Active Rules for Sensor Databases

M. Zoumboulakis, G. Roussos and A. Poulovassilis

Birkbeck University of London
Malet Street WC1 7HX
London UK
{mz, gr, ap}@dcs.bbk.ac.uk

Abstract

Recent years have witnessed a rapidly growing interest in query processing in sensor and actuator networks. This is mainly due to the increased awareness of query processing as the most appropriate computational paradigm for a wide range of sensor network applications, such as environmental monitoring. In this paper we propose a second database technology, namely active rules, that provides a natural computational paradigm for sensor network applications which require reactive behavior, such as security management and rapid forest fire response. Like query processing, efficient and effective active rule execution mechanisms have to address several technical challenges including language design, data aggregation, data verification, robustness under topology changes, routing, power management and many more. Nonetheless, active rules change the context and the requirements of these issues and hence a new set of solutions is appropriate. To this end, we outline the implications of active rules for sensor networks and contrast these against query processing. We then proceed to discuss work in progress carried out in project Asene that aims to effectively address these issues. Finally, we introduce our architecture for a decentralized event broker based on the publish/subscribe paradigm and our early design of an ECA language for sensor networks.

1 Introduction

Application development for sensor and actuator networks presents unique challenges since it has to address the complexities of distributed and often decentralized operation, the highly resource constrained nature of network nodes and the highly transient nature of network topology [4]. Moreover, applications must operate unattended for prolonged periods of time and still maintain their integrity and quality of service.

In recent years it has become clear that the investigation of higher level computational paradigms is necessary so as to abstract the complexity of systems development and offer application developers with a more amenable programming framework. To this end, query processing has attracted considerable interest and is rapidly becoming a popular computational paradigm for a plethora of sensor network applications. This approach has been seen to address well the complex requirements of application development in sensor networks in a variety of applications including environmental monitoring, distributed mapping and vehicle tracking [5, 12]. Prototype sensor network query processors have been implemented in Tiny DB [11] and Cougar [17] systems.

In this paper we argue that another database technology that may provide an appropriate computational model for a distinct set of sensor and actuator network applications is *event-condition-action (ECA) rules* [15] (also referred to as active rules). Indeed, sensor and actuator network applications often operate in one of either modes:

- in event-driven applications, for example detection of forest fires, security management or product detection in ubiquitous retailing [9], the system remains inactive until an event is generated in one of the nodes; then the event propagates through the system which subsequently initiates appropriate actions in response to this event,
- in demand-driven applications, for example environmental monitoring [5], activity is initiated in response to external requests, usually in the form of queries.

While query processing matches well the characteristics of the later class of applications, an ECA rule-

Proceedings of the First Workshop on Data Management for Sensor Networks (DMSN 2004), Toronto, Canada, August 30th, 2004.
http://db.cs.pitt.edu/dmsn04/

based approach offers a better fit for applications with execution profile that corresponds to the first pattern above. In such applications, the system needs to provide a timely response to events and although in principle this would still be possible using a sensor network query processor, its deployment would unnecessarily consume the limited resources by regularly checking for events that may not have occurred.

In the following Section we discuss ECA rules as a computational model for reactive systems with particular reference to sensor and actuator networks. We then proceed to compare more traditional ECA technologies with the novel needs of sensor networks. In Section 4, we discuss the requirements of ECA rules in this context and highlight the differences to the more well established sensor network query processors. Finally, we introduce the architecture of the Asene system for Active SEnsor NEtworks. We conclude with a discussion of work in progress and major challenges ahead.

2 Active Rules as a Model for Computation in Sensor Networks

We begin by examining in more detail the structure of a reactive sensor and actuator network application. A reactive application must be able to detect the occurrence of specific events or changes in the network state, and to respond by automatically executing the appropriate application logic. For example, in a security monitoring scenario sensors capable of detecting specific chemicals are deployed in the area under observation, for example a customs and excise enclosure in a port area. In addition to the sensor nodes, a smaller number of actuator nodes are also deployed with the capability to trigger alarms when activated. In this case, there is little scope for fixed network infrastructure due to the transient nature of most objects within the enclosure and the use of heavy machinery. The aim is to program the application so that when specific events are observed and specific conditions are met the network reacts in a predetermined way, for example when the concentration of particular chemical factors are observed and their concentration exceeds a set threshold within a small area the alarm in this and neighboring areas are activated.

ECA rules [15] are one way of implementing this kind of functionality. An ECA rule has the general syntax

```
on event
if condition
do actions
```

The *event* part specifies when the rule is triggered. The *condition* part is a query which determines if the sensor network is in a particular state, in which case the rule fires. The *action* part states the actions to be performed if the rule fires. A side effect of these actions may be that further events are generated, which may in turn cause more ECA rules to fire.

In sensor and actuator networks in particular the action part of an ECA rule may be either logical or physical. For example, the action may be to signal an actuator node to activate the alarm, or it may be a notification for a node to initiate a particular control sequence [16].

There are several advantages in using ECA rules to implement this kind of functionality compared to direct implementation in application code [2, 14]:

- ECA rules allow an application's reactive functionality to be specified and managed within a rule base rather than being encoded in diverse programs, thus enhancing the modularity, maintainability and extensibility of applications.
- ECA rules have a high-level, declarative syntax and are thus amenable to analysis and optimization techniques which cannot be easily applied if the same functionality is expressed directly in application code.
- ECA rules are a generic mechanism that can abstract a wide variety of reactive behaviors, in contrast to application code that is typically specialized to a particular kind of reactive scenario.

To illustrate the use of active rules to model reactive functionality we note that the application logic described at the beginning of this section could be encapsulated within the following rule

```
ON UPDATE toxicity
IF AVG(toxicity) > thres WITHIN radius r1
DO ACTIVATE alarm WITHIN radius r2
```

3 Sensor Networks and Traditional Active Database Systems

ECA rules in the context of a sensor and actuator network present a number of novel challenges against the traditional database view [8]. In traditional active database (and web-based) systems the condition and action parts of an ECA rule are most often tightly coupled, that is the execution model of a particular rule is [E][CA]: a database object is monitored and when modified in a predetermined way an event is generated. Rules whose event parts match this event are then triggered and, if their conditions hold, their actions are scheduled for execution. In all cases, execution of the condition query and the action part is driven by the same application logic. Hence, ECA functionality is tightly coupled and coordinated. Moreover, such systems are generally administered by database experts and often implement advanced failure-tolerance features, including clustering, power backups and replicated communication channels.

Sensor and actuator networks consist of a large number (often several hundreds) of loosely-coupled node elements [1]. Each node operates fairly independently and can make its own decisions about its wake-up/sleep cycle and the data it accepts to forward to nearby nodes. Nodes may also have different capabilities, for example sensors may be able to detect temperature, humidity, changes in the magnetic field of the Earth, different types of biosensing and so on. Actuators may be biomanipulators, microvalves and micropumps or they can simply be electrical switches. In addition to sensor and actuator nodes, nodes that have the sole purpose of providing communications and computational assistance may also be introduced in the system. In all cases, nodes will have high failure rates which may result in network fragmentation, that is the separation of network segments into isolated islands of system functionality.

Sensor and actuator networks are deployed in ad-hoc ways and thus the resulting topologies may be highly irregular and with highly heterogeneous density and connectivity patterns. Furthermore, the topology may often change rapidly during its pre-deployment, deployment, and re-deployment phases and possibly at very high speed. This is in stark contrast to traditional database management systems which assume that connectivity is fairly fixed and network topology is rarely of concern and dealt with outside the database management system.

Last but not least, sensor and actuator nodes are very limited in power, computational capability and holding capacity and are normally unavailable for regular repair or frequent battery recharge. Although Moore's law predicts that node capabilities will increase rapidly, they will always be less powerful than other embedded, portable or hand-held computing devices and most importantly battery power available for their operation will remain limited for the foreseeable future.

4 Challenges for Active Functionality in Sensor Networks

In this paper we propose that ECA rules can provide a natural computational paradigm to sensor and actuator network applications that require reactive behavior. While sensor network query processors (SNQP) [3, 5, 11] have proven very successful in providing appropriate abstractions for user interaction, ECA rules address the problem of unattended system behavior and can effectively model application logic in autonomic situations[1]. In the context of such applications, the system is required to provide a timely response to events at the lowest communications and computational cost. Although potentially a SNQP could be used for this type of application, in practice it would unnecessarily consume limited resources by regularly checking for events that may not have occurred. Indeed, SNQPs primarily address data acquisition from a relatively small number of vantage points. ECA rules may provide an effective and efficient mechanism to support reactive behavior by localizing control and by providing a mechanism to react to events rather than proactively test whether a particular event has occurred.

This difference in scope between SNQP and ECA rules implies that the two systems have very different execution profiles which also means that they also have very different requirements. In the following paragraphs we attempt to outline the most critical differences between the two approaches and in the following section we discuss our current work in trying to address the novel requirements of ECA execution within project Asene.

- **Vantage Points.** SNQPs assume that queries are initiated at a single or a relatively small number of vantage points, with data aggregation potentially carried out at a few intermediate locations, the so-called storage points. In ECA rules any sensor in the network may generate an event which may be used by any actuator also potentially placed at any network location. Thus, an ECA rule may fire at any node location within the network and may also activate any node within the network.

- **Communication Pattern.** SNQPs collect data in regular patterns which sensor nodes can use to synchronize and agree on wake-up/sleep cycles. ECA rules are reactive and thus rules fire at unpredictable, irregular intervals. Hence, wake-up/sleep schemes that can support this asynchronous mode of operation are required. Moreover, this irregular pattern implies that nodes consume power at different rates and for this reason node failure is more irregular and harder to predict.

- **Routing.** SNQPs currently mostly use tree-based routing mechanisms that flood the network at least once, during the tree construction stage. In this context the communications overhead placed by the route discovery stage is justified by the relatively large amount of data that is being collected. An ECA rule processor is characterized by small, incremental updates rather than a single data collection step and thus

[1]The scope of active functionality as described here should not be confused with the so-called event queries supported by Tiny DB. Event queries aim at providing user control over data acquisition so that users can register their interest for specific results returned by an acquisitional query and specify additional queries that should be carried out in response. Hence, supporting generic reactive functionality is well beyond the scope of event queries.

the route discovery stage of tree-based algorithms would dominate the communications cost. Consequently, globally optimal routes would probably not optimize power consumption for the network as a whole and localized routing algorithms could be more efficient [7].

- **Data Model.** SNQPs currently view the sensor network as a single data space. ECA rules require an alternative data model which distinguishes between the different types of objects that are being observed and generate events. In the following section we propose a mechanism for the construction of separate data spaces based on the so-called topic channels.

- **Aggregation.** Aggregation in ECA is carried out at the signal rather than the query layer which is the norm for SNQP. Although the mathematical techniques used for aggregation in SNQP [6] can also be used in ECA rule processing, this is done at a lower layer and within a particular topic channel in an approach akin to collaborative signal processing in distributed environments.

- **In-network storage.** Although both systems clearly benefit from in-network storage, SNQP develops hierarchical-directional mechanisms based on the tree-based routing algorithms employed, whereas ECA rules benefit from decentralized-flat and schemes at the topic channel level.

- **Network Segmentation.** ECA rules execute within the a specific network locality and thus can be relatively resistant to network segmentation for example due to loss of connectivity caused by intermediate node failure. ECA rules may still fire despite their isolation from a sink controller.

5 A System Architecture for ECA in Sensor Networks

One of the major challenges in implementing an ECA rule based architecture for sensor and actuator networks is the distribution of events in a computationally efficient manner. In this section we introduce the Asene approach to support ECA functionality in sensor and actuator networks.

Asene is built on top of *event channels* which are also viewed as data object primitives. An event channel has two elements: a collection of nodes that monitor the same attribute and associated algorithmic mechanisms that coordinate node operation. Within an event channel nodes carry out collaborative signal processing and data aggregation and are responsible for in-network storage and event generation. Finally, the node components of an event channel encapsulate internal structures that maintain shared descriptions of the channel.

Event channels are also responsible for the distribution of events following the so-called publish/subsribe (P/S) paradigm [13]. P/S systems are commonly used to bring together data sources and information consumers by transparently delivering events from the first to the second. In Asene, event channels are responsible for maintaining a list of subscribers to the particular event and for sending notifications. Thus, subscriber nodes may move freely and re-attach to the channel at alternative locations. Effectively, an event channel functions as a decentralised event broker following the P/S jargon.

The particular characteristics of sensor and actuator networks make them especially compatible with the P/S paradigm, in particular with regard to the need for in-network storage and processing:

- P/S systems are characterized by the same basic properties as sensor and actuator networks; that is, communication is anonymous, inherently asynchronous and multicasting in nature. P/S systems are also capable of quickly adapting to changing network topologies.

- P/S systems can support the decentralized operation of event management and delivery, transparently for both sensor and actuator nodes. This is particularly important since computation in sensor and actuator networks is highly asymmetric and thus local adaptability and local control is of great importance.

- The P/S anonymity property in particular implies that communicating nodes are not required to identify the party they wish to communicate with (that is, subscribers need only describe the characteristics of the events they want to receive instead of naming a specific publisher to receive events from) and thus data aggregation may be implemented transparently for the end application. Moreover, the anonymity property implies that flexile wake-up/sleep cycles can be developed since delivery of events to subscribed recipients does not depends on a single sensor node.

- Conceptually P/S systems deliver events to multicast groups, a communications mode that is a good fit for the provision of incremental updates to aggregation operators constructed on top of role-based spatial hierarchies of sensor and actuator networks nodes. The power saving potential of these multi-resolution data aggregation schemes can be considerable and more importantly their effectiveness increases rapidly with the number of nodes in the system. Moreover, it is possible to achieve relatively high performance by using the periodic beaconing performed of most medium access and topology control protocols for update delivery across a particular topic channel.

The properties that make P/S suitable for use in sensor and actuator networks also suggest a natural way to support node failures as a feature of the system rather than as a fault. Indeed, in this context data aggregation is performed independently by each node. Hence, loss of updates will affect accuracy locally and nodes will continue computation with whatever data available, on a best effort basis. This is a distinct advantage over techniques originating from more tightly coupled systems, where there would be a need for roll backs and data cleansing operations which are not appropriate in the case of sensor and actuator networks.

One of the expected advantages of this architecture is that it allows for complex wake/sleep schemes while at the same time maintaining a good quality of service via replication of the in-network stored data and of the subscription information.

The use of event channels as the core building block for Asene allows for the full decoupling of the [E], [C] and [A] components of ECA rules. Also note that queries associated with the condition part of an active rule can be answered locally and in some cases the data required could be disseminated at the same step as the event itself. It is also worth observing that new functionality can be introduced in the system via the simple insertion of new condition nodes, that is nodes that are responsible for checking for specific conditions in response to event notifications. Finally, constructing activation channels is also a viable alternative although often the expected number of actuator nodes would be much smaller than the number of sensors and it is probably not as cost efficient as an approach.

5.1 Heterogeneity

An interesting observation on the effects of the Asene architecture is that significant operational benefits may be achieved if heterogeneous sensor and actuator networks are constructed. Heterogeneity in this case is seen primarily in communication capability and in terms of the range of communication. Inserting a few nodes that have longer range capabilities (but also higher power consumption) can significantly increase the robustness of the event channel by increasing the connectivity across node clusters.

5.2 Composite Events

Using event channels as the main mechanism for data dissemination also suggests a clear way for constructing rules with composite events: the condition node needs only subscribe to all corresponding event channels. Compare this against the difficulty of dealing with multidimensional data in the context of SNQP.

6 Discussion and Conclusions

In this paper we have argued that, in addition to query processing, ECA rules is a database technology that may provide an appropriate computational model for a distinct set of sensor and actuator network applications. However, ECA rules in this context present several challenges which we highlighted in previous sections. We have also introduced Asene, an ongoing research project that aims to establish ECA rules as the common mechanism for the description of reactive functionality in sensor and actuator networks.

The current version of Asene supports simple event channels built on top of Tiny OS [10] primitives and a simple ECA language. We are currently developing further our algorithms for the efficient construction of event channels in sensor networks. Our focus is on a single-step approach that identifies all members of all registered event channels in a particular network and thus removes the need for duplication of the bootstrap phase. We are also improving on the data structures used to represent the internal state of a particular event channel and maintain the list of active subscriptions. Our work aims to balance the need for low communication between nodes and the asynchronous nature of event generation with regard to the wake-up/sleep node cycles. We are planning to conduct extensive experiments with the prototype implementation to better understand the tradeoffs involved.

In addition to the development of efficient and effective event channel management mechanisms, a second major objective of the Asene project is the definition of an appropriate lightweight ECA language that satisfies the requirements of the application domain. The brief example presented in Section 2 in the context of a security management application is taken from the current version of Asene. Clearly, further work in understanding the performance implications of the different constructs is required and balanced against language expressivity.

The next step for Asene is the integration of advanced aggregation algorithms and the study of localized routing algorithms for event dissemination. In doing so we favor a multi-resolution approach similar to the aggregation schemes discussed in [6] but more appropriate for the structure of our event channel construction algorithms. Finally, we intend to further investigate the relative merits of different routing strategies for event dissemination based on localized network descriptions. We anticipate both approaches to offer significant reduction in resource demands from the network.

References

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci. Wireless Sensor Networks: A Survey, *Computer Networks*, Vol. 38, pp. 393–422, 2002.

[2] J. Bailey, G. Papamarkos, A. Poulovassilis and P. T. Wood. An Event-Condition-Action Rule Lan-

guage for XML, in A. Poulovassilis and M. Levene (eds.) *Web Dynamics*, Springer-Verlag, to appear, 2004.

[3] P. Bonnet, J. Gehrke, and P. Seshadri. Towards Sensor Database Systems, *Proceedings of the Second International Conference on Mobile Data Management*, Hong Kong, 2001.

[4] D. Estrin, R. Govindan, J. Heidemann and S. Kumar. Next Century Challenges: Scalable Coordination in Sensor Networks, *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking*, Seattle, Washington, USA, pp. 263-270, 1999.

[5] J. Gehrke and S. Madden. Query Processing in Sensor Networks, *IEEE Pervasive Computing*, Vol. 3, No. 1, pp. 46-55, 2004.

[6] J. M. Hellerstein , W. Hong, S. Madden, K. Stanek. Beyond Average: Toward Sophisticated Sensing with Queries, F. Zhao and L. Guibas (eds.) *Proceedings of Second International Workshop Information Processing in Sensor Networks*, IPSN 2003, Palo Alto, CA, USA, April 22-23, pp. 63 - 79, 2003.

[7] A. Helmy. Location-free Contact Assisted Poer-Efficient Query Resolution for Sensor Networks, *Mobile COmputing and Comuunications Review*, Vol. 8, No. 1, pp. 27-47, 2004.

[8] K. Kulkarni, N. Mattos, and R. Cochrane. Active database features in SQL3, in N. Paton (ed.) *Active Rules in Database Systems*, Springer-Verlag, pp. 197-219, 1999.

[9] P. Kourouthanassis and G. Roussos. Developing Consumer-Friendly Pervasive Retail Systems, *IEEE Pervasive Computing*, Vol. 2, No. 2, pp. 32–39, 2003.

[10] P. Levis, S. Madden, D. Gay, J. Polastre, R. Szewczyk, A. Woo, E. Brewer and D. Culler. The Emergence of Networking Abstractions and Techniques in TinyOS, *Proceedings of the First USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI 2004)*, March 29-31, San Fransisco, CA, 2004.

[11] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. The Design of an Acquisitional Query Processor for Sensor Networks, *SIGMOD*, San Diego, CA, 2003.

[12] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. TAG: a tiny aggregation service for ad hoc sensor networks, in *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation*, 2002.

[13] C. Mascolo, L. Capra, and W. Emmerich. Middleware for Mobile Computing (A Survey), *Lecture Notes in Computer Science*, Vol. 2497, 2003.

[14] G. Papamarkos, A. Poulovassilis and P. T. Wood. Event-Condition-Action Rule Languages for the Semantic Web, *Proc. VLDB'03 Workshop on Semantic Web and Databases*, Berlin, September 2003.

[15] N. Paton and O. Diaz. Active Database Systems, *ACM Comp. Surveys*, Vol. 31, No. 1, pp. 63–103, 1999.

[16] Y. Yemini, A.V. Konstantinou and and D. Florissi. NESTOR: An Architecture for Self-Management and Organization, *IEEE Journal on Selected Areas in Communications*, Vol. 18, No. 5, pp. 758–766, 2000.

[17] Y. Yao and J. Gehrke. The Cougar Approach to In-Network Query Processing in Sensor Networks, *Sigmod Record*, Vol. 31, No. 3, 2001.