Shortening gate closing time to limit bandwidth waste when implementing Time-Triggered scheduling in TAS/TSN

Pierre-Julien Chaine, Marc Boyer

To cite this version:

Pierre-Julien Chaine, Marc Boyer. Shortening gate closing time to limit bandwidth waste when implementing Time-Triggered scheduling in TAS/TSN. International Conference on Real-Time Networks and Systems RTNS 2022, Jun 2022, Paris, France. hal-03771310

HAL Id: hal-03771310

https://hal.science/hal-03771310

Submitted on 7 Sep 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Abstract

Time-Triggered (TT) communications consists in specifying the instants (or time windows) at which any frame will be transmitted on any network link. In TSN Ethernet, this principle can be implemented using dedicated queues and time-triggered gates. Since TSN can mix several kinds of flows, the gates of non-TT flows are commonly closed during a TT window (when TT flows are transmitted). This is called “exclusive gating”. In this paper, we propose to use “protective gating”, that sets TT queues at the highest priority levels and reduces this closing time to its minimal value in order to save bandwidth while guaranteeing correct TT behavior.

1 Introduction

Time-Triggered (TT) communications consists in specifying offline the instants (or time windows) at which any frame will be transmitted [1]. In a store and forward network, a time window must be reserved for each frame on each link, and a good schedule keeps the waiting time of each frame as small as possible. Nevertheless, in complex systems, TT applications may coexist with Event-Triggered (ET) applications (i.e. producing ET data flows). In general, the networking technologies allow ET flows to use the medium outside the time windows dedicated to TT flows [2]. This approach is called “exclusive gating”. This time partitioning may lead to some waste of bandwidth. In fact, when the time window dedicated to a TT flow is not used, or partially used, the unused time cannot be used by ET flows.
In this paper, instead of protecting the whole TT time window, we propose to protect only its beginning, while setting TT queues at the highest priority levels. This approach is called “protective gating” and, in the context of a TSN (Time Sensitive Networking) network, can offer the same guarantees to TT traffic, while reducing the bandwidth waste.

2 Medium access in TSN

Let us first recall the architecture and frame selection rules in a TSN output port. A TSN output port is made of up to 8 queues. To each queue is associated a Transmission Selection Algorithm (TSA) and a gate. The port itself uses a single static priority arbiter (cf. Figure 1).

The TSA determines, at each instant, if the head of queue frame is “available for transmission” \[3, \S 8.6.8\]. There are currently 4 possible TSA (Static Priority, Credit-based Shaper, Enhanced Transmission Selection and Asynchronous Traffic Shaper) \[3, Table 8.6\], \[4\]. Each TSA has its own rules to determine the availability for transmission.

The gate is either open or closed. Each output port has its own Gate Control List (GCL). Each entry in the list has a duration (known as “TimeInterval”) and a state for each queue gate (open or closed). A state machine crosses the list, opening and closing each gate during the specified duration, in a cyclic way, as defined in the Time Aware Shaper (TAS) addendum \[5\].

The output port arbiter considers the set of queues satisfying three conditions: first, the head of queue frame is set as available for transmission by the queue TSA, second, the gate is open, and third, there is “sufficient time available to transmit the entirety of that frame before the next gate-close event” \[5, \S 8.6.8.4\]. Then, the arbiter selects, in this set, the head of the queue frame with the highest priority (i.e. with the highest queue number).

Note that this implies that when the gate of a queue is closed, it is not sending any frame.

Also note that the condition on next closing event is often presented as a “guard band”, but to highlight the fact that this condition is dynamic, depending on the frame size, we will call it “next closing dynamic guard band” or simply “dynamic guard band”.

3 TT communications in TSN

3.1 TT communications in store-and-forward networks

A TT schedule consists in considering a global time horizon (often the least common multiple - lcm - of the period of all data flows), and allocate, to each frame sent to the network during this period, a slot (aka time window) on every link along its path from source to destination (this path being considered as fixed).
Consider the network depicted in Figure 2 with three data flows, A, B, C, with routing depicted on the Figure, and assume a period $P$ for B, C and $P/2$ for A.

A simple example of TT schedule is presented in Figure 3 On each link, two slots, $A_1$ and $A_2$, are reserved for flow A, and only one slot, denoted $B_1$ (resp. $C$) is reserved for flow B (resp. flow C).

Consider first flow C: it has a slot scheduled on link $E_0 \rightarrow S_1$, and right after that (as soon as the frame is fully received, plus some margin due to clock error and internal commutation delay), another slot is scheduled on link $S_1 \rightarrow S_2$, and the same on link $S_2 \rightarrow E_3$. As shown in Figure 3, the frames of flow C spend the least amount of time in intermediary buffers. They thus benefit from the smallest end-to-end network latency. The flow A has the same kind of behavior,
with slots \(A_1\) and \(A_2\). The situation is different for \(B\): it is sent just after \(A_1\) on \(E_1 \rightarrow S_1\), but it stays longer in the buffers of \(S_1\) since it is forwarded only just before \(A_2\). It then experiences a latency larger than the other flows (but it may be sufficient w.r.t. its requirements).

This behavior has been chosen to illustrate several points. First, notice that it is not possible to ensure a minimal latency for all frames without changing the instants at which these frames are injected into the network (that may impose to change the instants when the embedded data are computed, or to induce some waiting time between computation and injection). Second, some frames are sent back-to-back, therefore, the sequence \(A_1B\) can be seen either as two adjacent slots, or as a single slot hosting two frames. Third, some trade-offs may exist on slot size w.r.t ET data flows: on the one hand, during a slot dedicated to TT frames, no ET frame can be forwarded, therefore a large TT slot increases the latency of ET flows, which favors small slots; but on the other hand, the time before a TT slot cannot always be used (i.e. a large ET frame cannot start its transmission if it cannot be completed before the TT slot), creating a per-slot penalty, which favors a small number of slots. Fourth, in this schedule, frame \(B\) is fully received by \(S_1\) before frame \(C\), but is forwarded after.

In summary, building a TT schedule consists in defining slots (without any encroachment between two slots) and assigning frames to slots while optimizing criteria such as number of slots, slot length, end-to-end frame latency, while satisfying that each frames will be received before its departure slot on any hops.

### 3.2 Implementing TT communications in TSN

As presented in Section 2, the TT behavior of TSN is based on queues, not on flows. Then, to send a frame during a predefined time window, this frame has to be written in a queue before the time window, when the queue gate is closed. The frame will be transmitted during the time window, when the gate is open (if the slot is large enough).

Also note that the order of frames in a queue is in general the arrival order\(^1\). Then, it is not possible to implement the scheduling of Figure 3 using a single queue for TT frames in \(S_1\), since \(B\) is received before \(C\) but forwarded after\(^2\). In this example, one solution would be to shift to the right the slot dedicated to \(B\) on link \(E_1 \rightarrow S_1\) (i.e. postpone \(B\) transmission). Then, it would be received after \(C\) and the output order will be the same as the input order. But for illustration purposes, let us keep this scheduling.

To implement such a schedule in TSN, the common practice consists in assigning one or several queues to TT flows (in the example depicted in Figure 4, queues \#7 and \#6), and opening the gate only during the slots of frames associated to this queue. All non-TT queues keep their gate closed when there is at

---

\(^1\)The exact requirements are specified in [3 § 8.6.6].

\(^2\)In TTEthernet, such a scheduling is not a problem since there is no order requirement between TT frames.
Figure 4: Example of GCL schedule implementing TT schedule for link $S_1 \rightarrow S_2$ in Figure 3 with $A, C$ at priority 7 and $B$ at priority 6.

The time-lines $Q#7$ and $Q#6$ represent the state of the gates for these TT queues, and $Q#5-0$, represents the state of the gates for all other queues, in the “exclusive gating” mode. The “exclusive gating” mode uses the same gate scheduling for queues $Q#7$ and $Q#6$ but uses $\hat{Q}#5-0$ for the others.

least one TT-queue open, and open it when all TT-queues are closed. The TT queues have no TSA. This is known as “exclusive gating”.

Note that exclusive gating does not require that the TT queues have the highest priority: during a slot, a TT queue has exclusive access to the output port, and the static priority arbiter is useless. Nevertheless, it is a common practice to do so.

4 State of the art

Building a TT schedule is a hard and old problem, and a complete overview can be found in [6].

The opportunity to support both TT and ET flows in an Ethernet context is a feature of TTEthernet. It relies on a per-frame time slot allocation, with dedicated hardware support, and not on a per queue gate opening and closing. Like TSN, it has to handle the case of ET frames trying to access the medium just before a TT slot, and supports 3 integration methods: timely block, shuffling and preemption [2]. A lot of work have been done on the efficient computation of a global schedule, see for example [7, 8, 9].

Once the IEEE have defined an ET real-time extension of Ethernet, known as AVB, which ensures guaranteed latency and controlled jitters, the need of “temporal isolation” for “scheduled traffic” (ST) appeared, and it was proposed to use a “separate class” (i.e. one dedicated queue) “in the highest priority” [10].

As presented in Section 3.2, the queue-based storage of frames requires to adapt the algorithms developed for TTEthernet.

In order to cope with the potential non-determinism induced by the loss of a frame, [11] adapts the constraints of [7] and introduces Flow Isolation and Frame Isolation. In order to take into account non-TT traffic while building the TT schedule, [12], [13] and [14] introduce strategies to modify the TT frame...
schedule by either spacing the frame offsets or gathering them.

Most recently, a second approach with configurations based on schedule per group of frames instead of per frame, has appeared. [15] applies the TTEthernet schedule generation methodology [16] to TSN networks. The authors introduce new sets of constraints adapted for group of frames schedules as well as Stream Isolation, a fusion of Frame isolation and Flow isolation to again cover the loss of a frame. In [17], the same authors use their new constraints to implement a configuration generator and compare their two approaches. More recently [18] proposes a group of frames configuration but chooses not to use exclusive gating like all other configuration generators. Moreover, it considers non-TSN end-stations (i.e. Ethernet) in their system.

The first use of the expression “exclusive gating” in the context of TSN seems to appear in [19].

5 Proposal: protective gating

Our proposal consists in relaxing the “exclusive gating” by only closing the gate of each non-TT queue during a time interval as small as possible (depending on the implementation[5]) just when the gate of a TT queue opens, and setting all TT queues to the highest priority levels (i.e. if there are 3 TT queues, they will be queues #7, #6, #5), eventually by reassigning queues, but maintaining their relative order. This reduction of the closing time is called “protective gating”, and is illustrated in Figure 4, where the P in gray boxes stands for “Protective”.

Note that it does not mean that each closed window in the non-TT scheduling is replaced by a single short “protective” closing event: it may be replaced by several ones. Consider Figure 4 if the third protective closing event was absent, if frame $B$ was lost or shorter than its slot, a non-TT frame would be able to start its transmission before the $A_2$ slot and prevent/delay the transmission of $A_2$ frame.

We claim that the small modification introduced by protective gating offers a small benefit without any cost.

5.1 Condition for no impact on TT traffic

If all the frames that must be sent during a slot are in the output queue before the slot opening, and if the TT queues have highest priorities, then “protective gating” offers the same behavior to these frames than “exclusive gating”.

Consider one TT queue, and one of its slot. Assume that there are $n$ frames to be sent during the slot and that all are in the output queue before the slot opening.

In an “exclusive gating” configuration, at the slot beginning, the gate of this queue opens and all other gates are closed (they are just closing or already

---

[5] The duration associated to each opening/closing event must be a multiple of an implementation-defined constant, TickGranularity [§8.6.9.4.16].
During the whole slot, only this queue has its gate open, and only this queue has access to the output port. Since all frames are already in the queue, they are sent back-to-back (there is no TSA to block a frame).

In a "protective gating" configuration, at the beginning of the slot, the gate of this queue opens and all other gates are closed (they are just closing or already closed). Then, the head of the TT queue can be sent. At the end of transmission of this first frame, the second frame of the slot is ready for transmission. It may exist frames ready for transmission in others queues, but these queues are non-TT. Since the TT queue has the highest priority, this second frame is sent just after the first one. The same argument holds at the end of this frame, and the next ones. Then, all frames of the slot are sent back-to-back, a behavior equivalent to the "exclusive gating" configuration.

5.2 Impact on non-TT traffic

After having shown that our approach do not penalize TT traffic, let us discuss the impact on non-TT traffic.

5.2.1 Bandwidth gain

In exclusive gating, the TT traffic has two negative impacts on the other flows. First, a frame can be blocked if it cannot be fully transmitted before the gate closing (called "dynamic guard band"). This blocking time ($B$ in Figure 5) is at most the transmission time of a frame of maximal size. Second, no frame can be sent during a TT slot, even if no TT frame is being sent. This unused time ($U$ in Figure 5) can be as large as the slot.

Our proposal does not address the blocking, but mitigates the unused time. It creates a gain when a slot is not fully used by TT frames. Let us now consider when this may happen. Indeed, as presented in Section 4, there are several ways to use the TAS mechanism in TSN. We use here the terminology from [18].

The strategy used to build the TT schedule has an impact on the possible gain, and the main strategies (0GCL, FGCL, WND, FWND) will be presented further.

But let us start with a global discussion on slot size. The simplest way to build a TT schedule consists in considering each TT frame independently, and assigning a slot to each frame. But this may lead to a high number of slots. And because of the dynamic guard band, this has a negative impact. Moreover, this may lead to "a large number of GCL events exceeding the hardware capabilities of existing TSN devices" [18]. For these two reasons, it is better to reduce the number of slots, putting several frames in the same slot. Nevertheless, a long
slot increases the latency of non-TT flows, and a good TT schedule has also to consider non-TT flows [20].

We now present the main ways to use TAS in TSN, and the associated gain.

0GCL, FGCL The goal of 0GCL is to ensure 0 jitter to TT flows [11]. Frame-to-Window based GCL (FGCL) relaxes the 0 jitter requirement [17]. But both consider schedules that assign to each slot a fixed set of ordered frames.

These methods are not perfect. In fact, it has been shown in [11] that, due to the queue-based storage, a loss of frame can break the schedule. Countermesures have been developed (called "flow isolation" and "frame isolation") but they tend to limit the number of frames per slot.

In these approaches, a slot may be not fully used either when TT frames are of variable size (then requiring a reservation for the maximal frame size), or when a TT frame is absent (because of loss at network level, or because the application did not produce the data, for example in case of oversampling).

The gain provided by protective gating is then limited.

WND, FWND The requirement of knowing in advance which frames will use which slot can be considered as an over-specification. The Window-based scheduling (WND) and Flexible Window-based scheduling (FWND) build schedules that ensure guaranteed latency for every frame, without knowing exactly which slot a frame will use. This may lead to under utilization of some slots [18, 21].

The protective gating then allows to re-use the unused part of these slots.

5.2.2 Impact on credit-based shaper

The gate closing (associated to TT slots) also has an impact on the Credit-based Shaper (CBS) Transmission Selection Algorithm. CBS is based on a credit associated to a queue. Its value decreases when the queue sends a frame, and increases either to refuel up to 0 or when the head of queue is blocked by a frame of another queue. The value of the refuel/increase slope is a network administration parameter. In case of gate closing, the value of the credit is frozen, but the slope value is globally updated to compensate this freeze time [5 §8.6.8.2 d)]. This rule seems to be designed to avoid burst when the gate re-opens, but its real impact is not well known [22]. In protective gating, most of this effect may be reduced.

5.2.3 Easy emergency traffic integration (and 802.1AS messages)

Emergency events, alarms, may be raised in an event-triggered mode in real time systems. And they may require as-soon-as-possible delivery, meaning that they need to have a priority higher than TT traffic. Let us call EmT such Emergency event-Triggered traffic, and assume a minimal inter-arrival time between two EmT messages (or between two bursts).
As shown in [23], these messages must be set in the highest priority queue, \( Q\#7 \), whose gate is always open. And TT queues are placed just below, in queues \( Q\#6 \), \( Q\#5 \)... But an EmT frame may then use a part of a TT slot, postpone the TT frames and break the TT schedule.

One solution would be to enlarge every TT slot to provision a possible burst of EmT frames. However, this may lead to a very high over-reservation.

Another solution consists in enhancing TAS behavior by dynamically enlarging TT slots in case of EmT frames [23].

Our claim is that such a feature can be implemented without any modification to the standard, but just by a modification of the algorithms building the schedule and by the use of protective gating. In fact, it is sufficient to build the schedule and the slots (using either 0GCL, FGCL, WND or FWND), as if a burst of EmT frames were present in each slot. But at run time, only the real frames will use the bandwidth.

Moreover, the computation of bounds on the latency for lower priority queues (with network calculus for example, as in [24]), can easily be updated to account correctly EmT frames. One just have to separate the interference created by TT and EmT. The EmT interference is computed using a minimal inter-arrival delay, and the TT interference is computed using the part of slots dedicated to TT flows.

Last, if a configuration devotes uses \( Q\#7 \) for EmT flows, one may also use it to forward synchronization messages devoted to time measurement and clock distribution [25].

5.3 Removing any protection?

In exclusive gating, the gate mechanism is used for two purposes: the opening/closing of a TT queue is used to schedule the TT frames within slots, and the opposite closing of other queues is used to protect the slots.

In protective gating, part of the protection given by gate closing is replaced by the static priority arbiter. One may wonder if the slot protection could be implemented using only static priority, to avoid the “blocking” penalty before a slot.

This solution has already been experimented in TTEthernet, and named “Shuffling” [2]. In such a case, a non-TT frame can encroach the beginning of a slot. Then, when building a schedule, each slot must be large enough to transmit the TT frames allocated to this slot, plus some non-TT perturbation. Without preemption, this effect is the maximal size of an Ethernet frame (1528 bytes, plus 20 bytes of IFG) whereas the use of preemption can decrease it to 143 bytes [26]. And comparison on latency between shuffling and “timely block” (which is equivalent to slot protection in TSN) can be found in the context of TTEthernet in [27]. The shuffling of course creates a jitter equivalent to the encroachment time.
6 Conclusion

One strength of TSN is its ability support both Time-Triggered (TT) and Event-Triggered (ET) flows. The implementation is usually done using “exclusive gating” where the gate mechanism is both used to implement the time slots devoted to TT frames and to protect these slots from ET frames. We propose in this paper “protective gating”, a small modification than can offer limited gains in bandwidth for ET flows but comes at no cost, then deserving consideration. It may also open a new freedom parameter for building schedules, since over-reservation in TT slots now comes at no cost.

References


