State-Filters for Enhanced Filtering in Sensor-based Publish/Subscribe Systems

Salman Taherian and Jean Bacon
Computer Laboratory
University of Cambridge
CB3 0FD, UK
Email: {Salman.Taherian, Jean.Bacon}@cl.cam.ac.uk

***Abstract*—Publish/Subscribe systems have been extensively studied in the context of distributed information-based systems, and have proven scalable in information-dissemination for many distributed applications that have motivated the research. With the emergence of sensor-based applications and sensor networks, researchers have proposed novel publish/subscribe protocols that address the problem of distributed event dissemination for sensor network characteristics and constraints. In this paper, we focus on primitive events and the emerging class of publishers, and argue for "State-Filters" as more useful and suitable means of filtering events (than content-based filtering) in sensor-based publish/subscribe systems. Using State-Filters, we claim to achieve higher efficiency by means of filtering redundant and correlated event notifications, suppress event duplicates, and capture lasting conditions that had been previously not possible using content-based filters. We evaluate our proposed filtering mechanism using real-world sensor data, and highlight some assumptions and pitfalls that motivate our future work in this area.**

I. Introduction

Publish/subscribe systems have been extensively studied in the context of information-based systems. They provide efficient and scalable means of information dissemination in large-scale networks, where many information producers (publishers) and consumers (subscribers) are involved. Their decoupling of event clients is suitable for sensor network environments, where data is prioritized over the identity of its producer, and interests are expressed in terms of topic or content of data as opposed to their publishers' identifiers or addresses. Where transparency of event clients is supported, publishers' join or leave operations are handled autonomously, such that the existing subscribers need not resubscribe or refresh their subscriptions in order to receive event notifications from the newly joined publisher clients.

Researchers have extended the publish/subscribe communication paradigm over wireless sensor networks[1][2][3][4]. Sensor devices are viewed as event publishers, and user clients (or sink nodes) as event subscribers. Novel routing and event dissemination protocols have been proposed to address the constraints and characteristics of sensor networks[3][2]. In these works, content-based filtering has been investigated as a simple means of filtering events and reducing the communication costs within the network. Nevertheless, a large class of the emerging publishers (scalar sensors) publish events that are *periodic*, *high-rate*, and *correlated in time*.

Users are often interested in capturing real-world conditions from these events (as in [5] and [4]). While unwanted, unrelated events can be discarded using content-based filters, correlated and redundant event notifications can not. In this paper, we present a state-based filtering mechanism that offers the following features.

- It **filters correlated and redundant** event notifications, that are of little interest to the subscribers. This also allows sensors to publish events more frequently, so as to enhance the accuracy of condition capturing without affecting efficiency or overheads.
- State-Filters can capture conditions that are **long-lasting**. Content-based filters can only capture momentary conditions.
- State-Filters can **scope the realization of a condition**. For example, they can **suppress duplicates**, in a setting where redundant sensors are deployed to allow for sensor device vulnerabilities.

Also related to our work are Composite Event Detection (CED) frameworks [6][4][7] that can support similar features through complex event patterns and operators. Source-side filtering (as we shall see in section IV) is key to the reduction of communication costs in sensor networks. These frameworks, nevertheless, are designed around heavy-weight components where processing and memory resources are not a concern, e.g. active databases, EAI brokers. The most related sensor network CED frameworks are [5] and [4], both of which associate events with fixed validity intervals to support lasting conditions. This approach is evaluated (as an Enhanced Content-based Filtering scheme) in section V.

Content-based filters, however, have been widely used within sensor networks and placed on sensor devices (e.g. [3]). State-Filters match content-based filters in simplicity and operation. They impose additional memory overheads for the storage of an additional filtering expression and a single memory bit. Nevertheless, for this additional storage cost they offer features that had been previously impossible to achieve using content-based filters, and that save significant communication within the system.

II. Event Filtering

Event filtering is a simple, yet effective, means of reducing the number of event notifications that are disseminated within

the system. Subscriber-given expressions are used to empower the publish/subscribe system to filter event notifications[1].

Content-based filtering is the most common type of event filtering in publish/subscribe systems, resulting in a "content-based publish/subscribe system". It views each event notification message as a set of attribute/value pairs, whose values may be examined according to subscriber-given conditions. Conditions are boolean expressions, that should evaluate to true for the event notification to be forwarded to the corresponding subscriber(s).

Let us consider a temperature sensor for example, used as part of a smart environment. The sensor is configured to measure and report temperature values at 5 second intervals. The measurements are wrapped into event notifications with other related information (such as the time and the location of the reading) and dispatched into the publish/subscribe system for distribution to the related clients.

Using a content-based publish/subscribe system, a user can subscribe to events with temperature values greater than a threshold value (e.g. 25°C, to be told when the environment has become warm). The user, nevertheless, is not only notified when this threshold is passed but is also subjected to a series of subsequent event notifications that continue for as long as the temperature value remains above 25°C. The problem relates to a *lasting real-world condition*, which we (in this case) refer to as "warmness". The problem disappears when the user-specified condition is evaluated to false. Nonetheless, a significant number of events may be relayed in this period, which are inherently correlated and potentially of little significance to the subscriber. With the proposition of some work (e.g. [5]) one can assign predefined validity intervals to the published events, such as to filter the subsequent event notifications that are published in this duration. This approach, however, is a simplification that leads to inaccuracies and is inappropriate in the context of generic sensor-based applications.

In addition, where redundant sensors are deployed to address vulnerabilities and support fault-tolerance, the filtering of duplicate events, across multiple event publishers, is not supported. This means that in addition to the highlighted redundant event notifications that relate to each individual scalar sensor node, there is a regional event notification redundancy issue that corresponds to the redundant deployment of sensors in an area. We assume that these redundant publishers introduce events with similar parameter values within their localized regions.

III. State-Filters

State-Filters provide a much more expressive and effective means of event filtering in sensor-based publish/subscribe systems. They evaluate events according to the subscriber-specified conditions, hence match content-based filters in simplicity and implementation. Nevertheless, State-Filters are designed around the notion of state, primarily to capture *lasting* conditions. Their effectiveness is realized through the *smaller number of event notifications* that are passed through the filter and the rise in the event notification's *informative value* within the system. State-Filters can be viewed as *transformation filters*, through which the scalar sensor's event notifications become discrete, uncorrelated, and highly informative about a condition.

[1]note that we consider the publish/subscribe system as a middleware layer that is not application-aware. Thus, duplicate suppression, such as that pursued in Directed Diffusion[8], can not be performed independently.

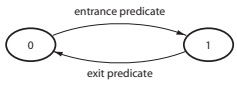

Fig. 1. Finite State Automata representation

State-Filters are expressed as follows.

State-Filter: [<entrance predicate> ; <exit predicate>]
Predicates are boolean expressions that examine user-defined conditions over the published events. Event parameters are used as operands, and logical ($\&\&, ||, !$), mathematical ($+, -, *, /, abs$), and comparative ($>, <, >=, <=, ==, !=$) operators are supported to examine relationships of interest over the event notification values.

A subscriber receives event notifications that successively match the entrance and exit predicates. Each State-Filter holds a *status-bit* that indicates its status (i.e. active or inactive). This also indicates the predicate that is used for evaluating each incoming event. When a predicate is satisfied, the event is passed through the filter and the status-bit is toggled (see figure 1).

Using State-Filters, users can subscribe to lasting conditions that are denoted by two explicit user-defined predicates. In the case of our earlier example, a user can subscribe to a *state of warmness*, as expressed below.

$warmness : [temperature > 25; temperature < 22]$
Unrelated entrance and exit conditions allow fine-grained specification and capture user interests in the system.

The user holds firm knowledge of the described state holding true for a continuous period that is bounded by two consecutive event notifications. Therefore, correlated events that relate to the same condition are filtered, and the user is no longer subjected to a series of redundant event notifications that follow the first satisfied event.

Using State-Filters, scalar sensors' events may be transformed into discrete events that are no longer periodic or correlated in time, but related to specific contexts or conditions. As such, users need not process all the received events to capture their conditions of interest. For example, one may subscribe to the *warmness* condition to feed the incoming events into a primitive actuator device (such as an air conditioning unit). In a reliable setting, the primitive actuator can simply toggle its operation based on the received events, without processing the contents.

A. Discrete Sensors

State-Filters can also capture *lasting* conditions, in the case of discrete sensors. These are often unattainable using content-based filters. Let's consider a building structure, equipped with distributed tag-reading sensors that identify nearby people and publish events including name (of the person identified) and location (of the identification/sensor). Now, if someone wishes to monitor the state of a person's (such as **John**'s) presence in a room (like **FE05**), then they may do so through the use of the following State-Filter expression.

$[(Name == \text{“}John'') \;\&\&\; (Location == \text{“}FE05'');$
$(Name == \text{“}John'') \;\&\&\; (Location \,!= \text{“}FE05'')]$

This captures the condition of interest, using simple identification sensors and without the complication of defining entrance and exit event topics that other related work depends upon. Primarily, this state detects **John**'s presence in the room when a sensor placed in the room makes such an observation, and concludes this presence when an outside sensor observes him. Note that the accuracy of this detection is dependent upon the distribution and density of the sensors placed in the building.

B. Momentary Conditions

Not all conditions of interest are long-lasting, some are momentary. The existing content-based publish/subscribe systems conceptually capture only momentary conditions that are tied to single event notifications. Such conditions can be expressed with State-Filters holding static "true" exit predicates. The "true" exit predicate results in an implicit exit transition which immediately follows after every state activation. This results in momentarily short state activations that correspond to momentary conditions. For example, a subscription to "all identifications of John in the building" can be expressed as $[Name == \text{“}John''; true]$, for which each event notification, that is not filtered, constitutes a momentary active state of "John's identification in the building". In turn, a static "false" exit predicate corresponds to a condition that upon detection is permanently valid (e.g. a failure detection state). Existing content-based filters can be transparently migrated into the State-Filter architecture using the static "true" exit predicate for each State-Filter expression.

IV. Distributed State-Filters

State-Filters provide the means of filtering and capturing user-interest conditions. However, the real benefits of communication and resource savings lie within the distribution and placement of these filters in the network. State-Filters are expressed by subscribers, as part of their subscription operation. A subscription operation often results in a path-establishment procedure that (indirectly) connects the subscriber to the relevant event publishers. The operation can be adopted from any existing content-based publish/subscribe system (see [9]), and has been omitted from the following discussions due to space limits.

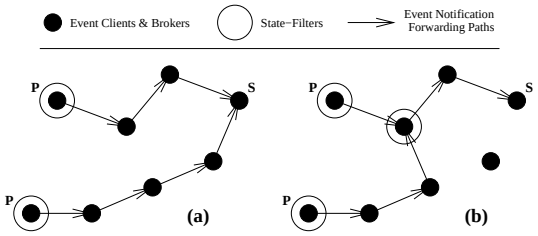

Fig. 2. (a) Source-side and (b) Scoped Filtering

A. Placement

Filters can theoretically be positioned anywhere along the event dissemination path, from the publisher's node (Source-Side Filtering, SSF) to the intermediate nodes (Intermediate-Node Filtering, INF) to the subscriber's node (Consumer-Side Filtering, CSF). Efficiency of the publish/subscribe protocol requires that the undesired events be filtered using as few resources as possible. This argues for source-side filtering, as undesired events are then filtered without any use of the communication resource (see figure 2(a)). Source-side filtering also enforces state detections over *totally ordered events* (because there is only one source, and no network propagation involved). Nevertheless, it only *captures conditions at the locality of the individual publishers*. Figure 2(a) shows how two publishers can forward events (that have passed through their source-side filters) to a subscriber.

B. Scoping

INF and CSF can *scope* the realization of user-interest conditions. The "John's presence in the room FE05" example, in section III, would use such a mechanism to capture the condition over an area that covers the room **FE05** and its surroundings. In the case of scalar sensors, this mechanism filters duplicate events that emerge from the redundant publishers in a scope (see figure 2(b)). Table I shows a summary of the placement options, the associated features and the supported coverage specifications.

Sensor Types	SSF	INF & CSF
Scalar	√ automated	Redundancy Scoping pre-defined + subscriber-specified
Discrete	√ subscriber-specified	State Detection Scoping subscriber-specified

TABLE I
State-Filter Placements

In scoping a detection, the State-Filter must be placed so as to *capture all event notifications published in the referenced scope*. This placement depends on the operation of the publish/subscribe protocol. In tree-based publish/subscribe systems, this implies that the subscriber-rooted tree must have a single event-forwarding branch covering the scope (see figure 2(b)). In cluster-based approaches, where brokers (cluster-heads) maintain local groups of publishers, a single cluster must cover the referenced scope for INF, or otherwise CSF

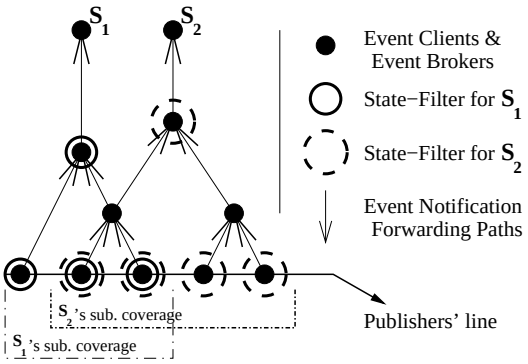

Fig. 3. Shared Event Dissemination Paths

must be used. In this study, we have implemented State-Filters over a tree-based publish/subscribe system.

C. Sharing

Subscribers, with the same (State-Filter) interests, can share events and filters. Where interests are not similar, they may share events and dissemination paths subject to the condition that *all related subscriber State-Filters are examined at each subscriber's State-Filter placement point*. Figure 3 shows an example (for tree-based publish/subscribe systems) in which two distinct State-Filter subscriptions share event dissemination paths. As shown, filters are co-located at all filtering points over the shared event dissemination paths.

D. Notification Forwarding

Event notifications are forwarded along event dissemination paths from the publishers to the subscribers. For scoped filtering, we support two event notification deliveries, *timely delivery* and *ordered delivery*. Where timeliness is important, events are processed according to a "first come first served" policy. Otherwise, events are buffered for finite durations and processed in the order of timestamps to support ordered delivery. The semantics of ordered delivery must be specified carefully. Events from different sources often cannot be ordered meaningfully using source timestamps because of clock drift. This may not be important for applications where causality is not an issue, but applications must be made aware that timestamp-based ordering is not precise.

Where multiple State-Filters are involved at an intermediate forwarding node, the event notification is evaluated against all the related (i.e. matching subscription coverage) State-Filters. An event notification is passed through a set of filters if it satisfies at least one state predicate.

V. Evaluation and Discussion

The performance and correctness of our proposed filtering mechanism was evaluated within a simulation environment. Use of real data in our evaluations dictated a specific application scenario which is outlined below.

A. Simulation Environment

The proposed framework has been implemented on Jist/Swans[10]. A two-dimensional outdoor environment was simulated, comprising sixteen equisize regions. Each region was allocated a temperature sensor that monitored the regional temperature. Regions were also equipped with a random number (between zero and two) of redundant sensors that reported on the same information, mainly for fault-tolerance purposes. The temperature sensors were programmed to report regional temperature values (in the form of event notifications) every three minutes. Additional wireless nodes were inserted into the environment to ensure network connectivity.

All simulated nodes used radio communications as a means of networking. Reliable MAC 802.11 was used for link-layer communications. GPSR[11], a geographical routing protocol, coupled with a tree-based publish/subscribe mechanism (similar to Directed Diffusion's one-phase pull protocol[12]), was used to interconnect the publishers and subscribers in a decentralized manner. Ten distributed subscriber nodes were simulated in the environment, with similar (but non-identical) interests over temperature changes. Subscribers wished to be notified when a certain threshold value has been exceeded in their chosen regions of interest. All threshold values were in the vicinity of 10°C, but different for each subscriber.

State-Filters (SF) were compared against an Enhanced Content-based Filtering (ECF) scheme. In ECF, satisfied events were given a fixed validity period of *thirty minutes*. Correlated events published within this validity interval were filtered by the ECF scheme. In the case of State-Filters, each outdoor region was also pre-defined as a redundancy scope, over which duplicate events were filtered by means of automated redundancy scoping. Simulation results, excluding failures and relating to thirty hours of real data gathered from outdoor sensors, are shown in table II.

Statistics	**SF**	**ECF**
Publishers	35	35
Subscribers	10	10
Subscriptions	10	10
Coverage Publishers	11	11
Source-Side Filters	27	27
Scoping Filters	10	N/A
Published Events	21000	21000
Covered Events	6600	6600
Delivered Events	20	620
Source-Side Filter P.R.	3.33e-3 (22#)	3.27e-2 (216#)
Duplicates Suppressed	14	0
Shared Events	16 (6)	192 (192)
Capturing Resolution	3mins	30mins

TABLE II
Simulation Results

B. Notification Filtering

From a total of 21000 published events in the system, only 6600 related to the subscribers' areas of interest. A condition capturing resolution of *three minutes* (in the case of State-

Filters) against the *thirty minutes* interval period of the ECF[2]. demonstrates the increased accuracy of condition capturing when using State-Filters. With ECF, a trade-off is realised between efficiency and the accuracy of condition capturing, where a larger validity interval increases the efficiency but compromises the accuracy by an even larger value.

A combined source-side filtering pass-ratio of 3.33e-3, corresponding to just 22 events (from a total of 6600) for State-Filters, compares to the 3.27e-2 of ECF. With nearly ten-fold higher source-side filtering and delivery of 600 fewer events to the subscribers than the content-based filtering scheme, State-Filters result in higher efficiency.

C. Messaging Costs

Table II shows that out of the 22 events (which passed through the source-filters), 14 events were further filtered at the redundancy scoping filters. The remaining 8 events were those which were disseminated to the ten distributed subscribers within the system. Content-based filters, with a lower source-side filtering and inability to filter duplicates, disseminated 216 events from the publishers to the subscribers.

D. Event Sharing

Similar subscription expressions were declared deliberately in order to observe event sharing among the subscribers. Prior to the duplicate suppressions, 16 events were shared for two or more subscribers in the SF scheme. This figure was lowered to 6 events following the duplicate suppressions. 192 events were shared in the ECF scheme.

VI. Future Work & Conclusions

In this paper, we presented State-Filters that extend content-based filters with capabilities to filter correlated and redundant event notifications across individual sensors as well as a group of redundant sensor deployments. These features were primarily motivated by the need to capture lasting conditions that the content-based filters were incapable of detecting. The proposed State-Filters accommodate the existing content-based filters, and match their simplicity in use and operation. In future work, we extend our evaluations over discrete sensors and examine fault-tolerance aspects of this approach. We also wish to address some imprecisions and uncertainties that may arise in the event notifications published by unreliable sensors in the system. Integration of primitive aggregation functions, as part of our filtering service, may prove useful in this respect.

Acknowledgment

This research was funded by a grant from *Microsoft Research Cambridge*.

[2]the largest observed inaccuracy with content-based filters in this experiment was 18 minutes.

References

[1] E. Souto, G. Guimares, G. Vasconcelos, M. Vieira, N. Rosa, and C. Ferraz, "A message-oriented middleware for sensor networks," in *Proceedings of the 2nd workshop on Middleware for pervasive and ad-hoc computing*. New York, NY, USA: ACM Press, 2004, pp. 127–134.

[2] P. Costa, G. P. Picco, and S. Rossetto, "Publish-Subscribe on Sensor Networks: A Semi-probabilistic Approach," in *Proceedings of the 2^{nd} IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS05)*, Washington DC, USA, Nov. 2005.

[3] C. P. Hall, A. Carzaniga, J. Rose, and A. L. Wolf, "A content-based networking protocol for sensor networks," Department of Computer Science, University of Colorado, Tech. Rep. CU-CS-979-04, Aug. 2004. [Online]. Available: http://serl.cs.colorado.edu/ carzanig/papers/

[4] A. V. U. P. Kumar, A. M. R. V, and D. Janakiram, "Distributed collaboration for event detection in wireless sensor networks," in *MPAC '05: Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing*. New York, NY, USA: ACM Press, 2005, pp. 1–8.

[5] S. Li, S. H. Son, and J. A. Stankovic, "Event detection services using data service middleware in distributed sensor networks." in *IPSN*, ser. Lecture Notes in Computer Science, F. Zhao and L. J. Guibas, Eds., vol. 2634. Springer, 2003, pp. 502–517.

[6] S. Courtenage, "Specifying and detecting composite events in content-based publish/subscribe systems," *icdcsw*, vol. 00, p. 602, 2002.

[7] P. R. Pietzuch, B. Shand, and J. Bacon, "Composite event detection as a generic middleware extension." *IEEE Network*, vol. 18, no. 1, pp. 44–55, 2004.

[8] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," *IEEE/ACM Trans. Netw.*, vol. 11, no. 1, pp. 2–16, 2003.

[9] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," *ACM Comput. Surv.*, vol. 35, no. 2, pp. 114–131, 2003.

[10] R. Barr, Z. J. Haas, and R. van Renesse, "Scalable wireless ad hoc network simulation," *Handbook on Theoretical and Algorithmic Aspect of Sensor, Ad hoc Wireless, and Peer-to-Peer Networks*, pp. 297–311, 2005.

[11] B. Karp and H. T. Kung, "GPSR: Greedy perimeter stateless routing for wireless networks," in *6th Annual International Conference on Mobile Computing and Networking*, Aug. 2000, pp. 243–254.

[12] J. Heidemann, F. Silva, and D. Estrin, "Matching data dissemination algorithms to application requirements," in *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*. New York, NY, USA: ACM Press, 2003, pp. 218–229.