnetwork 1 combined with the re-sequencing buffer at point 1 is an order-preserving element; this causes the RTO at point 3 to be independent of subnetwork 1 (Theorem 7). We obtain the timeout value \( T_3 = \lambda_2 + V_3 + \Delta V_3 \). Finally, the end-to-end delay jitter is increased by \( T_1 \) at point 1 and \( T_3 \) at point 3, thus it is equal to \( V_1 + T_1 + V_2 + \Delta V_2 + V_3 + \Delta V_3 + T_3 \), which gives an increase with respect to the baseline equal to \( \lambda_1 + \lambda_2 + V_3 + \Delta V_2 + 2\Delta V_3 \).

The reasoning is similar for the two other placements. The results are given in Table 1. A similar line of reasoning can be used to compute the required sizes of re-sequencing buffers.

We observe the following. In this scenario, we expect \( \lambda_i \) to be much smaller than \( V_i \), so if \( \Delta V_i \) is small, it is beneficial.
TABLE I: Re-sequencing buffer optimal timeout value and increase on end-to-end jitter and delay upper bound (with respect to the baseline with no re-sequencing buffer) for the four placement strategies in Section VI-A. $T_i$ is the timeout of re-sequencing buffer placed at point $i$. We see that placing re-sequencing buffers at 1 and 2 provides better end-to-end delay and jitter comparing to the placement at 1 and 3.

| Re-sequencing | $T_1$ | $T_2$ | $T_3$ | Increase to end-to-end jitter and delay with respect to no re-sequencing buffer at all. |
|---------------|-------|-------|-------|-------------------------------------------------------------------------------------|
| Only at 3     | -     | -     | $\lambda_1 + V_2 + V_3$ | 0 $\lambda_1 + V_2 + V_3$ |
| Only at 2     | -     | $\lambda_1 + V_2$ | - | 0 $\lambda_1 + V_2 + \Delta V_3$ |
| At 1 and 2    | $\lambda_1$ | $\lambda_2$ | - | 0 $\lambda_1 + \lambda_2 + \Delta V_2 + \Delta V_3$ |
| At 1 and 3    | $\lambda_1$ | - | $\lambda_2 + V_3(+\Delta V_3$ for lossy) | 0 $\lambda_1 + \lambda_2 + V_3 + \Delta V_2 + 2\Delta V_3$ |

Table showing re-sequencing buffer optimal timeout value and increase on end-to-end jitter and delay upper bound.

Due to place an intermediate re-sequencing buffer, and it is also beneficial to place one at the edge-node rather than at the destination. This is because intermediate re-sequencing reduces the downstream RTO and avoids the RTO amplification pattern. However, if $\Delta V_3$ is large this benefit may be lost due to the burstiness increase caused by re-sequencing under lossy operation. In the numerical examples of the next sections, we find that, except in one case, the former effect is largely dominant. Also note that if per-flow re-shaping would be performed at every hop, the latter effect would disappear and intermediate re-sequencing would always reduce the worst-case delay and jitter under lossy operation.

B. Case Study 1: Automotive Network

We apply the methodology in Section VI-A to the double star automotive network of [31] depicted in Fig. 5. To obtain the re-sequencing buffer size and timeout, as well as the jitter and delay upper bounds, we used TFA [32, 33] (details are in Appendix C).

![Double star automotive network](image)

Fig. 5: Double star automotive network [31].

The network consists of two switches and eight hosts and the rates of the links are $c = 1$ Gbps. The output ports are FIFO and the scheduling mechanism is non-preemptive strict priority. The output ports of $h_1, S_1$ and $S_2$ offer the same service curve $\beta(t) = 125e6[t - 12\mu]$ bytes to the highest priority queue. In each switch, the switching fabric is implemented in parallel stages, i.e., reordering of packets may occur. The delay of switching fabrics is between $0.5\mu$s to $2\mu$s [34].

According to [31], the traffic is made of various flows with different priorities. In our example, we focus on ControlData flow as the only highest priority flow; it is shown as flow $f$ in Fig. 5 and the path is taken from [31]; the flow is initiated by Control Data Unit ($h_1$) and destined to Control Unit ($h_2$) in the network. The source arrival curve for flow $f$ is leaky bucket with rate 6400 bytes per second and burstiness 6400 bytes. All packets are the same size and equal to 64 bytes.

We implemented the four placement strategies in Section VI. The results are in Table I. We see that re-sequencing at every switch fabric significantly reduces delay and jitter bounds. In contrast, re-sequencing at edge node ($S_2$ only) is not beneficial: this is an instance where the burstiness increase due to re-sequencing does have an impact. We also see that the required size of the re-sequencing buffer is independent of the placement strategy.

C. Case Study 2: Orion network

We now consider the Orion crew exploration vehicle network, as described in [35] and depicted in Fig. 6, taken from [32]. For the delay and jitter analysis, we used Fixed Point TFA [32, 33] as there are cyclic dependencies in the placement of flows. The output ports in the hosts and switches are connected to the links with a rate of 1 Gbps. The output ports use the non-preemptive TSN scheduler with Credit-based Shapers (CBSs) with per-class queuing [29, 36] from highest to lowest priority, the classes are Control Data Traffic (CDT), A, B, and Best Effort). The CBSs are used separately for classes A and B. The CBS parameters idleslopes are set to 50% and 25% of the link rate respectively for classes A and B [36]. In each switch, the switching fabric is implemented in parallel stages, i.e., reordering of packets may occur. The delay of switching fabrics is between $0.5\mu$s to $2\mu$s [34]. The CDT traffic has a leaky bucket arrival curve with rate 6.4 kilobytes per second and burst 64 bytes. The maximum packet length of classes B and BE is 1500 bytes. We focus on class A. Using the results in [29], a rate-latency service curve offered to class A is $\beta(t) = 62.49e6[t - t_0]^+$ bytes with $t_0 = 12.5\mu$s.

Class A contains 30 flows with constant packet size 147 bytes, which transmit 3 packets every 8 ms. Among these flows, 10 require in-order packet delivery. The flows traverse between 2 to 7 hops. We apply two placement strategies for re-sequencing buffers: at destinations only, and at every switch (immediately after the switching fabrics).

Fig. 7 shows the end-to-end delay and jitter bounds for the flows with in-order delivery requirement, for both strategies under lossless condition. The figure also shows the delay and jitter under lossless conditions, which are the same for both strategies and, as we know from Theorem 4, are also equal to the values when there is no re-sequencing buffer. First, we
lossy network condition; and it is equal to three packet size, i.e., $7 \cdot 24$ bytes, under lossy network conditions. This occurs even though the amount of reordering late time offset at every switching fabric is minuscule: every flow has at most 7 hops and the switching fabric re-orders packet by at most 324 ns at every hop (by Theorem 5 this is less than the jitter of the switching fabric). This illustrates the pattern of RTO amplification by downstream jitter. Second, we see that if re-sequencing is performed at every switch, the increase in delay and jitter under lossy conditions is negligible, as expected Section VI-A because such a strategy prevents amplification of RTO.

We also find that the size of re-sequencing buffers are the same for the two strategies; this implies that intermediate re-sequencing does not provide any benefit in terms of buffer size. It is equal to two packet size, i.e., 294 bytes, under lossless network condition; and it is equal to three packet size, i.e., 441 bytes, under lossy network conditions.

TABLE II: Bounds on end-to-end jitter and worst-case delay for case study 1 with the four placement strategies in Section VI-A under both lossless and lossy network conditions, followed by timeout value and size of the re-sequencing buffers.

![Diagram of network configuration](image)

Fig. 6: The Orion crew exploration vehicle network, taken from [25].

and simple properties of the network elements that are on its path. We applied the theory to evaluate the performance of re-sequencing strategies in industrial networks without re-shaping. Future work will focus on the interactions between re-sequencing and flow-re-shaping or flow-damping.

VIII. ACKNOWLEDGMENTS

This work was supported by Huawei Technologies Co., Ltd. in the framework of the project Large Scale Deterministic Network. The authors thank Bingyang Liu and Shoushou Ren for fruitful discussions.

REFERENCES

[1] IEEE, “Draft Standard for Local and metropolitan area networks — Time-Sensitive Networking Profile for Automotive In-Vehicle Ethernet Communications,” IEEE P802.1DG™/D1.1, vol. In IEEE802.1 private repository. To obtain the access credentials, visit https://www.ieee.org/proceedings/52/slides/bridge-0-std003.htm or contact the IEEE802.1 chair., Oct. 2019, http://www.ieee802.org/1/files/private/dg-drafts/d1/802-1DG-d1-1.pdf.

[2] IEC and IEEE, “IEC/IEEE 60802 - Time-Sensitive Networking Profile for Industrial Automation,” vol. IEC/IEEE 60802 (D1.1), 2019, http://www.ieee802.org/1/files/private/60802-drafts/d1/60802-d1-1.pdf.

[3] ECSS, “SpaceWire — Links, nodes, routers and networks (31 July 2008) — European Cooperation for Space Standardization,” ECSS-E-ST-50-12C, Jul. 2008, https://ecss.nl/standard/ecss-e-st-50-12c-spacewire-links-nodes-routers-and-networks/.

[4] A. Committee et al., “Aircraft Data Network Part 7, Avionics Full Duplex Switched Ethernet (AFDX) Network, ARINC Specification 664,” Annapolis, Maryland: Aeronautical Radio, 2002.

[5] H. Kopetz, A. Ademaj, P. Grillinger, and K. Steinhammer, “The Time-Triggered Ethernet (TTE) Design,” in Object-Oriented Real-Time Distributed Computing, 2005. IOSRC 2005, Eighth IEEE International Symposium on., IEEE, 2005, pp. 22–33.

[6] “IEEE standard for local and metropolitan area networks–audio video bridging (AVB) systems,” IEEE Std 802.1BA-2011, [Online]. Available: https://ieeexplore.ieee.org/document/6032690.

[7] IEEE, “IEEE Standard for Local and metropolitan area networks–Frame Replication and Elimination for Reliability,” IEEE Std 802.1CB-2017, pp. 1–102, Oct. 2017.

[8] J. Bennett, C. Partridge, and N. Shectman, “Packet reordering is not pathological network behavior,” IEEE/ACM Transactions on Networking, vol. 7, no. 6, pp. 789–798, Dec. 1999.

[9] M. Laor and L. Gendel, “The effect of packet reordering in a backbone link on application throughput,” IEEE Network, vol. 16, no. 5, pp. 28–36, Sep. 2002.

[10] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, and D. Towsley, “Measurement and Classification of Out-of-Sequence Packets in a Tier-1 IP Backbone,” IEEE/ACM Transactions on Networking, vol. 15, no. 1, pp. 54–66, Feb. 2007.

[11] J. Finn, P. Thubert, B. Varga, and J. Fararks. (2019) RFC6655: Deterministic networking architecture. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6655.

[12] N. M. Piratla and A. P. Jayasumana, “Metrics for packet reordering—A comparative analysis,” International Journal of Communication Systems, vol. 21, no. 1, pp. 99–113, 2008. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/dac.884.

[13] Y. Gao and Y. Q. Zhao, “Large deviations for re-sequencing buffer size,” IEEE Transactions on Information Theory, vol. 58, no. 2, pp. 1003–1009, 2012.
Fig. 7: The end-to-end delay bounds (left) and jitter bounds (right) for the flows with in-order delivery requirement for the two strategies, i.e., placing re-sequencing buffers at the destinations or at every switch. In the lossless condition, the delay and jitter bounds do not depend on the strategy.
APPENDIX A
COMPLEMENTARY BACKGROUND

A. Alternative Characterization of Arrival Curves

We will use the following alternative representation of arrival curve constraint [37]. Consider a flow with packets 1, 2, ...n, ... of sizes \(l_1, l_2, ..., l_n, ...\) and let \(A_n\) be the observation time of packet \(n\). Assume that the indices are in order of observation, i.e. \(A_n\) is wide-sense increasing. Then saying that the flow satisfies the arrival curve constraint \(\alpha\) is equivalent to saying that for all indices \(m \leq n\):

\[
A_n - A_m \geq \alpha \left( \sum_{k=m}^{n} l_k \right)
\]

(18)

where \(\alpha\) is the lower-pseudo inverse, defined in Section III-C. Similarly, the flow satisfies the packet-level arrival curve constraint \(\alpha_{\text{pkt}}\) if and only if for all indices \(m \leq n\):

\[
A_n - A_m \geq \alpha_{\text{pkt}} (n - m + 1)
\]

(19)

B. Re-sequencing Buffer Operation

A re-sequencing buffer stores the packets of a flow until the packets with smaller sequence numbers arrive; then it delivers them in the increasing order of their sequence numbers. A re-sequencing buffer has two parameters, a size in bytes, \(B\), and a timeout value, \(T\). It is described in terms of:

Algorithm 1 Packet arrival event code routine

Input: packet \(p\)

Shared variables: \(buf\) and \(N\)

1: if \(p.id \geq N\) then \(\triangleright \) if TRUE, \(p\) is a valid packet
2: if \(p.id > N\) then
3: \(\quad\) if \(buf\.len() + p\.len \leq B\) then
4: \(\quad\) TimerList.start\((p.id,\text{Time()} + T)\)
5: \(\quad\) \(buf\.enqueue(p)\)
6: \(\quad\) else \(\triangleright \) ERROR, OVERFLOW
7: \(\quad\) discard\((p)\)
8: \(\) end if
9: \(\) \(N \leftarrow p.id + 1\)
10: \(\) release\((p)\)
11: \(\) CHECK\_BUFFER()
12: end if
13: else \(\triangleright \) ERROR, INVALID PACKET
14: end if
15: function CHECK\_BUFFER(void)
16: \(\quad\) if \(buf\.contains(N)\) then
17: \(\quad\) \(p \leftarrow buf\.dequeue(N)\)
18: \(\quad\) \(N \leftarrow p.id + 1\)
19: \(\quad\) TimerList.stop\((p.id)\)
20: \(\quad\) release\((p)\)
21: \(\quad\) CHECK\_BUFFER()
22: end if
23: end function

Algorithm 2 Timeout event code routine

Input: packet id \(pid\)

Shared variables: \(buf\) and \(N\)

1: while \(N \leq pid\) do
2: \(\quad\) while \(!buf\.contains(N)\) do
3: \(\quad\) \(N \leftarrow N + 1\)
4: \(\quad\) end while
5: \(\quad\) \(p' \leftarrow buf\.dequeue(N)\)
6: \(\quad\) \(N \leftarrow p'.id + 1\)
7: \(\quad\) TimerList.stop\((p'.id)\)
8: \(\quad\) release\((p')\)
9: \(\) end while
10: CHECK\_BUFFER()

- Shared variables, that are manipulated by the code routines. These are 1) a list (buffer), \(buf\), containing the packets that are waiting for the packets with smaller sequence number; 2) an integer, \(N\), expressing the next sequence number that the buffer is expecting to receive
- Timers: The re-sequencing buffer sets a timer for each packet stored in the buffer. The object TimerList is the list of the timers for the packets. It has two functions, start\((\text{pid,deadline})\) and stop\((\text{pid})\). The former, starts a timer for the packet with sequence number \(\text{pid}\) with expiration time \(\text{deadline}\). The latter, stops the timer for the packet with sequence number \(\text{pid}\).
- Events, which trigger the execution of code routines. The events are packet arrival (Algorithm 1) and timeout (Algorithm 2). We assume that the execution of the code routines is serialized, namely, a code routine can start only after the code routine triggered by the previous event has completed (to avoid race conditions with shared variables).

When a new packet \(p\) arrives, the packet arrival code routine in Algorithm 1 is executed. If the packet sequence number \(p.id\) is smaller than \(N\), then the packet is considered invalid and is discarded (line 13). This is an error-case: packet \(N\) is expected, which means that packet \(N - 1\) was delivered. This packet is either a duplicate of packet \(N - 1\) or a packet with a smaller sequence number, and delivering it would violate in-order delivery.

If \(p.id > N\) (line 9), the packet should wait in the buffer for the packets with smaller sequence numbers to arrive. Then, it checks the current length of buffer, \(buf\.len()\); if addition of packet \(p\) with length of \(p\.len\) does not exceed the size of the buffer, \(B\), (line 5) a timer for this packet starts, which expires at time \(\text{Time()} + T\) (line 4); the function Time() returns the current time of the buffer. Then, it enqueues the packet in the buffer (line 5). Otherwise, if the buffer does not have enough capacity, buffer overflow occurs and the packet is discarded (line 6).

If \(p.id == N\) (line 8), the packet is the expected one. Then, \(N\) is incremented by 1 (line 9) and the packet is released (line 10). When packet \(p\) departs, the buffer should be checked to
release the packets that were only waiting for packet $p$; this is done by a recursive function CHECK_BUFFER() (line [15]). Accordingly, if the buffer contains a packet with a sequence number equal to the new value of $N$ (line [16]), the packet is dequeued from the buffer (line [17]). Then, $N$ is increased to the next sequence number of the flow; the corresponding timer is stopped; and then the packet is released (lines [18] to [20]). It recalls the functions recursively (line [21]) and the value of $N$ is increased every time the function executed; the recursion continues until the buffer does not have a packet with sequence numbers less than the value of $N$; all the packets with sequence numbers less than the value of $N$ are already released from the buffer.

When the timer for a packet with sequence number $pid$ expires, the timeout code routine in Algorithm 2 is executed. In this condition, the packet with sequence number $pid$ should be released. To provide in-order delivery, the buffer first releases all the packets in the buffer with sequence number less than or equal to $pid$ in increasing order. To do so, the loop at line [1] is executed to iterate the buffer for the packets with sequence number less than $pid$. The lines [2] to [4] increment $N$ to the smallest sequence number for a packet stored in the buffer; the function $buf.contains(N)$, returns TRUE if a packet with sequence number $N$ is in the buffer, otherwise, it returns FALSE. In line [5] the buffer contains a packet $p'$ with sequence number equal to $N$; then, it is dequeued from the buffer; the value of $N$ is incremented by 1; the timer for this packet is stopped; and it is released from the buffer (lines [5] to [8]). Whenever a packet is released from the buffer, its corresponding timer is stopped. The loop in line [1] is executed at least once, because when the timeout event occurs for a packet with sequence number $pid$, it is already in the buffer; this implies $N \leq p.id$. The loop in line [2] is executed at the latest when $N = pid$. In line [10] the buffer is checked to release the packets that were waiting for packets with id $pid$ and smaller; this is done by the function CHECK_BUFFER().

Observe that a packet is released if any one of the following conditions hold: 1) all packets with smaller sequence numbers are received, 2) its timer expires or 3) the timer of a received packet with a larger sequence number expires.

By construction, the re-sequencing buffer delivers the packets that it does not discard in increasing sequence numbers. Furthermore, a packet is discarded by the re-sequencing buffer either when the buffer is full or when the sequence number of the arriving packet is less than $N$. The latter occurs when the timeouts of packets are too early, compared to the lateness of misordered packets. Therefore, to avoid discarding packets, the re-sequencing buffer size and the timeout value should be large enough.

APPENDIX B

PROOFS

A. Proof of Lemma 7

Let $A_1 \leq A_2 \leq \ldots \leq A_n\ldots$ be the arrival times of packets at $S$; here the packet indices are in order of arrival (and are not necessarily equal to the sequence numbers). Let $D_n$ be the departure time of the packet with index $n$; the sequence $D_n$ need not be monotonic. Let $R, R'$ be the cumulative arrival and departure functions [37] defined by $R(t) = \sum_{n=1}^{+\infty} l_n 1_{A_n < t}$ and $R'(t) = \sum_{n=1}^{+\infty} l_n 1_{D_n < t}$. Here, $l_n$ is the length of packet $n$ and we allow $t \leq 0$ (in which case $R(t) = R'(t) = 0$). Let $\varphi(t)$ be the function $\mathbb{R} \to \mathbb{R}$ defined by $\varphi(t) = 0$ is $t \leq 0$ and $\varphi(t) = 1$ if $t > 0$, so that

$$R(t) = \sum_{n=1}^{+\infty} l_n \varphi(t - A_n), \quad R'(t) = \sum_{n=1}^{+\infty} l_n \varphi(t - D_n) \quad (20)$$

Let $0 \leq s \leq t$; the arrival curve constraint at the input means that $R(t) - R(s) \leq \alpha(t - s)$. This equation continues to hold if $s$ or $t$ is negative, with the convention that $\alpha(t) = 0$ whenever $t \leq 0$. Therefore

$$\forall s, t \in \mathbb{R}, \quad R(t) - R(s) \leq \alpha(t - s) \quad (21)$$

Furthermore,

$$R'(t) - R'(s) = \sum_{n=1}^{+\infty} l_n (\varphi(t - D_n) - \varphi(s - D_n)) \quad (22)$$

Let $d_{\text{min}}$ be the best-case delay of the flow, so that the worst-case delay is $\leq d_{\text{min}} + V$. For every packet index $n$, we have

$$A_n + d_{\text{min}} \leq D_n \leq A_n + d_{\text{min}} + V \quad (23)$$

thus

$$t - D_n \leq t - A_n - d_{\text{min}} \quad (24)$$

$$s - D_n \geq s - A_n - d_{\text{min}} - V \quad (25)$$

and, since $\varphi$ is wide-sense increasing

$$\varphi(t - D_n) - \varphi(s - D_n) \leq \varphi(t - A_n - d_{\text{min}}) - \varphi(s - A_n - d_{\text{min}} - V) \quad (26)$$

Combining with (22) and (21):

$$R'(t) - R'(s) \leq \sum_{n=1}^{+\infty} l_n (\varphi(t - A_n - d_{\text{min}}) - \varphi(s - A_n - d_{\text{min}} - V)) - (R(t - d_{\text{min}}) - R(s - d_{\text{min}} - V)) \leq \alpha(t - s + V) \quad (27)$$

\[ \square \]

B. Proof of Lemma 2

With the same notation as in the proof of Lemma 1, the backlog of this flow at time $t$ is $B(t) = R(t) - R'(t)$, so that

$$B(t) = \sum_{n=1}^{+\infty} l_n (\varphi(t - A_n) - \varphi(t - D_n)) \quad (28)$$

We have $t - D_n \geq t - A_n - U$ and, since $\varphi$ is wide-sense increasing:

$$B(t) \leq \sum_{n=1}^{+\infty} l_n (\varphi(t - A_n) - \varphi(t - A_n - U)) = R(t) - R(t - U) \leq \alpha(U) \quad (29)$$

where the last inequality is by (21). \[ \square \]
C. Proof of Theorem 7

(1) We prove (8) by induction on $n$.

Base case $n = 1$. According to the description of resequencing buffer in Section III, the initial value of next expected sequence number is $N = 1$; therefore, packet 1 is never stored in the buffer.

- If packet 1 arrives at some time $E_1 \neq +\infty$. Then, there are two cases for the timers at time $E_1$ of packets with index larger than 1:
  - No timer has expired at $E_1$. Then, $N = 1$ and packet 1 is released immediately after arrival ($D_1 = E_1$). Since no timer was expired, for any packet $p$ in the buffer, $p > 1$, we have $E_1 \leq E_p + T$. Thus, based on (9), $I_1 = E_1$ and finally $D_1 = I_1$ as required.
  - A timer has expired, say for packet index $p$, at $E_1$. Then, $N > 1$ and packet 1 is discarded ($D_1 = +\infty$). We have $E_1 > E_p + T$. (9) gives $I_1 = +\infty$ and finally $D_1 = I_1$ as required.

Induction step. Suppose that (8) holds for all $i$ with $i \leq n - 1$.

If packet $n$ arrives at some time $E_n \neq +\infty$. Then, there are two cases for the value of $N$:

- $N \leq n$. No timer for a packet $j$ in the buffer, $j > n$, has been expired (otherwise, the value of $N$ would be increased to the index of the packet with expired timer, i.e., $N > n$). Then, we have $E_n \leq E_j + T$, therefore, $E_n \leq T + \min_{j \geq n} \{E_j\}$. Then, based on (9), $I_n = E_n$. Now, the release time of packet $n$ depends on the status of packet $n-1$; there are two possible cases:
  * $N = n$. This implies that packet $n-1$ is already released, $D_{n-1} \leq E_n$. Therefore, packet $n$ is released immediately on arrival ($D_n = E_n$). Then, based on (8) and (9), we also have:
    \[ I_n = E_n, \]
    \[ G_n = \min \left\{ D_{n-1}, T + \min_{j \geq n} \{E_j\} \right\} \]
    \[ \leq D_{n-1} \leq E_n, \] (31)
    and then $\max(G_n, I_n) = E_n$, which shows that the right hand-side of (8) is $D_n$ as required.
  * $N < n$. This implies that packet $n-1$ is not yet released, $D_{n-1} > E_n$. Therefore, packet $n$ is stored in the buffer. Packet $n$ is released when:
    - The timer for a packet with sequence number larger than or equal to $n$ is expired before packet $n-1$ is released ($D_{n-1} \geq \min_{j \geq n} \{E_j\} + T$). Let us call $p$, where $p \geq n$, as the packet of which the timer is expired before the others ($T + \min_{j \geq n} \{E_j\} = T + E_p$). Then, any packet $j$ in the buffer $j \leq p$ are released in-order (Algorithm 2 line 8); therefore,
      \[ D_n = T + E_p = T + \min_{j \geq n} \{E_j\}. \]
      Then, based on (8) and (9), we also have:
      \[ I_n = E_n, \] (32)
      \[ G_n = \min \left\{ D_{n-1}, T + \min_{j \geq n} \{E_j\} \right\} \]
      \[ = T + \min_{j \geq n} \{E_j\}, \] (33)
      and then we have:
      \[ \max\{G_n, I_n\} = \max\{T + \min_{j \geq n} \{E_j\}, E_n\} = T + \min_{j \geq n} \{E_j\} = D_n, \] (34)
      which shows (9).
    - Or when packet $n-1$ is released before any timer of packets with sequence number larger than or equal to $n$ is expired ($D_{n-1} < T + \min_{j \geq n} \{E_j\}$). Then packet $n$ is released immediately after packet $n-1$ is released ($D_n = D_{n-1}$). Then, based on (8) and (9), we also have:
      \[ G_n = \min \left\{ D_{n-1}, T + \min_{j \geq n} \{E_j\} \right\} \]
      \[ = D_{n-1}. \] (35)
    Then,
    \[ \max\{G_n, I_n\} = \max\{G_n, E_n\} = D_{n-1} = D_n, \] (36)
    which shows (9).

(2) Consider Fig. 2 and a received packet $n$. Due to (3), for any packet $j \geq n$, we have $E_n \leq E_j + \lambda$; therefore, $E_n \leq \min_{j \geq n} \{E_j\} + \lambda$. Since $\lambda \leq T$:
    \[ E_n \leq \min_{j \geq n} \{E_j\} + T \implies I_n = E_n, \] (39)
    which proves (11).

(3) The proof of item (3) is by induction on $n \geq 1$.

Base case $n = 1$. Then, by (11), $D_1 = E_1$ and the statement is trivially proven.

Induction step. We assume that the statement holds for all packets $i$ with $i \leq n - 1$. Due to (3):

\[ \forall k < n, \forall p \geq n : \ E_k \leq E_p + \lambda \]
thus
\[ \forall k < n : \ E_k \leq T + \min_{j \geq n} \{E_j\}, \]
thus
\[ \max_{k \leq n-1} \{E_k\} \leq T + \min_{j \geq n} \{E_j\} \]
thus
\[ D_{n-1} \leq T + \min_{j \geq n} \{E_j\}. \] (40)
By (10) and (11):
\[ G_n = \min \left\{ D_{n-1}, T + \min_{j \geq n} \{ E_j \} \right\} = D_{n-1}, \quad (41) \]
\[ D_n = \max(G_n, E_n) = \max(D_{n-1}, E_n) \]
\[ = \max \left( \max_{k \leq n-1} \{ E_k \}, E_n \right) = \max_{k \leq n} \{ E_k \}. \quad (42) \]

D. Proof of Theorem 2

Consider Fig. 2. We consider the two possible cases for \( m \):

1. \( m = n - 1 \). Since \( n - 1 < n \), \( \min_{j \geq n-1} \{ E_j \} \leq \min_{j \geq n} \{ E_j \} \). Therefore, by (49), \( D_{n-1} \leq T + \min_{j \geq n} \{ E_j \} \). Now, using (44):
   \[ D_n = \max(G_n, E_n) \geq G_n = \min \left( D_{n-1}, T + \min_{j \geq n} \{ E_j \} \right) \]
   \[ = D_{n-1}. \]

2. \( m < n - 1 \). Then \( E_{n-1} = +\infty \) and in turn, \( D_{n-1} = \max \{ G_{n-1}, E_{n-1} \} = +\infty \). By (44), we have:
   \[ D_n = \max(G_n, E_n) \geq G_n = \min \left( D_{n-1}, T + \min_{j \geq n} \{ E_j \} \right) \]
   \[ = T + \min_{j \geq n} \{ E_j \}. \]

Since \( m < n - 1 \), \( \min_{j \geq m} \{ E_j \} \leq \min_{j \geq n} \{ E_j \} \). Therefore using (49) and (51), we have:
   \[ D_n \geq T + \min_{j \geq n} \{ E_j \} \geq T + \min_{j \geq m} \{ E_j \} \geq D_m. \]

This establishes (43).

Second, we show that if for any \( \lambda > 0 \), if \( T < \lambda \), there exists a scenario with RTO \( \lambda \) where the re-sequencing buffer discards a packet. Consider a trace with two packets 1 and 2, received at the re-sequence buffer at a times \( E_2 = t_0 \) and \( E_1 = t_0 + \lambda \) for some \( t_0 \geq 0 \). The RTO of this trace is \( \lambda \). By Theorem 1, \( D_1 = 1 \) and \( 1 = +\infty \) because \( E_1 > \min_{j \geq n} \{ E_j \} + T = E_2 + T \). Thus packet 1 is discarded by the re-sequencing buffer.

E. Proof of Theorem 3

**Item (1).** Consider Fig. 2. Assume the size of the re-sequence buffer is unlimited; the actual buffer content at time \( t \) is
\[ L(t) = \sum_{E_k < t, D_k \geq t} l_k, \quad (53) \]
First, we show that \( L(t) \leq \pi \) at all times \( t \) that immediately follow a packet arrival; this will imply that a buffer of size \( \pi \) is sufficient to avoid overflow.

Packet 1 is never stored in the buffer. Consider some fixed but arbitrary packet \( n > 1 \), with size \( l_n \), and define the set of indices \( \mathcal{X} \) by
\[ \mathcal{X} := \{ i \in \mathbb{Z}^+ \mid i < n, D_i \geq E_n \} \quad (54) \]
If \( \mathcal{X} \) is empty then \( D_{n-1} < E_n \); observe that \( D_n = \max(E_1, \ldots, E_n) = \max(D_{n-1}, E_n) = E_n \), i.e. packet \( n \) is not stored in the buffer, and therefore buffer overflow does not occur when packet \( n \) arrives. Hence, we assume that \( \mathcal{X} \) is not empty and let \( m = \min(\mathcal{X}) \). The actual content of the buffer just after the arrival of packet \( n \) is
\[ L(E_n) + l_n = \sum_{k, E_k < E_n \leq D_k} l_k \]
\[ = \sum_{k, E_k < E_n \leq D_k} l_k + l_n \]
\[ = \sum_{k \geq m, E_k < E_n \leq D_k} l_k + l_n \]
\[ = \sum_{k \geq m, E_k < E_n \leq D_k} l_k + l_n, \]
\[ \text{Eq. 48} \]
\[ D_m \leq T + \min_{j \geq m} \{ E_j \}. \]
\[ \text{Eq. 49} \]
We consider the two possible cases for \( m \): 1) \( m = n - 1 \), 2) \( m < n - 1 \).
where the last equality is because the first sum in $\sum_{k=1}^{\alpha} n_k$ is 0 by definition of $m$. By Lemma 3, $E_m \geq E_n$; since $m \neq n$ and we exclude simultaneous packet arrivals at the re-sequencing buffer, $E_m > E_n$. Thus

$$\left[(k \geq m \text{ and } E_k < E_n) \text{ or } (k = n)\right]$$

$$\implies (k > m \text{ and } E_k < E_m), \quad (58)$$

therefore

$$\sum_{k \geq m, E_k < E_n} l_k + l_n = \sum_{k, (k \geq m, E_k < E_n), \alpha (k = n)} l_k \leq \sum_{k > m, E_k < E_m} l_k = \pi_m. \quad (59)$$

Combined with (55)-(57), this shows that

$$L(E_n) + l_n \leq \pi_m \leq \pi \quad (60)$$

**Lemma 3.** $E_m \geq E_n$.

**Proof.** By construction, $m \in *$ thus $D_m \geq E_n$. Since $D_m = \max_{k \leq m} E_k$, it follows that

$$\exists k \in \{1...m\} \text{ such that } E_k \geq E_n \quad (61)$$

If $m = 1$ the conclusion follows. Else, $m - 1$ is not in $*$ thus $D_{m-1} < E_n$. Since $D_{m-1} = \max_{k \leq m-1} E_k$, it follows that $\forall k \in \{1...m - 1\}$, $E_k < E_n$. Combined with (61), this shows that $E_m \geq E_n$. $lacksquare$

Second, we show that for any possible $\lambda > 0$ and valid RBO value $\pi$ there exists one execution trace of a flow with packet sizes between $L^\min$ and $L^\max$, with RTO $\lambda$ and RBO $\pi$, that achieves a buffer content equal to $\pi$. First observe that, by definition, $\pi$ can be written as $\pi = \sum_{j=1}^{k} \ell_j$ for some positive integer $k$ and $\ell_j \in [L^\min, L^\max]$. The packet sequence is as follows. It has $k+1$ packets in total. Packet 1 has some arbitrary size $l_1 \in [L^\min, L^\max]$ and is observed at time $E_1 = \lambda$. Packets 2 to $k+1$ have sizes $l_2 = \ell_1, ..., l_{k+1} = \ell_k$ and are observed at times $E_j = \frac{(j-1)\lambda}{1+k}$. Packets 2 to $k+1$ arrive before packet 1, are stored in the buffer until packet 1 arrives, and the buffer content when packet 1 arrives is $E_j$. The RTOs are $\lambda_1 = \lambda, \lambda_2 = ... = \lambda_{k+1} = 0$ and the RTO of the trace is $\lambda$. The RBOs are $\pi_1 = \sum_{j=1}^{1} \ell_j = \pi, \pi_2 = ... = \pi_{k+1} = 0$ thus the RBO of the sequence is $\pi$.

**Item (2).** First we show that the buffer size is upper bounded by $\alpha(T + V)$. Since delay jitter from the source to the input of the re-sequencing buffer is $V$, by Lemma 1, the flow has arrival curve $\alpha'(t) = \alpha(t + V)$ at the input of re-sequencing buffer. Also, by Theorem 2 (the proof of which is independent of this result) the delay at the re-sequencing buffer is upper bounded by the time-out $T$; by Lemma 2, the amount of backlog inside the buffer is thus upper bounded by $\alpha'(T) = \alpha(V + T)$.

Fix some $\varepsilon > 0$, smaller than $V$ and $T$. By the second technical assumption at the end of Section 3, there exists an integer $n$ and a sequence of packet lengths $\ell_k \in [L^\min, L^\max]$ such that $\alpha(V + T - \varepsilon) = \sum_{k=1}^{n} \ell_k$. Since the arrival curve is achievable, there also exists a sequence of emission times $t_1 = 0, ..., t_n = V + T - \varepsilon$ such that the packet sequence $t_1, t_2, ..., t_n$ satisfies the arrival curve constraint $\alpha$. We now derive another packet sequence of $n + 1$ packets as follows.

1) Packet 1 is emitted at time $A_1 = 0$ and has size $l_1 = l^\max$.
2) For $k = 2...n + 1$, packet $k$ is emitted at time $A_k = t_0 + t_{k-1}$ and has size $l_k = \ell_{k-1}$, where $t_0$ is a positive number, large enough so that $\alpha(t_0) \geq \sum_{k=1}^{n+1} \ell_k$. Such a number exists because we assume $\lim_{t \to \infty} \alpha(t) = +\infty$.

We have thus $A_{n+1} - A_2 = V + \varepsilon$. The arrival times of the $n + 1$ packets to the input of re-sequencing buffer are as follows.

1) Packet 1 is lost, i.e. $E_1 = +\infty$.
2) Packet $k = 2$ arrives at time $E_2 = V + A_2 - \frac{\varepsilon}{T}$. For $k \geq 2$, for $k = 3...n + 1$, packet $k$ arrives at time $E_k = \max(E_{k-1}, A_k) + \frac{\varepsilon}{3(n-1)}$.

We now verify that our scenario satisfies all constraints. There is no simultaneous arrival at the re-sequencing buffer as required by our modelling assumptions. Obviously $E_k \geq A_k$ (the scenario is causal) and $E_k > E_{k-1}$ for $k \geq 3$ therefore there is no reordering, and any RTO or RBO constraint is satisfied.

We now verify that the jitter is $\leq V$. We first show by induction on $k \geq 2, n \geq 2$ that

$$E_k - A_k \leq V - \frac{\varepsilon(k - 2)}{3(n-1)} \quad (62)$$

For $k = 2$ it follows from the definition of $E_2$. Consider now $k \geq 3$ and assume it holds for $k - 1$. Then, by the induction hypothesis:

$$E_{k-1} \leq A_{k-1} + V - \frac{\varepsilon(k - 3)}{3(n-1)}$$

$$\leq A_k + V - \frac{\varepsilon(k - 3)}{3(n-1)}$$

thus

$$E_{k-1} + \frac{\varepsilon}{3(n-1)} \leq A_k + V - \frac{\varepsilon(k - 2)}{3(n-1)} \quad (63).$$

Also, as $V > \varepsilon$:

$$A_k + \frac{\varepsilon}{3(n-1)} \leq A_k + V - \frac{\varepsilon(k - 2)}{3(n-1)}. \quad (64)$$

Then, (63) and (64) give:

$$E_k = \max(E_{k-1}, A_k) + \frac{\varepsilon}{3(n-1)} \leq A_k + V - \frac{\varepsilon(k - 2)}{3(n-1)} \quad (65),$$

as required. It follows from (62) that the jitter of the trace is less than or equal to $V$.

Next, packet 2 arrives at time $V + A_2 - \frac{\varepsilon}{T}$ and is out of order (due to the loss of packet 1), which triggers a timeout at time $T + V + A_2 - \frac{\varepsilon}{T}$. We now verify that all packets $k \geq 3$ arrive before $V + T + A_2 - \frac{\varepsilon}{T}$. To this end, we show by induction on $k \geq 2$ that

$$E_k \leq T + V + A_2 - \frac{\varepsilon(2n - k)}{3(n-1)}. \quad (66)$$
For \( k = 2 \), it follows from the definition of \( E_2 \) and \( T > \varepsilon \). Assume it holds for \( k - 1 \). Then, by the induction hypothesis

\[
E_{k-1} \leq T + V + A_2 - \frac{\varepsilon(2n - k + 1)}{3(n-1)}
\]

thus

\[
E_k - \frac{\varepsilon}{3(n-1)} \leq T + V + A_2 - \frac{\varepsilon(2n - k)}{3(n-1)}.
\]

(67)

Also \( A_k \leq A_{n+1} = T + V + A_2 - \varepsilon \), therefore,

\[
A_k + \frac{\varepsilon}{3(n-1)} \leq T + V + A_2 - \frac{\varepsilon(3n - 2)}{3(n-1)}
\]

\[
\leq T + V + A_2 - \frac{\varepsilon(2n - k)}{3(n-1)},
\]

(68)

hence,

\[
E_k \leq T + V + A_2 - \frac{\varepsilon(2n - k)}{3(n-1)},
\]

(69)

as required.

Now \( \frac{\varepsilon(2n - k)}{3(n-1)} > \frac{\varepsilon}{3} \) for \( k = 3 \ldots n + 1 \) thus every packet other than 2 arrives before packet 2 times out. Thus the buffer content just after the arrival of packet \( n + 1 \) is all packets 2 to \( n + 1 \), i.e. its size is \( \alpha(V + T - \varepsilon) = \sum_{k=1}^{n} \ell_k \).

It remains to verify that the trace satisfies the arrival curve constraint. Let \( R(t) \) be the cumulative arrival function of the trace, i.e. \( R(t) = \sum_{s=1}^{t} 1_{A_s < t} \). First, the sequence of packets 2 to \( n + 1 \) is obtained by time-shifting by \( t_0 \) a sequence that satisfies the arrival curve constraint, therefore it also does, namely, \( R(t) - R(s) \leq \alpha(t-s) \) whenever \( s \leq t, s \geq t_0 \) and \( t \geq t_0 \). It remains to see the other cases:

- 0 < \( s \leq t < t_0 \): then \( R(t) - R(s) = 0 \leq \alpha(t-s) \)
- 0 < \( s < t < t_0 \): then \( R(t) - R(s) = R(t) - R(0) = l_1 \leq \alpha(0+0) \leq \alpha(t-s) \)
- 0 < \( s < t_0 \leq t \): then \( R(t) - R(s) = R(t) - R(t_0) \leq \alpha(t-t_0) \leq \alpha(t-s) \)
- 0 < \( s < t_0 \leq t \): then \( R(t) - R(s) = R(t) - R(0) = R(t) - R(t_0) + l_1 \leq \sum_{k=1}^{n} \ell_k \leq \alpha(t_0) \) by construction of \( t_0 \). Thus \( R(t) - R(t_0) \leq \alpha(t_0) \leq \alpha(t) \).

This shows that the arrival curve constraint is satisfied.

At this stage, we have shown that, for every \( \varepsilon \) small enough, there is a scenario where the backlog reaches \( \alpha(V + T - \varepsilon) \). Thus the minimal bound is at least \( \sup_{\varepsilon > 0} \alpha(V + T - \varepsilon) = \alpha(V + T) \) because arrival curves are left-continuous.

\textbf{F. Proof of Theorem 7}

\textbf{Proof.} We use the notations in Fig. 2. Suppose that the system has \( \delta_{\max} \) and \( \delta_{\min} \) as worst-case and best-case delays, thus, for any packet \( i \), \( \delta_{\min} \leq E_i - A_i \leq \delta_{\max} \). Now, consider a received packet \( n \). Since \( D_n \geq E_n \), \( D_n - A_n \geq E_n - A_n \geq \delta_{\min} \); this shows that the best-case delay is not decreased.

Part (1). The system is lossless, therefore by part (3) of Theorem 1, \( D_n = \max_{i \leq n} \{ E_i \} \). Therefore:

\[
D_n - A_n = \max_{i \leq n} \{ E_i - A_i \} = \max_{i \leq n} \{ E_i - A_n \}.
\]

(70)

Since \( \forall i \in \mathbb{Z}^+ \), \( A_i \leq A_{i+1} \):

\[
D_n - A_n \leq \max_{i \leq n} \{ E_i - A_i \} \leq \delta_{\max}.
\]

(71)

which proves that the worst-case delay is not increased. Since the best case delay is not decreased, the delay jitter is not increased.

Part (2). The system is not lossless. By part (2) of Theorem 1 we have:

\[
G_n = \min \left( D_{n-1}, T + \min_{j \geq n} E_j \right) \leq T + \min_{j \geq n} E_j \leq T + E_n,
\]

\[
D_n = \max(G_n, E_n) = \max(T + E_n, E_n) = T + E_n.
\]

(72)

Thus

\[
D_n - A_n \leq T + E_n - A_n \leq T + \delta_{\max},
\]

(73)

that proves part (2).

\textbf{G. Proof of Theorem 5}

\textbf{Proof.} First, we obtain an upper bound for RTO of the flow separately for each part of the theorem. Second we show that each bound is achievable.

Consider a packet \( n \). Let us denote \( A_n \) as the arrival time of a packet \( n \) (with size \( l_n \)) into the system and \( E_n \) as its exit time. Now, consider another packet \( m \) such that \( m \leq n - 1 \) and \( E_n < E_m \). Since \( V \) is a jitter bound for this system,

\[
(E_m - A_m) - (E_n - A_n) \leq V.
\]

(74)

Then:

\[
E_m - E_n \leq V - (A_n - A_m).
\]

(75)

If the flow has arrival curve \( \alpha \), then by (18),

\[
A_n - A_m \geq \alpha^+ \left( \sum_{k=m}^{n} l_k \right) \leq \alpha^+ \left( \sum_{k=m}^{n} l_k \right).
\]

(76)

Since \( \alpha^+ \) is wide-sense increasing and \( m \leq n - 1 \):

\[
E_m - E_n \leq V - \alpha^+ (l_{n-1} + l_n) \leq V - \alpha^+(2L_{\min}) \leq \left[ V - \alpha^+(2L_{\min}) \right]^+,
\]

(77)

that proves item (1) in the statement of theorem.

If the flow has packet-level arrival curve \( \alpha_{pk} \), then by (19),

\[
A_n - A_m \geq \alpha_{pk}^+ (n - m + 1) \leq \alpha_{pk}^+ (n - m + 1) \leq \alpha_{pk}^+ (n - m + 1) \leq \alpha_{pk}^+ (n - m + 1).
\]

(78)

Since \( n \) is integer and \( n > m \), then \( n - m \geq 1 \):

\[
E_m - E_n \leq V - \alpha_{pk}^+ (2) \leq \left[ V - \alpha_{pk}^+(2) \right]^+.
\]

(79)

which proves item (2) in the statement of theorem.

Second, we show that the bounds are achievable by constructing a scenario where the RTO for a packet of the flow reaches the bound in item (1) of the theorem.

Consider two packets 1 and 2 with sizes \( l_1 = l_2 = L_{\min} \) and a non order-preserving system with a jitter bound \( V \). Packet 1 is issued at \( A_1 = 0 \) and packet 2 arrives at \( A_2 = A_1 + \alpha^+(l_1 + l_2) \), i.e. \( (A_2 = t_2) \).
Packet 1 experiences a delay of $d + V$, $E_1 = A_1 + d + V$, and Packet 2 experiences a delay of $d$, $E_2 = A_2 + d$. Then we have:

$$E_1 - E_2 = A_1 + d + V - (A_2 + d) = V - (t_2 - t_1) = V - \alpha^i(l_1 + l_2) = V - \alpha^i(2L_{\text{min}}), \quad (80)$$

which shows that the RTO for packet 1 is equal to the bound in part (1) of the theorem.

Now, we verify that jitter bound and arrival curve assumptions are not violated. The difference between the delay of two packets is:

$$(E_1 - A_1) - (E_2 - A_2) = (d + V) - (d) = V, \quad (81)$$

which is equal to the jitter bound. Also:

$$A_2 - A_1 = t_2 - t_1 = \alpha^i(l_1 + l_2), \quad (82)$$

which shows the arrival curve constraint holds.

The tightness scenario for item (2) of the theorem is similar to the one for part (1). We set $t_2 = t_1 + \alpha_{\text{pkt}}^i(2)$, and the rest follows the description of item (1).

\[H. \ Proof \ of \ Theorem \ 6\]

Proof. First, we show the bounds in items (1) and (2).

Item (1). First, observe that, from Theorem 5 if $\alpha(V) < 2L_{\text{min}}$ then there is no reordering and the RBO is 0.

Next, assume that there is some reordering and consider a packet index $m$ such that $\lambda_m > 0$. Let $E_m = \{i \in \mathbb{Z}^+ | i > m, E_i < E_m\}$. Since $\lambda_m > 0$ and there is no simultaneous arrival of packets, $E_m$ is not empty. Then let $n = \max\{E_m\}$.

Due to the jitter bound of this system and since $E_n < E_m$, we have:

$$(E_m - A_m) - (E_n - A_n) \leq V,$$

thus

$$A_n - A_m \leq V + (E_n - E_m) < V. \quad (83)$$

Since the flow has arrival curve $\alpha$, then by (18),

$$A_n - A_m \geq \alpha^i(\sum_{k=m}^{n} l_k). \quad \text{Therefore,}$$

$$\alpha^i(\sum_{k=m}^{n} l_k) \leq A_n - A_m < V. \quad (84)$$

By (7), we obtain:

$$\sum_{k=m}^{n} l_k \leq \alpha(V). \quad (85)$$

We exclude packet $m$ from the left side of the above equation:

$$\sum_{k=m+1}^{n} l_k \leq \alpha(V) - l_m \leq \alpha(V) - L_{\text{min}}. \quad (86)$$

The reordering byte offset $\pi_m$, for packet $m$, defined in (4), includes only the packets with larger index and smaller exit time than packet $m$. Thus:

$$\pi_m = \sum_{k|k > m, E_k < E_m} l_k \leq \sum_{k=m+1}^{n} l_k. \quad (87)$$

Using (87) in (86), we have:

$$\pi_m \leq \alpha(V) - L_{\text{min}}, \quad (88)$$

which proves item (1) of the theorem.

Item (2). The flow has packet-level arrival curve $\alpha_{\text{pkt}}$. Observe that, from Theorem 5 if $\alpha_{\text{pkt}}(V) < 2$ then there is no reordering and the RBO is 0.

Next, since the flow has packet-level arrival curve $\alpha_{\text{pkt}}$, then by (19), $A_n - A_m \geq \alpha_{\text{pkt}}^i(n - m + 1)$. From (83), we have:

$$\alpha_{\text{pkt}}^i(n - m + 1) \leq A_n - A_m < V. \quad (89)$$

By (7), we obtain:

$$n - m + 1 \leq \alpha_{\text{pkt}}(V). \quad (90)$$

Since for any packet $k$, $l_k \leq L_{\text{max}}$:

$$\sum_{k=m+1}^{n} l_k \leq L_{\text{max}}(n - m) \leq L_{\text{max}}(\alpha_{\text{pkt}}(V) - 1). \quad (91)$$

Since $\pi_m \leq \sum_{k=m+1}^{n} l_k$, item (2) of the theorem is proven.

Second, we show the tightness.

Fix some $\varepsilon > 0$, smaller than $\lambda$. By the second technical assumption at the end of Section III-E, we show the tightness for the two cases, i) $L_{\text{max}} \geq 2L_{\text{min}}$, ii) $L_{\text{min}} \geq L_{\text{max}}$.

Case i) $L_{\text{max}} \geq 2L_{\text{min}}$. By assumption, we know that $\alpha(0^+) \geq L_{\text{max}}$; therefore, $\alpha(0^+) \geq 2L_{\text{min}}$.

By the second technical assumption at the end of Section III-E, there exists an integer $n$ and a sequence of packet lengths $l_k \in [L_{\text{min}}, L_{\text{max}}]$ such that $l_1 = L_{\text{min}}$ and $\sum_{k=2}^{n} l_k = \alpha(V - \varepsilon) - L_{\text{min}}$. Since $\varepsilon < \lambda$ and by Theorem 5, $\lambda \leq V$, $\alpha(V - \varepsilon) \geq 2L_{\text{min}}$; therefore, $n \geq 2$. Now, since the arrival curve is achievable, there also exists a sequence of emission times $A_1, ..., A_n = V - \varepsilon$ such that the packet sequence $A_1, ..., A_n, l_1, ..., l_n$ satisfies the arrival curve constraint $\alpha$.

Next, we construct the exit times of packets $k$ from the system as follows:

$$E_1 = V + \varepsilon, \quad E_k = V + \frac{(k - 2)\varepsilon}{n}, \quad k = 2, \ldots, n. \quad (92)$$

Observe that $E_2 < E_3 < \cdots < E_n < E_1$. Also note that $A_1 \leq A_2 \leq \cdots \leq A_n = V - \varepsilon < E_2 = V$.

Now, according to (7), the RBOs for packet 1 and packet $k$, $k = 2, \ldots, n$, are:

$$\pi_1 = \sum_{j|j > 1, E_j < E_1} l_j = \sum_{k=2}^{n} l_k = \alpha(V - \varepsilon) - L_{\text{min}},$$

$$\pi_k = \sum_{j|j > k, E_j < E_k} l_j = 0. \quad (93)$$

Therefore, $\pi = \max_{1 \leq i \leq n} \{\pi_i\} = \alpha(V - \varepsilon) - L_{\text{min}}$.

Finally, we verify that the assumptions are not violated: 1) arrival curve, 2) jitter bound, 3) RTO of the flow.

(1) The arrival curve constraint is satisfied by construction.

(2) For any packet $k \geq 1$, we have:

$$E_k - A_k \leq E_1 - A_1 = V + \varepsilon, \quad (94)$$

$$E_k - A_k \geq E_2 - A_n = V - (V - \varepsilon) = \varepsilon. \quad (95)$$
Therefore, the jitter is:
\[ \max \{ E_k - A_k \} - \min \{ E_k - A_k \} \leq V + \varepsilon - \varepsilon = V, \]  
which conforms the jitter constraint.

(3) For any packet \( k \geq 2 \), the packets are in order
\[ \lambda_k = E_k - \min_{j \mid j > k, E_{k} \leq E_{j}} E_k - E_{k} = 0. \]  
For packet 1, we have:
\[ \lambda_1 = E_1 - \min_{j \mid j > 1, E_{1} \leq E_{j}} E_1 - E_2 = (V + \varepsilon) - V = \varepsilon, \]
therefore, \( \lambda = \max_{1 \leq i \leq n} \{ \lambda_i \} = \varepsilon \), that satisfies the RTO constraint.

Thus, we have shown that, for every \( \varepsilon \) small enough, there is a scenario where the RBO reaches \( \alpha(V - \varepsilon) - L^{\min} \). Thus the minimal bound is at least \( \sup_{l > 0} \alpha(V - \varepsilon) - L^{\min} = \alpha(V) - L^{\min} \) because arrival curves are left-continuous.

By the second technical assumption at the end of Section II-E, there exists an integer \( n \) and a sequence of packets with length \( l \) such that \( \sum_{k=1}^{n} l_k = \alpha(V - \varepsilon) - l \). Now, if \( \alpha(V - \varepsilon) < 2l \), then \( n = 1 \) and therefore no reordering occurs and \( \pi = 0 \). Hence, we consider the case \( \alpha(V - \varepsilon) \geq 2l \); then \( n \geq 2 \). Now, since the arrival curve is achievable, there also exists a sequence of emission times \( t_1 = 0, \ldots, t_n = V - \varepsilon \) such that the packet sequence \( 1, \ldots, n \) satisfies the arrival curve constraint \( \alpha \). The rest of the proof follows exactly as case (i).

The tightness scenario for item (2) of the theorem is similar to the one for case (ii). We set \( \alpha = \max(V) \) and for any \( k = 1, \ldots, n \), \( l_k = L^{\max} \).

I. Proof of Theorem

Fig. 8: Notation for the sequence of network elements used in Theorem

**Proof.** First, we obtain an upper bound for RTO of the flow separately for each part of the theorem. Second we show that each bound is achievable.

Consider Fig. 8. We denote the arrival and exit times of a packet \( i \) at \( S_h \) by \( E_i^h \) and \( E_i^h \). Consider two packet indices \( m \) and \( n \) such that \( m < n \) and \( E_n^h < E_m^h \). Since \( S_s \) is the first system with nonzero RTO, for any system \( S_h \), \( h < s \), \( E_m^h < E_n^h \). Therefore, \( E_{m}^{h} - E_{n}^{h} \leq E_{n}^{h} - E_{m}^{h} \), i.e., at the output of system \( S_{s-1} \) (input of system \( S_s \)), packet \( m \) and \( n \) are in-order. Therefore, by definition of the RTO bound at \( S_s \),
\[ E_m^s - E_n^s \leq \lambda_s, \]
Now, according to the jitter bound for the concatenation of systems \( S_{s+1} \) to \( S_k \), we have:
\[ (E_m^k - E_m^s) - (E_n^k - E_n^s) \leq \sum_{h=s+1}^{K} V_h. \]
Then, we have:
\[ E_m^k - E_n^k \leq (E_n^s - E_m^s) + \sum_{h=s+1}^{K} V_h. \]

Combining with (99):
\[ E_m^k - E_n^k \leq \lambda_s + \sum_{h=s+1}^{K} V_h := \Lambda(K). \]

Second, we show tightness. We are given a sequence of systems with RTOs \( \lambda_h \) and jitters \( V_h \), and we construct a scenario that conforms with these parameters and where a packet reaches the RTO bound in Theorem 2 at system \( K \). We use the same notation as before. In particular, \( S_s \) is the first system in the sequence for which \( \lambda_s > 0 \). Now, consider a trace with two packets 1 and 2 entering \( S_1 \). Also, consider a set of positive values \( \{ d_1, d_1 + 1, \ldots, d_k \} \).

Packet 1 and 2 arrive at \( S_1 \) at times \( E_1 = 0 \) and \( E_2 = \varepsilon \), with \( \lambda_s > \varepsilon > 0 \). Each packet then has the same transfer time through system \( S_1 \) at time \( t_j \), \( j \in \{1, 2, \ldots, s - 1\} \), thus preserving order and \( E_{n}^{h} - E_{1}^{h} = \varepsilon + \varepsilon \).

The transfer times through \( S_s \) are \( d_s + \lambda_s \) for packet 1 and \( d_s \) for packet 2, i.e., \( E_1^s = E_1^{s-1} + d_s + \lambda_s \) and \( E_2^s = E_2^{s-1} + d_s \). Packet 1 and 2 experience the delays of \( d_h + V_h \) and \( d_h \) at system \( S_h \), \( h \in \{s + 1, s + 2, \ldots, K\} \):
\[ E_1^h = E_1^{h-1} + d_h + V_h, \]
\[ E_2^h = E_2^{h-1} + d_h. \]

Then at the output of system \( S_K \), we have:
\[ E_1^K = E_1^s + \sum_{h=s+1}^{K} (d_h + V_h) = E_1^{s-1} + \lambda_s + \sum_{h=s}^{K} d_h + \sum_{h=s+1}^{K} V_h, \]
\[ E_2^K = E_2^s + \sum_{h=s+1}^{K} d_h = E_2^{s-1} + \varepsilon + \sum_{h=s}^{K} d_h. \]

We now verify that the assumptions in the statement of theorem are not violated.

(1) We check that RTOs for all the systems are not violated. For any system \( S_j \), \( j \leq s \), packets 1 and 2 preserve order by construction, i.e., \( \lambda_j = 0 \).

For system \( S_s \), according to the departure times of packets 1 and 2 from \( S_s \), we have:
\[ E_1^s - E_2^s = \lambda_s - \varepsilon, \]
which satisfies the constraint \( \lambda_s \) on RTO conforms to RTO assumption for system \( S_s \) being the first system that \( \lambda_s \) is equal to \( \lambda_s \). At the output of system \( S_s \) we have \( E_2^s < E_2^s \).
i.e., packet 2 is prior to packet 1. For system $S_h$, $h \in \{s+1, s+2, \ldots, K\}$, we have
\[
E_2^h - E_1^h = \left( E_1^{s-1} + \varepsilon + \sum_{j=s}^h d_j \right) - \left( E_1^{s-1} + \lambda_s + \sum_{j=s}^h d_j + \sum_{j=s+1}^h V_j \right) = \varepsilon - \lambda_s - \sum_{j=s+1}^h V_j < 0, \tag{106}\]
that shows $E_2^h < E_1^h$, i.e., system $S_h$, $h \geq s+1$ preserves the order of its input. Therefore, RTO for each system $S_h$ is 0, which satisfies any RTO constraint.

(2) We check that the jitter bounds are not violated.

For systems $S_1$ to $S_{s-1}$, since both packets experience the same delay, the jitter is 0.

For system $S_s$, we have:
\[
(E_1^s - E_1^{s-1}) - (E_2^s - E_2^{s-1}) = \lambda_s. \tag{107}\]

Now, by Theorem 5, $\lambda_s \leq V_s$, thus the jitter bound assumption for $S_s$ is satisfied.

For any system $S_h$, where $h \in \{s+1, \ldots, K\}$, we have:
\[
(E_1^h - E_1^{h-1}) - (E_2^h - E_2^{h-1}) = (d_h + V_h) - (d_h) = V_h, \tag{111}\]
which shows that the jitter bound assumption for systems $S_{s+1}, S_{s+2}, \ldots, S_K$ are satisfied.

Last,
\[
E_1^K - E_2^K = \lambda_s - \varepsilon + \sum_{h=s+1}^K V_h, \tag{111}\]
thus the RTO for packet 1 is equal to the bound in Theorem 7 minus $\varepsilon$. Since $\varepsilon$ can be arbitrarily small, this shows the result.

\section*{Details of Computations for Case Study 1 in Section VI-B}

TABLE III: Arrival curve propagation from point 1 to point 9 under lossless network condition. Arrival curve at point $i$ is $\alpha_i(t) = \min(6.4e3 \ t + b_i, 125e6 \ t + M_i)$, where $b_i$ and $M_i$ are in bytes and shown in the table. Since under lossless network condition, the re-sequencing buffers do not increase delay and jitter bounds, the arrival curves are the same for the four placement strategies.

| Burst | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   |
|-------|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| $b_i$ | 6400| 6400| 6400| 6400| 6400| 6400|     |     |     |
| $M_i$ | 64  | 251 | 1751| 64  | 251 | 1751| 64  |     |     |

In these scenarios, we require arrival curve information to compute RTO, RBO, and delay upper bounds. Let us call $\alpha_i$ as arrival curve of flow $f$ at point $i$ in units of bytes. Tables III and IV show the propagated arrival curve at points 1 to 9 in Fig. 5. The arrival curve at point $i$ has the form of $\alpha_i(t) = \min(r \ t + b_i, c \ t + M_i)$, where $r = 6400$ bytes per second, $c = 125e5$ bytes per second (i.e. 1Gbps), $b_i$ and $M_i$ are shown in Tables III and IV. $\alpha_0(t) = r \ t + b_0$, where $b_0 = 6400$ bytes. The arrival curves at point 1, 5, and 9 capture the line shaping and packetizer effects:

\[
\alpha_1(t) = \min(r \ t + b_0, c \ t + 512), \tag{108}\]
\[
\alpha_5(t) = \min(r \ t + b_4, c \ t + 512), \tag{109}\]
\[
\alpha_9(t) = \min(r \ t + b_8, c \ t + 512). \tag{109}\]

The arrival curves at points 4 and 8 capture the effect of traversing the output FIFO systems with service curve $\beta(t) = 125e6[t - Q]^+$ bytes, where $Q = 12\mu s$:

\[
\alpha_4(t) = \min(r \ t + b_3 + rQ, c \ t + M_3 + cQ), \tag{110}\]
\[
\alpha_8(t) = \min(r \ t + b_7 + rQ, c \ t + M_7 + cQ). \tag{110}\]

The arrival curves at point 2 and 6 capture the effect of switching fabric with jitters bound $V_{sf}$:

\[
\alpha_2(t) = \min(r \ t + b_1 + rV_{sf}, c \ t + M_1 + cV_{sf}), \tag{111}\]
\[
\alpha_6(t) = \min(r \ t + b_5 + rV_{sf}, c \ t + M_5 + cV_{sf}). \tag{111}\]

Under lossless network condition, the re-sequencing buffers do not increase delay and jitter bounds, hence $\alpha_3 = \alpha_2$ and $\alpha_7 = \alpha_6$.

To compute delay bounds of output FIFO systems we use the results in [16, 58]. Accordingly, the delay bound of FIFO system of $h_1$ is $\delta_{\text{max, FIFO}, h_1} = 63.2\mu s$. Also, the minimum delay for each FIFO system is the transmission of a packet with minimum length, $\delta_{\text{min, FIFO}, h_1} = \delta_{\text{min, FIFO}, S_1} = \delta_{\text{min, FIFO}, S_2} = L_{\text{min}} / c = 512\mu s$. Now, since we know delay upper and lower bounds for the FIFO system of $h_1$, its jitter is $V_{\text{FIFO}, h_1} = 62.69\mu s$. Now we analyze the four strategies separately.

1) Re-sequencing only at $h_2$: We first obtain delay and jitter bounds of flow $f$ for the FIFO output ports of $S_1$ and $S_2$. Using the arrival curves in Tables III and IV.

\[
\delta_{\text{max, FIFO}, S_1} = \delta_{\text{max, FIFO}, S_2} = 14.01\mu s. \tag{122}\]

Since, we already compute the minimum delay for the FIFO systems of $S_1$ and $S_2$, the jitter bounds are:

\[
V_{\text{FIFO}, S_1} = V_{\text{FIFO}, S_2} = 13.5\mu s. \tag{113}\]

Having the knowledge on jitters of each element, we now compute the RTO bound of the flow at $h_2$. To obtain the RTO bound, we use Theorem 7 since the switching fabric (SF) in $S_1$ is the first non order-preserving element, we need to compute its corresponding RTO. Using Theorem 5.

\[
\lambda_{SF, S_1} = \left[ V_{SF, S_1} - \alpha_4^t(2L_{\text{min}}) \right]^+ = 1.5\mu s - 0.512\mu s = 0.988\mu s. \tag{114}\]
TABLE IV: Arrival curve propagation from point 1 to point 9 under lossy network condition. Arrival curve at point $i$ is $\alpha_i(t) = \min(6.4e3 \ t + b_i, 125e6 \ t + M_i)$, where $b_i$ and $M_i$ are in bytes and shown in the table.

| Re-sequencing | Burst | 1     | 2     | 3     | 4     | 5     | 6     | 7     | 8     | 9     |
|---------------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| Only at $h_2$| $b_1$ | 6400  | 6400  |       | 6400  | 6400  | 6400  |       |       |       |
|               | $M_1$ | 64    | 251   | 1751  | 64    | 251   | 1751  | 64    | 251   | 1751  |
| Only at $S_2$| $b_1$ | 6400  | 6400  |       | 6400  | 6400  | 6400  |       |       |       |
|               | $M_1$ | 64    | 251   | 1751  | 64    | 251   | 1751  | 64    | 251   | 1751  |
| At $S_1$ and $h_2$ | $b_1$ | 6400  | 6400  | 6400  | 6400  | 6400  | 6400  |       |       |       |
|               | $M_1$ | 64    | 251   | 626   | 1875  | 64    | 2012  | 64    | 251   | 375   |
| At $S_1$ and $S_2$| $b_1$ | 6400  | 6400  | 6400  | 6400  | 6400  | 6400  |       |       |       |
|                | $M_1$ | 64    | 251   | 626   | 1875  | 64    | 2012  | 64    | 251   | 375   |

Therefore the RTO at $h_2$ is:

$$\Lambda(h_2) = \lambda_{SF,S_1} + V_{FIFO,S_1} + \lambda_{SF,S_2} + V_{FIFO,S_2} = 0.988\mu + 13.5\mu + 13.5\mu + 29.49\mu s.$$ (115)

Then $T_{h_2} = \Lambda(h_2) = 29.49\mu s$.

Now, if the network is lossless, due to Theorem 4, re-sequencing is for free; therefore, the bounds are:

$$\delta_{e2e}^{0,\max} = \min(\Delta_{SF,h_1} + \delta_{SF,S_1}^{\max} + \Delta_{SF,S_2}^{\max}) = 63.2 + 2 + 14.01 = 95.22\mu s,$$

$$V_{e2e}^0 = V_{FIFO,h_1} + V_{SF,S_1} + V_{FIFO,S_1} + V_{SF,S_2} + V_{FIFO,S_2} = 62.69 + 1.5 + 13.5 + 13.5 = 92.69\mu s.$$ (116)

Using Corollary 2 the RBO bound at $h_2$ is:

$$\Pi(h_2) = \alpha(V_{FIFO,h_1} + V_{SF,S_1} + V_{FIFO,S_1} + V_{SF,S_2}) - L_{\min} = \alpha(79.19\mu s) - 64 = 6336\ bytes.$$ (117)

Then $B_{h_2} = \Pi(h_2) = 6336\ bytes$. Note that due to Theorem 5 we eliminate the FIFO system at $S_2$ as the switching fabric of $S_2$ is the last FIFO system.

If the network is lossy, by Theorem 4 the jitter and delay worst-case are increased by $T_{h_2} = 29.49\mu s$; therefore:

$$\delta_{e2e}^{\max} = 124.72\mu s, \hspace{1cm} V_{e2e} = 122.19\mu s.$$ (118)

The size of re-sequencing buffer is:

$$B_{h_2} = \alpha(V_{FIFO,h_1} + V_{SF,S_1} + V_{FIFO,S_1} + V_{SF,S_2} + V_{FIFO,S_2} + T_{h_2}) = \alpha(122.19\mu s) = 6400\ bytes.$$ (119)

2) Re-sequencing only at $S_2$: Similarly to the previous scenario, $\delta_{e2e}^{\max,S_1} = 14.01\mu s$ and $V_{e2e}^{S_1} = 13.5\mu s$; and therefore, we obtain RTO bound in switching fabric of $S_1$ as $\lambda_{SF,S_1} = 0.988\mu s$. Using Theorem 7 the RTO bound after the switching fabric of $S_2$ is:

$$\Lambda(S_2) = \lambda_{SF,S_1} + V_{FIFO,S_1} + V_{SF,S_2} = 0.988\mu + 13.5\mu + 1.5\mu = 15.99\mu s.$$ (120)

Then $T_{S_2} = \Lambda(S_2) = 15.99\mu s$.

Now, if the network is lossless, $\delta_{e2e}^{\max,S_2} = 14.01\mu s$. Then, due to Theorem 4 re-sequencing is for free; therefore, similarly to the previous strategy $\delta_{e2e}^{\max,h_2} = 95.22\mu s$ and $V_{e2e}^{h_2} = 92.69\mu s$. Also for the RBO bound at $S_2$, using Corollary 2 we have:

$$\Pi(S_2) = \alpha(V_{FIFO,h_1} + V_{SF,S_1} + V_{FIFO,S_1} + V_{SF,S_2}) - L_{\min} = \alpha(79.19\mu s) = 6336\ bytes.$$ (121)

By Theorem 2 and the size is $B_{S_2} = \Pi(S_2) = 6336\ bytes$.

If the network is lossy, by Theorem 5 the jitter and delay worst-case are increased by $T_{S_2} = 15.99\mu s$; therefore this affects the arrival curve at point 7 and in turn delay bound of output FIFO system at $S_2$. Then, $\delta_{e2e}^{\max,S_2} = 30\mu s$ and $V_{FIFO,S_2} = 29.49\mu s$.

Finally,

$$\delta_{e2e}^{\max} = \delta_{e2e}^{\max,h_2} + \delta_{e2e}^{\max,S_1} + \delta_{e2e}^{\max,S_2} = 127.22\mu s,$$

$$V_{e2e} = V_{FIFO,h_1} + V_{SF,S_1} + V_{FIFO,S_1} + V_{SF,S_2} + V_{FIFO,S_2} + T_{S_2} = 124.69\mu s.$$ (122)

The size of re-sequencing buffer is:

$$B_{S_2} = \alpha(V_{FIFO,h_1} + V_{SF,S_1} + V_{FIFO,S_1} + V_{SF,S_2}) = \alpha(95.18\mu s) = 6400\ bytes.$$ (123)

3) Re-sequencing at $S_1$ and $h_2$: Since we already computed arrival curve at point 1, we compute RBO bound switching fabric at $S_1$ using Theorem 6 $\lambda_{SF,S_1} = 0.988\mu s$. Then $T_{S_1} = \lambda_{SF,S_1} = 0.988\mu s$. Similarly to switching fabric of $S_1$, we have $\lambda_{SF,S_2} = 0.988\mu s$.

Now, if the network is lossless, using the arrival curves in Table III $\delta_{e2e}^{\max,S_1} = \delta_{e2e}^{\max,S_2} = 14.01\mu s$ and $V_{e2e} = 13.5\mu s$. Hence, due to Theorem 4 re-sequencing is for free; therefore, similarly to the previous strategy $\delta_{e2e}^{\max,h_2} = 95.22\mu s$ and $V_{e2e}^{h_2} = 92.69\mu s$. To compute RTO bound at $h_2$, we need to find RTO bound switching fabric of $S_2$ as the first non order-preserving element after re-sequencing buffer of $S_1$ (the output of which is in-order). Using Theorem 7 the RTO bound at $h_2$ is:

$$\Lambda(h_2) = \lambda_{SF,S_2} + V_{FIFO,S_2} = 0.988\mu + 13.5\mu + 1.5\mu = 14.49\mu s.$$ (124)

Then $T_{h_2} = \Lambda(h_2) = 14.49\mu s$. Also for the RBO bound at $S_1$ and $h_1$, using Corollary 2 we have:

$$\Pi(S_1) = \alpha(V_{FIFO,h_1} + V_{SF,S_1}) - L_{\min} = \alpha(64.19\mu s) = 64 = 6336\ bytes,$$

$$\Pi(h_2) = \alpha(V_{FIFO,h_1} + V_{SF,S_1} + V_{FIFO,S_1} + V_{SF,S_2}) - L_{\min} = \alpha(79.19\mu s) = 6336\ bytes.$$ (125)

By Theorem 2 $B_{S_1} = B_{h_2} = \Pi(S_2) = 6336\ bytes$.

If the network is lossy, the re-sequencing buffer at $S_1$ increases the jitter by $T_{S_1}$; the impact on arrival curves is shown in Table IV. Hence, $\delta_{e2e}^{\max,S_1} = 15\mu s$, $\delta_{e2e}^{\max,S_2} = 14.01\mu s$. Using the obtained lower and upper delay bounds,
The size of re-sequencing buffers are:

\[ \Lambda(h_2) = \lambda_{SF,S_2} + V_{FIFO,S_2} = 0.988\mu s + 13.5\mu s = 14.49\mu s. \]

Then \( T_{h_2} = \Lambda(h_2) = 14.49\mu s. \) Finally:

\[ \delta_{e2e}^{max} = \delta_{FIFO,h_1}^{max} + \delta_{SF,S_1}^{max} + \delta_{FIFO,S_1}^{max} + \delta_{SF,S_2}^{max} + \delta_{FIFO,S_2}^{max} + \Delta_{S_1} + T_{h_2} = 111.72\mu s, \]

\[ V_{e2e} = V_{FIFO,h_1} + V_{SF,S_1} + V_{FIFO,S_1} + V_{SF,S_2} + V_{FIFO,S_2} + \Delta_{S_1} + T_{h_2} = 109.19\mu s. \]

The size of re-sequencing buffers are:

\[ B_{S_1} = \alpha(V_{FIFO,h_1} + V_{SF,S_1} + T_{S_1}) = \alpha(65.18\mu s) = 6400 \text{ bytes}, \]

\[ B_{h_2} = \alpha(V_{FIFO,h_1} + V_{SF,S_1} + T_{S_1} + V_{FIFO,S_1} + V_{SF,S_2} + V_{FIFO,S_2} + \Delta_{S_1} + T_{h_2}) = \alpha(109.19\mu s) = 6400 \text{ bytes}. \]

4) Re-sequencing at \( S_1 \) and \( S_2 \): Similarly to the previous scenario, we have \( T_{S_1} = \lambda_{SF,S_1} = 0.988\mu s. \) Also \( \lambda_{SF,S_2} = 0.988\mu s. \) Now, if the network is lossless, using the arrival curves in Table III \( \delta_{FIFO,S_1}^{max} = \delta_{FIFO,S_2}^{max} = 14.01\mu s \) and \( V_{FIFO,S_1} = V_{FIFO,S_2} = 13.5\mu s. \) Hence, due to Theorem 4 re-sequencing is for free; therefore, similarly to the previous strategy \( \delta_{e2e}^{max} = 95.22\mu s \) and \( V_{e2e} = 92.69\mu s. \) To compute RTO bound at \( h_2 \), we need to find RTO bound switching fabric of \( S_2 \) as the first non order-preserving element after re-sequencing buffer of \( S_1 \) (the output of which is in-order). Using Theorem 7 the RTO bound at \( S_2 \) is \( \Lambda(S_2) = \lambda_{S_2} = 0.988\mu s. \) Also for the RBO bound at \( S_1 \) and \( S_2 \), using Corollary 2 we have:

\[ \Pi(S_1) = \alpha(V_{FIFO,h_1} + V_{SF,S_1}) - L^{min} \]

\[ = \alpha(64.19\mu s) - 64 = 6336 \text{ bytes}, \]

\[ \Pi(S_2) = \alpha(V_{FIFO,h_1} + V_{SF,S_1} + V_{FIFO,S_1} + V_{SF,S_2}) - L^{min} \]

\[ = \alpha(79.19\mu s) - 64 = 6336 \text{ bytes}. \]

By Theorem 2 \( B_{S_1} = B_{S_2} = \Pi(S_2) = 6336 \text{ bytes}. \)

If the network is lossy, the re-sequencing buffer at \( S_1 \) increases the jitter by \( T_{S_1} \); the impact on arrival curves is shown in Table IV. Hence, \( \delta_{FIFO,S_1}^{max} = 15\mu s, \delta_{FIFO,S_2}^{max} = 14.01\mu s. \) Using the obtained lower and upper delay bounds, \( V_{FIFO,S_1} = 14.49\mu s, V_{FIFO,S_2} = 13.5\mu s. \) To compute RTO bound at \( h_2 \), similarly to the lossless case, \( \Lambda(S_2) = \lambda_{SF,S_2} = 0.988\mu s. \) Then \( T_{S_2} = \Lambda(S_2) = 0.988\mu s. \) Finally:

\[ \delta_{e2e}^{max} = \delta_{FIFO,h_1}^{max} + \delta_{SF,S_1}^{max} + \delta_{FIFO,S_1}^{max} + \delta_{SF,S_2}^{max} + \delta_{FIFO,S_2}^{max} + \Delta_{S_1} + T_{S_2} = 99.22\mu s, \]

\[ V_{e2e} = V_{FIFO,h_1} + V_{SF,S_1} + V_{FIFO,S_1} + V_{SF,S_2} + V_{FIFO,S_2} + \Delta_{S_1} + T_{S_2} = 96.69\mu s. \]

The size of re-sequencing buffers are:

\[ B_{S_1} = \alpha(V_{FIFO,h_1} + V_{SF,S_1} + T_{S_1}) = \alpha(65.18\mu s) = 6400 \text{ bytes}, \]

\[ B_{S_2} = \alpha(V_{FIFO,h_1} + V_{SF,S_1} + T_{S_1} + V_{FIFO,S_1} + V_{SF,S_2} + \Delta_{S_1} + T_{h_2}) = \alpha(83.19\mu s) = 6400 \text{ bytes}. \]