Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2012, Article ID 567687, 15 pages
doi:10.1155/2012/567687

Research Article

Interconnection between IP Networks and Wireless Sensor Networks

Brandon Keith Maharrey, Alvin S. Lim, and Song Gao

Computer Science and Software Engineering, Auburn University, Auburn, AL 36849, USA

Correspondence should be addressed to Alvin S. Lim, lim@eng.auburn.edu

Received 15 June 2012; Revised 16 October 2012; Accepted 17 October 2012

Academic Editor: Chin-Feng Lai

With applications ranging from environmental and health monitoring to military surveillance and inventory tracking, wireless sensor networks (WSNs) are changing the way we collect and use data and will be a major part of our technological future. The decreased manufacturing cost of these small devices has made it reasonable to deploy many sensor nodes—tens to thousands and more—over large and small indoor and outdoor areas for sensing tasks. With this increase in density data-gathering problems come. It would be useful if an IP-based host could collect information from multiple remote data-centric networks via transparent communication among sensor nodes and IP-based hosts, using a common application programming interface (API). Two APIs are presented for efficiently producing data in WSN and retrieving the data from an IP network. An associated WSN middleware layer called dynamic service is used to effectively interconnect these two APIs. These three components work together in harmony to enable IP-based hosts to gather sensed data from one or more remote WSNs through application-layer gateways which provide seamless interconnection between different remote WSNs and the IP network.

1. Introduction

A wireless sensor network (WSN) mainly consists of many independent, low-power, low-cost devices capable of sensing, processing, and wireless communication [1]. Their main purpose is to collect and disseminate environmental data and possibly perform some calculations [2]. There has been a push, especially in industry, in recent years to make real-time data collected from WSNs more readily available to consumers of this information. However, there are no convenient tools or specific frameworks in place to allow instant access to this sensed information in a programming environment. Thus, one of the main problems with deploying WSNs is gathering the data they produce and using it in flexible ways. This paper provides a solution to this problem by enabling seamless interconnection between IP networks and wireless sensor networks.

This paper proposes a middleware layer, called dynamic service, that provides transparent communication between IP-based hosts and sensor nodes via gateway nodes, which are employed as access points for the purpose of interoperable information retrieval from WSNs. It also eases the implementation of sensor network applications by providing a standard common interface to the data-centric WSN networking protocol and IP-based applications alike. Dynamic service (DS) is placed between application layer and direct diffusion (DD) on sensor nodes. By exposing neat and clean APIs, it allows the sensor node application programmer to ignore the details of the DD networking protocol, but only cares about data they are producing or processing. In addition to easing programming, DS allows tasking of nodes. Applications register the name of data they can provide with the DS service layer. An application sleeps until it receives an interest for this registered named data. This conserves energy in information or data production.

Other parts in the system include an implemented gateway application that bridges between IP networks and sensor networks, and a set of APIs for external agents (EAs), which are IP-based hosts that are not a part of the sensor network. EA API enables IP network applications to submit requests to and receive data from sensor networks. By using DS API and EA API, programmers can build efficient applications running across wireless sensor networks and IP networks without concerning complicated communication

details, such as translating between IP packets and DD packets, and managing DD protocol details.

The following sections of this paper will be organized as follows. Section 2 describes the problem and challenges in IP enabled WSN and outlays the motivation of this research. Section 3 discusses relevant approaches to gathering data from WSNs connected to IP network. In Section 4, an overall summary of the work is given, including architecture of the system, proposed dynamic service, and a discussion of directed diffusion. In Section 5, design details of dynamic services, external agent role and gateway role are given. Section 6.1 and Section 6.2 describe the experimental environment and platform used in this paper, and the applications implemented on the system, respectively. In Section 7, experimental results are given, in terms of application level result, application lines of code metric, and dynamic service performance. A more detailed presentation of this research can be found in [3].

2. Research Motivations

The area of sensor networking applications is exploding rapidly. In the recent past, many new sensor networking applications have surfaced in the literature, and most notably among them are wildlife habitat monitoring [4], forest fire detection [5], alarm systems [6], and monitoring of volcanic eruptions [7]. These scenarios involve many unique issues and challenges in addition to the problem of gathering this sensed data in real time for analysis, computation, or storage. IP-based application programmers are faced with the difficult problem of interconnecting sensor networks with IP-based hosts. Therefore, a main motivation of this research is to ease the data-gathering problems that IP-based programmers face when gathering data from one or more remote WSNs.

There are subtle problems when interconnecting IP networks and WSNs. First, there are major differences in the data retrieval paradigm used in WSNs and IP networks, where WSNs use data-centric paradigm for retrieving sensor data, whereas IP networks use host-centric paradigm for querying individual sensor node. WSNs use data-centric retrieval methods since they have been shown to be more energy-efficient and scalable than address-centric retrieval methods used in IP networks which are based on proactive routing algorithms that have been shown to be less energy-efficient. Second, for an IP network application programmer, it is relatively difficult to develop programs for querying remote sensor nodes for their data due to the differences in data retrieval paradigms. Third, a mechanism must be designed to enable IP-based hosts to actually retrieve data from remote WSNs. Named data coming from data-centric WSN needs to be properly translated to an address-centric IP network. Fourth, since the data-centric network discussed in this paper is based on the publish-subscribe paradigm, IP-based hosts must take this fact into account. Subscriptions must be sent from IP-based hosts to WSNs properly for data production to begin.

Solving the above problems will make it possible for us to design and implement IP-enabled WSNs whereby sensor data may be retrieved, and tasks in WSNs can be initiated from IP networks. The main advantage of using this approach is that WSNs can still use more energy-efficient and scalable data-centric retrieval methods for accessing sensor data.

3. Related Work

Many researchers have done previous work on interconnecting wireless sensor and IP networks and gathering data from WSNs. The majority of the techniques, like the one presented in this paper, treat WSNs as a separate entity from the Internet [8]. The techniques are divided into two main approaches: a gateway-based approach and an approach in which all sensor nodes are TCP/IP-enabled—that is, capable of direct, end-to-end communication with IP-based hosts.

3.1. Gateway or Proxy-Based Approaches. The most common approach to connecting a WSN with an IP network is through a gateway or proxy node. In this approach, the gateway node acts as a relay to translate and forward packets from one network to the other [9–15]. The authors of [12] describe two gateway-based approaches: using the gateway as a relay or as a front-end. When the gateway acts as a relay to the WSN, it simply relays any information from the WSN to any registered IP-based host that wants that information. This approach is taken in this paper. When the gateway node acts as a front-end to the WSN, it actively collects and stores data from the WSN in some kind of database that users can query with SQL-like query languages.

One of challenges in gateway-based approaches is that the gateway node can be a bottleneck to the flow of network traffic, especially if a surge of data needs to be transmitted from the gateway node to an IP-based host [14]. An advantage of gateway-based approaches is that the two communication networks are totally decoupled, allowing for specialized and more efficient protocols, such as directed diffusion, to be implemented in the WSN. The gateway node can also act as a mediator for WSN data transmission by implementing security features such as user and data authentication [12].

3.2. IP-Enabled Approaches. Besides gateway-based approaches to interconnecting wireless sensor and IP networks, there also exists IP-enabled WSNs. One of these approaches assumes a full TCP/IP stack on each sensor node. In this approach, the WSN is directly connected to the IP network to enable direct communication between WSN sensor nodes and IP-based hosts [16–20].

The main advantage of using TCP/IP in this way is that there is no need for protocol conversion or gateways. However, the overhead for the full networking stack on an energy-constrained sensing device may be prohibitive, especially when the end-to-end retransmissions incurred by the TCP protocol cause even more undue retransmissions at intermediate nodes. It has been shown that the majority of energy in a WSN is used for wireless communication [21, 22]. Therefore, if one considers the protocol overhead for TCP/IP networks in the context of WSNs, it can be seen

that this overhead is prohibitive. A further disadvantage of this approach is that just because each sensor node is addressable does not necessarily ease the task of gathering data the sensors can produce. In this case, IP hosts must be supplied with each individual WSN node IP address it wishes to query for data. There could potentially be many WSN nodes in possibly multiple remote WSNs so this may be an inefficient method of information retrieval, especially considering the wasted energy with TCP retransmission attempts. Moreover, this solution does not lend itself well to specialized and energy-efficient WSN protocols inside the WSN. This approach also uses IP routing algorithms which are proactive and less energy-efficient than reactive routing algorithms, such as directed diffusion, used in WSNs.

3.3. Overlay Approaches. In overlay approaches, gateway nodes are used to interconnect WSNs with IP networks and assign virtual identification information to either IP-based hosts, sensor nodes, or both [23–25]. According to [8], overlay approaches come in two basic forms: sensor network overlay IP network and IP network overlay sensor network. These two approaches employ application layer gateways through which the WSN is identified and information is passed.

In the sensor network overlay IP network approach, IP-based hosts are required to register with the WSN application-layer gateway node and be assigned a virtual sensor node ID by the gateway node. Once a packet from a sensor node destined for a virtual sensor node ID reaches the gateway node, the gateway node encapsulates the whole packet into a TCP or UDP and IP packet, while the IP-based host communicates with sensor nodes by supplying the sensor node ID to the gateway node.

In the IP network overlay sensor network, sensor nodes are required to register with the WSN gateway node and are assigned a virtual IP address. Individual sensors themselves do not actually possess an IP address in the WSN. Sensor nodes are instead assigned a WSN-wide unique standard 16-bit TCP/UDP port number by the gateway node. IP-based hosts communicate with individual WSN nodes by supplying the IP address of the gateway node and port of the sensor node with which it wishes to communicate.

This scheme has several issues. Firstly, if the standard 16-bit unassigned TCP/IP port numbers are used to identify individual sensor nodes, only around 16,000 nodes can be uniquely addressed. Secondly, it suffers from the protocol overhead attributable to TCP/IP. Thirdly, it may suffer from large routing tables due to the fact that the gateway node must keep track of two different mappings. Aside from these issues, neither of these two overlay approaches truly simplifies the task of gathering data from WSNs.

3.4. 6LoWPAN and IEEE 802.15.4 Standards. The IEEE 802.15.4 standard [26] defines the physical layer and media access control for the wireless personal area network. The 6LoWPAN standard [27] defines encapsulation and header compression mechanisms that allow seamless IP integration over IEEE 802.15.4 [28–30]. [31] presents uIPv6, an IPv6 stack for memory-constrained devices that can run over IEEE 802.15.4/6LoWPAN. In 6LoWPAN, individual sensor nodes are addressable with standard IPv6 IP addresses without the overhead of sending full IP addresses when routing messages inside the WSN. This is because a gateway node connected to an IP network maps full IP addresses into 16-bit node IDs for more efficient bandwidth usage along wireless hops.

This standard, however, is only in its preliminary stages and thus will probably undergo more changes before the final standard is widely available. Like other IP-enabled approaches to interconnecting WSNs, this approach also requires that IP-based hosts know the specific IP addresses of sensor nodes with whom they wish to gather data. Further, the IEEE 802.15.4 standard defines a maximum bandwidth that may be unsuitable for WSN applications requiring larger bandwidths. Moreover, these standards together or separately do not necessarily ease the task of gathering data from sensor nodes in one or more WSNs, although it does reduce the amount of wasted energy with respect to transmitting end-point identification information for each packet along every wireless hop.

4. System Overview

4.1. Architecture. Our system uses an approach for providing seamless interconnection and transparent interoperability between different sensor data dissemination paradigms of IP and WSNs via gateways which also decouple the IP networks and WSNs, allowing for specialized and more efficient protocols to be implemented in WSNs [12]. Figure 1 shows the network architecture. There are three node roles in the network.

(i) *External Agents*: An IP-based host that is not a part of the WSN is termed an *external agent* (EA). An EA node is a full-fledged computer with full TCP/IP networking stack and can access the Internet or IP network.

(ii) *Sensor Nodes*: Sensor nodes provide data that is requested by EA. Given the attributes of robustness, scalability, and energy-efficiency in multi-hop communication [32], *directed diffusion* (DD) is used as routing protocol within sensor nodes. A middleware layer, *dynamic service* (DS), is designed on top of DD. DS will be introduced in Section 4.2.2.

(iii) *Gateway Node*: As shown in Figure 2, gateway node is on the boundary of WSN and directs incoming and outgoing traffic of WSN. It has both IP network stack and DD sensor network capability. Any interest or subscription from an IP-based host is processed through the gateway node, translated into WSN interest, and disseminated using DD protocol. Data returned by sensor nodes is also processed and forwarded by gateway node to IP-based hosts.

4.2. Underlying Concepts

4.2.1. Direct Diffusion. Directed diffusion (DD) is a data centric ad hoc networking protocol capable of robust,

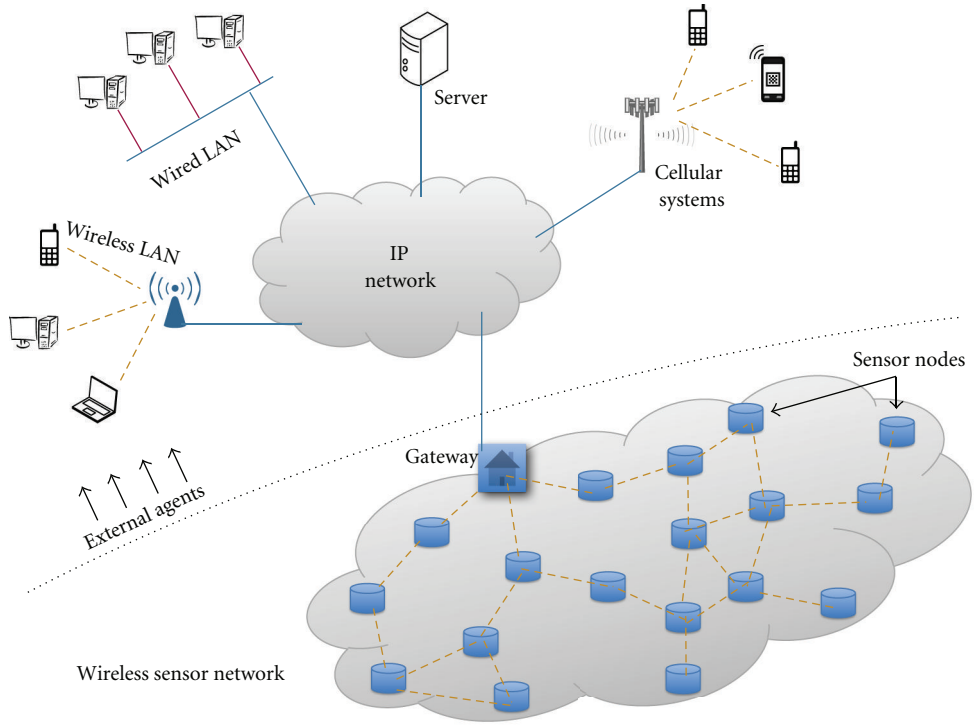


Figure 1: Network Architecture.

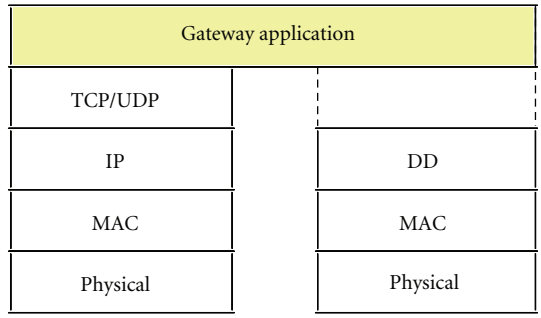
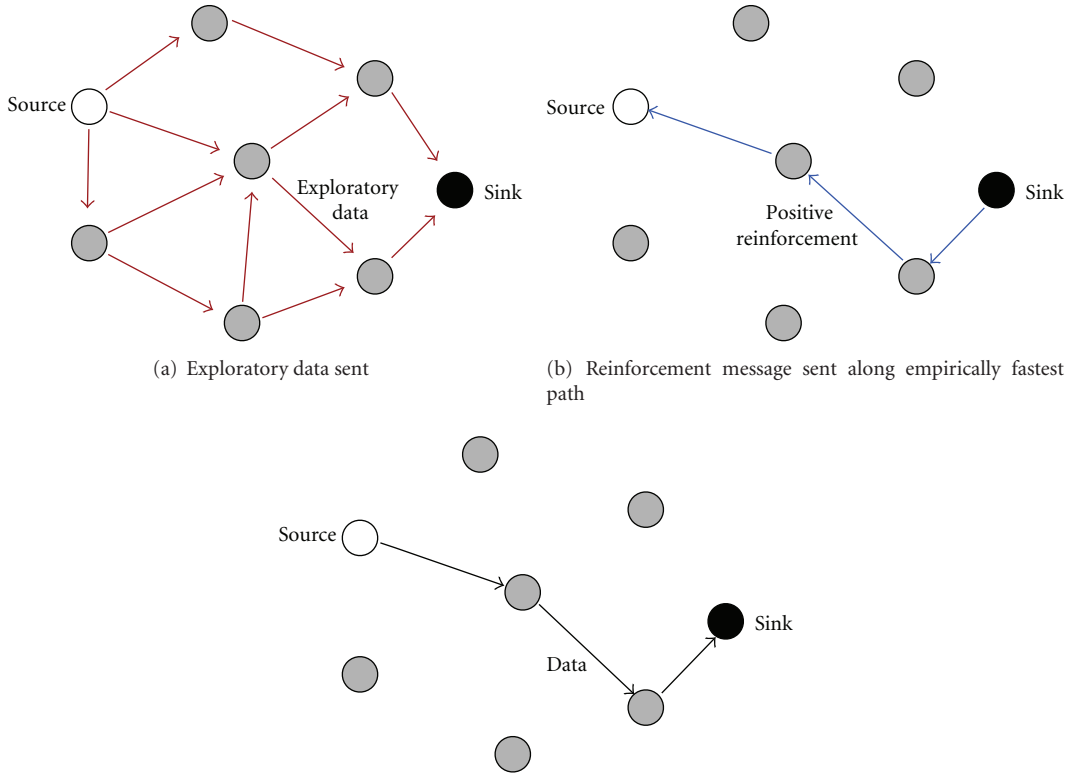


Figure 2: Gateway Networking Stacks.

scalable, and energy-efficient multi-hop communication [33, 34].

In DD, when a sensor node has the ability to produce named data, it specifies the name of this named data to the core DD routing algorithm. When the DD core receives an interest for named data that has previously been registered, a callback function is invoked to handle the production of data corresponding to this named interest. On the other hand, when the DD core inside the sensor node receives a named interest for named data that has not been previously registered, the DD core will either forward the interest message to its neighbors or drop the interest message altogether. When a node is interested in some data, it sends out an interest for that data. This node is referred to as *sink*. Interests are diffused throughout the network, and *gradients* are set up along the reverse path of travel of the interests.

Figure 3 shows what happens after the interest is diffused throughout the network. Once a sensor node receives an interest for data it can produce, this node, known as *source*, begins to produce that data. The data is forwarded hop-by-hop along multiple gradients and back toward the *sink*, establishing an empirically fastest path from the sink node to the source of the data, by way of exploratory data as shown in Figure 3(a). This exploratory data diffuses back across the network along the *gradients*. The empirically fastest path is chosen for reinforcement by the *sink* for fast data



(a) Exploratory data sent

(b) Reinforcement message sent along empirically fastest path

(c) Data delivered along reinforced path

Figure 3: Directed diffusion data diagram.

reception of future data packets, and the *sink* transmits a positive reinforcement message to the neighbor from which exploratory data was first received, as shown in Figure 3(b). The positive reinforcement message is sent along the gradient path until it reaches the source node. From this time onward, data is sent along the positively reinforced path, shown in Figure 3(c).

DD is chosen as the data-centric networking protocol in the WSN in this paper for several reasons.

(i) DD makes routing of the sensed data within the WSN more efficient compared to using TCP/IP on the sensor nodes. The responsibility of passing interests from the gateway node to individual sensor nodes, and subsequent passing of data from sensor nodes back to the gateway node, is given to DD. This makes the task of routing inside the network very simple, convenient, and efficient.

(ii) DD keeps routing tables inside the gateway node rather small. At the gateway node, it is only necessary to keep track of unique interests (subscriptions) and any IP address and port number of IP-based hosts interested in data provided by the WSN.

(iii) Because individual sensor nodes are not named in a DD network, there is no overhead in keeping track of or assigning sensor nodes unique IDs nor is there any sensor node energy wasted in transmitting endpoint identification information with each transmitted packet.

4.2.2. Dynamic Service. Dynamic service (DS) is a middleware layer built on top of DD protocol on sensor nodes, as shown in Figure 4. It provides the services necessary to facilitate IP-based information retrieval from sensor node applications built with this type of architecture. With DS inserted between DD and application layer, some WSN specific concepts are hidden from upper layer and communication details are made transparent, thus providing a more flexible and efficient way to develop WSN applications.

DS enables nodes tasking, specifically for data production. Applications register the name of data they can provide with the DS service layer. An application sleeps during normal status. When an interest for the registered named data is received by DS, DS awakens the node application to begin producing the corresponding data and send out through DS.

5. Design Details

Our design for seamless interconnection between IP networks and WSNs involves three main components: dynamic service, external agents, and gateways.

5.1. Dynamic Service. DS moves the complexity out of the individual sensor node applications and into the DS service

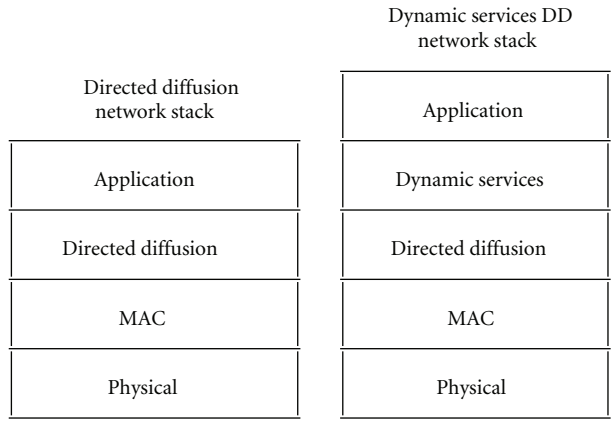


Figure 4: Network stack: pure DD versus DS enabled DD.

layer. DD and DS service layer must be running on the sensor node before applications are able to use the services provided by DS. The DS API functions available to the sensor node application are called within the sensor node application and information is passed from the sensor node application, through a message passing interface (MPI), to the DS service layer.

5.1.1. DS Architecture. As shown in Figure 4, DS is implemented on top of DD. It acts as a regular DD application; therefore, it contains a tasking thread and a main thread. However, since DS is a intermediate layer between DD and actual sensor node applications, the threads function differently from general DD applications.

(i) *Tasking Thread*: Instead of incrementing or decrementing value of shared variable, the tasking thread tasks sensor node applications through their respective message queues, when a tasking message is received from DD. In addition, it also passes data to the appropriate sensor node applications when a data message is received from DD.

(ii) *Main Thread*: Instead of polling shared variable for tasking, the main thread waits at message queue for requests or data from sensor node applications.

Figure 5 shows a diagram of the types of messages in which the DS service layer and the DS API communicate with each other. Although the messages are passed through MPI, the sensor nodes are unaware of communications details. They are encapsulated within DS API.

The *register* and *subscription* messages sent to DS through the DS API through DS's own message queue enter information into internal tables in which DS maintains to keep track of sensor node applications. These tables retain information regarding the data type each sensor node application produces and the data types to which the sensor node applications are subscribed as well as the message queue information of each sensor node application through which DS sends response or data to the sensor application.

5.1.2. Data Producers. There are two different classes of sensor node applications. One class of sensor node application, when tasked, simply produces the requested named data. These sensor node applications are called *simple producers*. The other class of sensor node applications, however, depends upon other named data types in the local WSN in order to produce its named data. These sensor node applications are called *complex producers*.

Simple Producer. When viewed from the network level, the simple producer messaging process looks like that shown in Figure 6. Figure 7 shows how messages are communicated between the DS API and the DS service layer for simple producers. The communication between DS API and DS is described in 4 phases as follows.

(1) *Registration*: Sensor applications register their intent to publish data. This registration message is passed to the DS service layer through the DS API, reaches the DD core, and a registration response message is eventually passed back to the sensor application through the DS service layer and through the DS API. If this registration response indicates a successful registration, the sensor application begins to await tasking.

(2) *Data Interest*: When an interest message flooding the DD network arrives at the sensor node at the DD core, the DD core realizes that a registration for this data type has been received in the past and invokes the tasking thread in DS. The tasking thread in DS then tasks the sensor application through the DS API.

(3) *Production Data*: The sensor application begins producing the named data it was programmed to produce. Once the sensor application has data to send out onto the network, it publishes this data through the DS API, through the DS service layer. The DD core, at last, actually sends the data out onto the network. A publish success message is propagated up to the sensor application through the DS service layer and through the DS API.

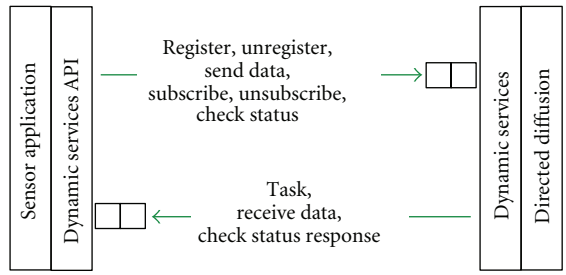
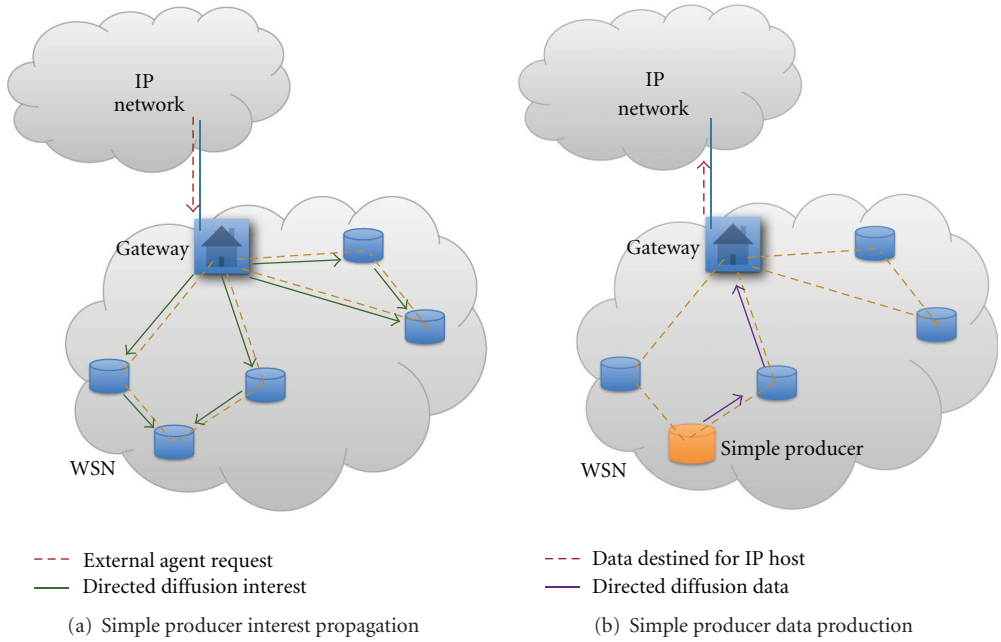


Figure 5: Types of messages that DS service layer and DS API communicate with each other.



(a) Simple producer interest propagation

(b) Simple producer data production

Figure 6: How data production takes place when a sensor node application known as a simple producer is tasked.

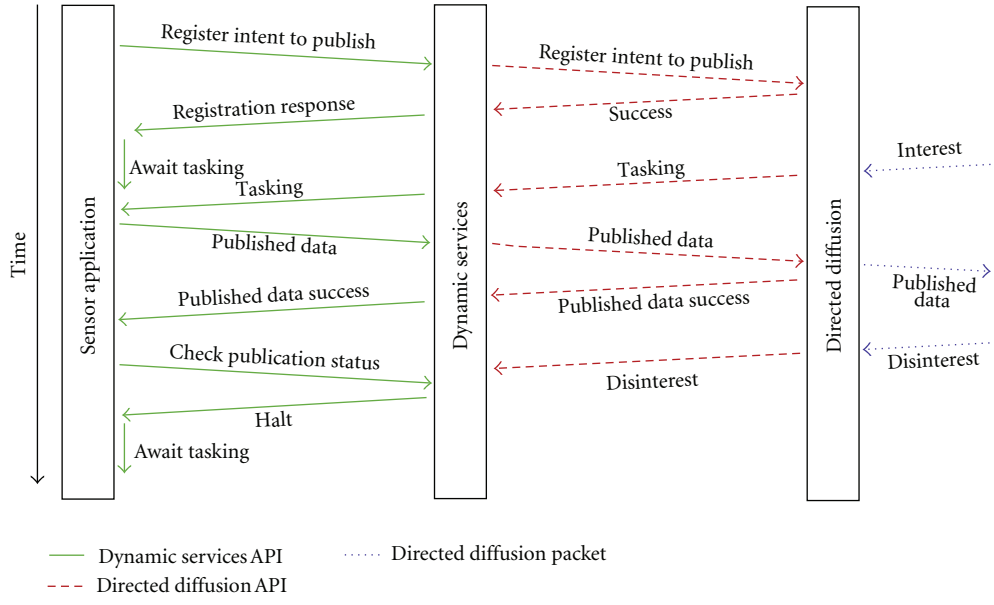


Figure 7: How messages are exchanged for simple producers.

(4) *Checking Status*: Ever so often, the sensor application should check whether it should continue producing the data it was once tasked to produce. The sensor application submits a message to the DS service layer requesting its status to continue producing data. If a disinterest message was received from the DD network before this request is made, the DS service layer replies to the check publication status message with a message indicating that data production should cease. The sensor application should then halt data production and continue to await tasking. The whole simple producer process repeats thus, henceforth.

Complex Producer. Unlike simple producers, when complex producers are tasked, they require named data produced by other sensor node applications within the local WSN in order to produce their named data.

Sensor applications first register with DS service layer their intent to publish data. When an interest message flooding the DD network arrives at the sensor node at DD core, DS layer tasking thread is invoked. DS then tasks the sensor application. Unlike a simple producer, the complex producer subscribes to other data types as well. The subscription is done through the DS API. When the required named data from other producer is received and passed up through DD core and DS service layer, the complex producer begins to produce its named data. Finally, a disinterest may be received to untask the complex producer.

5.2. External Agents. An external agent (EA) is any IP-based host not directly connected to the WSN. These devices are decoupled from the WSN protocol and are not a part of the WSN.

One of the purposes of the system is that sensor nodes do not expend energy on any sensing or transmission task until an EA submits an interest to the WSN. Therefore, EAs actually drive the activities of the WSN sensor nodes by sending requests to the gateway node. This is achieved by registration/subscription process described in Section 5.1. Sensor node applications register a data type it can produce through DS API and keep sleeping until they receive tasking information from some EA.

A simple API enables an EA to retrieve information from a DS enabled DD WSN. There are three functions provided by the API.

(1) *Subscribe*. Through this function, an EA initiates a subscription request to one or more WSNs by providing the address of gateway nodes as well as the types of data it is interested in. The EA API then interprets the subscription request and passes them to proper gateway nodes in WSNs, to be translated into WSN DD interests and disseminated within the WSNs. The function then returns a socket descriptor to EA application reference in the future.

(2) *Receive*. After *subscribe* is called, an EA can retrieve data from WSN through the *receive* function with the socket descriptor returned by *subscribe* function. It waits for the network until the requested data is available, interprets the received data, and stores them in an indicated buffer. When the function returns, the requested data is available in the buffer for EA applications to use.

(3) *Unsubscribe*. After the EA is no longer interested in a particular type of data from one or more WSNs, it can call this function with the socket descriptor and the data disinterested. The EA API passes the unsubscription to gateway nodes. Once no EAs are interested in a particular type of data, the gateway node (see Section 5.3) sends a disinterest into the WSN so that the named data is no longer being produced.

5.3. Gateway Role. The gateway role is the entity which physically enables communication between IP network and WSN. When the gateway node powers on, it waits for requests from EA in its main thread. When it receives a request from an EA, it translates this request into an interest packet that the DD networking protocol can understand. As this interest is diffused throughout the network, gradients are set up along the reverse path of this interest propagation. Named data later produced by a sensor node will traverse the network along these gradients, and the gateway will eventually positively reinforce gradients with empirically shortest delay.

In this way, named data is drawn towards the gateway node for which the gateway node previously sent out interest requests on behalf of EA. If the gateway node receives data for which there is no EA subscribed, the gateway node simply discards the packet. This could happen if the gateway node receives data from a sensor node that has not yet received the command to stop producing data.

The gateway node is a regular directed diffusion application which sits on the boundary of the wireless sensor and IP networks. The gateway nodes main thread, upon starting up, prepares an incoming socket on which to receive *requests* from EAs. The *request* structure is totally hidden from EAs therefore, the only thing that EAs need to know is the IP address and port number of the gateway node, the name of the named interest, the structure of the expected data, and how to use the EA API.

When a *request* is received from the an EA in the gateway nodes main thread, an entry is added to a local map structure. This map structure contains the IP address and port number on which the EA is awaiting named data. When named data arrives at the gateway node from the WSN, the DD core triggers the gateway nodes tasking thread. However, the tasking threads role in the gateway node has been redefined. Rather than using the tasking thread for tasking, the gateway nodes tasking thread looks into the IP address-port number map structure to determine to which EA(s) to forward this named data. If any EAs are found, the named data is forwarded to these EAs accordingly. If no EAs are found which has previously subscribed to this named data, the tasking thread completes dropping the received DD packet.

6. Implementation

6.1. System Implementation. The system is implemented on real devices including WSN, gateway role, and external

agent. Figure 10 shows a setup of the system implemented. Section 6.2 describes how applications are designed and implemented.

PC104 testbed, shown in Figure 8, is used as WSN nodes. Each PC104 has a 533 MHz VIA Mark processor, 256 MB RAM, and is equipped with a 1 GB flash card for external storage.

An Orinoco Gold wireless PCMCIA card is used on each PC104 for wireless communication. It works on 2.4 GHz and supports four speeds: 11 Mb/s, 5.5 Mb/s, 2 Mb/s, and 1 MB/s. An omnidirectional external antenna, as shown in Figure 9, is used to boost the wireless signal with +5 dBI gain.

A USB microphone is used on each PC104 for data production. It is a mono, high sensitivity, omnidirectional microphone with headphone amplifier and detects frequencies from 20 Hz to 20 KHz.

The PC104 sensor nodes use a Linux-based operating system Slax v6.0.7 to run all sensor node application software. Slax is chosen for its small size. Graphic user interface is removed to further save space on the compact flash card. All softwares used in the testbed, including the operating system, are around 75 MB.

A complete list of equipments used in the system is shown in Table 1.

6.2. Application Implementation. Previous theoretical research has developed algorithms for determining the location (at a particular point in time), speed, and direction of movement of a target which emits acoustic sound waves traveling through an array of acoustic sensors. The research was first mathematically formalized for flying airplanes in the more general three-dimensional scenario by Dommermuth in [35] and modified to the two-dimensional scenario by Yang et al. in [36–38] for use in tracking ground-based targets through an acoustic WSN.

A set of applications is implemented on top of the system, to determine the location, speed, and direction of movement of a target, meanwhile capturing video of the moving target with a pan-tilt-zoom camera. The target emits acoustic sound waves, and the sensor nodes monitor the sound waves to calculate location of the target. This is done through collaboration between a *simple producer* and a *complex producer*. An EA application can then utilize the data provided by WSN to drive the camera to track the moving target and capture video. Figure 10 shows the setup of the entire network. The rest of this section gives a detailed description of implemented applications.

CPA_INFO Simple Producer. The *CPA_INFO Simple Producer* senses the sound wave emitted by the target and produces closest-point-of-approach (CPA) information. Figure 11 shows a graph of the sound intensity as a target approaches the sensor node, moves through the CPA relative to the sensor node, and begins moving away from the node. The vertical line represents the time at which the target is at its CPA, and the timestamp of CPA is produced by the sensor application and forwarded to the cluster head, which runs *TARG_INFO complex producer* (see Section 6.2).

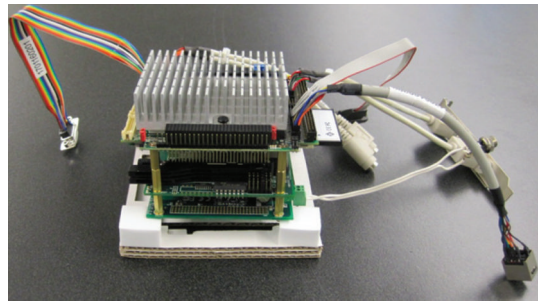


Figure 8: PC104 sensor node without casting.

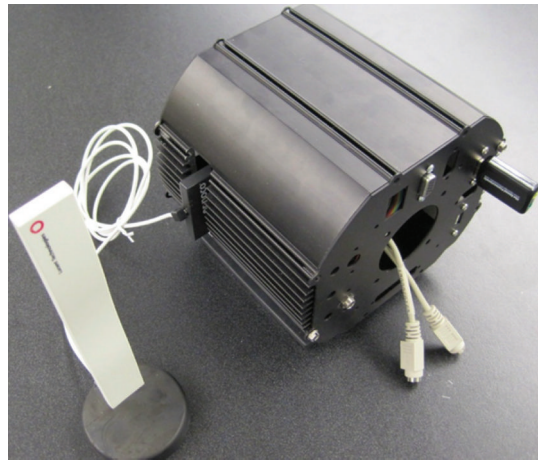


Figure 9: PC104 sensor node with casing and antenna.

Table 1: Equipments used in the system.

Device	Name/model
CPU module	PC104 with Aaeon PFM-550S 553 MHz
Power supply	Aaeon PFM-P13DW2
Wireless card	Orinoco Gold PCMCIA LAN
Antenna	2.4–2.5 GHz omnidirectional +5 dBi with 60″ cable
Storage	Type 1 compact flash 1 GB
RAM	Transcend 144 pin SDRAM 256 MB 133 MHz[a]
Microphone	Mono/omnidirectional SP-USB-MIC-1[b]
Battery	Power Sonic +12 V/5.0 amp hr
Camera	Sony EVI-D30 pan-tilt-zoom camera w/RCA-to-USB

[a] http://www.transcendusa.com/Support/DLCenter/Datasheet/TS32MSS64V6G_6755.pdf.
[b] http://www.soundprofessionals.com/cgi-bin/gold/item/SP-USB-MIC-1.

TARG_INFO Complex Producer. *TARG_INFO* is a *complex producer* that actually calculates the target's location, speed, and direction, using data obtained from other sensor nodes running *CPA_INFO simple producer*. It considers the GPS (Global Positioning System) coordinates of latitude and longitude with CPA data to calculate the target's location and use slope of the path of travel made by the moving target to calculate speed and direction of the target. The application

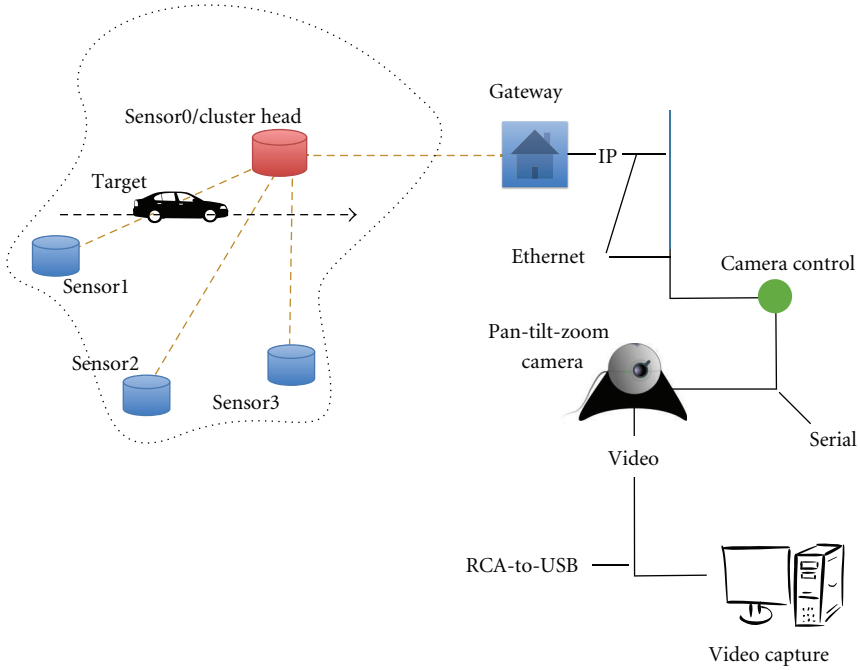


Figure 10: Setup of entire network.

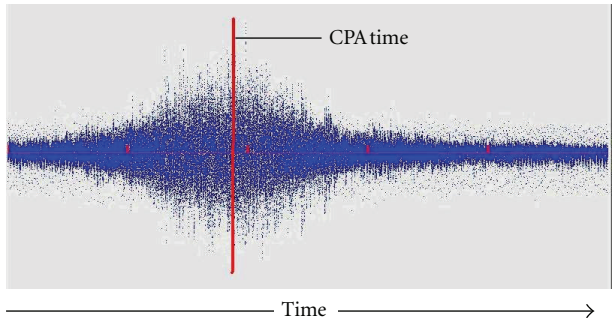


Figure 11: Sound Signature and CPA time of a target moving relative to a single acoustic sensor.

produces a named data TARG_INFO, which is forwarded to the IP network by gateway node.

Target Tracking Application. Target tracking application is implemented on an EA, the camera controller shown in Figure 10. The camera controller and the gateway node are both attached to an improvised IP Ethernet network. An EA application on camera controller sends subscriptions for named data TARG_INFO to the gateway and receives named data produced by *TARG_INFO complex producer* in WSN. It relies on the TARG_INFO data to drive the camera to track a moving target. The video capture by camera is stored in a separate PC in IP network ("video capture" in Figure 10).

Logging and Monitoring Application. Running on an IP network node, the logging and monitoring application implemented simply subscribes to all of the data types the WSN can produce. Upon receiving named data, the application checks the named data type and simply writes its contents to an appropriate log file for long-term storage or further offline analysis at a later time.

7. Experimental Results

7.1. Application Results. Several experiments are conducted, where a moving target travels through the acoustic WSN used for target tracking. Table 2 shows the results of eight experiment runs with the error for target position, speed, and angle of the target. The target position is the position of the target at the closest point of approach with respect to the cluster head. The angle of the target is the angle of the target trajectory from the latitude line. The minimum, maximum and average error for each of target position, speed, and angle is aggregated at the bottom of the table. As shown in the table, the average error of positioning is 1.78 meters, which is near the differential GPS's accuracy 1 meter [39].

Table 2: Target tracking experimental results.

Run	Target position (meters)		Speed (m/s)		Angle (degree)	
	Computed	Error	Computed	Error	Computed	Error
1st	(1099888.48, 195486.53)	1.01	12.52	0.89	90	3
2nd	(1099888.48, 195486.53)	1.01	12.07	1.34	90	3
3rd	(1099888.78, 195485.61)	2.23	18.78	5.36	84	3
4th	(1099886.65, 195486.22)	6.00	12.07	1.34	111	24
5th	(1099884.52, 195485.92)	1.01	12.96	0.45	127	1
6th	(1099884.21, 195486.22)	1.01	12.52	0.89	130	2
7th	(1099884.21, 195486.22)	1.01	12.96	0.45	131	3
8th	(1099884.52, 195486.53)	1.01	14.31	0.89	132	4
Min		1.01		0.45		1
Max		6.00		5.36		24
Average		1.78		1.45		5.375

7.2. LOC Metric. LOC, number of lines of code, is used as a metric to measure "easiness" of building sensor node applications. Table 3 shows the comparison of LOC of two different sensor node applications between with DS and without DS. There are two different sensor node applications: (1) CPA_INFO application which executes on each sensor node to produce the named data CPA_INFO and (2) TARG_INFO application which executes on the cluster head to produce its named data TARG_INFO using CPA_INFO received from each sensor node.

As shown in the table, DS helps to reduce the lines of code required for programming both simple producer application and complex producer application.

Table 3: Comparison of LOC required to write two different applications.

Sensor node application name	LOC without DS	LOC with DS
CPA_INFO application	>50	<10
TARG_INFO application	>65	<15

7.3. Dynamic Service Performance

7.3.1. Performance Metrics. To evaluate the performance of DS, it is necessary to evaluate the impact of DS on sensor node applications. A relatively simple timing analysis is employed to evaluate the impact of DS on sensor node applications' ability to be tasked and to publish data.

First, a consideration is made for the *tasking time* of pure DD and DS sensor node applications. For pure DD sensor applications, the *tasking time* is defined as the time between when the tasking thread is first entered and when the main thread realizes it has been tasked. For DS sensor applications, *tasking time* is defined as the time between when DSs tasking thread is first entered and when the application realizes it has been tasked. For DS, this will give an idea of the length of time it takes for this information to travel through DS, through the message queue, and into the sensor application.

Publishing time is defined as the time between when the sensor node application is tasked and when the data is handed over to DD for network transmission. For pure DD sensor applications, the *publishing time* is the time between when the application is tasked and when the application finished handing over the data to DD. For DS sensor applications, the *publishing time* is the time between when the application is tasked and when DS hands over the data to DD. For DS, this will give an idea of the length of time it takes for this information to travel through DSs message queue and through DS to the DD network.

Figures 12 and 13 illustrate *tasking time* and *publishing time* for sensor node applications with or without DS.

7.3.2. Results. *Tasking time* and *publishing time* are recorded in the experiments for different number of running sensor applications. Figures 14 and 15 are the plots of results.

In the case of tasking, DS has the ability to task the application in less than one millisecond, whereas the busy waiting, or polling, of pure DD applications increases the time to task the application dramatically as the number of running sensor applications increases.

For the publishing process, when with only a few applications running, DS publishes just as quickly as pure DD applications. As the number of running applications increases, however, it becomes clear that DS has an advantage. Since DS sensor applications are sleeping while awaiting tasking at their own message queues rather than busily waiting like pure DD sensor applications; the publishing time remains consistently better than the publishing time for pure DD sensor applications because it is not necessary for the tasked sensor node application to compete for CPU cycles.

8. Conclusions

This paper presents an approach for seamless interconnection between IP networks and WSNs, whereby IP-based hosts can access and manipulate IP-enabled WSNs. Our approach uses common dynamic services for transparent communication with IP-enabled WSNs that allows IP-based hosts to easily task and harvest data from remote dynamic services

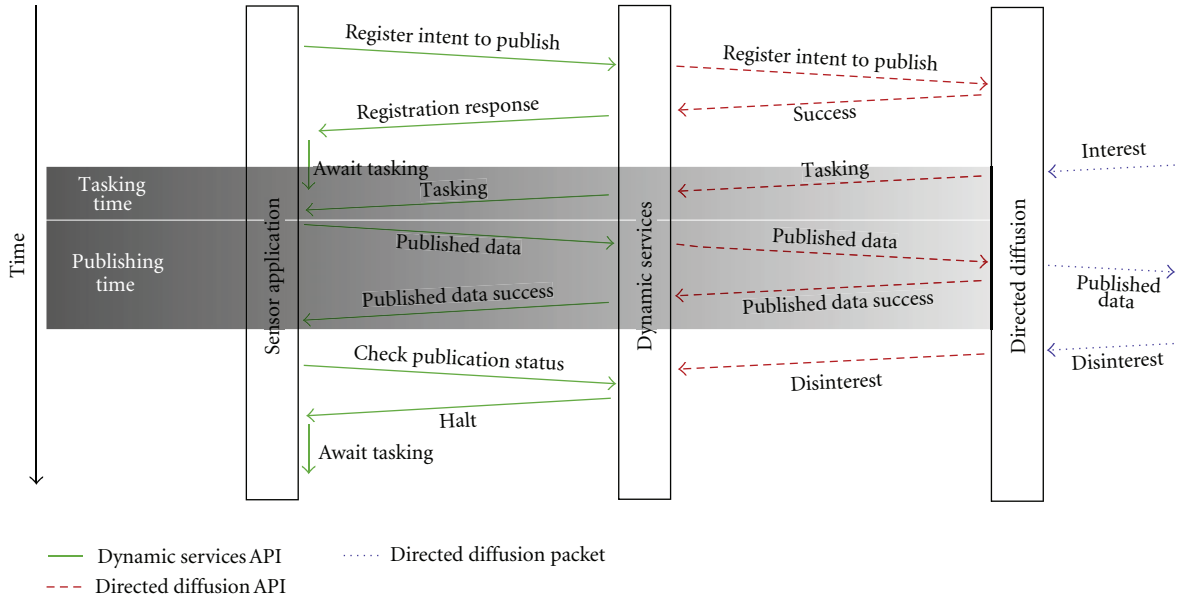


FIGURE 12: *Tasking time* and *publishing time* for sensor node applications with dynamic service.

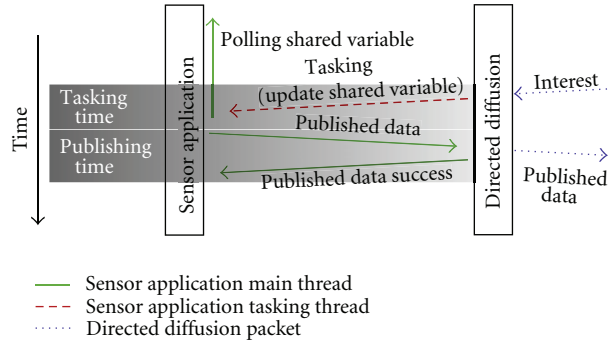


FIGURE 13: *Tasking time* and *publishing time* for sensor node applications without dynamic service.

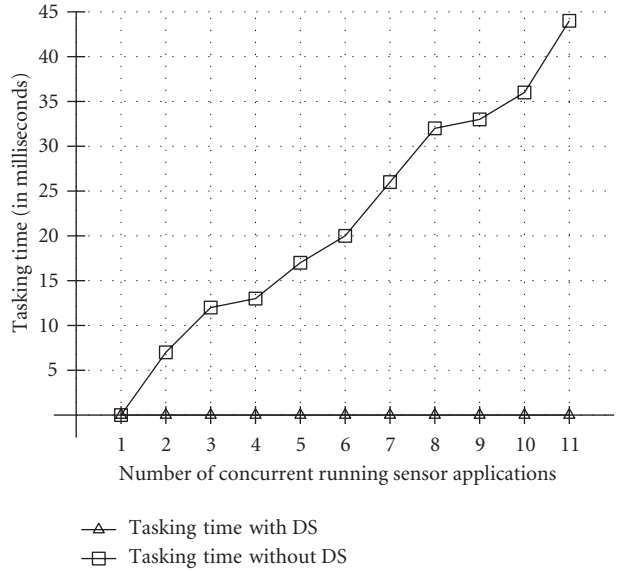


FIGURE 14: *Tasking time* for sensor applications both with and without Dynamic Services as the number of running sensor applications varies.

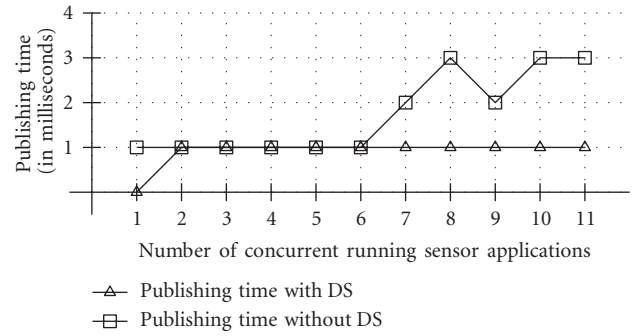


Figure 15: *Publishing time* for sensor applications both with and without dynamic services as the number of running sensor applications varies.

enabled directed diffusion wireless sensor networks. APIs for both IP-based and sensor node application programmers are presented. A description of an application-layer gateway has been given which is used to enable IP-based hosts to gather data from one or more remote WSNs.

A LOC count metric is used for comparing the pure DD API versus the DS API presented in this paper. In the two applications implemented using both the pure DD API and the DS API, it has been shown that the DS API significantly reduces the amount of programming work which must be done in implementing a sensor node application versus using the pure DD API. A performance analysis which clearly shows the value of using DS rather than relying on the polling of pure DD sensor applications is also shown. In short, the performance impact of DS, since it mostly relies on system V message queues for tasking and publishing, is negligible when compared to the performance of pure DD sensor applications for the same operations and actually improves performance due to the fact that pure DD applications busily wait to be tasked.

It is demonstrated that it is possible to, at least partially, bridge the gap between data-centric networks and host-centric networks like IP. Through using the gateway node's mapping service, it is possible to transfer data from a data-centric WSN to interested IP-based hosts.

Future work could follow many different paths.

(i) Currently only EAs submitting interests to WSNs are supported. Some work could be done in allowing the following:

  (1) sensor nodes in a WSN to submit interests for named data types to *other* remote WSNs;

  (2) sensor applications to submit interests for named data to EAs.

(ii) DS API could be extended to include all of the flexibility of the pure DD API.

(iii) DS could be ported to other data-centric networking protocols.

(iv) DD is a best effort service but essentially does not guarantee delivery of data. The DS API could be extended to ensure guaranteed delivery of data to the gateway node or to other sensor nodes.

(v) The current EA API only supports UDP. Extensions could be made to the API that also allow for TCP connections between the EA and the remote gateway node. By combining this and DS assisted guaranteed delivery described in the previous path, delivery of packets from individual sensor node to EAs could be guaranteed.

Acknowledgment

This research is supported in part by the US Army Night Vision Electronic Sensors Directorate (NVESD) under prime Contract no. DAAB07-03-D-C213-005, Subcontract no. SUB1170933RB.

References

[1] D. Culler, D. Estrin, and M. Srivastava, "Overview of sensor networks," *Computer*, vol. 37, no. 8, pp. 41–49, 2004.

[2] A. Bharathidasan, V. Anand, and S. Ponduru, "Sensor networks: an overview," in *Proceedings of the IEEE Infocom*, 2004.

[3] B. Keith Maharrey, *A gateway-based approach for information retrieval from data-centric wireless sensor networks from IP hosts [M.S. thesis]*, Auburn University, December 2010.

[4] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, pp. 88–97, Atlanta, Ga, USA, September 2002.

[5] S. N. Simic and S. Sastry, "Distributed environmental monitoring using random sensor networks," in *In Proceedings of the 2nd International Workshohp on Information Processing in Sensor Networks*, pp. 582–592, Palo Alto, Calif, USA, 2003.

[6] A. Dunkels, T. Voigt, N. Bergman, and M. Jnsson, "An IP-based sensor network as a rapidly deployable building security system," in *Swedish National Computer Networking Workshop*, Karlstad, Sweden, November 2004.

[7] G. Werner-Allen, J. Johnson, M. Ruiz, J. Lees, and M. Welsh, "Monitoring volcanic eruptions with a wireless sensor network," in *Proceedings of the 2nd European Workshop onWireless Sensor Networks (EWSN '05)*, pp. 108–120, February 2005.

[8] H. Dai and R. Han, "Unifying micro sensor networks with the internet via overlay networking," in *Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks (LCN '04)*, pp. 571–572, Tampa, Fla, USA, November 2004.

[9] M. El Barachi, A. Kadiwal, R. Glitho, F. Khendek, and R. Dssouli, "The design and implementation of architectural components for the integration of the IP multimedia subsystem and wireless sensor networks," *IEEE Communications Magazine*, vol. 48, no. 4, pp. 42–50, 2010.

[10] S. Hong, D. Kim, M. Ha et al., "SNAIL: an IP-based wireless sensor network approach to the Internet of things," *IEEE Wireless Communications*, vol. 17, no. 6, pp. 34–42, 2010.

[11] B. Campos, J. Rodrigues, L. Mendes, E. Nakamura, and C. Figueiredo, "Design and construction of wireless sensor network gateway with IPv4/IPv6 support," in *Proceedings of IEEE International Conference on Communications (ICC '11)*, pp. 1–5, June 2011.

[12] A. Dunkels, T. Voigt, J. Alonso, H. Ritter, and J. Schiller, "Connecting wireless sensornets with TCP/IP networks," in *Proceedings of the 2nd International Conference on Wired/Wireless Internet Communications (WWIC '04)*, Frankfurt, Germany, February 2004.

[13] K. A. Emara, M. Abdeen, and M. Hashem, "A gateway-based framework for transparent interconnection between WSN and IP network," in *Proceedings of the IEEE EUROCON (EUROCON '09)*, pp. 1775–1780, May 2009.

[14] S. Ping, C. Chang, L. Kejie, and S. Li, "The design and realization of embedded gateway based on WSN," in *International Conference on Computer Science and Software Engineering (CSSE '08)*, pp. 32–36, December 2008.

[15] P. K. Mohanty, "A framework for interconnecting wireless sensor and IP networks," in *Proceedings of the 18th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC '07)*, September 2007.

[16] A. Leonardi, S. Palazzo, F. Scoto, and S. Signorello, "Design and construction of wireless sensor network gateway with IPv4/IPv6 support," in *Proceedings of the 7th International Wireless Communications and Mobile Computing Conference (IWCMC '11)*, pp. 285–290, July 2011.

[17] R. Ding and H. Du, "Location-based IP addressing in IP-enable wireless sensor networks," in *Proceedings of the International Conference on Control, Automation and Systems Engineering (CASE '11)*, pp. 1–4, July 2011.

[18] A. Lewandowski, V. Köster, and C. Wietfeld, "Performance evaluation of AODV and OLSR-meshed IP-enabled IEEE802.15.4," in *Proceedings of the 3rd International Conference on Advances in Mesh Networks (MESH '10)*, pp. 7–12, July 2010.

[19] A. K. M. Azad, J. Kamruzzaman, B. Srinivasan, K. H. M. Alam, and S. Pervin, "Query processing over distributed heterogeneous sensor networks in future internet: scalable architecture and challenges," in *Proceedings of the 2nd International Conference on Advances in Future Internet (AFIN '10)*, pp. 75–81, July 2010.

[20] K. F. Navarro, E. Lawrence, and B. Lim, "Medical motecare: a distributed personal healthcare monitoring system," in *Proceedings of the International Conference on eHealth, Telemedicine, and Social Medicine (eTELEMED '09)*, pp. 25–30, February 2009.

[21] G. J. Pottie and W. J. Kaiser, "Embedding the Internet: wireless integrated network sensors," *Communications of the ACM*, vol. 43, no. 5, pp. 51–58, 2000.

[22] J. J. P. C. Rodrigues and P. A. C. S. Neves, "A survey on IP-based wireless sensor network solutions," *International Journal of Communication Systems*, vol. 23, no. 8, pp. 963–981, 2010.

[23] G. Wagenknecht, M. Anwander, and T. Braun, "SNOMC: an overlay multicast protocol for wireless sensor networks," in *Proceedings of the 9th Annual Conference on Wireless On-demand Network Systems and Services (WONS '12)*, pp. 75–78, January 2012.

[24] M. V. Pulgarin, R. Glitho, and A. Quintero, "An overlay gateway for the integration of IP multimedia subsystem and mobile sink based—wireless sensor networks," in *Proceedings of the 72nd IEEE Vehicular Technology Conference Fall (VTC '10-Fall)*, pp. 1–5, September 2010.

[25] N. Pollner, M. Daum, F. Dresslery, and K. Meyer-Wegener, "An overlay network for integration of WSNs in federated stream- processing environments," in *Proceedings of the10th IFIP Annual Mediterranean Ad Hoc Networking Workshop*, June 2011.

[26] S. Coleri Ergen, "ZigBEE/IEEE802.15.4 Summary," September 10, 2004.

[27] G. Mulligan, "The 6LoWPAN architecture," in *Proceedings of the 4th Workshop on Embedded Networked Sensors (EmNets '07)*, pp. 78–82, June 2007.

[28] G. Moritz, F. Golatowski, and D. Timmermann, "A lightweight SOAP over CoAP transport binding for resource constraint networks," in *Proceedings of the IEEE 8th International Conference on Mobile Adhoc and Sensor Systems (MASS '11)*, pp. 861–866, October 2011.

[29] S. Raza, S. Duquennoy, T. Chung, D. Yazar, T. Voigt, and U. Roedig, "Securing communication in 6LoWPAN with compressed IPsec," in *Proceedings of the International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS '11)*, pp. 1–8, June 2011.

[30] A. A. Hasbollah, S. H. S. Ariffin, and M. I. A. Hamini, "Performance analysis for 6loWPAN IEEE 802.15.4 with IPv6 network," in *Proceedings of the IEEE Region 10 Conference (TENCON '09)*, pp. 1–5, January 2009.

[31] M. Durvy, J. Abeillé, P. Wetterwald et al., "Making sensor networks IPv6 ready," in *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, pp. 421–422, 2008.

[32] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: a scalable and robust communication paradigm for sensor networks," in *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MOBICOM '00)*, pp. 56–67, Boston, Mass, USA, August 2000.

[33] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: a scalable and robust communication paradigm for sensor networks," in *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MOBICOM '00)*, pp. 56–67, Boston, Mass, USA, August 2000.

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

[35] F. M. Dommermuth, "The estimation of target motion parameters from CPA time measurements in a field of acoustic sensors," *The Journal of the Acoustical Society of America*, vol. 83, no. 4, pp. 1476–1480, 1988.

[36] Q. Yang, A. Lim, K. Casey, and R. K. Neelisetti, "An empirical study on real-time target tracking with enhanced CPAalgorithm in wireless sensor networks," *Ad-Hoc and Sensor Wireless Networks*, vol. 7, no. 3-4, pp. 225–249, 2009.

[37] Q. Yang, A. Lim, K. Casey, and R. K. Neelisetti, "An enhanced CPA algorithm for real-time target tracking in wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 5, no. 5, pp. 619–643, 2009.

[38] Q. Yang, A. Lim, K. Casey, and R. K. Neelisetti, "Real-time target tracking with CPA algorithm in wireless sensor networks," in *Proceedings of the 5th Annual IEEE Communications Society*

Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON '08), pp. 305–313, June 2008.

[39] R. Bajaj, S. L. Ranaweera, and D. P. Agrawal, "GPS: location tracking technology," *Computer*, vol. 35, no. 4, pp. 92–94, 2002.