Component Oriented Middleware for Distributed Collaboration Event Detection in Wireless Sensor Networks

D. Janakiram
Distributed and Object Systems Lab, Department of CS and E
Indian Institute of Technology Madras
Chennai, India
d.janakiram@cs.iitm.ernet.in

A V U Phani Kumar
Distributed and Object Systems Lab, Department of CS and E
Indian Institute of Technology Madras
Chennai, India
phani@cs.iitm.ernet.in

Adi Mallikarjuna Reddy V
Distributed and Object Systems Lab, Department of CS and E
Indian Institute of Technology Madras
Chennai, India
adi@cs.iitm.ernet.in

ABSTRACT

With the advancement of technology in micro-electronics and wireless communication, small miniature devices called sensor nodes can be used to perform various tasks by forming wireless sensor network (WSN). In WSN, event detection is one of the main requirements for most of the applications. An event can be a simple event or a combination of two or more simple events (Composite Event). Detecting and reporting an event desired by the application (user) inspite of stringent constraints of sensor nodes like low energy, low bandwidth, frequent failures etc., is one of the main challenges in WSN. This can be achieved with less uncertainty and masking failures by considering collaboration among sensor nodes. We propose a framework for composite event detection using distributed collaboration in WSN. The framework consists of two protocols that build a tree by using a communication model similar to the Publish-Subscribe paradigm. This framework is a part of COMiS (*C*omponent *O*riented *Mi*ddleware for *S*ensor networks). COMiS is component oriented middleware which is developed in terms of components. These components are loaded as and when required based on the application semantics. If collaboration is considered, the goal of the application can be easily accomplished even in case of failures of sensors and low energy of nodes.

Keywords: Wireless Sensor Networks;Event of Interest; Collaboration; Middleware;Publish/Subscribe;Simple Event; Composite Event.

1. INTRODUCTION

WSN is a collection of large number of independent nodes called sensor nodes. These nodes have capabilities for sending, receiving and sensing. But they are constrained in their resources such as processing power, memory, bandwidth etc. The nodes are deployed in harsh and inaccessible terrains like deserts, oceans, forest etc. They have no fixed topology, but they can configure themselves to work in such conditions. WSN now a days, is used in a wide variety of applications ranging from smart home systems to battlefield applications [3, 14].

Detecting events such as fire and explosion can be accomplished using sensor networks. An event can be a simple (or atomic) or a composite event [12]. The events such as temperature > 90 or light > 50 come under simple events. Events like explosion detection which is a combination of two or more simple events come under composite event. These events are detected by taking readings from multiple sensors present on the sensor node. Simple events require only participation of single sensor (say, light sensor) and composite events require two or more sensors for their detection.

There has been a general assumption in the literature that every sensor node should have capabilities to detect all simple events that form a composite event [6, 15, 1]. Hence a sensor node can contribute to a composite event only if it possesses capabilities to detect all simple events. This is not valid under the following conditions:

Condition 1: Sensor nodes deployed in an environment are manufactured in such a way that different nodes have different sensing capabilities.

Condition 2: There might be a situation where sensors can fail.
e.g.Node A has light, humidity and temperature sensors. But Node B has light and humidity sensors because temperature sensor on B has failed.

Condition 3: Sensor nodes could have purposefully stopped some of their sensors due to energy constraints

Condition 4: Sensor node is unable to use some of its sensor data due to lack of memory for storing data

Collaboration is often required while considering the above conditions. Suppose a node does not have all sensors to detect a composite event or it is lacking energy to use its sensor. In such a case it should collaborate with other nodes to fulfill the goal. Collaboration has many other advantages like transmission scheduling [9] that is only certain nodes are scheduled to transmit depending on energy they possess, longevity of network life, complete and accurate collection of information [16], less uncertainty in collected data[16].

Collaboration among sensor nodes is considered in literature for target tracking [16], failure detection [4] etc. In [16] nodes in the neighborhood of a suspected node, collaborate for detecting a faulty node. In [4], a convoy tree based collaborative framework is proposed for target detection which achieves high tree coverage and low energy consumption. In [8] collaborative algorithms for communication are proposed at traditional layers like MAC, transport and routing. Middleware like [6, 15] assume that all sensor nodes in the network are homogeneous in sensing capabilities and can detect all simple events that form a composite event. Hence every node should contribute to the event. This is not the case under conditions mentioned above. So collaboration is required in such circumstances for event detection in sensor networks.

In order to accomplish event detection in such conditions, a framework has been proposed for event detection using collaboration. It constitutes two protocols that build a tree to detect an event. These protocols consist of two phases namely initialization phase and collection phase. This framework is part of middleware COMiS (Component Oriented Middleware for Wireless Sensor Networks) [2]. COMiS is developed in terms of components to satisfy the resource constraints such as memory and power. Different components are present on different nodes depending upon the functionality of node. Hence components are loaded based on application semantics. Currently this middleware does not support simple event detection under above conditions and composite event detection also.

The communication model that is used here is Publish/ Subscribe paradigm [13]. It is scalable and most popular for event based middleware [11]. The main advantage of this model is that it entirely decouples the publishers from the subscribers and communication is done in an asynchronous manner. Subject-based and Content-based paradigms are two types of Publish-Subscribe paradigms[13, 11].

In subject-based Publish/Subscribe, the interest is expressed over the topic and all events are informed related to topic. In content-based Publish/Subscribe interest is expressed as filter expression over an event and events which satisfy this expression are only informed. We followed content-based Publish/Subscribe as our communication model.

The rest of the paper is organized as follows. Section 2 explains the framework for event detection. The proposed communication model is discussed in section 3. Section 4 details the protocols in two phases. An example of the working of protocols is illustrated in section 5. Section 6 explains the background of COMiS and the integration of proposed framework with COMiS. Section 7 concludes the paper with an overview of future directions.

2. FRAMEWORK

In this section, a framework for composite event detection using collaboration is explained.

2.1 Assumptions

- The capabilities of the sensors can be heterogeneous, that is different sensors can have different sensing capabilities.
- Sensor nodes are immobile in the system, but the mobility of the user node is allowed.
- No failures occur in sensors once this framework is initiated.
- Wireless broadcast is used for communication
- The sensors are aware of their location and position.
- The basic system model assumed is shown in figure 1.

2.2 Framework for Event Detection

The existing framework for event detection in COMiS does not consider event detection under conditions of low energy of nodes and failures of sensors. The proposed framework which is a part of COMiS deals with both simple and composite event detection by considering collaboration of nodes and hence making it robust . This framework consists of two protocols called simple event detection protocol and composite event detection protocol. Our paper proposes protocols for simple and composite event detection. Each protocol works in two phases. One is initialization phase and another is collection phase.

2.2.1 Initialization Phase

In the initialization phase application submits event of interest to the sensor network. Here application acts as subscriber for the interested event. The region where the phenomenon to be detected is also specified by giving start and end coordinates of the region. An *event based tree* (EBT) is constructed based on these events. Publish/Subscribe like communication paradigm is followed in constructing this tree. The event of interest can be simple event or combination of simple events. Nodes which can generate data related to any simple event can join the tree. Generation of data depends on sensing capabilities of node. Thus, an EBT is constructed as required by the application. This tree can be shared by different applications if a simple event is present in more than one event of interest supplied by them.

2.2.2 Collection Phase

Collection phase deals with collecting results after an event has happened. The aggregation happens at each node depending on the event and the corresponding generated data.

3. COMMUNICATION MODEL

Content-based Publish/Subscribe is followed for protocols explained in section 4. This is complex to implement than subject-based paradigm but more expressive and scalable [11]. We could also use the subject-based paradigm in these protocols, but it consumes more energy in constrained environments such as WSN.

Application submits predicates over attributes in the form of $A_1 > value_1$, $A_2 > value_2$.....$A_n > value_n$ in any programming language like [7][15]. Here A_1, A_2.., A_n are low level sensing attributes like temperature, pressure etc. Subscription messages with the above mentioned predicates are generated by the middleware. Middleware also plays the role of event disseminator in WSN. After receiving subscription message, each node checks whether it has the capability to sense attributes specified in the message. If so, it makes an entry of the application identifier (id) along with parent id i.e node from which it received the message. Application id

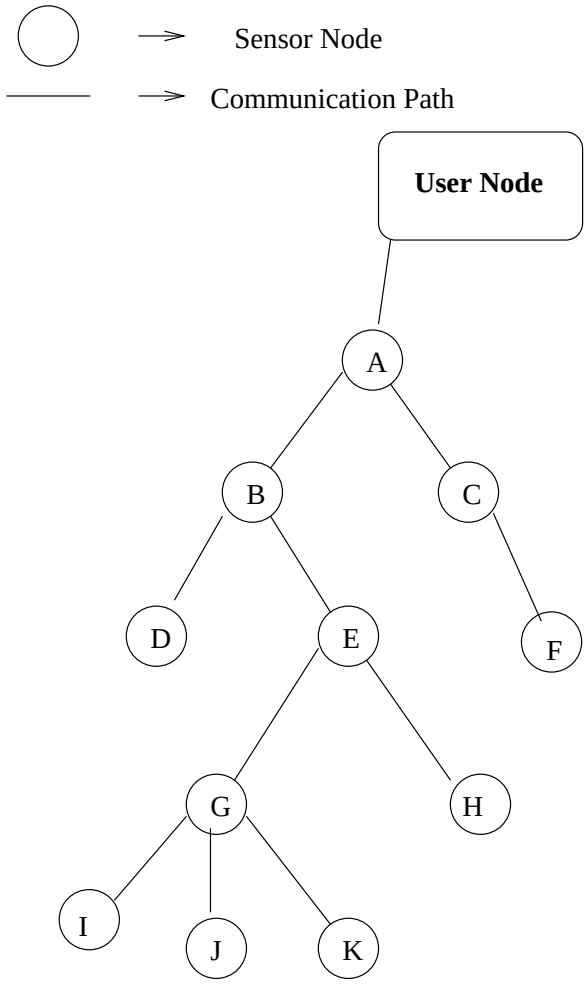

Figure 1: System Model

is maintained to distinguish between different subscriptions. The node that receives subscription from a user node will act as the sink and tree is constructed per subscription as shown in figure 1 with the sink as root. Here application acts as the subscriber. Multiple applications i.e multiple users subscribing from different locations, can share the tree if any simple events to be detected are common in specifications of event of interests.

Unsubscribe message can be sent by the middleware to stop subscription. This is sent at old location, if user moves to new location. New subscribe message is sent from new location. This is done by middleware transparently from application to allow results to be propagated through a new tree to new location of node. Hence *mobility of user node* is supported. Unsubscribe message can also be sent by a node with low energy due to the reason that it can no longer participate. This can be useful to adapt the tree according to dynamics of network. This is not explored in this paper and is considered as future work.

4. PROTOCOLS

4.1 Data Structures

- *StartRegion*: Starting point of region in co-ordinates.
- *EndRegion*: Ending point of region in co-ordinates.
- T(Event1, [Event2,....], Parent, ApplicationId) where *Event* is in the form of *SensingAttribute*>*Value*, For eg: Temperature>90.
- TopicOfInterest=(event1, event2.., eventN)
- M(TopicOfInterest, HopCount, StartRegion, EndRegion)
- CON(TopicOfInterest, HopCount)
- ValidRegion()[1] is dictated by the application and it gives the desired region in number of hops.
- D is the *delay* for which nodes have to wait to receive data from neighbors.
  Again this is application dependent and it is supplied by the application.
- EventCounter[$event_i$]
  The index corresponds to the event. The value of EventCounter[$event_i$] indicates how many nodes are currently detecting event i.
- Location(node) gives the position of the node in the sensor network
- THRC[Event] is the threshold of certainty of an event. It can be supplied either by the application or can be decided by the middleware.
- CONFIRMATION message is sent to nodes present at the top level in the hierarchy to indicate the partial set became complete set (all simple events that form composite event) and hence start sensing.

[1]desired region, region of interest are used interchangeably throughout this paper

4.2 Simple Event Detection Protocol

Here TopicOfInterest = (Event)

1. Initialization Phase

On receiving M
If (*Location(node)* is between *StartRegion* and *EndRegion*)
If capability matches with attribute of *M(TopicOfInterest)* and there is no entry in T
Eventcounter[Event]++;
If (*Eventcounter[Event]* $<$ *THRC[Event]*)
Add to T ,the parent, the event and application id
start sensing
Forward to neighbors

2. Collection Phase

On receiving data from local sensor
Eventcounter[Event]−−;
If (*Location(node)* $==$ *EndRegion*)
send to parent
Else
Wait for D sec;
On receiving data from neighboring nodes
If received data matches generated data
apply aggregation
If(*Eventcounter[Event]* $==$ *0*)
report the event
send to parent

4.3 Composite Event Detection Protocol

1. Initialization Phase

Let C be the composite event.
Let n be the number of simple events that forms C
TopicOfInterest = (event1,event2..,eventn)
A node on receiving M,
If (*Location(node)* is between *StartRegion* and *EndRegion*)
If this node capabilities matches with all attributes present in *M(TopicOfInterest)* then
make an entry in to T
Play aggregator role for all event data in *TopicOfInterest*
For *i=1* to n
If(*EventCounter[Event*$_i$*]* $<$ *THRC[Event*$_i$*]*)
EventCounter[Event$_i$*]++;*
Forward to neighbors
If this node capabilities matches with subset of attributes present in *M(TopicOfInterest)*
Let t out of n be the number of events that match.
Make an entry of matched subset of events along with parent and application id in to T
If (*M(HopCount)* $<$ *ValidRegion()*)
M(HopCount)++
For *i=1* to t
If(*EventCounter[Event*$_i$*]* $<$ *THRC[Event*$_i$*]*)
EventCounter[Event$_i$*]++*
Else If(*M(HopCount)* $==$ *ValidRegion()*)
If (*EventCounter[Event*$_i$*]* $<$ *THRC[Event*$_i$*]*)
Start Sensing the desired events.
Send *CONFIRMATION* message to the parent to indicate that partial set is complete.
CON(HopCount) $=$ *ValidRegion()*
M(HopCount) $=$ *0*
Forward to neighbors
On receiving *CONFIRMATION* message
If (*CON(HopCount)* $>$ *1* and *CON(HopCount)* $<$ *ValidRegion()*)
CON(HopCount)−−
Start sensing
Forward to parent
Else If (*CON(HopCount)* $==$ *1*)
Start sensing
Act as aggregator role for composite event

2. Collection Phase

On receiving data from local sensors
For i $=$ *1* to t
Eventcounter[Event$_i$*]−*
If (*Location(node)* $==$ *EndRegion*)
send to parent
Else
Wait for D sec;
On receiving data from neighboring nodes
If received data matches generated data for an simple event from 1 to t apply aggregation for that data
For i $=$ *1* to n
If (*Eventcounter[Event*$_i$*]* $==$ *0*)
report the event
Else
send it to parent

Explanation

Here the protocols (simple and composite event) works in two phases. One is initialization phase and another is collection phase. The assumption here is that sensor nodes are heterogeneous in sensing capabilities. The reason for this assumption is given as conditions in section 1.

Application submits the event to be reported in the form of a simple event. It should also supply start region and end region of location where event to be detected, number of hops and tolerable delay between events. *StartRegion* and *EndRegion* delimits the region of interest. The number of hops returned by *ValidRegion()* decides the number of groups to be formed in the desired region. This parameter is entirely dependent on application. For example, in target tracking application sensor nodes will form some number of groups in the region for detecting vehicle. One set of nodes will wake another set of nodes after detecting. In fire detection application only one group is formed in region of interest (assume very small area).

Assuming every sensor node is aware of its position, *Location(node)* gives position of node in the network. Using this, a node could check whether it is present in the desired region and hence decide whether to participate or not. *THRC* is the threshold of certainty. *THRC[i]* stores how many number of nodes are required to infer that really an event i has happened. If this exceeds in desired region the other nodes

may or may not participate depending on their energy requirements.

EventCounter is maintained to know how many nodes are currently detecting a particular event. Using this, if the number of nodes which are already scheduled to detect some event exceeds some threshold, the other nodes need not start detecting this particular event hence saving energy. For example, *EventCounter[temperature>70]=10* indicates ten nodes are currently monitoring temperature event. If any node recieves multiple *EventCounters* with different values it hold the maximum one but replies to all. Assume ten is the threshold in a region to indicate certainty that event really happened. So the other node need not start monitoring this and may contribute to other event. In this way nodes collaborate among themselves and contributes for event detection.

The communication model followed in the above protocols is content based Publish/Subscribe. If the *TopicOfInterest* are without any predicates then it becomes subject-based Publish/Subscribe. But it is more expensive in sensor network environment, because reporting happens whenever there is a change in the *TopicOfInterest*. But in case of content-based Publish/Subscribe, predicate is given as e.g., temperature>80, hence reporting happens only when temperature exceeds 80, but not whenever changes occur in temperature.

5. EXAMPLE

The protocols mentioned in section 4 are explained by taking explosion event as example in this section.
Application is interested in explosion event. This explosion event is combination of simple events like high temperature event, bright light event and loud acoustic event [12].
For eg., Explosion event is said to have occurred if Temperature > 80C (Centigrade) and light > 500L (lumen) and sound > 70D (decibels). This example is explained in two phases for composite event detection. Table 1 shows sensor nodes and their capabilities at some instance of time in the network. Figure 2 shows the example scenario. The status of event set is shown at each node. The √ mark that correspond to each event shows that the event can be detected by those set of nodes at that level.

Initialization
threshold=3
validregion()=3
Startregion=(0,0)
Endregion=(10,10)
D=2 millisec

Topic of interest=(Temperature > 80C, light > 50L(lumen), sound > 70D (decibels))
User is closer to node 1 and wants explosion event to be reported if the above conditions are satisfied.

Initialization phase
Broadcast down the Hierarchy: Broadcast 0: At node 1 : hopcount=0, Eventcounter[temp,sound,light]=[1,0,0]
Broadcast 1:

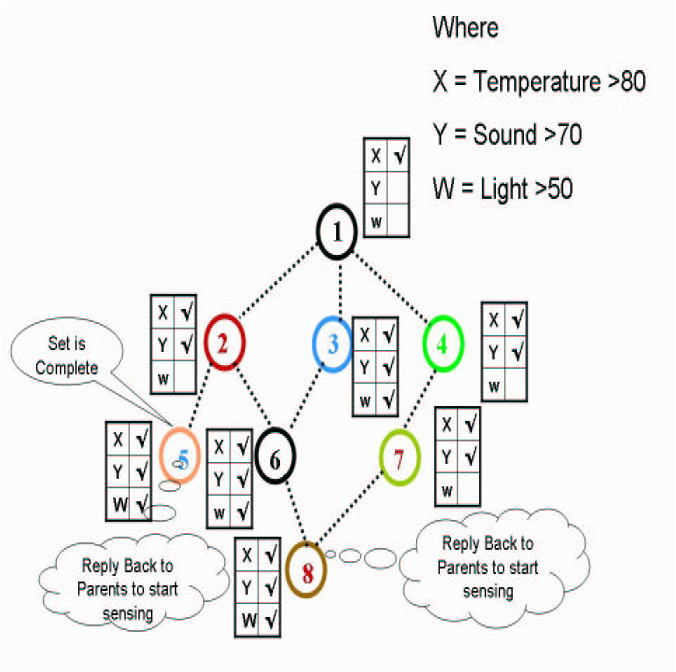

Figure 2: Example

Table 1: Nodes and their Sensing Capabilities

Node Id	Sensing Capabilities			Co-Ordinates
	Temperature	Sound	Light	
1	√	×	×	(0,1)
2	√	√	×	(1,2)
3	×	√	√	(1,3)
4	×	√	×	(2,2)
5	×	×	√	(2,3)
6	×	√	√	(4,2)
7	×	√	×	(4,3)
8	√	√	√	(4,5)

node 2,3,4 participates in this, hopcount=1
At node 2: EventCounter[temp,sound,light]=[2,1,0] parent=1
At node 3: EventCounter[temp,sound,light]=[1,1,1] parent=1
At node 4: EventCounter[temp,sound,light]=[1,1,0] parent=1
Broadcast 2: hopcount=2
nodes 5,6,7 participates in this, hopcount=2
At node 5: EventCounter[temp,sound,light]=[2,1,1] parent=2
At node 6: EventCounter[temp,sound,light]=[2,2,2] parent=2,3
At node 7: EventCounter[temp,sound,light]=[1,2,0] parent=4
Broadcast 3:
node 8 participates in this, hopcount=3
At node 8: EventCounter[temp,sound,light]=[3,3,3] parent=7
Here HopCount=3 and ValidRegion()=3

Since the set is complete i.e. by forming a group, they can detect composite event.

So node 8 sends confirmation message to its parent. Then it starts sensing and it will send this back to parent. This is repeated at every node until it reaches sink.

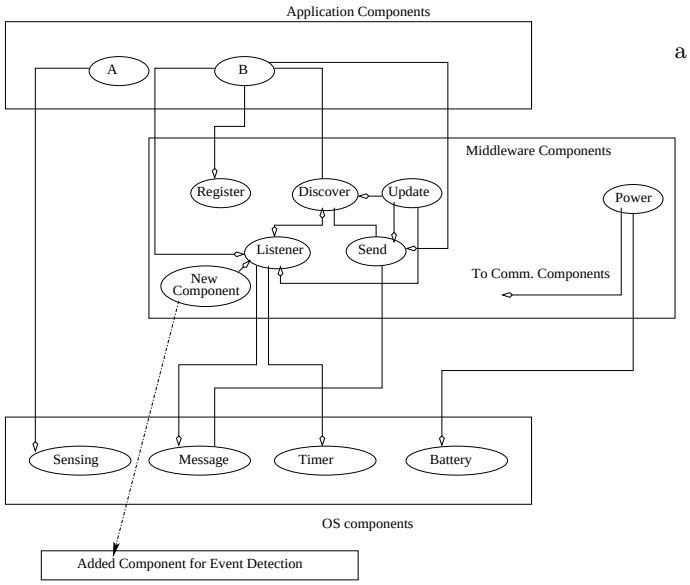

Figure 3: Middleware Components

Collection phase

After generating data, node 8 will send its data to its parent. Node 7 waits for 2ms to receive data from neighbors. After receiving data from its children it aggregates with only temperature and sound data because it can generate only temperature and sound data. In this way data gets aggregated if received data matches with generated data and is sent to sink. Event is reported to user node with corresponding data provided if all conditions are satisfies and threshold of uncertainty is satisfied.

6. IMPLEMENTATION DETAILS

6.1 COMiS Background

Component technology has many benefits such as extendability, reusability, low development cost and understandability [10]. COMiS [2], a component oriented middleware for sensor networks is developed using such a technology. Here middleware itself is composed of components. The main advantage with this is, components are loaded as and when actually needed[10]. The components are different for different sensor nodes in the network depending on sensor node functionality. Components are loaded in to memory as dictated by application semantics.

The basic application that is running above this middleware is assumed to be written in Distributed Compositional Language (DCL)[5]. DCL is a language developed for programming sensor networks which composes application by capturing the interaction between components present on different nodes. Middleware lies below the application in the form of components and provides some basic services.

It provides services such as finding components within a distance 'd', registration, component updating, power management etc. The middleware has six components namely *Listener*, *Discovery*, *Send*, *Register*, *Update* and *Power Management*. The components are shown in figure 3.

a

App. Id	Event1	Event1 Counter	Event2	Event2 Counter	...	EventN	EventN Counter	Threshold	No.Of Hops	StartOf Region	EndOf Region	Flag

b

App. Id	No.Of Hops	StartOf Region	EndOf Region	U

Figure 4: Message Formats

Table 2: Message Types

Flag	Message Type
S	Message in Initialization Phase
R	Message in Collection Phase
C	Confirmation Message

6.2 Components of COMiS

Brief explanation about each component of middleware

- **Listener:** This component is responsible for receiving all incoming requests. Depending upon the message type it can take various actions such as forwarding to neighbors, adding message to its own buffer etc.
- **Register:**This allows to register a component locally. After registration the registered component is available for discovery.
- **Discovery:**This is used for locating components existing on the network. This follows a broadcast over the desired region and get the ids of all the nodes which have the component.
- **Send:**This is used to send data to components present on various nodes.
- **Update:**This is mainly for deployment and updating of components. It also allows us to make even software updates to middleware components also and hence achieving adaptivity.
- **Power Management:**The power routines provided by the OS are extended by the middleware.

To integrate our proposed framework in to COMiS, new message types are introduced. The format of the message types are shown in figure 4. The figure shows the Event and *Event-Counter* along with it. It also shows the threshold which is same as *THRC* explained in section 4. StartRegion and EndRegion are specified in co-ordinates. A node participates only if its position is within the desired region.

The subscription and confirmation message formats are shown in Figure 4a. and the format of unsubscribe message is shown in Figure 4b.The message types for the corresponding flags are shown in table 2. The same message is used for subscription and confirmation by changing the flag. Whenever listener component encounters subscription message type, the event detection framework is plugged in and the protocols are initiated based on the type of event.

7. CONCLUSION AND FUTURE WORK

A framework for event detection using collaboration is proposed in this paper. Simple event detection and composite event detection protocols are studied. An event based tree is constructed using Publish/Subscribe communication paradigm to accomplish collaboration and user mobility. We showed its integration with component oriented middleware. The goal of detecting an event is accomplished through collaboration even there are failures. Currently the communication model does not support mobility of sensor nodes. In the proposed framework, priority of events is not considered in composite event detection. We are also working on language level issues to support collaboration.

8. ACKNOWLEDGMENTS

This work is supported by Honeywell Technology Solutions Laboratory Pvt Ltd (HTSL) India

9. ADDITIONAL AUTHORS

Additional author: S. Priya (Distributed and Object Systems Lab, email: `priya@cs.iitm.ernet.in`).

10. REFERENCES

[1] C. Shen, C. Srisathapornphat, C. Jaikaeo. Sensor information networking architecture and applications. *IEEE Personal Communications*, pages 52–59, 2001.

[2] D. Janakiram, R. Venkateswarlu, S. Nitin. COMiS: Component Oriented Middleware for Sensor Networks. In *To appear in the proceedings of 14th IEEE Workshop on Local Area and Metropolitan Networks (LANMAN)*, 2005.

[3] I. F.Akyildiz, W.Su, Y.Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. In *Computer Networks*, pages (38)393–422, 2002.

[4] G. C. Guiling Wang, Wensheng Zhang and T. La. On Supporting Distributed Collaboration In Sensor Networks. In *Proceedings of the IEEE Military Communications Conference (MILCOM)*, 2003.

[5] D. Janakiram and R. Venkateswarlu. A Distributed Compositional Language for Wireless Sensor Networks. In *IEEE Conference on Enabling Technologies for Smart Appliances (ETSA)*, 2005.

[6] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TinyDB: an acquisitional query processing system for sensor networks. *ACM Trans. Database Syst.*, 30(1):122–173, 2005.

[7] M. Mansouri-Samani and M. Sloman. GEM: a generalized event monitoring language for distributed systems. *Distributed Systems Engineering*, 4(2):96–108, 1997.

[8] T. Nieberg, S. Dulman, P. Havinga, L. van Hoesel, and J. Wu. Collaborative algorithms for communication in wireless sensor networks. pages 271–294, 2003.

[9] K. Obraczka, R. Manduchi, and J. Garcia-Luna-Aveces. Managing the Information Flow in Visual Sensor Networks. In *Proceedings of the Fifth International Symposium on Wireless Personal Multimedia Communications*, pages 27–30, 2002.

[10] N. Parlavantzas, G. Coulson, M. Clarke, and G. Blair. Towards a Reflective Component Based Middleware Architecture. In *On-Line Proceedings of ECOOP'2000 Workshop on Reflection and Metalevel Architectures, June 2000. Available at http://www.disi.unige.it/RMA2000.*, 2000.

[11] P. R. Pietzuch. *Hermes: A Scalable Event-Based Middleware*. PhD thesis, University of Cambridge, February 2004.

[12] S.Li, S.Son, and J.Stankovic. Event Detection Services Using Data Service Middleware in Distributed Sensor Networks. In *Proceedings of the 2nd Internnational Workshop on Information Processing in Sensor Networks*, 2003.

[13] W. W. Terpstra, S. Behnel, L. Fiege, A. Zeidler, and A. P. Buchmann. A peer-to-peer approach to content-based publish/subscribe. In *DEBS '03: Proceedings of the 2nd international workshop on Distributed event-based systems*, pages 1–8, New York, NY, USA, 2003. ACM Press.

[14] N. Xu. A Survey of Sensor Network Applications.

[15] Yong.Yao and Johannes.Gehrke. The Cougar Approach to In-Network Query Processing in Sensor Networks. In *SIGMOD*, 2002.

[16] W. Zhang and G. Cao. DCTC: Dynamic Convoy Tree-Based Collaboration for Target Tracking in Sensor Networks. *to appear in IEEE Transactions on Wireless Communication.*