Middleware Clocks for Sensing the Physical World

Ajay D. Kshemkalyani
University of Illinois at Chicago
ajay@uic.edu

ABSTRACT

An important task in sensor networks is to sense locally to detect global properties that hold at some instant in physical time. 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.

Categories and Subject Descriptors

C.2.4 [**Distributed systems**]: Distributed applications

General Terms

Theory, Design, Performance

Keywords

sensor networks, predicate detection, pervasive computing

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. The monitoring is achieved via tracking a time-dependent image of the (spatio-temporal activities in the) physical world. Evaluating predicates on that image is an important problem [16]. 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 is the "instantaneous" snapshot of the variables; although more complex timing relations exist, e.g., [9, 17, 18].

In this paper, we make the following contributions.

1. We propose a new 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. They are useful to observe system state in physical time at run-time. We use strobe clocks to evaluate predicates [2] under a *physical time* modality specification (*Instantaneous*) 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 [2] 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 [2], 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.

2. 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 [6]. 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 a asynchronous message-passing distributed execution (see [10]). We assume reliable FIFO communication and failure-free execution. Each sensor/ actuator is modeled as a pro-

MidSens'10, November 30, 2010 Bangalore, India.

cess $P_i(i \in [1 \dots n])$; 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. 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 mimic the communication among the entities of the world plane. Such messages attempt to capture the "true causality" [13] among the events in the world plane, to re-create the time-varying spatio-temporal 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 [2, 4] or a relational expression [2], 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 Motivation: 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, e.g., [19], 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. It also imposes a skew ϵ 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 2ϵ [15]. 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.

Problem: Given a conjunctive or a relational predicate ϕ on sensed attribute values of the world plane, detect *each* occurrence of ϕ 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 ϕ 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. TIME MODELS FOR SENSORNETS

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 [3, 14]. 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.

For traditional distributed program executions, the global state lattice [2, 14] 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 re-run, concurrent events may be reordered, leading to a different path in the state lattice.) But in sensornets, the physical world does not admit re-runs, and there are many *non-deterministic 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, that 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 consistent global states.

4. STROBE LOGICAL CLOCKS

Define Δ to be the bound on the asynchronous message transmission delay for a system-wide broadcast. Δ includes the delays for queuing in local incoming and outgoing buffers, 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, Δ 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, Δ 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.

4.1 Strobe Vector Clocks

A strobe vector clock $C_i[1..n]$ at process i consists of n integers. The *protocol* is given by rules SVC1 and SVC2.

SVC1. When process i executes (senses) a relevant event:
$C_i[i] = C_i[i] + 1$
System-wide_Broadcast (C)

SVC2. When process i receives a strobe T:
$(k \in N)\ C_i[k] = \max(C_i[k], T[k])$

4.2 Strobe Scalar Clocks

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

SSC1. When process i executes (senses) a relevant event:
$C_i = C_i + 1$
System-wide_Broadcast (C)

SSC2. When process i receives a strobe T:
$C_i = \max(C_i, T)$

It is weaker than the strobe vector clock but is lightweight (strobe size is $O(1)$, not $O(n)$) and it can be used to solve our problem under certain conditions.

4.3 Features

Although similar to the causality-based Mattern/Fidge vector clocks [3, 14], and Lamport scalar clocks [13], or "interval vector clocks" [1], there are differences. (1) Strobe clocks track the progress of 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, timestamps 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 values is sent on and only on computation messages so that the exact replica of the partial order is created. (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.

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 strobe vector 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 ϕ holds whenever the global state changes. Using strobe clock values to timestamp relevant events and hence intervals, we are using monotonic logical timestamps, 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. *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. $overlap(\mathcal{I}) = \min_i(I_i.t_f) - \max_i(I_i.t_s)$

Condition in Equation 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: 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: 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 Equation 8 holds.

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

Replacing $\leq$ by $<$ in the test of Equation 6 gives:

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

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

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

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. The control messages for the strobe clock create artificial causal dependencies which 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 consistent states in the sub-lattice 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, resulting in fat lattices, clock strobes get sent at each value change. This gives us the "*slim lattice postulate*" for consistent global states in sensornet observations. For these states, due to the inherent transmission delays, it is theoretically possible either to verify that the intervals overlapped, or to only suspect that they overlapped. But it is not possible to ensure polynomial time overhead.

5. DETECTION USING STROBE VECTORS

5.1 Analysis

To *characterize* the degree of imprecision in Equation 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 Figure 1 [7]. 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 past and future of the events $\min(X)$ and $\max(X)$, (induced by strobes). The distinction between those positions of Y with two labels each is analyzed in [7].

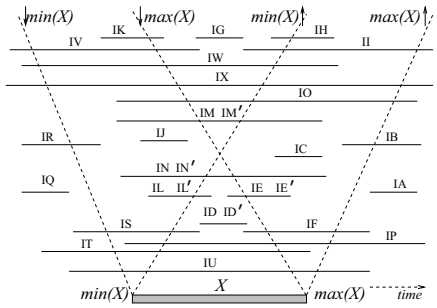

Figure 1: Timing diagram for complete set of orthogonal interaction types between intervals [7].

Of the 29, the following 18 logical placements [8]:

$$ID, IX, ID', IU, IE, IW, IE', IT, IF,$$
$$IS, IO, IL, IP, IL', IM, IM', IN, IN' \qquad (12)$$

satisfy Equation 3. Using Equations 4 and 2, observe that these placements imply physical time modality *Instantaneously*. If our algorithm evaluates the predicate only when these modalities in Equation 12 are satisfied among the intervals in $\mathcal{I}$, the algorithm will not detect any false positives.

Of the 29, the following 9 logical placements [8]:

$$IB, IR, IC, IV, IG, IH, IK, II, IJ \qquad (13)$$

may or may not overlap in physical time, and it is theoretically impossible using logical clocks and asynchronous unbounded transmission delays to determine the *Instantaneously* physical time modality specification in these cases. With multiple processes, even if intervals at one pair of processes are related by one of these 9 placements, to not raise false alarms, the algorithm should not detect *Instantaneously* even if the intervals happen to overlap. These are *potential* false negative cases. (If need be, these cases can also be detected; see the algorithms in [1] and Section 7.) The extent of false negatives will depend on the duration of intervals, their placements, and the bound Δ.

THEOREM 1. *Using vector strobe clocks, the correctness of detecting the* Instantaneously *modality using Equation 3 is characterized as follows.*

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

5.2 Algorithm

The algorithm uses logical time 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 (i) pairwise, they satisfy one of the placements in Equation 12, and hence from Equations 3 and 4, satisfy Equation 2; and (ii) ϕ is true over the attribute values in these intervals. The algorithm is given in Figure 2. Lines (8)-(22) are based on [1, 4].

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

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 ϕ 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 ϕ as strobes are generated [11].

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 consistent global states which cannot be done in polynomial time. This is discussed further in Section 7.

5.3 Complexity

To find the first solution that satisfies Equation 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 Equation 3, $O(n^2)$ time is needed to eliminate at least one interval; and let $O(f(\phi))(= \Omega(n))$ be the time to evaluate ϕ. 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 syn-*

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: $M[1 \ldots n]$

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 process P_z:
(3) Execute SVC2 (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) **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/actuate/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) $dequeue(head(IQ[k]))$

Figure 2: Vector strobe algorithm at P_i to detect predicates.

chronized physical clocks. However, if a multi-hop 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 assuming a tree or a linear array configuration, respectively, to reach the sink. 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 STROBE SCALARS

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

6.1 Analysis

Potential false positives: Consider the example execution in Figure 3, showing three intervals X, Y, and Z, at processes P_i, P_j, and P_k, respectively. (Assume all clocks are initially 0.) The beginning of each interval I is timestamped $I.C_s$ by the strobe scalar clock rules, SSC1 and SSC2. The end of the interval is timestamped $I.C_f$ by the start time of

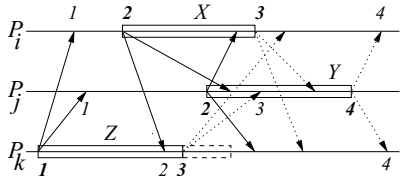

Figure 3: False positives with scalar strobes.

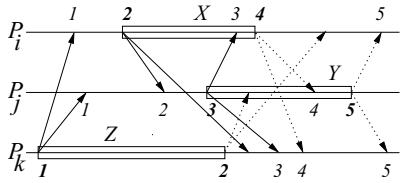

Figure 4: False negatives with scalar strobes.

the next interval at that process. Messages in regular lines are the strobe scalars 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. Timestamps in bold are the start and end timestamps of the intervals.

By applying the test of Equation 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 strobe scalar clocks would remain the same and the detection would have been a true positive.

Potential false negatives: Consider the example execution in Figure 4. By applying the test of Equation 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 strobe scalar clocks would remain the same and the non-satisfaction of Equation 6 would have been a true negative.

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

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

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

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

6.2 Algorithm

The algorithm, implementing Equation 6, is a modification of Figure 2. The changes are shown in Figure 6. 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).

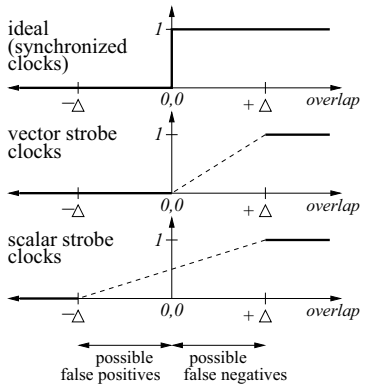

Figure 5: Probability of detecting overlap of intervals for the *Instantaneously* modality.

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 process 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) $dequeue(head(IQ[k]))$

Figure 6: Scalar strobe algorithm at P_i to detect predicates. The changes from Figure 2 are shown.

6.3 Complexity

To find the first solution that satisfies Equation 6, time overhead till each queue $IQ[k]$ has a head element is $O(n^2)$. For each set of intervals that satisfy Equation 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.

The message cost and analysis are the same as in Section 5.3, except that the strobe size is $O(1)$ instead of $O(n)$.

7. DISCUSSION

Table 1 compares predicate detection using the proposed middleware layer strobe clocks versus using synchronized clocks from a lower layer [15]. The algorithms also allow detection to be done by all observers at no additional messaging cost; results are observer-independent. They provide all nodes the choice to run the algorithm to know the outcome or to implement a rotating sink.

The false negative states in the vector strobe algorithm are the cases of Equation 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*. This bin would also include some of those non-overlap 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: (i) not miss any borderline case and let the application decide whether to raise an alarm; (ii) identify collections of states which collectively can ascertain that a relational predicate held. (See Section 5.2).

DEFINITION 3. *For any (state identified by) $\mathcal{I}$, there are three modalities on its observation:*

[$Instantaneously^{\mathcal{I}}$:] the intervals in $\mathcal{I}$ overlapped. 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.

Then, $Poss^{\mathcal{I}} \wedge \overline{Def^{\mathcal{I}}}$ means the intervals might or might not have overlapped and it is impossible to ascertain. This is the borderline bin. There are $O(p^n)$ such states. Also, $\overline{Poss^{\mathcal{I}}}$ means we can ascertain with certainty that the intervals did not overlap. There are $O(p^n)$ such states.

THEOREM 3. *For observing physical world phenomenon:*

$$Def^{\mathcal{I}} \Longrightarrow Instantaneously^{\mathcal{I}} \Longrightarrow Poss^{\mathcal{I}} \quad (14)$$

$$Poss^{\mathcal{I}} \not\Longrightarrow Instantaneously^{\mathcal{I}} \not\Longrightarrow Def^{\mathcal{I}} \quad (15)$$

The vector stobe algorithm detects $Def^{\mathcal{I}}$ and then evaluates ϕ. The condition tested in Equation 3 was used for testing for the *Definitely* modality [2] for *conjunctive* predicates [4, 7]. Informally writing, $Def^{\mathcal{I}}$ and $Poss^{\mathcal{I}}$ can be viewed as counterparts of $Definitely$ and $Possibly$ modalities [2] defined on predicates over executions of distributed programs, with different semantics. For the lattice of consistent global states induced by the strobes,

$$\phi(\mathcal{I}|Def^{\mathcal{I}}) \Rightarrow Definitely(\phi) \Rightarrow \phi(\mathcal{I}|Instantaneously^{\mathcal{I}}) \Rightarrow \phi(\mathcal{I}|Poss^{\mathcal{I}}) = Possibly(\phi) \quad (16)$$

For conjunctive predicates, $\phi(\mathcal{I}|Def^{\mathcal{I}}) \Leftrightarrow Definitely(\phi)$; for relational predicates, $\phi(\mathcal{I}|Def^{\mathcal{I}})$ serves as an approximation to $Definitely(\phi)$, determining which requires evaluating $\phi(\mathcal{I}|Poss^{\mathcal{I}})$ for all states, i.e., lattice evaluation.

The algorithms generate and evaluate a polynomial number of states, n^2p. These are good approximations to the actual np states that did occur; hence more likely candidates for the set of states that did occur. Figure 7 modifies the strobe vector algorithm (Figure 2) 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.

Recall that even with synchronized physical clocks, one has to cope with false negatives and false positives due to skew (of the order of microsecs or even millisecs using software protocols [19]), when there are "races" [15]. Hardware solutions can achieve skews of the order of nanosecs but are not practical in sensornets. In contrast, the bound Δ is of the order of hundreds of millisecs to secs in small-scale networks, e.g., smart offices and smart homes. However, middleware clocks may still be an option for sensing in small-scale networks and pervasive environments, because

Table 1: Middleware strobe clocks and synchronized (physical) clocks to sense physical world properties.

	Synchronized (physical) clock algo	Strobe vector clock algo	Strobe scalar 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 cost to evaluate ϕ	$O(n)$ per sensed event	$O(n^2)$ per sensed event	$O(n)$ per sensed event
false positives	none	none	if $-\Delta < overlap < 0$, may occur
false negatives	if overlap period $\leq 2\epsilon$, always	if $0 < overlap < \Delta$, may occur	if $0 < overlap < \Delta$, 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)

On P_i receiving event strobe $e = (z, a, val, C)$ from process 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/actuate/update world image
(28) **else** classify in borderline bin

Figure 7: Changes to vector strobe algorithm (Figure 2) to use borderline bin.

1. the additional message cost, over reporting sensed events to a sink, is relatively low (see Sections 5.3,6.3);
2. occurrence of false negatives may be low when n is low and/or the sensed event rate is low w.r.t. Δ (typical with human and object movements). Simulations in related work [5] to detect $Definitely(\phi)$ for a conjunctive ϕ 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 [5].
3. they avoid the cost of and dependence on synchronized physical clocks (giving layer-independence).

Our algorithms wait for an interval to complete before it is processed. So there may be a delay in detecting a predicate. Three algorithms [12] to detect the predicate immediately are compared with our algorithms, in [12].

8. REFERENCES

[1] P. Chandra, A. D. Kshemkalyani, Causality-based predicate detection across space and time. *IEEE Transactions on Computers*, 54(11): 1438-1453, 2005.

[2] R. Cooper, K. Marzullo, Consistent detection of global predicates. In *Proc. ACM/ONR Workshop on Parallel and Distributed Debugging*, 163-173, May 1991.

[3] C. Fidge, Timestamps in message-passing systems that preserve partial ordering. *Australian Computer Science Communications*, 10(1): 56-66, 1988.

[4] V. K. Garg, B. Waldecker, Detection of strong unstable predicates in distributed programs. *IEEE Trans. Parallel & Distributed Systems,* 7(12):1323-1333, 1996.

[5] Y. Huang, X. Ma, J. Cao, X. Tao, J. Lu, Concurrent event detection for asynchronous consistency checking of pervasive context. In *IEEE International Conference on Pervasive Computing and Communications*, 2009.

[6] L. Kaveti, S. Pulluri, G. Singh, Event ordering in pervasive sensor networks. In *IEEE Int. Conf. on Pervasive Computing and Comm. Workshops*, 2009.

[7] A.D. Kshemkalyani, Temporal interactions of intervals in distributed systems. *Journal of Computer and System Sciences*, 52(2): 287-298, April 1996.

[8] A. D. Kshemkalyani, A fine-grained modality classification for global predicates. *IEEE Trans. Parallel and Distributed Systems*, 14(8): 807-816, August 2003.

[9] A.D. Kshemkalyani, Temporal predicate detection using synchronized clocks. *IEEE Transactions on Computers*, 56(11): 1578-1584, November 2007.

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

[11] A.D. Kshemkalyani, Repeated detection of conjunctive predicates in distributed executions. *manuscript*, 2010.

[12] A.D. Kshemkalyani, Immediate detection of predicates in pervasive environments. In *9th International Workshop on Adaptive and Reflective Middleware (ARM'10)*, ACM Press, 2010.

[13] L. Lamport, Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7): 558-565, July 1978.

[14] F. Mattern, Virtual time and global states of distributed systems. In: *Parallel and Distributed Algorithms*, North-Holland, pp 215-226, 1989.

[15] J. Mayo, P. Kearns, Global predicates in rough real time. In *IEEE Symp. on Parallel and Distributed Processing*, 17-24, 1995.

[16] L. Mottola, G.P. Picco, Programming wireless sensor networks: fundamental concepts and state of the art. *ACM Computing Surveys*, 2010.

[17] P. Pietzuch, B. Shand, J. Bacon, Composite event detection as a generic middleware extension. *IEEE Network*, 18(1): 44-55, 2004.

[18] K. Romer, F. Mattern, Event-based systems for detecting real-world states with sensor networks: a critical analysis. In *DEST Workshop on Signal Processing in Wireless Sensor Networks at ISSNIP*, pp. 389-395, 2004.

[19] B. Sundararaman, U. Buy, A. D. Kshemkalyani, Clock synchronization for wireless sensor networks: a survey. *Ad-Hoc Networks*, 3(3): 281-323, May 2005.