MSYNC: A Generalized Formal Design Pattern for Virtually Synchronous Multirate Cyber-Physical Systems

KYUNGMIN BAE*, Pohang University of Science and Technology, South Korea
PETER CSABA ÖLVECZKY, University of Oslo, Norway

TTA and PALS are two prominent formal design patterns—with different strengths and weaknesses—for virtually synchronous distributed cyber-physical systems (CPSs). They greatly simplify the design and verification of such systems by allowing us to design and verify their underlying synchronous designs. In this paper we introduce and verify MSYNC as a formal design (and verification) pattern/synchronizer for hierarchical multirate CPSs that generalizes, and combines the advantages of, both TTA and (single-rate and multirate) PALS. We show that MSYNC outperforms both TTA and PALS in terms of allowing significantly shorter periods. Finally, as part of this work, we also define an extension of TTA to multirate CPSs.

1 INTRODUCTION

Many cyber-physical systems (CPSs) are *virtually synchronous*: they should behave as if they were logically synchronous—i.e., in each iteration of the system, all components read inputs, perform a transition, and generate outputs for the next iteration—but have to be realized in a distributed setting. Examples of virtually synchronous CPSs include avionics and automotive systems [5, 11, 17], networked medical devices [4, 8], and other distributed control systems such as the steam-boiler benchmark [1]. The underlying infrastructure of such systems often guarantees bounds on clock skews, network delays, and execution times. Designing and verifying such CPSs are difficult, due to having to account for asynchronous communication with message delays, execution times, imprecise local clocks, etc., while needing to meet stringent correctness and timeliness requirements.

Two prominent *formal design patterns*, TTA ("time-triggered architectures") [9, 16] and PALS ("physically asynchronous, logically synchronous") [3, 12], drastically reduce both the design and verification complexity of such virtually synchronous CPSs: Given bounds Γ on network delays, execution times, and clock skews, it is sufficient to design and verify the much simpler synchronous design *SD*. TTA and PALS then give us the corresponding distributed real-time systems $TTA(SD, \Gamma)$ and $PALS(SD, \Gamma)$, which are *stuttering bisimilar* to *SD*.

The TTA and PALS patterns are widely used for avionic and automotive CPSs [13, 14]. The power of this design and verification methodology is illustrated, e.g., in [12], where for an integrated modular avionics application, the synchronous design *SD* could be model checked in less than a second, whereas model checking the asynchronous system was infeasible even in the simple setting without execution times, with perfect local clocks, and with communication delays 0 and 1.

While the TTA and PALS *synchronous* models are fairly similar, the TTA and PALS *distributed real-time* models are somewhat different. In TTA, each "round" consists of two phases: each component sends a message in the first phase, and executes a transition in the second phase. In PALS, each component executes a transition and then sends a message, potentially after some delay. The shortest period in PALS is therefore smaller than that of TTA. Both TTA and PALS have been formally verified, and are also formalized in PVS and compared in [18].

*Corresponding author

Authors' addresses: Kyungmin Bae, kmbae@postech.ac.kr, Pohang University of Science and Technology, 77 Cheongam-Ro, Nam-Gu, Pohang, Gyeongbuk, 37673, South Korea; Peter Csaba Ölveczky, peterol@ifi.uio.no, University of Oslo, Norway.

2021. 0004-5411/2021/8-ART111 $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Different controllers often have different periods/frequencies. For example, the rudder and aileron controllers in commercial aircrafts typically have different frequencies (30–100 Hz for ailerons, and 30–50 Hz for rudders [2]), yet must synchronize to turn an airplane. PALS was therefore extended to the multirate setting in [2, 6], where faster components (or even entire faster subsystems) communicate with slower controllers/subsystems. One key thing is that *one* output by a slow controller needs to be the input for $k > 1$ rounds of the fast component, and k outputs from the fast component should become *one* input to the slow component. User-definable *input adaptors* θ define how one "slow" output o is transformed into a sequence of k inputs, and vice versa.

An important feature of Multirate PALS is *output cutoff numbers* of input adaptors. In multirate systems, since messages are sent after execution in PALS, all k outputs generated by the faster components may not reach the slow component by the beginning of the next iteration (since the last ones are generated later than the first ones). Therefore, in PALS, if the *last* j of these k fast outputs may not be received by the slow components, the output cutoff number is $k - j - 1$.

Multirate PALS has two important limitations:

(1) Different components may have different the worst-case execution times (WCETs). This is not taken into account in PALS, and therefore the shortest period given by Multirate PALS is hardly "optimal" in reality.
(2) The output cutoff number in Multirate PALS is useful, e.g., for an actuator that should react to all outputs of a slow controller, but is less suitable for a sensor or a monitor where a slow controller should collect all outputs.

The second problem can be addressed by introducing a different feature for multirate systems, namely, *input cutoff numbers*. Instead of a slow component discarding the last k fast outputs from a fast component, it is also possible that a fast component can "skip" first κ fast rounds before receiving an input from a slow component. In this paper, we introduce this notion of input cutoff numbers and investigate its consequence on the corresponding distributed real-time systems. We show that the input cutoff approach naturally is present in TTA distributed real-time systems, and define the first multirate extension of TTA, called Multirate TTA, in this paper.

To combine the advantages of both input and output cutoff approaches, we propose, define, and verify a new formal design pattern/synchronizer, called MSYNC ("mixed synchronizer"), for virtually synchronous multirate CPSs that generalizes both TTA and Multirate PALS by: (i) supporting both the input cutoff and the output cutoff approaches, and (ii) specifying component-specific WCETs to reduce the smallest possible period. In this paper, we show that the combination of the above two features allows MSYNC to have shorter periods that the "optimal" shortest periods in TTA and PALS, which makes our synchronizer applicable to more components.

The contributions of this paper can be summarized as follows:

- We propose and formalize the MSYNC for virtually synchronous multirate CPSs for the general hierarchical case with hierarchical subsystems that operate at their respective rates.
- We show that the shortest periods in *MSYNC* can be much smaller that those of Multirate PALS, and of Multirate TTA defined in the paper.
- We formally prove (in the complex multirate hierarchical setting) the precise relationship between the synchronous design *SD* and the asynchronous real-time system $\mathit{MSYNC}(\mathit{SD}, \Gamma')$.
- We discuss and exemplify how linear programming can be used to find optimal values of tunable parameters for different purposes.

TTA and PALS are part of family of "synchronizers" relating synchronous and asynchronous systems, and differ from most others by focusing on CPSs, where the computations need to satisfy strict time constraints. We refer to [12, Section 10] for a comparison of PALS and synchronizers in general. We are not aware of any formally verified synchronizer for *multirate* CPSs beyond

Multirate PALS. This also includes the loosely time-triggered architectures (LTTAs) of Benveniste and others [7], where a synchronous composition of Mealy machines (like our synchronous models) are deployed onto a distributed quasi-periodic architecture.

2 PRELIMINARIES ON PALS AND TTA

This section presents some background on PALS and TTA. Both patterns reduce the design and verification of distributed real-time systems to the much simpler tasks for the corresponding synchronous designs. Given a synchronous design SD, and bounds Γ on clock skews, network delays, and execution times, PALS and TTA provide the distributed real-time systems $PALS(SD, \Gamma)$ and $TTA(SD, \Gamma)$, respectively, that exhibit the same behaviors as the synchronous design SD.

2.1 Synchronous Model of Computation

The synchronous models of TTA and PALS specify a collection of components that execute in a series of rounds. For each round, components read inputs, change states, and generate the outputs for the next round. In this sense, the synchronous models are essentially the same, but are formalized in different ways. The synchronous model of TTA [16] involves rounds with tho phases: components send messages in the first phase, and perform transitions in the second phase. On the other hand, in the synchronous model of PALS [12], rounds do not involves such two phases.

This papers intend to generalize both PALS and TTA, which requires to associate the same synchronous model with both patterns. The synchronous model of TTA [16] has repercussions on the behaviors of distributed real-time systems provided by TTA, which also involve rounds with two phases. On the contrary, it is hard to associate the TTA synchronous model with PALS, because distributed real-time systems by PALS do not involve such two phases. Therefore, we use the PALS synchronous model [12] as the underlying synchronized model of both patterns.

We denote by $\mathcal{J} = \{A_i\}_{i \in J}$ a J-indexed family of sets A_i, $i \in J$, with the index set $J = index(\mathcal{J})$. The set $A_i \in \mathcal{J}$ is denoted by $\mathcal{J}(i)$. The Cartesian product $\prod_{i \in J} A_i$ is denoted by $\prod \mathcal{J}$, and an element $\vec{v} \in \prod \mathcal{J}$ is denoted by a J-indexed family $\vec{v} = (v_i)_{i \in J}$, where $v_i \in A_i$, $i \in J$. A synchronous design SD is specified as an collection of state machines with input and output ports.

Definition 2.1 ([12]). A *typed machine* is a tuple $M = (\mathcal{I}, \mathcal{O}, S, \delta_M)$ with (i) $\mathcal{I} = \{I_i\}_{i \in I}$ an *input set*, (ii) $\mathcal{O} = \{O_o\}_{o \in O}$ an *output set*, (iii) S a set of *states*, and (iv) $\delta_M \subseteq ((\prod \mathcal{I}) \times S) \times (S \times (\prod \mathcal{O}))$ a total *transition relation*,[1] where $\mathcal{I}$ and $\mathcal{O}$ are finite families of nonempty sets.

That is, a machine M has input ports $\mathcal{I}$ and output ports $\mathcal{O}$. An input to port $i \in index(\mathcal{I})$ is an element of the set $\mathcal{I}(i)$, and an output from port $o \in index(\mathcal{O})$ is an element of the set $\mathcal{O}(o)$.

Typed machines can be composed into a *machine ensemble* by means of a wiring diagram connecting their input and output ports, as depicted in Fig. 1. Each input port (j, p) (input port p of machine M_j) is assigned the source output port $(l, q) = src(j, p)$ (output port q of machine M_l), provided that the domain of the output is a subset of the corresponding input domain.

Definition 2.2 ([12]). A *machine ensemble* is a tuple $\mathcal{E} = (J \cup \{e\}, \{M_j\}_{j \in J \cup \{e\}}, src)$, where:

- J is a nonempty finite set of *machine indices*, and $e \notin J$ is the *interface index*;
- $M_j = (\mathcal{I}_j, \mathcal{O}_j, S_j, \delta_{M_j})$, $j \neq e$, is a typed machine;
- $M_e = (\mathcal{I}_e, \mathcal{O}_e)$, called an *ensemble interface*, is a pair of input set $\mathcal{I}_e$ and output set $\mathcal{O}_e$;
- $src : \bigcup_{j \in J \cup \{e\}} (\{j\} \times index(\mathcal{I}_j)) \to \bigcup_{j \in J \cup \{e\}} (\{j\} \times index(\mathcal{O}_j))$ is a bijection[2] such that:
  - if $src(j, p) = (l, q)$, then $\mathcal{I}_j(p) \supseteq \mathcal{O}_l(q)$, and

[1] For any input $\vec{i} \in \prod \mathcal{I}$ and state $s \in S$, there exist state $s' \in S$ and output $\vec{o} \in \prod \mathcal{O}$ such that $((\vec{i}, s), (s', \vec{o})) \in \delta_M$.
[2] Unlike [12], an output port is connected to a single input port. Any output port o with multiple destinations $i_1, \ldots, i_n$ can be transformed into multiple output ports $o_1, \ldots, o_n$ with a single destination, where each o_q is connected to i_q.

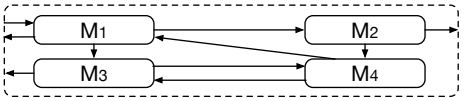


Fig. 1. A machine ensemble of four machines M_1, M_2, M_3, and M_4.

– if $src(j,p) = (e,q)$, then $j \neq e$ (no self-loops from the interface e to itself).

A value for an input port is obtained from the value for the corresponding output port. When $src(m,p) = (l,q)$, the value for input port (m,p) is the value of output port (l,q). The *input projection function* $in_m : \prod_{j\in J} \prod O_j \to \prod \mathcal{I}_m$ gives the values for input set $\mathcal{I}_m$ from the values for output sets $\{O_j\}_{j\in J}$: i.e., for $p \in index(\mathcal{I}_m)$ and $(l,q) = src(m,p)$, $\pi_p(in_m((\vec{o}_j)_{j\in J})) = \pi_q(\vec{o}_l)$.

The synchronous composition of an ensemble $\mathcal{E}$ is a single machine $M_\mathcal{E}$ that simultaneously performs the transitions of all the machines in lock-step. If a machine has a feedback wire to itself or to another machine, not to the interface e, the output becomes an input of the corresponding machine in the next round. To this end, the states of $M_\mathcal{E}$ also include such "feedback" outputs.

Definition 2.3 ([12]). The *synchronous composition* of an ensemble $\mathcal{E} = (J \cup \{e\}, \{M_j\}_{j\in J\cup\{e\}}, src)$ is the typed machine $M_\mathcal{E} = (\mathcal{I}_\mathcal{E}, O_\mathcal{E}, S_\mathcal{E}, \delta_{M_\mathcal{E}})$ such that:

- $\mathcal{I}_\mathcal{E} = O_e$ and $O_\mathcal{E} = \mathcal{I}_e$.
- $S_\mathcal{E} = \prod_{j\in J}(S_j \times \prod_{p\in FO_j} O_j(p))$, where $FO_j = \{p \in index(O_j) \mid (j,p) = src(l,q), l \in J\}$.
- $((\vec{v}_e, (s_j, \vec{v}_j)_{j\in J}), ((s'_j, \vec{v}'_j)_{j\in J}, \vec{v}'_e)) \in \delta_{M_\mathcal{E}}$ iff for each machine $l \in J$, $((\vec{i}_l, s_l), (s'_l, \vec{o}_l)) \in \delta_{M_l}$, where $\vec{i}_l = in_l((\vec{v}_j)_{j\in J\cup\{e\}})$, $\vec{v}'_l = (\pi_p(\vec{o}_l))_{p\in FO_j}$, and $\vec{v}'_e = in_e((\vec{o}_l)_{l\in J})$.

Intuitively, $\delta_{M_\mathcal{E}}$ combines the transitions δ_{M_l} of the single machines in $\mathcal{E}$: the input $\vec{i}_l$ for each machine M_l, $l \in J$, is obtained from the feedback outputs $(\vec{v}_j)_{j\in J}$ in the state and the interface input $\vec{v}_e$, and the feedback part $\vec{v}'_l$ of the resulting output $\vec{o}_l$ is included in the next state of $M_\mathcal{E}$.

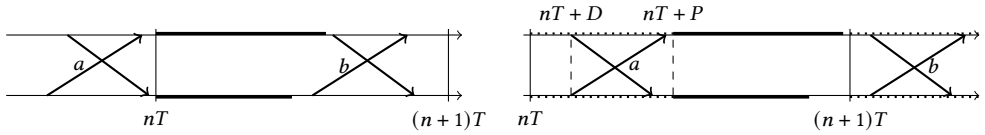
2.2 PALS and TTA Asynchronous Real-Time Models of Computation

In asynchronous distributed real-time models, each machine is equipped with a local clock and I/O buffers, and executes in rounds of a certain period according to its local clock. A local clock of machine M_j is formalized as a *clock function* $c_j : \mathbb{T}_R \to \mathbb{T}_C$, where $\mathbb{T}_R$ is a global *real time* domain, and $\mathbb{T}_C \subseteq \mathbb{T}_R$ is a local *clock time* domain measured by each machine [10, 16]. Intuitively, $c_j(t)$ gives the value of M_j's local clock at global time t. The actions of machines, such as reading input, executing a transition, and sending output, are performed according to its local clock. Following PALS [3, 12], the period of each round is given by a clock time constant $T \in \mathbb{T}_C$.

Both PALS and TTA assume that the underlying infrastructure guarantees bounds on clock skews, execution times, and network delays [12, 16, 18]. The values $\alpha_{\min}, \alpha_{\max} \in \mathbb{T}_C$ denote the minimum and maximum times for processing I/O and executing a transition, measured by the local clocks. The values $\mu_{\min}, \mu_{\max} \in \mathbb{T}_R$ denote the minimum and maximum message transmission delays in global time. PALS and TTA use different conditions for bounds on local clock skews.

2.2.1 PALS Asynchronous Model. There are two assumptions for local clocks in PALS [12]. (1) A local clock $c_j : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is a continuous and monotonic function with respect to $\leq$, where $\mathbb{T}_R = \mathbb{R}_{\geq 0}$ and $\mathbb{T}_C = \mathbb{R}_{\geq 0}$. (2) The skew between any local clock c_j and a global "perfect" clock—which is given by the identity function—is strictly less then a bound $\epsilon > 0$ (for any $t \in \mathbb{R}_{\geq 0}$, $|c_j(t) - t| < \epsilon$).

Each machine M_j begins its n-th round at clock time nT (i.e., at global time t when $c_j(t) = nT$). When a new round begins, machine M_j reads input from the input buffer, executes a transition, and puts the generated output into the output buffer. When the execution of a transition ends, the



(a) PALS (b) TTA

Fig. 2. Timelines for PALS and TTA asynchronous systems; diagonal arrows denote message transmission, thick horizontal lines denote the execution of a transition, and dotted horizontal lines denote Communication Phase; the arrows a denote the inputs for round n, and the arrows b denotes the inputs for round $n+1$.

messages in the output buffers are sent, potentially after some delay (up to $2\epsilon - \mu_{\min}$), to ensure that the output is not sent into the network too early.

As a consequence, given an ensemble $\mathcal{E}$ and performance parameters $\Gamma = (T, \epsilon, \alpha_{\max}, \mu_{\min}, \mu_{\max})$, PALS gives the asynchronous system $PALS(\mathcal{E}, \Gamma)$. To achieve the same behavior as the synchronous system $M_{\mathcal{E}}$, messages generated for round $n+1$ must be received after the corresponding machines have read their inputs for round n and before they read the inputs for round $n+1$. This condition can be ensured by the PALS period constraint: $T \geq \mu_{\max} + 2\epsilon + \max(2\epsilon - \mu_{\min}, \alpha_{\max})$ [3, 12].

2.2.2 TTA Asynchronous Model. TTA assumes the following conditions for local clocks [15, 16, 18]. (1) $c_j : \mathbb{R}_{\geq 0} \to \mathbb{N}$ is monotonic with respect to $\leq$, where $\mathbb{T}_R = \mathbb{R}_{\geq 0}$ and $\mathbb{T}_C = \mathbb{N}$. (2) The skew between two local clocks is bounded by $\Sigma \geq 0$ ($|c_j(t) - c_{j'}(t)| \leq \Sigma$ for any $t \in \mathbb{R}_{\geq 0}$). (3) The drift rate is bounded by $0 < \rho < 1$ ($\lfloor (1-\rho)(t_2 - t_1) \rfloor \leq c_j(t_2) - c_j(t_1) \leq \lceil (1+\rho)(t_2 - t_1) \rceil$ for $t_1 \leq t_2$).

The behavior of each machine M_j is parameterized by constants $D, P \in \mathbb{T}_C$. The machine begins its n-th round at clock time nT,[3] and each round n consists of two phases. In the first phase, called communication phase, M_j sends the output messages, generated in the previous round, at clock time $nT + D$, and receives input messages. In the second phase, called computation phase, M_j executes a transition, based on the inputs received in the first phase, at clock time $nT + P$.

Given an ensemble $\mathcal{E}$ and performance parameters $\Gamma = (T, D, P, \Sigma, \rho, \alpha_{\max}, \mu_{\max})$, we obtain the asynchronous system $TTA(\mathcal{E}, \Gamma)$. To achieve the same behavior as the synchronous system $M_{\mathcal{E}}$, the execution of a transition must be finished before the next round, and a message sent in round n must arrive during the communication phase of the target machine for round n. These conditions can be ensured by the constraints: $P + \alpha_{\max} < T$, $D \geq \Sigma$, and $P > D + \Sigma + (1+\rho)\mu_{\max}$ [16].

3 PRELIMINARIES ON MULTIRATE PALS

In many virtually synchronous CPSs, different sensors and actuators operate at different rates, and controllers need to coordinate such multirate subsystems. Therefore, PALS was extended to the multi-rate setting in [2, 6]: given a multirate synchronous design SD and bounds Γ, Multirate PALS provides the multirate distributed real-time system $PALS(SD, \Gamma)$ that is stuttering bisimilar to SD. This section presents some preliminaries on Multirate PALS.

3.1 Multirate Synchronous Model of Computation

In the *multirate* synchronous model [6], different machines may operate at different rates. A machine with rate $k \geq 1$ performs k transitions when a machine with rate 1 performs one transition. That is, a typed machine with rate $k \geq 1$ is equivalent to a "decelerated" machine, formalized in the

[3] In [16], the beginning of each round is scheduled by a function $sched : \mathbb{N} \to \mathbb{T}_C$, where the n-th round begins at $sched(n)$. To simplify the exposition, we use a period T. All the results in the paper can easily be generalized for $sched$.

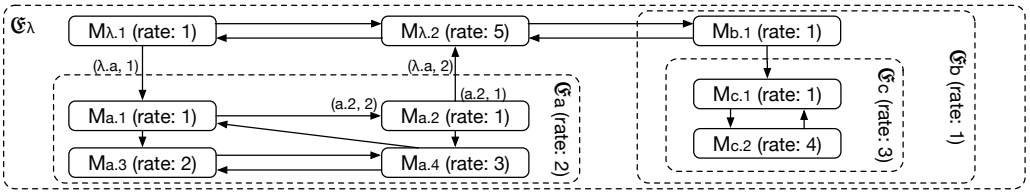


Fig. 3. A hierarchical ensemble $\mathscr{E}$ with four ensembles $\mathfrak{E}_a$, $\mathfrak{E}_b$, $\mathfrak{E}_c$, and $\mathfrak{E}_\lambda$, where $parent_a = \lambda$, $parent_b = \lambda$, $parent_c = b$; $J_\lambda = \{1, 2, a, b\}$, $J_b = \{1, c\}$, $J_c = \{1, 2\}$; $M_{\lambda.a} = M_{SR(\mathfrak{E}_a)}$, $M_{\lambda.b} = M_{SR(\mathfrak{E}_b)}$, $M_{b.c} = M_{SR(\mathfrak{E}_c)}$.

following definition, that reads k input values, performs k transitions, and outputs a k-tuple of values. Let $\mathcal{J}^{\times k} = \{(\mathcal{J}(j))^k\}_{j\in J}$ be a J-indexed family of the k-ary Cartesian powers of sets.

Definition 3.1 ([6]). For a machine $M = (\mathcal{I}, \mathcal{O}, S, \delta_M)$ with $I = index(\mathcal{I})$ and $O = index(\mathcal{O})$, the k-step *deceleration* is $M^{\times k} = (\mathcal{I}^{\times k}, \mathcal{O}^{\times k}, S, \delta_{M^{\times k}})$, where $(((\vec{i}_q)_{q\in I}, s_0), (s_k, (\vec{o}_l)_{l\in O})) \in \delta_{M^{\times k}}$ iff there exist states $s_1, \ldots, s_{k-1} \in S$ such that for each $1 \le i \le k$, $(((\pi_i(\vec{i}_q))_{q\in I}, s_{j-1}), (s_i, (\pi_i(\vec{o}_l))_{l\in O})) \in \delta_M$.

To connect an input port of a rate-k typed machine with an output port of a rate-l machine, a l-tuple output must be adapted into a k-tuple input. An *input adaptor* is a family of functions $\theta = \{\theta_i : \mathcal{I}'(i) \to \mathcal{I}(i)\}_{i\in I}$ from another input set $\mathcal{I}'$ to $\mathcal{I}$, where $I = index(\mathcal{I}) = index(\mathcal{I}')$. Let $\theta(\vec{v}) = (\theta_i(\pi_i(\vec{v})))_{i\in I}$ for $\vec{v} \in \prod \mathcal{I}'$. A machine with an input adaptor is equivalent to an ordinary machine that transforms inputs using the input adaptor and then performs a transition.

Definition 3.2 ([6]). The *adaptor closure* of a typed $M = (\mathcal{I}, \mathcal{O}, S, \delta_M)$ with an input adaptor $\theta : \mathcal{I}' \to \mathcal{I}$ is $M_\theta = (\mathcal{I}', \mathcal{O}, S, \delta_{M_\theta})$ such that $((\vec{i}, s), (s', \vec{o})) \in \delta_{M_\theta}$ iff $((\theta(\vec{i}), s), (s', \vec{o})) \in \delta_M$.

Typed machines with different rates are composed into a multirate machine ensemble. The rate of the ensemble itself is considered to be 1, and only "slow" machines with rate 1 communicate with the outside through the ensemble's interface e. A multirate machine ensemble $\mathfrak{E}$ can be reduced to an ordinary single-rate ensemble $SR(\mathfrak{E})$, where each machine $M_j \in \mathfrak{E}$ with rate ξ_j and input adaptor θ_j corresponds to the adaptor closure of the ξ_j-step deceleration of M_j in $SR(\mathfrak{E})$.

Definition 3.3 ([6]). A *multirate machine ensemble* is $\mathfrak{E} = (J \cup \{e\}, \{M_j, \xi_j, \theta_j\}_{j\in J\cup\{e\}}, src)$, where:

- $\xi_j \ge 1$ is a rate of M_j, and θ_j is an input adaptor for M_j;
- $\xi_e = 1$, and θ_e is a family of identity functions;
- if $src(j, p) = (l, q)$, then $\theta_j(p) : O_l(q)^{\xi_l} \to \mathcal{I}_j(p)^{\xi_j}$; and
- if $src(j, p) = (l, q)$, $j = e$ implies $\xi_l = 1$, and $l = e$ implies $\xi_j = 1$ and $\theta_j(p)$ is the identity;

such that $SR(\mathfrak{E}) = (J \cup \{e\}, \{(M_j^{\times\xi_j})_{\theta_j}\}_{j\in J} \cup \{M_e\}, src)$ is an ordinary (single-rate) ensemble.

The synchronous composition of a multirate ensemble $\mathfrak{E}$ is a single machine, namely, the synchronous composition $M_{SR(\mathfrak{E})}$ of the single-rate ensemble $SR(\mathfrak{E})$. A machine $M_{SR(\mathfrak{E})}$ can also be composed into a multirate ensemble with other machines. Such a tree hierarchy of ensembles, depicted in Fig. 3, can be formalized as a hierarchical multirate ensemble $\mathscr{E}$ as follows.

Definition 3.4 ([6]). A *hierarchical ensemble* is $\mathscr{E} = (H, \lambda, \{parent_h\}_{h\in H\setminus\{\lambda\}}, \{\mathfrak{E}_h\}_{h\in H})$, where

- $(H, \lambda, \{parent_h\}_{h\in H\setminus\{\lambda\}})$ represents a rooted tree with vertices H such that
  - H is a finite set of ensemble indices, where $\lambda \in H$ is the root index; and
  - $parent_h \in H$ denotes the parent of a non-root ensemble $h \in H \setminus \{\lambda\}$.
- $\mathfrak{E}_h = (J_h \cup \{e\}, \{M_{h.j}, \xi_{h.j}, \theta_{h.j}\}_{j\in J_h\cup\{e\}}, src_h)$ is a multirate machine ensemble such that
  - the synchronous composition $M_{SR(\mathfrak{E}_h)}$, $h \in H \setminus \{\lambda\}$, is a machine in $\mathfrak{E}_{parent_h}$, and

– if $src(j,p) = (e,q)$ or $src_h(e,q) = (j,p)$, then $j \notin H$ (not crossing multiple boundaries).

Suppose that an input port $(h.j,p)$ of slow machine $M_{h.j}$, with rate $\xi_{h.j} = 1$, is connected to an output port of an external machine in the parent ensemble $\mathfrak{E}_{parent_h}$ through an input port $(parent_h.h,q)$ of the ensemble $\mathfrak{E}_h$. In this case, the "true" input adaptor of port $(h.j,p)$ with respect to the connected port is the input adaptor $\theta_{parent_h.h}(q)$ for its *representative* $(parent_h.h,q)$.

Definition 3.5. For machine $M_{h.j}$ the *representative* of port $(h.j,p)$ is $rep(h.j,p) = (parent_h.h,q)$, if $(j,p) = src_h(e,q)$ or $(j,p) = src_h^{-1}(e,q)$; otherwise, $rep(h.j,p) = (h.j,p)$. The ensemble $\mathfrak{E}_{\bar{h}}$ of the representative $(\bar{h}.l,q) = rep(h.j,p)$ is called the *context ensemble* of port $(h.j,p)$.

For example, consider the machine $M_{a.2}$ in Fig. 3: the representative of output port $(a.2,1)$ is $rep(a.2,1) = (\lambda.a,2)$ and its representative rate is $\xi_{\lambda.a} = 2$, whereas the representative of input port $(a.2,2)$ is the same port $rep(a.2,2) = (a.2,2)$ and the representative rate is $\xi_{a.2} = 1$.

3.2 Multirate PALS Asynchronous Real-Time Model of Computation

Given a multirate ensemble $\mathfrak{E}$ and PALS parameters $\Gamma = (T, \epsilon, \alpha_{\max}, \mu_{\min}, \mu_{\max})$, Multirate PALS [2, 6] provides the multirate asynchronous system $PALS(\mathfrak{E},\Gamma)$. In this asynchronous model, each M_j with rate ξ_j mimics the behavior of the adaptor closure of the ξ_j-step deceleration. $PALS(\mathfrak{E},\Gamma)$ is basically the same as the single-rate system $PALS(SR(\mathfrak{E}),\Gamma)$, but with one important difference: each machine M_j performs an internal transition at the beginning of each "fast" period T/ξ_j.

It may happen that M_j *cannot* finish all of its ξ_j internal transitions in a slow period T before the output messages must be sent. When M_j can only send $k_j + 1 \le \xi_j$ outputs before the deadline, the input adaptor of the target machine must ignore the last $\xi_j - k_j - 1$ values in a ξ_j-tuple input. The number k_j, called an *output cutoff number*, is given by $k_j = \lfloor \max(0, T - (2\epsilon + \mu_{\max} + \alpha_j^{\max}))/(T/\xi_j) \rfloor$, where $\alpha_j^{\max}$ denotes the maximum execution time for the fast machine M_j [6].

The multirate asynchronous system $PALS(\mathfrak{E},\Gamma)$ and the synchronous system $M_{SR(\mathfrak{E})}$ achieve the same behavior, if all input adaptors satisfy the above requirement and Γ satisfies the PALS period constraint [6]. It is also shown that a multirate asynchronous system can be built for a hierarchical multirate ensemble $\mathscr{E}$, using the same performance parameters for all subensembles [6]. However, such a multirate asynchronous model has not yet been developed for TTA.

4 UNIFIED MULTIRATE ASYNCHRONOUS REAL-TIME MODEL

This section presents a generic asynchronous real-time model for hierarchical multirate ensembles. Our model generalizes the previous asynchronous real-time model for Multirate PALS [6], explained in Sec. 3.2, allowing different machines to have different communication and computation times for TTA. It can specify single-rate TTA systems as well as (single-rate and multirate) PALS systems. Moreover, multirate TTA systems, defined in Sec. 5.2, can be naturally specified in our model.

4.1 Asynchronous Real-Time Systems

For a local clock function $c_{h.j} : \mathbb{T}_R \to \mathbb{T}_C$, we assume that: (1) $\mathbb{T}_R$ and $\mathbb{T}_C$ are ordered commutative monoids with a minimum element 0; (2) $\mathbb{T}_R$ is complete; (3) $c_{h.j}$ is monotonic (for any $t_1 \le t_2$, $c_{h.j}(t_1) \le c_{h.j}(t_2)$); and (4) $c_{h.j}$ is unbounded (for any $u \in \mathbb{T}_C$, $u \le c_{h.j}(t)$ for some $t \in \mathbb{T}_R$). Typical time domains (e.g., $\mathbb{N}$, $\mathbb{Q}_{\ge 0}$, and $\mathbb{R}_{\ge 0}$) are such monoids, and $\mathbb{R}_{\ge 0}$ is complete. A clock synchronization condition is considered a performance parameter, not part of the assumptions.

The asynchronous system $\mathcal{A}(\mathscr{E})$ for a hierarchical ensemble $\mathscr{E}$ is composed of concurrent objects. Each machine object $\mathcal{A}(M_{h.j})$ has a local clock $c_{h.j}$, an input buffer for each input port, and an output buffer for each output port. Each port $(h.j,p)$ is assigned its representative $(\bar{h}.l,q) = rep(h.j,p)$, a

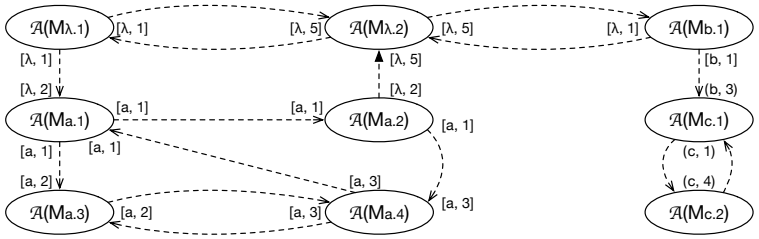


Fig. 4. An asynchronous system $\mathcal{A}(\mathscr{E})$; each port $(h.j, p)$ is annotated with a pair $[\bar{h}, k]$ of its context ensemble $\bar{h}$ and representative rate $\xi_{\bar{h}.l}$, where dashed lines denote connections from output ports to input ports.

representative rate $\xi_{\bar{h}.l}$, and, if it is an input port, a representative input adaptor $\theta_{\bar{h}.l}(q)$. For example, the hierarchical ensemble in Fig. 3 corresponds to the asynchronous system in Fig. 4.

A "global" state of $\mathcal{A}(\mathscr{E})$ consists of a current global time, and a *configuration* of machine objects—including messages in I/O buffers—and traveling messages from/to the machine objects. The buffer of a port $(\bar{h}.l, q) = rep(h.j, p)$, with representative rate $\xi_{\bar{h}.l}$, contains a vector of at most $\xi_{\bar{h}.l}$ values. Each traveling message involves a global time value indicating when the message arrives at its destination. An asynchronous system $\mathcal{A}(\mathscr{E})$ is specified as a transition system as follows.

Definition 4.1. An asynchronous real-time system is a tuple $\mathcal{A}(\mathscr{E}) = (S_{\mathcal{A}(\mathscr{E})}, \longrightarrow_{\mathcal{A}(\mathscr{E})})$ such that:

- $S_{\mathcal{A}(\mathscr{E})}$ is the set of global states of the form $\{C; t\}$, where C is a configuration and $t \in \mathbb{T}_R$;
- $\longrightarrow_{\mathcal{A}(\mathscr{E})} \subseteq S_{\mathcal{A}(\mathscr{E})} \times S_{\mathcal{A}(\mathscr{E})}$ is the set of system transitions, each of which is either:
  - a *discrete transition* $\{C; t\} \longrightarrow_{\mathcal{A}(\mathscr{E})} \{C'; t\}$ that models instantaneous actions, or
  - a *tick transition* $\{C; t\} \longrightarrow_{\mathcal{A}(\mathscr{E})} \{C; t'\}$, $t' \geq t$, that models the elapse of global time.

An *execution trace* (or simply trace) of an asynchronous real-time system $\mathcal{A}(\mathscr{E})$ is a finite or infinite alternating sequence of tick and discrete transitions of the form

$$\{C_0; t_0\} \longrightarrow_{\mathcal{A}(\mathscr{E})} \{C_0; t_1\} \longrightarrow_{\mathcal{A}(\mathscr{E})} \{C_1; t_1\} \longrightarrow_{\mathcal{A}(\mathscr{E})} \{C_1; t_2\} \longrightarrow_{\mathcal{A}(\mathscr{E})} \{C_2; t_2\} \longrightarrow_{\mathcal{A}(\mathscr{E})} \cdots$$

where $t_0 \leq t_1 \leq t_2 \leq t_3 \leq \cdots$. Intuitively, discrete transitions for machine objects and traveling messages are "scheduled" according to the current global time and local clocks, and tick transitions occur between those discrete transitions to advance global time.

4.2 Discrete Transitions

Each machine object $\mathcal{A}(M_{h.j})$ operates at its own rate according to its local clock $c_{h.j}$. A discrete transition of an asynchronous real-time system $\mathcal{A}(\mathscr{E})$ corresponds to an "action" of a single machine object. A tick transition advances the global time as long as no such actions are possible. These actions can change the states of machine objects, or generate/remove messages from the configuration. There are five kinds of actions for objects and messages as follows:

`newRound` A new round of $\mathcal{A}(M_{h.j})$ begins.
`beginExec` $\mathcal{A}(M_{h.j})$ applies input adaptors, reads input, and starts the execution of a transition.
`endExec` The execution of a transition finishes and the outputs are put into the output buffers.
`outputMsg` The messages in the buffer of an output port are sent.
`recvMsg` A received message is stored in the buffer of an input port.

4.2.1 Beginning a Round. Each machine object $\mathcal{A}(M_{h.j})$ is assigned a clock time *period* $T_{h.j} \in \mathbb{T}_C$. $\mathcal{A}(M_{h.j})$ begins the n-th round at local clock time $nT_{h.j}$ or, equivalently, at global time $t \in \mathbb{T}_R$ such that $c_{h.j}(t) = nT_{h.j}$ (newRound). The machine objects in the same ensemble are scheduled at the

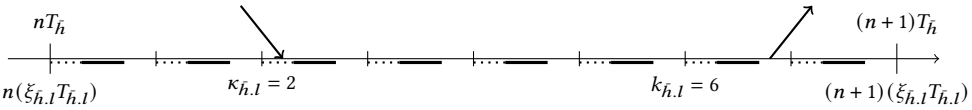


Fig. 5. Timeline for input/output port with $\xi_{\bar{h}.l} = 8$, where dotted horizontal lines denote the delays by $P_{\bar{h}.l}$.

same clock time when they are (hierarchically) synchronized according to their rates. That is, each multirate ensemble $\mathfrak{E}_h$ is assigned a clock time period $T_h \in \mathbb{T}_C$, where:

$$T_h = \xi_{h.j} \cdot T_{h.j}, \qquad T_h = T_{parent_h.h} \tag{1}$$

Consider the asynchronous system in Fig. 4, associated with the hierarchical ensemble in Fig. 3. Suppose the period of the root ensemble $\mathfrak{E}_\lambda$ is $T_\lambda = 60$. The period of $\mathfrak{E}_c$ is $T_c = 20$, and the periods of the machine objects in $\mathfrak{E}_c$ are $T_{c.1} = 20$ and $T_{c.2} = 5$. $\mathcal{A}(M_{c.1})$ begins the n-th round at global time t with $c_{c.1}(t) = 20n$, and $\mathcal{A}(M_{c.2})$ begins the m-th round at global time t with $c_{c.2}(t) = 5m$.

4.2.2 Executing a Transition. Each machine object $\mathcal{A}(M_{h.j})$ is assigned a clock time offset $P_{h.j} \in \mathbb{T}_C$ indicating when the execution of $\mathcal{A}(M_{h.j})$'s transition begins within a round, and clock time bounds $\alpha^{\min}_{h.j}, \alpha^{\max}_{h.j} \in \mathbb{T}_C$ indicating the minimum and maximum times, respectively, for processing input, executing a transition, and generating output. Unlike (single-rate) TTA and PALS, different components, possibly with different rates, can have different values for $P_{h.j}$, $\alpha^{\min}_{h.j}$, and $\alpha^{\max}_{h.j}$.

In each round n, machine object $\mathcal{A}(M_{h.j})$ begins the execution of a transition—including input processing with input adaptors—at clock time $nT_{h.j} + P_{h.j}$ (`beginExec`). If a given execution of $\mathcal{A}(M_{h.j})$ takes clock time $\alpha_{h.j}$, then the execution—including output generation—finishes at clock time $nT_{h.j} + P_{h.j} + \alpha_{h.j}$, where $\alpha^{\min}_{h.j} \le \alpha_{h.j} \le \alpha^{\max}_{h.j}$ (`endExec`). We require that the execution of a transition must finish before the next round of $\mathcal{A}(M_{h.j})$:

$$0 \le P_{h.j} + \alpha^{\max}_{h.j} < T_{h.j} \tag{2}$$

Consider an input port $(h.j, p)$ of machine object $\mathcal{A}(M_{h.j})$ with representative rate $\xi_{\bar{h}.l}$, where $(\bar{h}.l, q) = rep(h.j, p)$. It receives a *single* input message for each "slow" round of a period $T_{\bar{h}}$ for the context ensemble $\bar{h}$. The representative input adaptor $\theta_{\bar{h}.l}$ is then applied to obtain a $\xi_{\bar{h}.l}$-tuple value from the received input. For each "fast" round of a period $T_{h.j}$, the i-th item of the $\xi_{\bar{h}.l}$-tuple value in the input port is used to perform a transition, if it is the i-th fast round within a slow round.

Unlike Multirate PALS [2, 6], where all input messages must arrive before the beginning of a slow round, in our model, an input message may arrive after executing $\kappa_{\bar{h}.l}$ internal transitions, where $0 \le \kappa_{\bar{h}.l} < \xi_{\bar{h}.l}$. For example, in Fig. 5, the input message has arrived after executing $\kappa_{\bar{h}.l} = 2$ transitions. In this case, the input adaptor $\theta_{\bar{h}.l}$ can only be applied when the input message is available from the $(\kappa_{\bar{h}.l} + 1)$-th fast round. The number $\kappa_{\bar{h}.l}$ is called an *input cutoff number*.

Therefore, when $\mathcal{A}(M_{h.j})$ starts executing a transition, for each input port $(\bar{h}.l, q) = rep(h.j, p)$, the input adaptor $\theta_{\bar{h}.l}$ is applied in the $(\kappa_{\bar{h}.l} + 1)$-th fast round—when a message is first available in the buffer of the input port—to obtain a $\xi_{\bar{h}.l}$-tuple value for the input port. For the first $\kappa_{\bar{h}.l}$ fast rounds, where no value is available in the input port, $\mathcal{A}(M_{h.j})$ uses some default value (e.g., "don't care" value $\perp$) as the value of the input port to perform a transition.

4.2.3 Sending and Receiving Messages. Each output port $(h.j, p)$, where $(\bar{h}.l, q) = rep(h.j, p)$, sends a single output message (with a $\xi_{\bar{h}.l}$-tuple value) for each slow round. A function $D_{\bar{h}.l} : \mathbb{T}_C \to \mathbb{T}_C$ gives the clock time offset indicating when a message is sent. In each slow round n, a message is sent from the output port $(\bar{h}.l, q) = rep(h.j, p)$ at local clock time $nT_{\bar{h}} + D_{\bar{h}.l}(\alpha_{h.j})$, given $\mathcal{A}(M_{h.j})$'s

execution time $\alpha_{h.j}$ (outputMsg). An output message must be sent before the next slow round:

$$\forall \alpha_{h.j} \in [\alpha_{h.j}^{\min}, \alpha_{h.j}^{\max}].\ D_{\bar{h}.l}(\alpha_{h.j}) < T_{\bar{h}} \tag{3}$$

It may happen that $\mathcal{A}(M_{h.j})$ cannot finish all of $\xi_{\bar{h}.l}$ internal transitions before the message must be sent at local clock time $nT_{\bar{h}} + D_{\bar{h}.l}(\alpha_{h.j})$. If only $k_{\bar{h}.l} + 1 \leq \xi_{\bar{h}.l}$ values are available in the output buffer, then, like Multirate PALS [6], $\mathcal{A}(M_{h.j})$ sends a $\xi_{\bar{h}.l}$-tuple of values that are padded with $\xi_{\bar{h}.l} - k_{\bar{h}.l} - 1$ "don't care" values $\perp$, where $k_{\bar{h}.l}$ is called an *output cutoff number*. For example, in Fig. 5, $k_{\bar{h}.l} = 6$, and the output message is sent after executing 7 internal transitions.

Each ensemble $\bar{h}$ is assigned real time bounds $\mu_{\bar{h}}^{\min}, \mu_{\bar{h}}^{\max} \in \mathbb{T}_R$ indicating the minimum and maximum delays, respectively, for point-to-point message transmission. Unlike TTA and PALS, different ensembles, which may correspond to different system hierarchies, can have different bounds on network delays. A message sent at global time t arrives at global time $t + \mu_{\bar{h}}$, where $\mu_{\bar{h}}^{\min} \leq \mu_{\bar{h}} \leq \mu_{\bar{h}}^{\max}$, and is put into the buffer of the corresponding input port (recvMsg).

4.3 Tick Transitions and System Parameters

Consider a global state $\{C; t\}$ of $\mathcal{A}(\mathscr{E})$. Each object $\mathcal{A}(M_{h.j})$ involves the smallest clock time $\nu_{h.j}$ at which the "next" action can occur. Because $c_{h.j}$ is monotonic and unbounded and $\mathbb{T}_R$ is complete, the maximum amount of global time that can elapse before any action for $\mathcal{A}(M_{h.j})$ must occur is well defined; namely, $\sup(\{\tau_{h.j} \mid c_{h.j}(t + \tau_{h.j}) \leq u_{h.j}\})$. Let $mte(C)$ be the maximum amount of global time that can elapse before any action for *all* machine objects and traveling messages must occur. A tick transition is then defined by the rule: $\{C; t\} \to \{C; t + \Delta\}$ **if** $\Delta \leq mte(C)$.

As a consequence, given a hierarchical ensemble $\mathscr{E} = (H, \lambda, \{parent_h\}_{h \in H \setminus \{\lambda\}}, \{\mathfrak{E}_h\}_{h \in H})$, the behavior of the associated asynchronous real-time system $\mathcal{A}(\mathscr{E})$ is determined by the system parameters $\mathscr{B} = \{\mathfrak{B}_h\}_{h \in H}$, where for each ensemble $h \in H$:

$$\mathfrak{B}_h = (T_h, \mu_h^{\min}, \mu_h^{\max}, \{c_{h.j}, D_{h.j}, P_{h.j}, \alpha_{h.j}^{\min}, \alpha_{h.j}^{\max}\}_{j \in J_h \setminus H}).$$

We write $\mathcal{A}(\mathscr{E}, \mathscr{B})$ to denote the asynchronous real-time system with the system parameters $\mathscr{B}$. Different asynchronous systems can be obtained by instantiating these parameters.

5 MULTIRATE SYNCHRONIZERS: PALS AND TTA

This section presents two instances of asynchronous real-time systems $\mathcal{A}(\mathscr{E}, \mathscr{B})$. Section 5.1 shows that Multirate PALS [2, 6] is indeed an instance of our framework. Section 5.2 presents a new TTA-based synchronizer for a hierarchical multirate ensemble $\mathscr{E}$. Both synchronizers are defined as a set of real-time constraints on system parameters $\mathscr{B}$. For both cases, we show the lower bound on a period of each ensemble for global synchronous computation.

5.1 Multirate PALS Revisited

Multirate PALS asynchronous systems, explained in 3.2, can be characterized by constraints on system parameters $c_{h.j}$, $D_{h.j}$, and $P_{h.j}$. The skew between any local clock $c_{h.j}$ and a global perfect clock is strictly bounded by $\epsilon > 0$.[4] Each machine object $\mathcal{A}(M_{h.j})$ executes a transition immediately when a new round begin. A message is sent from each output port $(\bar{h}.l, q) = rep(h.j, p)$ after $\mathcal{A}(M_{h.j})$ executes the $(k_{\bar{h}.l} + 1)$-th transition, where $k_{\bar{h}.l}$ is the output cutoff number.

Definition 5.1. An asynchronous real-time system $\mathcal{A}(\mathscr{E}, \mathscr{B})$ is a *PALS system* if there exist a bound $\epsilon > 0$ and output cutoff numbers $\{k_{h.j}\}_{h \in H, j \in J_h}$ such that for each machine $h.j$:

- $|c_{h.j}(t) - t| < \epsilon$, for each $t \in \mathbb{T}_R$,
- $P_{h.j} = 0$, and

[4] The continuity of local clocks is not needed. Instead, the completeness of real time domain $\mathbb{T}_R$ is assumed in this paper.

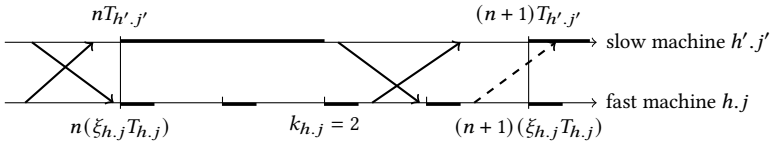


Fig. 6. Timeline for multirate PALS with $\xi_{h.j} = 4$ and $\xi_{h'.j'} = 1$

- $D_{\bar{h}.l}(\alpha_{h.j}) = \max(2\epsilon - \mu_{\bar{h}}^{\min},\ k_{\bar{h}.l}T_{\bar{h}.l} + \alpha_{h.j})$, for each output port $(\bar{h}.l, q) = rep(h.j, p)$.

It is worth noting that our definition generalizes the previous Multirate PALS framework [2, 6]: different ensembles can have different performance bounds, as explained in Sec. 4.2. This is very useful in practice, because higher-level controllers can have more relaxed performance bounds, e.g., longer execution times, greater network delays, and so on.

5.1.1 Input Adaptor Constraints. Consider a connection from an output port $(\bar{h}.l, q) = rep(h.j, p)$ to an input port $(\bar{h}.l', q') = rep(h'.j', p')$ with an output cutoff number $k_{\bar{h}.l} < \xi_{\bar{h}.l}$. As explained in Sec. 4.2.3, $\mathcal{A}(M_{h.j})$ sends a $\xi_{\bar{h}.l}$-tuple message that is padded with $\xi_{\bar{h}.l} - k_{\bar{h}.l} - 1$ "don't care" values $\perp$, because $\mathcal{A}(M_{h.j})$ may be able to only perform $(k_{\bar{h}.l} + 1)$ internal transitions before the deadline to send the message in the current slow round (see Fig. 6).

To ensure the same behavior as the synchronous system, the representative input adaptor $\theta_{\bar{h}.l'}(q')$ of the input port should only consider the first $(k_{\bar{h}.l} + 1)$ values and ignore the last $\xi_{\bar{h}.l} - k_{\bar{h}.l} - 1$ values in a $\xi_{\bar{h}.l}$-tuple input [6]. That is, for any two inputs $\vec{v_1}$ and $\vec{v_2}$, if the first $(k_{\bar{h}.l} + 1)$ items are the same, then the results of applying the input adaptor $\theta_{\bar{h}.l'}(q')$ must be the same:

$$(\pi_j(\vec{v_1}) = \pi_j(\vec{v_2}) \ \text{ for any } 1 \le j \le k_{\bar{h}.l} + 1) \implies \theta_{\bar{h}.l'}(q')(\vec{v_1}) = \theta_{\bar{h}.l'}(q')(\vec{v_2}) \tag{4}$$

5.1.2 Real-Time Constraints. Suppose that a message is sent in slow round n from an output port $(\bar{h}.l, q) = rep(h.j, p)$ at global time t, and arrives at an input port $(\bar{h}.l', q') = rep(h'.j', p')$ at global time $t + \mu_{\bar{h}}$. Then, $c_{h.j}(t) = nT_{\bar{h}} + D_{\bar{h}.l}(\alpha_{h.j})$. Because $|c_{h.j}(t) - t| < \epsilon$ and $\mu_{\bar{h}}^{\min} \le \mu_{\bar{h}} \le \mu_{\bar{h}}^{\max}$:

$$c_{h.j}(t) - \epsilon + \mu_h^{\min} < t + \mu_{\bar{h}} < c_{h.j}(t) + \epsilon + \mu_h^{\max}.$$

The message should be received after $\mathcal{A}(M_{h'.j'})$ has read its input for slow round n (say, at global time u), but before $\mathcal{A}(M_{h'.j'})$ reads its input for slow round $n+1$ (say, at global time v); i.e., we need $u < t + \mu_{\bar{h}} < v$. Because $P_{h'.j'} = 0$, $c_{h'.j'}(u) = nT_{\bar{h}}$ and $c_{h'.j'}(v) = (n+1)T_{\bar{h}}$. By the above condition, $|c_{h'.j'}(v) - v| < \epsilon$, and $|c_{h'.j'}(u) - u| < \epsilon$, the constraint $u < t + \mu_{\bar{h}} < v$ can be ensured by:

$$nT_{\bar{h}} + \epsilon \le nT_{\bar{h}} + D_{\bar{h}.l}(\alpha_{h.j}) - \epsilon + \mu_h^{\min} \qquad (n+1)T_{\bar{h}} - \epsilon \ge nT_{\bar{h}} + D_{\bar{h}.l}(\alpha_{h.j}) + \epsilon + \mu_h^{\max}$$

The former condition can be rewritten into $D_{\bar{h}.l}(\alpha_{h.j}) \ge 2\epsilon - \mu_{\bar{h}}^{\min}$, which holds by Def. 5.1. By the latter condition, Def. 5.1, Condition (1), and $\alpha_{h.j} \le \alpha_{h.j}^{\max}$, we obtain: $T_{\bar{h}} \ge \mu_{\bar{h}}^{\max} + 4\epsilon - \mu_{\bar{h}}^{\min}$ and $T_{\bar{h}} \ge (\mu_{\bar{h}}^{\max} + 2\epsilon + \alpha_{h.j}^{\max})/(1 - k_{\bar{h}.l}/\xi_{\bar{h}.l})$. Consequently, we obtain the Multirate PALS period bound (which is equivalent to the single-rate PALS period bound [12] if $\xi_{\bar{h}.l} = 1$ and $k_{\bar{h}.l} = 0$):

$$T_{\bar{h}} \ge \max(\mu_{\bar{h}}^{\max} + 4\epsilon - \mu_{\bar{h}}^{\min},\ (\mu_{\bar{h}}^{\max} + 2\epsilon + \alpha_{h.j}^{\max})/(1 - k_{\bar{h}.l}/\xi_{\bar{h}.l})) \tag{5}$$

The above condition gives the smallest possible bound for each ensemble in given a hierarchical ensemble $\mathscr{E}$. It is worth nothing this optimal bound for the multirate case is not known in the previous work [2, 6]. Specifically, only the upper bound of output cutoff numbers is shown in [6]: $k_{\bar{h}.l} \le (T_{\bar{h}} - (2\epsilon + \mu_{\bar{h}}^{\max} + \alpha_{h.j}^{\max}))/(T_{\bar{h}}/\xi_{\bar{h}.l})$ [6], which can be easily derived from Condition (5).

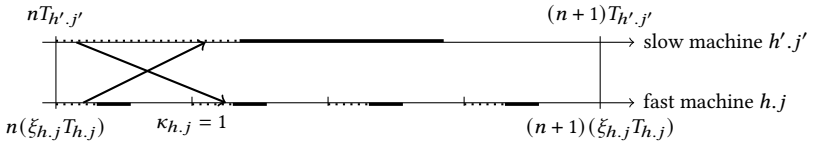


Fig. 7. Timeline for multirate TTA with $\xi_{h.j} = 4$ and $\xi_{h'.j'} = 1$.

5.2 Multirate TTA

We define TTA-based multirate asynchronous systems characterized by constraints on system parameters $c_{h.j}$, $D_{h.j}$, and $P_{h.j}$, where the underlying synchronous design is given by a hierarchical multirate ensemble $\mathscr{E}$. The skew between two local clocks is bounded by $\Gamma \geq 0$, and the drift rate of any local clock is bounded by $0 < \rho < 1$. Each round consists of two phases. In the communication phase, the machine objects in an ensemble $\mathfrak{E}_h$ send messages after some clock time delay D. In the computation phase, each machine object executes a transition.

Definition 5.2. An asynchronous real-time system $\mathscr{A}(\mathscr{E}, \mathscr{B})$ is a *TTA system*, if for $\Gamma \geq 0$, $0 < \rho < 1$, $D \in \mathbb{T}_C$, and input cutoff numbers $\{\kappa_{h.j}\}_{h \in H, j \in J_h}$:

- $|c_{h_1.j_1}(t) - c_{h_2.j_2}(t)| \leq \Sigma$, for any $t \in \mathbb{T}_R$,
- $(1-\rho)(t_1 - t_2) \leq c_{h_1.j_1}(t_1) - c_{h_2.j_2}(t_2) \leq (1+\rho)(t_1 - t_2)$,[5] for any $t_1 \leq t_2$,
- $P_{h.j} > D$ (the offset for the computation phase), and
- $D_{\bar{h}.l}(\alpha_{h.j}) = D$, for each output port $(\bar{h}.l, q) = rep(h.j, p)$.

In the above definition, we include input cutoff numbers $\{\kappa_{h.j}\}_{h \in H, j \in J_h}$, because they appear in input adaptor and real-time constraints for TTA. Also, as explained in Sec. 4.2, different ensembles can have different performance bounds on network delays and execution times.

5.2.1 Input Adaptor Constraints. Consider a connection to an input port $(\bar{h}.l, q) = rep(h.j, p)$ from an output port $(\bar{h}.l', q') = rep(h'.j', p')$ with an input cutoff number $\kappa_{\bar{h}.l} < \xi_{\bar{h}.l}$. A message may not be available in the input port before the computation phase of $\mathscr{A}(M_{h.j})$ for the first $\kappa_{\bar{h}.l}$ fast rounds. In Fig. 7, the output message sent from the slow machine with rate 1 only can arrive before the fast machine with rate 4 starts the computation phase of the second first round.

As explained in Sec. 4.2.2, some default value $\perp$ is used as the value of the input port $(\bar{h}.l, q)$ to execute a transition of $\mathscr{A}(M_{h.j})$ in the first $\kappa_{\bar{h}.l}$ fast rounds. To ensure the same behavior as the synchronous system, the representative input adaptor $\theta_{\bar{h}.l}(q)$ should always give some default value $\perp$ for the first $\kappa_{\bar{h}.l}$ items of a $\xi_{\bar{h}.l}$-tuple result. That is, for any input $\vec{v}$:

$$\text{for any } 1 \leq j \leq \kappa_{\bar{h}.l}, \quad \pi_j(\theta_{\bar{h}.l}(q)(\vec{v})) = \perp \tag{6}$$

5.2.2 Real-Time Constraints. Suppose that a message is sent in slow round n from an output port $(\bar{h}.l', q') = rep(h'.j', p')$ at global time t, and arrives at an input port $(\bar{h}.l, q) = rep(h.j, p)$ at global time $t + \mu_{\bar{h}}$. Then, $c_{h'.j'}(t) = nT_{\bar{h}} + D$. Because $c_{h.j}$ is monotonic, $|c_{h.j}(t) - c_{h'.j'}(t)| \leq \Sigma$, $\mu_{\bar{h}} \leq \mu_{\bar{h}}^{\max}$, and $c_{h.j}(t + \mu_{\bar{h}}) - c_{h.j}(t) \leq (1+\rho)\mu_{\bar{h}}$, we obtain the following inequalities:

$$nT_{\bar{h}} + D - \Sigma \leq c_{h.j}(t + \mu_{\bar{h}}) \leq nT_{\bar{h}} + D + \Sigma + (1+\rho)\mu_{\bar{h}}^{\max}$$

The message should be received *after* $\mathscr{A}(M_{h.j})$ has begun the slow round n (at local clock time $nT_{\bar{h}}$) but *before* $\mathscr{A}(M_{h.j})$ starts the computation phase of the $(\kappa_{\bar{h}.l} + 1)$-th fast round (at local clock

[5] To simplify the exposition, we assume that $\mathbb{T}_C$ is dense and omit the floor and ceiling operators.

time $T_{\bar{h}.l}(n\xi_{\bar{h}.l} + \kappa_{\bar{h}.l}) + P_{h.j}$). That is, we need: $nT_{\bar{h}} \le c_{h.j}(t + \mu_{\bar{h}}) < T_{\bar{h}.l}(n\xi_{\bar{h}.l} + \kappa_{\bar{h}.l}) + P_{h.j}$. By the above inequalities and $T_{\bar{h}} = \xi_{\bar{h}.l}T_{h.j}$ (Condition (1)), this constraint can be ensured by:

$$D \ge \Sigma, \qquad P_{h.j} > D + \Sigma + (1+\rho)\mu_{\bar{h}}^{\max} - \kappa_{\bar{h}.l}T_{\bar{h}.l} \tag{7}$$

Notice that when $\kappa_{\bar{h}.l} = 0$, these conditions are exactly the same as the usual TTA constraints [16]: $D \ge \Sigma$ and $P_{h.j} > D + \Sigma + (1+\rho)\mu_{\bar{h}}^{\max}$. Finally, because Condition (1) and $P_{h.j} + \alpha_{h.j}^{\max} \le T_{h.j}$ (Condition (2)), we obtain the following Mulitrate TTA period bound:

$$T_{\bar{h}} > (2\Sigma + (1+\rho)\mu_{\bar{h}}^{\max} + \alpha_{h.j}^{\max})/((\kappa_{\bar{h}.l} + 1)/\xi_{\bar{h}.l}) \tag{8}$$

5.3 Comparing Multirate PALS and Multirate TTA

For the single-rate case, the main difference between PALS and TTA is that, as depicted in Fig. 2, messages for one round are sent at the same clock time in TTA, whereas they can be sent at different clock times in PALS. As a result, the smallest possible period of the PALS asynchronous system is smaller than the smallest possible period of the TTA asynchronous system. Other differences between PALS and TTA, such as clock constraints, are mostly interchangeable [18].

For the multirate case, PALS and TTA have different input adaptor constraints. Multirate PALS uses input adaptors with *output cutoff numbers* $k_{h.j}$ to ignore outputs values generated by the last $k_{h.j}$ fast rounds. Multirate TTA, on the contrary, uses input adaptors with *input cutoff numbers* $\kappa_{h.j}$ to ignore input values received in the first $\kappa_{h.j}$ fast rounds. Accordingly, the corresponding synchronous systems for Multirate PALS and Multirate TTA may have different behaviors.

These two approaches have complementary advantages and disadvantages. Suppose that a fast component M_F is governed by a slow controller M_S. When M_F should react to all outputs of M_S, the output cutoff approach is useful (e.g., when M_F is an actuator). When M_S should collect all outputs of M_F, the input cutoff approach is suitable (e.g., when M_F is a sensor). Therefore, both kinds of input adaptors with input and output cutoff numbers need to be supported.

6 A GENERALIZED MULTIRATE SYNCHRONIZER

As explained above, sometimes it is necessary to support *both* "skipping" the first inputs from faster components and skipping the last outputs to slower components. This section therefore presents a new multirate synchronizer for asynchronous real-time systems $\mathcal{A}(\mathscr{E}, \mathscr{B})$, defined using real-time constraints on system parameters $\mathscr{B}$. Combining the ideas from PALS and TTA, our synchronizer enjoys the advantages of both input and output cutoff approaches. Furthermore, this generalization allows for a much short period of the asynchronous system $\mathcal{A}(\mathscr{E}, \mathscr{B})$ than PALS and TTA.

6.1 Mixed Multirate Synchronizer (MSYNC)

The key idea of our synchronizer is simple: send messages early (as in PALS), but delay execution if necessary (as in TTA). Fig. 8 illustrates how extra delay before execution can reduce the smallest possible period without loss of any information. In PALS and TTA, one fast transition must be ignored ($k_{h.j} = 2$ in Fig. 6, and $\kappa_{h.j} = 1$ in Fig. 7). However, with a small delay before execution of the slow machine, all fast transitions can be performed before sending the message.

Based on this observation, we present the *mixed multirate synchronizer* (MSYNC). Since the clock constraints for PALS and TTA are interchangeable [18], we use the clock constraints of PALS. Each machine object $\mathcal{A}(M_{h.j})$ executes a transition, potentially after some delay $P_{h.j} \ge 0$, when a new round begins. For each output port with an output cutoff number k, a message is sent when $\mathcal{A}(M_{h.j})$ finishes the $(k+1)$-th fast transition, potentially after some delay.

Definition 6.1. An asynchronous real-time system $\mathcal{A}(\mathscr{E}, \mathscr{B})$ is a *MSYNC system*, if there exist a bound $\epsilon > 0$ and input/output cutoff numbers $\{\kappa_{h.j}, k_{h.j}\}_{h \in H, j \in J_h}$ such that for each $h.j$:

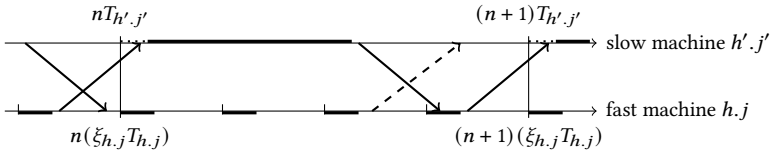


Fig. 8. Timeline for MSYNC with $\xi_{h.j} = 4$ and $\xi_{h'.j'} = 1$

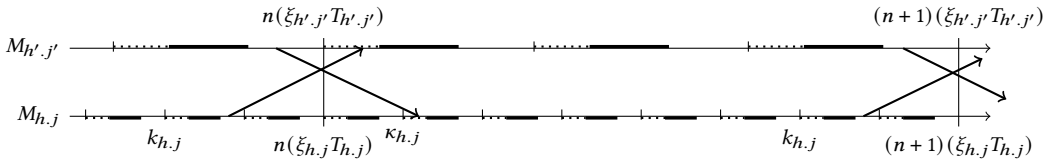


Fig. 9. Timeline for MSYNC with $\xi_{h.j} = 8$ and $\xi_{h'.j'} = 3$.

- $|c_{h.j}(t) - t| < \epsilon$, for any $t \in \mathbb{T}_R$,
- $P_{h.j} \leq D_{h.j}(0)$, and
- $D_{\bar{h}.l}(\alpha_{h.j}) = \max\left(delay,\ k_{\bar{h}.l}T_{\bar{h}.l} + P_{h.j} + \alpha_{h.j}\right)$, for each output port $(\bar{h}.l, q) = rep(h.j, p)$, where $delay = 2\epsilon - \mu^{\min}_{\bar{h}} + \kappa_{\bar{h}.l'}T_{\bar{h}.l'} + P_{h'.j'}$.

Figure 9 depicts a high-level view of the timeline with input cutoff number $\kappa_{h.j}$ and output cutoff number $k_{h.j}$. The input message from $\mathcal{A}(M_{h'.j'})$ is received before $\mathcal{A}(M_{h.j})$ executes the $(\kappa_{h.j} + 1)$-th fast transition. The output message is sent after $\mathcal{A}(M_{h.j})$ executes the $(k_{h.j} + 1)$-th fast transition, and arrives before $\mathcal{A}(M_{h'.j'})$ executes its $(\kappa_{h'.j'} + 1)$-th fast transition.

6.1.1 Timeline Analysis of Machine Objects. Consider a machine object $\mathcal{A}(M_{h.j})$. At local clock time $nT_{h.j}$ (i.e., at real time t such that $c_{h.j}(t) = nT_{h.j}$), $\mathcal{A}(M_{h.j})$ begins each fast round n. By $|c_{h.j}(t) - t| < \epsilon$, each fast round n of $\mathcal{A}(M_{h.j})$ begins at some point in the global time interval:

$$(nT_{h.j} - \epsilon,\ nT_{h.j} + \epsilon)$$

At local clock time $nT_{h.j} + P_{h.j}$, $\mathcal{A}(M_{h.j})$ applies input adaptors, gets values from the input ports, executes a transition, and puts the outputs into the output ports. This process starts at some point in the global time interval $B_{h.j}(n)$, and ends at some point in the global time interval $E_{h.j}(n)$:

$$B_{h.j}(n) = (nT_{h.j} + P_{h.j} - \epsilon,\ nT_{h.j} + P_{h.j} + \epsilon),$$
$$E_{h.j}(n) = (nT_{h.j} + P_{h.j} - \epsilon + \alpha^{\min}_{h.j},\ nT_{h.j} + P_{h.j} + \epsilon + \alpha^{\max}_{h.j})$$

Consider an output port $(\bar{h}.l, q) = rep(h.j, p)$. Recall that if p is connected with $\mathfrak{E}_h$'s interface e, then $\bar{h}.l = parent_h.h$. A message is sent at local clock time $nT_{\bar{h}} + D_{\bar{h}.l}(\alpha_{h.j})$ for each slow round n (i.e., for each $\xi_{\bar{h}.l}$-th fast round), which happens sometime in the global time interval $O_{\bar{h}.l.q}(n)$:

$$O_{\bar{h}.l.q}(n) = (nT_{\bar{h}} + D_{\bar{h}.l}(\alpha^{\min}_{h.j}) - \epsilon,\ nT_{\bar{h}} + D_{\bar{h}.l}(\alpha^{\max}_{h.j}) + \epsilon)$$

Consider an input port $(\bar{h}.l, q') = rep(h.j, p')$. A message generated for slow round $n + 1$ is received sometime in $I_{\bar{h}.l.q}(n)$, after $\mathcal{A}(M_{h.j})$ has read the input for the $(\kappa_{\bar{h}.l} + 1)$-th fast round in slow round n, but before it reads the input for the $(\kappa_{\bar{h}.l} + 1)$-th fast round in slow round $n + 1$:

$$I_{\bar{h}.l.q'}(n+1) = (nT_{\bar{h}} + \kappa_{\bar{h}.l}T_{\bar{h}.l} + P_{h.j} + \epsilon,\ (n+1)T_{\bar{h}} + \kappa_{\bar{h}.l}T_{\bar{h}.l} + P_{h.j} - \epsilon)$$

$\mathcal{A}(M_{h.j})$ is synchronized with the machine objects in the same ensemble $\mathfrak{E}_h$ for each $\xi_{h.j}$-th fast round. If $\mathcal{A}(M_{h.j})$ is connected with the interface e, then $\xi_{h.j} = 1$ and $\mathcal{A}(M_{h.j})$ is also synchronized with the machine objects in the parent ensemble $\mathfrak{E}_{parent_h}$ for each $\xi_{parent_h.h}$-th fast round.

6.1.2 Input Adaptor Constraints. To ensure the same behavior as the synchronous system, MSYNC must satisfy both of Condition (4) and Condition (6) for input and output cutoff numbers. That is, consider a connection from an output port $(\bar{h}.l, q) = rep(h.j, p)$, with an output cutoff number $k_{\bar{h}.l} < \xi_{\bar{h}.l}$, to an input port $(\bar{h}.l', q') = rep(h'.j', p')$, with an input cutoff number $\kappa_{\bar{h}.l'} < \xi_{\bar{h}.l'}$. The representative input adaptor $\theta_{\bar{h}.l'}(q')$ should ignore the last $\xi_{\bar{h}.l} - k_{\bar{h}.l} - 1$ values in a $\xi_{\bar{h}.l}$-tuple input, and give default value $\perp$ for the first $\kappa_{\bar{h}.l'}$ values of a $\xi_{\bar{h}.l'}$-tuple result.

6.1.3 Real-Time Constraints. Consider a connection from an output port $(\bar{h}.l, q) = rep(h.j, p)$ to an input port $(\bar{h}.l', q') = rep(h'.j', p')$. A message sent at any time in the interval $O_{\bar{h}.l.q}(n)$ is received sometime in the global time interval $O_{\bar{h}.l.q}(n) + [\mu^{\min}_{\bar{h}}, \mu^{\max}_{\bar{h}}]$. As mentioned, this message arrival interval must be a subset of $I_{\bar{h}.l'.q'}(n+1)$. This implies that we must have:

$$D_{\bar{h}.l}(\alpha^{\min}_{h.j}) \geq 2\epsilon - \mu^{\min}_{\bar{h}} + \kappa_{\bar{h}.l'} T_{\bar{h}.l'} + P_{h'.j'}, \qquad T_{\bar{h}} \geq D_{\bar{h}.l}(\alpha^{\max}_{h.j}) + \mu^{\max}_{\bar{h}} + 2\epsilon - \kappa_{\bar{h}.l'} T_{\bar{h}.l'} - P_{h'.j'}$$

By Def. 6.1, $D_{\bar{h}.l}(\alpha_{h.j}) = \max(2\epsilon - \mu^{\min}_{\bar{h}} + \kappa_{\bar{h}.l'} T_{\bar{h}.l'} + P_{h'.j'},\ k_{\bar{h}.l} T_{\bar{h}.l} + P_{h.j} + \alpha_{h.j})$. Therefore, the first condition trivially hold. From the second condition and Def. 6.1, because $T_{\bar{h}.l} = T_{\bar{h}}/\xi_{\bar{h}.l}$ and $T_{\bar{h}.l'} = T_{\bar{h}}/\xi_{\bar{h}.l'}$ by Condition (1), we obtain the following constraints: $T_{\bar{h}} \geq \mu^{\max}_{\bar{h}} + 4\epsilon - \mu^{\min}_{\bar{h}}$, and $T_{\bar{h}}(1 - k_{\bar{h}.l}/\xi_{\bar{h}.l} + \kappa_{\bar{h}.l'}/\xi_{\bar{h}.l'}) \geq \mu^{\max}_{\bar{h}} + 2\epsilon + \alpha^{\max}_{h.j} + P_{h.j} - P_{h'.j'}$.

We also need that the `endExec` and `outputMsg` actions for one round should finish before the next round. That is, $E_{h.j}(n)$ is bounded by the global time $(n+1)T_{h.j} - \epsilon$ and $O_{\bar{h}.l.q}(n)$ is bounded by the global time $(n+1)T_{\bar{h}} - \epsilon$. These conditions imply the following constraints: $T_{h.j} \geq 2\epsilon + P_{h.j} + \alpha^{\max}_{h.j}$, and $T_{\bar{h}}(1 - \kappa_{\bar{h}.l'}/\xi_{\bar{h}.l'}) \geq 4\epsilon - \mu^{\min}_{\bar{h}} + P_{h'.j'}$. Consequently, we obtain the MSYNC period bound:

$$T_{\bar{h}} \geq \max\left(\begin{array}{r} \xi_{\bar{h}.l}(2\epsilon + P_{h.j} + \alpha^{\max}_{h.j}),\ \ \mu^{\max}_{\bar{h}} + 4\epsilon - \mu^{\min}_{\bar{h}},\ \ (4\epsilon - \mu^{\min}_{\bar{h}} + P_{h'.j'})/(1 - \kappa_{\bar{h}.l'}/\xi_{\bar{h}.l'}), \\ (\mu^{\max}_{\bar{h}} + 2\epsilon + \alpha^{\max}_{h.j} + P_{h.j} - P_{h'.j'})/(1 - k_{\bar{h}.l}/\xi_{\bar{h}.l} + \kappa_{\bar{h}.l'}/\xi_{\bar{h}.l'}) \end{array}\right) \tag{9}$$

6.2 Comparison with Multirate PALS and Multirate TTA

MSYNC extends Multirate PALS by adding offsets $P_{h.j}$ and input cutoff numbers $\kappa_{h.j}$. When $P_{h.j} = 0$ and $\kappa_{h.j} = 0$ for each component $h.j$, the definition of $D_{\bar{h}.l}(\alpha_{h.j})$ for MSYNC is the exactly same as one for Multirate PALS in Def. 5.1, and the MSYNC period bound is equivalent to the Multirate PALS period bound in Condition (5). As illustrated in Fig. 8 (and also demonstrated in Sec. 6.4), the MSYNC period bound can be strictly smaller than the Multirate PALS period bound.

Consider the single-rate case, where $\xi_{h.j} = 1$, $\kappa_{h.j} = 0$, and $k_{h.j} = 0$ for each component $h.j$. From Condition (9), we obtain the following "single-rate" MSYNC period bound:

$$L_M = \max(2\epsilon + P_{h.j} + \alpha^{\max}_{h.j},\ \mu^{\max}_{\bar{h}} + 4\epsilon - \mu^{\min}_{\bar{h}},\ 4\epsilon - \mu^{\min}_{\bar{h}} + P_{h'.j'},\ \mu^{\max}_{\bar{h}} + 2\epsilon + \alpha^{\max}_{h.j} + P_{h.j} - P_{h'.j'})$$

Recall that the single-rate PALS bound is given by $L_P = \mu^{\max}_{\bar{h}} + 2\epsilon + \max(2\epsilon - \mu^{\min}_{\bar{h}}, \alpha_{\max})$, where $\alpha_{\max} = \max_{h.j}(\alpha^{\max}_{h.j})$. We have $L_M \leq L_P$, if the following condition holds: $P_{h.j} \leq \mu^{\max}_{\bar{h}}$, $P_{h'.j'} \leq \mu^{\max}_{\bar{h}}$, and $\alpha^{\max}_{h.j} + P_{h.j} - P_{h'.j'} \leq \alpha_{\max}$.

The last condition indicates that MSYNC can achieve a *strictly* smaller bound than PALS even for the single-rate case. This means that MSYNC can also have a strictly smaller bound than single-rate TTA. For example, suppose $\alpha^{\max}_{h.j} = 5$, $\alpha^{\max}_{h'.j'} = 1$, $P_{h.j} = 0$, and $P_{h'.j'} = 2$. Then, $\alpha_{\max} = 5$, $\alpha^{\max}_{h.j} + P_{h.j} - P_{h'.j'} = 3$, and $\alpha^{\max}_{h'.j'} + P_{h'.j'} - P_{h.j} = 3$. When $2\epsilon - \mu^{\min}_{\bar{h}}$ is small and $\mu^{\max}_{\bar{h}} \geq 2$, the MSYNC period bound is $\mu^{\max}_{\bar{h}} + 2\epsilon + 3$, whereas the PALS period bound is $\mu^{\max}_{\bar{h}} + 2\epsilon + 5$.

Single-rate PALS [3, 12] assumes that: (i) all machines have the same maximum execution time, and (ii) all machines begin their rounds at the same local clock time. In this case, MSYNC and PALS have the same smallest possible period, as mentioned for the multirate case. MSYNC achieves a smaller optimal bound than PALS, when some machines can always finish its execution earlier than other machines, and the execution of such "fast" machines can be delayed.

To compare with Multirate TTA, suppose $P_{h.j} = 0$ and $k_{h.j} = 0$ for each component $h.j$. We then obtain the following bound from Condition (9): $T_{\bar{h}} \geq \max(\xi_{\bar{h}.l}(2\epsilon + \alpha^{\max}_{h.j}),\ \mu^{\max}_{\bar{h}} + 4\epsilon - \mu^{\min}_{\bar{h}},\ (4\epsilon - \mu^{\min}_{\bar{h}})/(1 - \kappa_{\bar{h}.l'}/\xi_{\bar{h}.l'}),\ (\mu^{\max}_{\bar{h}} + 2\epsilon + \alpha^{\max}_{h.j})/(1 + \kappa_{\bar{h}.l'}/\xi_{\bar{h}.l'}))$. Provided $\kappa_{\bar{h}.l'}$ is reasonably small (i.e., $\xi_{\bar{h}.l'} > 2\kappa_{\bar{h}.l'} + 1$), this bound is shorter than or equal to the multirate TTA bound (Condition (8)). As mentioned, even a smaller bound can be obtained using non-zero values for $P_{h.j}$ and $P_{h'.j'}$.

6.3 MSYNC Deployment

The important question is how to determine the period $T_{h.j}$ and offset $P_{h.j}$ of each machine $\mathcal{A}(M_{h.j})$ for MSYNC systems. As mentioned above, the lower bounds of the periods may decrease if some offsets decrease, unlike in the cases of Multirate PALS and Multirate TTA in Sec. 5. We are interested in finding *optimal* values for these periods and offsets that satisfy Condition (1) and Condition (9). It is worth noting that Condition (2) and Condition (3) follow from those conditions.

Definition 6.2. The *MSYNC deployment problem* is to determine the optimal value of the variables $\mathcal{V}_{\mathscr{E}} = \{T_h \mid h \in H\} \cup \{P_{h.j} \mid h \in H, j \in J_h \setminus H\}$ that satisfy the following constraints for $\bar{h} \in H$:

rate $T_{\bar{h}} = \xi_{\bar{h}.h} \cdot T_h$, for each subensemble $h \in J_{\bar{h}}$;
network $T_{\bar{h}} \geq \mu^{\max}_{\bar{h}} + 4\epsilon - \mu^{\min}_{\bar{h}}$;
execution $T_{\bar{h}}/\xi_{\bar{h}.j} \geq 2\epsilon + P_{\bar{h}.j} + \alpha^{\max}_{\bar{h}.j}$, for each machine $j \in J_{\bar{h}}$;
input-port $T_{\bar{h}} \cdot (1 - \kappa_{\bar{h}.l'}/\xi_{\bar{h}.l'}) \geq 4\epsilon - \mu^{\min}_{\bar{h}} + P_{h'.j'}$, for each input port $(\bar{h}.l', q') = rep(h'.j', p')$;
connection $T_{\bar{h}} \cdot (1 - k_{\bar{h}.l}/\xi_{\bar{h}.l} + \kappa_{\bar{h}.l'}/\xi_{\bar{h}.l'}) \geq \mu^{\max}_{\bar{h}} + 2\epsilon + \alpha^{\max}_{h.j} + P_{h.j} - P_{h'.j'}$, for each connection from output port $(\bar{h}.l, q) = rep(h.j, p)$ to input port $(\bar{h}.l', q') = rep(h'.j', p')$.

We are interested in minimizing the root period T_λ of the entire system $\mathscr{E}$, while maintaining each offset $P_{h.j}$ as small as possible. The periods of the subensembles and machines are then determined using the root period T_λ. When the period of each machine $T_{h.j}$ is given, the offset $P_{h.j}$ can have any value in a certain interval to satisfy the MSYNC constraints. We assign to $P_{h.j}$ the smallest possible value from such an interval, which is often 0 unless constrained by the `connection` constraints.

The MSYNC deployment problem can be solved using linear programming, since the constraints in Def. 6.2 only involve linear terms with respect to the variables $\mathcal{V}_{\mathscr{E}}$. We apply preemptive goal programming with two goals, namely, minimizing T_λ and $\sum P_{h.j}$: (1) minimize T_λ, subject to the MSYNC constraint; let $T^{\min}_\lambda$ be the resulting minimum period; and (2) minimizes $\sum P_{h.j}$, subject to the MSYNC constraint, along with the new constraint $T_\lambda = T^{\min}_\lambda$.

Our approach provides a flexible and configurable methodology for deploying asynchronous real-time systems associated with hierarchical ensembles. Thanks to our generic asynchronous real-time model presented in Sec. 4, different subensembles and machine objects can have totally different performance parameters (except for the maximum clock skew ϵ). The above algorithm then provides the smallest possible period for each subensemble and machine object.

We can reduce the smallest possible period without changing the infrastructure assumptions. Suppose that a shorter period is needed to meet the system requirements. By analyzing which constraints in Def. 6.2 are violated, we can find the root cause of the failure. If one of the `input-port` and `connection` constraints are violated, then we can modify input and output cutoff numbers (either incrementing $\kappa_{h.j}$ or decrementing $k_{h.j}$) to obtain a shorter period.

$$\Gamma_\lambda = \begin{bmatrix} \mu_\lambda^{\min} = 0.1 & \alpha_{\lambda.1}^{\max} = 2 & \kappa_{\lambda.1} = 0 & k_{\lambda.1} = 0 \\ \mu_\lambda^{\max} = 4 & \alpha_{\lambda.2}^{\max} = 0.5 & \kappa_{\lambda.2} = 0 & k_{\lambda.2} = 4 \\ & \cdot & \kappa_{\lambda.a} = 0 & k_{\lambda.a} = 1 \\ & \cdot & \kappa_{\lambda.b} = 0 & k_{\lambda.b} = 0 \end{bmatrix} \quad \Gamma_a = \begin{bmatrix} \mu_a^{\min} = 0 & \alpha_{a.1}^{\max} = 1 & \kappa_{a.1} = 0 & k_{a.1} = 0 \\ \mu_a^{\max} = 2 & \alpha_{a.2}^{\max} = 1 & \kappa_{a.2} = 0 & k_{a.2} = 0 \\ & \alpha_{a.3}^{\max} = 0.5 & \kappa_{a.3} = 0 & k_{a.3} = 1 \\ & \alpha_{a.4}^{\max} = 0.4 & \kappa_{a.4} = 0 & k_{a.4} = 2 \end{bmatrix}$$

$$\Gamma_b = \begin{bmatrix} \mu_b^{\min} = 0.3 & \alpha_{b.1}^{\max} = 2 & \kappa_{b.1} = 0 & k_{b.1} = 0 \\ \mu_b^{\max} = 3 & \cdot & \kappa_{b.c} = 0 & k_{b.c} = 2 \end{bmatrix} \quad \Gamma_c = \begin{bmatrix} \mu_c^{\min} = 0 & \alpha_{c.1}^{\max} = 1 & \kappa_{c.1} = 0 & k_{c.1} = 0 \\ \mu_c^{\max} = 1 & \alpha_{c.2}^{\max} = 0.3 & \kappa_{c.2} = 0 & k_{c.2} = 3 \end{bmatrix}$$

Fig. 10. Performance parameters of four ensembles $\mathfrak{E}_\lambda$, $\mathfrak{E}_a$, $\mathfrak{E}_b$, and $\mathfrak{E}_c$, where $\epsilon = 0.15$.

- $T_\lambda = 2T_a$, $T_\lambda = T_b$, $T_b = 3T_c$;
- $T_\lambda \geq 4.5$, $T_a \geq 2.6$, $T_b \geq 3.3$, $T_c \geq 1.6$;
- $T_\lambda \geq P_{\lambda.1} + 2.3$, $T_\lambda/5 \geq P_{\lambda.2} + 0.8$, $T_a/2 \geq P_{a.3} + 0.8$, $T_a \geq P_{a.1} + 1.3$, $T_a \geq P_{a.2} + 1.3$, $T_b \geq P_{b.1} + 2.3$, $T_c/4 \geq P_{c.2} + 0.6$; $T_a/3 \geq P_{a.4} + 0.7$, $T_c \geq P_{c.1} + 1.3$,
- $T_\lambda \geq P_{\lambda.1} + 0.5$, $T_\lambda \geq P_{\lambda.2} + 0.5$, $T_\lambda \geq P_{a.2} + 0.5$, $T_\lambda \geq P_{b.1} + 0.5$, $T_a \geq P_{a.1} + 0.6$, $T_a \geq P_{a.2} + 0.6$, $T_a \geq P_{a.3} + 0.6$, $T_a \geq P_{a.4} + 0.6$, $T_b \geq P_{b.1} + 0.3$, $T_c \geq P_{c.1} + 0.6$, $T_c \geq P_{c.2} + 0.6$;
- $T_\lambda \geq P_{\lambda.1} - P_{\lambda.2} + 6.3$, $T_\lambda * 0.2 \geq P_{\lambda.2} - P_{\lambda.1} + 4.8$, $T_a \geq P_{a.2} - P_{a.4} + 3.3$, $T_c * 0.25 \geq P_{c.2} - P_{c.1} + 1.6$, $T_\lambda \geq P_{\lambda.1} - P_{a.2} + 6.3$, $T_\lambda * 0.5 \geq P_{a.2} - P_{\lambda.2} + 5.3$, $T_b \geq P_{b.1} - P_{c.1} + 5.3$, $T_a * 0.5 \geq P_{a.3} - P_{a.4} + 2.7$, $T_a \geq P_{a.1} - P_{a.3} + 3.3$, $T_\lambda * 0.2 \geq P_{\lambda.2} - P_{b.1} + 4.8$, $T_\lambda \geq P_{b.1} - P_{\lambda.2} + 6.3$, $T_a * 1/3 \geq P_{a.4} - P_{a.3} + 2.7$, $T_c \geq P_{c.1} - P_{c.2} + 2.3$, $T_a * 1/3 \geq P_{a.4} - P_{a.1} + 2.7$, $T_a \geq P_{a.1} - P_{a.2} + 3.3$.

Fig. 11. The MSYNC constraints

6.4 An Example

We consider again the hierarchical multirate ensemble $\mathscr{E}$ in Fig. 3. Suppose that the performance parameters in Fig. 10 are imposed by the underlying infrastructure, where $\epsilon = 0.15$. There is no loss of information due to input and output cutoff numbers, because $\kappa_{h.j} = 0$ and $k_{h.j} = \xi_{h.j} - 1$ for each component $h.j$. Figure 11 shows the complete list of the MSYNC constraints by Def. 6.2 over the variables $\mathcal{V}_\mathscr{E} = \{T_\lambda, T_a, T_b, T_c, P_{\lambda.1}, P_{\lambda.2}, P_{a.1}, P_{a.2}, P_{a.3}, P_{a.4}, P_{b.1}, P_{c.1}, P_{c.2}\}$

Using linear programming as described above, we obtain the following values of the variables $\mathcal{V}_\mathscr{E}$ that minimize T_λ and then $\sum P_{h.j}$, subject to the MSYNC constraints in Fig. 11:

$$T_\lambda = 13.2, \quad P_{\lambda.1} = P_{b.1} = 2.16, \quad P_{a.1} = P_{a.3} = P_{c.1} = 0.5, \quad P_{\lambda.2} = P_{a.2} = P_{a.4} = P_{c.2} = 0.$$

If we assume $P_{c.1} = 0$ and $P_{c.2} = 0$, then the minimum value of T_λ is 19.2. If $P_{h.j} = 0$ is assumed for every machine $h.j$, the minimum value of T_λ is 24, which is 182 % greater than the optimal value. As seen, using non-zero offsets significantly reduces the smallest possible period of T_λ.

Now suppose that we need to further reduce the period, say, $T_\lambda \leq 12$, due to the system requirements. In this case, the MSYNC constraints are infeasible because the `connection` constraint $T_a * 1/3 \geq P_{a.4} - P_{a.3} + 2.7$ cannot be satisfied. To deal with this problem, we can either increment $\kappa_{a.3}$ (the receiver) or decrement $k_{a.4}$ (the sender). For both cases, the minimum value of T_λ by the new MSYNC constraints is about 9.647, which satisfies the requirement $T_\lambda \leq 12$.

7 CORRECTNESS OF THE MSYNC SYNCHRONIZER

Given a hierarchical ensemble $\mathscr{E}$, system parameters $\mathscr{B}$, a bound $\epsilon > 0$, and input/output cutoff numbers $\{\kappa_{h.j}, k_{h.j}\}_{h \in H, j \in J_h}$, MSYNC defines the asynchronous real-time system $\mathscr{A}(\mathscr{E})$, provided that the MSYNC constraints are satisfied. This section formalizes the precise relationship between

the synchronous composition $M_{\mathscr{E}}$ and the asynchronous real-time system $\mathcal{A}(\mathscr{E})$. Specifically, $M_{\mathscr{E}}$ and $\mathcal{A}(\mathscr{E})$ are stuttering bisimilar with respect to "adaptor-equivalent" inputs and outputs.

Global Snapshots. We follow the approach in [6, 12] to identify global snapshots of the system $\mathcal{A}(\mathscr{E})$, at times $nT - \epsilon$, just before each component starts a new round. In principle, these snapshot states correspond to states of the synchronous composition $M_{\mathscr{E}}$. In PALS, such global states are "stable" in the sense that there are no messages in transit in those states [6, 12], because all messages arrive at their destination before the beginning of a new round.

In MSYNC asynchronous systems, however, such stable snapshots do not always exist. In MSYNC, a message sent to an input port $(\bar{h}.l, q') = rep(h.j, p')$ in slow round n may arrive after the beginning of slow round $n+1$ but before the beginning of the $(\kappa_{\bar{h}.l} + 1)$-th fast round of slow round $n+1$; as explained in Sec. 6.1.1, the message arrives at some point in global time interval $I_{\bar{h}.l.q}(n+1)$. Therefore, we consider "MSYNC-snapshot" states at times $nT - \epsilon$ for each subensemble of $\mathscr{E}$.

Definition 7.1. A global state $\{C; t\}$ is called an *MSYNC snapshot* with respect to an ensemble $\mathfrak{E}_{\bar{h}}$, written $\{C; t\} \in \mathcal{SS}_{\bar{h}}(\mathcal{A}(\mathscr{E}))$, if there exists $n \in \mathbb{N}$ such that $t = nT_{\bar{h}} - \epsilon$ and:

- for each input port $(\bar{h}.l, q') = rep(h.j, p')$, either (i) the buffer of the input port is full, or (ii) there exists a message in C destined for the input port with arrival time in $I_{\bar{h}.l.q}(n)$;
- for each output port $(\bar{h}.l, q) = rep(h.j, p)$, the buffer of the output port is empty; and
- there is no other message in C with the context ensemble $\bar{h}$.

Consider an MSYNC snapshot $\{C; t\} \in \mathcal{SS}_{\bar{h}}(\mathcal{A}(\mathscr{E}))$. Each machine $\mathcal{A}(M_{h.j})$ is in slow round $n-1$, because $\mathcal{A}(M_{h.j})$ starts slow round n at local clock time $nT_{\bar{h}}$. Each output buffer of $\mathcal{A}(M_{h.j})$ is empty, since a message must be sent before the next slow round (Condition (3)). Each input port $(\bar{h}.l, q') = rep(h.j, p')$ must receive a message sometime in global time interval $I_{\bar{h}.l.q}(n)$; either the message is in transit in the configuration C or has already arrived before t.

A "big-step" transition $\{C; t\} \longrightarrow_{\mathcal{SS}} \{C'; t'\}$ is defined between two MSYNC-snapshot states with respect to $\mathfrak{E}_{\bar{h}}$ iff there exists an execution sequence $\{C; t\} \longrightarrow_{\mathcal{A}(\mathscr{E})} \cdots \longrightarrow_{\mathcal{A}(\mathscr{E})} \{C', t'\}$ in $\mathcal{A}(\mathscr{E})$ such that $\{C; t\}, \{C'; t'\} \in \mathcal{SS}_{\bar{h}}(\mathcal{A}(\mathscr{E}))$ and $t' = T + T_{\bar{h}}$. Notice that $\longrightarrow_{\mathcal{SS}}$ captures one slow round of the ensemble $\mathfrak{E}_{\bar{h}}$ in $\mathcal{A}(\mathscr{E})$, and therefore $\longrightarrow_{\mathcal{SS}}$ is a total transition relation on $\mathcal{SS}_{\bar{h}}(\mathcal{A}(\mathscr{E}))$. Let $(\mathcal{SS}_{\bar{h}}(\mathcal{A}(\mathscr{E})), \longrightarrow_{\mathcal{SS}})$ be the resulting "big-step" transition system.

Following [6, 12], initial states of $\mathcal{A}(\mathscr{E})$ are defined as MSYNC-snapshot states in $\mathcal{SS}_{\lambda}(\mathcal{A}(\mathscr{E}))$ at *global time* $T_{\lambda} - \epsilon$ *with respect to the root ensemble* $\mathfrak{E}_{\lambda}$ in the hierarchical ensemble $\mathscr{E}$. Every machine object in $\mathcal{A}(\mathscr{E})$ is just about to start a new round at global time $T_{\lambda} - \epsilon$, because T_{λ} is a multiple of the period of each subensemble and machine object in $\mathscr{E}$. Notice that initial MSYNC-snapshot states also include "stable" initial states with all input buffers full.

Substituting Leaf Ensembles. Suppose that $\{C; t\}$ is an MSYNC snapshot with respect to a *leaf* ensemble $\mathfrak{E}_{\eta}$. In this case, the *flat sub-configuration* C_{η} of C only includes the machine objects of $\mathfrak{E}_{\eta}$ and the messages with $\mathfrak{E}_{\eta}$ as its context ensemble. Recall that a state of $M_{SR(\mathfrak{E}_{\eta})}$ is a collection of the state of each machine in $\mathfrak{E}_{\eta}$ and the values in the feedback wires in $\mathfrak{E}_{\eta}$ (Def. 2.3). There is a function $sync_{\eta}$ mapping flat subconfigurations C_{η} to states of $M_{SR(\mathfrak{E}_{\eta})}$ in a similar way to [6].

Definition 7.2. For a flat sub-configuration C_{η} of $\{C; t\} \in \mathcal{SS}_{\eta}(\mathcal{A}(\mathscr{E}))$, $sync_{\eta}(C_{\eta})$ is the tuple $(s_j, \vec{v_j})_{j \in J_{\eta}}$, where: (i) the state of $\mathcal{A}(M_{\eta.j})$ in C_{η} is s_j, and (ii) the messages in C_{η}, either in the input buffer of $\mathcal{A}(M_{\eta.j})$ or being sent to $\mathcal{A}(M_{\eta.j})$, give the values of the feedback wires $\vec{v_j}$.

Consider a hierarchical ensemble $\mathscr{E} = (H, \lambda, \{parent_h\}_{h \in H \setminus \{\lambda\}}, \{\mathfrak{E}_h\}_{h \in H})$. For a leaf ensemble index $\eta \in H$, let $\mathscr{E}|_{H \setminus \{\eta\}}$ be the hierarchical ensemble obtained by replacing the entire subensemble

$\mathfrak{E}_\eta$ with its synchronous composition $M_{SR(\mathfrak{E}_\eta)}$.[6] Then, $\mathcal{A}(\mathscr{E}|_{H\setminus\{\eta\}})$ is the asynchronous system that includes $\mathcal{A}(M_{SR(\mathfrak{E}_\eta)})$ instead of the machine objects in $\mathfrak{E}_\eta$. The $sync_\eta$ function relates a flat sub-configuration of $\{C;t\} \in \mathcal{SS}_\eta(\mathcal{A}(\mathscr{E}))$ to a state of the machine object $\mathcal{A}(M_{SR(\mathfrak{E}_\eta)})$.

The replacement object $\mathcal{A}(M_{SR(\mathfrak{E}_\eta)})$ has the same input/output timings as the corresponding objects of $\mathfrak{E}_\eta$ in $\mathcal{A}(\mathscr{E})$. Each output port $(\eta.l,q)$ of $\mathcal{A}(M_{SR(\mathfrak{E}_\eta)})$ corresponds to some output port $(h.j,p)$ of $\mathcal{A}(M_{h.j})$, where $(\eta.l,q) = rep(h.j,p)$. For each round n, a message is sent from each output port $(\eta.l,q)$ of $\mathcal{A}(M_{SR(\mathfrak{E}_\eta)})$ sometime in global time interval $O_{\eta.l.q}(n)$. Similarly, a message is received by each input port $(\eta.l,q)$ of $\mathcal{A}(M_{SR(\mathfrak{E}_\eta)})$ sometime in global time interval $I_{\eta.l.q}(n)$.

However, the systems $\mathcal{A}(\mathscr{E})$ and $\mathcal{A}(\mathscr{E}|_{H\setminus\{\eta\}})$ do not have "exactly" the same behaviors in general. Specifically, a machine object $\mathcal{A}(M_{\eta.j})$ in $\mathcal{A}(\mathscr{E})$ may receive an input message after executing $\kappa_{\eta.j} \le \xi_{\eta.j}$ transitions, and send an output message after executing only $k_{\eta.j}+1 \le \xi_{\eta.j}$ transitions, whereas the machine $M_{\eta.j}$ receives a complete input and sends a complete output in $\mathcal{A}(M_{SR(\mathfrak{E}_\eta)})$. In MSYNC, the input adaptor constraints (Sec. 6.1.2) are used to ensure the equivalence.

Consider two states $(s_j,\vec{v}_j)_{j\in J_h}$ and $(s_j,\vec{w}_j)_{j\in J_h}$ of the synchronous composition $M_{SR(\mathfrak{E}_h)}$ with feedback wire values $\vec{v}_j$ and $\vec{w}_j$. By the input adaptor constraint, for a value $\vec{v}$, input adaptor θ with input cutoff number κ and output cutoff number k only depends on the first $k+1$ items of $\vec{v}$, and the first κ items of the result $\theta(\vec{v})$ is $\perp$. Following [6], we define an *adaptor-equivalence* $\equiv_A$ such that $(s_j,\vec{v}_j)_{j\in J_h} \equiv_A (s_j,\vec{w}_j)_{j\in J_h}$ iff no input adaptor in $\mathfrak{E}_h$ can distinguish $\vec{v}_j$ and $\vec{w}_j$. In this case, $(s_j,\vec{v}_j)_{j\in J_h}$ and $(s_j,\vec{w}_j)_{j\in J_h}$ result in exactly the same next states of $M_{SR(\mathfrak{E}_h)}$.

Using the function $sync_h$ and the adaptor-equivalence $\equiv_A$, we can relate an MSYNC-snapshot state of $\mathcal{A}(\mathscr{E})$ and a global state of the substituted system $\mathcal{A}(\mathscr{E}|_{H\setminus\{\eta\}})$ as follows.

Definition 7.3. For a leaf ensemble $\mathfrak{E}_\eta$, let $\{D\cup C_\eta;t\} \sim_\eta \{D\cup O_\eta;t\}$ iff $sync_\eta(C_\eta) \equiv_A O_\eta$, where $\{D\cup C_\eta;t\} \in \mathcal{SS}_\eta(\mathcal{A}(\mathscr{E}))$, $\{D\cup O_\eta;t\}$ is a global state of $\mathcal{A}(\mathscr{E}|_{H\setminus\{\eta\}})$, C_η is a flat sub-configuration for $\mathfrak{E}_\eta$, and O_η is a single machine object $\mathcal{A}(M_{SR(\mathfrak{E}_\eta)})$.

Relating Two Systems $\mathcal{A}(\mathscr{E})$ and $\mathcal{A}(\mathscr{E}|_{H\setminus\{\eta\}})$. We state that the relation $\sim_\eta$ is a bisimulation between MSYNC-snapshot states of $\mathcal{A}(\mathscr{E})$ and MSYNC-snapshot states of $\mathcal{A}(\mathscr{E}|_{H\setminus\{\eta\}})$ with respect to the parent $\mathfrak{E}_{parent_\eta}$ of the leaf ensemble $\mathfrak{E}_\eta$ in the following lemma. This equivalence easily follows from the input-adaptor constraints (Sec. 6.1.2) and the real-time constraints (Sec. 6.1.3).

Lemma 7.4. *$\sim_\eta \subseteq \mathcal{SS}_{parent_\eta}(\mathcal{A}(\mathscr{E})) \times \mathcal{SS}_{parent_\eta}(\mathcal{A}(\mathscr{E}|_{H\setminus\{\eta\}}))$ is a bisimulation relation between the transition systems $(\mathcal{SS}_{parent_\eta}(\mathcal{A}(\mathscr{E})), \longrightarrow_{\mathcal{SS}})$ and $(\mathcal{SS}_{parent_\eta}(\mathcal{A}(\mathscr{E}|_{H\setminus\{\eta\}})), \longrightarrow_{\mathcal{SS}})$.*

Proof Sketch. Consider a big-step transition $\{C;t\} \longrightarrow_{\mathcal{SS}} \{C';t'\}$ for two MSYNC-snapshots $\{C;t\}, \{C';t'\} \in \mathcal{SS}_{parent_\eta}(\mathcal{A}(\mathscr{E}))$, where $t = nT_{parent_\eta} - \epsilon$ and $t' = (n+1)T_{parent_\eta} - \epsilon$. By Condition (1), $t = n(\xi_{parent_\eta.\eta}T_\eta) - \epsilon$ and $t' = (n+1)(\xi_{parent_\eta.\eta}T_\eta) - \epsilon$. Therefore, each machine object $\mathcal{A}(M_{\eta.j})$ of $\mathfrak{E}_\eta$ is just before starting slow round $n\xi_{parent_\eta.\eta}$ in C, and just before starting slow round $(n+1)\xi_{parent_\eta.\eta}$ in C'. In between $\{C;t\}$ and $\{C';t'\}$, $\mathcal{A}(M_{\eta.j})$ runs $\xi_{parent_\eta.\eta}\xi_{\eta.j}$ fast rounds.

Since $\{C;t\}$ is an MSYNC-snapshot, each input port $(\bar{h}.l,q') = rep(\eta.j,p')$ of $\mathfrak{E}_\eta$ has a message $\vec{v}_{\bar{h}.l.q'}$ in C to execute a next transition either in the buffer or to arrive in the corresponding time interval by $I_{\bar{h}.l.q'}$. When running $\{C;t\} \longrightarrow_{\mathcal{SS}} \{C';t'\}$, a message is sent from each output port $(\bar{h}.l,q) = rep(\eta.j,p)$ sometime in the corresponding time interval by $O_{\bar{h}.l.q}$. By the MSYNC real-time constraints, all messages are sent and received in a round-consistent way.

Suppose $\{C;t\} \sim_\eta \{D;t\}$ for $\{D;t\} \in \mathcal{SS}_{parent_\eta}(\mathcal{A}(\mathscr{E}|_{H\setminus\{\eta\}}))$. Each input port $(\bar{h}.l,q')$ of $\mathcal{A}(M_{SR(\mathfrak{E}_\eta)})$, with $\bar{h}.l = parent_\eta.\eta$, has a message $\vec{w}_{\bar{h}.l.q'}$ in D to (already) be arrived in the interval by $I_{\bar{h}.l.q'}$. Because $\{C;t\} \sim_\eta \{D;t\}$, $\vec{w}_{\bar{h}.l.q'}$ is not distinguishable with $\vec{v}_{\bar{h}.l.q'}$ (the corresponding input

[6] More precisely, according to Def. 3.4, it is just $\mathscr{E}|_{H\setminus\{\eta\}} = (H\setminus\{\eta\}, \lambda, \{parent_h\}_{h\in H\setminus\{\lambda,\eta\}}, \{\mathfrak{E}_h\}_{h\in H\setminus\{\eta\}})$.

in C) by the input adaptor $\theta_{\tilde{h}.l,q'}$. $\mathcal{A}(M_{SR(\mathfrak{E}_\eta)})$ can be executed in an "adaptor-equivalent" way as the machine objects of $\mathfrak{E}_\eta$ in $\mathcal{A}(\mathscr{E})$. As a consequence, there exists $\{D'; t'\} \in \mathcal{SS}_{parent_\eta}(\mathcal{A}(\mathscr{E}|_{H\setminus\{\eta\}}))$ such that $\{D; t\} \longrightarrow_{\mathcal{SS}} \{D'; t'\}$ and $\{C'; t'\} \sim_\eta \{D'; t'\}$. The other direction is similar. □

Using Lemma 7.4, we can inductively build a bisimulation between $(\mathcal{SS}_\lambda(\mathcal{A}(\mathscr{E})), \longrightarrow_{\mathcal{SS}})$ and $(\mathcal{SS}_\lambda(\mathcal{A}(M_{\mathscr{E}})), \longrightarrow_{\mathcal{SS}})$, where, by definition, $M_{\mathscr{E}}$ is $M_{SR(\mathfrak{E}_\lambda)}$ for the root ensemble $\mathfrak{E}_\lambda$. For each step, the machine objects of a leaf ensemble $\mathfrak{E}_\eta$ are substituted by the single machine object $\mathcal{A}(M_{SR(\mathfrak{E}_\eta)})$, and we finally obtain the single object $\mathcal{A}(M_{SR(\mathfrak{E}_\lambda)})$. For a *closed* machine $M = (\cdot, \cdot, S, \delta_M)$ with no input and output ports, let $tr(M) = (S, \longrightarrow_M)$ be the transition system such that $s \longrightarrow_M s'$ iff $((\cdot, s), (s, \cdot)) \in \delta_M$. The following theorem then follows from the obvious bisimulation between $tr(M_{\mathscr{E}})$ and $(\mathcal{SS}_\lambda(\mathcal{A}(M_{\mathscr{E}})), \longrightarrow_{\mathcal{SS}})$ and the fact that bisimulations compose.

Theorem 7.5. *For a closed hierarchical multirate ensemble $\mathscr{E}$, there exists a bisimulation between the two transition systems $(\mathcal{SS}_\lambda(\mathcal{A}(\mathscr{E})), \longrightarrow_{\mathcal{SS}})$ and $tr(M_{\mathscr{E}})$.*

8 CONCLUDING REMARKS

We have introduced the MSYNC formal design pattern as a synchronizer for multirate virtually synchronous CPSs when the underlying infrastructure guarantees bounds on network delays, clock skews, and execution times. MSYNC generalizes and combines the advantages of TTA and Multirate PALS, by allowing us to declare local worst-case execution and communication times, and by allowing us to define when to perform certain actions. We define the synchronous and asynchronous models of MSYNC in the fully general hierarchical multirate setting, and prove a precise correspondence between them. We also explain how linear programming can be used to solve constraints to arrive at optimal deployments.

As part of our work, we also define for the first time a multirate extension of TTA, and show that the optimal period of MSYNC is shorter than those of both TTA and PALS.

In future work, we should investigate the effectiveness of MSYNC on state-of-the-art applications, and provide modeling and verification support for MSYNC in engineer-friendly modeling tools.

REFERENCES

[1] Jean-Raymond Abrial, Egon Börger, and Hans Langmaack (Eds.). 1996. *Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control.* LNCS, Vol. 1165. Springer.

[2] Abdullah Al-Nayeem, Lui Sha, Darren D Cofer, and Steven M Miller. 2012. Pattern-based composition and analysis of virtually synchronized real-time distributed systems. In *ICCPS*. IEEE, 65–74.

[3] Abdullah Al-Nayeem, Mu Sun, Xiaokang Qiu, Lui Sha, Steven P. Miller, and Darren D. Cofer. 2009. A Formal Architecture Pattern for Real-Time Distributed Systems. In *RTSS*. IEEE, 161–170.

[4] David Arney, Raoul Jetley, Paul Jones, Insup Lee, and Oleg Sokolsky. 2007. Formal methods based development of a PCA infusion pump reference model: Generic infusion pump (GIP) project. In *HCMDSS-MDPnP*. IEEE, 23–33.

[5] Kyungmin Bae, Joshua Krisiloff, José Meseguer, and Peter Csaba Ölveczky. 2015. Designing and verifying distributed cyber-physical systems using Multirate PALS: An airplane turning control system case study. *Science of Computer Programming* 103 (2015), 13–50.

[6] Kyungmin Bae, José Meseguer, and Peter Csaba Ölveczky. 2014. Formal patterns for multirate distributed real-time systems. *Science of Computer Programming* 91 (2014), 3–44.

[7] Guillaume Baudart, Albert Benveniste, and Timothy Bourke. 2016. Loosely time-triggered architectures: Improvements and comparisons. *ACM Transactions on Embedded Computing Systems (TECS)* 15, 4 (2016), 1–26.

[8] Cheolgi Kim, Mu Sun, Sibin Mohan, Heechul Yun, Lui Sha, and Tarek F Abdelzaher. 2010. A framework for the safe interoperability of medical devices in the presence of network failures. In *ICCPS*. 149–158.

[9] Hermann Kopetz and Günther Bauer. 2003. The time-triggered architecture. *Proc. IEEE* 91, 1 (2003), 112–126.

[10] Leslie Lamport and P Michael Melliar-Smith. 1985. Synchronizing clocks in the presence of faults. *Journal of the ACM (JACM)* 32, 1 (1985), 52–78.

[11] Gabriel Leen, Donal Heffernan, and Alan Dunne. 1999. Digital networks in the automotive vehicle. *Computing & Control Engineering Journal* 10, 6 (1999), 257–266.

[12] José Meseguer and Peter Csaba Ölveczky. 2012. Formalization and correctness of the PALS architectural pattern for distributed real-time systems. *Theoretical Computer Science* 451 (2012), 1–37.
[13] Steven P Miller, Darren D Cofer, Lui Sha, Jose Meseguer, and Abdullah Al-Nayeem. 2009. Implementing logical synchrony in integrated modular avionics. In *2009 IEEE/AIAA 28th Digital Avionics Systems Conference*. IEEE, 1–A.
[14] Roman Obermaisser. 2018. *Time-triggered communication*. CRC Press.
[15] Lee Pike. 2006. A note on inconsistent axioms in Rushby's" systematic formal verification for fault-tolerant time-triggered algorithms". *IEEE Transactions on Software Engineering* 32, 5 (2006), 347–348.
[16] John Rushby. 1999. Systematic formal verification for fault-tolerant time-triggered algorithms. *IEEE Transactions on Software Engineering* 25, 5 (1999), 651–660.
[17] Wilfried Steiner, Günther Bauer, Brendan Hall, Michael Paulitsch, and Srivatsan Varadarajan. 2009. TTEthernet dataflow concept. In *2009 Eighth IEEE International Symposium on Network Computing and Applications*. IEEE, 319–322.
[18] Wilfried Steiner and John Rushby. 2011. TTA and PALS: Formally verified design patterns for distributed cyber-physical systems. In *2011 IEEE/AIAA 30th Digital Avionics Systems Conference*. IEEE, 7B5–1.