Soundness of the Quasi-Synchronous Abstraction

Guillaume Baudart[*‡] Timothy Bourke[‡*] Marc Pouzet[*†‡]

[*] École normale supérieure, PSL Research University
[†] Sorbonne Universités, UPMC Univ. Paris 06
[‡] Inria Paris

***Abstract*—Many critical real-time embedded systems are implemented as a set of processes that execute periodically with bounded jitter and communicate with bounded transmission delay. The *quasi-synchronous abstraction* was introduced by P. Caspi for model-checking the safety properties of applications running on such systems. The simplicity of the abstraction is appealing: the only events are process activations; logical steps account for transmission delays; and no process may be activated more than twice between two successive activations of any other.**

We formalize the relation between the real-time model and the quasi-synchronous abstraction by introducing the notion of a unitary discretization. Even though the abstraction has been applied several times in the literature, we show, surprisingly, that it is not sound for general systems of more than two processes. Our central result is to propose necessary and sufficient conditions on both communication topologies and timing parameters to recover soundness.

I. Introduction

The *Synchronous Real-Time Model* [2], [10] characterizes many distributed embedded systems: it applies whenever bounds exist on successive process executions and transmission delays. In particular, whenever computing units that execute periodically with jitter are connected together by network links. It is commonly employed in critical aerospace, power, and rail systems.

The *quasi-synchronous approach* [6], [8] formalizes a set of techniques for building distributed control systems that were observed by P. Caspi while consulting at Airbus on the distributed deployment of Lustre/SCADE[1] [17] designs. One of the key ideas is to model the computing units, network links, and shared memories themselves as a synchronous program [6, §3]. Such models can be verified using model-checking tools for discrete programs. This approach has, for instance, been applied to a Proximity Flight Safety (PFS) case-study from EADS Space Transportation [19] and to the analysis of systems specified in the Architecture Analysis and Design Language (AADL) [5], [20], [28].

An alternative way of developing real-time applications is to synchronize process executions. The Time-Triggered Architecture (TTA) [21], [22] thoroughly develops this approach and there are several clock synchronization protocols suitable for embedded systems. Once a clock synchronization scheme is adopted and assumed or verified correct, modeling and reasoning about applications is greatly simplified because non-determinism, in the form of possible interleavings, is either eliminated or reduced. The quasi-synchronous approach is nevertheless appropriate in certain applications either due to their simplicity, for example, microprocessors communicating directly over serial links, or the need for complete independence between subsystems, for example, as in redundant subnetworks connected only at voting units.

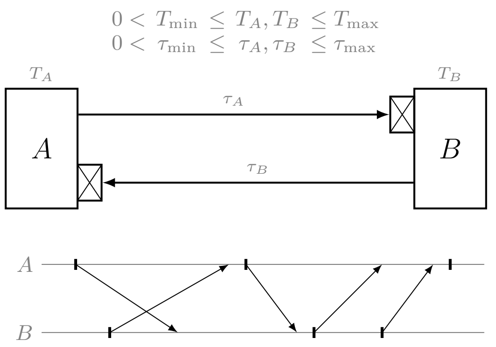
Figure 1 gives an overview of the quasi-synchronous approach. On the left is a real-time model comprising two processes, A and B, communicating through network links. Processes and links are annotated with timing bounds on executions ($T_{\min}$ and $T_{\max}$) and transmission delays ($\tau_{\min}$ and $\tau_{\max}$). Underneath is an example trace showing process activations and corresponding message transmissions. On the right is a discrete-time abstraction in which timing parameters are replaced by a discrete program called *Scheduler* that overapproximates their effect by controlling process activations, and, importantly, message transmissions are modeled by a single logical step. Underneath is a trace of the discrete-time model.

The ultimate aim is to verify properties of the real-time model in the simpler discrete-time model. The essential property is that every sequence of states that occurs in the real-time model can also occur in the discrete-time model.[2] Such an association guarantees soundness: all safety properties provable in the discrete-time model also hold of the real-time model. Since changes in state are directly related to received messages, we focus on traces without modeling process and network states explicitly. This means that a discrete model is a valid abstraction if every real-time trace has a discrete-time counterpart.

Contributions: We formalize the relation between real-time and discrete-time traces in the quasi-synchronous approach by introducing a *unitary discretization* based on the respective causality relations of the two models. With this tool we show that abstracting transmission delays as unit delays is not sound in general. We state and prove necessary and sufficient conditions on communication topologies and timing characteristics to recover soundness. We provide practical criteria for using the

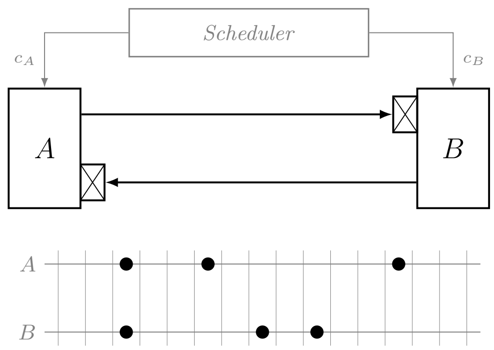
[1] http://www.ansys.com/Products/Embedded-Software/ANSYS-SCADE-Suite

[2] Assuming the state of the processes does not reference real time.



$0 < T_{\min} \leq T_A, T_B \leq T_{\max}$
$0 < \tau_{\min} \leq \tau_A, \tau_B \leq \tau_{\max}$
T_A
T_B
τ_A
τ_B
A
B
A
B

(a) Real-time model (*RT*)



(b) Discrete-time model (*DT*)

Fig. 1: Soundness: A property φ that can be verified in the discrete-time model will also holds for the real-time model, $RT \models \varphi \impliedby DT \models \varphi$.

quasi-synchronous abstraction to formally verify real-time systems in discrete-time model-checking tools [16], [18].

A. The Real-time Model

We consider the classic synchronous real-time model [2], [10], noting that 'synchronous' does not mean 'lock step'.

Definition 1 (Synchronous Real-Time Model)**.** *A* synchronous real-time model *is a finite set of processes $\mathcal{P}$, where for every process, the delay T between two successive activations is bounded:*

$$0 \leq T_{\min} \leq T \leq T_{\max}. \tag{RP}$$

Values are transmitted between processes with a delay $\tau \in \mathbb{R}$, bounded by $\tau_{\min}$ and $\tau_{\max}$:

$$0 \leq \tau_{\min} \leq \tau \leq \tau_{\max}. \tag{RT}$$

Each message is buffered at receivers until a newer value is received. Execution time (τ_{exec}) can be modeled either as a part of the communication delay ($\tau = \tau_{\text{exec}} + \tau_{\text{trans}}$), or as part of the activation period ($\tau_{\text{exec}} < T_{\min}$) with the convention that application components communicate through logical delays: values computed in one reaction are sent at the beginning of the next one.

For readability, we assume global bounds on successive process activations, but our results are readily generalized to multirate systems (see appendix C).

B. The Discrete-time Model

The simplest discrete abstraction is the *asynchronous model* where time is ignored altogether and process activations may be interleaved arbitrarily. This is sound but far from complete: many properties that hold in the real-time model cannot be shown in the discrete one. Furthermore, the many possible interleavings complicate reasoning about or model-checking the discrete-time model.

A finer abstraction was proposed by Caspi for processes that execute 'almost periodically', that is, $T_{\min} \approx T_{\max}$. He realized that the interleavings of systems satisfying RP can be constrained [7, §3.2]:

> It is not the case that a component process executes more than twice between two successive executions of another process.

Furthermore, he observed that when transmission delays are 'significantly shorter than the periods of [process activations]' they can be modeled by unit delays in the discrete-time model, but that 'if longer transmission delays are needed, modeling should be more complex' [6, §3.2.1]. A unit delay models the fact that a message sent at one logical instant is received at the next one.

More complex modeling refers to the standard approach of placing buffer processes between communicating processes. Such buffers provide receive and send events and maintain internal state to track messages in transmission. The quasi-synchronous abstraction eschews explicit link models thereby simplifying scheduling logic and halving the number of variables needed to model communication.

These observations allow abstraction from the timing details of the real-time model in definition 1 to give a non-deterministic and discrete-time model of systems termed *quasi-synchronous.* In the discrete-time model, boolean variables called *clocks* are set to *true* to activate processes.

Definition 2 (Quasi-Synchronous Model)**.** *A* quasi-synchronous model *comprises a scheduler and finite set of processes $\mathcal{P}$. The scheduler is connected to each process by a discrete clock signal. It activates the processes non-deterministically but ensures that no pair of clock signals (c_A, c_B), for a pair of processes $A, B \in \mathcal{P}$, ever contains the subsequence*

$$\begin{bmatrix} t \\ \_ \end{bmatrix} \cdot \begin{bmatrix} f \\ f \end{bmatrix}^* \cdot \begin{bmatrix} t \\ f \end{bmatrix} \cdot \begin{bmatrix} f \\ f \end{bmatrix}^* \cdot \begin{bmatrix} t \\ \_ \end{bmatrix},$$

where t indicates an activation, f means no activation, and _ means either of the two. Processes communicate through unit delays activated at every scheduler tick.

This restriction on subsequences of pairs of clock signals [6, §3.2.2] expresses formally the constraint quoted beforehand. The forbidden subsequence involves at least three activations of one process (A) between two successive activations of another (B). A finite state scheduler that produces valid sequences is readily constructed from the given regular expression (using, for instance, the `reglo` tool [29]). The processes and unit delays can be modeled directly in Lustre [17], for instance, and verified by model-checking [5], [19], [20], [28].

The quasi-synchronous model aims to reduce the state-space of a model in two ways: 1) by limiting the interleavings of process activations and 2) by simplifying message transmission modeling. In this paper, we show how the constraints imposed by the latter choice limit the applicability of the abstraction.

C. Relating Real time and Discrete time

Given definitions 1 and 2, it is natural to query the exact relationship between them, namely: *what are necessary and sufficient conditions on the architecture to ensure the soundness of the abstraction?*

The first step is to formalize real-time traces and their causality (section II). The main contribution of this paper is then to characterize the link between this causality relation and the causalities expressible in the discrete model. Specifically, we define a 'unitary discretization' that relates real-time traces to discrete-time traces (section III). It is quite constraining due to the modeling of communications as unit delays, but it still allows for the treatment of practically-relevant systems of two processes [19], [20] and those with certain communication topologies. Based on these results, we define precisely when the quasi-synchronous model can be applied to a real-time system (section IV). We relate our work to classic distributed systems models, to the expression of causality in distributed systems, and to existing work on the quasi-synchronous abstraction (section V).

II. Traces and Causality

We define a formal model for reasoning about real-time models and their discretization. It has two components: (real-time) traces and their induced causality relations. In the following, we fix an arbitrary real-time model with processes $\mathcal{P}$ and parameters $T_{\min}$, $T_{\max}$, $\tau_{\min}$, and $\tau_{\max}$ that satisfy definition 1. We formalize pairs of sending and receiving processes using a *communicates-with* relation, written $\rightrightarrows$, between the processes of a real-time model. This relation is not necessarily symmetric, $A \rightrightarrows B$ need not imply $B \rightrightarrows A$, but it must be reflexive ($A \rightrightarrows A$).

Definition 3 (Trace)**.** *A (real-time)* trace $\mathcal{E}$ *is a set of* activation events $\{A_i \mid A \in \mathcal{P} \wedge i \in \mathbb{N}\}$ *and two functions:*

- *$t(A_i)$, the date of event A_i with respect to an ideal reference clock, and*
- *$\tau(A_i, B)$, the transmission delay of the message sent at A_i to a process B.*

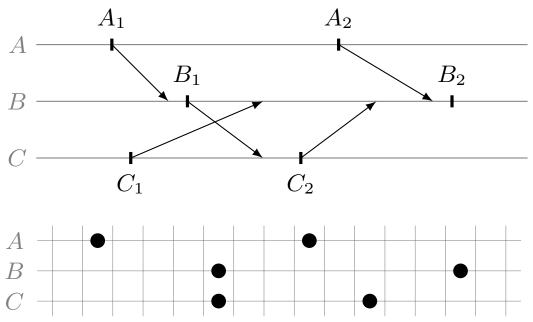


Fig. 2: A trace (above) and a possible unitary discretization.

Both $t(A_i)$ and $\tau(A_i, B)$ are non-negative reals satisfying the constraints of definition 1, namely if $A \rightrightarrows B$,

$$\begin{aligned} 0 \le T_{\min} &\le t(A_{i+1}) - t(A_i) \le T_{\max}, \text{ and} \\ 0 \le \tau_{\min} &\le \tau(A_i, B) \le \tau_{\max}. \end{aligned}$$

The causality relation between events within a given trace is essentially the *happened before* relation of Lamport [23]. Unlike Lamport, however, we do not explicitly model message reception. A message is received if the next execution of the receiver occurs after the corresponding transmission delay.

Definition 4 (Happened Before)**.** *For a trace $\mathcal{E}$, let $\rightarrow$ be the smallest relation on activation events that satisfies*

(local) *If $i < j$ then $A_i \rightarrow A_j$, and*
(recv) *If $A \rightrightarrows B$ and $t(A_i) + \tau(A_i, B) \le t(B_j)$ then $A_i \rightarrow B_j$.*

Activations at a single process are totally ordered (*local*); an activation at one process happens before an activation at another process when a message sent at the former is received before the latter (*recv*).

Compared to Lamport, we do not close the relation by transitivity. In this way, $A_i \rightarrow B_j$ means that B_j occurs strictly after the reception of the message sent by process A at A_i (otherwise, appendix A shows a counterexample). The same technique is used elsewhere [31, definition 1].

III. Unitary Discretization

We now address the central question of relating the real-time and discrete-time models. The problem is essentially one of correctly discretizing real-time traces.

If process A sends messages to process B, the most general approach is to ensure that when an event A_i occurs before an event B_j in the discrete-time trace, A_i happens before B_j ($A_i \rightarrow B_j$) in the corresponding real-time trace and vice versa. Figure 2 shows an example trace for a three-process system and a possible unitary discretization.

Definition 5 (Unitary Discretization)**.**
A function $f : \mathcal{E} \rightarrow \mathbb{N}$ that assigns each event in a (real-time) trace to a logical instant of a corresponding discrete trace, is a unitary discretization *if for all $A_i, B_j \in \mathcal{E}$,*

$$A_i \rightarrow B_j \iff (f(A_i) < f(B_j) \text{ and } A \rightrightarrows B)\,. \qquad \text{(UD)}$$

Discretizing a real-time model satisfying definition 1 to a model of the form given in definition 2 amounts to finding a unitary discretization for *each* of its (real-time) traces. The forward half of the equivalence comes from the fact that the $\rightarrow$ relation induces a partial order on events. Completing this relation to a total order gives a discretization that respects the causality of the real-time model [23].

A unitary discretization links the causality of events in the real-time model to the causality implicit in the discrete-time model. The backward direction of the equivalence imposes that if an event y occurs after an event x in the discrete-time model, that is, $f(x) < f(y)$, it is either because y is a later activation of the same process as x, or because y occurs strictly after the receipt of the message sent at x. It is the communication through unit delays on a common clock that tightly links the two causality relations.

In distributed systems terminology, condition UD is called *strong consistency* [30]. The problem of finding a unitary discretization is thus equivalent to the problem of finding a strongly consistent scalar clock. Raynal and Singhal report in their survey [30] that this is not possible in general, that is, there is no scalar clock function f that satisfies UD. This was already noted by Lamport in his original paper: 'We cannot expect the converse condition to hold as well [...]' [23, p.560].

The aim is to formulate sufficient conditions on the (static) $\rightrightarrows$ relation and on the timing characteristics of the real-time model to guarantee the existence of a unitary discretization. The following proposition will be useful.

Proposition 1. *If f is a unitary discretization for a trace, for a pair of processes where $A \rightrightarrows B$ we have that*

$$A_i \rightarrow B_j \implies f(A_i) < f(B_j), \text{ and}$$
$$A_i \not\rightarrow B_j \implies f(A_i) \geq f(B_j).$$

Proof. The first implication is a direct consequence of the definition of a unitary discretization. The second one follows by contraposition. If $f(A_i) < f(B_j)$, and since $A \rightrightarrows B$, we have $A_i \rightarrow B_j$ by the definition of f. $\square$

An intermediate step to defining a static condition on communications is to characterize traces for which there is no unitary discretization. Our characterization will be based on a graph of the constraints of proposition 1.

Definition 6 (Trace Graph)**.** *Given a trace $\mathcal{E}$, its directed, weighted* trace graph *$\mathcal{G}$ has as vertices $\{A_i \mid A \in \mathcal{P} \wedge i \in \mathbb{N}\}$ and as edges the smallest relations that satisfy*

1) If $A_i \rightarrow B_j$ then $A_i \xrightarrow{1} B_j$, and
2) If $A \rightrightarrows B$ and $A_i \not\rightarrow B_j$ then $B_j \xrightarrow{0} A_i$.

An example trace graph is shown in figure 3. Edges labeled with one ($x \xrightarrow{1} y$) represent the constraints $f(x) < f(y)$. Each such edge indicates that the source activation must come before the destination activation in a unitary discretization, that is, the value of f, from source to destination, must increase by at least one. Edges labeled

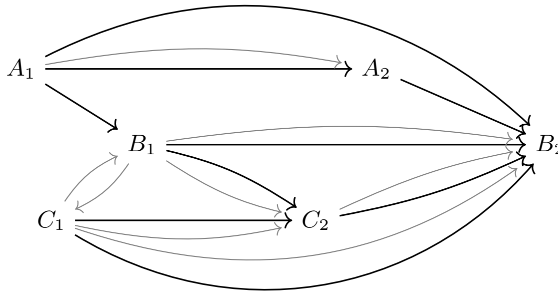


Fig. 3: The trace (sub-)graph of the trace in figure 2. Black thick arrows denote $x \xrightarrow{1} y / f(x) < f(y)$. Thin gray arrows denote $x \xrightarrow{0} y / f(x) \leq f(y)$.

with zeros ($x \xrightarrow{0} y$) represent the constraints $f(x) \leq f(y)$. Each such edge indicates that the source activation cannot be placed before the destination activation in a unitary discretization, that is, the value of f, from source to destination, must be the same or larger. A path through several activations defines their relative ordering in all unitary discretizations.

The satisfaction of the required constraints, or the impossibility of satisfying them, can now be phrased in terms of cycles in the graph. A cycle comprising only $\xrightarrow{0}$'s is acceptable: its activations are all assigned the same discrete slot (for example, B_1 and C_1 in figure 3). Any cycle containing a $\xrightarrow{1}$ represents a set of unsatisfiable constraints: one of the events must be placed in two different slots.

Lemma 1 ($\exists$UD $\iff \overline{\exists\text{PC}}$)**.** *For a trace $\mathcal{E}$, there is a unitary discretization* ($\exists$UD) *if and only if there is no cycle of positive weight in the corresponding trace graph $\mathcal{G}$* ($\overline{\exists\text{PC}}$)*.*

Proof. Assume there is a cycle of positive weight. By the construction of $\mathcal{G}$ there is an event x such that, for any unitary discretization function, $f(x) < f(x)$, which is impossible.

Conversely, if there are no cycles of positive weight, we may define a function f that maps each event x to the weight of the longest path in $\mathcal{G}$ that leads to x. By construction, $A_i \rightarrow B_j \implies f(A_i) < f(B_j)$, which is the forward implication of UD (definition 5). The other direction of UD follows by contraposition. Assume $A_i \not\rightarrow B_j$. If $A \rightrightarrows B$, we have $B_j \xrightarrow{0} A_i$ and thus, by the definition of f, that $f(B_j) \leq f(A_i)$. This gives $\neg(f(A_i) < f(B_j))$ as required. The other case, $A \not\rightrightarrows B$, is trivial. $\square$

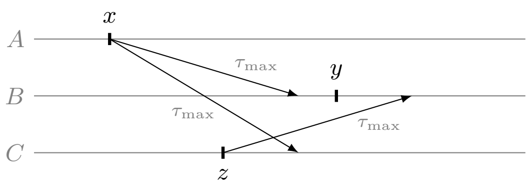
The unitary discretization described in the proof above is the most concise one and can be expressed as

$$f(x) = \max\ (\{f(y) + 1 \mid y \xrightarrow{1} x\} \cup \{f(z) \mid z \xrightarrow{0} x\} \cup \{0\}).$$

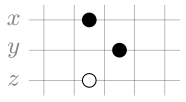
Other discretizations are constructed by adding 'extra' instants between process activations as in figure 2.

A. Discretizing general systems

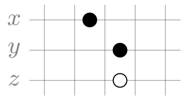
One might expect that real-time models are unitary discretizable if the transmission delays are 'significantly



(a) Real-time trace.



(b) $z \to y$.



(c) $x \to z$.

Fig. 4: A real-time trace that is not unitary discretizable ($x \xrightarrow{1} y \xrightarrow{0} z \xrightarrow{0} x$) and that may occur whenever $\tau_{\max} > 0$.

shorter' than the period of the process, that is $\tau_{\max} \ll T_{\min}$. Unfortunately this is not the case.

Theorem 1 (No General Unitary Discretization). *General real-time models with three processes or more communicating non-instantaneously are not unitary discretizable.*

Proof. If $\tau_{\max} > 0$, figure 4a shows a trace with a cycle of positive weight, $x \xrightarrow{1} y \xrightarrow{0} z \xrightarrow{0} x$, for which there is no unitary discretization (lemma 1). ☐

Figure 4 shows the two possible discretizations of the counterexample. In figure 4b the message sent at z should have been received at y ($z \to y$); whereas in figure 4c the message sent at x should have been received at z ($x \to z$). Neither correctly abstracts the real-time trace of figure 4a.

B. Recovering Soundness

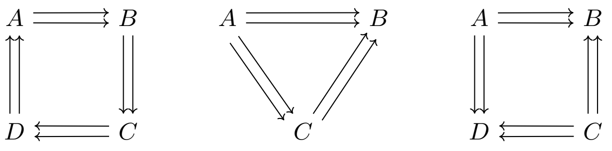
The counterexample of figure 4 shows that when three processes communicate such that $A \rightrightarrows B \leftleftarrows C \leftleftarrows A$, there is at least one trace that has no unitary discretization. Problematic cycles in traces can be prevented either by constraining the timing parameters of the model or by restricting communication graphs: forbidding $A \rightrightarrows B$ removes $A_i \xrightarrow{1} B_j$ and $B_j \xrightarrow{0} A_i$, for all i and j, in associated trace graphs (if $A \neq B$). We propose conditions that preclude cycles of positive weight in all possible traces and thus guarantee the existence of unitary discretizations.

Theorem 2. *Let L_c be the size of the longest elementary cycle in the communication graph. A real-time model satisfying definition 1 is unitary discretizable if and only if,*

1) *all u-cycles of the communication graph are cycles or balanced u-cycles, or $\tau_{\max} = 0$, and*
2) *there is no balanced u-cycle in the communication graph or $\tau_{\min} = \tau_{\max}$, and*
3) *there is no cycle in the communication graph or*

$$T_{\min} \geq L_c \tau_{\max}. \tag{CD}$$

A *u-cycle* is an elementary cycle in the undirected communication graph, that is, the graph obtained from the communication graph by forgetting the direction of the edges. A balanced u-cycle has the same number of edges in both directions. Figure 5 shows three examples of u-cycles, the rightmost one is also a balanced u-cycle. In the following $\mathcal{C}$, $u\mathcal{C}$, and $b\mathcal{C}$ denote the sets of cycles, u-cycles, and balanced u-cycles, respectively.



(a) cycle (b) u-cycle (c) b-cycle

Fig. 5: Examples of communication topologies.

In simpler terms, theorem 2 states that communication topologies containing u-cycles are only permissible if communication is perfectly instantaneous. Cycles can be allowed by imposing the additional constraint CD and balanced u-cycles can be allowed by imposing $\tau_{\min} = \tau_{\max}$. The following proposition is needed in the proof.

Proposition 2. *If a trace graph has a cycle of positive weight, then it has a cycle of positive weight of the form:*[3]

$$A^+ \xrightarrow{b_0} B^+ \xrightarrow{b_1} C^+ \xrightarrow{b_2} \dots \xrightarrow{b_n} A^+$$

where processes $A, B, C, \dots$ are pairwise distinct.

We write A^+ to denote successive activations of process A.

Proof. From any cycle of positive weight one can build another cycle of positive weight with the correct form. The proof is given in appendix B-A. ☐

We now present a proof sketch for theorem 2, the complete proof can be found in appendix B-A.

Proof. The proof is by contraposition in both directions. Using lemma 1 we have $\overline{\exists \mathrm{UD}} \iff \exists \mathrm{PC}$. Therefore we prove the following statement, which is equivalent to theorem 2.

$$\exists \mathrm{PC} \iff \left| \begin{array}{lr} \exists c \in \mathcal{C} \text{ and } \overline{\mathrm{CD}}, \text{ or,} & (\mathrm{C}_1) \\ \exists c \in b\mathcal{C} \text{ and } \tau_{\min} < \tau_{\max}, \text{ or,} & (\mathrm{C}_2) \\ \exists c \in u\mathcal{C} \setminus (\mathcal{C} \cup b\mathcal{C}) \text{ and } \tau_{\max} > 0 & (\mathrm{C}_3) \end{array} \right.$$

To prove that C_1 or C_2 or $\mathrm{C}_3 \implies \exists \mathrm{PC}$ we show that in each of the three possible cases one can build a trace with a cycle of positive weight. Figure 6 shows such a counterexample for a u-cycle of 5 processes.

To prove that $\exists \mathrm{PC} \implies \mathrm{C}_1$ or C_2 or C_3, suppose that there exists a trace with a cycle of positive weight. By proposition 2, there also exists a trace with a cycle of positive weight of the form:

$$A^+ \xrightarrow{b_0} B^+ \xrightarrow{b_1} C^+ \xrightarrow{b_2} \dots \xrightarrow{b_n} A^+, \tag{1}$$

where processes A, B, C, $\dots$ are pairwise distinct. By proposition 2, for two processes A and B, a transition

[3] $\xrightarrow{b_i}$ is used as a generic notation for either $\xrightarrow{1}$ or $\xrightarrow{0}$.

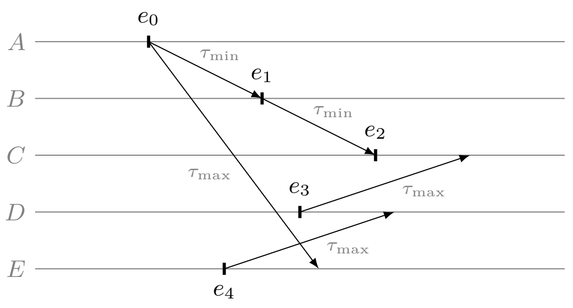


Fig. 6: Counterexample $e_0 \xrightarrow{1} e_1 \xrightarrow{1} e_2 \xrightarrow{0} e_3 \xrightarrow{0} e_4 \xrightarrow{0} e_0$, based on the u-cycle $A \rightrightarrows B \rightrightarrows C \leftleftarrows D \leftleftarrows E \leftleftarrows A$.

$A_i \xrightarrow{0} B_j$ corresponds to a communication channel $A \leftleftarrows B$ with $A \neq B$ and a transition $A_i \xrightarrow{1} B_j$ corresponds to a communication channel in the opposite direction $A \rightrightarrows B$ with the possibility that $A = B$ for activations of the same process. Since the processes of (1) are pairwise distinct, the sequence of processes $c = A, B, C, \ldots, A$ forms a u-cycle of the communication graph. There are three cases:

1) $c \in \mathcal{C}$ imposes $T_{\min} < L_c \tau_{\max}$ ($\overline{\text{CD}}$), hence C_1 holds.
2) $c \in b\mathcal{C}$ imposes $\tau_{\min} < \tau_{\max}$ and C_2 holds.
3) $c \in u\mathcal{C} \setminus (\mathcal{C} \cup b\mathcal{C})$ imposes $\tau_{\max} > 0$ and C_3 holds.

□

Theorem 1 is a particular case of theorem 2. Without assumptions on the communication graph there could be a u-cycle that is neither a cycle nor a balanced u-cycle.

Corollary 1 (2-process Unitary Discretization)**.** *A real-time model satisfying definition 1 with two processes can be unitarily discretized if and only if*

$$T_{\min} \geq 2\tau_{\max}. \tag{2D}$$

Proof. Direct consequence of theorem 2: for systems of two processes, $L_c = 2$ and CD becomes $T_{\min} \geq 2\tau_{\max}$. □

Two-process models were the focus of the original work on the quasi-synchronous approach [6] and they are relevant in practice [19], [20]. This result is coherent with Caspi's requirement that transmission delays be 'significantly shorter than the periods of [process activations]' [6, §3.2.1].

IV. The Quasi-Synchronous Abstraction

We now apply the preceding definitions and results on unitary discretizations to precisely describe when the quasi-synchronous model can be applied to a real-time system.

A discrete-time model is termed quasi-synchronous if 'it is not the case that a component process executes more than twice between two successive executions of another process' [7, §3.2]. Since any given process only detects the activations of another by receiving the corresponding messages, the quasi-synchronous condition corresponds to two constraints. For any process, 1) there are no more than two activations between two message receptions, and 2) there are no more than two message receptions between two activations. This definition can be formalized using unitary discretizations.

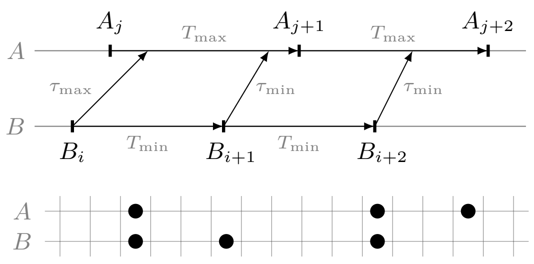


Fig. 7: A trace (above) and a possible discretization that violates definition 2.

Definition 7 (Quasi-Synchronous Model)**.** *A real-time model is quasi-synchronous if, for every trace t,*

1) *it has a unitary discretization f, and*
2) *for processes $A \leftleftarrows B$, there are no i and j such that*

$$\begin{aligned} &f(B_j) < f(A_i) < f(A_{i+2}) \leq f(B_{j+1}) \text{ or,} \\ &f(A_j) \leq f(B_i) < f(B_{i+2}) < f(A_{j+1}). \end{aligned} \tag{QS}$$

This definition expresses the two central features of quasi-synchrony: 1) communications as 'logical' unit delays, and 2) constraints on interleavings of process activations.

Condition QS is less constraining than definition 2 from section I-B. That definition, proposed by Caspi, has the advantage of forbidding a single symmetric subsequence, but the link with process interleavings is obscured. In fact, the proposition below shows that it is violated in any real-time system with unidirectional communications ($A \leftleftarrows B$ but $A \not\rightrightarrows B$) that is not perfectly synchronous. So, while definition 7 does not directly translate definition 2, we argue that it more faithfully describes quasi-synchronous systems in terms of process interleavings.

Proposition 3. *A pair of (real-time) processes A and B where $A \leftleftarrows B$ but $A \not\rightrightarrows B$ cannot be quasi-synchronous in the sense of definition 2 if $T_{\min} + \tau_{\min} < T_{\max} + \tau_{\max}$.*

Proof. If $T_{\min} + \tau_{\min} < T_{\max} + \tau_{\max}$, figure 7 shows an execution trace where $A_j \xrightarrow{0} B_i \xrightarrow{1} B_{i+1} \xrightarrow{1} A_{j+1} \xrightarrow{0} B_{i+2}$. A discretization f with $f(A_j) = f(B_i)$ and $f(A_{j+1}) = f(B_{i+2})$ is a valid unitary discretization that violates the condition of definition 2. □

While definition 7 conveys the essence of quasi-synchrony, its conditions are rather abstract. The following theorem incorporates the results of sections II and III to state concrete requirements on real-time parameters and communication topologies. Appendix B-B gives a full proof.

Theorem 3. *A real-time model satisfying definition 1 is quasi-synchronous (condition QS) if and only if,*

1) *the conditions of theorem 2 hold, and*
2) *the following condition holds,*

$$2T_{\min} + \tau_{\min} \geq T_{\max} + \tau_{\max}. \tag{QT}$$

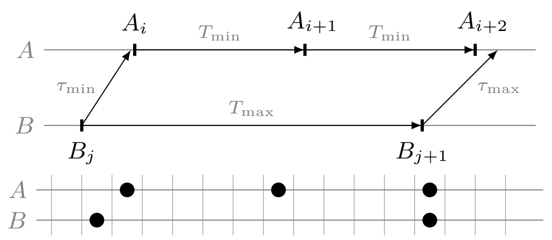


Fig. 8: Witness for QS $\implies$ QT.

Proof. The first condition ensures that the system is unitary discretizable. Now, if QS does not hold, there is a chain of events such that either

$$f(B_j) < f(A_i) < f(A_{i+2}) \le f(B_{j+1}) \text{ or,}$$
$$f(A_j) \le f(B_i) < f(B_{i+2}) < f(A_{j+1}).$$

This gives $B_j \to A_i$, and $B_{j+1} \not\to A_{i+2}$ in the first case; and, $B_i \not\to A_j$ and $B_{i+2} \to A_{j+1}$ in the second; which implies $2T_{\min} + \tau_{\min} < T_{\max} + \tau_{\max}$, that is, $\overline{\mathrm{QT}}$.

Conversely, if QT does not hold, then figure 8 shows a trace where $B_j \xrightarrow{1} A_i \xrightarrow{1} A_{i+1} \xrightarrow{1} A_{i+2} \xrightarrow{0} B_{j+1}$. Then, by definition 5, the discretization f such that

$$f(B_j) < f(A_i) < f(A_{i+2}) = f(B_{j+1}).$$

is a valid unitary discretization that violates QS. □

Theorem 3 states precisely when the quasi-synchronous abstraction is sound. If a real-time system satisfies the given constraints on (logical) topology and timing, then the quasi-synchronous abstraction can be used to formally verify its properties. To give a few concrete examples, providing condition QT holds, it applies to: 1) topologies without feedback, for example, three filter sequences connected to a triple voter; 2) trees of communicating pairs if $T_{\min} \ge 2\tau_{\max}$, for example, a 'daisy chain' or a star of intercommunicating neighbours; and 3) any feedback loop of n nodes if $T_{\min} \ge n\tau_{\max}$, for example, unidirectional ring networks or filters with 'non-overlapping' feedback loops. It does not apply if condition QT is violated, or in topologies with certain cycles, notably those with more than one path between two processes. Figures 8 and 9 in the appendices show a few examples of allowed and forbidden topologies.

V. Related Work

a) Distributed systems: The spectrum of formal models for distributed systems runs from completely synchronous (definition 1) to completely asynchronous [26]. The completely synchronous model makes the strongest timing assumptions—though they are not unreasonable for embedded systems—and it is possible to simulate round-based applications and solve problems like consensus and leader election even in the presence of failures [2], [10].

The impossibility of consensus in the asynchronous model [14] and the desire to treat more general systems than the synchronous model motivates the study of *partially synchronous* models [26, Part III]. There are models with bounds on transmissions and the relative speeds of processes, and these bounds are not necessarily known or may only hold eventually [12]. In the *θ-model* [33] bounds are not given on transmissions but rather on the ratio of the longest and shortest end-to-end delays of messages simultaneously in transit. The *Finite Average Response* time model [13] only assumes a lower bound on activations and a finite average response time for transmissions. Timing assumptions may also be allowed to vary across different communication links [1]. The *Asynchronous Bounded-Cycle model* [31] avoids any reference to transmission delays or bounds on activations and instead constrains the causality chains induced by transmission.

We treat the standard synchronous distributed systems model and our treatment of causality and timing constraints has nothing to do with recovering possibility results or determining algorithmic complexity in a partially synchronous model. We study a different question: when is a very specific discrete abstraction sound for the synchronous real-time model? Our main problem comes from the unusual but potentially advantageous modeling of transmissions as unit delays. This gives rise to a unique form of causality—the unitary discretization—that is relevant to the model-checking problem we consider but not to the theory of distributed computing.

Unsurprisingly, our notion of causality follows Lamport's seminal work [23]. His causality relation was recently extended to a *syncausality* relation [3, Definition 2] by using upper bounds on transmission delays to complete causality chains. Our causality relation is similar but message reception is not modeled explicitly, the *(recv)* clause is based on actual transmission delays not an upper bound, and transitivity is not avoided. The syncausality relation is developed into 'centipede' and 'centibroom' abstractions to study coordination problems, whereas we develop the unitary discretization to verify the soundness of a discrete model. Our approach is closer to work on *execution graphs* [31]: we also use a non-transitive relation and count along causality chains. But our trace graphs incorporate two types of constraints ($\xrightarrow{0}$ and $\xrightarrow{1}$) due to the different nature of the problem we study. Furthermore, the work on execution graphs focuses on asynchronous systems and does not propose constraining real-time parameters and communication topologies to eliminate cycles.

b) Logical clocks: As already mentioned in section III, the existence of a unitary discretization is equivalent to the problem of finding a strongly consistent scalar clock. As this is not possible in general [23], [30], research has sought more powerful mechanisms, like vector clocks [27] and matrix clocks [15], for capturing the causalities of events. These mechanisms do not resolve the problem posed in this paper, since the modeling of transmissions as unit delays and the activations of processes on boolean streams require the total ordering given by a global scalar clock: a synchronous modeling of an asynchronous system.

c) Quasi-synchrony: Most existing work on the quasi-synchronous abstraction either assumes instantaneous communication [5], [28]—which may be valid in a shared memory model but not a message-passing one—or takes the discrete model as given and applies it directly to model and analyze systems [19], [20], [32]. We seek to clarify the original definitions [6] and to precisely define the relation between the real-time and discrete-time models. This leads to the understanding of discretization in terms of causality and the restrictions on process intercommunications and timing which are the central contributions of this paper.

Our work is complementary to the development of abstract domains to statically analyze synchronous real-time systems [4], and to the verification of properties like maximal lost messages, message inversions, and message latency, in an interactive theorem prover [24], [25].

In *n-synchrony*, unlike in *quasi-synchrony*, the difference of cumulative process activation counts is bounded [9]. The relation between a similar model and real-time has recently been studied [11]. Both n-synchrony and quasi-synchrony can be related to 'clock bounds' and 'drift bounds' [32].

VI. Conclusion

The quasi-synchronous abstraction provides a way to model and reason about a class of distributed embedded systems whose processes communicate by sampling with bounded jitter. Given a real-time model satisfying certain constraints on timing parameters and communication topologies, properties obtained of the corresponding quasi-synchronous model are also true of the original model. In other words, a precise class of practically-relevant distributed control systems can be verified without resorting to timed formalisms and tools, and by modeling message transmission as a unit delay, but not all of them.

Acknowledgments

We thank Cesare Tinelli for sharing his experiences on model-checking quasi-synchronous models and Thibault Rieutord and our anonymous reviewers for their suggestions and comments.

References

[1] M. K. Aguilera, C. Delporte-Gallet, H. Fauconnier, and S. Toueg. Communication-efficient leader election and consensus with limited link synchrony. In *PODC*, pages 328–337, 2004.

[2] H. Attiya, C. Dwork, N. Lynch, and L. Stockmeyer. Bounds on the time to reach agreement in the presence of timing uncertainty. *JACM*, 41(1):122–152, 1994.

[3] I. Ben-Zvi and Y. Moses. Beyond Lamport's happened-before: On time bounds and the ordering of events in distributed systems. In *DISC*, pages 421–436, 2010.

[4] J. Bertrane. *Static analysis of communicating imperfectly-clocked synchronous systems using continuous-time abstract domains.* PhD thesis, École Polytechnique, 2008.

[5] S. Bhattacharyya, S. Miller, J. Yang, S. Smolka, B. Meng, C. Sticksel, and C. Tinelli. Verification of quasi-synchronous systems with Uppaal. In *DASC*, pages 8A4–1–8A4–12, 2014.

[6] P. Caspi. The quasi-synchronous approach to distributed control systems. Technical Report CMA/009931, Verimag, Crysis Project, 2000. "The Cooking Book".

[7] P. Caspi. Embedded control: From asynchrony to synchrony and back. In *EMSOFT*, pages 80–96, 2001.

[8] P. Caspi, C. Mazuet, and N. Reynaud Paligot. About the design of distributed control systems: The quasi-synchronous approach. In *SAFECOMP*, pages 215–226, 2001.

[9] A. Cohen, M. Duranton, C. Eisenbeis, C. Pagetti, F. Plateau, and M. Pouzet. N-synchronous Kahn networks: a relaxed model of synchrony for real-time systems. In *POPL*, pages 180–193, 2006.

[10] F. Cristian. Synchronous and asynchronous group communication (long version). *CACM*, 1996.

[11] A. Desai, S. A. Seshia, S. Qadeer, D. Broman, and J. C. Eidson. Approximate synchrony: An abstraction for distributed almost-synchronous systems. In *CAV*, pages 429–448, 2015.

[12] C. Dwork, N. Lynch, and L. Stockmeyer. Consensus in the presence of partial synchrony. *JACM*, 35(2):288–323, 1988.

[13] C. Fetzer, U. Schmid, and M. Süßkraut. On the possibility of consensus in asynchronous systems with finite average response times. In *ICDCS*, pages 271–280, 2005.

[14] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *JACM*, 32(2):374–382, 1985.

[15] M. J. Fischer and A. Michael. Sacrificing serializability to attain high availability of data in an unreliable network. In *PODS*, pages 70–75, 1982.

[16] G. Hagen and C. Tinelli. Scaling up the formal verification of Lustre programs with SMT-based techniques. In A. Cimatti and R. B. Jones, editors, *FMCAD*, pages 15:1–15:9, 2008.

[17] N. Halbwachs, P. Caspi, P. Raymond, and D. Pilaud. The synchronous dataflow programming language Lustre. *Proc. IEEE*, 79(9):1305–1320, 1991.

[18] N. Halbwachs, F. Lagnier, and C. Ratel. Programming and verifying real-time systems by means of the synchronous dataflow language LUSTRE. *IEEE Trans. Software Engineering*, 18(9):785–793, 1992.

[19] N. Halbwachs and L. Mandel. Simulation and verification of asynchronous systems by means of a synchronous model. In *ACSD*, pages 3–14, 2006.

[20] E. Jahier, N. Halbwachs, and P. Raymond. Synchronous modeling and validation of schedulers dealing with shared resources. Technical Report 2008-10, Verimag, 2008.

[21] H. Kopetz. *Real-time systems: design principles for distributed embedded applications.* Springer-Verlag, 2011.

[22] H. Kopetz and G. Bauer. The time-triggered architecture. *Proc. IEEE*, 91(1):112–126, 2003.

[23] L. Lamport. Time, clocks, and the ordering of events in a distributed system. *CACM*, 21(7):558–565, 1978.

[24] R. Larrieu and N. Shankar. A framework for high-assurance quasi-synchronous systems. In *MEMOCODE*, pages 72–83, 2014.

[25] W. Li, L. Gérard, and N. Shankar. Design and verification of multi-rate distributed systems. In *MEMOCODE*, pages 20–29, 2015.

[26] N. A. Lynch. *Distributed Algorithms.* Morgan Kaufmann, 1996.

[27] F. Mattern. Virtual time and global states of distributed systems. *Parallel and Distributed Algorithms*, 1(23):215–226, 1989.

[28] S. Miller, S. Bhattacharyya, C. Tinelli, S. Smolka, C. Sticksel, B. Meng, and J. Yang. Formal verification of quasi-synchronous systems. Technical report, DTIC Document, 2015.

[29] P. Raymond. Recognizing regular expressions by means of dataflow networks. In *ICALP*, pages 336–347, 1996.

[30] M. Raynal and M. Singhal. Logical time: Capturing causality in distributed systems. *IEEE Computer*, 29(2):49–56, 1996.

[31] P. Robinson and U. Schmid. The asynchronous bounded-cycle model. *TCS*, 412(1):5580–5601, 2011.

[32] G. Smeding and G. Goessler. A correlation preserving performance analysis for stream processing systems. In *MEMOCODE*, pages 11–20, July 2012.

[33] J. Widder and U. Schmid. The theta-model: achieving synchrony without clocks. *Distributed Computing*, 22(1):29–47, 2009.

Appendix A
Transmissions and the Happened Before Relation

Unlike Lamport, we do not close the $\rightarrow$ relation by transitivity. Hence $A_i \rightarrow B_j$ means that the message sent by process A at A_i is received by B strictly before B_j. Otherwise figure 1 shows an example with three events x, y and z, where the message sent by B to C at x is not received at z even though $x \rightarrow y \rightarrow z$.

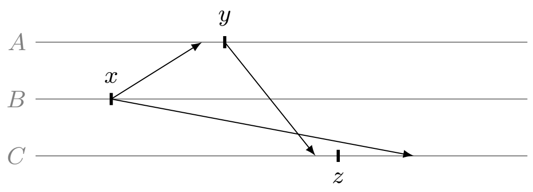


Fig. 1: Example trace where $x \not\rightarrow z$, but $x \rightarrow y \rightarrow z$.

Remark 1. The trace of figure 1 is not unitary discretizable. Indeed, $x \rightarrow y \rightarrow z$ imposes $f(x) < f(y) < f(z)$, but $x \not\rightarrow z$ also imposes $f(x) \geq f(z)$, which is impossible. This is a particular case of lemma 1. The corresponding trace graph is $x \xrightarrow{1} y \xrightarrow{1} z \xrightarrow{0} x$, that is, a cycle of positive weight. This counterexample is thus impossible under the assumptions of theorem 2. It requires $B \rightrightarrows A \rightrightarrows C \leftleftarrows A$, which is a u-cycle that is neither a cycle nor a balanced u-cycle.

Appendix B
Detailed Proofs

Notation. We write $T_i^A = t(A_{i+1}) - t(A_i)$ to denote the delay between the ith and $(i+1)$th activations of process A. For an event e_k we write P_k for the corresponding process.

A. Unitary Discretization

Proposition 2. *If a trace graph has a cycle of positive weight, then it has a cycle of positive weight of the form:*[4]

$$A^+ \xrightarrow{b_0} B^+ \xrightarrow{b_1} C^+ \xrightarrow{b_2} \dots \xrightarrow{b_n} A^+$$

where processes $A, B, C, \dots$ are pairwise distinct.

Remark 2. If there is a transition $P_i \xrightarrow{0} P_j$ in a block of activation P^+ we also have $P_i \xrightarrow{1} P_j$ by definition 6. To simplify the proofs we assume in the following that a block P^+ only contains $\xrightarrow{1}$ edges.

Proof. Consider a cycle of positive weight c that is not of the form:

$$A^+ \xrightarrow{b_0} B^+ \xrightarrow{b_1} C^+ \xrightarrow{b_2} \dots \xrightarrow{b_n} A^+$$

where processes $A, B, C, \dots$ are pairwise distinct. Then there exist two processes P and Q where $P \neq Q$, such that

$$c = e_0 \xrightarrow{b'_0} \dots \xrightarrow{b'_k} P_i \xrightarrow{b'_l} Q_k \xrightarrow{b'_m} \dots \xrightarrow{b'_n} P_j \xrightarrow{b'_p} \dots \xrightarrow{b'_q} e_0$$

[4] $\xrightarrow{b_i}$ is used as a generic notation for either $\xrightarrow{1}$ or $\xrightarrow{0}$.

Now i and j can be related in three different ways. For each there is another cycle of positive weight where P_i and P_j are regrouped into a block of successive activations P^+.[5]

- If $i < j$, we take $e_0 \xrightarrow{b'_0} \dots \xrightarrow{b'_k} P_{i\to j} \xrightarrow{b'_p} \dots e_0$.
- If $i > j$, we take $P_i \xrightarrow{b'_l} Q_k \xrightarrow{b'_m} \dots \xrightarrow{b'_n} P_{j\to i}$.
- If $i = j$, since c has a positive weight, one of the following two cycles has a positive weight:
  - $e_0 \xrightarrow{b'_0} \dots \xrightarrow{b'_k} P_i \xrightarrow{b'_p} \dots \xrightarrow{b'_q} e_0$, or
  - $P_i \xrightarrow{b'_l} Q_k \xrightarrow{b'_m} \dots \xrightarrow{b'_n} P_i$.

By iterating this result, we obtain a cycle of positive weight of the form:

$$A^+ \xrightarrow{b_0} B^+ \xrightarrow{b_1} C^+ \xrightarrow{b_2} \dots \xrightarrow{b_n} A^+$$

where processes $A, B, C, \dots$ are pairwise distinct. $\square$

Theorem 2. *Let L_c be the size of the longest elementary cycle in the communication graph. A real-time model satisfying definition 1 is unitary discretizable if and only if,*

1) *all u-cycles of the communication graph are cycles or balanced u-cycles, or $\tau_{\max} = 0$, and*
2) *there is no balanced u-cycle in the communication graph or $\tau_{\min} = \tau_{\max}$, and*
3) *there is no cycle in the communication graph or*

$$T_{\min} \geq L_c \tau_{\max}. \tag{CD}$$

Proof. The proof is by contraposition in both directions. Using lemma 1 we also have $\overline{\exists \text{UD}} \iff \exists\text{PC}$. Therefore we will prove the following result which is logically equivalent to theorem 2.

$$\exists\text{PC} \iff \left|\begin{array}{ll} \exists c \in \mathcal{C} \text{ and } \overline{\text{CD}}, \text{ or,} & (\text{C}_1) \\ \exists c \in b\mathcal{C} \text{ and } \tau_{\min} < \tau_{\max}, \text{ or,} & (\text{C}_2) \\ \exists c \in u\mathcal{C} \setminus (\mathcal{C} \cup b\mathcal{C}) \text{ and } \tau_{\max} > 0 & (\text{C}_3) \end{array}\right.$$

1) C$_1$ or C$_2$ or C$_3$ $\implies \exists$PC*:* We show that each condition C$_1$, C$_2$, or C$_3$, allows the construction of a trace that contains a cycle of positive weight.

a) C$_1$*:* Assume that

$$\exists c \in \mathcal{C} \text{ and } T_{\min} < L_c \tau_{\max} \tag{C$_1$}$$

Consider one of the longest cycles of the communication graph: $P_0 \leftleftarrows P_1 \leftleftarrows \dots \leftleftarrows P_{L_c} \leftleftarrows P_0$. We define $\varepsilon = (L_c \tau_{\max} - T_{\min})/L_c > 0$ and $\mathcal{E}$ a trace where for all events e the transmission delay is as long as possible, $\forall e \in \mathcal{E},\ \tau(e, \_) = \tau_{\max}$, and

$$\begin{aligned} t(e'_0) &= 0 \\ t(e_0) &= T_{\min} \\ t(e_{i+1}) &= t(e_i) - (\tau_{\max} - \varepsilon), \qquad \forall 0 \leq i \leq L_c \end{aligned}$$

with $P_{L_c+1} = P_0$. We thus have

$$t(e_{L_c+1}) = t(e'_0) + T_{\min} - L_c(\tau_{\max} - \varepsilon) = t(e'_0),$$

[5] $P_{i\to j}$ denotes the successive activations of P between P_i and P_j.

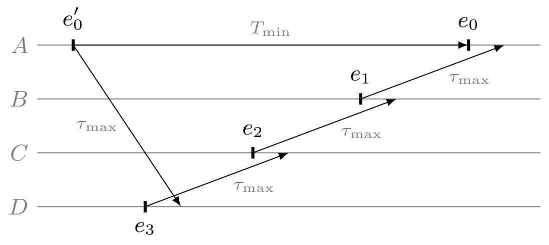


Fig. 2: Counterexample $e'_0 \xrightarrow{1} e_0 \xrightarrow{0} e_1 \xrightarrow{0} e_2 \xrightarrow{0} e_3 \xrightarrow{0} e'_0$, based on the cycle $A \leftleftarrows B \leftleftarrows C \leftleftarrows D \leftleftarrows A$.

that is, $e_{L_c+1} = e'_0$ and we have $\forall 0 \le i \le L_c$

$$\begin{aligned} &P_{i+1} \rightrightarrows P_i \\ &t(e_i) < t(e_{i+1}) + \tau(e_{i+1}, P_i). \end{aligned}$$

Hence $e'_0 \xrightarrow{1} e_0$ and $\forall 0 \le i \le L_c$ we have $e_i \xrightarrow{0} e_{i+1}$. This is a cycle of weight 1. Figure 2 gives an example of such a trace for four processes.

b) C$_2$, C$_3$*:* Suppose

$$\exists c \in b\mathcal{C} \text{ and } \tau_{\min} < \tau_{\max}, \text{ or,} \qquad (\text{C}_2)$$
$$\exists c \in u\mathcal{C} \setminus (\mathcal{C} \cup b\mathcal{C}) \text{ and } \tau_{\max} > 0. \qquad (\text{C}_3)$$

In both cases c is a chain of processes $P_0, \ldots, P_n, P_0$. Let p be the number of $\rightrightarrows$ edges, and q the number of $\leftleftarrows$ edges. One can assume without loss of generality that $q \ge p > 0$. Note that $p > 0$, otherwise c would be a cycle, contradicting the assumptions $c \in b\mathcal{C}$ or $c \in u\mathcal{C} \setminus (\mathcal{C} \cup b\mathcal{C})$.

Let $\varepsilon = (q\tau_{\max} - p\tau_{\min})/q$. In both cases, $\varepsilon > 0$. Indeed if $c \in b\mathcal{C}$ we would have $p = q$ but also $0 \le \tau_{\min} < \tau_{\max}$, and, conversely, if $c \in u\mathcal{C} \setminus (\mathcal{C} \cup b\mathcal{C})$ we have $q > p$ and $\tau_{\max} > 0$. Finally, let $\mathcal{E}$ be a trace where $t(e_0) = 0$ and $\forall 0 \le i \le n$,

$$P_i \rightrightarrows P_{i+1} \Longrightarrow \begin{cases} t(e_{i+1}) = t(e_i) + \tau_{\min} \\ \tau(e_i, P_{i+1}) = \tau_{\min} \end{cases}$$

$$P_i \leftleftarrows P_{i+1} \Longrightarrow \begin{cases} t(e_{i+1}) = t(e_i) - (\tau_{\max} - \varepsilon) \\ \tau(e_{i+1}, P_i) = \tau_{\max}, \end{cases}$$

with $P_{n+1} = P_0$. We thus have

$$t(e_{n+1}) = t(e_0) + p\tau_{\min} - q(\tau_{\max} - \varepsilon) = t(e_0),$$

that is, $e_{n+1} = e_0$ and $\forall 0 \le i \le n$

$$\begin{aligned} P_i \rightrightarrows P_{i+1} &\Longrightarrow t(e_{i+1}) \ge t(e_i) + \tau(e_i, P_{i+1}) \\ P_i \leftleftarrows P_{i+1} &\Longrightarrow \quad t(e_i) \quad < t(e_{i+1}) + \tau(e_{i+1}, P_i). \end{aligned}$$

Hence

$$\begin{aligned} P_i \rightrightarrows P_{i+1} &\Longrightarrow e_i \xrightarrow{1} e_{i+1} \\ P_i \leftleftarrows P_{i+1} &\Longrightarrow e_i \xrightarrow{0} e_{i+1}. \end{aligned}$$

This is a cycle of weight $p > 0$. Figure 3 gives an example of such a trace for five processes.

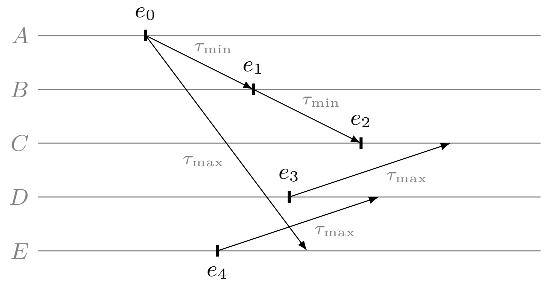


Fig. 3: Counterexample $e_0 \xrightarrow{1} e_1 \xrightarrow{1} e_2 \xrightarrow{0} e_3 \xrightarrow{0} e_4 \xrightarrow{0} e_0$ based on the u-cycle $A \rightrightarrows B \rightrightarrows C \leftleftarrows D \leftleftarrows E \leftleftarrows A$.

2) $\exists$PC $\Longrightarrow$ C$_1$ or C$_2$ or C$_3$*:* Suppose that there exists a trace that contains a cycle of positive weight. By proposition 2, there exists a trace with a cycle of positive weight of the form:

$$x = A^+ \xrightarrow{b_0} B^+ \xrightarrow{b_1} C^+ \xrightarrow{b_2} \ldots \xrightarrow{b_n} A^+ \qquad (2)$$

where the processes $A, B, C, \ldots$ are pairwise distinct. For two processes A and B, by proposition 2, a transition $A_i \xrightarrow{0} B_j$ corresponds to a communication channel $A \leftleftarrows B$, and a transition $A_i \xrightarrow{1} B_j$ corresponds to a communication channel in the opposite direction, $A \rightrightarrows B$, with the possibility that $A = B$ for activations of the same process. Since the processes of (2) are pairwise distinct, the sequence of processes $c = A, B, C \ldots, A$ forms a u-cycle in the communication graph.

We define $x = e_0 \xrightarrow{b_0} e_1 \xrightarrow{b_1} \ldots \xrightarrow{b_n} e_n \xrightarrow{b_{n+1}} e_0$ to refer to the particular activation e_k, and P_k as the corresponding process. Depending on the nature of c there are three cases to consider (see figure 4):

a) $c \in \mathcal{C}$ *(cycle):* Note that according to definition 1 we have $x \xrightarrow{1} y \Longrightarrow t(x) < t(y)$. This implies that a cycle of positive weight cannot contain only $\xrightarrow{1}$ edges. Furthermore if a cycle of positive weight is based on a cycle of the communication graph, an edge $\xrightarrow{1}$ can only reflect subsequent activations of the same process. Otherwise $\xrightarrow{0}$ and $\xrightarrow{1}$ edges correspond to communications in opposite directions and c cannot be a cycle.

Let p be the number of $\xrightarrow{1}$ edges and q the number of $\xrightarrow{0}$ edges in trace x. We have by definition of $\xrightarrow{1}$ and $\xrightarrow{0}$ that

$$\begin{aligned} e_i \xrightarrow{1} e_{i+1} \Longrightarrow t(e_{i+1}) &\ge t(e_i) + T_i^{P_i} \\ &\ge t(e_i) + T_{\min}, \text{ and} \end{aligned}$$

$$\begin{aligned} e_i \xrightarrow{0} e_{i+1} \Longrightarrow t(e_{i+1}) &> t(e_i) - \tau(e_{i+1}, P_i) \\ &\ge t(e_i) - \tau_{\max}. \end{aligned}$$

Following the cycle we have $t(e_0) + pT_{\min} - q\tau_{\max} < t(e_0)$ and hence $q\tau_{\max} > pT_{\min}$. By definition L_c is the maximum length of a cycle thus $L_c \ge q$ and x is a cycle of positive weight, that is, $p \ge 1$. Hence,

$$L_c\, \tau_{\max} \ge q\tau_{\max} > pT_{\min} \ge T_{\min}$$

which violates CD. Therefore C$_1$ holds.

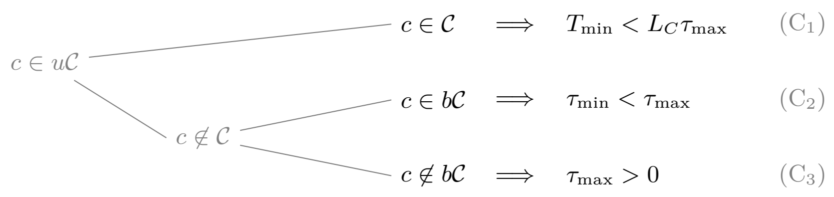

$c \in u\mathcal{C}$
$c \in \mathcal{C} \implies T_{\min} < L_C \tau_{\max}$ (C$_1$)
$c \notin \mathcal{C}$
$c \in b\mathcal{C} \implies \tau_{\min} < \tau_{\max}$ (C$_2$)
$c \notin b\mathcal{C} \implies \tau_{\max} > 0$ (C$_3$)

Fig. 4: Proof scheme for $\exists$PC $\implies$ C$_1$ or C$_2$ or C$_3$. There are three possible topologies for a u-cycle c, each implies one of the conditions C$_1$, C$_2$, or C$_3$.

b) $c \in b\mathcal{C}$ (balanced u-cycle): Let p be the number of edges $e_i \xrightarrow{1} e_{i+1}$ in x such that $P_i \neq P_{i+1}$ (message transmission), r the number of edges $e_i \xrightarrow{1} e_{i+1}$ such that $P_i = P_{i+1}$ (subsequent activations of the same process, denoted here by $\xrightarrow{1}_N$), and q the number of edges $\xrightarrow{0}$. By definition of $\xrightarrow{1}$, $\xrightarrow{1}_N$, and $\xrightarrow{0}$, we have

$$\begin{aligned} e_i \xrightarrow{1} e_{i+1} \implies t(e_{i+1}) &\geq t(e_i) + \tau(e_i, P_{i+1}) \\ &\geq t(e_i) + \tau_{\min} \end{aligned}$$

$$\begin{aligned} e_i \xrightarrow{1}_N e_{i+1} \implies t(e_{i+1}) &\geq t(e_i) + T_i^{P_i} \\ &\geq t(e_i) + T_{\min} \end{aligned}$$

$$\begin{aligned} e_i \xrightarrow{0} e_{i+1} \implies t(e_{i+1}) &> t(e_i) - \tau(e_{i+1}, P_i) \\ &\geq t(e_i) - \tau_{\max}. \end{aligned}$$

Along the cycle x: $t(e_0) + p\tau_{\min} + rT_{\min} - q\tau_{\max} < t(e_0)$. Since $c \in b\mathcal{C}$, we also have $p = q$, thus $p(\tau_{\min} - \tau_{\max}) + rT_{\min} < 0$ which imposes $\tau_{\min} < \tau_{\max}$. Therefore C$_2$ holds.

c) $c \in u\mathcal{C} \setminus (\mathcal{C} \cup b\mathcal{C})$ (general u-cycle): In the same notation, we also have $t(e_0) + p\tau_{\min} + rT_{\min} - q\tau_{\max} < t(e_0)$, that is, $p\tau_{\min} + rT_{\min} - q\tau_{\max} < 0$. Since $\tau_{\min}, T_{\min} \geq 0$ and $p, q, r \geq 0$ this implies $\tau_{\max} > 0$. Hence C$_3$ holds. □

B. The Quasi-Synchronous Abstraction

Theorem 3. *A real-time model satisfying definition 1 is quasi-synchronous (condition QS) if and only if,*

1) *the conditions of theorem 2 hold, and*
2) *the following condition holds,*

$$2T_{\min} + \tau_{\min} \geq T_{\max} + \tau_{\max}. \tag{QT}$$

Proof. Consider a pair of communicating processes A and B where $A \leftleftarrows B$.

To show that QT $\implies$ QS, assume that QS does not hold, that is, there exist i and j such that

$$\begin{aligned} &f(B_j) < f(A_i) < f(A_{i+2}) \leq f(B_{j+1}) \text{ or,} \\ &f(A_j) \leq f(B_i) < f(B_{i+2}) < f(A_{j+1}). \end{aligned}$$

In the first case we have $B_j \to A_i$ and $B_{j+1} \not\to A_{i+2}$. Then, from definition 4,

$$\begin{aligned} t(A_i) &\geq t(B_j) + \tau(B_j, A) \\ t(A_{i+2}) &< t(B_{j+1}) + \tau(B_{j+1}, A) \\ t(B_{j+1}) &= t(B_j) + T_j^B \\ t(A_{i+2}) &= t(A_i) + T_i^A + T_{i+1}^A \end{aligned}$$

From definition 3 we obtain:

$$\begin{aligned} t(A_i) &\geq t(B_j) + \tau_{\min} \\ t(A_{i+2}) &< t(B_{j+1}) + \tau_{\max} \\ t(B_{j+1}) &\leq t(B_j) + T_{\max} \\ t(A_{i+2}) &\geq t(A_i) + 2T_{\min} \end{aligned}$$

Giving

$$\begin{aligned} t(B_j) + \tau_{\min} + 2T_{\min} &\leq t(A_i) + 2T_{\min} \\ &\leq t(A_{i+2}) \\ &< t(B_{j+1}) + \tau_{\max} \\ &\leq t(B_j) + T_{\max} + \tau_{\max} \end{aligned}$$

Hence $2T_{\min} + \tau_{\min} < \tau_{\max} + T_{\max}$, that is $\overline{\text{QT}}$.

The second case is similar. We have $B_i \not\to A_j$ and $B_{i+2} \to A_{j+1}$. Then, from definition 4,

$$\begin{aligned} t(A_j) &< t(B_i) + \tau(B_i, A) \\ t(A_{j+1}) &\geq t(B_{i+2}) + \tau(B_{i+2}, A) \\ t(A_{j+1}) &= t(A_j) + T_j^A \\ t(B_{i+2}) &= t(B_i) + T_i^B + T_{i+1}^B \end{aligned}$$

From definition 3 we obtain:

$$\begin{aligned} t(A_j) &< t(B_i) + \tau_{\max} \\ t(A_{j+1}) &\geq t(B_{i+2}) + \tau_{\min} \\ t(A_{j+1}) &\leq t(A_j) + T_{\max} \\ t(B_{i+2}) &\geq t(B_i) + 2T_{\min} \end{aligned}$$

Giving

$$\begin{aligned} t(A_j) + 2T_{\min} + \tau_{\min} &< t(B_i) + \tau_{\max} + 2T_{\min} + \tau_{\min} \\ &\leq t(B_{i+2}) + \tau_{\min} + \tau_{\max} \\ &\leq t(A_{j+1}) + \tau_{\max} \\ &\leq t(A_j) + T_{\max} + \tau_{\max} \end{aligned}$$

Hence $2T_{\min} + \tau_{\min} < \tau_{\max} + T_{\max}$, that is $\overline{\text{QT}}$.

On the other hand, if QT does not hold:

$$2T_{\min} + \tau_{\min} < T_{\max} + \tau_{\max},$$

figure 5 shows a trace where

$$B_j \xrightarrow{1} A_i \xrightarrow{1} A_{i+1} \xrightarrow{1} \ldots \xrightarrow{1} A_{i+n} \xrightarrow{0} B_{j+m-1}.$$

Then a discretization f such that $f(B_j) < f(A_i)$ and $f(A_{i+2}) = f(B_{j+1})$ is a valid unitary discretization which violates QS. □

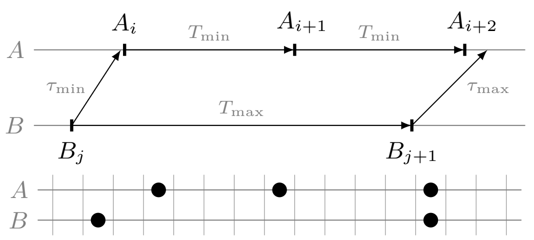


Fig. 5: Witness for QS $\Longrightarrow$ QT.

Appendix C
Multirate Systems

In this section we extend our results to multirate systems. Definition 1 becomes:

Definition 8 (Synchronous Real-Time Model). *A synchronous real-time model is a finite set of processes $\mathcal{P}$, where for each process P, the delay T^P between two successive activations is bounded.*

$$0 \le T^P_{\min} \le T^P \le T^P_{\max}. \tag{RP}$$

Values are transmitted between processes with a delay $\tau \in \mathbb{R}$, bounded by $\tau_{\min}$ and $\tau_{\max}$.

$$0 \le \tau_{\min} \le \tau \le \tau_{\max}. \tag{RT}$$

We still assume global bounds on the transmission delay. This generalization requires very few changes to the results. Theorem 2 becomes

Theorem 4. *A real-time model satisfying definition 1 is unitary discretizable if and only if,*

1) *all u-cycles of the communication graph are cycles or balanced u-cycles, or $\tau_{\max} = 0$,*
2) *there is no balanced u-cycle in the communication graph, or $\tau_{\min} = \tau_{\max}$,*
3) *there is no cycle in the communication graph, or for all cycle c*

$$T^c_{\min} \ge L_c \tau_{\max} \tag{CD}$$

where $L_c = \text{size}(c)$, and $T^c_{\min} = \min\{T^P_{\min} \mid P \in c\}$.

Proof. The proof is similar to that of theorem 2. The only difference is the treatment of cycles. Condition C_1 becomes:

$$\exists c \in \mathcal{C} \text{ such that } T^c_{\min} < L_c \tau_{\max} \tag{C$_1$}$$

Assume that C_1 holds. Let c be such a cycle

$$c = P_0 \leftleftarrows P_1 \leftleftarrows \ldots \leftleftarrows P_{L_c} \leftleftarrows P_0$$

where P_0 is the process with the smallest lower bound in c,

$$T^{P_0}_{\min} = T^c_{\min}.$$

Following the proof of theorem 2, we show it is possible to build a trace with a cycle of positive weight based on c. Let $\varepsilon = (L_c\tau_{\max} - T_{\min})/L_c > 0$ and $\mathcal{E}$ be a trace where for all events e_i the transmission delay is as long as possible, $\forall e \in \mathcal{E},\ \tau(e, \_) = \tau_{\max}$, and

$$\begin{aligned} t(e'_0) &= 0 \\ t(e_0) &= T^{P_0}_{\min} = T^c_{\min} \\ t(e_{i+1}) &= t(e_i) - (\tau_{\max} - \varepsilon), \qquad \forall 0 \le i \le L_c \end{aligned}$$

with $P_{L_c+1} = P_0$. We thus have

$$t(e_{L_c+1}) = t(e'_0) + T^c_{\min} - L_c(\tau_{\max} - \varepsilon) = t(e'_0),$$

that is, $e_{L_c+1} = e'_0$ and we have $\forall 0 \le i \le L_c$,

$$\begin{aligned} &P_{i+1} \rightrightarrows P_i \\ &t(e_i) < t(e_{i+1}) + \tau(e_{i+1}, P_i). \end{aligned}$$

which is a cycle of weight 1.

On the other hand suppose there exists a cycle of positive weight of the form

$$x = A^+ \xrightarrow{b_0} B^+ \xrightarrow{b_1} C^+ \xrightarrow{b_2} \ldots \xrightarrow{b_n} A^+ \tag{3}$$

such that the sequence $c = A, B, C \ldots, A$ is a cycle.

Let p be the number of $\xrightarrow{1}$ edges and q the number of $\xrightarrow{0}$ edges in trace x. If a cycle of positive weight is based on a cycle of the communication graph, an edge $\xrightarrow{1}$ can only reflect subsequent activations of the same process. Hence $L_c = \text{size}(c) = q$. We have by definition of $\xrightarrow{1}$ and $\xrightarrow{0}$, that

$$\begin{aligned} e_i \xrightarrow{1} e_{i+1} \Longrightarrow t(e_{i+1}) &\ge t(e_i) + T^{P_i}_i \\ &\ge t(e_i) + T^{P_i}_{\min} \\ &\ge t(e_i) + T^c_{\min}, \text{ and} \\ e_i \xrightarrow{0} e_{i+1} \Longrightarrow t(e_{i+1}) &> t(e_i) - \tau(e_{i+1}, P_i) \\ &\ge t(e_i) - \tau_{\max}. \end{aligned}$$

Following the cycle we have $t(e_0) + pT^c_{\min} - q\tau_{\max} < t(e_0)$ and hence $q\tau_{\max} > pT_{\min}$. Since x is a cycle of positive weight, $p \ge 1$. Hence,

$$L_c\ \tau_{\max} = q\tau_{\max} > pT^c_{\min} \ge T^c_{\min}$$

which violates CD. Therefore C_1 holds. □

We now apply the preceding definitions and results on unitary discretizations to generalize the quasi-synchronous model to multi-rate systems after the work of [28], [32].

A discrete-time model is termed n/m-quasi-synchronous if there are no more than n activations of one process between m successive activations of another. Since any given process only detects the activations of another by receiving the corresponding messages, the quasi-synchronous condition corresponds to two constraints. For any two processes, 1) there are no more than n activations between m message receptions, and 2) there are no more than n message receptions between m activations. This definition can be formalized using unitary discretizations.

Definition 9 (n/m-Quasi-Synchronous Model). *A real-time model is n/m-quasi-synchronous with $n \ge m > 1$ if, for every trace t,*

1) it has a unitary discretization f, and
2) for processes $A \leftleftarrows B$, there is no chain of activations of length greater than n, that is, no i and j such that

$$f(B_j) < f(A_i) < \cdots < f(A_{i+n}) \leq f(B_{j+m-1}) \text{ or,} \\ f(A_j) \leq f(B_i) < \cdots < f(B_{i+n}) < f(A_{j+m-1}). \tag{QS}$$

Remark 3. Definition 7 of section IV is a particular case of definition 9 with $n = m = 2$.

Theorem 5. *A real-time model satisfying definition 1 is n/m-quasi-synchronous (condition QS) if and only if,*
1) the conditions of theorem 2 hold, and
2) the following conditions hold,

$$nT^A_{\min} + \tau_{\min} \geq (m-1)T^B_{\max} + \tau_{\max} \tag{QT$_1$}$$
$$nT^B_{\min} + \tau_{\min} \geq (m-1)T^A_{\max} + \tau_{\max} \tag{QT$_2$}$$

QT_1 ensures that there are no more that n activations of A between m message receptions from B, and QT_2 ensures that there are no more than n message receptions from B between m activations of A.

Proof. The proof follows that of theorem 3. For $n \geq m > 1$, consider a pair of communicating processes A and B where $A \leftleftarrows B$. To show that QT_1 and $\mathrm{QT}_2 \implies \mathrm{QS}$, assume that QS does not hold, there then exist i and j such that

$$f(B_j) < f(A_i) < \cdots < f(A_{i+n}) \leq f(B_{j+m-1}) \text{ or,} \\ f(A_j) \leq f(B_i) < \cdots < f(B_{i+n}) < f(A_{j+m-1}).$$

In the first case we have $B_j \to A_i$ and $B_{j+m-1} \not\to A_{i+n}$. Then, from definitions 3 and 4 we obtain:

$$\begin{array}{rcl} t(A_i) & \geq & t(B_j) + \tau_{\min} \\ t(A_{i+n}) & \leq & t(B_{j+m-1}) + \tau_{\max} \\ t(B_{j+m-1}) & \leq & t(B_j) + (m-1)T^B_{\max} \\ t(A_{i+n}) & \geq & t(A_i) + nT^A_{\min} \end{array}$$

Giving

$$\begin{aligned} t(B_j) + \tau_{\min} + nT^A_{\min} &\leq t(A_i) + nT^A_{\min} \\ &\leq t(A_{i+n}) \\ &< t(B_{j+m-1}) + \tau_{\max} \\ &\leq t(B_j) + (m-1)T^B_{\max} + \tau_{\max} \end{aligned}$$

Hence $nT^A_{\min} + \tau_{\min} < \tau_{\max} + (m-1)T^B_{\max}$, that is $\overline{\mathrm{QT}_1}$.

The second case is similar. We have $B_i \not\to A_j$ and $B_{i+n} \to A_{j+m-1}$. Then, from definitions 3 and 4 we obtain:

$$\begin{array}{rcl} t(A_j) & \leq & t(B_i) + \tau_{\max} \\ t(A_{j+m-1}) & \geq & t(B_{i+n}) + \tau_{\min} \\ t(A_{j+m-1}) & \leq & t(A_j) + (m-1)T^A_{\max} \\ t(B_{i+n}) & \geq & t(B_i) + nT^B_{\min} \end{array}$$

Giving

$$\begin{aligned} t(A_j) + nT^B_{\min} + \tau_{\min} &< t(B_i) + \tau_{\max} + nT^B_{\min} + \tau_{\min} \\ &\leq t(B_{i+n}) + \tau_{\min} + \tau_{\max} \\ &\leq t(A_{j+m-1}) + \tau_{\max} \\ &\leq t(A_j) + (m-1)T^A_{\max} + \tau_{\max} \end{aligned}$$

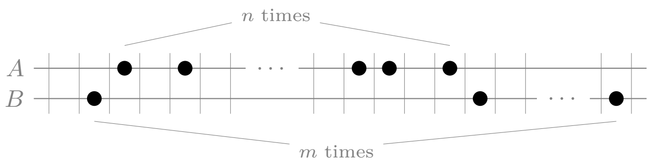


Fig. 6: Maximal overwrites and oversamplings.

Hence $nT^B_{\min} + \tau_{\min} < \tau_{\max} + (m-1)T^A_{\max}$, that is $\overline{\mathrm{QT}_2}$.

Conversely, if either of the conditions QT_1 or QT_2 does not hold, figure 7 shows traces and valid unitary discretizations that violate QS. ☐

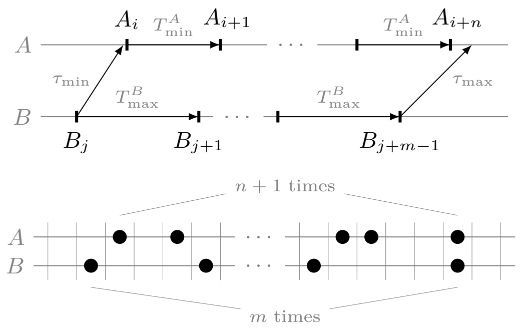
A classic property of the quasi-synchronous abstraction is the existence of bounds on the numbers of successive overwrites (message losses) and oversamplings (message duplications) [6, §3.2.3]. These properties follow directly from the n/m-quasi-synchronous model (definition 7). Figure 6 shows the worst acceptable case: a chain of n activations of a process A between two successive activations of another process B and then no activation of A before $m-1$ activations of B. This trace respects QS.

Proposition 4 (Overwrites, Oversamples)**.** *The maximum number of successive overwrites or oversamplings in an n/m-quasi-synchronous system is $n-1$.*

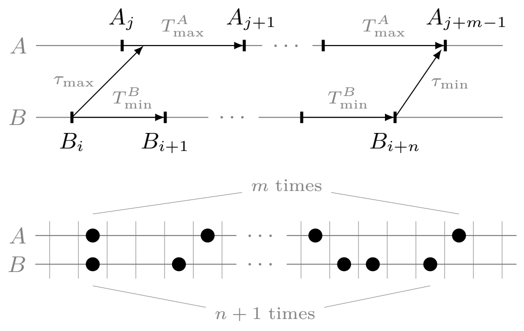
Proof. Consider a pair of processes A and B such that $A \rightrightarrows B$ and $A \leftleftarrows B$. The proof is straightforward given definition 5 and the worst acceptable case shown in figure 6:

$$B_j \to A_i \to A_{i+1} \to \ldots \to A_{i+n-1} \to B_{j+1}.$$

For the maximum number of overwrites, the $n-1$ messages sent between A_i and A_{i+n-2} are overwritten by the message sent at A_{i+n-1} which is received by B at B_{j+1}. Symmetrically, for the maximum number of oversamplings, the $n-1$ activations of A between A_{i+1} and A_{i+n-1} oversample the value sent by B at B_j which is already received by A at A_i. ☐



(a) QS $\implies$ QT$_1$



(b) QS $\implies$ QT$_2$

Fig. 7: Witness for QS $\implies$ (QT$_1$ and QT$_2$) and the associated discretizations.

$$\begin{tikzcd}
A \arrow[r, Rightarrow] & B & C \arrow[l, Rightarrow] \\
D \arrow[u, Rightarrow] & E \arrow[u, Rightarrow] & F \arrow[u, Rightarrow]
\end{tikzcd}$$

(a) Voter

$$\begin{tikzcd}
A \arrow[r, Leftrightarrow] & B \arrow[r, Leftrightarrow] & C \arrow[d, Leftrightarrow] \\
D \arrow[r, Leftrightarrow] & E \arrow[r, Leftrightarrow] & F
\end{tikzcd}$$

(b) Daisy chain
$T_{\min} \geq 2\tau_{\max}$

$$\begin{tikzcd}
A \arrow[r, Leftrightarrow] & B \arrow[r, Leftrightarrow] \arrow[dl, Leftrightarrow] \arrow[d, Leftrightarrow] \arrow[dr, Leftrightarrow] & C \\
D & E & F
\end{tikzcd}$$

(c) Star
$T_{\min} \geq 2\tau_{\max}$

$$\begin{tikzcd}
A \arrow[r, Rightarrow] & B \arrow[r, Rightarrow] & C \arrow[d, Rightarrow] \\
D \arrow[u, Rightarrow] & E \arrow[l, Rightarrow] & F \arrow[l, Rightarrow]
\end{tikzcd}$$

(d) Unidirectional ring
$T_{\min} \geq 6\tau_{\max}$

Fig. 8: Some examples of allowed communication topologies and associated timing constraints.
In all these cases, condition $2T_{\min} + \tau_{\min} \geq T_{\max} + \tau_{\max}$ holds.

$$\begin{tikzcd}
A \arrow[r, Rightarrow] \arrow[d, Rightarrow] & B \arrow[r, Rightarrow] & C \\
D \arrow[r, Rightarrow] & E \arrow[r, Rightarrow] & F \arrow[u, Rightarrow]
\end{tikzcd}$$

(a) u-cycle
$\tau_{\max} = 0$

$$\begin{tikzcd}
A \arrow[r, Rightarrow] \arrow[d, Rightarrow] & B \arrow[r, Rightarrow] & C \arrow[d, Rightarrow] \\
D \arrow[r, Rightarrow] & E \arrow[r, Rightarrow] & F
\end{tikzcd}$$

(b) b-cycle
$\tau_{\max} = \tau_{\min}$

$$\begin{tikzcd}
A \arrow[d, Rightarrow] & B \arrow[l, Rightarrow] \arrow[r, Rightarrow] & C \arrow[dl, Rightarrow] \\
D \arrow[r, Rightarrow] & E \arrow[u, Rightarrow] &
\end{tikzcd}$$

(c) Cycle composition
$\tau_{\max} = 0$

$$\begin{tikzcd}
A \arrow[r, Leftrightarrow] \arrow[d, Leftrightarrow] & B \arrow[r, Leftrightarrow] & C \arrow[d, Leftrightarrow] \\
D \arrow[r, Leftrightarrow] & E \arrow[r, Leftrightarrow] & F
\end{tikzcd}$$

(d) Bidirectional ring
$\tau_{\max} = 0$

Fig. 9: Some examples of forbidden communication topologies and associated timing constraints.
In each of these cases, there is more than one path between two processes.