Packet Ordering Functions in Low-power Wireless Deterministic Networks

Juan-Cruz Piñero*†, Alberto Blanc*, J. Ignacio Alcareaz Hamelin†, Georgios Z. Papadopoulos*
* IMT-Atlantique, IRISA, France
Email: {alberto.blanc, georgios.papadopoulos}@imt-atlantique.fr
† Universidad de Buenos Aires, Argentina
Email: jpinero@fi.uba.ar, ihameli@cnet.fi.uba.ar

***Abstract*—Industrial Internet of Things applications need strict Quality of Service guarantees, including when they use wireless networks. While existing standardization efforts, such as Deterministic Networks (IETF) and Delay Tolerant Networks (IEEE), have focused mainly on wired networks, more recent activities address wireless networks, like the Reliable and Available Wireless (RAW) Working Group of the IETF. By duplicating packets over multiple paths, it is possible to improve availability and reliability, but at the potential expense of more out-of-order packets. Existing reordering algorithms, including two recently standardized by the IETF, are not always well suited to wireless networks. We propose a new reordering algorithm and simulate a 6TiSCH wireless network to compare its performance with the existing ones. We also extend existing Network Calculus bounds on the reordering timer to obtain path-dependant values that can reduce the average latency.**

I. Introduction

To support industrial applications, Internet of Things (IoT) technologies must offer strict Quality of Service (QoS) guarantees, in particular reliability (no packet losses), low jitter, and in-order packet delivery [1]. For instance, consecutive packet losses are especially harmful in industrial automation applications as they can halt production lines.

Wired networks are best placed to deliver such guarantees but at a significantly higher cost, especially when using redundant paths to increase reliability, hence the need for technical solutions to allow wireless networks to offer these guarantees. Within the Internet Engineering Task Force (IETF), the goal of the Reliable and Available Wireless (RAW) Working Group (WG) is to extend the results of the Deterministic Networking (DetNet) group to wireless networks. The latter has defined Packet Replication Function (PRF) and Packet Elimination Function (PEF) [2] to achieve low packet loss by sending multiple copies of the same packet over different paths. The downside of such a solution is the possibility of out-of-order packet delivery, which may not be acceptable for some applications, as highlighted in the DetNet network architecture document [2], where the in-order delivery is a crucial QoS metric. The document also defines a Packet Ordering Function (POF) without any implementation details.

More recently, the DetNet published a draft [3] giving possible implementations of a POF in a deterministic network. While in traditional IP networks, the transport or application layer at the receiving host handles out-of-order packets, not Layers 2 or 3, the draft outlines a few packet reordering algorithms for the IP layer. Several reordering algorithms have been proposed without focusing on wireless networks or paying attention to the memory needed to buffer packets while reordering them. Memory can be limited in resource-constrained scenarios like wireless networks with battery-powered nodes.

In this paper, we evaluate the performance of existing reordering algorithms [3], [4] in a simulated IPv6 over the TSCH mode of IEEE 802.15.4e (6TiSCH) network, including their memory usage, and we propose a new one that uses less memory than the existing algorithms while offering a lower latency. We also extend existing Network Calculus results to obtain a tighter per-path timeout bound.

II. Related Work

The reordering problem has been studied in different environments, along with its causes, impact on performance, and possible solutions in different domains, such as broadband digital television [5], reliable transport protocols [6], video traffic over UDP [7].

Kaspar et al. [8] address reordering when using multiple network interfaces in parallel and propose a sender-side solution to reduce the buffer requirements by dropping the most delayed packet, i.e., the one with the lowest sequence number, whenever the buffer is full. Evensen and et al. [9] address the same problem and propose a network layer proxy on the sender side, achieving transport protocol transparency.

Narasiodeyar and et al. [4] propose the Lowest-First Re-Sequencing Algorithm (LFRA) to reorder packets, with the resequencing buffer memory as the limiting resource (see Section IV-C for a detailed description of the algorithm). They evaluate the algorithm with simulations using prerecorded traces with out-of-order packets but do not derive delay bounds.

The IETF DetNet working group is standardizing a POF [3], including a basic reordering algorithm and two advanced versions. The basic version uses an unbounded buffer to store out-of-order packets and a timeout mechanism to forward them when it is no longer possible for the missing packet to arrive. The advanced version uses a timeout value that depends on the path to reduce the average latency (see Section IV-A for a detailed description of the algorithms).

ISBN 978-3-903176-63-8 © 2024 IFIP

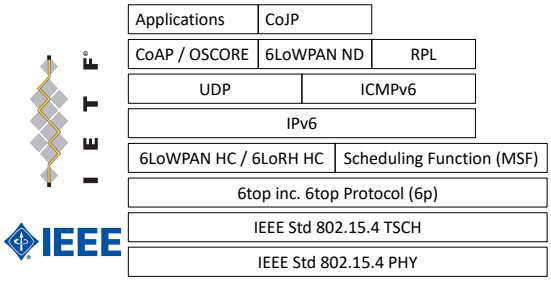


Fig. 1. 6TiSCH Protocol Stack [14].

Koutsiamanis and et al. [10] propose a multipath routing algorithm that exploits the properties of the shared wireless medium to provide redundancy in an IoT environment. The algorithm outperforms the single-path retransmission-based approach of IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) plus Time Slotted Channel Hopping (TSCH) and the LinkPeek solution when considering resource allocation, overhead, jitter, delay, and Packet Delivery Ratio (PDR). The authors did not address packet reordering, even though the proposed algorithm uses multiple paths in parallel.

When reordering packets in a lossy network, one has to decide how long to wait for out-of-order packets. The Network Calculus framework [11] is ideally suited to model Time-Sensitive Networking (TSN) systems and to derive performance bounds, including lower bounds on the timeout values when reordering packets [12], as well as arrival curves for the output of packet reordering functions[13].

III. System Description

This paper focuses on the 6TiSCH technology [14]. The 6TiSCH architecture proposes a protocol stack rooted in the Time Slotted Channel Hopping (TSCH) mode of the IEEE 802.15.4-2015 standard [15], supports multi-hop topologies with the RPL routing protocol [16], and is IPv6-ready through IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) [17]. 6TiSCH has defined the missing control plane protocols to match link-layer resources to the routing topology and application communication needs: the 6TiSCH Operation Sublayer (6top) and the 6top Protocol (6P) [18]. All these protocols form the 6TiSCH protocol stack, shown in Figure 1. Note that the Institute of Electrical and Electronics Engineers (IEEE) standardizes the lower layers, while the IETF standardizes the upper layers.

A. IEEE Std 802.15.4-2015 TSCH

Under TSCH, communications among the nodes are orchestrated by a schedule; see Figure 2. TSCH enables nodes in the network to coordinate through a communication scheduler, which uses Time Division Multiple Access (TDMA) for the time dimension and Frequency Division Multiple Access (FDMA) for the frequency dimension. More specifically, time is divided into discrete *timeslots* of equal length (e.g., 10 ms), sufficient for a node to transmit a frame and to receive an acknowledgment. The available frequency is divided into 16

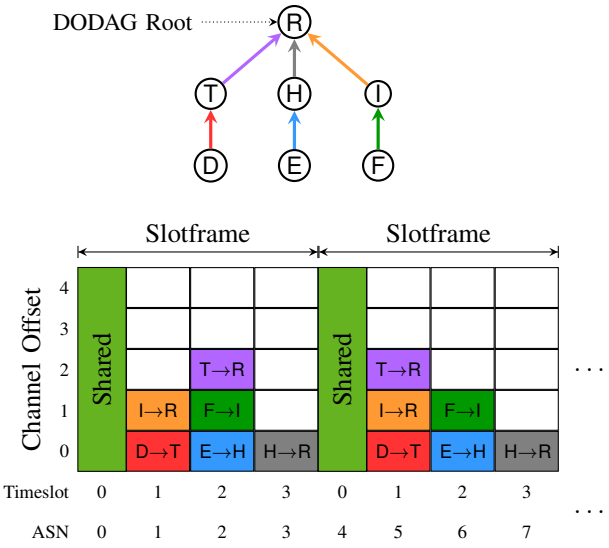


Fig. 2. A RPL DODAG topology (top) and TSCH schedule example (bottom).

non-overlapping physical *radio channels* operating at 2.4 GHz, where each radio channel has a bandwidth of 2 MHz and a channel separation of 5 MHz.

A group of consecutive timeslots forms a slotframe that repeats perpetually, and, according to the standard, it consists of 101 timeslots, but this value is configurable. The timeslots are identified by an Absolute Sequence Number (ASN) counter that increments as time elapses, i.e., the ASN counts the number of timeslots since the establishment of the network. Thanks to the periodic transmission of the Enhanced Beacon (EB) frames from the nodes in the TSCH network, all nodes are aware of the current ASN value, and they continuously re-synchronize with their neighbors. An EBs frame contains time and channel frequency information and information about the slotframe for new nodes to join the network.

To build a TSCH schedule, a collection of timeslots and channel offsets is assigned to each node, called “cells”. The cells are classified into two types, the Carrier Sense Multiple Access (CSMA) contention-based cells or the *shared cells*, and the contention-free cells or the *dedicated cells*, see Figure 2. The employed schedule indicates, for each timeslot, whether a node has to turn its radio *ON* to transmit or receive a frame or turn its radio *OFF* to save energy.

B. 6TiSCH Operation Sublayer (6top)

To enable an IPv6-based low-power wireless mesh architecture based on TSCH mode of the IEEE Std 802.15.4-2015 standard [14], IETF 6TiSCH WG specified a new sublayer, called 6top, to fill the gap between the 6LoWPAN and the IEEE Std 802.15.4-2015 TSCH layers, see Figure 1. 6top sublayer is composed of the 6top Protocol (6P) [18] and one or more Scheduling Functions (SFs) [19].

1) 6top Protocol (6P): 6P enables distributed scheduling in 6TiSCH-based networks by facilitating the negotiation of cells between two neighboring nodes, i.e., between a child and a parent node. In particular, 6P defines the transaction (control)

messages to "add", "delete", or "relocate" cells within the slotframe, i.e., node schedules. Note that when and how many cells to add, delete, or relocate is left to a SF.

2) Scheduling Function (SF): As previously stated, 6P only provides the necessary transaction messages. In contrast, a 6TiSCH SF decides when to add, delete, or relocate a cell according to the application requirements, routing changes, or schedule collisions. Consequently, it triggers the 6P transaction messages.

SFs can be classified into two broad categories. The *distributed* where the nodes build their own schedule solely based on the information exchanged locally with their direct neighbors, and the *centralized* where the schedule is computed and distributed by a single central entity. We use the *centralized scheduling* approach in this paper.

C. The RPL Routing Protocol

Industrial applications require low-power (wireless) mesh networks that consist of hundreds and thousands of nodes [1]. Therefore, to extend the industrial network beyond the radio coverage of one node, mesh technology enables some nodes to act as relays for others. However, beyond one hop, it requires a routing protocol.

RPL is one of the most adopted routing protocols for low-power (wireless) mesh networking. RPL is a proactive link-layer agnostic distance vector routing protocol, and thus it can operate over wireless or Power Line Communication (PLC) networks [16]. In the low-power and lossy networks we target, the topology is not predefined; thus, RPL discovers and selects nodes to construct optimal routes.

The RPL routing protocol nodes construct a DODAG based on a distance-vector technique. DODAG is a directed acyclic graph. Moreover, a single node is considered the DODAG Root, which serves as the gateway to other non-RPL networks. Based on a common Objective Function (OF) each node selects one or more parent(s), acting as a relay toward the DODAG Root. In such an acyclic network, the traffic from DODAG Root towards non-Root nodes is called the downstream traffic, while the reverse is called the upstream traffic. Figure 2 shows a DODAG topology that consists of 7 nodes with one DODAG Root.

D. DetNet, RAW, and PAREO Functions

In 2023, at the IETF, the DetNet and RAW WGs were merged, and thus, DetNet WG was chartered to include wireless technologies. Indeed, the DetNet WG will also work on defining how DetNet solutions operate over networks that include wired and wireless network technologies. This work focuses on Layer 3 to support applications requiring DetNet high reliability and availability and can operate over a wide range of scheduled-based wireless media, such as IEEE 802.15.4-2015 TSCH, 3GPP 5G Ultra-Reliable Low Latency Communications (URLLC), and IEEE 802.11ax/be.

One of the fundamental pieces of achieving reliable and available wireless networking is the Packet (hybrid) ARQ, Replication, Elimination and Ordering (PAREO) functions. PAREO is a superset of DetNet's Packet Replication Elimination and Ordering Functions (PREOF) that includes wireless lower-layer techniques such as constructive interference, overhearing, and PHY rate and other Modulation Coding Scheme (MCS) adaptation to increase the end-to-end reliability and to guarantee availability [20].

PAREO consists of the following individual functions:

- **Automatic Repeat reQuest (ARQ)**: if a failed packet transmission is detected, ARQ function resends a packet up to a predefined maximum count.
- **Replication**: this function duplicates each data packet, and each copy is forwarded over a different parent node over multiple paths in the network.
- **Elimination**: packet replication function introduces multiple copies of the same packet that traverse the multi-hop multi-path network independently. The elimination function is applied to remove unnecessary duplication.
- **Overhearing**: considering that a wireless transmission is broadcast by nature, any direct neighbor of a transmitter may "overhear" the transmission. The promiscuous Overhearing function enables multiple nodes to receive the same packet with one transmission if the receiver nodes are all within propagation range.
- **Ordering**: when RAW/DetNet flow packets are received out of order at the relay or at the final destination node, this function reorders packets.

In the literature, most of these functions have been evaluated [21]. However, to the best of our knowledge, the Ordering Function has not been studied in low-power wireless mesh networks. In this paper, we aim to exploit the advantages of the Ordering Function over the 6TiSCH technology.

IV. Reordering Algorithms

We briefly describe existing reordering algorithms and present a new algorithm better suited for wireless networks. Varga et al. [3] propose a basic and an advanced version of the algorithm meant to address some limitations of the basic algorithm at the cost of higher complexity, while Narasiodeyar and et al. [4] proposes only one.

To simplify the comparison between the different algorithms, we use a uniform notation and variable names (where applicable) for all the algorithms, different from those used by the authors of each algorithm. We assume that each packet has a sequence number that is incremented by one for each successive packet sent by the source. When a packet is replicated, each replica has the same sequence number as the original packet. For the sake of simplicity, we assume monotonically increasing sequence numbers (i.e., we do not consider wrap-around).

A. Basic POF algorithm

Let n (`POFLastSent`) be the sequence number of the last packet forwarded, i (`seq_num`) be the sequence number of the i-th packet, and T (`POFMaxDelay`) be the time after

which a packet stored in the buffer is sent even if it is out-of-order. In parenthesis, the names of the variables in the original document [3].

The pseudocode in Algorithm 1 is equivalent to the basic algorithm [3]. The $conn\_state$ variable keeps track of the state of each connection, including the sequence number of the last forwarded packet ($conn\_state\,.\,n$). Packets whose sequence number is less than $n+1$ are forwarded immediately (line 2 in PROCESS-PACKET). If the sequence number is equal to $n+1$ (line 3), n is incremented, and the RELEASE-PCKTS checks if any buffered packet can now be forwarded. The call to Get-Packet($conn\_state\,.\,n, buffer$) (line 2) returns the packet with sequence number $conn\_state\,.\,n$ from the $buffer$, if such a packet is stored in the buffer, and NONE otherwise.

If the sequence number is greater than n, packets are buffered (line 7), and a timer is started. When the timer expires after T units of time, it calls TIMER-EXPIRED with parameters $pkt\,.\,seq\_num, conn\_state, buffer$.

Algorithm 1 The basic POF algorithm

```
RELEASE-PCKTS(pkt, conn_state, buffer)
1  repeat
2      pkt' = Get-Packet(conn_state.n, buffer)
3      if pkt' ≠ NONE
4          Forward-Packet(pkt')
5          Stop-Timer(conn_state.n)
6          conn_state.n = conn_state.n + 1
7  until pkt' == NONE

TIMER-EXPIRED(pkt, conn_state, buffer)
1  Forward-Packet(pkt)
2  conn_state.n = pkt.seq_num
3  RELEASE-PCKTS(pkt, conn_state, buffer)

PROCESS-PACKET(pkt, conn_state, buffer)
1  if pkt.seq_num ≤ (conn_state.n) + 1
2      Forward-Packet(pkt)
3      if pkt.seq_num == (conn_state.n) + 1
4          conn_state.n = conn_state.n + 1
5          RELEASE-PCKTS(pkt, conn_state, buffer)
6  else
7      buffer.store(pkt)
8      START-TIMER(pkt.seq_num, T,
                   TIMER-EXPIRED(pkt, conn_state,
                                 buffer))
```

B. Advanced POF Algorithm

The basic algorithm uses a single timeout value T, which does not depend on the path taken by each packet. This is sub-optimal: if different paths have different (maximum) delays, as is often the case, one must always use the timeout value for the slowest path. The advanced algorithm replaces a single timeout value with an array of timeouts T_p one for each path p, reducing the average latency. (Except in corner cases where all the packet losses are on the faster paths, forcing the algorithm always to use the largest T_p anyway.)

This version requires the reordering node to know the path taken by each packet. This can be achieved, for example, by setting an ID value in the packet header at the replication point [3].

Another enhancement is to handle explicitly the first packets in each connection. We do not consider this further optimized version, as, for the sake of simplicity, we focus on long-lived connections.

C. LFRA Algorithm

The basic and advanced POF algorithms above assume that there is always space in the buffer storing the out-of-order packets. On the contrary, the LFRA [4] algorithm assumes a bounded buffer (see Algorithm 2). As long as the buffer is not full, the algorithm stores out-of-order packets until they can be sent in order (lines 2-9); if the buffer is full, it sends the packet with the smallest sequence number among all the packets in the buffer and the last received packet (lines 11-18). The call to Get-Min-Packet($buffer$) (line 14) returns the packet with the smallest sequence number currently stored in the buffer.

Algorithm 2 The LFRA algorithm

```
PROCESS-PACKET(pkt, conn_state, buffer)
 1  if pkt.seq_num == (conn_state.n) + 1)
 2      Forward-Packet(pkt)
 3      conn_state.n = conn_state.n + 1
 4      repeat
 5          pkt' = Get-Packet(conn_state.n)buffer)
 6          if pkt' ≠ NONE
 7              Forward-Packet(pkt')
 8              conn_state.n = conn_state.n + 1
 9      until pkt' == NONE
10  else
11      if buffer.full == FALSE
12          buffer.store(pkt)
13      else
14          pkt' = Get-Min-Packet(buffer)
15          if pkt'.seq_num < pkt.seq_num
16              Forward-Packet(pkt')
17          else
18              Forward-Packet(pkt)
```

D. PBAAPOF Algorithm

Given that nodes in IoT networks often have limited memory, we propose a new algorithm that combines ideas from the algorithms presented above (see Algorithm 3), called Path and Buffer Aware Advanced Packet Ordering Function (PBAAPOF). As the POF algorithm, it immediately forwards packets whose sequence number is less than $n+1$ (line 2). If the sequence number is $n+1$, it forwards the packet, increments the n (line 4), and then calls the same RELEASE-PKTS function (see Algorithm 1).

Unlike the POF algorithm, if the sequence number of the packet is greater than $n+1$, it checks if the buffer is full, like the LFRA algorithm (line 7). If it is not full, it calls the STORE-PKT function to store the packet in the buffer, computes the timeout value based on the path of the packet, and starts a time that will call the TIME-EXPIRED function (see Algorithm 1). If the buffer is full, it finds the packet with the smallest sequence number among all the packets in the buffer and the current packet (line 10, and it forwards it (stopping the timer if needed).

Algorithm 3 The PBAAPOF algorithm

```
STORE-PKT(pkt, conn_state, buffer)
1  buffer.store(pkt)
2  T = Compute-Timer-Value(pkt)
3  START-TIMER(pkt.seq_num, T,
                TIMER-EXPIRED(pkt, conn_state,
                              buffer))

PROCESS-PACKET(pkt, conn_state, buffer)
 1  if pkt.seq_num ≤ conn_state.n + 1
 2      Forward-Packet(pkt)
 3      if pkt.seq_num == conn_state.n + 1
 4          conn_state.n = conn_state.n + 1
 5          RELEASE-PCKTS(pkt, conn_state, buffer)
 6  else
 7      if buffer.full == FALSE
 8          STORE-PKT(pkt, conn_state, buffer)
 9      else
10          pkt' = Get-Min-Packet(buffer)
11          if pkt'.seq_num < pkt.seq_num
12              Forward-Packet(pkt')
13              Stop-Timer(pkt'.seq_num)
14              STORE-PKT(pkt, conn_state, buffer)
15          else
16              Forward-Packet(pkt)
```

V. REORDERING METRICS

RFC 4737 [22] defines several metrics applicable to DetNet systems. Among these, Reordering late-time offset (RTO) and Reordering Byte Offset (RBO) are relevant for the reordering algorithms as they can determine the timeout values and buffer size, respectively. The Reordered packet Ratio is a useful metric when comparing different simulations, as it quantifies the amount of packet reordering.

A. Reordered Packet Ratio

Let k be the next expected sequence number and s the sequence number of the incoming packet. The packet is considered in order if $s \geq k$, in which case $k \leftarrow s+1$. Otherwise (i.e., if $s < k$), the packet is considered reordered, and k is not incremented.

Let L be the number of unique sequence numbers observed (i.e., excluding duplicated packets). Then the Reordered Packet Ratio is:

$$R = \frac{r}{L}$$

where r is the total number of reordered packets. When receiving a packet multiple times, only the first copy is considered when computing r (Section 4.1.3 of RFC 4737 [22]).

B. Reordering late-time offset

Let E_i be the time packet i arrives at the reordering function or, more generally, at any given observation point. For duplicated packets, E_i is the earliest arrival time for one of the copies. Given that packets can be out of order, it is possible for packet j, with $j > i$, to arrive before packet i, i.e., $E_j < E_i$. The Reordering late-time offset (RTO) λ_i of packet i is the largest amount of time by which a packet j (with $j > i$) has to wait for packet k:

$$\lambda_i = E_i - \min_{j|j \geq i, E_j < E_i} E_j$$

For a connection, the RTO is $\max_i \lambda_i$, where the maximum is taken over all received packets, ignoring lost packets, for which E_i is ∞. For a given connection, the RTO is the maximum time any packet has to wait in the reordering buffer before it can be released in order.

C. Reordering Byte Offset

Similarly to the RTO, the RBO for packet i is the sum of all the bytes in the packets that have arrived before packet i ($E_j < E_i$) and whose sequence number greater than i:

$$\pi_i = \sum_{j|j>n, E_j < E_i} l_j$$

where l_j is the length of packet j. For a connection, the RBO is $\max_i \pi_i$, where the maximum is taken over all received packets, ignoring lost packets, for which π_i is not defined. RBO indicates how many bytes are needed in the reordering buffer to store out-of-order packets.

VI. NETWORK CALCULUS BOUNDS

The definitions of RTO and RBO presented in the previous section can be used only when one knows the arrival times of all the packets of a connection. It is, therefore, preferable to obtain upper bounds, ideally achievable, for these quantities. Mohammadpour and Le Boudec [12] use the Network Calculus framework to derive tight bounds on the RTO and RBO under the assumption that the delay in the network is bounded and the input flow satisfies an envelope.

After an overview of the existing results, we conclude the section with a new bound for the RTO for path-aware reordering algorithms, such as the advanced version of the POF algorithm and the PBAAPOF algorithm.

A. RTO and RBO bounds for the POF algorithm

Let d_i be the delay of the i-th packet and a given observation point (the reordering function in our case). Considering only the packets that are not lost of a connection, one can define the delay jitter V as:

$$V = \max_i\{d_i\} - \min_j\{d_j\}. \quad (1)$$

Theorem 5 of [12] states that, if the arrival flow has input envelope α, then:

$$\left[V - \alpha^{\downarrow}(2l), 0\right]^+ \quad (2)$$

is an upper bound for the RTO, where $[a]^+ = \max[a, 0]$, α is the input envelope, l is the minimum packet length of the flow, and $\alpha^{\downarrow}$ is the lower pseudoinverse of the envelope, defined as $F^{\downarrow}(x) = \inf\{s \geq 0 | F(s) \geq x\}$ [23].

Theorem 6 of [12] gives the following upper bound for the RBO, assuming, again, that the arrival flow has input envelope α: $\max(\alpha(V) - l, 0)$.

B. RTO bound for the advanced POF and PBAAPOF algorithms

As explained in Section IV, the advanced POF and PBAAPOF algorithms use a per-path timeout. Therefore, while still valid, the bound provided above is not the best one for all the paths. Given that, in most cases, each path has a different maximum and minimum delay, we can define $D^{(p)}$ and $d^{(p)}$ as the maximum and minimum delay, respectively, on path p.

Proposition 1. *If all the paths are First In First Out (FIFO), and if a flow has input envelope α, then for packets received on path p:*

$$E_{n-1} - E_n \leq \left[\max_{q \neq p}\{D^{(q)}\} - d^{(p)} - \alpha^{\downarrow}(2l)\right]^+. \quad (3)$$

Proof. Let A_n be the time when the source sends packet n. If the arrival process has envelope α, it follows that [12]:

$$A_n - A_{n-1} \geq \alpha^{\downarrow}(l_n + l_{n-1}).$$

Given that α is wide sense increasing, we have that:

$$A_{n-1} \leq A_n - \alpha^{\downarrow}(2l), \quad (4)$$

where l is the minimum packet size.

If packet n has arrived on path p, we have that:

$$E_{n-1} \leq A_{n-1} + \max_{q \neq p} D^{(p)} \quad (5)$$

$$\leq A_n - \alpha^{\downarrow}(2l) + \max_{q \neq p} D^{(p)} \quad (6)$$

$$\leq E_n - d^{(p)} - \alpha^{\downarrow}(2l) + \max_{q \neq p} D^{(p)} \quad (7)$$

where (5) follows from the definition of $A_n, E_n, D^{(p)}$ and the fact that, as each path is FIFO, if packet n has arrived on path p, and any packet $m \leq n-1$ is missing, it suffices to wait for packet $n-1$, which, if it arrives, it will arrive on a different path. Equation (4) implies (6), and (7) follows from the fact that packet n was received at time E_n on path p, hence $A_n \leq E_n - d^{(p)}$. The claim follows immediately by rearranging the terms. □

VII. Simulation Configuration

We have used the 6TiSCH simulator[1] to compare the performance of the algorithms discussed in Section IV in a 6TiSCH network. We added support for multipath to the simulator, as the official version does not include this feature.

Figure 3 shows the simulated topology: node 6 is the source, and node 0 is the destination. There are two paths from the source to the destination: *path 1* (nodes 6, 3, 1, 0) and *path 2* (nodes 6, 5, 4, 2, 1, 0). The first path is shorter but has two links with higher packet loss probability (low quality in the figure). While all the links in the longer path have a lower packet loss probability. The source sends two copies of each packet, one on each path. Node 1 runs the elimination function and the reordering algorithm.

The simulations use the routing protocol RPL, with the OF0 objective function, which ranks neighbors based on the expected number of transmissions needed to successfully transmit a packet (known as the ETX metric).

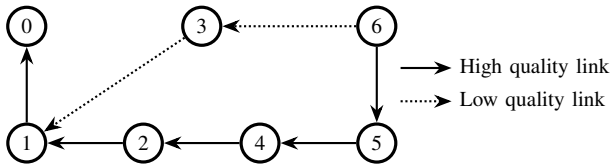


Fig. 3. The network topology used for the simulations.

The transmission schedule is the same throughout each simulation and is negotiated by the 6top layer in the setup phase of each simulation. We reduced the standard slot frame size from 101 cells to nine so that each link in Figure 3 has exactly one dedicated cell in each slot frame. Figure 4 shows a schedule example.

To guarantee a bounded delay, in the case of a lost packet on any of the links, in each simulation, we consider either no retransmissions or, at most, one. As both the schedule and the retransmission strategy are fixed for each simulation, it is possible to combine these two elements to easily compute the minimum and maximum delay for a packet between the source and the merge node (Node 1).

For example, for the schedule given in Figure 4, the minimum delay is five slots for path 1 and six for path 2 when the source (Node 6) has a packet to send right before the beginning of slot 0. In the case of no retransmissions, the worst case is 14 cells on path 1 and 15 on path 2, when the source has a packet to send right after the beginning of slot 0. In the case of one retransmission, the worst case is $T(N+1)$ where T is the number of slots in a cycle (9), and N is the number of hops: two for path 1 and four for path 2.

The input traffic has envelope $\alpha(t) = \sigma + \rho t$, with σ set to five packets, and ρ to one packet every nine slots in the simulations with no retransmissions and one packet every 18 slots in the simulations with one retransmission. In all the simulations, the slot duration is 10 ms, and the queue size at all the nodes, except the merge node, is 15 packets.

[1] https://bitbucket.org/6tisch/simulator/src

	0	1	2	3	4	5	6	7	8
Ch. 1	Shared	$6 \to 3$		$5 \to 4$	$4 \to 2$	$3 \to 1$	$2 \to 1$	$1 \to 0$	AutoRx
Ch. 2	Shared		$6 \to 5$						AutoRx

Fig. 4. A schedule example.

As explained in Section VI, we can derive an upper bound on the RTO by using equation 2 (Theorem 5 of [12]). In our case, the lower pseudo-inverse of the input envelope is [12] $\alpha(x)^{\downarrow} = [(x-\sigma)/\rho]^{+}$. Hence, equation (2) becomes:

$$\left[V - \left[\frac{2l-\sigma}{\rho}\right]^{+}\right]^{+} \tag{8}$$

where l is the (minimum) packet size, which is constant in our case.

Similarly, the RTO bound for the advanced POF and Path and Buffer Aware Advanced Packet Ordering Function (PBAAPOF) algorithms, based on equation (3) becomes:

$$\left[\max_{q \neq p}\{D^{(q)}\} - d^{(p)} - \left[\frac{2l-\sigma}{\rho}\right]^{+}\right]^{+} \tag{9}$$

VIII. Simulation Results

We ran two sets of simulations: one to quantify the amount of packet reordering in the network shown in Figure 3 and another to compare the performance of different recording algorithms. In each simulation, the source (Node 6) sends 5000 packets to the destination (Node 0).

A. Packet Reordering Ratio

We pick a random schedule at the beginning of each simulation to ensure that the results do not depend on the specific TSCH schedule. The PDR of the high-quality and low-quality links are either $(0.86, 0.68)$ respectively, or $(0.95, 0.75)$; and the retransmissions (RTX) are either zero or one. We ran 100 simulations for each of the four possible combinations of these parameters.

Figure 5 shows the Cumulative Distribution Function (CDF) of the Reordering Packet Ratio, the box plot of the number of packets successfully delivered, and the box plot of the end-to-end delay (latency). Each CDF of the reordering ratio is concentrated around three values, with a bigger "jump" around 0, i.e., no packet reordering at all.

This can be explained as follows: let a_i and b_i be the copies of each packet (with sequence number i) sent on the two different paths in 3. When there are no retransmissions and no packet losses, the difference between the sequence numbers of two consecutive packets received at the merge node is constant. Suppose that path a is faster, and the delay difference between the two paths is less than a slotframe, then packet a_i is followed by packet b_i, which is followed by packet a_{i+1}, and so on, again assuming that these packets are not lost. In this case, we say that the "sequence number offset" k between the two paths is 0.

Similarly, if the delay difference between the two paths is more than a slotframe but less than two, packet a_i is followed by packet b_{i-1}, which is followed by packet a_{i+1} (again assuming that all three packets arrive successfully). In this case, $k = 1$.

Figure 6 shows one example for three values of k (an 'x' represents a lost packet). If $k = 0$, reordering is not possible. No matter which packets are lost, none of the correctly received packets can be considered as reordered. Recall that the definition of reordered packets considers only the first copy of the received packet. If $k = 1$ or $k = 2$, reordering is possible, as shown in the example with packet b_2 ($k = 1$), and packets b_2 and b_3 ($k = 2$).

The three jumps in Figure 5(a) with no retransmissions (RTX=0) correspond to $k = 0, 1, 2$, with only a few schedules resulting in a difference of two between sequence numbers. Hence, the last jump in each CDF is smaller than the others.

In the case of retransmissions, k is not a constant that depends only on the schedule; it can change from one packet to another. In this case, more packets are correctly delivered, reducing the reordering. As expected, the delivery ratio and the latency are higher with retransmissions, as shown in Figure 5(b) and (c).

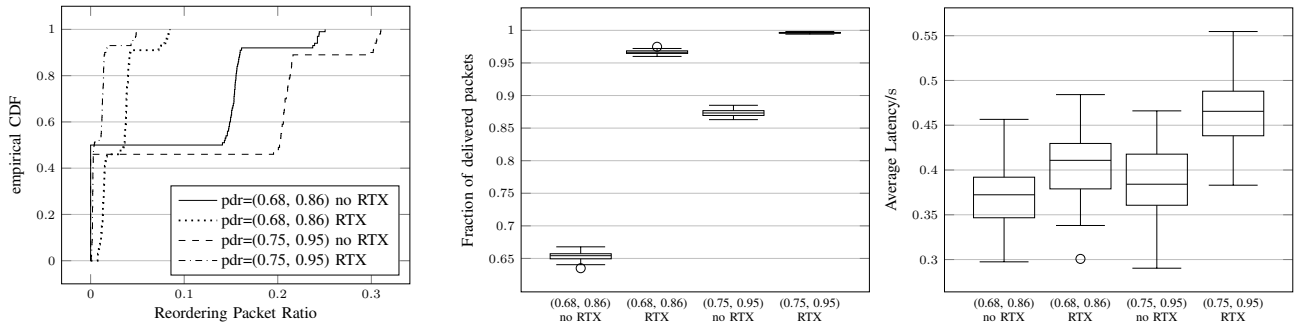
B. Reordering Algorithms Comparison

Reordering is more prevalent when there are no retransmissions. This setting has the advantage of having a lower and constant delay (for the packets that are delivered), which is especially relevant in a deterministic network setting.

We assess the performance of the algorithms presented in Section IV in a second set of simulations, with the same topology (see Figure 3), no retransmissions, high-quality and low-quality link PDR of 0.75 and 0.68. We selected one of the schedules that resulted in a Reordering Packet Ratio of roughly 30% among those used in the first set of simulations. We use the same schedule for all the simulations in this set. The reordering algorithm runs on node 1, i.e., where the two paths merge, which is the best location for lossy networks [12].

Each simulation consists of 5000 packets and uses the same random seed so that precisely the same packets are lost for all the algorithms. The interval between the packets the source sends is nine or ten slots (with equal probability). Therefore, the input envelope is a (σ, ρ) envelope with bursts (σ) of one packet and a rate (ρ) of one packet every nine slots. With these values, the RBO bound is three packets (see Section VI).

Figure 7 shows the empirical CDF of the end-to-end latency for different algorithms for the packets that were successfully delivered to the destination. When the buffer size is three, the PBAAPOF and advanced POF algorithms have the same



(a) (b) (c)

Fig. 5. Reordering Packet Ratio levels measured at destination (a), number of successfully delivered packets (b), and average end-to-end latencies (c) for different link quality and retransmission configurations. Each setup was run 100 times. In each run, the source sends 5000 packets replicated on the two paths in Figure 3.

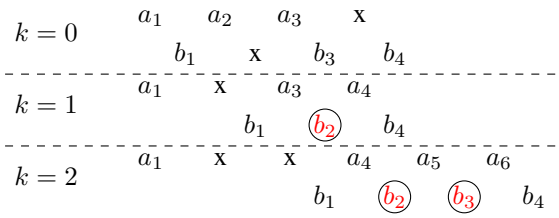


Fig. 6. A few reordering examples. Lost packets are represented by an 'x'. Circled packets are reordered.

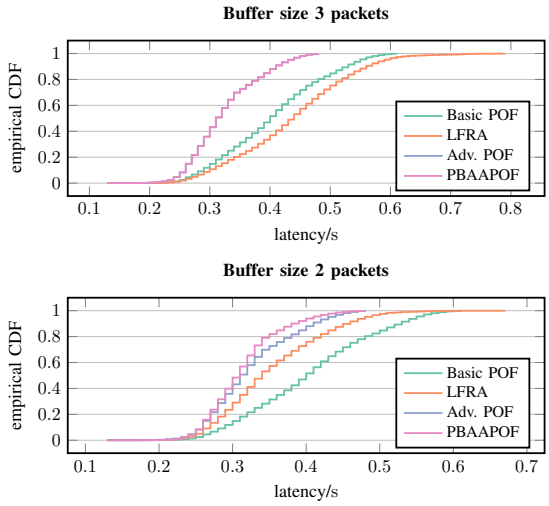


Fig. 7. The empirical CDF of the end-to-end latency (in seconds) with different reordering algorithms, with a buffer size of 3 packets (top) and 2 packets (bottom).

latency distribution, which dominates the distribution of the basic POF and LFRA algorithms.

While the network calculus RBO bound is achievable, it is a conservative value: only a few packets can achieve it. Hence, it is interesting to evaluate what happens with a smaller buffer, as shown in the bottom part of Figure 7. In this case, the PBAAPOF algorithm outperforms all the others, delivering a smaller latency.

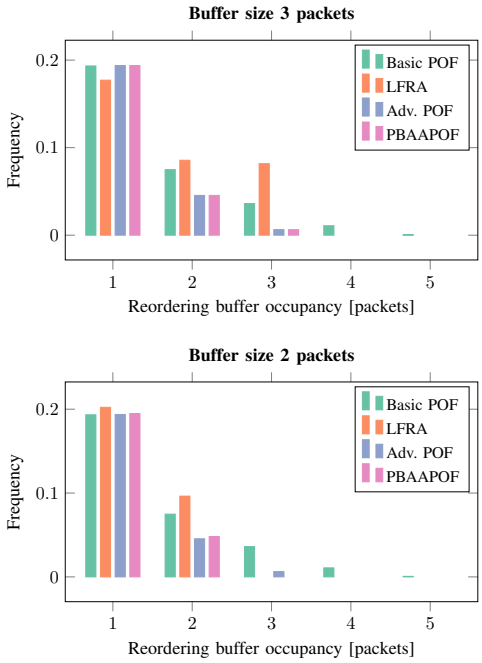


Fig. 8. The buffer occupancy with a buffer size of 3 packets (top) and 2 packets (bottom). The frequencies of an empty buffer are not shown.

Figure 8 shows the buffer occupancy when the buffer size for the LFRA and PBAAPOF algorithms is three and two packets. We omitted the bar for zero packets, i.e., empty buffer, to highlight the differences between the algorithms. Hence, the sum of the frequencies is not one. As the basic and advanced POF algorithms assume an unbounded buffer, we used a larger buffer in the simulations for these two algorithms.

Given that the LFRA algorithm does not use a timer, it has the highest buffer occupancy frequency for two and three packets because it stores out-of-order packets for a needlessly long time. This explains why its latency is the highest when the buffer size is three, while it's the second highest when it is two. In this case, the basic POF algorithm has a higher latency because its frequencies for three and four packets are not negligible, as shown in Figure 8.

TABLE I
FRACTION OF PACKET SUCCESSFULLY DELIVERED.

Algorithm	Buffer Size 3	Buffer Size 2
Basic POF	0.8672	0.8672
LFRA	0.8678	0.8674
Advanced POF	0.8764	0.8764
PBAAPOF	0.8764	0.8686

Table I shows the fraction of successfully delivered packets. The PBAAPOF and Advanced POF algorithms have the highest value when the buffer size is three. When the buffer size is two, the PBAAPOF algorithm has a lower value, which is still higher than the one for LFRA. Recall that the Basic and Advanced POF algorithms use the same and larger buffer size. It is, therefore, not surprising that they have a higher percentage of delivered packets, as they use more resources.

IX. CONCLUSIONS

Exploiting path diversity in a 6TiSCH network can increase the reliability even without retransmissions, but at the cost of packet reordering, which can reach 30%. Retransmissions increase reliability and decrease packet reordering but, unsurprisingly, result in higher latency, which is a drawback in delay-sensitive networks, such as those in Industrial IoT. Reordering algorithms implemented at the node(s) where multiple paths join can alleviate the packet reordering problem.

Existing reordering algorithms use different mechanisms, such as timers with an unbounded buffer or smart handling of bounded buffers. We proposed a new algorithm (PBAAPOF) that combines both approaches, resulting in lower latency and lower buffer occupancy, which are desirable properties in resource-constrained networks. We extended existing Network Calculus results to obtain potentially smaller timeout values, which can be applied in the Advanced POF and PBAAPOF.

The downside of PBAAPOF, shared by the Advanced POF algorithm, is the need to know the path taken by each packet. This can be avoided by combining the buffer-size-aware part of the algorithm, taken from the LFRA algorithm, with the basic version of the POF algorithm. This solution will be the topic of further studies.

REFERENCES

[1] C. J. Bernardos, G. Z. Papadopoulos, P. Thubert, and F. Theoleyre, "Reliable and Available Wireless (RAW) Use Cases," IETF, RFC 9450, August 2023.

[2] N. Finn, P. Thubert, B. Varga, and J. Farkas, "Deterministic Networking Architecture," IETF RFC 8655, Oct 2019.

[3] B. Varga, J. Farkas, S. Kehrer, and T. Heer, "Deterministic Networking (DetNet): Packet Ordering Function," IETF, I-D draft-ietf-detnet-pof-11, January 2024.

[4] R. M. Narasiodeyar and et al., "Improvement in packet-reordering with limited re-sequencing buffers: An analysis," in *Proc. of the 38th IEEE LCN*, 2013, pp. 416–424.

[5] S. Spirou, "Packet reordering effects on the subjective quality of broadband digital television," in *IEEE ISCE*, 2006, pp. 1–6.

[6] E. Blanton and M. Allman, "On making tcp more robust to packet reordering," *ACM SIGCOMM Computer Communication Review*, vol. 32, 02 2002.

[7] C. Arthur and et al., "The effects of packet reordering in a wireless multimedia environment," in *Proceedings of the 1st ISWCS*, 2004, pp. 453–457.

[8] D. Kaspar, K. Evensen, A. F. Hansen, P. Engelstad, P. Halvorsen, and C. Griwodz, "An analysis of the heterogeneity and ip packet reordering over multiple wireless networks," in *Proceedings of the 14th IEEE ISCC*, 2009.

[9] K. Evensen and et al., "A network-layer proxy for bandwidth aggregation and reduction of ip packet reordering," in *Proc. of the 34th IEEE LCN*, 2009, pp. 585–592.

[10] R.-A. Koutsiamanis and et al., "From best effort to deterministic packet delivery for wireless industrial iot networks," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4468–4480, 2018.

[11] J.-Y. L. Boudec and P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*, ser. Lecture Notes in Computer Science. Springer, 2001.

[12] E. Mohammadpour and J.-Y. Le Boudec, "On packet reordering in time-sensitive networks," *IEEE/ACM Trans. on Networking*, vol. 30, no. 3, pp. 1045–1057, 2022.

[13] L. Thomas, A. Mifdaoui, and J.-Y. Le Boudec, "Worst-case delay bounds in time-sensitive networks with packet replication and elimination," *IEEE/ACM Transactions on Networking*, vol. 30, no. 6, pp. 2701–2715, 2022.

[14] P. Thubert, "An Architecture for IPv6 over the Time-Slotted Channel Hopping Mode of IEEE 802.15.4 (6TiSCH)," IETF, RFC 9030, May 2021.

[15] "IEEE Standard for Low-Rate Wireless Personal Area Networks (LR-WPANs)," IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011), April 2016.

[16] T. Winter and et al., "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," IETF, RFC 6550, March 2012.

[17] J. Hui and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks," IETF, RFC 6282, September 2011.

[18] Q. Wang, X. Vilajosana, and T. Watteyne, "6TiSCH Operation Sublayer (6top) Protocol (6P)," IETF, RFC 8480, November 2018.

[19] T. Chang and et al., "6TiSCH Minimal Scheduling Function (MSF)," IETF, RFC 9033, May 2021.

[20] P. Thubert, "Reliable and Available Wireless Architecture," IETF, I-D draft-ietf-raw-architecture-16, Oct 2023.

[21] T. L. Jenschke and et al., "ODeSe: On-Demand Selection for Multi-path RPL Networks," *Elsevier Ad Hoc Networks*, vol. 114, p. 102431, 2021.

[22] A. Morton, G. Ramachandran, S. Shalunov, L. Ciavattone, and J. Perser, "Packet Reordering Metrics," RFC 4737, Nov. 2006.

[23] J. Liebeherr, "Duality of the max-plus and min-plus network calculus," *Foundations and Trends in Networking*, vol. 11, no. 3-4, pp. 139–282, 2017.