Soundness of the Quasi-Synchronous Abstraction

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

*DI, École normale supérieure
†Inria, Paris-Rocquencourt
‡Univ. Pierre et Marie Curie

***Abstract*—We study the link between real-time *quasi-periodic architectures* where computing units execute 'almost periodically' and the discrete-time *quasi-synchronous abstraction* that P. Caspi proposed for analyzing them. The simplicity of the abstraction is appealing: the only events are node activations; logical steps account for transmission delays; and no node may be activated more than twice between two successive activations of any other. The motivation is to verify properties of real-time distributed systems in the simpler discrete model.**

By formalizing the relation between quasi-periodic architectures and the quasi-synchronous abstraction using L. Lamport's *happened before relation*, we show that the abstraction is sound for systems of two nodes. After showing that the abstraction is not sound for general systems with three or more nodes, we give sufficient restrictions on communication topologies to recover soundness.

I. Introduction

Quasi-periodic architectures arise naturally whenever computing units executing periodically are connected together by network links. Computing units are triggered at regular intervals with an eventual jitter and communicate by sending messages to be later sampled by receivers. This simple and robust scheme arose naturally in the historical development of distributed embedded controllers [3, §2]. It is commonly employed in critical aerospace, power, and rail systems.

The *quasi-synchronous approach* [3], [5] 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 [6], [7] designs. One of the key ideas is to model the computing units, network links, and shared memories themselves as a synchronous program [3, §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 [8] and to the analysis of systems specified in the Architecture Analysis and Design Language (AADL) [2], [9].

An alternative way of developing applications for quasi-periodic architectures is to synchronize process executions across nodes. The Time-Triggered Architecture (TTA) [10], [11] thoroughly develops this approach and there are several clock synchronization protocols suitable for use in 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 completely eliminated or greatly 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.

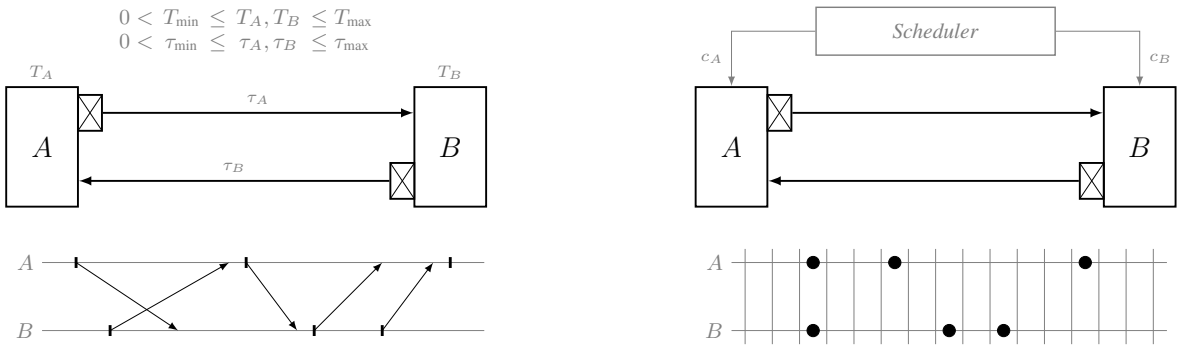
Figure 1 gives an overview of the quasi-synchronous approach and the key elements of this paper. At left is a real-time model comprising two nodes, A and B, communicating through network links. The nodes and links are annotated with timing parameters that are explained in the next section. Underneath is an example trace, showing the activations of nodes and the corresponding message transmissions. At right is a discrete-time abstraction. Timing parameters have been replaced by a discrete program called *Scheduler* that overapproximates their effect by controlling node activations. 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. 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 node 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 by introducing a notion of *unitary discretization* based on the respective causality relations of the two models. Using this tool, we show that the abstraction is sound for systems of two nodes but not for more general systems. We give sufficient conditions on communication topologies to ensure soundness.

A. The Real-time Model

The quasi-synchronous approach exploits two facts about quasi-periodic architectures. 1) Even though the actual time between executions of a process will deviate from the nominal period due to clock drift and other system activity, this *jitter* is bounded. 2) Communication delays are bounded. 3) Many control applications are robust to variations in sampling time and even occasional lost or oversampled values, and subsystems involving discrete logic or state changes can often be adapted



(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, $RT \models \varphi \iff DT \models \varphi$.

to accommodate such effects, albeit with the introduction of additional delays.

Definition 1 (Quasi-Periodic Architecture)**.** *A* quasi-periodic architecture *is a finite set of computing units or nodes $\mathcal{N}$, where every node $A \in \mathcal{N}$ executes periodically but the actual time between any two activations $T \in \mathbb{R}$ may vary between known bounds during an execution:*

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

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

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

Each is buffered at receivers until a newer value is received.

This definition comes from Caspi [3, §2.6], but rather than 'short undetermined transmission delays' [3, §3.2.1], we require a strictly non-zero lower bound and a fixed upper bound. This addition is significant, and anyway, we do not think $\tau_{\min} = 0$ or $T_{\min} = 0$ are reasonable assumptions for real distributed systems. We also assume that each activation completes within the bounds $T_{\min}$ and $T_{\max}$. The definition is very general and applies to a large class of distributed, real-time, systems.

B. The Discrete-time Model

The simplest discrete abstraction is to ignore time altogether and to model arbitrary interleavings of node activations. This is sound but far from complete: many properties that hold of the architecture cannot be shown in the model. Furthermore, the many possible interleavings complicate reasoning about or model-checking the discrete-time model.

A finer abstraction was proposed by Caspi. He realized that the interleavings of systems satisfying equation (RP) can be constrained [4, §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 read and write clocks' they can be modeled by unit delays on the base clock of the discrete-time model, but that 'if longer transmission delays are needed, modeling should be more complex' [3, §3.2.1]. A unit delay is a mechanism for modeling the fact that a message sent at one logical instant is received at the next instant. More complex modeling essentially means modeling the transmission medium and its activations as a separate component [1], [8], which introduces additional interleavings.

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

Definition 2 (Quasi-Synchronous Model)**.** *A* quasi-synchronous model *comprises a scheduler and finite set of nodes $\mathcal{N}$. The scheduler is connected to each node by a discrete clock signal. It activates the nodes non-deterministically but ensures that no pair of clock signals (c_A, c_B), for a pair of nodes $A, B \in N$, 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 t or f. Nodes communicate through unit delays activated at every scheduler tick.

The restriction on subsequences of pairs of clock signals [3, §3.2.2] expresses formally the constraint quoted above. The forbidden subsequence involves at least three activations of one node (A) between two successive activations of another (B). A finite state scheduler that produces valid sequences is readily constructed from the given regular expression. The nodes and unit delays can be modeled directly in Lustre [7], for instance, and verified by model-checking [2], [8], [9].

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 define a discretization function that relates any execution of the real-time model to an execution of the discrete-time model (section II). We show how this function characterizes the link between the causality relations of the two models (section III). This function is quite constrained due to the modeling of inter-node communications as unit delays, but it still allows for the treatment of practically-relevant systems of two nodes [8], [9] an those without cyclic or 'cross-over' communication topologies (section IV). We prove that it is sound (section V).

Most of the definitions and proofs within this paper have been mechanized in the Isabelle/HOL Interactive Theorem Prover (ITP) [13]. They are available online[1]. Mechanized definitions and properties are indicated by the symbol ✓. More traditional proofs can also be found in the appendix.

II. Traces and Causality

We define a simple model for reasoning about quasi-periodic architectures and their discretization. It has two components: (real-time) traces and their induced causality relations. For the remainder of the paper, we fix an arbitrary real-time model with nodes $\mathcal{N}$ and parameters T_{min}, T_{max}, τ_{min}, and τ_{max} that satisfy definition 1.

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

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

Both $t(A_i)$ *and* $\tau(A_i)$ *give non-negative reals satisfying the constraints of definition 1, namely,*

$$0 < T_{min} \leq t(A_{i+1}) - t(A_i) \leq T_{max}, \text{ and}$$
$$0 < \tau_{min} \leq \tau(A_i) \leq \tau_{max}.$$

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 node A.

For now, we do not make any assumptions on which nodes intercommunicate, we consider that a node sends a message at each activation and that it is potentially received at all other nodes after an associated transmission delay. Assigning a different transmission delay for each receiver would complicate the model without altering its fundamental properties. The possibility that messages are not sent to all nodes is discussed in section IV.

The causality relation between events within a given trace is defined using the *happened before* relation of Lamport [12]. Unlike Lamport, we do not explicitly model message reception. A message is received if the next execution of the receiver occurs after the corresponding transmission delay.

[1]http://www.di.ens.fr/~baudart/fmcad2015/

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* $A_i \rightarrow A_j$*,*
(recv) *If* $t(A_i) + \tau(A_i) \leq t(B_j)$ *then* $A_i \rightarrow B_j$*, and*
(trans) *If* $A_i \rightarrow B_j$ *and* $B_j \rightarrow C_k$ *then* $A_i \rightarrow C_k$*.*

Activations at a single node are totally ordered (*local*); an activation at one node happens before an activation at another node when a message sent at the former is received before being sampled at the latter (*recv*); and this ordering is closed by transitivity (*trans*).

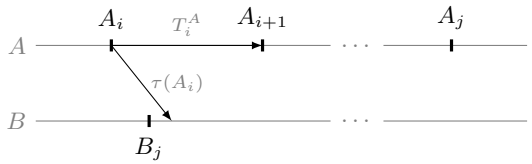


Fig. 2: $A_i \not\rightarrow B_j$ and $A_i \rightarrow A_j$.

The causality relation $\rightarrow$ induced by a trace implies bounds on the timing of the underlying activations and transmissions.

Proposition 1. ✓ *Given a trace* $\mathcal{E}$*,*

$$A_i \not\rightarrow B_j \Longrightarrow t(A_i) + \tau(A_i) > t(B_j)$$
$$A_i \rightarrow A_j \Longrightarrow (j-i)T_{min} \leq t(A_j) - t(A_i) \leq (j-i)T_{max}.$$

Proof. Both statements follow directly from definitions 3 and 4. Figure 2 shows the key intuition. If $t(A_i) + \tau(A_i) \leq t(B_j)$, the message sent at A_i would have been received at B_j, giving $A_i \rightarrow B_j$. The converse is not true in general. The second equation is shown by induction over $(j-i)$ after noting that $t(A_j) = t(A_i) + \sum_{k=i}^{j-1} T_k^A$, and for each k, $T_{\text{min}} \leq T_k^A \leq T_{\text{max}}$. □

This lemma shows the link between a trace of a discrete model and the corresponding set of real-time traces.

We end this section with two useful properties of the real-time model. First, due to variable transmission delays, a message sent by a node may arrive after a later message sent by the same node. An example of such a *message inversion* is shown in figure 3. But message inversion is not possible if the maximum transmission jitter is less than the minimum time between activations.

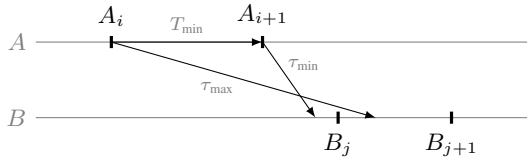


Fig. 3: Message inversion.

Proposition 2 (Message Inversion)**.** ✓ *Message inversion cannot occur if the following condition holds.*

$$T_{min} + \tau_{min} > \tau_{max} \quad \text{(MI)}$$

Proof. Assuming condition MI together with the worst-case scenario show in figure 3 leads to an absurdity. □

Now, assuming that message inversion cannot occur, the converse of proposition 1 holds.

Proposition 3. ✓ *Given a trace $\mathcal{E}$ without message inversions,*

$$A_i \to B_j \iff t(A_i) + \tau(A_i) \leq t(B_j).$$

The $\to$ relation is completely characterized by the delay between events, even those occurring on distant nodes.

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. The simplest technique, of defining a sampling period and using it to finely slice a real-time trace into a sequence of discrete ticks does not work. No matter how small the transmission time, provided it is strictly greater than zero, it may still be split in two, with the sending activation A_i on one side, and a non-receiving activation B_j, that is, $t(A_i) < t(B_j) < t(A_i) + \tau(A_i)$, on the other side. This violates the principle that a single logical step accounts for message transmission. In fact, the most general approach is to ensure that when one event x occurs before another event y in the discrete-time trace, x happens before y, that is, $x \to y$, in the corresponding real-time trace, and vice versa.

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

$$\forall x, y \in \mathcal{E},\ x \to y \iff f(x) < f(y). \tag{UD}$$

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 every (real-time) trace. The forward direction of the equivalence comes from the fact that the $\to$ 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 [12].

A unitary discretization links the causality of events in the real-time model to the causality imposed by 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 at the same node as x, or because y occurs after the receipt of the message sent at x and thus with the value sent at x or a later one sent from the same node. It is the communication through unit delays on a common clock that gives the tight link between the two causality relations.

The existence of a discretization satisfying condition UD is rather abstract but the question is in fact equivalent to a simpler condition.

Lemma 1 (UD $\iff$ UC)**.** ✓ *A unitary discretization, for a trace $\mathcal{E}$, satisfying condition UD exists if and only if:*

$$\nexists\, x, y, z \in \mathcal{E},\ x \to y \text{ and } x \parallel z \text{ and } y \parallel z. \tag{UC}$$

(a) $z \to y$. (b) $x \to z$.

Fig. 4: No unitary discretization for $x \to y$, $x \parallel z$ and $y \parallel z$.

The notation $x \parallel y$ indicates that events x and y are *concurrent*, that is, not causally related: $x \not\to y$ and $y \not\to x$. Condition UC states that two causally dependent events cannot both be concurrent with the same third event.

Before presenting the proof of this lemma, we describe the basic intuition using the illustration in figure 4. Assume that there are three events $x, y, z \in \mathcal{E}$ such that, $x \to y$, $x \parallel z$ and $y \parallel z$. If there is a unitary discretization f, we must have $f(x) < f(y)$. Since $x \parallel z$ we must also have, from definition 5, $f(x) = f(z)$. But then we would have $f(z) < f(y)$ which implies $z \to y$ (figure 4a). On the other hand, since $y \parallel z$ we must have $f(z) = f(y)$. But this gives $f(x) < f(z)$ which implies $x \to z$ (figure 4b).

Proof. To show that $\exists$UD $\implies$ UC, assume that there exists a discretization function $f : \mathcal{E} \to \mathbb{N}$, such that both conditions UD and $\overline{\text{UC}}$ hold:

$$\forall x, y \in \mathcal{E},\ x \to y \iff f(x) < f(y) \tag{UD}$$

$$\exists\, x, y, z \in \mathcal{E},\ x \to y \text{ and } x \parallel z \text{ and } y \parallel z \tag{$\overline{\text{UC}}$}$$

Then, we have $f(x) = f(z)$, $f(y) = f(z)$ and $f(x) < f(y)$, which is absurd. Hence $\exists$UD $\implies$ UC.

To show that UC $\implies$ $\exists$UD, we first define $f : \mathcal{E} \to \mathbb{N}$:

$$f(y) = \begin{cases} 0 & \text{if } \nexists\, x \in \mathcal{E} \text{ such that } x \to y \\ \max\limits_{x \to y} f(x) + 1 & \text{otherwise.} \end{cases} \tag{1.1}$$

We must show that f satisfies condition UD. By construction, we have that $\forall x, y \in \mathcal{E},\ x \to y \implies f(x) < f(y)$. Now, we show by contraposition that

$$UC \implies (\forall x, y \in \mathcal{E},\ f(x) < f(y) \implies x \to y).$$

Assume there are $x, y \in \mathcal{E}$ such that $x \not\to y$ and $f(x) < f(y)$. From the latter, it follows that $y \not\to x$, otherwise we would have $f(x) > f(y)$. We may thus conclude that

$$y \parallel x. \tag{1.2}$$

Moreover, by the definition of f, there is a $z \in \mathcal{E}$ such that:

$$z \to y \tag{1.3}$$

$$f(y) = f(z) + 1 \tag{1.4}$$

Since we assume $x \not\to y$, we have $x \not\to z$, otherwise we would have $x \to z \to y$. On the other hand, $z \not\to x$, otherwise we would have $f(z) < f(x)$, but from equation (1.4) we know that $f(x) \leq f(z) < f(y)$. We thus conclude that

$$z \parallel x. \tag{1.5}$$

The conjunction of equations (1.2), (1.3), and (1.5) is precisely $\overline{\text{UC}}$, the negation of condition UC. Thus UC $\Longrightarrow \exists$UD. □

Equation (1.1) provides a discretization function mapping real-time traces to discrete-time ones. Basically, for each activation, it counts back along the longest chain of events that happened before it, and reserves one logical instant for each. The definition is well founded because the causality relation is not cyclic, and the set of earlier node activations is finite. It is not only a valid unitary discretization, it is also the most compact one: all other unitary discretizations can be obtained from this one by adding 'mute' instants, that is, logical steps where no node executes.

A. Discretizing two node systems

We first consider models of only two nodes. Such models were the focus of the original work on the quasi-synchronous approach [3] and they are also relevant in practice [8], [9].

Theorem 1 (Two-node Unitary Discretizability)**.** ✓ *A real-time model satisfying definition 1 with two nodes ($|\mathcal{N}| = 2$) can be discretized if and only if*

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

This result is coherent with Caspi's requirement that transmission delays be constrained to be 'significantly shorter than the periods of read and write clocks'.

We do not present the full proof. The basic idea is to show that condition 2D is equivalent to condition UC which gives the possibility of a unitary discretization (lemma 1). If condition UC does not hold, then there are three events A_i, A_j, B_k such that $A_i \to B_j$, $A_i \parallel B_k$, and $A_j \parallel B_k$, which in the worst case implies $T_{\text{min}} < 2\tau_{\text{max}}$, that is, the negation of condition 2D. Conversely, if condition 2D does not hold, figure 5 shows that there exists a trace violating condition UC.

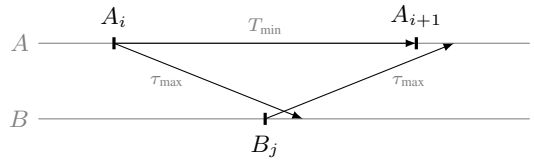


Fig. 5: Witness for UC $\Longrightarrow$ 2D.

B. Discretizing general systems

One might expect to use the discretizability for two nodes as a building block to extend the result to general models with three or more nodes. Unfortunately, this is not possible.

Theorem 2 (No Unitary Discretization)**.** ✓ *There is no unitary discretization for a real-time model with more than two nodes.*

Proof. In real-time models with more than two nodes, condition UC, implies instantaneous communication, that is, $\tau_{\text{max}} = 0$, which is incompatible with definition 3. Therefore such systems cannot be unitary discretized.

To show that UC $\Longrightarrow (\tau_{\text{max}} = 0)$, assume that $\tau_{\text{max}} > 0$. Then, as figure 6 shows, there exists a trace where $A_i \to B_j$, $A_i \parallel C_k$, and $B_j \parallel C_k$, thus violating condition UC. □

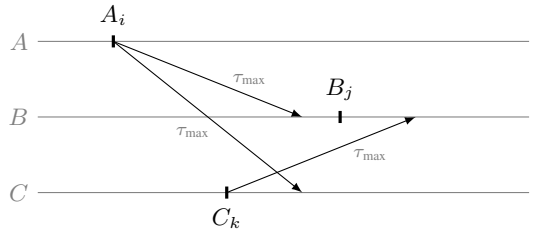


Fig. 6: Witness for UC $\Longrightarrow (\tau_{\text{max}} = 0)$.

This theorem implies that a real-time model of more than two nodes cannot be unitary discretized to a model satisfying definition 2 (unless one is willing to accept $\tau_{\text{max}} = 0$). The discrete-time model is only sound for systems of pairs of quasi-periodic nodes where the transmission delay is sufficiently short. Systems of more than two nodes cannot be unitary discretized without making additional assumptions.

IV. Constraining Communications

Although the quasi-synchronous abstraction is not sound for general systems of more than two nodes (theorem 2), we now show that it may still apply when node interactions are limited.

Figure 7 shows a trace of a three node system. Since $A_1 \to B_1$, $A_1 \parallel C_1$, and $B_1 \parallel C_1$, lemma 1 implies that unitary discretization is impossible. But if we know that nodes A and C do not communicate, we also know that $A_i \to C_j$ is only possible by transitivity. Thus $f(A_i) < f(C_j)$ alone does not imply $A_i \to C_j$. The discretization function need only be constrained by nodes that communicate with each other. For instance, since $B_1 \parallel C_1$ and nodes B and C communicate with each other, we must have $f(B_1) = f(C_1)$, but there is no constraint between $f(A_1)$ and $f(C_1)$. Note that there is no direct link between the timestamp of an event and the logical instant assigned to that event by a unitary discretization. For instance, $f(A_2) < f(C_2)$ even though C_2 occurs before A_2.

We formalize these ideas using a *communicates with* relation, written $\rightrightarrows$, between the nodes of a real-time model. Note that this relation is not necessarily symmetric, $A \rightrightarrows B$ need not imply $B \rightrightarrows A$, but it must be reflexive ($A \rightrightarrows A$).

We also modify the *(recv)* clause of definition 4:

(recv') If $A \rightrightarrows B$ and $t(A_i) + \tau(A_i) \leq t(B_j)$ then $A_i \to B_j$.

A message is received if two nodes communicate and the activation of the receiver occurs after the transmission delay. The definition of unitary discretizations (definition 5) retains its form but its implications change: condition UD constrains $f(A_i)$ and $f(B_j)$ only if $A \rightrightarrows B$. Propositions 1 to 3 continue to hold trivially. Lemma 1 holds with the updated definition of a unitary discretization. Theorem 1 holds under the assumption that $A \rightrightarrows B$ and $B \rightrightarrows A$ when $\mathcal{N} = \{A, B\}$. We now show how to exclude the counter-examples in the proof of theorem 2 to permit unitary discretizations for a larger class of systems.

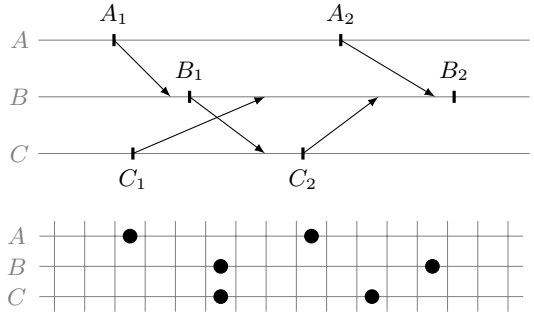


Fig. 7: A trace (above) and a possible unitary discretization.

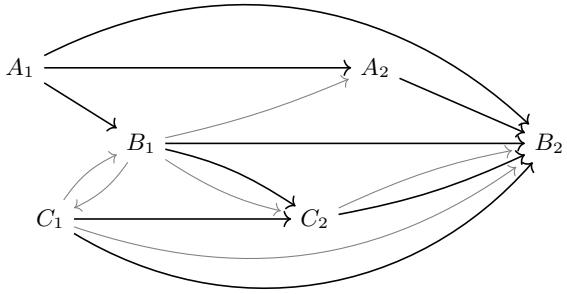


Fig. 8: The trace (sub-)graph of the trace in figure 7. Black thick arrows denote $\xrightarrow{1}$, thin gray ones $\xrightarrow{0}$.

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

Proposition 4. *If f is a unitary discretization for a trace, for a pair of nodes where $A \rightrightarrows B$ we have that*

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

Proof. The first equation is a direct consequence of the definition of a unitary discretization. The second one follows by contradiction. Assume that $A_i \not\to B_j$ and $f(A_i) < f(B_j)$, then by the definition of f we have $A_i \to B_j$. □

An intermediate step to defining a static condition on communications is to more finely characterize traces that do not have unitary discretizations. Our characterization will be based on a graph of the constraints of proposition 4.

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

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

An example trace graph is shown in figure 8. The edges labeled with one ($x \xrightarrow{1} y$) represent the constraints $f(x) < f(y)$. Such an edge indicates that the source activation must be placed before the destination activation in any unitary discretization, that is, the value of f, from source to destination, must increase by at least one. Those labeled with zeros ($x \xrightarrow{0} y$) represent the constraints $f(x) \leq f(y)$. Such an edge indicates that the source activation cannot be placed before the destination activation in any 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 8). Any cycle containing a $\xrightarrow{1}$ represents a set of unsatisfiable constraints: one of the activations must be placed in two different slots.

Theorem 3 (Acyclic Unitary Discretization)**.** *For a trace $\mathcal{E}$, there exists a unitary discretization if and only if the corresponding graph $\mathcal{G}$ has no cycle of positive weight.*

Proof. We show that $\exists\text{UD} \implies \overline{\exists\text{PC}}$ by contraposition. Assume that there exists a cycle of positive weight. By construction of $\mathcal{G}$ there is an event A_i such that, for any unitary discretization function, $f(A_i) < f(A_i)$, which is impossible.

We must now show that $\overline{\exists\text{PC}} \implies \exists\text{UD}$. Since there are no cycles of positive weight, we may define a function f that maps each event A_i to the weight of the longest path in $\mathcal{G}$ that leads to A_i. By construction $A_i \to B_j \implies f(A_i) < f(B_j)$, which is half of condition UD of definition 5.

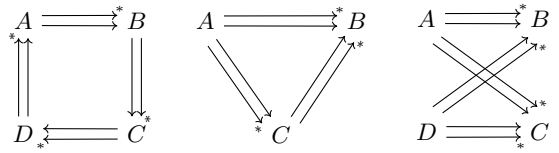
The other half of condition UD follows by contraposition. Assume $A_i \not\to B_j$. If $A \rightrightarrows B$ then by proposition 4 we have $f(B_j) \leq f(A_i)$ and thus $\neg(f(A_i) < f(B_j))$. The other case, $\neg(A \rightrightarrows B)$, is trivial. □

Similarly to the function defined for lemma 1, see equation (1.1), the unitary discretization described in the proof above is the most concise one. Other discretizations are constructed by adding 'mute' instants where no node executes, as in figure 7.

Problematic cycles in traces can be precluded by imposing restrictions on communication patterns: 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 $A \neq B$). For example, the counterexample of figure 6 shows that when three nodes communicate such that $C \leftleftarrows A \rightrightarrows B \leftleftarrows C$, there is at least one trace for which there is no unitary discretization. By forbidding this pattern, which we hereafter call $\mathcal{C}_0$, and two others, we guarantee the existence of unitary discretizations for all traces.

Theorem 4. *If the transitive closure of the communicates with relation $\rightrightarrows^*$ of a real-time model is acyclic and contains neither $\mathcal{C}_0$ nor alternating cycles, then every trace of the model has a unitary discretization.*

In an alternating cycle, or *crossover*, the direction of the communication links change when the node does. A crossover with four nodes $A \rightrightarrows^* B \ {}^*\!\!\leftleftarrows C \rightrightarrows^* D \ {}^*\!\!\leftleftarrows A$ is shown in figure 9 together with examples of the two other forbidden communication topologies.



(a) Cycle (b) $\mathcal{C}_0$ (c) Crossover

Fig. 9: Forbidden communication topologies.

Proof. We fix a trace and show $\overline{\exists \text{FT}} \implies \exists \text{UD}$ by contraposition. Assuming there is no unitary discretization, theorem 3 implies that $\mathcal{G}$ has a cycle of positive weight:

$$e_0 \overset{0}{\to}^* e_{k_1} \overset{1}{\to}^* \ldots \overset{0}{\to}^* e_{k_n} \overset{1}{\to}^* e_0.$$

We will write N_k for the node corresponding to activation e_k. By definition 6, a chain $e_i \overset{0}{\to}^* e_j$, corresponds to a communication link $N_i \ {}^*\!\leftleftarrows N_j$ with $N_i \neq N_j$, and a chain $e_i \overset{1}{\to}^* e_j$ corresponds to a communication link in the opposite direction $N_i \rightrightarrows^* N_j$ but with the possibility that $N_i = N_j$, for activations of the same node. Thus we have that

$$N_0 \ {}^*\!\leftleftarrows N_{k_1} \rightrightarrows^* \ldots \ {}^*\!\leftleftarrows N_{k_n} \rightrightarrows^* N_0.$$

There are three possible cases:

1) If $N_i = N_j$ for all $e_i \overset{1}{\to}^* e_j$, there is a cycle $N_0 \ {}^*\!\leftleftarrows N_0$.
2) If there is exactly one $e_i \overset{1}{\to}^* e_j$ with $N_i \neq N_j$, there is a $\mathcal{C}_0$ pattern: $N_0 \ {}^*\!\leftleftarrows N_i \rightrightarrows^* N_j \ {}^*\!\leftleftarrows N_0$.
3) If there are two or more $N_i \rightrightarrows^* N_j$ with $N_i \neq N_j$, there is an alternating cycle.

Each case shows the existence of a forbidden topology. □

This theorem provides sufficient conditions for the unitary discretization of a real-time model. The conditions are, however, too restrictive. They exclude, for instance, the two-node systems studied in section III-A. The weakest condition that ensures the unitary discretization of a system remains to be found, and may involve conditions similar that of theorem 1. That said, theorem 4 already allows the application of the quasi-synchronous approach to architectures where nodes send data forward to a single controller, as in triple modular redundancy and filtering systems, which are important in practice.

V. The Quasi-Synchronous Abstraction

In this final section, we generalize the quasi-synchronous model[2] and also the definitions and results on unitary discretizations. This allows us to precisely describe when the quasi-synchronous model can be applied to a real-time system.

A discrete-time model is termed n-quasi-synchronous if there are no more than n activations of one node between any two successive activations of another. This definition can be formalized using unitary discretizations.

[2]As suggested by the *Same_Period$_n$* predicate of [3, §3.2.2].

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

1) there exists a unitary discretization f, and
2) for any nodes A and B, there is no chain of activations of length greater than n, that is, no i and j such that

$$f(B_j) \leq f(A_i) < \cdots < f(A_{i+n}) \leq f(B_{j+1}). \quad \text{(QS)}$$

This definition expresses the two central features of quasi-synchrony: communications as 'logical' unit delays and constraints on node interleavings.

Definition 7 is consistent with the natural generalization of definition 2 from section I-B. Using the unitary discretization of definition 7(1), the activation instants of a node A can be represented by a boolean clock: $c_A(i) = t$ iff $f(A_j) = i$ for some j. Then condition QS is equivalent to the fact that the pair of clocks (c_A, c_B) associated to nodes A and B never contains the subsequence

$$\begin{bmatrix} t \\ \_ \end{bmatrix} \cdot \begin{bmatrix} f \\ f \end{bmatrix}^* \cdot \begin{bmatrix} t \\ f \end{bmatrix}^{n-1} \cdot \begin{bmatrix} f \\ f \end{bmatrix}^* \cdot \begin{bmatrix} t \\ \_ \end{bmatrix}.$$

While definition 7 captures the essence of quasi-synchrony, the requirements are rather abstract. We now present two theorems that incorporate the results of previous sections to give concrete requirements on real-time parameters and communication topologies.

Theorem 5. ✓ *A real-time model satisfying definition 1 with two nodes ($|\mathcal{N}| = 2$) is n-quasi-synchronous if and only if,*

1) condition 2D holds, that is $T_{min} \geq 2\tau_{max}$, and,
2) the following condition holds:

$$nT_{min} \geq T_{max} + 2\tau_{max} \quad \text{(QT)}$$

The first condition is equivalent to the existence of a unitary discretization (theorem 1). Given a discretization function, condition QT is equivalent to condition QS when two nodes A and B intercommunicate, that is, $A \rightrightarrows B$ and $B \rightrightarrows A$.

If condition QS does not hold, there is a chain of events such that $f(B_j) \leq f(A_i) < \cdots < A_{i+n} \leq f(B_{j+1})$. This gives $A_i \not\to B_j$ and $B_{j+1} \not\to A_{i+n}$, which implies in the worst case that $nT_{\text{min}} < T_{\text{max}} + 2\tau_{\text{max}}$, that is, the negation of condition QT. On the other hand, if condition QT does not hold, then figure 10 shows that there exists a trace where $B_j \parallel A_i \to A_{i+1} \to \ldots \to A_{i+n} \parallel B_{j+1}$. Then by the definition of a unitary discretization we have that

$$f(B_j) \leq f(A_i) < \cdots < f(A_{i+n}) \leq f(B_{j+1}),$$

which violates condition QS.

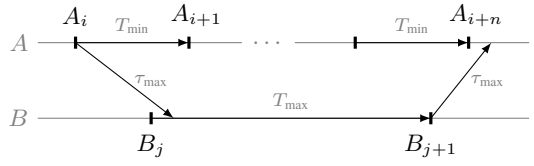


Fig. 10: Witness for QS $\implies$ QT.

To obtain a similar result for systems with three or more nodes, we work within the rules prescribed by theorem 4. In particular, pairs of intercommunicating nodes, for instance, $A \rightrightarrows B$ and $B \rightrightarrows A$ cannot be accepted and condition QT is no longer relevant.

Theorem 6. ✓ *A real-time model satisfying definition 1 is n-quasi-synchronous if,*

1) *its communication graph has no forbidden topologies,*
2) *message inversion does not occur, and,*
3) *the following condition holds:*

$$nT_{min} + \tau_{min} \geq 2T_{max} + \tau_{max} \qquad \text{(QT')}$$

The first condition ensures the existence of a unitary discretization (theorem 4), but does not, in this case, give an equivalence. Given a discretization function, condition QT' is equivalent to condition QS when the communication graph is acyclic, that is, for a pair of communicating nodes A and B, either $A \rightrightarrows B$ or $B \rightrightarrows A$.

The proof that QT' $\iff$ QS is similar to that of theorem 5. If condition QS does not hold, there is a chain of events where

$$f(B_j) \leq f(A_i) < \cdots < f(A_{i+n}) \leq f(B_{j+1}) < f(B_{j+2}).$$

Since there is no communication $B \rightrightarrows A$, we have $A_i \not\to B_j$, $A_{i+n} \not\to B_{j+1}$ and $A_{i+n} \to B_{j+2}$, which, in the worst case, implies $nT_{\text{min}} + \tau_{\text{min}} < 2T_{\text{max}} + \tau_{\text{max}}$, the negation of condition QT'. The counterexample, shown in figure 11, is more subtle than the previous one. If condition QT' does not hold, there is a trace in which $A_i \not\to B_j$, $A_{i+n} \not\to B_{j+1}$, and $A_{i+n} \to B_{j+2}$. Thus the only constraints on the discretization function are $f(A_i) \geq f(B_j)$, $f(A_{i+n}) \geq f(B_{j+1})$, and $f(A_{i+n}) < f(B_{j+2})$, which permits a discretization where $f(A_{i+n}) = f(B_{j+1})$ in violation of condition QS.

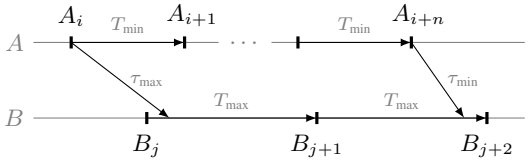


Fig. 11: Witness for QS $\implies$ QT'.

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) [3, §3.2.3]. These properties follow directly from the n-quasi-synchronous model (definition 7). Figure 12 shows the worst acceptable case: a chain of n activations of a node A between two successive activations of another node B.

Proposition 5 (Overwrites, Oversamples)**.** *The maximum number of successive overwrites or oversamplings in an n-quasi-synchronous system is $n-1$.*

Proof. The proof is straightforward given definition 5 and the worst acceptable case shown in figure 12:

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

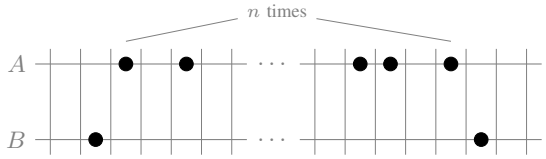


Fig. 12: Maximal overwrites and oversamplings.

For the maximum number of overwrites, the $n-1$ messages sent at $A_i, A_{i+1}, \ldots, A_{i+n-1}$ are overwritten by the message sent at A_{i+n} which is received by B at B_{j+1}.

For the maximum number of oversamplings, the $n-1$ activations $A_{i+1}, A_{i+2}, \ldots, A_{i+n}$ oversample the value sent by B at B_j which is received by A at A_i. □

VI. Conclusion

The quasi-synchronous abstraction provides a way to model and reason about a class of distributed embedded systems whose nodes communicate by sampling with bounded jitter. Given a real-time model satisfying certain constraints on timing parameters and communication topology, properties obtained of the corresponding quasi-synchronous model are also shared by the system itself. 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. This result contradicts a common belief that the quasi-synchronous abstraction applies to any quasi-periodic system.

References

[1] A. Benveniste, P. Caspi, P. Le Guernic, H. Marchand, J.-P. Talpin, and S. Tripakis. A protocol for loosely time-triggered architectures. In *EMSOFT'02*, pages 252–265, Oct. 2002.

[2] S. Bhattacharyya, S. Miller, J. Yang, S. Smolka, B. Meng, C. Sticksel, and C. Tinelli. Verification of quasi-synchronous systems with Uppaal. In *DASC'14*, pages 8A4–1, Oct. 2014.

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

[4] P. Caspi. Embedded control: From asynchrony to synchrony and back. In *EMSOFT'01*, pages 80–96, Oct. 2001.

[5] P. Caspi, C. Mazuet, and N. Reynaud Paligot. About the design of distributed control systems: The quasi-synchronous approach. In *SAFECOMP'01*, pages 215–226, Sept. 2001.

[6] Esterel Technologies. Scade suite. http://www.esterel-technologies.com/products/scade-suite/.

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

[8] N. Halbwachs and L. Mandel. Simulation and verification of aysnchronous systems by means of a synchronous model. In *ACSD'06*, pages 3–14, June 2006.

[9] E. Jahier, N. Halbwachs, and P. Raymond. Synchronous modeling and validation of schedulers dealing with shared resources. Technical Report TR-2008-10, VERIMAG, July 2000.

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

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

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

[13] T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. 2002.

APPENDIX

DETAILED PROOFS

A. Traces and Causality

Proposition 2 (Message Inversion)**.** ✓ *Message inversion cannot occur if the following condition holds.*

$$T_{min} + \tau_{min} > \tau_{max} \tag{MI}$$

Proof. The idea of the proof is to forbid the worst case scenario that allows a message inversion illustrated in figure 3.

Assume a message inversion occurs: there exist two subsequent activations of the same node $A_i \to A_j$ such that the message sent at A_i is received later than the message sent at A_j.

$$t(A_i) + \tau(A_i) \geq t(A_j) + \tau(A_j). \tag{2.1}$$

Using definition 1 we can bound the transmission delays:

$$t(A_i) + \tau(A_i) \geq t(A_i) + \tau_{\max} \tag{2.2}$$
$$t(A_j) + \tau(A_j) \leq t(A_j) + \tau_{\min}. \tag{2.3}$$

Besides, from proposition 1 we get:

$$t(A_j) \geq t(A_i) + T_i^A \geq t(A_i) + T_{\min}. \tag{2.4}$$

Then, combining equations (2.1) to (2.4) we obtain:

$$\begin{aligned} t(A_i) + T_{\min} + \tau_{\min} &\leq t(A_j) + \tau_{\min} \\ &\leq t(A_i) + \tau_{\max}. \end{aligned}$$

Thus $T_{\min} + \tau_{\min} \leq \tau_{\max}$, that is, $\overline{\text{MI}}$ the negation of condition MI. Hence if condition MI holds, no message inversion can occur. □

Proposition 3. ✓ *Given a trace $\mathcal{E}$ without message inversions,*

$$A_i \to B_j \iff t(A_i) + \tau(A_i) \leq t(B_j).$$

Proof. For two nodes $A, B \in \mathcal{N}$, assume that,

$$t(A_i) + \tau(A_i) > t(B_j) \tag{3.1}$$
$$A_i \to B_j. \tag{3.2}$$

In other words, A_i happened before B_j, but B_j does not read the message sent at A_i. Hence there exists A_k, B_l such that $A_i \to A_k \to B_l \to B_j$ where B_l is the reading of the message sent at A_k:

$$t(A_k) + \tau(A_k) \leq t(B_l). \tag{3.3}$$

Note that it is possible that $B_l = B_j$. Hence, we get from equation (3.1):

$$t(A_i) + \tau(A_i) > t(B_l). \tag{3.4}$$

Then, from proposition 1 we get:

$$t(A_k) \geq t(A_i) + T_i^A. \tag{3.5}$$
$$\tag{3.6}$$

The conjunction of equations (3.3) and (3.4) corresponds to a message inversion. □

B. Discretizing two node systems

Theorem 1 (Two-node Unitary Discretizability)**.** ✓ *A real-time model satisfying definition 1 and comprising two nodes ($|\mathcal{N}| = 2$) can be discretized if and only if*

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

Proof. We prove here that condition 2D is equivalent to condition UC which reflect the possibility of a unitary discretization (lemma 1).

To prove that 2D $\implies$ UC, assume that condition UC does not hold. It is obvious that two concurrent events cannot be activations of the same nodes. Hence condition UC becomes:

$$A_i \to A_j \text{ and } A_i \parallel B_k \text{ and } A_j \parallel B_k \tag{$\overline{\text{UC}}$}$$

From definition 1, proposition 1 and condition $\overline{\text{UC}}$ we get:

$$t(B_k) < t(A_i) + \tau(A_i) \ \leq t(A_i) + \tau_{\max} \tag{4.1}$$
$$t(A_j) < t(B_k) + \tau(B_k) \leq t(B_k) + \tau_{\max} \tag{4.2}$$
$$t(A_j) \geq \quad t(A_i) + T_i^A \ \geq t(A_i) + T_{\min} \tag{4.3}$$

Then from equations (4.1) to (4.3) we get:

$$\begin{aligned} t(A_i) + T_{\min} &\leq t(A_j) \\ &< t(B_k) + \tau_{\max} \\ &< t(A_i) + 2\tau_{\max} \end{aligned}$$

Thus $T_{\min} < 2\tau_{\max}$, that is, $\overline{\text{2D}}$. Hence 2D $\implies$ UC.

To prove that UC $\implies$ 2D, assume that condition 2D does not hold. Then figure 5 shows that there exists a trace which violates condition UC. □

C. The Quasi-Synchronous Abstraction

Theorem 5. ✓ *A real-time model satisfying definition 1 is n-quasi-synchronous if,*

1) *its communication graph has no forbidden topologies,*
2) *message inversion does not occur, and,*
3) *the following condition holds:*

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

Proof. The first condition is equivalent to the existence of a unitary discretization f (theorem 1).

Then we can prove that QT $\iff$ QS. To show that QT $\implies$ QS, assume that condition QS does not hold, that is, there exists i and j such that:

$$f(B_j) \leq f(A_i) < \cdots < f(A_{i+n}) \leq f(B_{j+1}).$$

From the definition of a unitary discretization we get $A_i \not\to B_j$. Besides condition 2D is more restrictive that condition MI. We can thus apply definition 1 and proposition 3 and get:

$$t(B_j) < \quad t(A_i) + \tau(A_i) \leq t(A_i) + \tau_{\max} \tag{5.1}$$

From proposition 1 we also have:

$$t(B_{j+1}) \leq \quad t(B_i) + T_j^B \quad \leq t(B_i) + T_{\max} \tag{5.2}$$
$$t(A_{i+n}) = t(a_i) + \sum_{k=i}^{i+n-1} T_k^A \geq t(A_i) + nT_{\min} \tag{5.3}$$

There are two possible cases:

1) If $f(A_{i+n}) < f(B_{j+1})$ we have $A_{i+n} \to B_{j+1}$ and thus:

$$t(B_{j+1}) > t(A_{i+n}) + \tau_{\min} \tag{5.4}$$

Then from equations (5.1) to (5.4) we get:

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

Thus we get: $nT_{\min} + \tau_{\min} < \tau_{\max} + T_{\max}$ which implies $nT_{\min} < T_{\max} + 2\tau_{\max}$ that is $\overline{\text{QT}}$.

2) If $f(A_{i+n}) = f(B_{j+1})$ we have $B_{j+1} \not\to A_{i+n}$ and thus

$$t(A_{i+n}) < t(B_{j+1}) + \tau_{\max} \tag{5.5}$$

Then from equations (5.1) to (5.3) and (5.5) we get:

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

Thus we get: $nT_{\min} < T_{\max} + 2\tau_{\max}$ that is $\overline{\text{QT}}$.

In both cases QT $\implies$ QS.

To show that QS $\implies$ QT, assume that condition QT does not hold. Then figure 6 shows that there exists a trace with $B_j \parallel A_i \to A_{i+1} \to \ldots \to A_{i+n} \to B_{j+1}$. Then using definition 5 we get:

$$f(B_j) \leq f(A_i) < \cdots < f(A_{i+n}) \leq f(B_{j+1})$$

which violates condition QS. □

Theorem 6. ✓ *A real-time model satisfying definition 1 is n-quasi-synchronous if,*

1) its communication graph has no forbidden topologies,
2) message inversion does not occur, and,
3) the following condition holds:

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

Proof. The first condition ensures that there exists a unitary discretization f (theorem 4).

Then we can prove that QT $\iff$ QS. The proof is similar to the one of theorem 5 except that there is no possible communication between B and A. To show that QT' $\implies$ QS, assume that condition QS does not hold, that is, there exists i and j such that:

$$f(B_j) \leq f(A_i) < \cdots < f(A_{i+n}) \leq f(B_{j+1}).$$

From the definition of a unitary discretization we get $A_i \not\to B_j$. We can thus apply definition 1 and proposition 3 and get:

$$t(B_j) < \quad t(A_i) + \tau(A_i) \leq t(A_i) + \tau_{\max} \tag{6.1}$$

From proposition 1 we also have:

$$t(B_{j+2}) \leq t(B_i) + 2T_{\max} \tag{6.2}$$

$$t(A_{i+n}) \geq t(A_i) + nT_{\min} \tag{6.3}$$

There are two possible cases:

1) If $f(A_{i+n}) < f(B_{j+1})$ we have $A_{i+n} \to B_{j+1}$ and thus:

$$t(B_{j+1}) > t(A_{i+n}) + \tau_{\min} \tag{6.4}$$

Then from equations (6.1) to (6.4) we get:

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

Thus we get: $nT_{\min} + \tau_{\min} < \tau_{\max} + T_{\max}$ which implies $nT_{\min} + \tau_{\min} < 2T_{\max} + \tau_{\max}$ that is $\overline{\text{QT'}}$.

2) If $f(A_{i+n}) = f(B_{j+1})$ we have $A_{i+n} \not\to B_{j+1}$ but also $A_{i+n} \to B_{j+2}$ since $f(A_{i+n}) \leq f(B_{j+1}) < f(B_{j+2})$ and thus

$$t(B_{j+2}) < t(B_{j+1}) + \tau_{\max} \tag{6.5}$$

Then from equations (6.1) to (6.3) and (6.5) we get:

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

Thus we get: $nT_{\min} + \tau_{\min} < 2T_{\max} + \tau_{\max}$ that is $\overline{\text{QT'}}$.

In both cases QT' $\implies$ QS.

To show that QS $\implies$ QT', assume that condition QT' does not hold. Then figure 11 shows that there exists a trace with $B_j \parallel A_i \to A_{i+1} \to \ldots \to A_{i+n} \to B_{j+1}$. Then using definition 5 we get:

$$f(B_j) \leq f(A_i) < \cdots < f(A_{i+n}) \leq f(B_{j+1})$$

which violates condition QS. □