A distributed middleware for self-configurable wireless sensor networks

Lei Zhang

School of Computer Science and Technology,
Tianjin University,
Tianjin Key Laboratory of Cognitive Computing and Application,
Tianjin, 300072, China
E-mail: lzhang@tju.edu.cn

Alvin Lim

Department of Computer Science and Software Engineering,
Auburn University,
Auburn, AL 36849, USA
E-mail: limalvi@auburn.edu

Yi Pan

Department of Computer Science,
Georgia State University,
Atlanta, GA 30303, USA
E-mail: yipan@gsu.edu

Bin Wu*

School of Computer Science and Technology,
Tianjin University,
Tianjin Key Laboratory of Cognitive Computing and Application,
Tianjin 300072, China
E-mail: binwu.tju@gmail.com
*Corresponding author

Abstract: Heterogeneity, mobility, and resource constraints are the inherent features of the wireless sensor network. In addition, for the wireless sensor network development, the developers have to design the low-level system details adaptive to specific upper layer non-compatible protocols. The distributed services are a middleware, which can make large-scale and resource-constrained wireless sensor network adapt to the dynamic environment in an effective way. It can not only guarantee transmission quality, but also can make application development simple and efficient. In the distributed services, the composition service deals with the dynamics efficiently by managing the nodes to form and maintain task-oriented groups, which not only increases the degree of transparency to the applications, but also improves the reliability and energy efficiency. The entire process releases developers from the tedious low-level work. In this paper, we focus on the demonstration of composition service and its achievement for the self-configurable wireless sensor network.

Keywords: distributed services; composition service; distributed middleware; self-configurable; wireless sensor networks.

Reference to this paper should be made as follows: Zhang, L., Lim, A., Pan, Y. and Wu, B. (xxxx) 'A distributed middleware for self-configurable wireless sensor networks', *Int. J. Sensor Networks*, Vol. x, No. x, pp.xxx–xxx.

Biographical notes: Lei Zhang received her PhD in Computer Science from Auburn University (Auburn, AL, USA) in 2008. She worked as an Assistant Professor from 2008 to 2011 in the Department of Computer Science of Frostburg State University (Frostburg, MD, USA). She is now an Assistant Professor in the School of Computer Science and Technology at Tianjin University (Tianjin, China). Her research interests include wireless sensor networks, distributed algorithms, and network security.

Alvin Lim is currently an Associate Professor of Computer Science and Software Engineering at Auburn University. He received his PhD in Computer Science from University of Wisconsin at Madison in 1993. His research interests include self-organising sensor networks, mobile and pervasive computing, network security, wireless networks, reliable and dynamically re-configurable distributed systems, complex distributed systems, mobile and distributed databases, distributed operating systems, and performance measurement and analysis. He has published widely in journals and conferences in these networking and distributed systems areas.

Yi Pan is the Chair and a Professor in the Department of Computer Science at Georgia State University. He received his Bachelor and Master degrees in Computer Engineering from Tsinghua University, China, in 1982 and 1984, respectively, and his PhD in Computer Science from the University of Pittsburgh, USA, in 1991. His research interests include parallel and cloud computing, wireless networks, and bioinformatics.

Bin Wu received his PhD in Electrical and Electronic Engineering from the University of Hong Kong (Pokfulam, Hong Kong) in 2007. He worked as a Postdoctoral Research Fellow from 2007–2012 in the ECE Department at University of Waterloo (Waterloo, Canada). He is now a Professor in the School of Computer Science and Technology at Tianjin University (Tianjin, China). His research interests include computer systems and networking as well as communications.

1 Introduction

Many dynamic wireless sensor network (WSN) systems must be controlled by adaptive methods that utilise critical and real-time data gathered from many sensor devices (Lim, 2001, 2002), such as target tracking, which is a typical application of WSN. These sensor nodes are connected in an ad hoc fashion collecting and processing data in a distributed way. Nodes need to find and communicate with specific even possibly remote nodes. Nodes may fail at anytime and anywhere, negatively affecting the whole system. Even if new nodes are available to replace the failed ones, existing nodes have to be aware of their availability. Based on the requirements of specific application, building such a network is difficult for the following reasons. Firstly, there are many different types of sensors with different capabilities. The sensors may be deployed with specialised and possibly non-compatible networking protocols and different application requirements; secondly, sensor nodes may be deployed with little or no pre-planning; thirdly, the network must survive harsh environmental conditions, dynamic nodes composition and task requirements changes, device failure, and nodes mobility. These make (WSN) research challenging.

So far, a lot of WSN research work focuses on developing platforms and protocols such as the hardware and system developments reported in Lim (2002). However, the heterogeneous nature of the development environments is a big block to the system developments. Software is always developed over a particular platform, while it is hardly ported to other different hardware or operating systems. The development of system has to involve developers in the low-level system details in order to adapt to specific upper layer protocols. As addressed in Hadim and Mohamed (2006), Radhika and Malarvizhi (2012), a systematic framework that provides developers with a conceptual view of the network and hides lower layers details is necessary. The distributed middleware is proposed as an ideal solution for the complicated WSN development (Laukkarinen et al., 2011; Mohamed and Al-Jaroodi, 2011; Laukkarinen et al., 2012). The focus of more and more research in WSN, such as Afzal et al. (2009) has thus turned to define a middleware, which sits above the operating system and below the application, and abstracts lower-level functionality such as network connectivity and provides a coordination interface to the applications. Much of the work (Hadim and Mohamed, 2006; Mottola and Picco, 2011; Sakthidharan and Chitra, 2012) has targeted on the development of middleware platforms specifically designed to meet the challenges of the large-scale resource-constrained WSN.

However, the pervious research on the middleware research has their drawbacks. Such as, middleware Atlas (King et al., 2006) was developed on the assumption that WSNs have unlimited source of energy, which is not true in reality. In addition, lacking real-time handling mechanisms, Atlas can hardly adapt to the real-time environment. SINA (Haghighi and Cliff, 2013a) is a novel agent-based middleware for WSN, which is written in Java and runs on networks of various Java-enabled embedded systems. It can support nodes clustering. However, the idea of agent-based middleware development poses a challenge on how to define an appropriate design process, which not only supports the development, but also provides an extensible and maintainable mechanism to align requirements and environmental variability at run-time. This can only be achieved when the design processes are clear.

A new middleware, called the distributed service layer, based on which other networking services could be spontaneously specified and re-configured, has been proposed and addressed in Lim (2001, 2002). All these services are designed to simplify and facilitate the application developments and to help the network adapt to the dynamics with the consistency to existing network programming paradigms.

As proposed in Intanagonwiwat et al. (2000) and Intanagonwiwat et al. (2003), the directed diffusion is a widely used distributed routing protocol for WSN. It makes routing decisions according to the data requested. One of the most significant issues of the directed diffusion protocol is the packet flooding. The region filter developed in Ivester and Lim (2006) is an improvement to the directed diffusion, since it maintains message re-transmission mechanisms within a specific region and filters out some traffic in the networks.

The other important issue of directed diffusion is node failure and mobility treatment. There are some mechanisms responding to topology changes that are built in the directed diffusion. When data stop arriving at one node due to its upstream node going down or moving away, directed diffusion will trigger the event of flooding similar interest packets in the entire network. This basic form of adaptability is adequate for simple query-response systems, but not sufficient for complex applications due to long delay required to find a replacement, network flooding, and the undetermined physical location of the substitution. Additionally, the task of reconfiguring the upstream nodes will introduce extra overhead when adapt to directed diffusion changes at the upper layer. This can not be effectively achieved.

This research involves two previous researches. The distributed services middleware layer (Lim, 2001, 2002), which proposed only the main idea of distributed services, but not implementation details of each service. The directed diffusion routing protocol (Intanagonwiwat et al., 2000, 2003) is the other work involved. The following goals are to be achieved in the research:

- to design and implement application programming interface (APIs) for the distributed composition service, which is the most important component of the distributed services
- to maintain the normal networking functionalities under abnormal networking conditions with the help of distributed services
- to achieve the design principles of WSN middleware (Radhika and Malarvizhi, 2012), data-centric, dynamic adaptive, user-transparent, scalability, and energy-efficient.

In general, the major contributions of this research work can be summarised as follows: firstly, abstracting lower-level functionalities, providing a coordination interface to the applications, and making the heterogeneous WSN development simple and efficient; secondly, designing and implementing the API based on the unique features of WSN, such as resource constraints, complex link quality, dynamic topology and scalability, tackling these special issues of WSN. These solutions are evaluated by the ISEE, a self-developed testing environment and they are outperformed compared with directed diffusion. Finally, this research is the preparation for the target tracking application under the real-world scenarios.

The remaining of the paper is organised as follows: Section 2 presents some technologies related to the distributed composition service. Section 3 discusses the architecture of the composition service with its APIs. The real implementation is demonstrated in Section 4. The performance is evaluated in Section 5. Section 6 draws the conclusion by summarising the research work.

2 Related work

2.1 Directed diffusion

As proposed in Intanagonwiwat et al. (2000, 2003), the directed diffusion is a distributed routing protocol with routing decisions made according to the data requested. The directed diffusion has some creative features including data-centric dissemination, reinforcement-based adaptation to the empirically best path, in-network data aggregation and caching. Directed diffusion uses both *positive reinforcement* and *negative reinforcement,* which sends data to the lower delay neighbouring node with a higher data rate and to the higher delay neighbouring nodes with lower data rate. Through this method, directed diffusion inhibits nodes on the long delay path from packet forwarding. The directed diffusion offers accessibility to the diffusion core behaviours through *publish* and *subscribe* APIs. Though some mechanisms reacting to the dynamic topology changes are built in the directed diffusion, there is still a long recovery delay and network traffic generated at the time. Thus, there is much work to be done to improve its performance regarding to the time-sensitive and resource-constrained WSN applications.

2.2 Middleware for WSN

Software development for WSN requires novel *programming* paradigms and technologies. Conventional principles of communication are mostly inapplicable due to dynamic topology changes and the need for cooperative task processing in WSN (Hadim and Mohamed, 2006a, 2006b; Radhika and Malarvizhi, 2012). As addressed by Laukkarinen et al. (2011, 2012) and Mohamed and Al-Jaroodi (2011), introducing distributed middleware to the WSN application development has become a breakthrough in the WSN development, which brings adaptability, simplicity, and efficiency to this field. In the WSN development, the distributed middleware refers to two aspects: WSN application on one hand and the distributed application which interacts over the network on the other hand. Primary objective of the middleware layer is to hide the complexity of the network environment by isolating the application from protocol handling, memory management, network functionality, and parallelism. The benefits of adopting middleware to the WSN implementations can be summarised as follows:

- Shield software developers from low-level, tedious, and error-prone platform details, such as socket-level network programming.
- Provide a consistent set of higher-level network-oriented abstractions that are much closer to application requirements in order to simplify the development of distributed and embedded systems.
- Invoke operations on target objects to perform interactions and functionality needed to achieve application goals. A wide variety of middleware-based services are made available off the shelf to simplify application development. Aggregations of these simple, middleware-mediated interactions form the basis of large-scale distributed system deployments.

Next, we introduce some existing middlewares of WSN.

2.2.1 Data services middleware (DSWare)

The data services middleware (Li et al., 2004) is based on the notion of events, whereby the application specifies interest in the state changes of the physical world, called basic events. Upon detecting an event, the node sends an event notification towards interested applications. The application can also specify a certain pattern of events such that the application is only notified if occurred events match this pattern. The real data generated by the sensors can be stored or forwarded to other nodes for processing. Since this functionality is needed for various applications, a data-services middleware can avoid the re-development.

DSWare provides support for group creation. Additionally, it also supports data-centric routing protocols like directed diffusion. Though computationally intensive, it provides facilities for information storage and retrieval. However, the creation of a group is triggered by a real-world event and requires the node to correctly identify and categorise an event. In the heterogeneous WSN, not all nodes have the computational capability for this complex task. There is also no single entity to manage the group formation and handle group information. This capability is vital for many WSN applications, especially for large-scale applications.

2.2.2 QoS infrastructure base on service and middleware (QISM)

QISM (Nan et al., 2009) is a service-oriented infrastructure for WSN. The main characteristics of QISM are the active mechanisms which are based on feedback and negotiation between applications and network. The architecture is based on middleware and service publishing and subscribing. The methods are based on tasks and functional domain. Through the active communications between the applications and network, QISM can achieve the network support applications and the applications adapt to the network. Therefore, the QoS of the application is better guaranteed and the lifetime of network is prolonged as well. In general, the architecture is application independent and it can support complex applications.

Though QISM makes a progress on the road of middleware development for WSN, it cannot guarantee the scalability. In addition, it does not involve specific mechanisms dealing with network dynamics, such as node failure.

2.2.3 MiSense

MiSense (Khedo and Subramanian, 2009) is a service-oriented component-based middleware to support distributed WSN applications with various performance requirements. MiSense reduces complexity by imposing a structure on top of the component model by offering well-defined service-specific interfaces to the rest of the system. MiSense breaks up the middleware design into fine, self-contained, and richly interacting components in order to resolve the gap between the optimisation requirements for specific scenarios and the needs for flexibility and convenience for energy-efficient WSN applications development.

MisSense is only an idea. It has not been implemented or tested for reliability, scalability, and adaptation. There might be some issues if it is really tested with complicated real-time WSN applications.

2.2.4 Sensomax

Sensomax (Haghighi and Cliff, 2013b) is one of the newest middleware developed for WSN applications. It presents a novel combination of several best practices from existing solutions, facilitating fully distributed and decentralised bulk programming and/or updating of sensor nodes, serving multiple simultaneous applications deployed by single or multiple users, allowing dynamic run-time changes in the application requirements, and offering on-the-fly switching between time-driven, data-driven, and event-driven operational paradigms. Sensomax provides a sophisticated set of APIs, a feature-rich desktop application, a web application for cloud-based distributed networks.

However, the idea of agent-based middleware development poses a challenge on how to define an appropriate design process, which not only supports the development, but also provides an extensible and maintainable mechanism to align requirements and environmental variability at run-time. This can only be achieved when the design processes are clear.

2.3 Distributed services

Based on the idea of the distributed middleware and its significant contribution to the WSN development, the idea of distributed services (Lim, 2001) has been proposed. The distributed services establish the foundation on which other applications can be built. The dynamic adaptation service collaborates with the distributed lookup service and composition service to monitor the failures of the sensor nodes and manage the correct schedules of failure recovering procedures. The distributed services support efficient replacement of faulty nodes with minimal disruption and continuous interactions among the nodes in the network. The sequence of recovery procedures will preserve the level of reliability required by a specific application. These distributed services execute over the direct diffusion routing layer that alleviate some of the problems of mobility, disconnection, dynamic configuration, and limited power. The three proposed fundamental distributed services include the lookup service, the adaptation service, and the composition service. However, the distributed services are the proposed ideas. Only by implementing these ideas, other network services can be specified spontaneously in the network.

2.3.1 Lookup service

A node registers a resource that it maintains or services that it can perform with a lookup server (Lim, 2001, 2002). The lookup server contains location information on services or resources of multiple clusters. The nodes require the service or the resource may request it through a lookup server. Thus, the primary function of the lookup service is to store the information of the services and resources offered and to distribute the information when needed. For instance, when a node fails, the lookup service will take the responsibility to find a replacement node with the same function.

2.3.2 Adaptation service

Adaptation server (Lim, 2001, 2002) utilises information from composition servers, lookup servers, and analytical tools to control smart nodes during dynamic reconfiguration and failure recovery. Adaptation servers monitor clusters of smart nodes during normal execution either by probing the smart nodes, spontaneous signal from the sensors, or explicit network management directives for reconfiguration and failure recovery. When a run-time reconfiguration is requested or triggered, the adaptation server will generate the appropriate schedule of the reconfiguration operations that will ensure the reconfigured and affected sensor nodes are globally consistent. To ensure correct adaptation and maintain consistency, the adaptation server makes use of analytical tools for dependency analysis and relevant information from compositional servers and lookup servers. In general, the primary function of the adaptation service is to help networks to adapt to the dynamic changes.

2.3.3 Composition service

The composition server (Lim, 2001, 2002) manages clusters of sensor nodes by allowing various smart nodes that may be added to or removed from the network. Besides this, it is also in charge of network abstractions or group behaviours of the clusters and hierarchical composition of clusters. It not only simplifies dynamic reconfiguration of services provided by each sensor node, but also makes the development of large self-organising sensor networks much easier, since it allows individual node and cluster to be specified and designed independently. Composition servers enhance compositionality and clustering abstraction of WSN. By adopting cluster-based hierarchical architecture, group communications can be efficient and easy. Synchronisation constraints associated with network protocols and system services among the nodes can be specified by the means of clusters. The capability to specify the composite clusters enables designers to build large and complex WSN by clustering nodes at each level together. In general, the composition service is in charge of nodes formation and works corporately with the adaptation service to mask node failure and guarantee normal transmission. In the previous research, only the ideas of distributed composition service have been proposed, and implementation has not been done. We implement the ideas of distributed composition services in this research.

2.4 The common issues of existing WSN middleware

In this section, we do the literature reviews regarding the distributed services implementation, such as the underlying routing protocol, directed diffusion, benefits of WSN middleware, the advantage of distributed services. For the existing WSN middleware, we analyse the advantages of each one. However, there are still some fundamental issues, which can be summarised as: firstly, they do not inherently support any clustering techniques, which are very important for the large-scale WSN; secondly, there is no directed diffusion or any other routing protocols' API available; thirdly, they do not have the capability to deal with dynamic and complicated situations.

In the following, we will address the implementation of distributed composition service. The distributed composition service is a unique middleware of WSN. Building on the data-centric routing protocol, directed diffusion, the composition service takes advantages of previous middlewares and improves their efficiency. As an indispensable component of the distributed system, it can maintain the scalability by introducing the important feature of clustering. In addition, it integrates seamlessly with other two distributed services to achieve the common goal of adaptation and stability while still satisfying the various applications' requirements.

3 Composition service implementation

The previous section summarises the various areas of research on which this work is based and also gives several examples of middlewares in WSN. In this section, we will discuss composition service implementation in detail.

Nodes can be dynamically composed into impromptu networked clusters under the control of a compositional server. The clustered nodes can provide distributed services corporately, such as data filtering and aggregation. In addition, some nodes can provide services to other nodes and simultaneously they can be consumers of services the other nodes provide. We apply the component-based methodology to design composition service in order to make it efficient, easy, and inter-operable with other network services.

Clustered nodes encapsulate the networking and system capabilities provided by the group of the nodes. There is a cluster head responsible for the control of the inter-cluster communications and networking functions. Inter- group communication can be efficiently implemented by sending a message first to the cluster head which then multicasts it to the cluster members. The nodes in one cluster may cooperate to perform a common networking and system functions. The composition server is in charge of various nodes adding or removal from the clusters in the WSN.

The composition service comprises the composition server and the composition clients (the nodes sending the request to composition server to join or leave the group). Both of them make use of the composition service API for group joining or leaving messages exchange.

The following significant components correspond to the composition service implementation.

3.1 Sensor object model

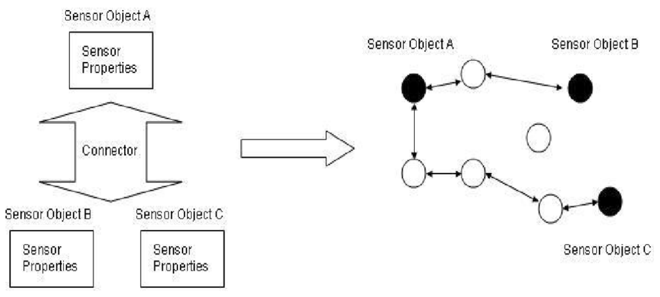
The data-centric attributes of the WSN protocols identify the type of nodes by the data they generate. Learning from this idea, the sensor object model facilitates the communication by categorising the nodes based on the data generated, the services, QoS they offer, and their location. An application can use the sensor object model to encapsulate the properties of a node. A real-world event is encapsulated by the event object. The event object has location information, which can be used by the composition server when assigning a node to the group within a predefined region. The composition server associates a sensor object with an event object and uses this information to make group formation decisions, grouping objects that sense similar events together.

3.2 The connector model

The connector model is an indispensable component of the composition service. It masks the problems in the transmission such as node failure and mobility and provides the application with a consistent view of the underlying network.

As shown in Figure 1, a connector represents a data exchanging relationship between two or more sensor objects. A connector is made up of links. A link is the physical representation of the connection between two or more sensor objects. On the right side of Figure 1, there is one link between node A and node B and another link between node C and node A. Data flow can reach those sensor objects through the links. Links are in charge of the endpoints of the connection, work with the routing protocol to identify these endpoints, and efficiently discover and maintain paths to them. As illustrated on the left side of Figure 1, a connector can contain links to more than one sensor object depending on the type of the connector.

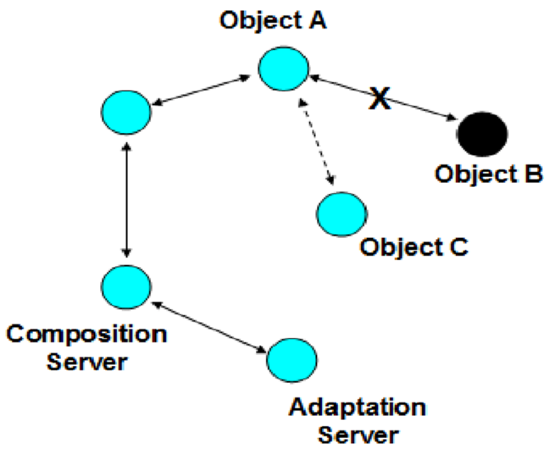
Figure 1 Relationship between connector and sensor object



A connector maintains a reference to the invoking sensor object and the group ID to which the sensor object belongs. Using the group ID, the connector can establish a data path among the sensor objects in the same group, so that they can communicate with each other. In Figure 1, nodes A, B, and C have similar attributes and are close to each other. Their corresponding sensor objects and connectors are set up to facilitate their communication in the duration of the task.

As Figure 2 shows, connectors are reconfigurable in the sense that the adaptation server can replace inactive sensor objects to maintain reliable data association between existing sensor objects. Links can be deleted or added to a connector if needed. For instance, as in Figure 2, if node B fails, its link will be removed from the connector. The adaptation service is queried by the composition server to find a substitution node with similar properties in the neighbourhood. A new link between node A and C is established as the substitution and is added to the connector. Hence, the connectors have the capability of being reconfigurable in an application-transparent manner, ensuring application requirements are met and any offered service is available.

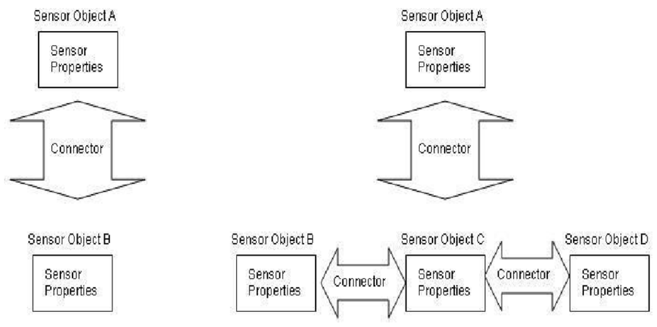
Figure 2 Connector for the node replacement (see online version for colours)



Associated with each connector is the connector type described in terms of the number of endpoints addressed by the connector. As shown in Figure 3, there are two kinds of connectors: one to one and one to many. The connector between node A and node B is one-to-one type. The connector connecting nodes A and B, C, D is one-to-many type.

Communication interface interfaces the connector with the routing protocol. It is the core component of the connector and provides mechanisms for receiving and sending data used by the connector. The application does not interact with the communication interface. In this manner, the routing protocol can be modified without changing the application layer and affecting the behaviours of the connector. Communication interface supports three kinds of communications: remote procedure call (RPC), multicast, and peer to peer.

Figure 3 Two types of connectors

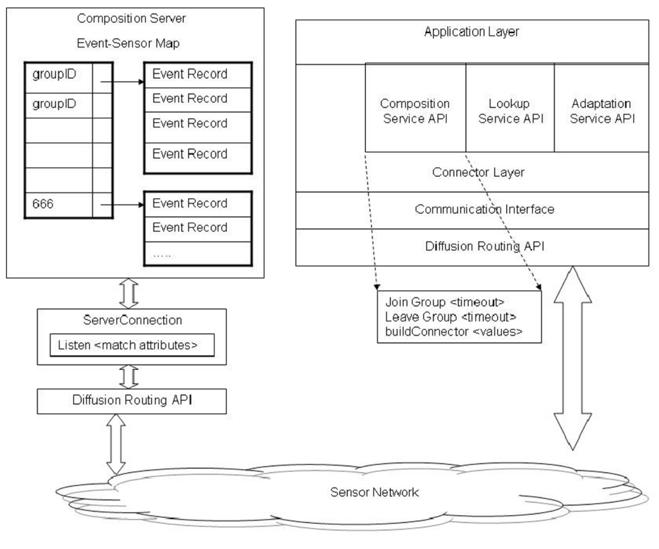


3.3 *System architecture*

The composition service implementation is spread across several layers. Each layer provides a set of interfaces to the layer above it. These interfaces shield the upper layer from the functional complexity encapsulated by the current layer and also allow to invoke the desired behaviour from the current layer. As shown in Figure 4, composition server system architecture includes the following parts:

The task of group formation and maintenance is undertaken by the composition server. We can see from Figure 4 that the composition server manages the clustering by a hash table which has the groupID as the key and the corresponding EventRecord as the value. The nodes detecting events are grouped into certain clusters according to the types of the events.

Figure 4 Composition server system architecture



The application layer: The applications supporting a variety of distributed tasks, such as collaborative clustering, target tracking, and in-network aggregation can operate at the application layer. Once the nodes establish the channel of communication by means of the connectors, they can maintain the data flows among themselves in accordance with the needs of a particular application.

The composition server: The responsibilities of the composition server can be given as follows:

- assist with group formation based on location and event characteristics
- inform the nodes in the group about their status and that of other nodes in their group
- maintain time-sensitive group information and inter-operate with the adaptation server for group reconfiguration in case of node failure
- do the group composition and management in a distributed network environment with other composition servers.

The composition server is not only in charge of a collection of detected events, but also the group structure changes in response to the dynamics in the network.

The service client: The service client detects an event and sends a request to the composition server in order to join a group using the composition service API. The composition server assigns group ID and ID of each node in the group including cluster head to the service client. After the service client contacts its cluster head, the cluster head updates the multicast channel information. Through the multicast channel, all nodes in the group send details of detected events to each other. Through this approach, the composition service can provide a reliable path and mask network inconsistencies.

The composition service API: The main task of the composition service API is to make the applications communicate with the composition server as seamless as possible by masking the complexity of sending and receiving. The composition server makes use of the composition service API to listen to incoming requests for joining or leaving a group. The detail of how the composition service API work can be summarised as follows:

- It uses the underlying connector interface and sets up a stable communication channel to the composition server. This channel is maintained as long as the application needs to interact with the composition server.
- In the context of directed diffusion, the composition service API also makes the best efforts to minimise flooding of interest messages when discovering a reliable path to the composition server and other nodes in the group.

- The composition service API also maintains or reconfigures group information when nodes are failed. It helps the composition server work with the adaptation server to discover substitute nodes and add them to pre-existing groups.

In general, the composition service API is in charge of nodes join/leave the group, create/delete the message delivery connector, and listen to the request.

The connector layer: As shown in Figure 4, the connector layer lies just below the application layer in the system architecture. The connectors provide applications with a consistent feel of the network and maintain relationship with members of the group even if group composition changes. Connectors also work with lower layers dealing with data transmission and routing. A connector has the capability of sending and receiving data to and from other nodes. Before the communication, the connector must construct links to these nodes. As described earlier, a link object encapsulates the properties of a physical link to the other node. In general, the connector layer API is in charge of creating/removing a link and sending/receiving data.

Communication interface: As shown in Figure 4, the communication interface layer sits above the directed diffusion routing protocol layer. It bears the responsibility of interacting with the routing layer to discover efficient paths. Sending and receiving data or interest packets matching the diffusion protocol standards are also done at this layer.

Diffusion routing protocol: The composition service uses directed diffusion as the routing protocol. It makes judicious use of the built-in types and libraries provided by the directed diffusion. Though this makes the application somewhat dependent on the directed diffusion, it eliminates the extra processing needed to convert application defined attributes to diffusion attributes.

3.4 Cluster formation

The composition server listens for interest messages from event sensing nodes using the ServerConnection listen() function. The listen() function returns a one-to-one type connector through which the composition server sends a reply. The reply is sent to each of the nodes that composition server receives a matching interest. When a reply to a join or leave group request is sent, the composition server also publishes its own attributes. For all essential operations, the composition server defines its internal (private) cluster member functions and uses them extensively.

The sequence of events for joining and leaving groups are explained below:

- On receiving a join group request the composition server first checks the pre-existing groups' properties.
- The composition server matches the latitude and longitude of the events for all the nodes involved. If such a group is found, the reply is sent to the new node with the group ID and group members.
- Otherwise, the composition server attempts to add it to the list with a special group ID. This can be done by using the insertEntry() method.
- Once a sufficient number of similar nodes are obtained in the special group list, the composition server will form a group and assign a new group ID. The exact area to be considered depends on the composition server. In this research, all nodes that are within 50 units (for example, 50 by 50 meters) of the event occurred area are grouped together. The composition server uses formGroup() function for this purpose. The group ID and list of nodes in the group will be recorded at the composition server.
- Duplicate requests exist due to the broadcast nature of directed diffusion or the node sent out more than one interest messages. Checking duplicate information can be done using the composition server member contains() method. This method will use the sensor ID to distinguish one node from the other.
- The composition server composes a reply using the attributes defined by the application and sends the reply back to all the nodes in the new group. The attributes are put into a NRAttrVec vector and passed to the connector send() function. The connector sends the packets using the functions provided by the underlying CommunicationInterface object.
- If a leave group request is received, the leaving node's information is retrieved. Its information is removed by removeEntry() function. The composition server composes a leave group reply sent back to the leaving node.

3.5 Relationship between composition service and diffusion attributes

For the composition server, it is important to identify the sensor nodes along with the events they detected and form groups based on this information. Clustering information has to be sent back to the nodes so that they can start communicating with other nodes in the group. It is thus necessary to define attributes that represent the current state of the events-node pair in order to support the composition service mechanism. The application can define the required attributes using the attribute factories provided by the directed diffusion according to its needs.

4 Composition server operating mechanisms

As addressed before, the composition server is in charge of cluster formation and dynamic changes handling.

The composition server API is also used to maintain or reconfigure group information when nodes are unable to communicate due to failure. The composition server also works with the adaptation server to detect new nodes and add them to the existing groups.

In the data-centric network, nodes are addressed by the data (or services) they offer. Similarly, in the distributed services, the service providers or source nodes have to obtain the properties from composition server so that they can get to know what kinds of services or data they can offer. The interest message generated by the service seeking node will be initially flooded over the entire network resulting in energy cost. The idea of region filter (Shen, 2002) is implemented with directed diffusion to limit the initial flooding. The composition server takes the responsibility of bridging the service providers and service seekers by maintaining the data-centric properties, which greatly reduces the network-wide flooding. In order to let the service-seeking or sink nodes find it when the composition server is at a distant location, the composition server registers with the lookup server.

In order to manage the group formation events, the composition server not only listens to the interest messages from event sensing nodes, but also publishes its own attributes when sending a reply to a joining group or leaving group request. A node in the group can produce data, offer services, or be the data and service consumer. There can be a multiple services offered by the nodes. The composition server needs a way to classify the requests according to the services types and respond to the requests.

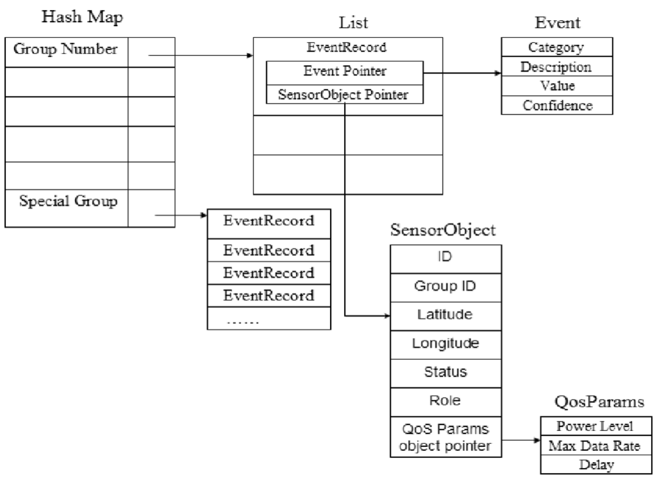
Converting the messages into diffusion-level attributes, pre-defining attributes, setting up matching rules for data-centric directed diffusion according to each application's desire are the major works of composition service API.

Through this way, the relationship of different attributes and network traffic to each corresponding node has been set up. Though the composition service helps the applications to define their needs, the applications still have their own responsibility to satisfy the specific requirements.

The composition server employs hash map to store the properties of the nodes. For easy retrieval of group member information, group ID is adopted. Detailed hash map is shown in Figure 5. In Figure 5, the composition server employs cascaded hash map to store the properties of the nodes, which is indexed by group ID for easy retrieval. The group maps to a list of EventRecord objects. For each event record in the list, there are two properties, Event Pointer and SensorObject pointer. Event Pointer points to event's attributes, such as category, description, etc. SensorObject Pointer points to SensorObject, which includes sensor object's attributes. There is a kind of record called Special Group. Special Group includes the groups with only one node for the corresponding event.

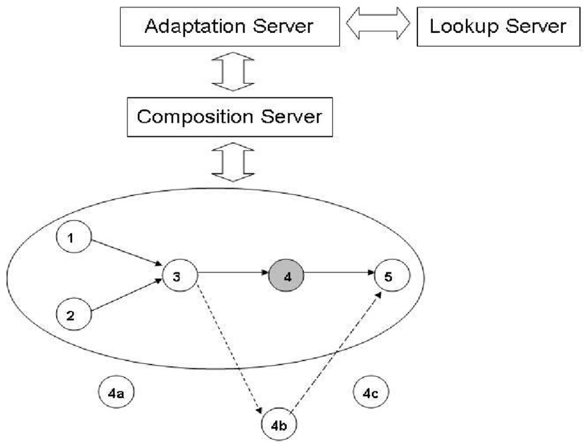
The most significant job of composition service is redirection or reconfiguration. This is performed by the connector when a node in the group fails. The composition server works with the adaptation server to discover a substitute node within the vicinity of the failure node and replaces it. This entire operation is transparent to the application. At first glance, it seems that the replacement node can be chosen by the composition server but not the adaptation server, which seems to minimise the communication overhead. However, the main purpose of the composition server is to form groups based on location and properties. Thus, this should be best implemented at the adaptation server under the help of composition server.

Figure 5 Composition server storing structure



The network is expected to survive from node failure. In a large-scale network, there might be many substitutes to the failure nodes. How to choose the best replacement node will be a key issue for the protocol design. For instance, in Figure 6, nodes 1, 2, 3, 4, and 5 form a group in a large-scale WSN. The solid lines indicate the data flow. Nodes 1 and 2 sense the raw data and forward to node 3 and then nodes 4. Node 5 takes the responsibility of improving data transmission efficiency with data integration and aggregation. The failure of node 4 produces negative impact on the established data flow. Though, nodes 4a and 4c might be the candidates of the replacement, redirecting the data flow through node 4b will be the best choice. The reason behind this is that node 4b is in the closest neighbour of node 4 and does the similar job with node 4. The collaboration of the distributed services can make the whole transformation easy, efficient, quick, and transparent to the application. This reduces the negative impact on the established data flow to the lowest level.

Figure 6 Service node replacement



The way for the failure recovery is that the adaptation server accesses the repository that maintains node information like service provided or the group information and so on, which is maintained by the lookup and composition servers. The adaptation service needs a mechanism to access the other services indicating a degree of communication and data exchange between two services.

5 Performance evaluation

5.1 Interactive sensor network execution environment (ISEE)

ISEE in Ivester and Lim (2006) has been developed as an environment to execute and monitor services of a WSN easily and effectively. It simulates a WSN by setting up nodes over fixed and available infrastructure and runs distributed applications on top of that. The combination of ISEE with the distributed services allows language independent users and developers to interact with WSNs without difficulties. It provides both simple run-time framework for repeatable experimentation that allows for easy transfer to real time on-site setups, event capturing, and measurement to portray an accurate view of real-world WSN events. The framework transparently supports testing and emulating the real WSNs.

5.2 Evaluation metrics

The performance evaluation of the distributed service mechanism is presented in this section. The performance evaluation has been divided into two parts according to the size of the network: smaller scale and larger scale. In the smaller scale testing, we check with the failure recovery time and average number of packets in the network. In the larger scale, average end-to-end delay, energy dissipation, and packet delivery ratio have been tested under different scenarios. When smaller scale can obtain a satisfactory performance, larger scale will be meaningful. This is the reason we begin testing with the smaller scale.

The testing environment is ISEE (Ivester and Lim, 2006). The number of packets refers to the interest packets and the data packets. At each node, the network traffic is measured by a system filter.

5.3 Evaluation for smaller scale

In the ISEE testing environment, a simple client-server target tracking event detection application runs over directed diffusion. Emulation runs are conducted to compare the application that takes advantage of the distributed services and the same application without running the distributed service but using directed diffusion directly. There are 10 randomly generated sources and sinks pairs for each experiment. Each node has a radio range of 50m. Each source node detects the target and creates one event every second. Each event is modelled as constant 512 byte packet and sent out every second. Interest packets are periodically created every 5 s with duration of 15 s. The negative reinforcement is set to be 2 s. Each result is the average value of 30 independent tests. The sensor fields are generated by randomly placing the nodes in a 200 m by 200 m square area.

We have implemented a basic application with 30 random generated static nodes. The data is collected by implementing groups with 6 nodes each during the time. The metric used for the comparison is the time taken for failure node detection and replacement, called *recovery time*, and the increase of the *average number of packets* in the network.

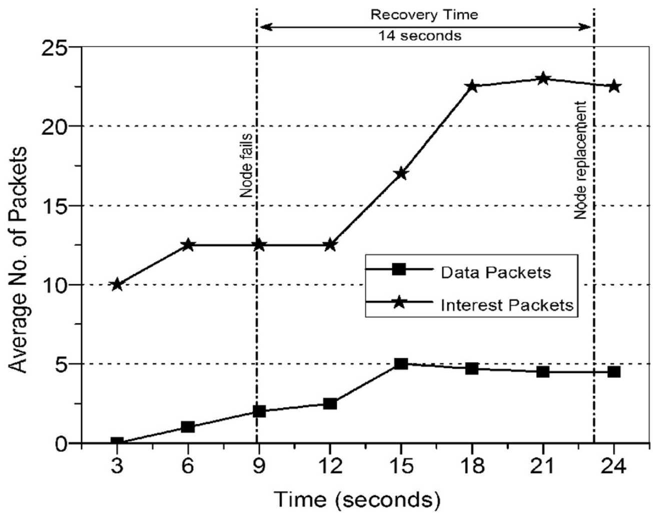
As shown in Figure 7(a), after a node fails, there is hardly immediate interest packets increase although data packets increase. After a while, the interest packets increase dramatically, and data packets first increase and then decrease gradually. This implies that directed diffusion detects and reacts to the node failure slowly. The directed diffusion does not have specific mechanisms to detect and deal with the node failure. The only action taken by directed diffusion when a node fails is to flood the entire network with interest packets.

At the same time, the data packets going through the reinforced paths still arrive at the sink but with a lower rate. When a substitute has been found, the number of interest packets intends to be constant, just as shown in Figure 7(a). There might be the case that a new source is found, albeit at an unknown location which may be far off from the other sink nodes. This is the key reason that the recovery process of the directed diffusion is slow, as can be gauged from the lengthy recovery time. The application suffers from data loss for a relatively long time.

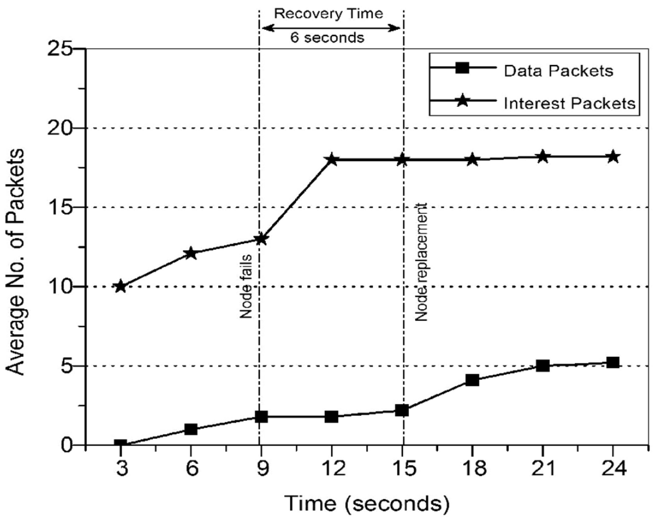
Comparatively, Figure 7(b) shows the adaptation process using distributed services. After an upstream node fails, one of its downstream nodes sends a failure indication to the adaptation server with the event features causing an immediate increase of interested packets and hardly any increase of data packets. The adaptation server then contacts the local lookup server which is registered with all substitute nodes in the vicinity. The adaptation server chooses a suitable candidate and informs the composition server with its decision. All these actions cause a slight increase of the interested packets. This implies that the distributed service respond and react to the node failure quickly and efficiently. The composition server then sends an indication to the substitute, explaining the resumption of the data packet flow. It is a significant difference that the system recovery time for directed diffusion is 14 s as compared to 6 s for the distributed services. The distributed services combined with the region filter can also result in a controlled flooding of exploratory packets, which can be found from the lower interested packet level in Figure 7(b). The flooding control, efficient, and quicker reaction to the node failure surely leads to the deduction of total transmission energy.

To check the efficiency of distributed service clustering, we test the adaptation process with 3 groups, each of which includes 10 nodes, as illustrated in Figure 8(a) and (b). It is clear that the adaptation to the dynamic by the distributed services results in a considerably smaller recovery time of 16 s compared with the directed diffusion of 31 s. Applications taking advantage of the distributed services can respond to the node failure quicker. Directed diffusion tries to deal with loss of data by flooding the interested packets over the entire network until an alternate is found. It takes a long time for the directed diffusion to detect the node failure event, as shown in Figure 8(a), interest packets increase gradually as data packets increase during the recovery time. Almost half of the recovery time, about 15 s, has been taken for the failure detection; the other 16 s to find the alternate node.

Figure 7 Failure recovery time vs. average number of packets for one group: (a) standard directed diffusion and (b) distributed services



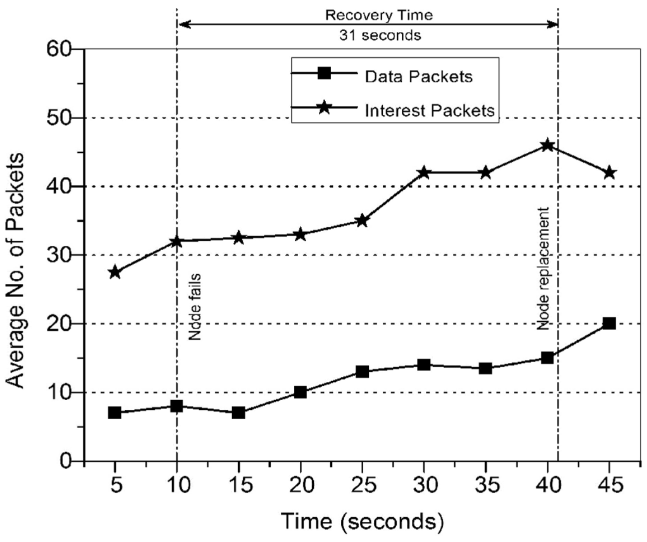
(a)



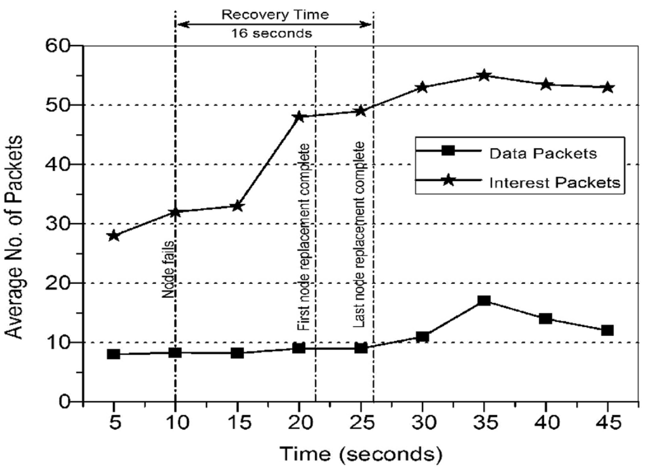
(b)

For the distributed services, one indication to the adaptation server about node failure is sent as soon as failure is detected. As shown in Figure 8(b), it takes about 5 s to detect the failure event, much quicker than the directed diffusion. It is obvious that the clustering architecture of distributed network performs very well. Similar to Figures 7(a) and 8(a), during the recovery time, the data packet hardly increases in Figure 8(b), which also limits the network traffic during the time. The entire recovery process can be carried out by the distributed services using controlled flooding which results in a much shorter recovery time and less energy cost. There is slight increase of the interested packets due to the communication among the services including the messages sent from the failure node's downstream nodes to the adaptation server. During the recovery time, there are fewer interest packets in the whole network for the distributed services. It takes less time to get recovered from the node failure. In general, this mechanism is more energy efficient and robust than the directed diffusion.

Figure 8 Failure recovery time vs. average number of packets for three group: (a) standard directed diffusion and (b) distributed services



(a)



(b)

5.4 *Evaluation for larger Scale*

Larger scale mainly means bigger network size. Larger scale is a key issue in the WSN system design since when the network size becomes big, some of the system performance goes down. However, most WSN applications require the system to be scalable and stable. These are the reasons we make efforts to test the scalability.

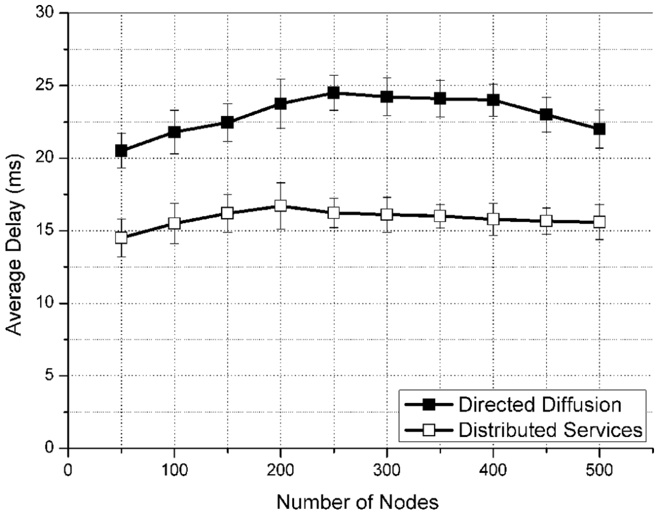
In the ISEE testing environment, a simple client-server target tracking event detection application runs over directed diffusion. Emulation runs are conducted to compare the application that takes advantage of the distributed services and the same application without running the distributed service but using directed diffusion directly. There are 10 randomly generated sources and sinks pairs for each experiment.

In order to study the performance in larger scale, various sensor fields with different numbers of nodes are generated. We study ten different sensor fields, ranging from 50 to 500 nodes in increments of 50 nodes each time. Each result is the average value of 30 independent tests. The sensor fields are generated by randomly placing the nodes in a 200m by 200 m square area. Each node has a radio range of 50m. Each source node detects the target and creates one event every second. Each event is modelled as constant 512 byte packet and sent out every second. Interest packets are periodically created every 5 s with duration of 15 s. The negative reinforcement is set to be 2 s.

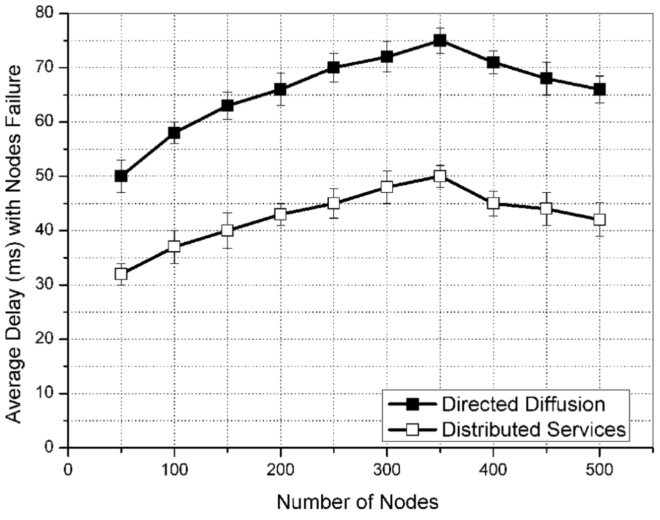
The first metric to evaluate the performance is *average delay*, which measures the average one-way latency observed between transmitting an event and receiving it at each sink. The second metric is *average packets delivery ratio,* which is the ratio of the number of distinct events received to the number originally sent. The third metric is *average dissipation energy* in terms of average number of hops traversed by the message from the source to the sink, which is proportional to the total amount of energy consumed for the data delivery. We study these metrics as a function of network size. In order to test the real robustness of the system, we compare the performance of each metric under the condition of without nodes failure and with nodes failure. In the nodes failure testing, our dynamics experiment imposes fairly adverse conditions for a data dissemination protocol. At any instant, 20% of the nodes in the network are unusable.

Figure 9 shows the average delay as function of network size with and without nodes failure. As shown in Figure 9(b), the curves of both protocols intend to be increase and then decrease. When network size reaches certain threshold 350, network nodes seem to be saturated, the protocol can always find the route and 20% nodes failure does not influence the average delay. This is very distinct in the node failure case (in Figure 9(b)). As shown in Figure 9(a), the distributed service, the number of nodes change does not influence the average delay very much – the average delay approximately changes from 15 ms to 18 ms. This implies that the average delay of distributed services is not sensitive to the node densities. It is obvious that the distributed services have a noticeable advantage over the directed diffusion in terms of average delay even under the condition of 20% nodes failure. Even though directed diffusion makes the latency as the only metric to determine the route, its inherent strategy is to depend on interest flooding to find the substitution for failure replacement. Flooding interest not only slows down the messages propagation, but also reduces the energy efficiency. This is the fundamental problem of directed diffusion. Contrary to it, the distributed servers cooperate with each other, prune off higher latency paths, and find another service provider in the similar location to the failure node. It is encouraging to see that directed diffusion always suffers from higher average delay compared with the distributed services. There is noticeably better average delay as can be found from Figure 9. Without node failure their difference ranges approximately 30–60% less average delay, with nodes failure ranges approximately 50–60% less average delay. This feature of distributed services is very appropriate for real time WSN applications.

Figure 9 Impact of number of nodes on average delay: (a) average delay without failure and (b) average delay with 20% failure



(a)

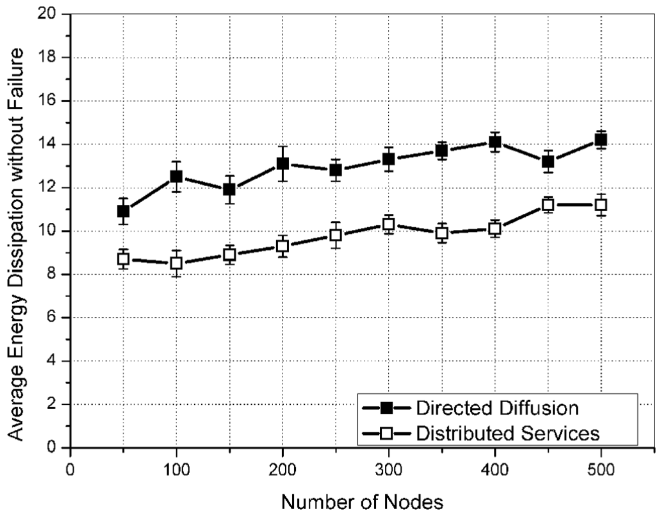


(b)

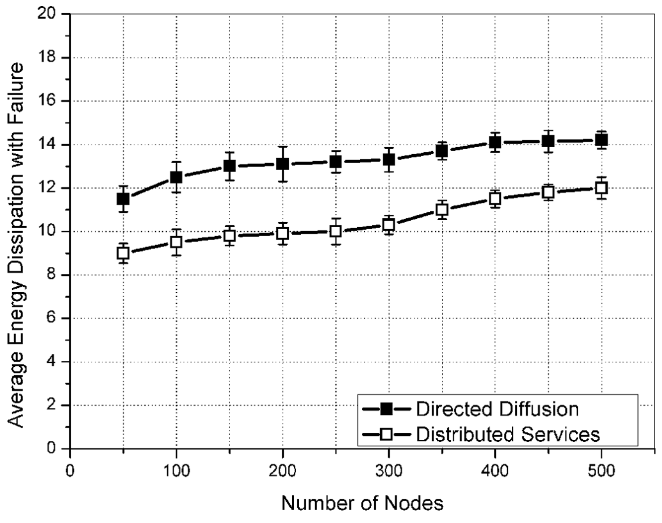
Figure 9 shows that when network size increases, the average energy dissipation does not increase very much for both protocols under the adverse condition. Even under such adverse condition as 20% node die, it is prominent that the distributed services are more energy-efficient than directed diffusion. In an uncongested WSN without obstructions, lowest delay path is also the shortest path and most energy-efficient path. Thus, the curve (Figures 9 and 10) implies that under the same situation, the distributed service can achieve shorter delay, lower number of hops, as well as less energy cost. These results coincide with the average delay demonstrated above. It is interesting to see that when 20% nodes die, the average energy cost for both protocols does not increase dramatically compared with the average energy cost without nodes failure. Both protocols pay attention when dealing with dynamic conditions and actions can be taken effectively without delay as long as failure is detected. The directed diffusion builds backup path (reinforcement) at the flooding stage and avoids network-wide flooding to repair the failure, although it mainly depends on flooding as the method to dealing with failures. In addition, diffusion benefits significantly from *in-network aggregation*, which greatly reduces the energy cost. It can also keep several reinforced paths alive in normal operation as long as the failure is discovered; the substitution paths can be effective immediately even though they are not high quality paths in terms of delay. Those contribute to the energy efficiency of directed diffusion and improving the energy dissipation of the directed diffusion. Distributed services can achieve approximately 20–0% more energy saving than directed diffusion even under the worst case. The distributed services have embedded dynamic treatment mechanisms with the prompt cooperation of three servers, which can hardly produce extra flooding. They keep location information of each service provider, which not only helps to find the optimal route but also find the optimal replacement node. When nodes fail, there is no extra energy consumed and no broadcast for the replacement. These facts determine the results that energy cost does not increase greatly in the worst-case scenario. In addition, the distributed services are built on the directed diffusion and have the features of both the directed diffusion and their own. This qualified the distributed services to win even in the worst case as 20% nodes die. This is very appropriate for the resource-constrained WSN.

We can tell from Figure 11 that the distributed services can achieve more satisfactory average packet delivery ratio than directed diffusion. Both schemes incurred an average packets delivery ratio of nearly one as there is no failure, since these experiments ignore network dynamics and are congestion-free. Without node failure, the difference between the two approximately ranges 5% to 10%, while with nodes failure the difference is more noticeable, which ranges approximately from 20% to 30%. This verifies the fact that the distributed services not only can maintain real time and energy efficiency, but also can guarantee the quality of transmission. In addition, the distributed services have embedded mechanisms to keep all the service providers in record and replace the failure node just after the failure detection without delay. The quality of transmission is only slightly influenced during the time. The analysis above indicates that the distributed services have higher degree of adaptation. Even under the dynamic conditions, they can keep noticeably better average packets delivery ratio without user interventions.

Figure 10 Impact of number of nodes on average energy dissipation: (a) average energy dissipation without failure and (b) average energy dissipation with 20% failure



(a)

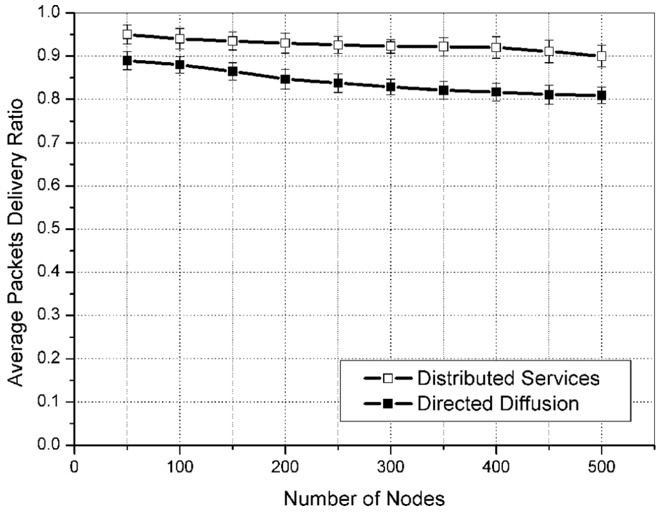


(b)

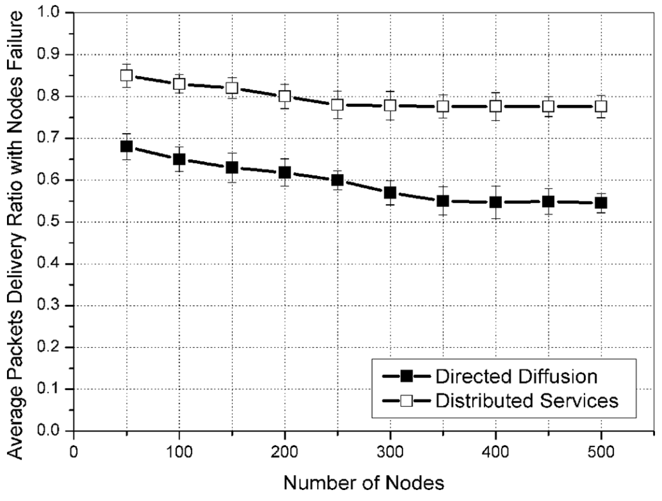
6 Conclusions

In this paper, we proposed the distributed services as a WSN middleware, and implemented the composition service as its key component. This simplifies and facilitates the development of WSN as well as helps the network adapt to the dynamics. It has bridged the gap between complex applications and the large-scale resource-constrained WSNs. The distributed middleware tailors the features of WSN to the applications' requirements and makes the development easier and efficient. Our simulation by ISEE shows that the composition service collaborating with other two services can achieve robustness, reliability, scalability, real time, and adaptation. It incurs remarkably better performance in terms of the failure recovery time, network flooding avoidance, average delay, energy dissipation, and packets delivery ratio compared with the original directed diffusion protocol.

Figure 11 Impact of number of nodes on average packets delivery ratio: (a) average packets delivery ratio without failure and (b) average packets delivery ratio with 20% failure



(a)



(b)

Our final goal is to implement the distributed services in the real-world scenario for the target tracking application. This work is the full preparation for the future implementation. In the future, we will also improve the distributed service from network security aspect. We will begin with discovering the fatal threats among the three services, especially under the dynamic conditions with larger scale. The threats compromising the message exchanges and degrading the overall performance will be found out. The countermeasures will be proposed accordingly.

Acknowledgements

We thank Udayan Naik for his help in this study. This work was supported in part by NSF China under grants No. 61202379, No. 61372085 and Doctoral Fund of Ministry of Education of China (No. 20120032120041).

References

Afzal, S.R., Huygens, C. and Joosen, W. (2009) 'Extending middleware frameworks for Wireless Sensor Networks', *Proc. of the International Conference on Ultra Modern Telecommunications*, 12–14 October, 2009, St. Petersburg, Russia, pp.1–7.

Hadim, S. and Mohamed, N. (2006) 'Middleware for wireless sensor networks: a survey', *Proc. of Communication System Software and Middleware*, January, New Delhi, India.

Hadim, S. and Mohamed, N. (2006) 'Middleware: middleware challenges and approaches for wireless sensor networks', *Distributed Systems Online, IEEE*, Vol. 7, No. 3, pp.1–1.

Haghighi, M. and Cliff, D. (2013a) 'Multi-agent support for multiple concurrent applications and dynamic data-gathering in wireless sensor networks', *Proc. The Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, July, Taiwan.

Haghighi, M. and Cliff, D. (2013b) 'Sensomax: an agent-based middleware for decentralized dynamic data-gathering in wireless sensor networks', *Proc. The 2013 International Conference on Collaboration Technologies and Systems*, May 2013, San Diego, USA.

Intanagonwiwat, C., Govindan, R. and Estrin, D. (2000) 'Directed diffusion: a scalable and robust communication paradigm for sensor networks', *Proc. The 6th Annual International Conference on Mobile Computing and Networking (Mobicom'2000)*, Boston, MA, USA.

Intanagonwiwat, C., Govindan, R., Estrin, D., Heidemann, J. and Silva, F. (2003) 'Directed diffusion for wireless sensor networking', *IEEE/ACM Trans. Networking*, Vol. 11, No. 1, pp.2–16.

Ivester, M. and Lim, A. (2006) 'Interactive and extensible runtime framework for execution and monitoring of sensor network services', *First International Conference on Communication System Software and Middleware*, January, New Delhi, India.

Khedo, K.K. and Subramanian, R.K. (2009) 'A service-oriented component-based middleware architecture for wireless sensor networks', *International Journal of Computer Science and Network Security*, Vol. 9, No. 3, pp.174–182.

King, J., Bose, R., Yang, H-I., Pickles, S. and Helal, A. (2006) 'Atlas: a service-oriented sensor platform: hardware and middleware to enable programmable pervasive spaces', *The 31st IEEE Conference on Local Computer Networks*, November 2006, Tampa, FL, USA.

Laukkarinen, T., Suhonen, J., Hämäläinen, T.D. and Hännikäinen, M. (2011) 'Pilot studies of wireless sensor networks: Practical experiences', *Proc. Conference on Design and Architectures for Signal and Image Processing*, November 2011, Tampere, Finland.

Laukkarinen, T., Suhonen, J. and Hännikäinen, M. (2012) ‘A survey of wireless sensor network abstraction for application development’, *International Journal of Distributed Sensor Networks*, pp.1–12.

Li, S., Lin, Y., Son, S.H., Stankovic, J.A. and Wei, Y. (2004) ‘Event detection services using data service middleware in distributed sensor networks’, *Telecommunication Systems*, Vol. 26, Nos. 2–4, pp.351–368.

Lim, A. (2001) ‘Distributed services for information dissemination in self-organizing sensor networks’, *Journal of the Franklin Institute-Engineering and Applied Mathematics*, Vol. 338, No. 6, pp.707–727.

Lim, A. (2002) ‘Support for reliability in self-organizing sensor networks’, *Proc. 5th International Conference on Information Fusion*, July 2002, Annapolis, Maryland, USA.

Mohamed, N. and Al-Jaroodi, J. (2011) ‘Service-oriented middleware approaches for wireless sensor networks’, *Proc. 44th Hawaii International Conference on System Sciences*, January, Manoa, Hawaii, USA.

Mottola, L. and Picco, G.P. (2011) ‘Programming wireless sensor networks: fundamental concepts and state of the art’, *ACM Comput. Surv.*, Vol. 43, No. 3, pp.1–51.

Nan, H., Ning, Y. and Yi, G. (2009) ‘Research on service oriented and middleware based active qos infrastructure of wireless sensor networks’, *Proc. 10th International Symposium on Pervasive Systems, Algorithms, and Networks*, December, Kaohsiung, Taiwan.

Radhika, J. and Malarvizhi, S. (2012) ‘Middleware approaches for wireless sensor networks: an overview’, *International Journal of Computer Science Issues*, Vol. 9, No. 6, pp.224–229.

Sakthidharan, G.R. and Chitra, S. (2012) ‘A survey on wireless sensor network: an application perspective’, *Proc. 2012 International Conference on Computer Communication and Informatics*, January, Coimbatore, India.

Shen, Q. (2002) *A Distributed Composition Service for Self-Organizing Sensor Networks*, Thesis for Master, Department of Computer Science and Software Engineering, Auburn University.