# Predicate Detection in Asynchronous Pervasive Environments

Ajay D. Kshemkalyani, *Senior Member*, *IEEE*, and Jiannong Cao, *Senior Member*, *IEEE*

**Abstract**—An important task in sensor networks is to sense locally to detect global properties that hold at some instant in physical time, namely, *Instantaneously*. We propose software logical clocks, called *strobe clocks*, that can be implemented by the middleware when synchronized physical clocks are not available or are too expensive in resource-constrained environments. Strobe clocks come in two flavors—scalar and vector. Let $n$ be the number of sensors and $p$ be the upper bound on the number of relevant events sensed at a sensor. We propose an algorithm using vector strobes that can detect all occurrences of a conjunctive predicate in time $O(n^3p)$. The algorithm has some false negatives but this is the best achievable accuracy in the face of race conditions. We also present a variant algorithm using scalar strobes; it needs time $O(n^2p)$ but may also suffer from some false positives. We provide a characterization of the errors. Both algorithms can also detect relational predicates but with a greater chance of error. The message complexity of strobe clocks (scalar and vector) and both algorithms is $O(np)$, which is the same as that of reporting each sensed event for detection of the predicate even with synchronized physical clocks. We formalize the physical time modality, *Instantaneously*, and show its relationship to the logical time modalities *Definitely* and *Possibly*.

**Index Terms**—Sensor networks, predicate detection, pervasive computing, middleware, strobe clocks, distributed system

## 1 Introduction

A Sensor-actuator network is an asynchronous distributed system of networked-embedded sensors and actuators that aim to sense-monitor-actuate the physical world [18]. The monitoring is achieved via tracking a time-dependent image of the spatiotemporal activities in the physical world [14]. Evaluating predicates on that image is an important problem. The temporal component of the predicate specifies various timing relations on the observed values of the variables that need to satisfy the predicate. The most common of these timing relations is the "instantaneous" snapshot of the variables, namely, the *Instantaneously* modality; although more complex timing relations exist, for example, [3], [17], [26], [29].

Pervasive systems are typically asynchronous distributed systems that sense-monitor-actuate the physical world. A pervasive application is context-aware in that it can adapt its behavior based on the characteristics of the environment [1], [2], [10], [11], [12], [28], [30], [33], [34], [35]. The context is, in the general case, determined by multiple decentralized devices/sensors for context collection. For example, $\psi$: $\texttt{number of persons in room} > 5 \wedge \texttt{temp} > 30^\circ C$. When $\psi$ is true, reset thermostat to 27°C. Here, the radio frequency identification (RFID) reader and the temperature sensor are distributed devices. As another example, $\mu$: $x_i + y_j \geq 400$, where $x_i$ and $y_j$ are the number of people in adjoining rooms, lounge $i$ and dining room $j$, respectively. When $\mu$ is true, switch on the sign: "No Admittance: Please wait." Here again, the two RFID readers are distributed. In both examples, asynchronous communication (wired or wireless) is required to operate the sense-monitor-actuate loop. Note also that the monitoring problem is essentially that of predicate detection.

*Problem Motivation*. A central problem in pervasive systems is to take an "instantaneous" snapshot of the variables, to capture their values at the same instant in physical time in the asynchronous message-passing distributed setting. The existing literature on predicate detection for pervasive environments, for example, [1], [2], [10], [11], [28], [30], [33], [34], [35], except for [12], assumes that it can take instantaneous snapshots of the system. This is possible with physically synchronized clocks. There is a wide body of literature on providing tight clock synchronization for wireless sensor networks (WSNs), for example, RBS, TPSN, TinySync, TSync, see [32] for a survey. However, we observe the following:

- No physically synchronized clock service may be available from a lower layer, as might be the case for very resource-constrained sensors or those in remote environments.
- Furthermore, even if one of the many clock synchronization protocols for WSNs, for example, from [32], is available, it may not be affordable in terms of energy consumption. Such service is not for free as the costs are incurred by a different layer.
- Clock synchronization also imposes a skew $\epsilon$, which leads to imprecision in detecting predicates in physical time. For example, there will be false negatives when the overlap period of the local intervals, during which the global predicate is true, is less than $2\epsilon$ [25].

- *A.D. Kshemkalyani is with the Department of Computer Science, University of Illinois at Chicago, 851 S. Morgan Street, Chicago, IL 60607-7053. E-mail: ajay@uic.edu.*
- *J. Cao is with the Department of Computing, Hong Kong Polytechnic University, PQ816, Mong Man Wai Building, Hung Hom, Kowloon, Hong Kong. E-mail: csjcao@comp.polyu.edu.hk.*

*Manuscript received 5 Mar. 2012; revised 1 June 2012; accepted 17 June 2012; published online 25 June 2012.*
*Recommended for acceptance by J. Wu.*
*For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-2012-03-0175.*
*Digital Object Identifier no. 10.1109/TC.2012.162.*

- Such clocks impose cross-layer dependence and hamper portability.
- Physically synchronized clocks also provide more exposure to privacy concerns and security issues by requiring all users to participate in the network layer synchronization protocol.

For these reasons, we explore the option of using lightweight middleware layer *logical clocks* to detect global predicates. Such clocks also provide layer-independence and allow portability.

*Contributions*. We make the following contributions:

1. We formally propose a software logical clock, called *strobe clock*, that can be implemented by the middleware when synchronized physical clocks are not available or are too expensive in resource-constrained environments. These clocks "synchronize" only at relevant events, and are based on system-wide broadcasts that are similar to those used by physical clocks at lower layers. They are useful to observe system state in physical time at runtime. We use strobe clocks to evaluate predicates [6] under a *physical time* modality specification (*Instantaneously*) using *no physical clocks*. Strobe clocks come in two flavors—scalar and vector.
2. Using vector strobes, we propose an algorithm that can detect all occurrences of a conjunctive predicate [6] in time $O(n^3p)$, where $n$ is the number of sensors and $p$ is the upper bound on the number of relevant events sensed at a sensor. The algorithm suffers from some false negatives in the face of race conditions but this is the best achievable accuracy. We also present a variant algorithm using scalar strobes; it needs time $O(n^2p)$ but may also suffer from some false positives. We characterize the degree of accuracy of the predicate detection algorithms. Both algorithms can also detect the harder class of relational predicates [6], but with a greater chance of error. The message complexity of strobe clocks (scalar and vector) and both algorithms is $O(np)$, which is the *same as* that of reporting each sensed event for detection of the predicate even with synchronized physical clocks.
3. We formalize the *physical time* modality specification (*Instantaneously*) and show how to detect it using *no physical clocks*. In the process, we are using logical strobe clocks for the detection. The algorithms for detecting a predicate in the asynchronous distributed environment use algorithms based on detecting the traditional *Definitely* and *Possibly* modalities for distributed program executions. We formally show the relationship between the *Instantaneously* modality of physical time under the single time axis model and the *Definitely* and *Possibly* modalities of logical time under the multiple time axis partial order time model.

*Organization*. Section 2 discusses related work. Section 3 gives the system model, execution model, and time model for a pervasive system. Section 4 gives the proposed logical strobe clocks—scalar and vector—and discusses how they can be used to simulate physical time. Section 5 gives the predicate detection algorithm using vector strobes and analyzes the inherent inaccuracies. Section 6 gives the predicate detection algorithm using scalar strobes and analyzes the inherent inaccuracies. Section 7 relates the physical time modality *Instantaneously* being detected by the algorithms to the *Definitely* and *Possibly* modalities that are logical time modalities used in the analysis of distributed program executions. Section 8 gives a concluding discussion.

## 2 Related Work

In the distributed computing literature, the Chandy-Lamport global snapshot algorithm is useful to detect stable predicates [5]. Cooper and Marzullo [6] presented the first algorithm to detect unstable relational predicates. The algorithms to detect conjunctive predicates under the *Definitely* and *Possibly* modalities were given by Garg and Waldecker [8], [9]; they have several drawbacks:

1. The algorithms detect only the first occurrence of a predicate and hang; they cannot detect further occurrences of predicates.
2. The algorithms can detect only conjunctive predicates and not relational predicates.
3. The algorithms cannot detect the *Instantaneously* modality.
4. They cannot detect predicates on sensed physical world values but only on in-network variables, because they rely on passively piggybacking time stamps on in-network messages to advance vector time, and there are no in-network messages in sensornets.

Mayo and Kearns [25] gave an algorithm to detect distributed predicates that held at some instant in time in a system using approximately synchronized physical clocks. Stoller [31] likewise gives an algorithm to detect global state predicates with approximately synchronized real-time clocks. In both these approaches, predicates using the *Instantaneously* modality on the execution events are detected using a physical time reference.

An algorithm to detect all occurrences of the *Definitely* modality for conjunctive predicates was given in [20]. Notable differences of the vector strobe algorithm we present to detect the physical time modality *Instantaneously* (Algorithm 1 in Section 5.2) from [20] are the following. First, our algorithm detects conjunctive and relational predicates, not just conjunctive predicates. Second, our algorithm is applicable to sensornets where there is no in-network message communication whereas the algorithm in [20] works only for traditional in-network distributed executions because it relies on passively piggybacking time stamps on in-network messages to advance vector time. Third, our algorithm works in conjunction with the strobe clocks we propose for sensornets, whereas the algorithm in [20] works with traditional Mattern/Fidge vector clocks [7], [24], [27]. Fourth, our algorithm aims to detect the *Instantaneously* modality, not the *Definitely* modality detected in [20]. Fifth, our algorithm is subject to the detection of some false negatives (because of the limitations of our model), whereas the algorithm in [20] is not.

In the pervasive computing community [1], [2], [10], [11], [12], [28], [30], [33], [34], [35], the default assumption made was that instantaneous snapshots could be taken. By considering a pervasive environment as an asynchronous distributed environment, Huang et al. [12] used vector clocks on control messages to detect the *Definitely* modality over a conjunctive predicate defined on physical world variables. Our work builds on this in multiple directions. For example, first, we formalize and define the *Instantaneously* modality of physical time as a criterion of correctness in pervasive environments. Second, we show its relationship to the *Possibly* and *Definitely* modalities of logical time. Third, we formalize vector strobe clocks and scalar strobe clocks for evaluating predicates on values sensed from the physical world. Fourth, we give two algorithms using vector and scalar strobe clocks, respectively, to detect the *Instantaneously* modality. Fifth, we characterize and qualify the errors that occur in these two detection algorithms. Sixth, we are able to detect each and every occurrence of when the predicate holds, (subject to the characterized errors). Detecting all occurrences of a predicate is substantially more complex than detecting its first occurrence only, which is what [12] detects. Seventh, the algorithms presented work for relational predicates in addition to conjunctive predicates, although the level of accuracy is lower for relational predicates.

Our algorithms wait for an interval to complete before it is processed. So there may be a delay in detecting a predicate. Three algorithms [21] to detect the predicate immediately are compared with our algorithms, in [21]. Our vector strobe-based Algorithm 1 does not suffer from false positives whereas the algorithms in [21] suffer from false positives. Our algorithms have lower message complexity than the consensus-based algorithm in [21].

A complete and orthogonal set of interaction types, i.e., placements of two intervals with respect to each other, in terms of how they causally affect one another was presented in [15]. We use this result to analyze the false negative errors that occur in our algorithms.

A preliminary version of this paper, with a sketchy system and execution model section, without any proofs, and without any presentation of logical and physical time modalities, was presented in [19].

## 3 System and Execution Model

Sensor-actuator networks and pervasive environments are distributed systems that interact with the physical world in a sense-and-respond manner [13], [22]. The *world plane* consists of the physical world entities and the interactions among them. The *network plane* consists of the sensors/actuators and the computer network connecting them.

For the network plane, we adapt the standard model of an asynchronous message-passing distributed execution (see [18]). Each sensor/actuator is modeled as a process $P_i(i \in [1, \dots, n])$; the local execution is a sequence of alternating states and state transitions caused by "relevant" events. Assume a maximum of $p$ such events at any process. We assume FIFO communication among the $n$ processes and a reliable system. The communication messages in the network plane may be of two types: 1) Messages to assemble/monitor global properties from locally sensed values, and to output to/actuate the controlled devices. 2) Messages that mimick the communication among the entities of the world plane. Such messages attempt to capture the "true causality" [23] among the events in the world plane, to recreate the time-varying spatiotemporal image of the world plane. The communication in the world plane happens along what the literature terms as *covert channels*; currently science does not know how to detect this communication.

We have used an event-driven execution model. An event occurs whenever a monitored value, whether discrete or continuous, changes significantly. The time duration between two successive events at a process identifies an interval. We model the event-driven activity at processes in terms of intervals. The application seeks to detect a *predicate* that is defined on attribute variables connected as a conjunctive expression (a conjunctive predicate, e.g., $\psi$ in Section 1) [6], [8] or a relational expression (a relational predicate, e.g., $\mu$ in Section 1) [6], and that also specifies certain timing relationships on the intervals in which the attribute values hold. The most popular timing relationship, "concurrent" or "simultaneous" or the "*Instantaneously*" modality, captures the notion of the instantaneous observation of the physical world.

*Problem*. Given a conjunctive or a relational predicate $\phi$ on sensed attribute values of the world plane, detect *each* occurrence of $\phi$ under the *Instantaneously* modality (i.e., holding at the same instant), without using physically synchronized clocks, in the network plane having asynchronous message transmissions.

The algorithms we present are based on detecting overlap among intervals, and then evaluating $\phi$ on overlapping intervals. The algorithms can detect both conjunctive and relational predicates. Our characterization of accuracy is the same for both types of predicates but the level of accuracy is lower for relational predicates.

### 3.1 Time Model

To provide a time base in the middleware, the option is either the partial order model or the linear order model. The partial order of time is isomorphic to the partial order of the traditional distributed execution in the network plane, and is encoded by the causality-based Mattern/Fidge clocks [7], [24]. But, in sensornets, it is unknown how to track the communication over the *covert channels* that induce the causal chains in the world plane; hence, this communication cannot be simulated in the network plane and there are $p^n$ possible consistent global states (CGSs).

For traditional distributed program executions, the global state lattice [6], [24] derived from the causality-based partial order of time is useful to reason about properties of global states. This reasoning is not just for one run, but across *all* runs of the same *deterministic* distributed program. (In a rerun, concurrent events may be reordered, leading to a different path in the state lattice.) But in sensornets and pervasive environments, the physical world does not admit reruns, and there are many *nondeterministic factors* such as nature and human will; so usually applications need to observe the *actual* $np$ states in the *actual* execution. Hence, there does not seem any need to deal with the state lattice.

But logical time—scalar or vector—need not be based strictly on causality as defined by application layer message-passing. We identify a need to build a partial order of time that is not strictly causality based but still useful to observe the world plane under the *Instantaneously* modality of physical time. In the absence of a synchronized physical clock service, we require some time base. The idea is simple—logical time can simply be used to provide a base of linear order/partial order time. Just as lower network layer physical clock synchronization protocols periodically bring multiple hardware clocks (scalars) "in sync" after some drift, so also the middleware layer *strobe clock* that we formalize periodically brings "in sync" the drifting scalars or vectors at each process. Without a strobe, logical clocks drift—they simply tick asynchronously at each relevant local event. The strobe clock is a logical (scalar or vector) clock synchronization service to synchronize the local clocks at "critical events." The strobe clock only needs to guarantee monotonicity of logical time. It can be issued by a process at any time, but no more frequently than when relevant events are locally sensed.

Strobe clock messages are control messages and induce a partial order. This partial order is artificial and arbitrary, unlike the case for distributed programs, where the partial order is induced explicitly by in-network semantic sends and receives. It is important to observe that if the image of the physical world could also track causality, then clock needs to be different from the strobe clock. If it is not, it will introduce false causality induced by the strobes and, hence, infer fake causal relationships and eliminate possible equivalent CGSs.

## 4 Logical Strobe Clocks

We characterize the accuracy of our strobe-clock-based algorithms to detect predicates using a parameter $\Delta$. Define $\Delta$ to be the bound on the asynchronous message transmission delay for a system-wide broadcast. $\Delta$ includes the delays for queuing in local outgoing and incoming buffers, propagation, process scheduling, context switching, and possible retransmissions (to provide reliability), until the received message is processed. In WSNs, and in closed environments such as smart homes, $\Delta$ may be of the order of hundreds of millisecs to secs. This is still small compared to speeds of human and object movements. Although difficult to estimate, $\Delta$ is *not used* by the clock protocols or the predicate detection algorithms; it serves *only* to characterize the degree of imprecision in detecting the predicates. In practice, the accuracy is determined by the actual message transmission delay $\Delta_{actual}$ in any particular race condition, and $\Delta_{actual} \leq \Delta$. Thus, the accuracy of the algorithms is adaptable to the actual operating conditions of the sensor network and can be much better than predicted using the upper bound $\Delta$.

The skew that governs the imprecision using physical clocks is of the order of microsecs to millisecs if software protocols are available. (Hardware solutions achieve nanosec skews but are impractical in sensornets.) Although $\Delta$, that determines the accuracy of our algorithms, is of the order of hundreds of millisecs to secs in small-scale networks, such as smart offices and smart homes, it may be adequate when the number of processes is low and/or the rate of occurrence of sensed events is comparatively low. This is the case for several environments such as office, home, habitat, wildlife, nature, and structure monitoring. In urban settings, such as smart homes and smart offices, the number of sensors is typically low. Furthermore, lifeform and physical object movements are typically much slower than $\Delta$. And in the wild, remote terrain, nature monitoring, events are often rare, compared to $\Delta$. Thus, we may not need the precision (in urban settings or the wild) or be able to afford the associated cost (in the wild) of synchronized physical clocks.

### 4.1 Vector Strobe Clocks

A vector strobe clock $C_i[1,\ldots,n]$ at process $i$ consists of $n$ integers. The *protocol* is given by rules VSC1 and VSC2:

VSC 1. When process $i$ executes (senses) a relevant event:
$C_i[i] = C_i[i] + 1$
System-wide_Broadcast ($C_i$).

VSC 2. When process $i$ receives a strobe $T$:
$(\forall k)\ C_i[k] = \max(C_i[k], T[k])$.

### 4.2 Scalar Strobe Clocks

A scalar strobe clock $C_i$ is maintained by each process $i$. The *protocol* is given by rules SSC1 and SSC2:

SSC 1. When process $i$ executes (senses) a relevant event:
$C_i = C_i + 1$
System-wide_Broadcast ($C_i$).

SSC 2. When process $i$ receives a strobe $T$:
$C_i = \max(C_i, T)$.

It is weaker than the vector strobe clock but is lightweight (strobe size is $O(1)$, not $O(n)$) and it can be used to solve our problem if some false positives can be tolerated.

### 4.3 Features

Although similar to the causality-based Mattern/Fidge vector clocks [7], [24], and Lamport scalar clocks [23], or "interval vector clocks" [4], there are differences:

1. Strobe clocks track the progress of the local logical time counter at each process by catching up or synchronizing on the latest known time of other processes, and do not track the causality induced by message communication; causality-based clocks track the causality induced by the message sends and receives.
2. All strobes are control messages; in causality-based clocks, time stamps are sent on and only on computation messages.
3. On receiving a strobe, the receiver updates its clock but does not tick locally; in causality-based clocks, the receiver ticks on receiving a message.
4. The strobe clock protocol broadcasts its clock no more frequently than at each relevant event (after ticking its local component); in causality-based clocks, the clock value is sent only on all computation messages and vector clock values are isomorphic to the partial order of events.
5. If $\Delta = 0$ (synchronous communication) and the protocol strobes at each relevant event, strobe vectors can be replaced by strobe scalars without sacrificing correctness or accuracy. This is not so for

the causality-based clocks even if $\Delta = 0$; Mattern/Fidge clocks are still more powerful than Lamport clocks when reasoning about the partial order of distributed program executions. This difference arises because strobe clocks broadcast at each sensed event, thus synchronizing all the local clocks instantaneously.

6. As we show, strobe clocks can be used to detect predicates over sensed values of the physical world, where there are no in-network computation messages in the network plane sensornet. Vector clocks have been used to detect predicates on in-network program variables [6], [8], because they rely on passively piggybacking time stamps on in-network messages to advance vector time.

### 4.4 Application: Simulating Physical Time

We show how strobe clocks can be used to detect conjunctive or relational predicates that held at some instant in physical time (to simulate a single time axis). We approximate the physical time axis as best as theoretically possible using vector strobe clocks.

A relevant event is modeled as a quadruple $e = (P_i, a_j, val, t_s)$ to represent the host process, attribute sensed, attribute's value, and the physical time of its occurrence (unknown in our model). For each process/attribute pair, an interval is represented by a value, start time, and finish time as $I = (val, t_s, t_f)$. The interval is defined by two consecutive events $(P_i, a_j, val1, t1)$ and $(P_i, a_j, val2, t2)$ for that $(P_i, a_j)$ pair as: $I = (val1, t1, t2)$. Our goal is to evaluate whether a predicate $\phi$ holds whenever the global state changes. Using strobe clock values to time stamp relevant events and hence intervals, we are using monotonic logical time stamps that are synchronized as tightly as theoretically possible.

Let $\mathcal{I} = \{I_1, \ldots, I_n\}$ be a set of intervals, one per process.

**Definition 1 ($Instantaneously^{\mathcal{I}}$ modality of overlap).** *All $I_i \in \mathcal{I}$ overlap in physical time, i.e., $Instantaneously^{\mathcal{I}}$, iff*

$$\max_i(I_i.t_s) < \min_i(I_i.t_f). \tag{1}$$

For this set of intervals $\mathcal{I}$, we define a number:

**Definition 2 (Overlap of intervals).** $overlap(\mathcal{I}) = \min_i(I_i.t_f) - \max_i(I_i.t_s)$

Condition in (1) is equivalent to

$$\forall i \forall j, I_i.t_s < I_j.t_f. \tag{2}$$

As we do not have access to physical time, we use logical time $C$ values as a best approximation. Here, $I.C_s$ and $I.C_f$ denote the start and finish logical clock values of interval $I$.

*Approximation using vector strobes*. To approximate (2), we check for:

$$\forall i \forall j, I_i.C_s[i] \leq I_j.C_f[i]. \tag{3}$$

For $\Delta > 0$, we have (see Section 5.1)

$$I_i.C_s[i] \leq I_j.C_f[i] (\Longrightarrow \wedge \not\Longleftarrow) I_i.t_s < I_j.t_f. \tag{4}$$

For $\Delta = 0$, we have

$$I_i.C_s[i] \leq I_j.C_f[i] \Longleftrightarrow I_i.t_s < I_j.t_f. \tag{5}$$

*Approximation using scalar strobes*. To approximate (2), we check for:

$$\forall i \forall j, I_i.C_s \leq I_j.C_f. \tag{6}$$

For $\Delta > 0$, the following holds (see Section 6.1):

$$I_i.C_s \leq I_j.C_f (\not\Longrightarrow \wedge \not\Longleftarrow) I_i.t_s < I_j.t_f. \tag{7}$$

Its utility is nevertheless shown (see Section 6.1).

When $\Delta = 0$, the scalar strobe clock behaves exactly like the vector strobe clock, and the following equation holds:

$$I_i.C_s \leq I_j.C_f \Longleftrightarrow I_i.t_s < I_j.t_f. \tag{8}$$

To approximate (2), if we were to replace $\leq$ by $<$ in the test of (6), we get

$$\forall i \forall j, I_i.C_s < I_j.C_f, \tag{9}$$

$$I_i.C_s < I_j.C_f (\not\Longrightarrow \wedge \not\Longleftarrow) I_i.t_s < I_j.t_f \qquad \text{when } \Delta > 0 \tag{10}$$

$$I_i.C_s < I_j.C_f \Longleftrightarrow I_i.t_s < I_j.t_f \qquad \text{when } \Delta = 0. \tag{11}$$

There is a tradeoff in the use of the approximation of (6) versus the approximation of (9). This tradeoff will be analyzed in Section 6.1.

*Approximations*. The physical world execution traces one path (of the $O(\frac{(pn)!}{(p!)^n})$ possible paths) through $np$ of the $O(p^n)$ states in the state lattice. The goal is to identify the states in this path and evaluate the predicate in them. At each sensed event, a strobe is broadcast atomically at a send event. The send nor the receive events for the strobes have any semantic significance. The control messages for the strobe clock create artificial causal dependencies that help to approximate instantaneous observation because they eliminate many of the $O(p^n)$ states in which the intervals did not overlap. But the number of possible CGSs in the sublattice induced by the strobes is still $O(p^n)$. The faster the strobe transmissions, the leaner the lattice; in the limit, $\Delta = 0$, we get a linear order of $np$ states. Unlike executions of distributed programs where program-determined semantic messages may not get sent for long periods while local variables' values change multiple times, resulting in fat lattices, clock strobes get sent at each value change. This gives us the "*slim lattice postulate*" for CGSs in sensornet observations. A message gets sent for each sensed event, causing a reduction in the number of CGSs. Fig. 1 shows an example execution and corresponding value-lattice. A control message is broadcast atomically with each sensed event; there are no underlying computation messages in the network plane.

For these CGSs, due to the inherent transmission delays, it is theoretically possible either to verify that the intervals overlapped, or to only suspect that they overlapped. To identify all cases where overlap is verified can be done in polynomial time. This is because there are at most $np$ such candidate solutions and each candidate solution takes at most $O(n^2)$ time for verification in our Algorithm 1. But to identify all cases where overlap is suspected requires exponential time, even if $\phi$ is conjunctive. This is because this task requires evaluating each state in the lattice of

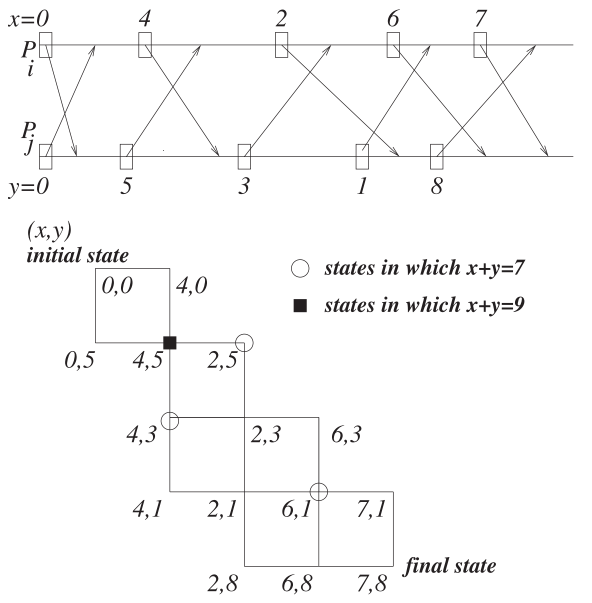


Fig. 1. An example execution of strobe messages based on sensed values, and its corresponding lattice of CGSs.

CGSs, even if those states did not occur in the actual execution. This is discussed further in Section 7.

## 5 Detection Using Vector Strobes

### 5.1 Characterization and Analysis of Accuracy

To *characterize* the degree of imprecision in (4) that creeps in when we simulate physical clocks with logical vector strobe clocks, consider any pair of intervals $X_i$ and $Y_j$ at $P_i$ and $P_j$, respectively. These intervals can be placed with respect to each other in one of 29 mutually *orthogonal* ways *only*, as shown in the left part of Fig. 2 [15]. $X$ is shown in a fixed position as a rectangle whereas the possible relative positions of $Y$ are shown by horizontal lines. The dashed lines indicate the boundaries of the causal past and causal future of the events $\min(X)$ and $\max(X)$, (induced by strobes). Thus, $\downarrow\min(X)$ and $\min(X)\uparrow$ indicate the boundaries of the causal past and the causal future of event $\min(X)$. $\downarrow\max(X)$ and $\max(X)\uparrow$ indicate the boundaries of the causal past and the causal future of event $\max(X)$. The distinction between those positions of $Y$ with two labels each is shown in the right part of Fig. 2. In each pair, the difference lies in how $\downarrow\max(Y)$ and $\min(Y)\uparrow$ intersect with $X$.

Of the 29 mutually orthogonal placements, the following 18 logical placements (labeled in normal font in the figure):

$$\begin{aligned}&ID, IX, ID', IU, IE, IW, IE', IT, IF,\\ &IS, IO, IL, IP, IL', IM, IM', IN, IN'\end{aligned} \tag{12}$$

satisfy (3). These are exactly the placements of interval $Y$ that stretch to the right of the dashed line $\min(X)\uparrow$ (for these $I_i.C_s[i] \leq I_j.C_f[i]$) and which also stretch to the left of the dashed line $\downarrow\max(X)$ (for these, $I_j.C_s[j] \leq I_i.C_f[j]$). Using (4) and (2), observe that these placements imply physical time modality *Instantaneously*. If our algorithm evaluates the predicate only when these modalities in (12) are satisfied among the intervals in $\mathcal{I}$, the algorithm will not detect any false positives and can verify overlap. This is achieved by our proposed Algorithm 1.

Observe that placements $IA$ and $IQ$ of interval $Y$ (labeled in italics in the figure) will never overlap with interval $X$. Of the 29 mutually orthogonal placements, excluding $IA$, $IQ$, and the 18 placements of (12), we have the following remaining nine logical placements (labeled in boldface in the figure):

$$IB, IR, IC, IV, IG, IH, IK, II, IJ, \tag{13}$$

which may or may not overlap in physical time, and it is theoretically impossible using logical clocks and asynchronous communication to determine the *Instantaneously* physical time modality specification in these cases. (Thus, it is impossible to verify whether the intervals $X$ and $Y$ actually overlapped.)

With multiple processes, even if intervals at one pair of processes are related by one of these nine placements, to not raise false alarms, the algorithm should not detect *Instantaneously* even if the intervals happen to overlap. One can only suspect that overlap occurred. These are, thus, *potential* false negative cases. (If need be, these cases can also be detected; see the algorithms in [4] and Section 7.) The extent

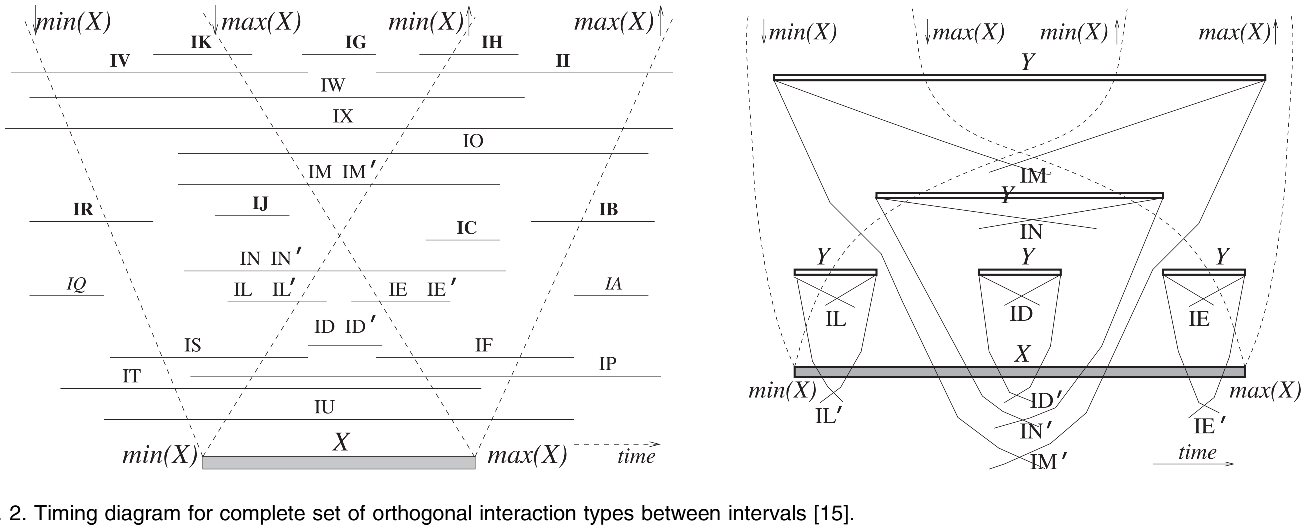


Fig. 2. Timing diagram for complete set of orthogonal interaction types between intervals [15].

of false negatives will depend on the duration of intervals, their placements, and the bound $\Delta$.

**Theorem 1.** *Using vector strobe clocks, the correctness of detecting the* Instantaneously *modality using (3) is characterized as follows:*

1. *If overlap $\geq \Delta$, then the modality can be correctly detected.*
2. *If $0 \leq$ overlap $< \Delta$, then a false negative* may *occur.*
3. *If overlap $< 0$, then the modality can be correctly detected as not holding.*

**Proof.**

1. We have $\forall i \forall j$, $I_j.t_f - I_i.t_s \geq \Delta$, which implies (2). Also, we have that the vector strobe sent by $P_i$ at $I_i.t_s$ will reach $P_j$ before $I_j.t_f$ and hence $I_i.C_s[i] \leq I_j.C_f[i]$ as per VSC1 and VSC2. Hence, (2) implies (3).

   Equation (3) implies (2) due to the implementation of VSC1 and VSC2 and the nonnegative transmission times involved.

   Hence, the *Instantaneously* modality will be correctly declared using (3) as an approximation to (2).
2. There are three possible subcases:

   a. Every pair of intervals is related by one of the relations in (12).

      Then, from Fig. 2, observe that for each of these relations, $I_i.t_s < I_j.t_f$ (using a simple left-to-right scan) and also $I_i.C_s \leq I_j.C_f$ because the dashed lines indicate the traversal of the vector strobes. (It cannot be visually depicted that $0 < overlap < \Delta$ because that is a physical time specification whereas the diagram is a logical time depiction.) In this subcase, (2) if and only if (3), and hence, the modality holds and can be correctly detected using (3) as an approximation to (2).

   b. At least one pair of intervals is related by one of the relations in (13) and all other pairs of intervals are related by one of the relations in (12) or (13).

      For any pair of intervals (say, $I_i$ and $I_j$) related by (13), it is possible that the overlap duration between them is less than $\Delta$. If so, the vector strobe sent by $I_i$ at $I_i.t_s$ *may not* reach (directly or transitively) $P_j$ before $I_j.t_f$, in which case $I_j.C_f[i] \not\geq I_i.C_s[i]$. Equation (3) does not hold and overlap between $I_i$ and $I_j$ will not be detected using (3) as an approximation to (2)—leading to a false negative.

   c. At least one pair of intervals is related by one of the two relations $IA$ and $IQ$ not in (12) or (13).

      Observe that in this subcase, $overlap < 0$, contradicting the case assumption.
3. There is at least one pair of intervals $I_i$ and $I_j$ in $\mathcal{I}$ such that $I_i.t_f - I_j.t_s < 0$. The vector strobe sent by $P_j$ at the start of $I_j$ with time stamp $I_j.C_s$ will not reach $P_i$ before $I_i$ ends, and by VSC1 and VSC2, $I_i.C_f[j] \not\geq I_j.C_s[j]$. Hence, (3) will not be satisfied and the *Instantaneously* modality will not be declared, correctly. Hence, no false positives will occur. □

## 5.2 Detection Algorithm Using Vector Strobes

**Algorithm 1** Vector strobe algorithm at $P_i$ to detect predicates.

```
queue of event: $(\forall z)EQ[z] = \langle(z, a, init\_val, C_{init})\rangle$
queue of interval: $(\forall z)IQ[z] = \langle\rangle$
set of int: $updatedQs$, $newUpdatedQs = \{\}$
int: $M[1 \dots n]$

When event $e = (P_i, a_i, val, C)$ occurs at $P_i$:
1) $C_i[i] = C_i[i] + 1$
2) System-wide_Broadcast $(P_i, a_i, val, C)$
On $P_i$ receiving event strobe $e = (z, a, val, C)$ from $P_z$:
3) Execute VSC2 (Section 4.1)
4) $(z, a, v, C') = dequeue(EQ[z])$
5) $enqueue(IQ[z], (v, C_s = C', C_f = C))$
6) $enqueue(EQ[z], (z, a, val, C))$
7) if (number of intervals on $IQ[z]$ is 1) then
8)     $updatedQs = \{z\}$
9)     while ($updatedQs$ is not empty)
10)        $newUpdatedQs$=$\{\}$
11)        for each $h \in updatedQs$
12)            if ($IQ[h]$ is non-empty) then
13)                $X$ = head of $IQ[h]$
14)                for $j = 1$ to $n$ ($j \neq h$)
15)                    if ($IQ[j]$ is non-empty) then
16)                        $Y$ = head of $IQ[j]$
17)                        if ($X.C_f[j] < Y.C_s[j]$) then
18)                            $newUpdatedQs =$
                               $\{h\} \cup newUpdatedQs$
19)                        if ($Y.C_f[h] < X.C_s[h]$) then
20)                            $newUpdatedQs =$
                               $\{j\} \cup newUpdatedQs$
21)        Delete heads of all $Q_k$, where $k \in newUpdatedQs$
22)        $updatedQs = newUpdatedQs$
23)        if (all queues are non-empty) and
           ($updatedQs = \emptyset$) then
24)            heads of queues satisfy Eqn 3 (hence Eqn 2)
25)            if $\phi((\forall k)head(IQ[k]).val)$ is true then
26)                raise alarm/update world image
27)            $(\forall k)M[k] = head(IQ[k]).C_f[k]$
28)            for $k = 1$ to $n$
29)                if $(\forall j, (j \neq k))$
                   $head(IQ[k]).C_f[j] < M[j]$ then
30)                    mark $k$ for dequeuing
31)            for $k = 1$ to $n$
32)                if $k$ is marked for dequeuing then
33)                    $dequeue(head(IQ[k]))$
34)                    $updatedQs = updatedQs \cup \{k\}$
```

Algorithm 1 uses strobe clocks to detect a physical time modality, namely *Instantaneously*. To design the algorithm, we guarantee no false positives. We seek to detect a set of intervals, one per process, such that 1) pairwise, they satisfy one of the placements in (12), and hence from (3) and (4), satisfy (2); and 2) $\phi$ is true over the attribute values in these intervals. Line 7 onwards encode the logic for the repeated detection of $\phi$.

Each process $P_i$ maintains an interval queue $IQ[z]$ and an event queue $EQ[z]$, for strobes received from process $P_z$. (Assume one attribute per process.) A strobe also piggybacks the sensed event's quad-descriptor. Note, for online detection of properties of the world plane, a sensor has to immediately report the event to a monitor *even while using* *(synchronized) physical clocks*; so a message transmission is unavoidable. This algorithm detects both conjunctive and relational predicates:

- For a conjunctive $\phi, val$ at any process alternates between 0 and 1; when $val = 1$ on a received strobe, the completed interval having $val = 0$ need not be processed (skip steps 5 and 7 onwards).
- For a relational $\phi$, each interval is processed.

Visually speaking, we step through the execution, state by state, in physical time, by constructing states from the interval queues. The challenge is doing so without using physical clocks. The approach can be viewed as jumping through a virtual state lattice of the partial order induced by the strobes, without actually constructing one. In a nutshell, 1) we loop, queuing intervals from each process, until we identify a set $\mathcal{I}$ having $overlap > 0$. If $\phi$ holds over the values in $\mathcal{I}$, raise an alarm. 2) We then prune the $IQs$ judiciously to ensure progress (more solutions can be detected) and safety (not to miss any solution). In the process, we have to ensure polynomial time overhead. An important feature of the algorithm is that it does *repeated* detection of $\phi$ as strobes are generated, each time $\phi$ becomes true.

## 5.3 Correctness Argument

The correctness for a conjunctive predicate, subject to the inherent false negatives constraints of the model as explained in Section 5.1, follows from the following observations:

1. The interval set forming the first solution is correctly detected using the logic of lines 7-22. Specifically, as observed in Section 5.1, the 18 placements of (12) satisfy the tests of (3):
   a. *Safety*. These tests are implemented in lines 17, 19, and eliminate only those intervals that do not satisfy the test.
   b. *Liveness*. In at most $n(n-1)$ executions of lines 17, 19, a solution set $\mathcal{I}$ gets found or an interval gets deleted.
2. *Safety*. Once a solution $\mathcal{I}$ is detected, only intervals $X_i \in \mathcal{I}$ that cannot be part of another solution are deleted from their queues $IQ[i]$, in line 23 onwards.
3. *Liveness*. For any solution set $\mathcal{I}$, at least one interval gets deleted from its queue in line 23 onwards.
4. The next solution is again found by the logic of lines 7-22.
5. The number of solution interval sets is bounded by the total number of intervals, which is $np$. As a solution set contains $n$ intervals, this bound is more accurately stated as $n(p-1)+1$.

In addition to conjunctive predicates, we aim to detect relational predicates as best as possible. We are detecting each global state that satisfies the relational predicate in the *Instantaneously* modality, subject to the inherent false negative constraints of the model and a polynomial time overhead. The algorithm satisfies the characterization of Section 5.1 and Theorem 1 for both conjunctive and relational predicates. But the level of accuracy is lower for relational predicates. This is because the algorithm verifies whether a global state must have occurred. For a relational predicate, it may happen that no one state must have occurred, but collectively examined, one of some set of states must have occurred. To examine collectively requires building the lattice of CGSs, which cannot be done in polynomial time despite the slim lattice postulate.

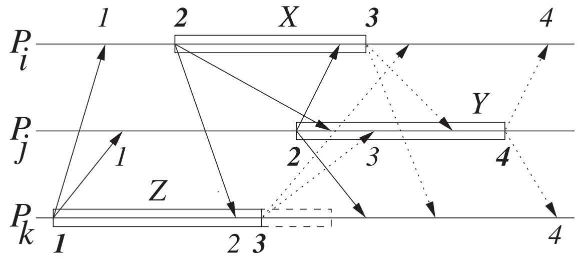


Fig. 3. False positives with scalar strobes.

Consider the example execution and corresponding value-lattice in Fig. 1. No one state $(x,y) = (4,3)$ and $(x,y) = (2,5)$ must have occurred, but collectively, one of $\{(4,3),(2,5)\}$ must have occurred because every path from the initial state to the final state passes through one of these two states. Hence, to determine whether a state in which $x+y=7$ must have occurred, it is necessary to build the state lattice, which is expensive. Without the state lattice, the algorithm cannot verify that either of $(4,3)$ and $(2,5)$ must have occurred; hence, the level of accuracy is lower for relational predicates than for conjunctive predicates. This is discussed further in Section 7.

## 5.4 Complexity

To find the first solution that satisfies (3), the algorithm continues till each interval queue $IQ[k]$ has a head element. The time overhead for this is $O(n^2)$. For each set of intervals that satisfy (3), $O(n^2)$ time is needed to eliminate at least one interval; and let $O(f(\phi))$, the time to evaluate $\phi$, be $O(n)$. Such a set needs to be considered at most $np$ times. The time complexity is $O(np(n^2 + O(f(\phi))))$, or simply $O(n^3p)$, to find all solutions.

Message cost is $O(np)$ system-wide broadcasts, each of $n$ integers. Note that the lower bound is $\Omega(np)$ (wireless) broadcasts, of 1 integer each, because each sensed event needs to be reported to a sink for assembly *even with synchronized physical clocks*. However, if a multihop network or a point-to-point medium was used in the network plane, the $O(np)$ broadcasts would become $O(n^2p)$ messages. In contrast, if physically synchronized clocks were used, the $O(np)$ transmissions of the sensed events to the sink would require $O(np \cdot \log n)$ or even $O(n^2p)$ message (re)transmissions to reach the sink, assuming a tree or a linear array configuration, respectively. In the worst case, assuming a tree configuration, the message complexity is a factor of $\frac{n}{\log n}$ more than with synchronized hardware clocks.

## 6 Detection Using Scalar Strobes

Using the test in (6), as the best approximation to (2), (7) indicated that strobe scalars do not guarantee correctness of detecting the *Instantaneously* modality.

## 6.1 Characterization and Analysis of Accuracy

*Potential false positives*. Consider the example execution in Fig. 3, showing three intervals $X$, $Y$, and $Z$, at processes $P_i$, $P_j$, and $P_k$, respectively. (Assume all clocks are initially 0.)

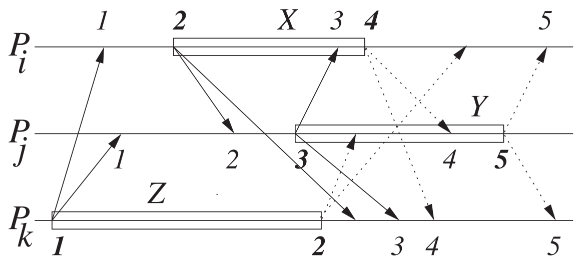


Fig. 4. False negatives with scalar strobes.

The beginning of each interval $I$ is time stamped $I.C_s$ by the scalar strobe clock rules, SSC1 and SSC2. The end of the interval is time stamped $I.C_f$ by the start time of the next interval at that process. Messages in regular lines are the scalar strobes sent at the start of $X$, $Y$, and $Z$; messages is dotted lines are those sent at the start of the next intervals following these. The local clock value is shown only when it changes. Time stamps in bold are the start and end time stamps of the intervals.

By applying the test of (6) for each pair of intervals, observe that the test is satisfied; but although $Y.t_s = 2 \leq Z.t_f = 3$, they do not overlap. This is a false positive. If $Z$ had extended in time as indicated by the dashed extension, then the scalar strobe clocks would remain the same and the detection would have been a true positive.

*Potential false negatives*. Consider the example execution in Fig. 4. By applying the test of (6) for each pair of intervals, observe that the test is not satisfied because $Y.t_s = 3 \not\leq Z.t_f = 2$. However, they do overlap. This is a false negative. If $Z$ had finished earlier in physical time than the start of $Y$ and the dotted strobes sent at the end of $Z$ reached at the same time as they do currently in the figure, then the scalar strobe clocks would remain the same and the nonsatisfaction of (6) would have been a true negative.

Using (9) instead of (6) still gives false positives and false negatives. Observe, when $Y.C_s = Z.C_f$, (6) risks a false positive whereas (9) risks a false negative.

**Theorem 2.** *Using scalar strobe clocks, the correctness of detecting the* Instantaneously *modality using (6) (or (9)) is characterized as follows:*

1. *If overlap $\geq \Delta$, then the modality can be correctly detected.*
2. *If $0 \leq$ overlap $< \Delta$, then a false negative* may *occur.*
3. *If $-\Delta <$ overlap $\leq 0$, then a false positive* may *occur.*
4. *If overlap $\leq -\Delta$, then the modality can be correctly detected as not holding.*

**Proof.**

1. We have $\forall i \forall j$, $I_j.t_f - I_i.t_s \geq \Delta$, which implies (2). Also, we have that the scalar strobe sent by $P_i$ at $I_i.t_s$ will reach $P_j$ at or before $I_j.t_f$ and, hence, $I_i.C_s \leq I_j.C_f$ as per SSC1 and SSC2. Hence, (2) implies (6).

   Equation (6) implies (2) due to the implementation of SSC1 and SSC2 and the nonnegative transmission times involved.

   Hence, the *Instantaneously* modality will be correctly declared using (6) as an approximation to (2).
2. A false negative may or may not occur, as explained earlier using Fig. 4. The scalar strobe sent by $P_j$ at the start of $Y$ may not reach $P_k$ before the end of $Z$ if $0 \leq$ *overlap* $< \Delta$, i.e., $Z.t_f - Y.t_s < \Delta$. If the scalar strobe does not reach $P_k$ before $Z.t_f$, the false negative occurs, else there is no false negative.
3. A false positive may or may not occur, as explained earlier using Fig. 3. The scalar strobe sent by $P_k$ at the end of $Z$ may not reach $P_j$ before the start of $Y$ if $-\Delta <$ *overlap* $\leq 0$, i.e., $Z.t_f - Y.t_s > -\Delta$. If the scalar strobe does not reach $P_j$ before $Y.t_s$, the false positive occurs, else there is no false positive.
4. There is at least one pair of intervals $I_i$ and $I_j$ in $\mathcal{I}$ such that $I_i.t_f - I_j.t_s \leq -\Delta$. The scalar strobe sent by $P_i$ at the end of $I_i$ with time stamp $I_i.C_f$ will reach $P_j$ at or before $I_j$ begins, and by SSC1 and SSC2, $I_j.C_s > I_i.C_f$. Hence, (6) will not be satisfied and the *Instantaneously* modality will not be declared, correctly. Hence, no false positives will occur. □

Fig. 5 depicts the results of Theorems 1 and 2. The dashed lines are indicative and do not imply a linear relation.

### 6.2 Detection Algorithm Using Scalar Strobes

The algorithm, implementing (6), is a modification of Algorithm 1. The changes are shown in Algorithm 2. Vector $M$ is scalar $Min\_Finish$; lines (1, 3) update scalar clocks; lines (17, 19) use scalar tests; lines (27-33) that delete intervals get replaced by the scalar analogs in lines (27-30).

**Algorithm 2** Scalar strobe algorithm at $P_i$ to detect predicates. The changes from Algorithm 1 are shown.

```
int: $Min\_Finish$

When event $e = (P_i, a_i, val, C)$ occurs at $P_i$:
(1) $C_i = C_i + 1$

On $P_i$ receiving event strobe $e = (z, a, val, C)$ from $P_z$:
3)  Execute SSC2 (Section 4.2)

17)                 if $(X.C_f < Y.C_s)$ then

19)                 if $(Y.C_f < X.C_s)$ then

27)         $Min\_finish = \min_k((head(IQ[k]).C_f)$
28)         for $k = 1$ to $n$
29)             if $head(IQ[k]).C_f = Min\_finish$ then
30)                 $dequeue(head(IQ[k]))$
```

### 6.3 Complexity

To find the first solution that satisfies (6), time overhead till each queue $IQ[k]$ has a head element is $O(n^2)$. For each set of intervals that satisfy (6), $O(n)$ time is needed to eliminate at least one interval. Such a set needs to be considered at most $np$ times. The time complexity is $O(np(n + O(f(\phi))))$, or simply $O(n^2p)$, to find all solutions.

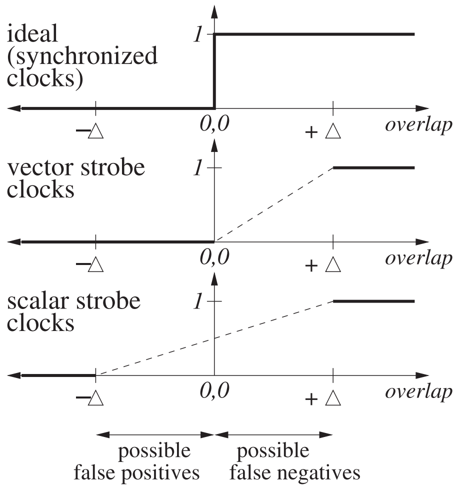


Fig. 5. Probability of detecting overlap of intervals for the *Instantaneously* modality.

The message cost and analysis are the same as in Section 5.4, except that the strobe size is $O(1)$ instead of $O(n)$.

## 7 Physical Time and Logical Time Modalities

Two logical time modalities have been traditionally defined under which a predicate can hold for distributed executions [6].

**Definition 3 (Logical time modalities) [6].**

- $Possibly(\phi)$. *There exists a consistent observation of the execution such that* $\phi$ *holds in a global state of that execution.*
- $Definitely(\phi)$. *For every consistent observation of the execution, there exists a global state of it in which* $\phi$ *holds.*

Our algorithm uses the logic of the logical time modality *Definitely* to detect the physical time modality *Instantaneously*. In this section, we analyze the types of errors that occur due to this approximation.

The false negative outcomes in the vector strobe algorithm are the cases of (13), where it is impossible to determine whether an overlap in the considered $\mathcal{I}$ actually occurred because of races. There are at most $pn$ such instances because only $pn$ states occurred in the actual execution. These cases *may* occur only if $0 < overlap < \Delta$ and can be flagged as *potential* false negatives by classifying them in a new bin: *borderline* [21]. This bin would also include some of those nonoverlap cases with $-\Delta < overlap < 0$, which could equally have been overlapping; unfortunately, there are $O(p^n)$ such states in the borderline bin, most of which never occurred. They can be flagged by enumerating the state lattice. Although impractical despite the *slim lattice* postulate, enumerating them is useful to: 1) not miss any borderline case and let the application decide whether to raise an alarm; 2) identify collections of states which collectively can ascertain that a relational predicate held. (See the example at the end of Section 5.3).

**Definition 4 (Physical time modalities on overlap in $\mathcal{I}$).** *For any (global state identified by)* $\mathcal{I}$ *in the actual execution that occurred, there are three modalities on its observation:*

- $Instantaneously^{\mathcal{I}}$. *The intervals in* $\mathcal{I}$ *overlapped (as defined in Definition 1). There are* $np$ *such states.*
- $Def^{\mathcal{I}}$. *It can be verified that the intervals in* $\mathcal{I}$ *overlapped. There are at most* $np$ *such states.*
- $Poss^{\mathcal{I}}$. *Intervals in* $\mathcal{I}$ *can be verified to have overlapped, or they might have overlapped but it is impossible to ascertain. There are* $O(p^n)$ *such states.*

$Def^{\mathcal{I}}$ means the intervals overlapped and it is possible to *verify* or ascertain. This case is ascertained when only the relationships from (12) hold between every pair the intervals in $\mathcal{I}$. This is what the vector strobe algorithm detects; see the mapping from the relationships from (12) to the test in [16].

In the example execution of Fig. 1, let $X$ and $Y$ denote the intervals at $P_i$ and $P_j$, respectively. Then, for example, for $x = 4 \wedge y = 5$, we have the relationship $IM(X, Y)$ from (12).

$Poss^{\mathcal{I}} \wedge \overline{Def^{\mathcal{I}}}$ means the intervals might or might not have overlapped and it is impossible to ascertain. This case occurs when at least one of the relationships from (13) holds between at least one pair of the intervals in $\mathcal{I}$ and none of the relationships $IA$ or $IQ$ holds between any of the pairs of intervals in $\mathcal{I}$ (see [16]).

With respect to the execution in Fig. 1, let $X$ and $Y$ denote the intervals at $P_i$ and $P_j$, respectively. Then, for example,

- for $x = 4 \wedge y = 3$, we have relationship $IC(X, Y)$;
- for $x = 2 \wedge y = 5$, we have $IR(X, Y)$;
- for $x = 6 \wedge y = 1$, we have $IK(X, Y)$; and
- for $x = 4 \wedge y = 1$, we have $IB(X, Y)$.

Such cases can be classified in the borderline bin. There are $O(p^n)$ such states because it requires examining all the states of the lattice of CGSs that even do not lie on the actual path traced through it.

Also, $\overline{Poss^{\mathcal{I}}}$ means we can ascertain with certainty that the intervals did not overlap. This case is ascertained when of the relationships $IA$ or $IQ$ holds between at least one of the intervals in $\mathcal{I}$. There are $O(p^n)$ such states, and these are not even represented in the lattice of CGSs.

**Theorem 3.** *For observing physical world phenomenon:*

$$Def^{\mathcal{I}} \Longrightarrow Instantaneously^{\mathcal{I}} \Longrightarrow Poss^{\mathcal{I}}, \tag{14}$$

$$Poss^{\mathcal{I}} \not\Longrightarrow Instantaneously^{\mathcal{I}} \not\Longrightarrow Def^{\mathcal{I}}. \tag{15}$$

**Proof.** Follows from Definition 4. □

Let $Set\_Instantaneously^{\mathcal{I}}$, $Set\_Def^{\mathcal{I}}$, and $Set\_Poss^{\mathcal{I}}$ denote the sets of sets of intervals satisfying the respective physical time modalities. Theorem 3 is illustrated in Fig. 6. The maximum numbers of solution sets that satisfy the three modalities are indicated in the Venn Diagram.

**Definition 5 (Physical time modalities under which predicate holds).**

- $Instantaneously^{\mathcal{I}}(\phi)$. *In the actual execution that occurred, the intervals in* $\mathcal{I}$ *overlapped and the value of* $\phi$ *evaluated over* $\mathcal{I}$ *is true.*

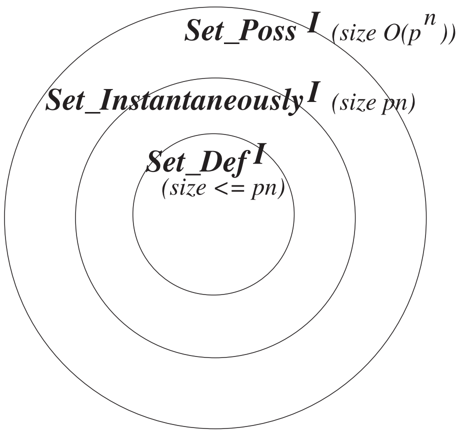


Fig. 6. Hierarchy of $Def^{\mathcal{I}}$, $Instantaneously^{\mathcal{I}}$, and $Poss^{\mathcal{I}}$ physical time modalities on overlap in $\mathcal{I}$.

- *$Def^{\mathcal{I}}(\phi)$. In the actual execution that occurred, it can be verified that the intervals in $\mathcal{I}$ overlapped, and the value of $\phi$ evaluated over $\mathcal{I}$ is true.*
- *$Poss^{\mathcal{I}}(\phi)$. In the actual execution that occurred, it can be verified that the intervals in $\mathcal{I}$ overlapped or they might have overlapped but it is impossible to ascertain, and the value of $\phi$ evaluated over $\mathcal{I}$ is true.*

The vector strobe algorithm detects $Def^{\mathcal{I}}$ and then evaluates $\phi$. Thus, it evaluates $Def^{\mathcal{I}}(\phi)$ as an approximation to $Instantaneously^{\mathcal{I}}(\phi)$. Up to the $np$ instances of $Def^{\mathcal{I}}$ can be detected in polynomial time $O(n^3p)$.

In contrast, to enumerate the *borderline* bin, it is necessary to evaluate $Poss^{\mathcal{I}}(\phi)$. Even for a conjunctive predicate, this requires exponential time because of the $O(p^n)$ instances of $Poss^{\mathcal{I}}$ that occur in the lattice of CGSs.

The condition tested in (3) was used for testing for the *Definitely* modality [6] for *conjunctive* predicates [8], [15], [20]. Informally writing, $Def^{\mathcal{I}}$ and $Poss^{\mathcal{I}}$ can be viewed as counterparts of *Definitely* and *Possibly* modalities [6] defined on predicates over executions of distributed programs, with different semantics. We now relate the physical time modalities to the logical time modalities.

**Theorem 4.** *For the lattice of consistent global states induced by the artificial strobes,*

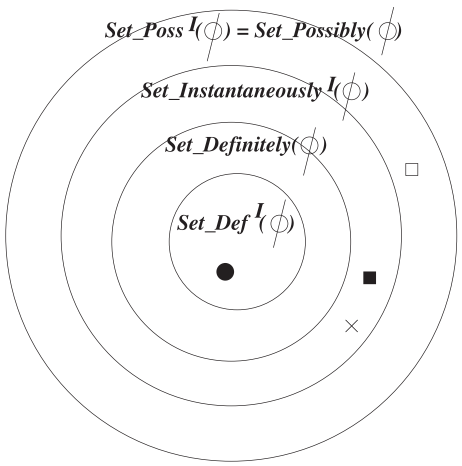
$$Def^{\mathcal{I}}(\phi) \Longrightarrow Definitely(\phi) \Longrightarrow Instantaneously^{\mathcal{I}}(\phi) \Longrightarrow Poss^{\mathcal{I}}(\phi) = Possibly(\phi). \tag{16}$$

**Proof.**

1. $Def^{\mathcal{I}}(\phi) \Longrightarrow Definitely(\phi)$. The right-hand side (R.H.S.) follows from Definitions 4 and 5.

   For conjunctive predicates, $Def^{\mathcal{I}}(\phi) \Longleftrightarrow Definitely(\phi)$ because the left-hand side (L.H.S.) was shown in [16].

   Note that for relational predicates, the *Definitely* modality does not imply there is a $\mathcal{I}$ over which the predicate $\phi$ holds. See the counterexample using Fig. 1 discussed at the end of Section 5.3. $Def^{\mathcal{I}}(\phi)$ serves as an approximation to $Definitely(\phi)$, determining which requires evaluating $Poss^{\mathcal{I}}(\phi)$ for all states, i.e., lattice evaluation.
2. $Definitely(\phi) \Longrightarrow Instantaneously^{\mathcal{I}}(\phi)$. For a conjunctive $\phi$, this follows trivially from Definition 4. For a relational $\phi$, in every execution, there is some set $\mathcal{I}$ of intervals which overlap, leading to the R.H.S. being true, although the $\mathcal{I}$ is different in different executions.
3. $Instantaneously^{\mathcal{I}}(\phi) \Longrightarrow Poss^{\mathcal{I}}(\phi)$. Follows from Theorem 3 for a conjunctive predicate. Definition of $Instantaneously^{\mathcal{I}}$ is independent of predicate type. So if a relational predicate is true over a single $\mathcal{I}$, the R.H.S. also holds.
4. $Poss^{\mathcal{I}}(\phi) = Possibly(\phi)$. We note that $Poss^{\mathcal{I}}(\phi) = Possibly(\phi)$ for both relational predicates and conjunctive predicates because both require only a single state in the state lattice in which $\phi$ holds. □



● *detected by vector strobe algorithm (for conjunctive and relational predicates)*
■ *actual physical time modality to be detected*
× *potential false negative*
□ *borderline bin entry may be false positive*

Fig. 7. Hierarchy of physical time modalities and logical time modalities on a predicate $\phi$.

Let

$$Set\_Def^{\mathcal{I}}(\phi), Set\_Definitely(\phi),$$
$$Set\_Instantaneously^{\mathcal{I}}(\phi), Set\_Poss^{\mathcal{I}}(\phi),$$

and $Set\_Possibly(\phi)$ denote the sets of predicates that are true under the corresponding modalities. Fig. 7 gives the hierarchy of modalities on the predicate by relating the physical time modalities of Definition 4 to the logical time modalities of Definition 3 using a Venn diagram. For a conjunctive predicate, the inner two circles are equal in size.

In summary, the vector strobe algorithm evaluates $Def^{\mathcal{I}}(\phi)$ as an approximation to $Instantaneously^{\mathcal{I}}(\phi)$:

1. *Conjunctive predicate.* $Def^{\mathcal{I}}(\phi) = Definitely(\phi)$ and is the best possible approximation to

$$Instantaneously^{\mathcal{I}}(\phi).$$

TABLE 1
Comparison of Middleware Strobe Clock Based and Synchronized (Physical) Clock-Based Predicate Detection Algorithms to Sense Physical World Properties

| | Synchronized (physical) clock algo. | Vector strobe clock algo. | Scalar strobe clock algo. |
|---|---|---|---|
| Lower layer messaging for clock synchronization | ongoing cost of synchronization protocol | none (zero cost) | none (zero cost) |
| Middleware layer messages for clock synchronization | none (zero cost) | one $O(n)$ broadcast per sensed event | one $O(1)$ broadcast per sensed event |
| Middleware layer messages for reporting events | one $O(1)$ message to sink, per sensed event (i.e., one broadcast in small wireless network) | free (piggyback on strobe) | free (piggyback on strobe) |
| Time complexity to evaluate all occurrences of $\phi$ | $O(n^2p)$ | $O(n^3p)$ | $O(n^2p)$ |
| False positives | none | none | if $-\Delta < overlap < 0$, these may occur |
| False negatives | if overlap period $\leq 2\epsilon$, always | if $0 < overlap < \Delta$, these may occur | if $0 < overlap < \Delta$, these may occur |
| Distributed predicate detection by all sensors | one $O(1)$ broadcast instead of message to sink, per sensed event | free (no messaging cost) | free (no messaging cost) |

2. *Relational predicate.* $Def^{\mathcal{I}}(\phi) \Longrightarrow Definitely(\phi)$ and is the best approximation to $Instantaneously^{\mathcal{I}}(\phi)$ that can be made without increasing the $O(n^3p)$ complexity. To use the approximation of $Definitely(\phi)$ would give greater accuracy but that requires evaluating $\phi$ over the lattice of CGSs.

Algorithm 3 modifies the vector strobe algorithm (Algorithm 1) to classify some of the (false negative) states encountered, in the borderline bin, without increasing the $O(n^3p)$ time. The bin would also include some true negative states having $-\Delta < overlap < 0$ that are encountered. Some false negatives will still persist as we are not generating the state lattice. The modification is based on detecting $Poss^{\mathcal{I}}$ in lines 17-20; and then checking for $Def^{\mathcal{I}}$ in line 26. If $Def^{\mathcal{I}}$ does not hold, then the detection is classified in the borderline bin.

**Algorithm 3** Changes to vector strobe algorithm (Algorithm 1) to use borderline bin while retaining polynomial complexity.

```
On $P_i$ receiving event strobe $e = (z, a, val, C)$ from $P_z$:

17)                         if ($X.C_f[h] \leq Y.C_s[h]$) then

19)                         if ($Y.C_f[j] \leq X.C_s[j]$) then

24)             // queue heads satisfy
                // $\forall h \forall j, I_h.C_f[h] > I_j.C_s[h]$
25)             if $\phi((\forall k) head(IQ[k]).val)$ is true then
26)                 if $X.C_f[j] \geq Y.C_s[j] \wedge$
                    $Y.C_f[h] \geq X.C_s[h]$ then
27)                     raise alarm/update world image
28)                 else classify in borderline bin
```

## 8 Discussion

Table 1 compares predicate detection using the proposed middleware layer strobe clocks versus using synchronized clocks from a lower layer [25]. The algorithms also allow detection to be done by all observers at no additional messaging cost. Results are observer-independent. The algorithms provide all nodes the choice to run the algorithm to know the outcome or to implement a rotating sink. To implement a rotating sink, all nonsink nodes execute only lines 1-3 in the predicate detection algorithms proposed.

The algorithms also have the following properties. If communication were unreliable by way of message loss, a lost message may lead to a wrong inference in its temporal vicinity, but it has no ripple effect on future detection. A crash failure of a sensor node impacts the future detection of those predicates whose values depend on the variables reported by the crashed node.

Recall that even with synchronized physical clocks, one has to cope with false negatives due to skew (of the order of microsecs or even millisecs using software protocols [32]), when there are "races" [25]. In contrast, the bound $\Delta$ is of the order of hundreds of millisecs to secs in small-scale networks, for example, smart offices and smart homes. However, middleware clocks are a viable option for sensing in small-scale networks and pervasive environments, because of the following:

1. The additional message cost, over reporting sensed events to a sink, is relatively low (see Table 1 and Sections 5.4 and 6.3);
2. Occurrence of false negatives is low when $n$ is low and/or the sensed event rate is low with respect to $\Delta$. This is the case for environments such as office, home, habitat, wildlife, nature, and structure monitoring:
    a. For example, in urban settings such as smart homes and smart offices, the number of sensors is typically low.
    b. Furthermore, lifeform and physical object movements are typically much slower than $\Delta$. And in the wild, remote terrain, nature monitoring, events are often rare, compared to $\Delta$.

   Thus, we may not need the precision (in urban settings or the wild) or be able to afford the associated cost (in the wild) of synchronized physical clocks.
3. Simulations in related work [12] to detect $Definitely(\phi)$ for a conjunctive $\phi$ in a realistic model of a smart office showed that despite increasing

the average message delay over a large range, the probability of correct detection is quite high. The simulations were backed by an analytical model with supporting numerical results [12].

4. Middleware clocks avoid the cost of and dependence on synchronized physical clocks in areas where they are not available or not affordable. They also give layer-independence.
5. They reduce the security issues and privacy concerns that come into play when physical clock synchronization is performed.

## REFERENCES

[1] Y. Bu, T. Gu, X. Tao, J. Li, S. Chen, and J. Lu, "Managing Quality of Context in Pervasive Computing," *Proc. Int'l Conf. Quality Software,* pp. 193-200, 2006.

[2] Y. Bu, S. Chen, J. Li, X. Tao, and J. Lu, "Context Consistency Management Using Ontology Based Model," *Proc. Int'l Conf. Current Trends in Database Technology,* pp. 741-755, 2006.

[3] R. Cardell-Oliver, M. Renolds, and M. Kranz, "A Space and Time Requirements Logic for Sensor Networks," *Proc. Second Int'l Symp. Leveraging Applications of Formal Methods, Verification, and Validation,* pp. 283-289, 2006.

[4] P. Chandra and A.D. Kshemkalyani, "Causality-Based Predicate Detection across Space and Time," *IEEE Trans. Computers,* vol. 54, no. 11, pp. 1438-1453, Nov. 2005.

[5] K.M. Chandy and L. Lamport, "Distributed Snapshots: Determining Global States of Distributed Systems," *ACM Trans. Computer Systems,* vol. 3, no. 1, pp. 63-75, 1985.

[6] R. Cooper and K. Marzullo, "Consistent Detection of Global Predicates," *Proc. ACM/ONR Workshop Parallel and Distributed Debugging,* pp. 163-173, May 1991.

[7] C. Fidge, "Timestamps in Message-Passing Systems that Preserve Partial Ordering," *Australian Computer Science Comm.,* vol. 10, no. 1, pp. 56-66, 1988.

[8] V.K. Garg and B. Waldecker, "Detection of Strong Unstable Predicates in Distributed Programs," *IEEE Trans. Parallel and Distributed Systems,* vol. 7, no. 12, pp. 1323-1333, Dec. 1996.

[9] V.K. Garg and B. Waldecker, "Detection of Weak Unstable Predicates in Distributed Programs," *IEEE Trans. Parallel and Distributed Systems,* vol. 5, no. 3, pp. 299-307, Mar. 1994.

[10] K. Henricksen and J. Indulska, "A Software Engineering Framework for Context-Aware Pervasive Computing," *Proc. IEEE Int'l Conf. Pervasive Computing and Comm. (Percom),* pp. 77-86, 2004.

[11] P. Hu, J. Indulska, and R. Robinson, "An Autonomic Context Management System for Pervasive Computing," *Proc. IEEE Int'l Conf. Pervasive Computing and Comm. (Percom),* pp. 213-223, 2008.

[12] Y. Huang, X. Ma, J. Cao, X. Tao, and J. Lu, "Concurrent Event Detection for Asynchronous Consistency Checking of Pervasive Context," *Proc. IEEE Int'l Conf. Pervasive Computing and Comm. (Percom),* 2009.

[13] L. Kaveti, S. Pulluri, and G. Singh, "Event Ordering in Pervasive Sensor Networks," *Proc. IEEE Int'l Conf. Pervasive Computing and Comm. Workshops (Percom),* 2009.

[14] A. Khelil, F. Shaikh, B. Ayari, and N. Suri, "MWM: A Map-Based World Model for Wireless Sensor Networks," *Proc. Second Int'l Conf. Autonomic Computing and Comm. Systems (Autonomics),* 2008.

[15] A.D. Kshemkalyani, "Temporal Interactions of Intervals in Distributed Systems," *J. Computer and System Sciences,* vol. 52, no. 2, pp. 287-298, Apr. 1996.

[16] A.D. Kshemkalyani, "A Fine-Grained Modality Classification for Global Predicates," *IEEE Trans. Parallel and Distributed Systems,* vol. 14, no. 8, pp. 807-816, Aug. 2003.

[17] A.D. Kshemkalyani, "Temporal Predicate Detection Using Synchronized Clocks," *IEEE Trans. Computers,* vol. 56, no. 11, pp. 1578-1584, Nov. 2007.

[18] A.D. Kshemkalyani and M. Singhal, *Distributed Computing: Principles, Algorithms, and Systems.* Cambridge Univ. Press, 2008.

[19] A.D. Kshemkalyani, "Middleware Clocks for Sensing the Physical World," *Proc Int'l Workshop Middleware Tools, Services, and Run-Time Support for Sensor Networks (MidSens '10),* pp. 15-21, 2010.

[20] A.D. Kshemkalyani, "Repeated Detection of Conjunctive Predicates in Distributed Executions," *Information Processing Letters,* vol. 111, no. 9, pp. 447-452, Apr. 2011.

[21] A.D. Kshemkalyani, "Immediate Detection of Predicates in Pervasive Environments," *J. Parallel and Distributed Computing,* vol. 72, no. 2, pp. 219-230, Feb. 2012.

[22] A.D. Kshemkalyani, A. Khokhar, and M. Shen, "Execution and Time Models for Pervasive Sensor Networks," *Int'l J. Networking and Computation,* vol. 2, no. 1, pp. 2-17, 2012.

[23] L. Lamport, "Time, Clocks, and the Ordering of Events in a Distributed System," *Comm. ACM,* vol. 21, no. 7, pp. 558-565, July 1978.

[24] F. Mattern, "Virtual Time and Global States of Distributed Systems," *Proc. Int'l Workshop Parallel and Distributed Algorithms,* pp. 215-226, 1989.

[25] J. Mayo and P. Kearns, "Global Predicates in Rough Real Time," *Proc. IEEE Symp. Parallel and Distributed Processing,* pp. 17-24, 1995.

[26] P. Pietzuch, B. Shand, and J. Bacon, "Composite Event Detection as a Generic Middleware Extension," *IEEE Network,* vol. 18, no. 1, pp. 44-55, Jan./Feb. 2004.

[27] M. Raynal and M. Singhal, "Logical Time: Capturing Causality in Distributed Systems," *Computer,* vol. 29, no. 2, pp. 49-56, Feb. 1996.

[28] M. Roman, C. Hess, R. Cerqeira, A. Ranganathan, R.H. Campbell, and K. Nahrstedt, "A Middleware Infrastructure for Active Spaces," *IEEE Pervasive Computing,* vol. 1, no. 4, pp. 74-83, Oct.-Dec. 2002.

[29] K. Romer and F. Mattern, "Event-Based Systems for Detecting Real-World States with Sensor Networks: A Critical Analysis," *Proc. DEST Workshop Signal Processing in Wireless Sensor Networks ISSNIP,* pp. 389-395, 2004.

[30] M. Sama, D.S. Rosenblum, Z. Wang, and S. Elbaum, "Model-Based Fault Detection in Context-Aware Adaptive Applications," *Proc. 16th ACM SIGSOFT Int'l Symp. Foundations Software Eng. (SIGSOFT '08/FSE-16),* pp. 261-271, 2008.

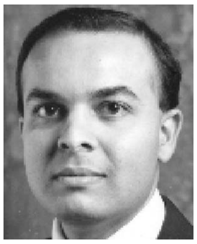
[31] S. Stoller, "Detecting Global Predicates in Distributed Systems with Clocks," *Distributed Computing,* vol. 13, pp. 85-98, 2000.

[32] B. Sundararaman, U. Buy, and A.D. Kshemkalyani, "Clock Synchronization for Wireless Sensor Networks: A Survey," *Ad-Hoc Networks,* vol. 3, no. 3, pp. 281-323, May 2005.

[33] C. Xu, S. Cheung, W. Chan, and C. Ye, "Partial Constraint Checking for Context Consistency in Pervasive Computing," *ACM Trans. Software Eng. Methodologies,* vol. 19, no. 3, pp. 1-61, 2010.

[34] C. Xu and S.C. Cheung, "Inconsistency Detection and Resolution for Context-Aware Middleware Support," *Proc. ACM SIGSOFT Int'l Symp. Foundations Software Eng.,* pp. 336-345, 2005.

[35] C. Xu, S.C. Cheung, W. Chan, and C. Ye, "Heuristics-Based Strategies for Resolving Context Inconsistencies in Pervasive Computing Applications," *Proc. IEEE 28th Int'l Conf. Distributed Computing Systems,* pp. 713-721, 2008.



**Ajay D. Kshemkalyani** received the BTech degree in computer science and engineering from the Indian Institute of Technology, Bombay, in 1987, and the MS and PhD degrees in computer and information science from The Ohio State University in 1988 and 1991, respectively. He spent six years at IBM Research Triangle Park working on various aspects of computer networks, before joining academia. He is currently a professor in the Department of Computer Science at the University of Illinois at Chicago. His research interests include distributed computing, distributed algorithms, computer networks, and concurrent systems. In 1999, he received the US National Science Foundation Career Award. He previously served on the editorial board of the Elsevier journal *Computer Networks,* and is currently an editor of the *IEEE Transactions on Parallel and Distributed Systems*. He has coauthored a book entitled *Distributed Computing: Principles, Algorithms, and Systems* (Cambridge University Press, 2008). He is a distinguished scientist of the ACM and a senior member of the IEEE.

**Jiannong Cao** received the BSc degree in computer science from Nanjing University, Nanjing, China, in 1982, and the MSc and PhD degrees in computer science from Washington State University, Pullman, in 1986 and 1990, respectively. He is currently a professor in the Department of Computing at Hong Kong Polytechnic University, Hong Kong. Before joining Hong Kong Polytechnic University, he was on the faculty of computer science at James Cook University and the University of Adelaide in Australia, and City University of Hong Kong. His research interests include parallel and distributed computing, networking, mobile and wireless computing, fault tolerance, and distributed software architecture. He has published more than 200 technical papers in the above areas. His recent research has focused on mobile and pervasive computing systems, developing testbed, protocols, middleware, and applications. He has served as a member of editorial boards of several international journals, a reviewer for international journals/conference proceedings, and also as an organizing/program committee member for many international conferences. He is a senior member of the China Computer Federation, a senior member of the IEEE, including the IEEE Computer Society and the IEEE Communication Society, and a member of the ACM. He is also a member of the IEEE Technical Committee on Distributed Processing, IEEE Technical Committee on Parallel Processing, IEEE Technical Committee on Fault Tolerant Computing.