A Distributed Computing Model for Dataflow, Controlflow, and Workflow in Fractionated Cyber-Physical Systems

Mark-Oliver Stehr, Minyoung Kim, and Tim McCarthy

SRI International,
firstname.lastname@sri.com

Abstract. With the ongoing trend to parallelize computations for scalability, better performance, and reliability, distributed dataflow models are attracting interest at all design levels, ranging from processor architectures to local- and wide-area computing clusters in the cloud. Data-driven computation has also been an important paradigm in sensor networks and embedded systems, which have evolved into a larger research effort on networked cyber-physical systems (NCPS), that can sense and affect their environment. Fractionated cyber-physical systems (FCPS) are an interesting subclass of NCPS where the redundancy and diversity of many unreliable and potentially heterogeneous networked components is exploited to improve scalability, reliability, and verifiability of the overall system. In this paper we present the theory and an application of a new distributed computing model for such systems as a first step toward a model-based design methodology for FCPS. To uniformly capture dataflow, controlflow, and workflow, we use a subclass of Petri nets as an intuitive high-level model, which is translated into a weaker model — namely, a new variant of Petri nets that does not make any atomicity assumptions but instead uses a partial order to ensure eventual consistency. By combining our theory with the concept of virtual potential functions, we briefly discuss an application to unmanned aerial vehicle (UAV) swarms, which has been implemented on top of a prototype of our theory for both simulation models and real world deployments.

Dedication

The first author is dedicating this paper to Jozef Gruska, who in his role as visiting professor was his early advisor during his studies at the University of Hamburg. One topic was Carl Adam Petri's theory of concurrency, which has influcenced this paper to a great degree, although in an unexpected direction. This paper explores a world in which atomic non-deterministic choices do not exist or are too expensive to implement with high reliability. The motivating application domain are loosely coupled distributed systems, such as wireless networks of mobile cyber-physical devices, which can be regarded as the new computing resources of our time. Interestingly, observable non-determinism is also a challenge in quantum computing, a potential computing resource of the future, due to the associated and often undesirable collapse of the quantum state. Hence, potential applications of our model in this domain would be worthwhile to explore, but they are beyond the scope of this paper.

1 Introduction

Dataflow models have a long history in computer science as witnessed by the large body of literature on dataflow graphs, Kahn networks, stream processing languages, and Petri nets. Dataflow models come in many flavors. Some use graphical representations, while others are presented as formal systems, such as linear logic [16] or rewriting logic [30]. Such models have been used not only to describe software and hardware designs but also workflow in various domains from manufacturing to government, where the individual tasks of a workflow are not computational but may entail interaction with the physical world or the social/organizational context. Dataflow models make explicit the information flow and hence the causal structure of computations, which is only implicitly represented in traditional algorithms. Explicitly representing the structure of a distributed computation is a form of partial reflection that enables the runtime system to make better decisions on how to map computation to the available resources. Another attractive feature of dataflow/workflow models is that they are declarative, in the sense that data objects, functions, and causal dependencies can be specified, e.g., in the form of a Petri net, and at the same time they can be operational, i.e., executable.

Distributed dataflow approaches are gaining popularity, most recently because of their potential to exploit the parallelism offered by modern many-core architectures (stream processing) and cloud computing. Google's MapReduce framework [9] can be seen as an execution engine for distributed dataflow graphs of a certain two-level shape. Programming languages for the distributed execution of larger classes of dataflow graphs are being developed, as for instance CIEL [34]. Stream processing frameworks such as OpenCL [1] can express data-parallel programs in terms of kernel functions that can be efficiently mapped to many-core architectures such as modern graphical processing units (GPUs). Thanks to their capability to separate timing and functionality, dataflow models enable scalability and are also becoming increasingly popular in hardware design as witnessed by approaches such as BlueSpec [2], which is influenced by Petri nets and rewriting-based approaches, and is becoming a real competitor for VHSIC hardware description language (VHDL) and Verilog. Data-oriented approaches have also traditionally been used in sensor networks with in-network processing (mostly asynchronous) and model-based design of embedded systems (mostly synchronous), which have become part of a larger research effort on sensor/actor networks [12] or networked cyber-physical systems (NCPS). There is also a parallel trend to make general-purpose networking architectures more data oriented and aware of the semantics of the data, as witnessed by recent efforts such as disruption-tolerant networking [13], declarative networking [29], and content-based networking [24, 18].

The real potential of distributed dataflow models for programming or model-based design has not been reached yet, because existing dataflow engines are implemented as classical distributed algorithms that can operate successfully only in very constrained and cooperative environments. Although some traditional fault tolerance can be offered by dataflow engines, operating under high failure

rates, with highly unreliable connectivity, or even in the context of node mobility is still a major challenge arising in many real-world large-scale distributed systems or systems operating in hostile/uncontrollable environments. The objective of this paper is to develop a dataflow model for fractionated software and hardware systems. Such systems are motivated by the strong trend toward a truly distributed world in which potentially unreliable and resource-constrained devices and powerful computing resources coexist without assuming continuous network connectivity, and where a large number of such components need to be operating as an ensemble to produce sufficient performance and reliability.

In Section 2, we introduce some background on fractionated cyber-physical systems (FCPS), the partially ordered knowledge-sharing model, which sets the broader framework for this work, and basic terminology on Petri nets, which are widely used in model-based design and in this paper will be used to specify dataflow, controlflow, and workflow for fractionated systems. In Section 3, we define the high-level workflow model and its mapping to a weaker model that is amenable to a fully distributed fractionated implementation. Most important, the close correspondence between these models is established. In Section 4, we use a specific case study to illustrate how our theory can be combined with the concept of virtual potential functions and applied for the distributed control of fractionated systems, in this case UAV swarms on a surveillance mission. In Section 5, we discuss some related work, before we conclude in Section 6.

2 Background

2.1 Fractionated Cyber-Physical Systems

NCPS are distributed systems consisting of heterogeneous software and hardware components that can sense and affect the environment. Typically, the environment plays an important role and can have many dimensions. For instance, manufacturing systems, process control systems, automobiles, aircraft, satellites, and robots are classical examples of NCPS, but also systems that support social interaction or collaborative activities with humans in the loop. The notion of fractionated software is inspired by hardware fractionation in DARPA's F6[1] space program [5] and has been proposed in [49] as a new basis to improve reliability and verifiability of systems and for future systems built on fundamentally unreliable components. Fractionated cyber-physical systems (FCPS) can be seen as a class of NCPS in which the operation of the system is fully distributed without tying a function to a particular piece of hardware, which can also be regarded as an extreme form of virtualization. Hardware is highly redundant so that functions can be performed even in the presence of continuous failures without the need for global coordination. FCPS should operate in a meaningful way even if network communication is extremely unreliable and network partitioning may occur. FCPS also need to provide scalability in the sense that the amount

[1] Future, Fast, Flexible, Formation-Flying, Fractionated Spacecraft united by Information eXchange

of hardware/software elements (so-called fracments) can be adjusted, without requiring changes in the application. Many classical distributed algorithms that may be sufficient for NCPS are not suitable for FCPS, because they make strong assumptions about the network — e.g., have bounded delays, use nonscalable primitives such as transactions, or introduce temporary bottlenecks, e.g., synchronization or leader election phases. Aspects of FCPS can be found in sensor networks, ensembles or swarms of vehicles, or biological systems, more generally systems of a large number of unreliable components that are beyond the reach of classical distributed algorithms.

2.2 Partially Ordered Knowledge Sharing

The partially ordered knowledge sharing model for loosely coupled distributed computing was introduced in [21], which presents a prototypical implementation of what we call the *cyber-application framework*, a software framework for NCPS. A logical version of this model was studied in [46, 22] and a particular instance was used for disruption-tolerant networking [48].

In a nutshell, this model postulates an NCPS with a finite set of so-called *cyber-nodes* that provide *computing* resources, can have volatile and/or persistent *storage*, and are all equipped with *networking* capabilities. Cyber-nodes can have additional devices such as *sensors* and *actuators*, through which they can observe and control their environment, but only to a limited degree (including possibly their own physical state, e.g., their orientation/position). Cyber-nodes can be fixed or mobile, and for the general model no assumption is made about the computing or storage resources or about the network and the communication capabilities or opportunities that it provides. Hence, this model covers a broad range of *heterogeneous* technologies (e.g., wireless/wired, unicast/broadcast) and potentially challenging environment conditions, where networking characteristics can range from high-quality persistent connectivity to intermittent/episodic connectivity. The cyber-physical system is *open* in the sense that new nodes can join and leave the network at any time. Permanent or temporary communication or node *failures* are admitted by this model.

As a step toward semantic networking [36], partially ordered knowledge sharing allows the network to be aware of the application semantics (in this paper, the semantics of our workflow model), and hence a real-world distributed implementation of our model has a well-defined behavior under network partitioning and network merging, and can tolerate unbounded network delays and node mobility. Due to the transactionless nature of our model, disconnections do not have to be announced or predicted as in distributed tuple space models such as [33]. Thanks to in-network caching, delay- and disruption-tolerant communication scenarios — e.g., highly unreliable wireless networks, short windows of opportunity, and message ferrying — can be supported. There is no need for continuous clock synchronization, as long as logical time [27] is respected.

Applications are event-driven; that is, they can post and respond to local events. For coordination, knowledge can be posted into the network by a *semantic broadcast* that can be implemented by many different protocols, ranging

from physical broadcast to gossip-style protocols. Knowledge can also be cached in the network by intermediate nodes and at the endpoints. To partially capture the semantics of knowledge for the purpose of distributed knowledge sharing, we assume an application-specific strict partial order $\prec$ on units of knowledge that we refer to as *replacement order*, with the intuition that $k \prec k'$ means that k' replaces/overwrites k, and hence if k has not been delivered yet to the application, the knowledge-sharing model may discard it without delivering it, if k' has already been received.

2.3 Petri Nets

Petri nets are the prototypical model for dataflow that has inspired many other formalisms. Here, we use nets with individual tokens, specifically colored nets, as a common specification of dataflow, controlflow, and workflow. We begin with some basic notation and Petri net terminology.

A *finite multiset* over a set S is a function m from S to $\mathbb{N}$ such that its support $\mathcal{S}(m) = \{s \in S \mid m(s) > 0\}$ is finite. We denote by $\mathcal{MS}(S)$ the set of finite multisets over S. We write $\emptyset_S$ for the empty multiset over S and $\langle e_1, \ldots, e_n \rangle_S$ for the multiset containing the listed elements (we usually omit S if it is clear from the context), and, overloading some set operators, we use the standard definitions of multiset membership $\in$, inclusion $\subseteq$, union $+$, and difference $-$.

A *net* $\mathcal{N}$ consists of a finite set of *places* $P_\mathcal{N}$, a finite set of *transitions* $T_\mathcal{N}$ disjoint from $P_\mathcal{N}$, and a *flow relation* $F_\mathcal{N} \subseteq (P_\mathcal{N} \times T_\mathcal{N}) \cup (T_\mathcal{N} \times P_\mathcal{N})$ such that the *preset* ${}^\bullet t$ and the *postset* $t^\bullet$ are nonempty for each $t \in T_\mathcal{N}$, where the *preset* ${}^\bullet x$ and *postset* $x^\bullet$ of $x \in P \cup T$ are defined as $\{y \mid y \; F_\mathcal{N} \; x\}$ and $\{y \mid x \; F_\mathcal{N} \; y\}$, respectively. A net is *pure* iff $x^\bullet \cap {}^\bullet x = \emptyset$ for all $x \in P \cup T$.

Whenever we use nonpure nets, we interpret symmetric pairs of the flow relation to represent what is usually called test or read arcs, rather then self-loops, which have a different concurrency and refinement semantics.

Let $\mathcal{N}$ be a net. A *(black) token* is simply defined by a place $p \in P_\mathcal{N}$. A *marking* is a multiset of tokens. An *occurrence* is defined by a transition $t \in T_\mathcal{N}$. The sets of *tokens*, *markings*, and *occurrences* of $\mathcal{N}$ are denoted by $\mathcal{K}_\mathcal{N}$, $\mathcal{M}_\mathcal{N}$ and $\mathcal{O}_\mathcal{N}$, respectively. The *semantics* of a net $\mathcal{N}$ is given by the labeled transition system that has $\mathcal{M}_\mathcal{N}$ as its set of states, $\mathcal{O}_\mathcal{N}$ as its set of labels, and a transition relation $\longrightarrow$ given by the *occurrence transition relation* $\longrightarrow_o \subseteq \mathcal{M}_\mathcal{N} \times \mathcal{O}_\mathcal{N} \times \mathcal{M}_\mathcal{N}$ defined by $m_1 \xrightarrow{t}_o m_2$ iff there is a marking m such that $m_1 = m + {}^\bullet t$ and $m_2 = m + t^\bullet$. Writing the occurrence rule in the way given above makes it evident that the occurrence of a transition t replaces its preset by its postset, whereas the remainder of the marking, here denoted by m, is not involved in this process. We say that a transition t is *enabled* at a marking m iff $m \xrightarrow{t}_o m'$ for some marking m'. We also define a *binary transition relation* $\longrightarrow \subseteq \mathcal{M}_\mathcal{N} \times \mathcal{M}_\mathcal{N}$ by $m_1 \longrightarrow m_2$ iff $m_1 \xrightarrow{l}_o m_2$ for some label l, its transitive closure $\xrightarrow{+}$, and its reflexive and transitive closure $\xrightarrow{*}$. We say that m_2 is *reachable from* m_1 iff $m_1 \xrightarrow{*} m_2$. We write $\mathcal{R}(\mathcal{N}, m)$ to denote the set of markings reachable from m.

An *execution* of a net $\mathcal{N}$ is a finite sequence $\pi = m_0, l_0, m_1, l_1, \ldots, l_n, m_{n+1}$, or an infinite sequence $\pi = m_0, l_0, m_1, l_1, \ldots$ of markings m_i and labels l_i such that $m_i \xrightarrow{l_i}_o m_{i+1}$ for all indices $i, i+1$ of π. Typically, a net $\mathcal{N}$ is specified together with an initial marking m_0, giving rise to a *net system* $(\mathcal{N}, m_0)$, in which case executions of $(\mathcal{N}, m_0)$ are the subset of executions of $\mathcal{N}$ starting with m_0.

Let $(\mathcal{N}, m_0)$ be a net system. We say that $(\mathcal{N}, m_0)$ is *safe* iff for each marking $m \in \mathcal{R}(\mathcal{N}, m_0)$ we have $m(p) \leq 1$ for all $p \in P_\mathcal{N}$. Furthermore, $(\mathcal{N}, m_0)$ is *live* iff for each marking $m \in \mathcal{R}(\mathcal{N}, m_0)$ and each transition $t' \in T_\mathcal{N}$ there exists m' with $m \xrightarrow{*}_o m'$ such that t' is enabled at m'. Safety and liveness imply boundedness and deadlock freedom, respectively [11].

Abstract workflows of many different kinds [52, 42] are often modeled using free-choice nets [4, 11], which exhibit the property that conflicts are localized and structurally defined. A *free-choice (FC) net* $\mathcal{N}$ is a net[2] such that for all transitions $t, t' \in T_\mathcal{N}$ with ${}^\bullet t \cap {}^\bullet t' \neq \emptyset$, we have ${}^\bullet t = {}^\bullet t' = \{p\}$ for some $p \in P_\mathcal{N}$.

Motivated by our application, we will only consider safe and live free-choice (SLFC) net systems (i.e. nets sytems with an underlying FC net) in this paper. In our proofs, we frequently exploit their well-understood structural properties that can be most intuitively captured by reduction or synthesis rules [11].

Colored nets [19] are nets with places, transitions, and arcs inscribed with additional information given by functions C and W. The color set $C(p)$ of a place p is the set of possible objects p can carry. The color set $C(t)$ of a transition t can be seen as a set of bindings (also called modes) under which t may occur. We use a specialization of colored nets, where the arc inscription W defines individual objects ("colored" tokens) that are transported by an arc when the associated transition occurs. In fact, this object may depend on the binding under which the transition occurs, which is why $W(p,t)$ and $W(t,p)$ take the form of functions in the definition below. In a graphical representation, W is represented by variables or terms on the arcs, and the color set of a transition is implicitly defined by all the possible bindings of all its local variables, which can be restricted by an optional condition.

A *colored net* N consists of (1) a net $N_\mathcal{N}$; (2) a set of *color sets* $CS_\mathcal{N}$; (3) a *color function* $C_\mathcal{N} : P_\mathcal{N} \cup T_\mathcal{N} \to CS_\mathcal{N}$; and (4) an *arc inscription* $W_\mathcal{N}$ on $F_\mathcal{N}$ such that $W_\mathcal{N}(p,t) : C_\mathcal{N}(t) \to C_\mathcal{N}(p)$ and $W_\mathcal{N}(t,p) : C_\mathcal{N}(t) \to C_\mathcal{N}(p)$. $W_\mathcal{N}$ is extended to a function on $(P_\mathcal{N} \times T_\mathcal{N}) \cup (T_\mathcal{N} \times P_\mathcal{N})$ in such a way that $(p,t) \notin F_\mathcal{N}$ implies $W_\mathcal{N}(p,t)(b) = \emptyset$, and $(t,p) \notin F_\mathcal{N}$ implies $W_\mathcal{N}(t,p)(b) = \emptyset$ for each $b \in C_\mathcal{N}(t)$. A colored net N is *pure* iff the underlying net $N_\mathcal{N}$ is pure.

The semantics of colored nets can be reduced to that of ordinary nets by a flattening operation [47], but we give a direct definition for the purpose of this paper. Let $\mathcal{N}$ be a colored net. A *token* is of the form (p, d) where $p \in P_\mathcal{N}$ and $d \in C(p)$, also called a *data object* for p. A *marking* is a multiset of tokens. An *occurrence* is of the form (t, b) where $t \in T_\mathcal{N}$ and $b \in C(t)$, also called a *binding* for t. The sets of *tokens*, *markings*, and *occurrences* of $\mathcal{N}$ are

[2] To simplify the treatment we do not consider the slightly more general class of extended free-choice nets in this paper.

denoted by $\mathcal{K}_\mathcal{N}$, $\mathcal{M}_\mathcal{N}$, and $\mathcal{O}_\mathcal{N}$, respectively. Given a marking m we define the projection on $p \in P_\mathcal{N}$, written $p(m)$, by $p(m)(p,d) = m(p,d)$ and $p(m)(p',d') = 0$ if $p' \neq p$. Given an occurrence (t,b) we define the multisets ${}^\bullet(t,b)$ and $(t,b)^\bullet$ by ${}^\bullet(t,b)(p,d) = W(p,t)(b)(d)$ and $(t,b)^\bullet(p,d) = W(t,p)(b)(d)$. The *semantics* of a colored net $\mathcal{N}$ is given by the labeled transition system that has $\mathcal{M}_\mathcal{N}$ as its set of states, $\mathcal{O}_\mathcal{N}$ as its set of labels and a transition relation $\longrightarrow$ given by the *occurrence transition relation* $\longrightarrow_o \subseteq \mathcal{M}_\mathcal{N} \times \mathcal{O}_\mathcal{N} \times \mathcal{M}_\mathcal{N}$ defined by $m_1 \xrightarrow{(t,b)}_o m_2$ iff there is a marking m such that $m_1 = m + {}^\bullet(t,b)$ and $m_2 = m + (t,b)^\bullet$. We say that a transition t is *enabled* under a binding b or equivalently that the occurrence (t,b) is *enabled* at a marking m iff $m \xrightarrow{(t,b)}_o m'$ for some marking m'. Typically, a colored net $\mathcal{N}$ is specified together with an initial marking m_0, giving rise to a *colored net system* $(\mathcal{N}, m_0)$. The *binary transition relation*, *reachable* markings, and *executions* are defined as for nets and net systems, respectively.

Given a colored net system $(\mathcal{N}, m_0)$, we define the *flow abstraction* $(\hat{\mathcal{N}}, \hat{m}_0)$ as the net system consisting of the underlying net $\hat{\mathcal{N}}$ of $(\mathcal{N}, m_0)$ and a marking $\hat{m}_0$ of $\hat{\mathcal{N}}$ that is defined as $\eta(m_0)$ where $\eta : \mathcal{M}_\mathcal{N} \to \mathcal{M}_{\hat{\mathcal{N}}}$ lifts $\eta : \mathcal{K}_\mathcal{N} \to \mathcal{K}_{\hat{\mathcal{N}}}$ to multisets, which maps each token (p,d) in $\mathcal{N}$ to a (black) token p in $\hat{\mathcal{N}}$, abstracting from the data. A corresponding function $\eta : \mathcal{O}_\mathcal{N} \to \mathcal{O}_{\hat{\mathcal{N}}}$ maps each occurrence (t,b) in $\mathcal{N}$ to an occurrence t in $\hat{\mathcal{N}}$, abstracting from the binding. From our definitions, it is obvious that η preserves the semantics — i.e., every $m \xrightarrow{o}_o m'$ implies $\eta(m) \xrightarrow{\eta(o)}_o \eta(m')$, but the converse does not typically hold.

3 Toward a Truly Distributed Workflow Model for Fractionated Systems

We introduce two Petri-net-based models, a high-level workflow model, and an implementation-level model, and establish a formal correspondence. We begin with the latter, but we first enrich colored nets with a notion of timestamps that will enable the mapping between these models.

A colored net system $(\mathcal{N}, m_0)$ *maintains timestamps* iff there is a function $ts : \mathcal{K} \cup \mathcal{O} \to \mathbb{R}$ such that the following timestamp conditions hold: (1) $ts(p,d) = 0$ for all $(p,d) \in m_0$ and (2) for each occurrence $(t,b) \in \mathcal{O}$ we have $ts(p',d') = ts(t,b) > ts(p,d)$ for each $(p,d) \in {}^\bullet(t,b)$ and $(p',d') \in (t,b)^\bullet$.

Since we are interested in fully distributed executions of nets, we cannot assume the existence of a global clock. Instead, we are using the weaker logical time [27].

For colored nets that maintain timestamps we canonically restrict the sets of *tokens* $\mathcal{K}_\mathcal{N}$ so that $ts(k) = ts(k')$ implies $k = k'$ for all $k, k' \in \mathcal{K}_\mathcal{N}$, and similarly we restrict the set of *occurrences* $\mathcal{O}_\mathcal{N}$ so that $ts(o) = ts(o')$ implies $o = o'$ for all $o, o' \in \mathcal{O}_\mathcal{N}$. Furthermore, we canonically restrict the *occurrence transition relation* $\longrightarrow_o \subseteq \mathcal{M}_\mathcal{N} \times \mathcal{O}_\mathcal{N} \times \mathcal{M}_\mathcal{N}$ so that $m \xrightarrow{o}_o m' \xrightarrow{*}_o m''$ and $ts(o) = ts(o')$ implies that o' is not enabled at m'', which in particular entails that the timestamps of occurrences on each execution are distinct. Such

requirements of distinct timestamps are satisfied in practice with arbitrary high probability if the timestamp precision is sufficiently high.

Lemma 1. *Given a colored net system $(\mathcal{N}, m_0)$ there is an equivalent colored net system $(\bar{\mathcal{N}}, \bar{m}_0)$ that maintains timestamps in the sense that there are functions $\phi : \mathcal{M}_{\bar{\mathcal{N}}} \to \mathcal{M}_{\mathcal{N}}$ and $\phi : \mathcal{O}_{\bar{\mathcal{N}}} \to \mathcal{O}_{\mathcal{N}}$ such that for all $\bar{m} \in \mathcal{R}(\bar{\mathcal{N}}, \bar{m}_0)$ we have (1) $\bar{m} \xrightarrow{\bar{o}}_o \bar{m}'$ implies $\phi(\bar{m}) \xrightarrow{\phi(\bar{o})}_o \phi(\bar{m}')$, and (2) $\phi(\bar{m}) = m$ and $m \xrightarrow{o}_o m'$ implies $\bar{m} \xrightarrow{\bar{o}}_o \bar{m}'$ for some $\bar{o}$ and $\bar{m}'$ with $\phi(\bar{o}) = o$ and $\phi(\bar{m}') = m'$.*

Proof. By construction of a colored net $\bar{\mathcal{N}}$ that adds a timestamp component to each token and each occurrence so that the timestamp conditions are satisfied (different choices of timestamps can be expressed by different occurrences, i.e., nondeterministically). The functions $\phi : \mathcal{K}_{\bar{\mathcal{N}}} \to \mathcal{K}_{\mathcal{N}}$ and $\phi : \mathcal{O}_{\bar{\mathcal{N}}} \to \mathcal{O}_{\mathcal{N}}$ are defined as a projection removing the timestamp component. The former is lifted to multisets to obtain $\phi : \mathcal{M}_{\bar{\mathcal{N}}} \to \mathcal{M}_{\mathcal{N}}$.

A colored net $\mathcal{N}$ is *monotonic* iff $(p,t) \in F_{\mathcal{N}}$ implies $(t,p) \in F_{\mathcal{N}}$ for all $p \in P_{\mathcal{N}}$ and $t \in T_{\mathcal{N}}$ and $W(p,t) = W(t,p)$ for all $(p,t) \in F_{\mathcal{N}}$. Given a pure colored net $\mathcal{N}$ we define the *monotonic closure* $\bar{\mathcal{N}}$ as the colored net that is identical to $\mathcal{N}$ except that $F_{\bar{\mathcal{N}}} = F_{\mathcal{N}} \cup \{(t,p) \mid (p,t) \in F_{\mathcal{N}}\}$, $W_{\bar{\mathcal{N}}}$ is identical to $W_{\mathcal{N}}$ on $F_{\mathcal{N}}$, and $W_{\bar{\mathcal{N}}}(t,p)(b) = W_{\mathcal{N}}(p,t)(b)$ for all $(p,t) \in F_{\mathcal{N}}$.

Clearly, monotonic nets have monotonic executions where the markings can only grow, because tokens cannot be removed by transition occurrences. Different from Petri nets with test or read arcs, monotonic nets require that the preset of a transition is exclusively accessed through read arcs. The only way to eliminate tokens is by means of an ordering, which leads to the next definition.

A *partially ordered net* $(\mathcal{N}, \prec_{\mathcal{N}})$ consists of a colored net $N_{\mathcal{N}}$ and a strict partial order $\prec_{\mathcal{N}} \subseteq \mathcal{K}_{\mathcal{N}} \times \mathcal{K}_{\mathcal{N}}$ called the *replacement order*. The *occurrence semantics* of $(\mathcal{N}, \prec_{\mathcal{N}})$ extends the occurrence semantics of $\mathcal{N}$ by an additional *replacement transition relation* $\longrightarrow_r \subseteq \mathcal{M}_{\mathcal{N}} \times \mathcal{K}_{\mathcal{N}} \times \mathcal{M}_{\mathcal{N}}$ defined by $m_1 \xrightarrow{(p,d)}_r m_2$ iff $(p,d) \in m_1$, $(p',d') \in m_1$, $(p,d) \prec (p',d')$, and $m_2 = m_1 - \langle (p,d) \rangle$. In this case, we also write $m_1 \longrightarrow_r m_2$. A *partially ordered net system* $(\mathcal{N}, \prec_{\mathcal{N}}, m_0)$ is a partially ordered net $(\mathcal{N}, \prec_{\mathcal{N}})$ together with an initial marking $m_0 \in \mathcal{M}_{\mathcal{N}}$. The *semantics* of a partially ordered net $(\mathcal{N}, \prec_{\mathcal{N}})$ and a partially ordered net system $(\mathcal{N}, \prec_{\mathcal{N}}, m_0)$ is defined as for colored nets or net systems but using an extended transition relation $\longrightarrow$, which is the union of the occurrence transition relation $\longrightarrow_o$ and the replacement transition relation $\longrightarrow_r$.

Partially ordered net systems with an underlying monotonic net will serve as our implementation-level model, which allows inconsistencies to enable efficient distributed execution as long as they can be resolved when needed. Hence, our approach can be best characterized as optimistic, in contrast to a pessimistic approach, which would block progress to make sure that inconsistencies can never arise. Our high-level workflow model is introduced next.

A *multiround workflow system* $(\mathcal{N}, m_0)$ is a colored net system with an initial place p_0 and an initial data object $d_0 \in C(p_0)$, giving rise to an initial marking $m_0 = \langle (p_0, d_0) \rangle$, such that the following conditions are satisfied. (1) The flow

abstraction of $(\mathcal{N}, m_0)$ is a pure SLFC net system. (2) For each marking $m \in \mathcal{R}(\mathcal{N}, m_0)$ we have $|m| = 1$ whenever $m(p_0) \neq \emptyset$. (3) There exists a function $r : \mathcal{K} \to \mathbb{N}$ such that $r(p_0, d_0) = 0$ and (3a) for each occurrence $(t, b) \in \mathcal{O}$ and for all $(p, d), (p', d') \in {}^\bullet(t, b)$ we have $r(p', d') = r(p, d)$ and (3b) for all $(p, d) \in {}^\bullet(t, b)$, $(p', d') \in (t, b)^\bullet$ we have $r(p', d') = r(p, d) + 1$ if $p' = p_0$ and $r(p', d') = r(p, d)$ otherwise.

From this definition it follows that $\langle p_0 \rangle$ is a home marking in the flow abstraction, meaning that it is reachable from any reachable marking [11]. If should be noted that although the flow abstraction is live, it is only an overapproximation of the behavior of $(\mathcal{N}, m_0)$, which due to conditional transitions (as expressed by valid bindings) may terminate (possibly after a number of rounds).

To explicitly track causality and define an appropriate replacement ordering, we enrich each token in our workflow system with some local provenance information.

A multiround workflow system $(\mathcal{N}, m_0)$ *maintains provenance* iff it maintains timestamps and there is a function $h : \mathcal{K} \to 2^{\mathcal{K} \cup \mathcal{O}}$ such that the following provenance conditions hold: (1) $h(p_0, d) = \emptyset$ for $d \in C(p_0)$ and (2) for each occurrence $(t, b) \in \mathcal{O}$ and $(p', d') \in (t, b)^\bullet$ with $p' \neq p_0$ we have $h(p', d') = \bigcup\{h(p, d) \mid (p, d) \in {}^\bullet(t, b)\} \cup \{(t, b)\} \cup \{(p', d')\}$.

We note that provenance is represented by the past causality cone within each round and gives rise to a causal order on tokens defined by set-theoretic inclusion. Our multiround scheme together with Condition (1) ensures that the accumulated provenance information remains bounded, which is essential for any implementation. Clearly, this is only a sufficient condition and not necessary, and more general ways to achieve boundedness are conceivable.

Lemma 2. *Given a multiround workflow system $(\mathcal{N}, m_0)$ there is an equivalent colored net system $(\bar{\mathcal{N}}, \bar{m}_0)$ that maintains provenance in the sense that there are functions $\phi : \mathcal{M}_{\bar{\mathcal{N}}} \to \mathcal{M}_{\mathcal{N}}$ and $\phi : \mathcal{O}_{\bar{\mathcal{N}}} \to \mathcal{O}_{\mathcal{N}}$ such that for all $\bar{m} \in \mathcal{R}(\bar{\mathcal{N}}, \bar{m}_0)$ we have (1) $\bar{m} \xrightarrow{\bar{o}}_o \bar{m}'$ implies $\phi(\bar{m}) \xrightarrow{\phi(\bar{o})}_o \phi(\bar{m}')$, and (2) $\phi(\bar{m}) = m$ and $m \xrightarrow{o}_o m'$ implies $\bar{m} \xrightarrow{\bar{o}}_o \bar{m}'$ for some $\bar{o}$ and $\bar{m}'$ with $\phi(\bar{o}) = o$ and $\phi(\bar{m}') = m'$.*

Proof. By construction of a colored net that adds a provenance component to each token, which is uniquely defined by the provenance conditions above. The functions $\phi : \mathcal{K}_{\bar{\mathcal{N}}} \to \mathcal{K}_{\mathcal{N}}$ and $\phi : \mathcal{O}_{\bar{\mathcal{N}}} \to \mathcal{O}_{\mathcal{N}}$ are defined as a projection removing the provenance component. The former is lifted to multisets to obtain $\phi : \mathcal{M}_{\bar{\mathcal{N}}} \to \mathcal{M}_{\mathcal{N}}$.

Lemma 3 (Deterministic Reversibility). *Let $(\mathcal{N}, m_0)$ be a multiround workflow system that maintains provenance and $m' \in \mathcal{R}(\mathcal{N}, m_0)$. Lifting h from tokens to markings in the natural way, we have the following property. For each occurrence $(t, b) \in h(m')$ that is maximal, i.e., there is no occurrence $(t', b') \in h(m')$ such that $(t, b)^\bullet \cap {}^\bullet(t', b') \neq \emptyset$, there exists a unique marking m such that $m \xrightarrow{(t,b)}_o m'$ and $m \in \mathcal{R}(\mathcal{N}, m_0)$.*

Proof. Since it is not empty, $h(m')$ contains an occurrence that removes the token from the initial place p_0, which defines the marking $m'_0 \in \mathcal{R}(\mathcal{N}, m_0)$ at

the beginning of the current round, i.e., the round of m'. Now m can be reached from m'_0 by executing all occurrences in $h(m')$ except for (t, b) in any order consistent with the causal order induced by $h(m')$, which implies $m \in \mathcal{R}(\mathcal{N}, m_0)$ and $m \xrightarrow{(t,b)}_o m''$. Since each occurrence has a deterministic effect, any causally equivalent execution of a set of occurrences yields the same result, and we obtain $m'' = m'$.

Utilizing the embedded provenance information we can define a replacement ordering that has two purposes and hence two components. First, the system should move forward in time by making sure that more recent tokens replace obsolete tokens (causal replacement). Second, inconsistencies — e.g., violations of mutual exclusion — must be eventually resolved (conflict replacement).

Given a multiround workflow system $(\mathcal{N}, m_0)$ that maintains provenance we define a *causal replacement relation* $\prec_{li} \subseteq \mathcal{K} \times \mathcal{K}$, where $\mathcal{K} = \mathcal{K}(\mathcal{N}, m_0)$ is defined as the union of the support of all reachable markings $\mathcal{R}(\mathcal{N}, m_0)$, as follows: $k \prec_{li} k'$ iff $r(k) < r(k')$, or $k \in h(k')$ and $r(k) = r(k')$. We also define a *conflict replacement relation* $\prec_{al} \subseteq \mathcal{K} \times \mathcal{K}$ as follows: $k \prec_{al} k'$ iff $r(k) = r(k')$ and (1) $p(k) = p(k') = p_0$ and $ts(k) > ts(k')$ or (2) there exists a timestamp-minimal pair (o, o') of occurrences $o \in h(k)$ and $o' \in h(k')$ such that ${}^\bullet o \cap {}^\bullet o' \neq \emptyset$ and $ts(o) > ts(o')$. Here, a pair of occurrences is *timestamp-minimal* in a set of pairs if $ts(o, o')$ is minimal, where the *timestamp* $ts(o, o')$ of a pair (o, o') is defined as $\min(ts(o), ts(o'))$. The *full replacement relation* $\prec_{\mathcal{N}}$ is defined as $\prec_{li} \cup \prec_{al}$.

Lemma 4 (Localized Conflicts). *Let $(\mathcal{N}, m_0)$ be a multiround workflow system that maintains provenance. If, given $k, k' \in \mathcal{K}(\mathcal{N}, m_0)$, there exist distinct occurrences $o \in h(k)$ and $o' \in h(k')$ such that ${}^\bullet o \cap {}^\bullet o' \neq \emptyset$, then (1) $k \neq k'$, (2) there exists a unique timestamp-minimal pair $(\bar{o}, \bar{o}')$ of such occurrences and (3) ${}^\bullet \bar{o} \cap {}^\bullet \bar{o}' = \{\bar{k}\}$ for some token $\bar{k}$.*

Proof. Assume $o \in h(k)$ and $o' \in h(k')$ such that $o \neq o'$ and ${}^\bullet o \cap {}^\bullet o' \neq \emptyset$. Clearly, (1) $k \neq k'$. Otherwise, there is one token $k = k'$ with two mutually exclusive occurrences o and o' in its causal past, which is impossible due to the SLFC property of the flow abstraction. Statement (3) stating that ${}^\bullet o \cap {}^\bullet o'$ is a singleton holds due to the FC property. It remains to prove (2), for which we assume another pair (o_2, o'_2) such that $o_2 \in h(k)$ and $o'_2 \in h(k')$. We first exclude the Case (a) that $o_2 \prec_{li} o$ and $o'_2 \prec_{li} o'$ (similar argument for the symmetric case), in which case we have $ts(o_2, o'_2) < ts(o, o')$ and (o, o') cannot be timestamp minimal. Next we exclude Case (b) that is $o_2 \prec_{li} o$ and $o' \prec_{li} o'_2$ (similar argument for the symmetric case). This case is incompatible with SLFC property. Finally, we exclude the remaining Case (c). Clearly, o and o_2 are causally before k and there must be a transition $\bar{o}$ that merges the tokens from o and o_2. Similarly, there must be a transition $\bar{o}'$ causally before k' that merges o' and o'_2. An intermediate place between o and $\bar{o}$, and similarly between o' and $\bar{o}'$, between o_2 and $\bar{o}$, or between o'_2 and $\bar{o}'$, violates the SLFC property, hence $o = \bar{o}$, and similarly $o' = \bar{o}$, $o_2 = \bar{o}$, and $o'_2 = \bar{o}'$. So we obtain $o = o_2$ and $o' = o'_2$.

Lemma 4, which will be utilized in the following theorem, states that conflists have a unique canonical source and hence allows us to propagate the conflict

relation $\prec_{al}$ along $\prec_{li}$, because no ambiguity about the source can be introduced over time.

The overall coherence of our previous definitions — namely, that $\prec$ results in a strict partial order — is essential and hence established in the subsequent theorem.

Theorem 1 (Well-Formedness). *A monotonic multi-round workflow system $(\mathcal{N}, m_0)$ that maintains provenance together with $\prec_{\mathcal{N}}$ forms a partially ordered net system $(\mathcal{N}, \prec_{\mathcal{N}}, m_0)$.*

Proof. We have to show that $\prec_{\mathcal{N}} = \prec_{li} \cup \prec_{al}$ is a strict partial order. Irreflexivity can be verified by inspecting the definitions of $\prec_{li}$ and $\prec_{al}$, and using Lemma 4 for the latter. For transitivity, it is sufficient to show the following four statements: (1) $\prec_{li} \circ \prec_{li} \subseteq \prec_{li}$, (2) $\prec_{al} \circ \prec_{al} \subseteq \prec_{al}$, (3) $\prec_{li} \circ \prec_{al} \subseteq \prec_{li} \cup \prec_{al}$, and (4) $\prec_{al} \circ \prec_{li} \subseteq \prec_{li} \cup \prec_{al}$. Statement (1) is easy to verify from the definition. For Statement (2) assume $x \prec_{al} y \prec_{al} z$. Clearly, $r(x) = r(y) = r(z)$. If $p(x) = p(y) = p_0$ we must have $p(y) = p(z) = p_0$, because $h(y)$ is empty, and $x \prec_{al} z$ follows from transitivity of the timestamp ordering. The case of $p(y) = p(z) = p_0$ is analogous. Otherwise, we have conflicts at u and u' such that $u \in {}^\bullet o_x \cap {}^\bullet o_y$ and $u' \in {}^\bullet o'_y \cap {}^\bullet o'_z$ for $o_x \in h(x)$, $o_y, o'_y \in h(y)$, and $o'_z \in h(z)$. We consider two cases: (a) $o_y = o'_y$ and (b) $o_y \neq o'_y$. In Case (a), the SLFC property of the flow abstraction implies ${}^\bullet o_x = {}^\bullet o_y = {}^\bullet o_z = \{u\}$. Hence $x \prec_{al} z$ follows from transitivity of the timestamp ordering. In Case (b), there must be an occurrence o''_y eventually joining some output from o_y and o'_y. To cover the choice of o'_z while maintaining the SLFC property of the flow abstraction there must be an occurrence o''_z joining some output from o_y and o'_z. Similarly, there must be an occurrence o''_x joining some output from o_x and o'_y. However, the SLFC property implies ${}^\bullet o''_x = {}^\bullet o''_y = {}^\bullet o''_z = \{u''\}$ for some u'', which contradicts the fact that these are joining occurences. To prove Statement (3) assume $x \prec_{li} y \prec_{al} z$. Clearly, $r(y) = r(z)$. If $r(x) < r(z)$, we are done. If $r(x) > r(z)$ we have $r(x) > r(y)$, which is a contradiction. So $r(x) = r(y) = r(z)$. By definition $y \prec_{al} z$ means that there is a cannonical conflict source $u \prec_{li} y, z$. So we have $u \prec_{li} x \prec_{li} y$ or $x \prec_{li} y$ must factor through u due to the SLFC property, meaning that $x \prec_{li} u \prec_{li} y$. In the latter case, we have $x \prec_{li} u \prec_{li} z$, and we are done by (1). In the former case, it follows that $x \prec_{al} z$ by propagating back the relation $y \prec_{al} z$. Finally, to prove Statement (4), assume $x \prec_{al} y \prec_{li} z$ and as in (3) we can assume $r(x) = r(y) = r(z)$. By definition of $\prec_{al}$ there is a $u \prec_{li} x, y$ at which the conflict occurs first, but the same u witnesses a conflict between z and x. By Lemma 4 the canonical conflict source u is uniquely defined and hence the conflict is resolved in the same direction, giving rise to $x \prec_{al} z$ by propagating $x \prec_{al} y$ forward.

For the reminder of this section, we assume a multiround workflow system $(\mathcal{N}, m_0)$ that maintains provenance and the partially ordered net system $(\bar{\mathcal{N}}, \prec_{\bar{\mathcal{N}}}, m_0)$ where $\bar{\mathcal{N}}$ is the monotonic closure of $\mathcal{N}$. Note that $\mathcal{M}_{\bar{\mathcal{N}}} = \mathcal{M}_{\mathcal{N}}$ and $\mathcal{O}_{\bar{\mathcal{N}}} = \mathcal{O}_{\mathcal{N}}$. For better readability, we use a few conventions. The notation ${}^\bullet x$ and $x^\bullet$ will always refer to $\mathcal{N}$. If not otherwise noted, transition relations will

refer to $\mathcal{N}$, except in the case where they are used with variables like $\bar{m}$ and $\bar{m}'$, in which case they refer to $\bar{\mathcal{N}}$.

As a first observation stated in the following theorem, a causality cone does not contain inconsistencies, and hence conflicts are never visible in provenance of individual tokens (and hence this set is preserved).

Theorem 2. $\mathcal{K}(\mathcal{N}, m_0) = \mathcal{K}(\bar{\mathcal{N}}, m_0)$.

Proof. The direction $\mathcal{K}(\mathcal{N}, m_0) \subseteq \mathcal{K}(\bar{\mathcal{N}}, m_0)$ follows directly from monotonicity. To prove $\mathcal{K}(\bar{\mathcal{N}}, m_0) \subseteq \mathcal{K}(\mathcal{N}, m_0)$ we must deal with the case where a token in $\mathcal{K}(\bar{\mathcal{N}}, m_0)$ contains two occurrences o and o' that are mutually exclusive in $\mathcal{N}$ but both executed in $\bar{\mathcal{N}}$. Given the SLFC property it is impossible to fuse two conflicting branches (in the flow abstraction) such a token cannot exist.

In the following, we use $\psi(m)$ to denote the submarking given by the maximal elements of m w.r.t. $\prec$. This gives rise to an induced equivalence relation $\equiv$ defined by $m \equiv m'$ iff $\psi(m) = \psi(m')$. The following theorems use ψ to state equivalence modulo replacement between our high-level model, the multiround workflow system, and the implementation-level model, given by a monotonic partially ordered net system. By using the function ψ we can state eventual consistency results for the global state, in the sense that when all knowledge would be accumulated by some observer, the replacement ordering will ensure consistency of the observation. Of course, the goal of using this ordering at runtime is to ensure local consistency after each step while the system is running, so that inconsistencies are unlikely to propagate and computational and networking resources are not wasted.

Theorem 3 (Equivalence). *For each marking $\bar{m} \in \mathcal{R}(\bar{\mathcal{N}}, \prec_{\bar{\mathcal{N}}}, m_0)$ we have (1) $\bar{m} \xrightarrow{o}_o \bar{m}'$ (in $\bar{\mathcal{N}}$) implies $\bar{m} \xrightarrow{o}_o m'$ (in $\mathcal{N}$) for some m' such that $\bar{m}' \equiv m'$, (2) $\bar{m} \longrightarrow_r \bar{m}'$ (in $\bar{\mathcal{N}}$) implies $\bar{m} \equiv \bar{m}'$, and (3) for each $m \in \mathcal{R}(\mathcal{N}, m_0)$ we have $m \xrightarrow{o}_o m'$ implies $m \xrightarrow{o}_o \bar{m}'$ (in $\bar{\mathcal{N}}$) for some $\bar{m}'$ such that $\bar{m}' \equiv m'$.*

Proof. To prove Statement (1) assume $\bar{m} \xrightarrow{o}_o \bar{m}'$. Since compared with $\mathcal{N}$, the monotonic closure $\bar{\mathcal{N}}$ has additional output arcs, we have $\bar{m} \xrightarrow{o}_o m'$ (in $\mathcal{N}$) for some $m' = \bar{m}' - {}^\bullet o$. For each $k \in {}^\bullet o$ there exists $k' \in o^\bullet \subseteq \bar{m}'$ such that $k \prec_{li} k'$. Hence, $\psi(\bar{m}') = \psi(m')$. Statement (2) is obvious, because maximal elements are not affected by a replacement transition. To prove Statement (3) assume $m \in \mathcal{R}(\mathcal{N}, m_0)$ and $m \xrightarrow{o}_o m'$. Similar to (1) we have $m \xrightarrow{o}_o \bar{m}'$ for some $\bar{m}' = m' + {}^\bullet o$. For each $k \in {}^\bullet o$ there exists $k' \in o^\bullet \subseteq m'$ such that $k \prec_{li} k'$. Hence, $\psi(\bar{m}') = \psi(m')$.

Completeness as formulated in the following theorem means that our implementation-level model can capture each step of our workflow.

Theorem 4 (Completeness). *Let $\bar{m} \in \mathcal{R}(\bar{\mathcal{N}}, \prec_{\bar{\mathcal{N}}}, m_0)$. Then $\psi(\bar{m}) = m$ and $m \xrightarrow{o}_o m'$ (in $\mathcal{N}$) implies $\bar{m} \xrightarrow{o}_o \bar{m}'$ (in $\bar{\mathcal{N}}$) for some $\bar{m}'$ with $\psi(\bar{m}') = m'$.*

Proof. Assume $\psi(\bar{m}) = m$ and $m \xrightarrow{o}_o m'$. Clearly, $m \subseteq \bar{m}$, so we have $\bar{m} \xrightarrow{o}_o \bar{m}'$ for some $\bar{m}'$, and by monotonicity $\bar{m} \subseteq \bar{m}'$. We have to show only that $\psi(\bar{m}') = m'$, which we decompose into **(1)** $m' \subseteq \psi(\bar{m}')$ and **(2)** $\psi(\bar{m}') \subseteq m'$.

To prove **(1)** assume $k' \in m'$. We consider two cases (a) $k' \in o^\bullet$ and (b) $k' \notin o^\bullet$. Case **(a)** immediately yields $k' \in \bar{m}'$. In Case **(b)** we obtain $k' \in m \subseteq \bar{m} \subseteq \bar{m}'$, and again $k' \in \bar{m}'$. Hence, in both cases it remains to prove that k' is maximal in $\bar{m}'$. So, assume $k' \prec k''$ for some $k'' \in \bar{m}'$ for a proof by contradiction, which is carried out by considering four Cases (a)-(d):

Case **(a)** Assume $k'' \in o^\bullet$ and $k' \in o^\bullet$. This case can be immediately excluded because by definition $\prec$ cannot hold between elements of the postset of an occurrence. Case **(b)** Assume $k'' \in o^\bullet$ and $k' \notin o^\bullet$. It follows that $k' \in m$ and hence k' is maximal in $\bar{m}$. Tracing $k'' \in o^\bullet$ backwards, there must exist $k''' \in {}^\bullet o \subseteq \bar{m}$ such that $k''' \preceq k''$ and $k' \prec k'''$ (relationship to k' does not change with the occurence). This contradicts, however, the maximality of k' just established. Case **(c)** Assume $k'' \notin o^\bullet$ and $k' \in o^\bullet$. Tracing k' backwards, there exists $k''' \in {}^\bullet o \subseteq m$ such that $k''' \prec k' \prec k''$. This means, however, that k''' is not maximal in $\bar{m}$, which contradicts $k''' \in m$. Case **(d)** Assume $k'' \notin o^\bullet$ and $k' \notin o^\bullet$. If follows that $k' \in m - {}^\bullet o$ and hence k' is maximal in $\bar{m}$. The latter, however, is impossible due to $k' \prec k'' \in \bar{m}$.

To prove **(2)** assume $k' \in \psi(\bar{m}')$, which implies that k' is maximal in $\bar{m}'$. If $k' \in o^\bullet$, we obtain $k' \in m'$ and we are done. If $k' \notin o^\bullet$ we conclude that $k' \in \bar{m}$ and hence is maximal in $\bar{m}$, and therefore $k' \in m$. Now there are two cases: If $k' \notin {}^\bullet o$ we obtain $k' \in m'$, and we are done. Otherwise, we have $k' \in {}^\bullet o$, which implies that there exists $\bar{k}' \in o^\bullet \subseteq \bar{m}'$ such that $k' \prec \bar{k}'$. But this contradicts the fact that k' is maximal in $\bar{m}'$.

The following result shows that the implementation-level model is sound. Each of its steps corresponds to a step in the workflow, possibly after rolling back (i.e., eliminating) some redundant occurrences.

Theorem 5 (Soundness). *Let $\bar{m} \in \mathcal{R}(\bar{\mathcal{N}}, \prec_{\bar{\mathcal{N}}}, m_0)$. Then we have (1) $\bar{m} \xrightarrow{o}_o \bar{m}'$ (in $\bar{\mathcal{N}}$) implies $\psi(\bar{m}) \xleftarrow{*}_o \circ \xrightarrow{o}_o \psi(\bar{m}')$ (in $\mathcal{N}$) or $\psi(\bar{m}) = \psi(\bar{m}')$, and (2) $\bar{m} \longrightarrow_r \bar{m}'$ (in $\bar{\mathcal{N}}$) implies $\psi(\bar{m}) = \psi(\bar{m}')$.*

Proof. To prove Statement (1) assume $\bar{m} \in \mathcal{R}(\bar{\mathcal{N}}, \prec_{\bar{\mathcal{N}}}, m_0)$ and $\bar{m} \xrightarrow{o}_o \bar{m}'$. By monotonicity we have $\bar{m} \subseteq \bar{m}'$. We define $m = \psi(\bar{m})$ and consider two cases. If $m \xrightarrow{o}_o m'$ for some m', it is sufficient to show that $\psi(\bar{m}') = m'$, and we can proceed as in the proof of the previous theorem. Otherwise, the occurrence o is not enabled at m because of a replaced token $k'' \in {}^\bullet o$ and $k'' \prec \bar{k}$ for some replacement $\bar{k} \in \bar{m} \subseteq \bar{m}'$. Now there are two possibilities: **(a)** $k'' \prec_{al} \bar{k}$ or **(b)** $k'' \prec_{li} \bar{k}$. In Case (a) we propagate the conflict relation (using Lemma 4) and conclude that $k \prec_{al} \bar{k}$ for all $k \in o^\bullet$, which implies that $\psi(\bar{m}) = \psi(\bar{m}')$ — i.e., the firing of o in $\bar{\mathcal{N}}$ is invisible under ψ, and we are done. In Case (b), k'' must be the source of a conflict, and depending on the occurrence timestamp order there are two subcases **(i)** $k'' \prec_{al} \bar{k}$ for all $k'' \in o^\bullet$ or **(ii)** $\bar{k} \prec_{al} k''$ for all $k'' \in o^\bullet$. In Case (i) we proceed as in Case (a) above. In Case (ii) we use Lemma 3 to roll back

the occurrences in the union of all causal chains $k'' \prec_{li} \bar{k}$ for each $\bar{k}$ satisfying the conditions above ($k'' \prec \bar{k}$ and $\bar{k} \in \bar{m}$). This yields a marking $m'' \xrightarrow{*}_o m$. Now each replaced k'' is restored — i.e., $k'' \in m''$ and $m'' \xrightarrow{o}_o m'$ for some m'. For each $k' \in o^\bullet$, which is nonempty, we have $\bar{k} \prec_{al} k'$ for each $\bar{k} \in m - m''$, i.e., for all $\bar{k}$ that were removed during the rollback. Hence, $\psi(\bar{m}') = m'$ and we are done. Statement (2) of the theorem is obvious, because maximal elements are not affected by a replacement transition.

Theorems 5 and 4 can be composed with Lemma 2 to see that every multiround workflow system can be equipped with provenance information and by taking the monotonic closure executed in a distributed fashion while preserving the behavior of the original workflow specification. Usually, the workflow specification already uses a notion of time, but by an additional composition with Lemma 1 it is also possible to first add timestamps if the original workflow specification is untimed.

Lemma 5. *$\psi(m) = m$ for all $m \in \mathcal{R}(\mathcal{N}, m_0)$.*

Proof. Follows from the fact that all $m \in \mathcal{R}(\mathcal{N}, m_0)$ are consistent, meaning that neither $\prec_{li}$ nor $\prec_{al}$ relates any two tokens of m, which can be established by induction.

Lemma 6. *$\mathcal{R}(\mathcal{N}, m_0) \subseteq \mathcal{R}(\bar{\mathcal{N}}, \prec_{\bar{\mathcal{N}}}, m_0)$.*

Proof. Follows from the fact that each occurrence step in $(\mathcal{N}, m_0)$ from a reachable marking can be simulated by an occurrence step followed by a replacement step in $(\bar{\mathcal{N}}, \prec_{\bar{\mathcal{N}}}, m_0)$.

The following theorem is a concise summary of our results in terms of reachable markings:

Theorem 6. *$\psi(\mathcal{R}(\bar{\mathcal{N}}, \prec_{\bar{\mathcal{N}}}, m_0)) = \mathcal{R}(\mathcal{N}, m_0)$.*

Proof. We can obtain $\mathcal{R}(\mathcal{N}, m_0) \subseteq \psi(\mathcal{R}(\bar{\mathcal{N}}, \prec_{\bar{\mathcal{N}}}, m_0))$ by combining Lemmas 5 and 6. The converse $\psi(\mathcal{R}(\bar{\mathcal{N}}, \prec_{\bar{\mathcal{N}}}, m_0)) \subseteq \mathcal{R}(\mathcal{N}, m_0)$ follows from Theorem 5 by induction. In this process we use Lemma 3 to show that backward steps preserve reachability.

We have implemented a distributed execution engine on top of our NCPS framework [21] that allows the user to specify the multiround workflow system and synthesizes a partially ordered net system under the hood. It also allows the specification of the timing distributions of transitions to allow for randomized delays that reduce the likelihood of conflicts (in fact, it is traded against speed).

Redundancy and diversity as they occur in biological systems are key features of fractionated systems. For example, randomized scheduling of transition occurrences allows us to reduce the likelihood of inconsistencies without coordination overhead by exploiting relative timing. Hence, asynchronous distributed operation, which is often considered as a challenge, is turned into an advantage. FCPS

can be designed with a suitable degree of redundancy, but we cannot generally assume that each logical function can be executed by any hardware/device, at any node/location, or by any agent/person. Hence, some transitions may have additional constraints that could be captured as part of their conditions.

Given that a higher-level model is automatically mapped into a lower-level model that can be directly implemented, our approach is a first step toward model-based design for fractionated systems. The high-level workflow model has a well-defined semantics and can be analyzed or verified using many existing Petri net tools or through a mapping into other formalisms such as rewriting logic [47], which is implemented in the Maude system [8]. In this way, it is possible to perform structural analysis, reachability and deadlock analysis, and more generally model checking of temporal logic properties at a high level of abstraction.

4 Application to Distributed Control of Fractionated Cyber-Physical Systems

To illustrate the application of the theory we use our prototype of a distributed execution engine and a specific case study in the context of mission control for fractionated systems. The example scenario considered here is that of a swarm of UAVs that perform a distributed surveillance mission by flying to a suspicious location and returning to a base location in a formation that creates an effective sensing grid. Figure 1 exemplifies the scenario with a swarm flying to a target location via two possible routes and returning to the initial location. On its way to the target location, a swarm collects information (e.g., images) and uses that information to decide its return route. Note that this presents a highly relevant scenario with current technologies for swarms of UAVs and other micro-vehicles such as robots, smart buoys, and pico-satellites. In this case study, UAVs can communicate and exchange up-to-date information about the progress of their workflow, which enables a distributed and cooperative execution. This problem is representative of a class of FCPS for which transparent scaling and fault tolerance without the need for global coordination is essential.

We leverage the concept of a virtual potential field [20, 41, 6, 38, 28, 25], an attractive approach inspired by physical, chemical, and biological phenomena, to control ensembles like a team of robots or a swarm of UAVs. In a nutshell, each member is driven by the desire to minimize its perception of, and hence its own contribution to, the virtual potential. The advantage of this approach is not only that it is independent of the size of the ensemble and naturally scalable, but also that it is declarative in the sense that the virtual potential can be seen as a specification of the desired state of a system. Recent work in [41] further developed hierarchical potential fields in the context of multirobot systems imposing network topology and collision avoidance constraints with simultaneous path tracking and formation objectives. Instead of deriving the dynamics from a potential it is also possible to directly define artificial forces as in [43], but

the virtual potential approach seems preferable because it offers a higher level of abstraction.

To capture our distributed surveillance mission by a swarm in formation, the potential field needs to be designed to guide UAVs positioned at desired pairwise distance and to be adaptive when a new target location is selected. In the simplest case, the potential field can be a constant around the center of the desired location and increase when the distance from the center grows. To cope with local minima, we use a distributed version of simulated annealing [31] instead of simple gradient descent. In our prototype, the potential field is implemented using the potential function explained next.

We define the potential p capturing the quality of a given swarm configuration as a weighted sum, $p = w_f p_f + w_l p_l$, where p_f and p_l are potentials corresponding to formation and desired location, respectively. The formation potential p_f should be minimal when a swarm creates a hexagonal lattice, utilizing one of several possible distributed sensing grids discussed in [44]. We define the formation potential of a swarm with n UAVs as $p_f = \frac{1}{n}\sum_{i=1}^{n} p_f^i$, where

$$p_f^i = \begin{cases} (R - r_i) & \text{if } r_i < r_c \\ c_r(\frac{1}{r_i} - \frac{1}{R}) & \text{if } r_c \leq r_i \leq R \\ c_a(\frac{1}{R} - \frac{1}{r_i}) & \text{if } R < r_i \end{cases}$$

In the above equation, R is the desired distance between UAVs, and r_i is the current distance between itself and another UAV_i. If they are located at the desired distance ($r_i = R$), then p_f^i presents the lowest potential. Otherwise, p_f^i has repulsive impact if the UAVs are closer than R and attractive impact if they are farther than R. This corresponds to repulsive and attractive forces in artificial physics, where a force law is defined between two particles [45]. In [45] the repulsive force is restricted to remain constant when the distance between two particles gets too small, and we use this threshold distance as a range constant r_c in the above equation. When the UAVs are further apart, p_f^i is defined to be inversely proportional to the distance r with the constants for repulsive impact c_r and for attractive impact c_a. We define c_r to ensure that the potential field is continuous. A location potential p_l is defined as a function of distance between the center of a swarm and the desired location.

The multiround workflow system in Figure 2, composed of $Phase_1$ (from L_0 to L_3) and $Phase_2$ (from L_3 to L_0), describes a sample surveillance mission in the scenario of Figure 1. A goal is injected into our UAV swarm as depicted as an initial token at place G_0 (i.e., goal for location L_0). Distributed simulated annealing attempts to satisfy the goal by flying to location L_0 in a formation using the potential fields explained above. The transition GS generates a token in place F_0 (i.e., a fact representing the location L_0) once goal satisfaction is observed. Since the target location L_3 can be accessed via either L_1 or L_2 as shown in Figure 1, a token at place F_0 enables the NG transitions, which give rise to a *free choice* of a route in $Phase_1$. As explained earlier, the swarm takes a snapshot of the route (i.e., firing of transition TS when the swarm is located at L_1 or L_2 as indicated by a token at place F_1 or F_2, respectively), and processes it by executing transitions IP and FE on images i and i', respectively. This

computation is done as a *parallel* thread while the swarm is flying to L_3. In $Phase_2$ of a round, a conditional transition NG selects a return route to the start location L_0 based on the colored token e at place IF capturing the features extracted from the image taken.

We explain the overall structure of our prototype implementation of our distributed execution engine for multi-round workflow systems using the sample net in Figure 2. It is implemented as an application on top of cyber-application framework [21] that supports simulation models as well as real world deployments of our distributed execution engine. In the beginning, an initial marking is generated by each UAV_i and a token at place G_0 is posted as knowledge. The distributed execution engine allows knowledge sharing about the progress of the workflow when connectivity among UAVs is available. When each UAV has received a token, an event is scheduled to trigger firing of transitions enabled by the corresponding token. For example, transition GS is scheduled to be fired when knowledge about a token at place G_0 is received. The simulated annealing algorithm enables firing of transition GS once goal satisfaction has been observed, which leads the application to post knowledge about a token generated at place F_0 to signal local progress in the workflow.

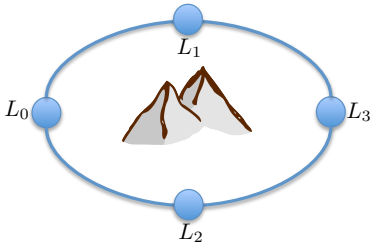

Fig. 1. UAV swarm is initially located at L_0 and needs to fly to the target location L_3 via either L_1 or L_2 due to possible obstacles (e.g., mountains). On its way to L_3, swarm takes snapshot of location L_1 or L_2, which will be used to determine a return route to start location L_0.

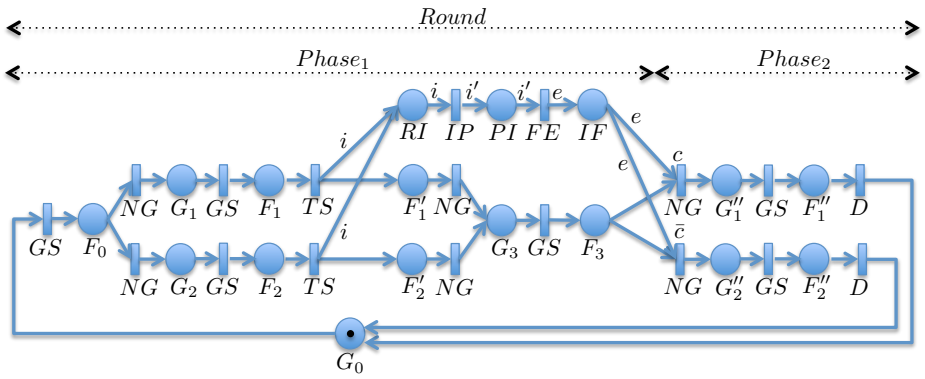

Fig. 2. A sample *multiround* colored workflow system, composed of $Phase_1$ (from L_0 to L_3) and $Phase_2$ (from L_3 to L_0), describes the surveillance scenario in Figure 1. Our distributed engine allows *free-choice* execution (i.e., a token at place F_0 triggers transition NG firing) to determine a route in $Phase_1$ as well as *parallel* execution (i.e., firing of transition TS generates a colored token i in place RI and a black token in place F_1' or F_2') to perform computation on i while the swarm is flying to L_3 so that a conditional transition NG with condition c or $\bar{c}$ can decide a return route based on a colored token e at place IF and a black token at place F_3.

Thanks to $\prec_{li}$, a token at place F_0 can replace a token at place G_0 from which it was generated. Next, transition NG can be fired to generate new goals at places G_1 and G_2. Note that $\prec_{al}$ plays a role here to eliminate a potential redundancy in the distributed execution. In other words, even though transition NG can occasionally generate tokens at both places G_1 and G_2 (due to randomized scheduling of transitions) before $\prec_{li}$ eliminates the token at F_0, $\prec_{al}$ prevents redundant execution paths by replacing inferior tokens (in terms of their timestamps). Since a token indicates current progress of an entire swarm, each UAV can have different notion of the status. For example, UAV_1 is aware of a swarm located at L_0 (i.e., a token at place F_0) and is subsequently choosing to proceed with a new goal to fly to L_1 (i.e., a token at place G_1), while the progress of workflow at UAV_2 still remains at place G_0 due to its lack of computational resources or due to the network delay incurred in knowledge (i.e., token) dissemination.

Once a token is available at place F_1 (or F_2), the transition TS generates a colored token i (indicating that a snapshot has been taken) in place RI and a black token in place F_1' (or F_2'). The swarm is now flying to the target location L_3, which leads to a token at place F_3. In parallel, a raw image is processed (IP) and features are extracted (FE), which leads to a colored token e at place IF. The conditional transitions NG decide a return route based on their conditions c and $\bar{c}$. For instance, if e indicated a potential hazard in the location L_1 (or L_2), NG can avoid to choosing the same route to minimize the risk. It should be noted that by using c and its negation $\bar{c}$ we allow only one execution path in $Phase_2$, which ends with a token at place G_0, but mutual exclusion is required neither by our theory nor by the implementation.

Our distributed execution engine for multiround workflow systems is implemented in a fully decentralized fashion to cope with failures of all kinds. New UAVs can join and leave the swarm at any time since FCPS are open systems, which can also be deployed and scaled up or down incrementally. When a new UAV joins the swarm, knowledge about current progress is exchanged among connected UAVs, which through the ordering can expedite the execution of the joining UAV's workflow system. The underlying cyber-application framework [21] with its partially ordered knowledge-sharing model can shield the distributed execution engine from the complexities of dealing with dynamic topologies (e.g., partitioning and merging), delays/disruptions, and node- /communication failures. For example, in some circumstances (e.g., due to external interference) a swarm may be split into two smaller swarms that get disconnected from each other while they are executing a net with a *free choice*, as in Figure 2. The swarms can take different routes (via L_1 or L_2) before they may reconnect and resynchronize in the future (e.g., at L_3). Since goals are associated with virtual potentials, the underlying optimization algorithm is responsible for resolution of inconsistencies arising in their physical state. Virtual potential fields also allow independent specification of multiple low-level objectives (e.g., location, formation, and connectivity constraints) that can evolve concurrently as part of the workflow.

5 Related Work

The adequate modeling of distributed systems, ranging from information processing systems to workflow in organizations and most notably physical processes, has always been a primary concern for C. A. Petri. One thread of his research was to axiomatize certain classes of Petri nets [37] and to study their causal structure [26], because not all instances correspond to physical phenomena or adequate models of reality, or might simply not be implementable. To lift the level of abstraction, various kinds of high-level nets have been proposed, including colored nets [19] and algebraic nets [39], which can be seen as compact notations for classical Petri nets. Because of their expressiveness, the proper use of Petri nets for the modeling of distributed algorithms remains an important concern; see, e.g., [39]. In addition, various extensions have been studied, which cannot be directly reduced to the classical Petri net model by means of homomorphisms. For instance, the need to model shared resources and nondestructive access — e.g., for applications in delay-tolerant designs (e.g., asynchronous circuits) — has led to the study of Petri nets with test/read arcs (e.g., [53]), which have also been studied under the name contextual nets in [32]. Petri nets and such extensions are also closely related to linear logic and rewriting logic (e.g., [17, 47]). Not surprisingly, Petri nets with read arcs have been found to be important in the modeling of biological systems [50], which are real-world examples of highly distributed and fractionated systems. Monotonic and partially ordered net systems, as introduced in this paper, can be seen as a very fundamental attempt to tame the expressiveness of Petri nets. They can only use read arcs to access tokens and allow tokens to be replaced by means of a partial order. The key idea is to give up the concept of a resource as an atomic object with the rationale that a fully distributed (in fact, fractionated) implementation must always be possible for a large well-defined class of models.

Petri nets have been used to support the model-based design of embedded software (see, e.g., [42] for an approach utilizing free-choice nets to model dataflow). Furthermore, various approaches to the distributed execution of Petri nets have been developed including [7, 14, 3, 23] but all of them use a more or less fine-grained partitioning of the net, in contrast to our fractionated approach that relies on an inherently distributed model that does not require any partitioning. In FCPS, each computing resource can (in the absence of other constraints) execute any transition in an opportunistic and optimistic fashion.

Undirected causality, concurrency, and conflict relations were studied by C.A. Petri in various unpublished lectures as symmetric relational structures (X, li, co, al), and the consistent orientation of causality was investigated in [26]. The conflict relation has also been studied in the context of event structures [54], which are partial orders equipped with a symmetric conflict relation. In this paper, we combine causality and conflict relations into a single consistent ordering by embedding provenance information (also referred to as lineage) in each token to capture its causality cone. Enriching systems with provenance can be seen as a form of distributed reflection, because it allows runtime awareness about the computational process, which in our case is exploited to resolve incon-

sistencies in a fractionated implementation. Systems for maintaining distributed provenance for other purposes are emerging [10], and the early work [15] uses a bounded tree structure as a representation, similar to our causality cones.

A mission control language for a single autonomous underwater vehicle is mapped into Petri nets [35], but the fully distributed control of ensembles addressed in the present paper is much more challenging. A distributed resource in our workflow can be simply the state of distributed software but in cyber-physical systems may also have a counterpart in the real world, like a team of robots or a swarm of vehicles. We use the notion of a virtual potential field, which has been developed in a work including [6, 38, 28, 25, 41].

The computational field model (CFM) [51] is based on concurrent objects in an open distributed environment guided by a metric space defined by mass, distance, gravitational force, repulsive force, and inertia of objects. Its communication model is different from message passing and knowledge sharing. Since even messages are represented as objects, communication becomes a special case of the migration of objects to their optimal/satisfactory locations. The tight integration between the physical world and computations and the notion of FCPS have not been investigated in the context of the CFM.

Simulated annealing has been used for a single robot manipulator in the early work [6] and it is naturally suitable for loosely coupled distributed operation of fractionated systems. The control of a swarm of UAVs can also be seen as a special case of the aggregate motion control as proposed for birds that are flocking based on a distributed behavioral model [40], where each simulated bird navigates according to its local perception of the dynamic environment, the laws of simulated physics, and a set of programmed behaviors. Our approach can also be combined with distributed versions of gradient descent or more powerful search heuristics such as A^* as in [41].

The integration of declarative and quantitative techniques is an active area of research and has been approached in many ways, ranging from probabilistic/ quantitative logics to various forms of stochastic automata, stochastic Petri nets, and weighted rule-based approaches. Virtual potentials can be seen as a generalization of qualitative goals, and potentially consist of multiple weighted subpotentials (corresponding to quantitative subgoals). Here, we use evolving potentials driven by workflows, giving rise to a new form of quantitative declarative specifications for dynamic systems that can serve an executable model. Since this form is independent of the number of participating nodes, it is scalable and naturally suitable for fractionated systems.

6 Conclusion and Future Work

The trend toward model-based design has been a primary driver for executable models, but most work is currently concerned with synchronous systems or systems that can be viewed as such by means of a suitable transformation. This paper investigates the other end of the spectrum — namely, loosely coupled asynchronous systems, and a possible direction for model-based design of frac-

tionated software and more generally FCPS. The idea is that the correct and timely functioning of the system becomes a statistical property if sufficiently many networked resources are available to overcome failures and other deficiencies in the environment.

We introduced a subclass of nets that can be executed in a fully fractionated fashion by means of a partially ordered net model that, unlike traditional models, does not rely on the atomicity of tokens and a corresponding transactional semantics. The most interesting research question is how far can this class be generalized while maintaining these key results. We conjecture that the multiround structure and the free-choice assumptions can be relaxed. It might be possible to cover an even-larger class by relaxing the key theorems. It is also clear that only a subclass of fractionated systems based on the partially ordered knowledge-sharing model can be captured in a Petri net style model, which makes the generalization of our ideas to other well-known formalisms, such as classical logics, linear logic, term rewriting, or process algebra, a natural long-term goal. A purely logical approach to distributed control for a different class of NCPS in [46, 22] can be an alternative. Studying the precise relationship and possible integration of these approaches would be another interesting direction.

Our framework already supports different forms of probabilistic analysis, which can be applied to fractionated systems before system deployment. To improve verifiability of systems, more work is needed on the theoretical side to develop new runtime verification methods such as those proposed in [49], because fractionated systems are typically incrementally deployed, are highly reconfigurable, and operate in a broad range of different environments.

On the practical side, a more efficient implementation and a quantitative evaluation of performance are important next steps. To test our theory in an experimental real-world deployment with distributed-control applications such as those described in the appendix, we are currently building an NCPS testbed consisting of micro-UAVs. Specifically, we are extending inexpensive quadricopters (Parrot AR.Drones) running an on-board Linux system (running low-level flight control software) with additional hardware and sensing capabilities, such as an additional on-board computer (Gumstix), a WiFi module, a digital compass, and a GPS module. The additional on-board computer will enable us to run high-level flight control software and the cyber-application framework with networking and workflow execution capabilities.

Already there are many other applications that could benefit from fractionated software, fractionated hardware, or more generally FCPS at a coarse-grained level. Two particular areas of interest are robotic teams that operate in microfactories and scientific satellite missions involving a large number of inexpensive pico-satellites (e.g., CubeSats), which can act as distributed sensors in space. In the longer-term future, there might be potential to apply these ideas at a more fine-grained level. Since today's hardware designs are essentially networks of subsystems (e.g., many-core architectures), it would be natural to ask if by utilizing sufficient redundancy and diversity, a form of fractionated software can perform useful computations on high-density chips that contain so many highly

unreliable components that failure becomes a normal part of every computation. In this case, we would like to have a clear high-level specification of the computation (e.g., a workflow describing a multistage processing pipeline with many branches), but the mapping and binding to the actual hardware must happen at runtime without the need for explicit global coordination.

An open source prototype is available as part of our multi-platform cyber-application framework at http://ncps.csl.sri.com.

Acknowledgments Support from National Science Foundation Grant 0932397 (A Logical Framework for Self-Optimizing Networked Cyber-Physical Systems) and Office of Naval Research Grant N00014-10-1-0365 (Principles and Foundations for Fractionated Networked Cyber-Physical Systems) is gratefully acknowledged. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of NSF or ONR.

References

1. *OpenCL 1.1 Specification*, Sept. 2010.
2. Arvind. Bluespec: A language for hardware design, simulation, synthesis and verification. In *Proc. First ACM/IEEE Int. Conf. Formal Methods and Models for Co-Design*, MEMOCODE'03, page 249, 2003.
3. F. Baccelli, N. Furmento, and B. Gaujal. Parallel and distributed simulation of free choice Petri nets. *SIGSIM Simul. Dig.*, 25:3–10, July 1995.
4. E. Best. Structure theory of Petri nets: The free choice hiatus. In *Advances in Petri Nets 1986, Part I on Petri Nets: Central Models and Their Properties*, pages 168–205. Springer-Verlag, 1987.
5. O. Brown and P. Eremenko. Fractionated space architectures: A vision for responsive space. In *4th Responsive Space Conf.*, 2006.
6. W. Carriker, P. Khosla, and B. Krogh. The use of simulated annealing to solve the mobile manipulator path planning problem. In *Proc. 1990 IEEE Int. Conf. Robotics and Automation (ICRA '90)*, volume 1, pages 204 – 209, May 1990.
7. G. Chiola and A. Ferscha. Distributed simulation of timed Petri nets: Exploiting the net structure to obtain efficiency. In *14th Int. Conf. Application and Theory of Petri Nets*, pages 14–6, 1993.
8. M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott. All about Maude, a high-performance logical framework. *LNCS*, 4350, 2007.
9. J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. *Commun. ACM*, 51:107–113, January 2008.
10. E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good, A. Laity, J. C. Jacob, and D. S. Katz. Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Sci. Program.*, 13:219–237, July 2005.
11. J. Desel and J. Esparza. *Free Choice Petri Nets.* Cambridge University Press, 1995.
12. F. Dressler. *Self-Organization in Sensor and Actor Networks.* Wiley, 2008.
13. S. Farrell and V. Cahill. *Delay- and Disruption-Tolerant Networking.* Artech House, Inc., Norwood, MA, USA, 2006.
14. A. Ferscha. Optimistic distributed execution of business process models. In *Proc. 31st Annual Hawaii Int. Conf. System Sciences-Volume 7*, HICSS '98, pages 723–732, 1998.

15. A. Gehani and U. Lindqvist. Bonsai: Balanced lineage authentication. In *IEEE Annual Computer Security Applications Conf. (ACSAC)*, 2007.
16. J.-Y. Girard. Linear logic. *Theor. Comput. Sci.*, 50:1–102, January 1987.
17. C. Girault and R. Valk. *Petri Nets for System Engineering: A Guide to Modeling, Verification, and Applications*. Springer-Verlag, 2001.
18. V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard. Networking named content. In *Proc. 5th Int. Conf. Emerging Networking Experiments and Technologies*, CoNEXT '09, pages 1–12, 2009.
19. K. Jensen. *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use, Vol. 1*. Springer-Verlag, 1995.
20. O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *Int. J. Rob. Res.*, 5:90–98, April 1986.
21. M. Kim, M.-O. Stehr, J. Kim, and S. Ha. An application framework for loosely coupled networked cyber-physical systems. In *IEEE/IFIP Int. Conf. Embedded and Ubiquitous Computing*, EUC'10, pages 144–153, 2010.
22. M. Kim, M.-O. Stehr, and C. Talcott. A distributed logic for networked cyber-physical systems. In *Proc. IPM Int. Conf. Fundamentals of Software Engineering*, FSEN'11, 2011.
23. M. Knoke and A. Zimmermann. Distributed simulation of colored stochastic Petri nets with timenet 4.0. In *Proc. 3rd Int. Conf. Quantitative Evaluation of Systems*, pages 117–118, 2006.
24. T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica. A data-oriented (and beyond) network architecture. *SIGCOMM Comput. Commun. Rev.*, 37:181–192, August 2007.
25. W. Kowalczy and K. Kozlowski. Artificial potential based control for a large scale formation of mobile robots. In *Proc. 4th Int. Workshop on Robot Motion and Control*, pages 285–291, 2004.
26. O. Kummer and M.-O. Stehr. Petri's axioms of concurrency- a selection of recent results. In *Proc. 18th Int. Conf. Application and Theory of Petri Nets*, pages 195–214. Springer-Verlag, 1997.
27. L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM*, 21:558–565, July 1978.
28. B. H. Lee. An analytic approach to moving obstacle avoidance using an artificial potential field. In *Proc. Int. Conf. Intelligent Robots and Systems - Volume 2*, page 2482, 1995.
29. B. T. Loo, T. Condie, M. Garofalakis, D. E. Gay, J. M. Hellerstein, P. Maniatis, R. Ramakrishnan, T. Roscoe, and I. Stoica. Declarative networking: Language, execution and optimization. In *Proc. 2006 ACM SIGMOD Int. Conf. Management of Data*, pages 97–108, 2006.
30. J. Meseguer. Conditional rewriting logic as a unified model of concurrency. *Theo. Comput. Sci.*, 96:73–155, 1992.
31. M. Miki, T. Hiroyasu, and K. Ono. Simulated annealing with advanced adaptive neighborhood. In *Second Int. Workshop on Intelligent Systems Design and Application*, pages 113–118, 2002.
32. U. Montanari and F. Rossi. Contextual nets. *Acta Informatica*, 32(6), 1995.
33. A. L. Murphy, G. P. Picco, and G.-C. Roman. Lime: A coordination model and middleware supporting mobility of hosts and agents. *ACM Trans. Softw. Eng. Methodol.*, 15(3):279–328, 2006.
34. D. G. Murray, M. Schwarzkopf, C. Smowton, S. Smith, A. Madhavapeddy, and S. Hand. CIEL: A universal execution engine for distributed data-flow comput-

ing. In *Proc. 8th USENIX Conf. Networked Systems Design and Implementation*, NSDI'11, 2011.
35. N. Palomeras, P. Ridao, M. Carreras, and C. Silvestre. Using Petri nets to specify and execute missions for autonomous underwater vehicles. In *Int. Conf. Intelligent Robots and Systems*, IROS'09, pages 4439–4444, 2009.
36. J. Pereira, L. Rodrigues, and R. Oliveira. Semantically reliable multicast: Definition, implementation, and performance evaluation. *IEEE Trans. Comput.*, 52(2):150–165, 2003.
37. C. A. Petri. Nets, time and space. *Theor. Comput. Sci.*, 153:3–48, January 1996.
38. J. H. Reif and H. Wang. Social potential fields: A distributed behavioral control for autonomous robots. In *Proc. Workshop on Algorithmic Foundations of Robotics*, pages 331–345, 1995.
39. W. Reisig. *Elements of Distributed Algorithms: Modeling and Analysis with Petri Nets*. Springer-Verlag, 1998.
40. C. W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *SIGGRAPH '87*, pages 25–34. ACM, 1987.
41. L. Sentis, M. Mintz, A. Ayyagari, C. Battles, S. Ying, and O. Khatib. Large scale multi-robot coordination under network and geographical constraints. In *Proc. IEEE Int. Symp. Industrial Electronics*, pages 1046–1053, jul 2009.
42. M. Sgroi, L. Lavagno, Y. Watanabe, and A. Sangiovanni-Vincentelli. Synthesis of embedded software using free-choice Petri nets. In *Proc. ACM/IEEE Design Automation Conf.*, DAC '99, pages 805–810, 1999.
43. W. M. Spears and D. F. Gordon. Using artificial physics to control agents. In *Proc. Int. Conf. Information Intelligence and Systems*, ICIIS '99, pages 281–288. IEEE Computer Society, 1999.
44. W. M. Spears, R. Heil, and D. Zarzhitsky. Artificial physics for mobile robot formations. In *IEEE Int. Conf. Systems*, pages 2287–2292, 2005.
45. W. M. Spears, D. F. Spears, J. C. Hamann, and R. Heil. Distributed, physics-based control of swarms of vehicles. *Auton. Robots*, 17(2-3):137–162, 2004.
46. M.-O. Stehr, M. Kim, and C. Talcott. Toward distributed declarative control of networked cyber-physical systems. In *Proc. 7th Int. Conf. Ubiquitous Intelligence and Computing*, UIC'10, pages 397–413. Springer-Verlag, 2010.
47. M.-O. Stehr, J. Meseguer, and P. C. Ölveczky. Rewriting logic as a unifying framework for Petri nets. In *Unifying Petri Nets, Advances in Petri Nets*, pages 250–303. Springer-Verlag, 2001.
48. M.-O. Stehr and C. Talcott. Planning and learning algorithms for routing in disruption-tolerant networks. In *IEEE Military Communications Conf.*, 2008.
49. M.-O. Stehr, C. Talcott, J. Rushby, P. Lincoln, M. Kim, S. Cheung, and A. Poggio. Fractionated software for networked cyber-physical systems: Research directions and a long-term vision. volume 7000 of *LNCS*, pages 110–143, 2011.
50. C. Talcott and D. L. Dill. Multiple representations of biological processes. *Trans. Computational Systems Biology*, 2006.
51. M. Tokoro. Computational field model: Toward a new computing model/methodology for open distributed environment. In *Proc. 2nd IEEE Workshop on Future Trends in Distributed Computing Systems*, 1990.
52. W. M. P. van der Aalst. The application of Petri nets to workflow management. *J. of Circuits, Systems, and Computers*, 8(1):21–66, 1998.
53. W. Vogler, A. L. Semenov, and A. Yakovlev. Unfolding and finite prefix for nets with read arcs. In *Proc. 9th Int. Conf. Concurrency Theory*, CONCUR'98, pages 501–516, 1998.

54. G. Winskel. Event structures. In *Advances in Petri Nets 1986, Part II on Petri nets: Applications and Relationships to Other Models of Concurrency*, pages 325–392, 1987.