COSMOS: A Middleware for Integrated Data Processing over Heterogeneous Sensor Networks

Marie Kim, Jun Wook Lee, Yong Joon Lee, and Jae-Cheol Ryou

With the increasing need for intelligent environment monitoring applications and the decreasing cost of manufacturing sensor devices, it is likely that a wide variety of sensor networks will be deployed in the near future. In this environment, the way to access heterogeneous sensor networks and the way to integrate various sensor data are very important. This paper proposes the common system for middleware of sensor networks (COSMOS), which provides integrated data processing over multiple heterogeneous sensor networks based on sensor network abstraction called the sensor network common interface. Specifically, this paper introduces the sensor network common interface which defines a standardized communication protocol and message formats used between the COSMOS and sensor networks.

Keywords: Sensor network abstraction, RFID, standardized interface, sensor network simulator, sensor network common interface.

Manuscript received Jan. 30, 2008; revised Aug. 4, 2008; accepted Aug. 22, 2008.

This work was supported by the IT R&D program of MKE/IITA [2006-S-022-01, Development of USN Middleware Platform Technology] and by a grant (#07KLSGC02) from Cutting-Edge Urban Development–Korean Land Spatialization Research Project funded by Ministry of Construction & Transportation of Korean government.

The fourth author was supported by the Ministry of Knowledge Economy, Rep. of Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute of Information Technology Advancement) (IITA-2008-C1090-0801-0016).

Marie Kim (phone: + 82 42 860 1590, email: mariekim@etri.re.kr), Jun Wook Lee (email: junux@etri.re.kr), Yong Joon Lee (email: yjl@etri.re.kr) are with the IT Convergence Technology Research Laboratory, ETRI, Daejeon, Rep. of Korea.

Jae-Cheol Ryou (email: jcryou@home.cnu.ac.kr) is with the Department of Computer Science, Chungnam National University, Daejeon, Rep. of Korea.

I. Introduction

Many studies on sensor networks have focused on routing within a sensor network [1], [2], energy conservation [2], in-network query processing [3], and so on. Usually, these kinds of studies do not pay attention to the heterogeneity among sensor networks. They focus on specific issues, such as routing, query optimization, and energy conservation under the assumption that all kinds of sensor networks are homogeneous. In this context, those studies are theoretical rather than practical.

Recently, with the increasing prevalence of web services, studies about the sensor web [4] or global sensor network (GSN) [5] have been actively progressing. The goal of these studies is the development of a web which is composed of accessible sensors. In the Open Geospatial Consortium [4], the relevant languages and interfaces are standardized and verified by implementation.

The common system for middleware of sensor networks (COSMOS) [6] is a common application platform over multiple heterogeneous sensor networks. The COSMOS is designed to support real field applications that interact with the environment without human intervention. The COSMOS provides sensor network abstraction (sensor network common interface), query optimization, integration of data from various sensors, sensor network monitoring, and intelligent sensor data processing, such as event handling, sensor data mining [7], and context information processing. The COSMOS handles both sensor networks and RFID readers. It is a ubiquitous sensor network (USN) middleware, not just a sensor network middleware. A USN is composed of devices which interact with an environment by obtaining environmental values (*sense*) and/or taking actions toward the environment (*actuate*). It includes wireless sensor networks, RFID readers, actuator

networks, wired sensor networks, laptop computers, webcams, speakers, and so on.

The rest of this paper is organized as follows. Section II compares this study with other relevant studies. The COSMOS is introduced in section III. Section IV describes the sensor network common interface. From a sensor network abstraction view point, the implementation and the performance of COSMOS are reported in section V.

II. Comparisons with Related Studies

Research regarding sensor networks can be classified according to the general methodological approach into two categories: the "bottom-up approach" and the "top-down approach". The sensor network services platform (SNSP) [8], Cougar [9], sensor information networking architecture and applications (SINA) [10], and COSMOS fall under the category of the top-down approach. The top-down approach analyzes functional requirements from an application point of view. This approach focuses on optimal solutions to process distributed sensor data and to provide an access interface to the sensor network from the application standpoint. However, this is different from the global computing paradigm [11] which aims at a high throughput using distributed computing resources. Cougar and SINA attempt to find optimal sensor data processing methods over sensor networks. SNSP and COSMOS define an implementation level application interface to access multiple heterogeneous sensor networks and organize the sensor network framework systemically according to services, such as query processing, metadata storage, application programming interface service, and so on. In addition, SNSP and COSMOS classify several possible queries for sensor data into instant queries, continuous queries, and event queries. Of these studies, only COSMOS considers the heterogeneity of sensor networks.

SensorML [12], GSN, and IEEE1451 [13] fall under the category of the bottom-up approach. They model sensors and sensor networks from the device point of view. IEEE1451 defines a smart transducer interface to access the smart transducer and uses a transducer electronic data sheet to manipulate the metadata of transducers. The SensorML models the processes related to sensors. SensorML models and handles sensors consistently. GSN defines *virtual sensors*. A virtual sensor can be a simple local sensor, a remote sensor, or a sensor network. A virtual sensor can make a hierarchy to operate systemically. Basically, all these bottom-up approaches are based on the self-description method. To join the services, sensors, actuators, and sensor networks have to publish their abilities and metadata before performing their functions. This self-description is used for the resource discovery process and the sensor data query process. This approach is very flexible and scalable. However, the main drawback of these bottom-up approaches is that they are limited in their ability to support sensor network management and sensor data processing over multiple heterogeneous sensor networks. This is because this approach only focuses on device modeling and device access methods rather than overall data processing and resource management over multiple sensor networks.

The main functions of the COSMOS can be summarized as follows:

- It defines the service architecture, including metadata repository, query processor, sensor network monitor, security, application interface, sensor network interface, and so on.
- It defines the sensor network hierarchy from a transducer to a sensor network (transducer → node → network).
- It defines a consistent application interface (Open API) for various applications.
- It defines a sensor network common interface to access heterogeneous sensor networks in a consistent way.
- It provides integrated sensor data processing over multiple heterogeneous sensor networks.

The COSMOS defines USN service architecture, develops necessary functional blocks, and defines sensor network hierarchy. It differs from the approach of a virtual sensor in SNSP and GSN. In SNSP and GSN, there is no service architecture or sensor network hierarchy. The virtual sensor can be a transducer, a sensor node, or a sensor network, and this concept is a very flexible and scalable instrument. However, the COSMOS defines the concept of the sensor network. A transducer is either a sensor or an actuator, and a sensor node may include one or more transducers, while multiple sensor nodes comprise a sensor network. This approach is rather inflexible but more instinctive and more effective to implement than virtual sensors.

The COSMOS separates sensor networks from applications. Heterogeneous sensor networks and RFID readers communicate with the COSMOS via the sensor network common interface. Non-standardized sensor networks can be connected to the COSMOS easily via appropriate adaptors. Adaptors are logical nodes which transform and transfer messages between the sensor networks and the COSMOS. They can be implemented on a sink node, a gateway, or in COSMOS. The sensor network common interface is defined from the user's point of view. It does not deal with device specifics.

In addition, the COSMOS monitors sensor networks and keeps their current status regarding topology, battery level of nodes, and so on. Integrated sensor data processing is based on the sensor network metadata and the current status of the sensor networks.

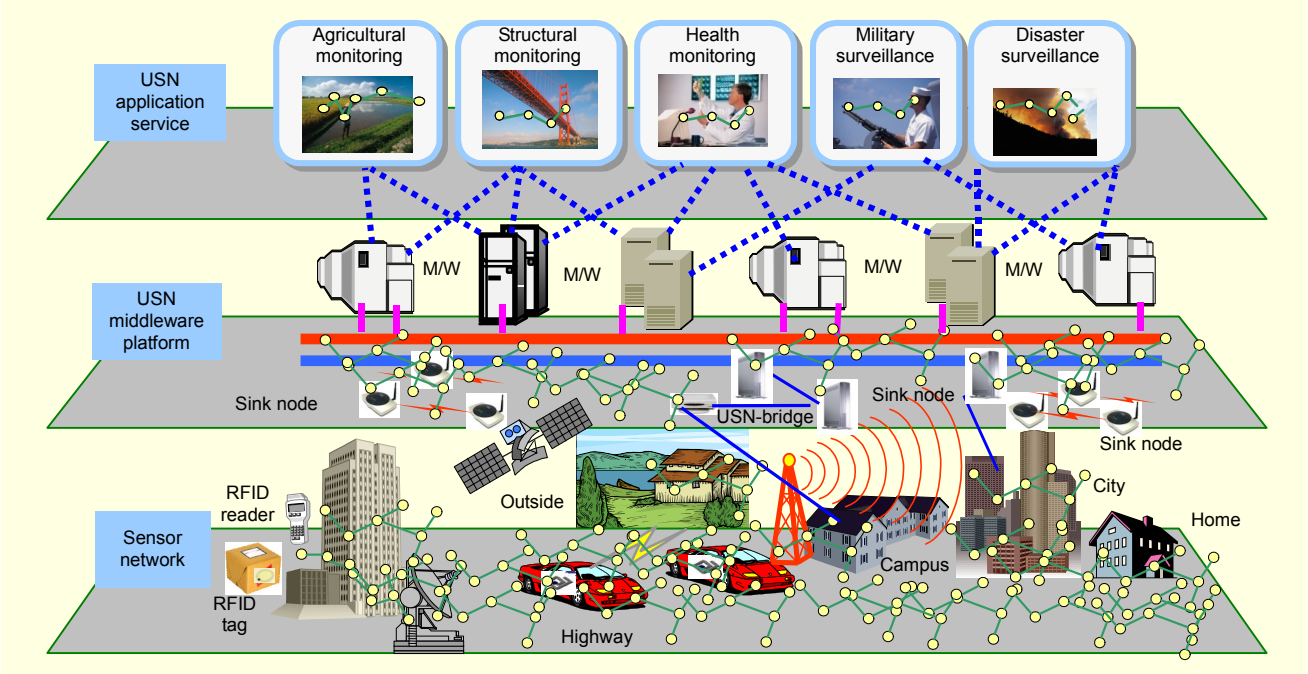

Fig. 1. USN service architecture.

III. USN Middleware (COSMOS)

1. USN Service Architecture

As mentioned in the introduction, USN is not a traditional network; rather, it is a business scale network. Figure 1 gives an illustration of USN service architecture, which is a redrawing based on [14]. USN applications use sensor networks via USN middleware. Figure 1 shows various types of sensor networks based on various technologies, such as IEEE 802.11 wireless sensor networks, ZigBee wireless sensor networks, Bluetooth wireless sensor networks, RFID readers, mobile RFID, and so on. The sensor networks provide sensor data or RFID tag data to the various USN applications. Multiple USN applications use sensor networks for their own purposes. USN applications can use specific sensor networks exclusively, or they can share among other applications if necessary.

In this service architecture, it is very effective to place a common application platform between the heterogeneous sensor networks and various applications. The USN middleware provides common functions required for various applications. These common functions are derived from application viewpoints, and they include sensor network access, sensor network monitoring, metadata storage, and integrated query processing functions.

2. USN Middleware (COSMOS) Architecture

Figure 2 describes the COSMOS architecture. It consists of three tiers: the sensor network abstraction tier, the sensor information processing tier, and the USN service tier.

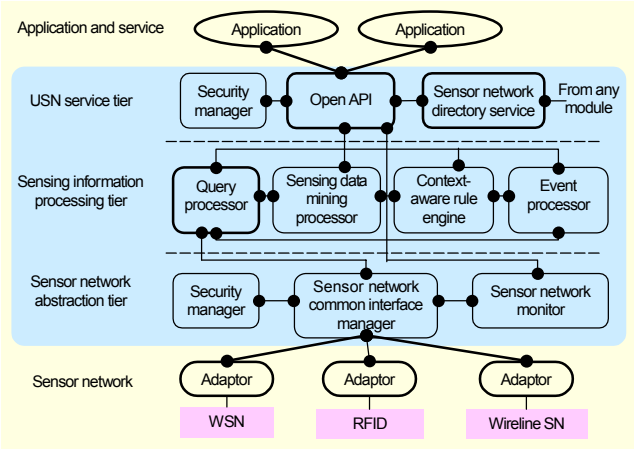

Fig. 2. COSMOS architecture.

A. Sensor Network Abstraction Tier

The sensor network abstraction tier includes the sensor network common interface manager, the sensor network monitor, and the security manager functionalities. The sensor network common interface manager plays an intermediate role between the COSMOS and various sensor networks (including RFID readers). To communicate with the sensor network common interface manager, the sensor network has to implement an appropriate adaptor. The security manager controls access to COSMOS against the sensor networks. The sensor network monitor

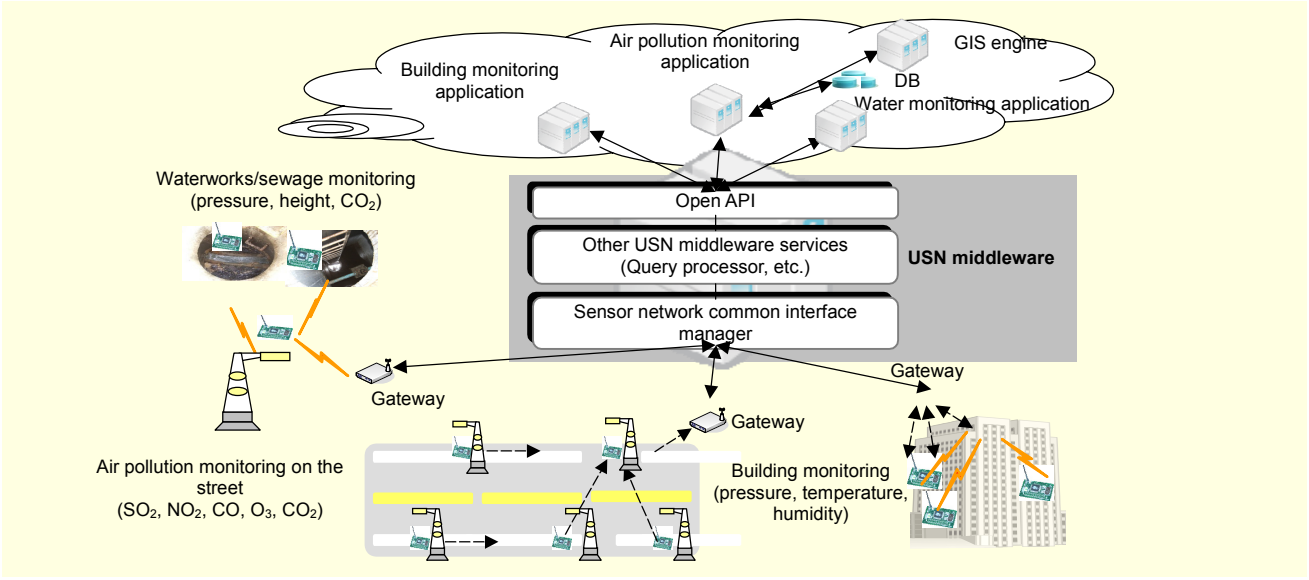

Fig. 3. Example services that use USN middleware.

provides real-time sensor network monitoring functions.

B. Sensing Information Processing Tier

The sensing information processing tier includes the query processor, the sensing data mining processor, the context-aware rule engine, and the event processor. Among those functions, only the query processor is mandatory. The query processor is responsible for making optimal query plans, executing the queries, and processing the received sensor data. The sensor data mining processor detects outliers, analyzes patterns, and predicts some events. The event processor creates events within the COSMOS based on the sensor data. The context-aware rule engine processes context-aware rules created by applications. If an application needs a context-aware sensor data processing service, then it specifies appropriate rules based on the related background knowledge. Before requesting sensor data from the COSMOS, the application has to input the rule to the COSMOS using Open API. For example, the healthcare application inputs a rule which describes the abnormal conditions of patients to the context-aware rule engine before requesting sensor data as in the following rule example: if temperature > 37 & momentum < threshold, then turn on the red light located in the office. Then, the healthcare application requests sensor data from the COSMOS. While receiving sensor data related to the patients, the context-aware rule engine processes sensor data based on the rules. If an abnormal condition is detected, then the context-aware rule engine processes the situation.

C. USN Service Tier

The USN service tier includes the Open API, the USN directory service, and the security manager functionalities. The Open API provides a communication interface (login, query, report, logout, and so on) to the applications and is implemented using web service. This is the entrance point to the COSMOS for the applications. The USN directory service provides the metadata of sensor networks. The applications and other functions provided by the COSMOS refer sensor network metadata to USN directory service. The security manager provides authentication, authorization and confidentiality functions to protect the COSMOS.

Figure 3 describes possible service examples which use the USN middleware to acquire sensor data from sensor networks. There are three applications: building monitoring, air pollution monitoring, and water monitoring. There are also three sensor networks: waterwork/sewage monitoring, air pollution monitoring on the street, and building monitoring. All of the sensor networks and applications are connected via the USN middleware. If all applications operate with each sensor network directly, each application developer has to know the details of each sensor network. By using the USN middleware, an application developer only needs to implement the Open API. A sensor network developer only needs to know the sensor network common interface. The deployed sensor networks can be shared among applications with appropriate authorization policy.

IV. Sensor Network Common Interface

In Korea since 2006, there have been many pilot projects to verify the feasibility of sensor related technology for real life, which have tested sensor networks in contexts such as ocean

monitoring, weather monitoring, mountain monitoring, and so on. These pilot projects have achieved meaningful results. Sensor network technologies have been proven to be viable in real-life automatic monitoring systems. However, the interoperability of sensor networks has become a serious issue. Sensor networks implemented for certain applications cannot be used by other applications because sensor networks are implemented according to specific application requirements. Each application requires its own sampling rate, sensing type, connection method, and communication interface between application and sensor networks, sensor node software platform, and so on.

To ensure independence and to provide a communication interface at the same time, there is a need for standardized interfaces between sensor networks and applications. From the COSMOS perspective, Open API and the sensor network common interface provide sensor network abstraction at different levels. If COSMOS is used, application developers use Open API to access sensor networks, and then the COSMOS uses the sensor network common interface to communicate with sensor networks. If an application is to access sensor networks directly, then the application developer has to implement the sensor network common interface. In Korea, the sensor network common interface was accepted as a domestic standard in 2007 by the Telecommunication and Technology Association (TTA) [15]. At the same time, we have been trying to make this interface an international standard with ITU-T since 2007 [16]-[18].

In this section, the sensor network common interface is described. This interface defines the communication protocol and messages at the implementation level. The messages are defined and added by the application requirements; therefore, the interface is inclusive. The sensor network developer does not need to implement all the messages exhaustively. In this paper, a host and COSMOS are used interchangeably. Basically, a host is an entity which uses sensor network(s).

Figure 4 shows the communication model of the COSMOS

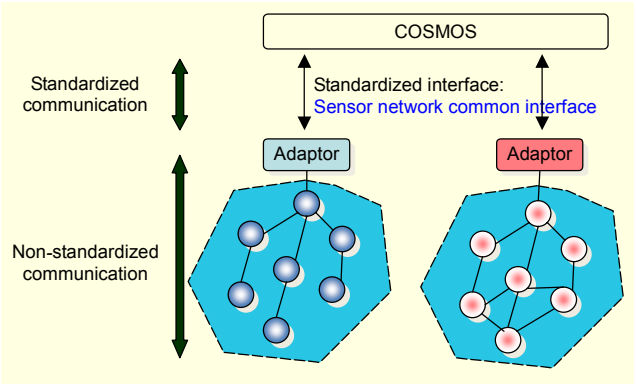

Fig. 4. COSMOS communication model.

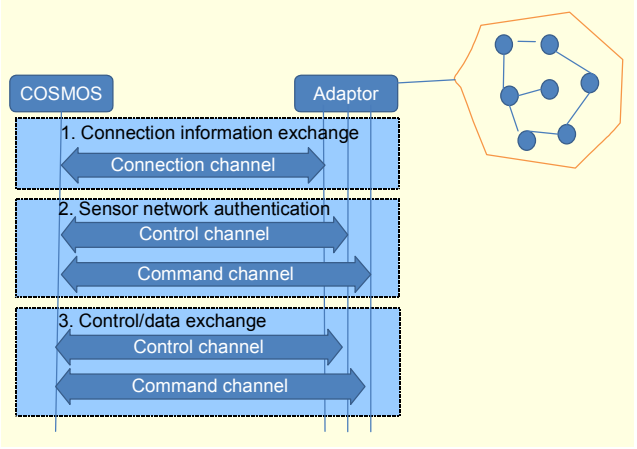

Fig. 5. Communication protocol.

from the sensor network developer's point of view. The coverage of the sensor network common interface is between an adaptor and the COSMOS. From the viewpoint of the COSMOS, the protocol used within a sensor network is beyond its scope.

In COSMOS, two kinds of message transport binding (MTB) are defined: binary message over TCP and XML over HTTP. Depending on the adaptor implementation, the sensor network developers choose one of them to communicate with the COSMOS. Using XML is a good way to exchange information because it is quite extensible and used widely. However, for devices with limited resources, such as low computing resource, limited battery power, and short storage, XML messages may not be appropriate because the XML parser has to be loaded on the resource restrained device to process the XML messages. In such a case, the binary message over TCP MTB may be a more appropriate choice. Whichever MTB is used, the communication protocol and message contents are the same.

1. Communication Protocol

The sensor network common interface defines three channels and a three step communication protocol. The three channels include the connection channel, the command channel, and the control channel. The three steps are the following: connection information exchange, sensor network authentication, and control/data exchange. Figure 5 shows the three channels and three steps.

The connection channel delivers connection information to the sensor network. Using connection information delivered to the sensor network, the adaptor tries to open control and command channels.

• Connection channel: The connection information of command channel and control channel are delivered to the sensor network via the connection channel. The connection

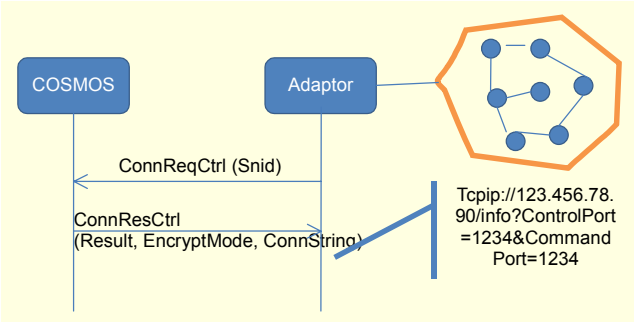

Fig. 6. Connection channel information flow.

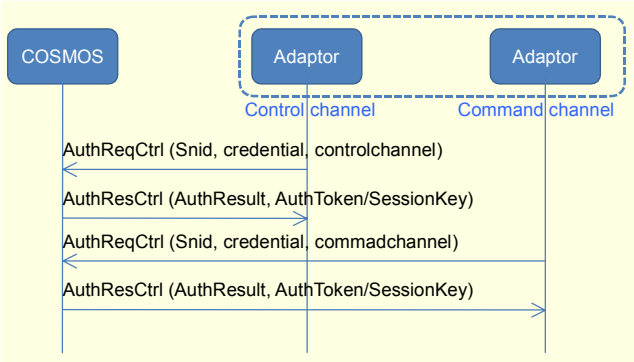

Fig. 7. Establishment of command and control channels.

channel information is fixed ahead of runtime.

• Command channel: The commands (sensor data request, monitoring data request) and reports are delivered using the command channel. The COSMOS can handle other jobs while waiting for the reports.

• Control channel: The controls (sensor node on/off, sensor network metadata request, and so on) and responses are delivered via the control channel. The COSMOS has to wait for the response without performing other tasks.

Figure 6 shows an example of the connection channel information flow. In this example, binary messages are used and the sensor network starts to connect. The adaptor requests the COSMOS to connect using the *ConnReqCtrl* message. Then the COSMOS checks the Snid contained in *ConnReqCtrl*. If the Snid is valid, then the COSMOS provides connection information (*ConnString*) to the adaptor using a *ConnResCtrl* message. In this case, the *ConnString* contains socket addresses which are used for the command and control channels. The control channel uses IP address = 123.456.78.90 and port# = 1234. The command channel uses IP address = 123.456.78.90 and port# =4567. The *EncryptMode* indicates how to encrypt the messages exchanged.

Figure 7 shows the process of establishing command and control channels. In this stage, sensor network authentication is performed using *AuthReqCtrl/AuthResCtrl* messages. The authentication related information (mode, key, algorithm, and so on) have been previously registered at the USN directory service. Two modes are supported in the COSMOS:

Table 1. Messages descriptions (C: COSMOS, S: sensor network).

Category	Message	Description
Connect Disconnect	ReqConnCtrl	Connect request (C→S)
	ConnReqCtrl	Connect request (C←S)
	ConnResCtrl	Connect response (C→S)
	DisConnReqCtrl	Disconnect ctrl (C↔S)
	AuthReqCtrl	Authentication request
	AuthResCtrl	Authentication response
Request	NetworkInfoReq	Network metadata request (C→S)
	BufferDataReq	Buffered data request (C→S)
	CmdActionReq	Command action request (C→S)
	UpdateCmdReq	Command update request (C→S)
	ControlNetworkReq	Network control (C→S)
	ControlNodeReq	Node control (C→S)
Response	NetworkInfoRes	Sensor network metadata response (C↔S)
	BufferDataRes	Buffered data response (C↔S)
	CmdActionRes	Command action response (C↔S)
	UpdateCmdRes	Command update response (C↔S)
	ControlNetworkRes	Network control response (C↔S)
	ControlNodeRes	Node control response (C↔S)
Command	InstantCmd	Instant command (C→S)
	ContinuousCmd	Continuous command (C→S)
	InstantEventCmd	Event command (C→S)
	InstantAggCmd	Instant aggregation command (C→S)
	ContinuousAggCmd	Continuous aggregation command (C→S)
	RunActuatorCmd	Run actuator command (C→S)
	MonitoringStartCmd	Monitoring start command (C→S)
	MonitoringStopCmd	Monitoring stop command (C→S)
Report	SensingValueRpt	Sensor value report (C↔S)
	RunActuatorRpt	Run actuator report (C↔S)
	FinishRpt	Finish report
	MonitoringRpt	Monitoring report
	ErrorRpt	Error report
	UpdateRpt	Update report
Check	ChannelCheckCtrl	Channel check control
	ChannelConfirmCtrl	Channel confirm control
	NakChk	Nagative acknowledgment check

ID/password-based authentication and certificate-based authentication. After authentication, the command/control can be delivered to the sensor networks, and the response/report can be delivered to the COSMOS.

2. Message Categories

The sensor network common interface can be extended if there are other application requests which cannot be covered by the existing messages. Messages can be categorized as follows.

- Connect/disconnect: channel establishment
- Request/response: handles sensor network metadata, sensor network control, and command action control
- Command/report: handles sensor data, monitoring data, and actuator manipulation
- Check: channel check and error check

Table 1 describes the messages. Among these, the messages related with the connection, authentication, and the sensor network metadata exchange are mandatory.

3. Functions Provided

By exchanging the messages defined in Table 1, the following functions are provided.

- Connection management: connects/disconnects the channel between the COSMOS and the sensor network.
- Authentication: authenticates the sensor networks.
- Sensor network metadata exchange: as soon as connected, the sensor networks deliver metadata to COSMOS to inform the current status of sensor network.
- Network/node control: frequency control, topology control, and node control requests/responses are exchanged.
- Buffered data exchange: the buffered data stored within an adaptor (if any) is delivered to the COSMOS.
- Command processing: the sensor data related commands and reports are exchanged.
- Monitoring processing: the monitoring related commands and reports are exchanged.
- Actuator processing: the actuator control.
- Error report: reports sensor network error to the COSMOS.
- Channel check: checks channel status.

4. Sensor Data Processing

The COSMOS embraces sensor networks and RFID reader networks. In the COSMOS, RFID tag data is processed differently than other sensor data. Sensor data represents the current spatiotemporal situation. Therefore, no sensor data is dropped. However, RFID tag data is often redundant. The COSMOS provides filtering functions to reduce redundant RFID tag data before sending them to applications. The sensor network common interface manager creates events to filter redundant RFID tag data. Five events, namely, *TagSeen*, *TagVanished*, *SoftRead*, *FirRead*, and *TagExpired* are defined.

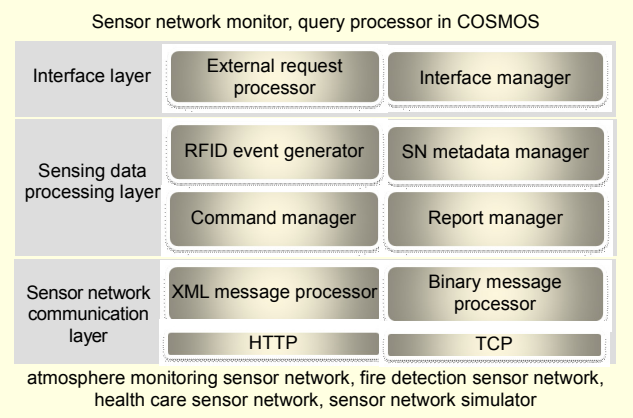

Fig. 8. Sensor network common interface manager structure.

In the COSMOS, the sensor network operation modes are defined as *pull mode* and *push mode*. In pull mode, sensor networks send sensor data in response to explicit commands from the host. To operate in pull mode, sensor nodes have to process commands. The capability of processing commands and the maximum number of commands that can be processed at the same time have to be registered at the USN directory service before runtime. By using this capability data, COSMOS does not send more commands to the sensor network if the number of running commands equals the maximum number of commands that can be processed concurrently. In push mode, sensor networks send sensor data periodically. Then, the COSMOS processes the sensor data according to requests by applications. An RFID reader usually operates in push mode.

Figure 8 shows the sensor network common interface manager structure. The sensor network communication layer communicates with the adaptors, and it supports HTTP and TCP transport protocols. The sensing data processing layer handles command/request and report/response. The interface layer provides a interface to and from the sensor network monitor and query processor.

5. XML Schema Example

Figure 9 shows an example of the XML schema. The differential characteristic of the COSMOS is that it handles a wide spectrum of sensor networks. If a sensor network operates in push mode, COSMOS receives sensor data periodically, and then processes it according to the application requirements. If the sensor network can process commands with condition(s), then COSMOS sends the commands with conditions to the sensor network for effective in-network query processing. For

```
<!-- InstantCmd -->
  <xsd:complexType name="instantCmd">
    <xsd:sequence>
      <xsd:element name="commandID" type="xsd:int"
            minOccurs="1" maxOccurs="1"/>
      <xsd:element name="nodeID" type="xsd:long"
            minOccurs="1" maxOccurs="unbounded"/>
      <xsd:element name="sensingTypeID" type="msg:sensingTypeID"
            minOccurs="1" maxOccurs="unbounded"/>
      <xsd:element name="condition" type="msg:condition"
            nillable="true" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>

  <!-- SensingValueRpt -->
  <xsd:complexType name="sensingValueRpt">
    <xsd:sequence>
      <xsd:element name="commandID" type="xsd:int"
            minOccurs="1" maxOccurs="1"/>
      <xsd:element name="nodeID" type="xsd:long"
            minOccurs="1" maxOccurs="1"/>
      <xsd:element name="sendingTime" type="xsd:int"
            minOccurs="1" maxOccurs="1"/>
      <xsd:element name="sensingValue" type="msg:sensingValue"
            minOccurs="1" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
```

Fig. 9. Example of the XML schema.

Table 2. Sensing type list (partial).

ID	Type	Unit	Length
0x1710	Temperature	°C	4 bytes, float
0x1702	Salinity	%	4 bytes, float
0x1703	Dissolved oxygen	%	4 bytes, float
0x1704	Battery	V	4 bytes, float
0x1705	Battery	%	4 bytes, float
0x1706	Pulse	bpm	4 bytes, int
0x1707	Momentum	mG (1 G=980 cm/s^2)	4 bytes, int
0x1708	Blob	-	N bytes, char
0x1709	Pressure	gf/cm^2	2 bytes, int
0x170A	Height	cm	2 bytes, int
0x170B	CO	ppm	4 bytes, float

example, a command may be like this: "Report {temperature, humidity, CO_2} from sensor network #1 if (temperature> 30°C) and (CO_2>threshold)." It may be expressed as an *InstantCmd* message, and the sensor data collected is delivered to the COSMOS using a *SensingValueRpt*. In push-mode sensor networks, then, the sensor networks periodically transmit *SensingValueRpt* messages to the COSMOS.

6. Types of Sensing

In COSMOS, several sensing types are defined to make implementation efficient and interoperable. Table 2 shows part of the defined sensing type list.

V. Implementation and Performance

The COSMOS functionalities are continually being enhanced. Therefore, a full performance assessment of the COSMOS is not provided here. All of our tests were performed using a sensor network simulator developed within the COSMOS project. These performance tests focused on the performance of sensor network common interface processing between the COSMOS and the sensor network (simulator). For a client using all of the services provided in the COSMOS, the processing time would be longer than the results given here. The COSMOS was developed in Java, and it uses the SPRING framework. In the case of Open API, the Axis is used to provide a web service. The test-bed to measure performance was designed with the following specifications:

COSMOS (sensor network interface manager)
- Language: Java 1.5.0.13
- Framework: Spring 2.0
- Database: derby 10.3.1.4 (for logging)
- CPU: Mobile DualCore Intel Core Duo T2500, 2,000 MHz
- Memory: 2,048 MB
- HDD: 100 GB, 5,400 RPM, S-ATA
- OS: Microsoft Windows XP Professional

Sensor network simulator (SN simulator)
- Language: Java 1.5.0.13
- CPU: QuadCore Intel Xeon E5345, 2,333 MHz
- Memory: 8,192 MB (8 GB)
- VGA: ATI Radeon HD 2600 XT (256 MB)
- HDD: 400 GB, IDE
- OS: Microsoft Windows Server 2003, Enterprise Edition

1. Sensor Network Simulator

The SN simulator simulates both an adaptor and a sensor network. It provides menus to create a sensor network and an RFID reader. When creating a sensor network, a user can select and/or input an MTB, a sensing mode, a sensor network identifier, a maximum command number, connection information (IP address, port number), a monitoring type, a buffering interval, a number of sensor nodes, supported sensor types, and so on. These parameters can be easily selected by using a graphical user interface (GUI) as shown in Fig. 10. A simulator user can also modify necessary metadata of a sensor network after creating it.

With the simulator, the user can dynamically invoke some abnormal conditions, such as sensor node addition or deletion, network connection failure, and other error conditions. After creating a sensor network or an RFID reader, a user can save their profiles in XML and reopen them later. The battery level decreases with the passing time to reflect a real situation.

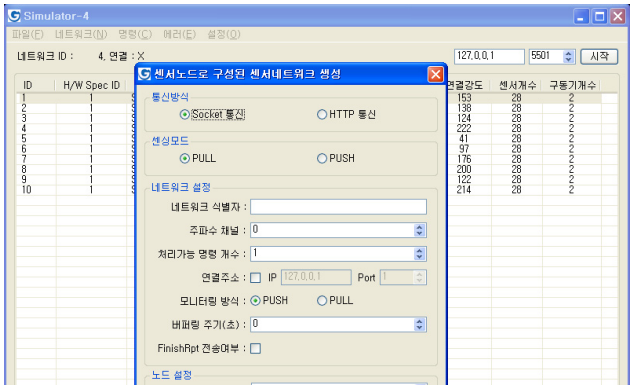

Fig. 10. COSMOS sensor network simulator.

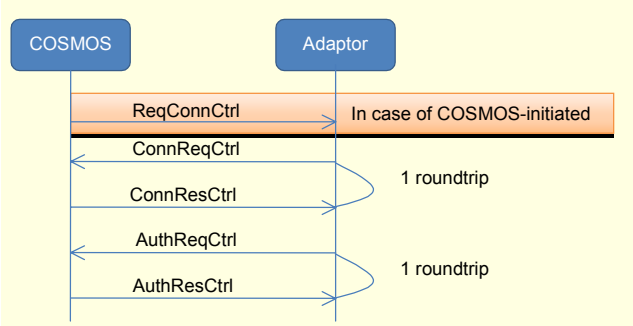

Fig. 11. Sensor network connection messages flow.

Sensor data can be generated in four modes: incremental, decremented, random, and file.

2. Sensor Network Connection Establishment

The COSMOS supports two modes of sensor network connection establishment. For COSMOS-initiated sensor network connection, the COSMOS requests the sensor network to connect to it. This type of connection requires 5 messages, that is, 2.5 roundtrips. For sensor network-initiated sensor network connection, the sensor network requests connection to the COSMOS and requires 4 messages, that is, 2 roundtrips. Figure 11 shows the message flow of both cases. Authentication is performed by using the ID/password or certificate. The method is determined at registration time at the USN directory service.

The purpose of this test was to measure the connection times via XML message over HTTP and via binary message over TCP. In each case, both authentication methods (ID/password-based, certificate-based) were tested. Therefore, there are eight cases. Table 2 shows the measured times for each case.

- COSMOS-initiated connection:
  - ID/passwd-based authentication

$$T_{\text{total\_connect}}=\{T_{\text{cmp}}+T_{\text{amp}}+T_{\text{td}}\}\times 5+T_{\text{auth\_idpasswd}}$$

  - Certificate-based authentication

$$T_{\text{total\_connect}}=\{T_{\text{cmp}}+T_{\text{amp}}+T_{\text{td}}\}\times 5+T_{\text{auth\_certificate}}$$

- SN-initiated connection:
  - ID/passwd-based authentication

$$T_{\text{total\_connect}}=\{T_{\text{cmp}}+T_{\text{amp}}+T_{\text{td}}\}\times 4+T_{\text{auth\_idpasswd}}$$

  - Certificate-based authentication

$$T_{\text{total\_connect}}=\{T_{\text{cmp}}+T_{\text{amp}}+T_{\text{td}}\}\times 4+T_{\text{auth\_certificate}}$$

$T_{\text{total\_connect}}$: total connection time
T_{cmp}: processing time for a message in the COSMOS
T_{amp}: processing time for a message in the adaptor
T_{td}: transmission delay (COSMOS ⇔ adaptor)
$T_{\text{auth\_idpasswd}}$: processing time for ID/passwd authentication
$T_{\text{auth\_certificate}}$: processing time for certificate authentication

Table 3. COSMOS-adaptor connection time.

	XML over HTTP		Binary over TCP	
	ID/Password	Certificate	ID/ Password	Certificate
COSMOS-initiated	2,560 ms	2,547 ms	2,106 ms	2,015 ms
Adaptor-initiated	2,331 ms	2,228 ms	2,141 ms	2,325 ms

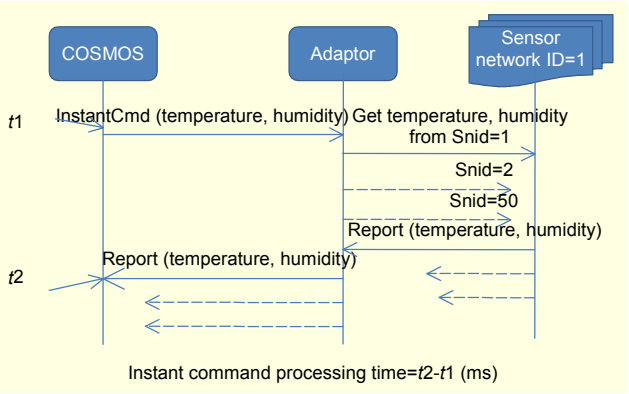

Fig. 12. Query processing flow.

As seen in Table 3, the connection time when the XML over HTTP communication protocol is used is a little bit longer than the processing time when the binary over TCP communication protocol is used. Regarding the authentication method, the difference is negligible. According to the results, the connection establishment takes about 2 seconds in all cases, which is tolerable in real fields.

3. Query Processing in Multiple Sensor Networks

The purpose of this test was to assess the performance degradation of the COSMOS when the numbers of sensor networks which process the same query simultaneously are gradually increased. The processing times were measured from

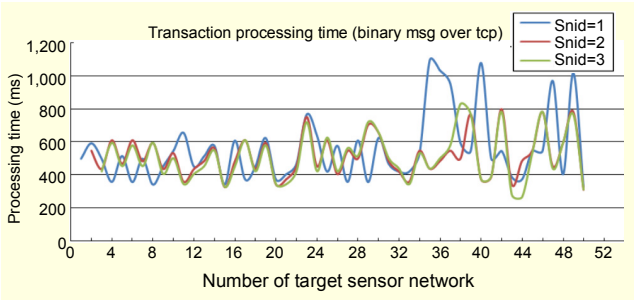

Fig. 13. Transaction processing time.

the query sending time ($t1$) at the COSMOS to the receiving time ($t2$) of sensor data at the COSMOS. Figure 12 shows the test method.

- The tests were performed under the following conditions.
- The binary over TCP MTB was used.
- 50 sensor networks were prepared with 100 sensor nodes each.
- Each sensor node was equipped with two sensors (temperature and humidity).
- The number of sensor networks which process queries at the same time was increased from 1 to 50.
- An instant query was sent to each connected sensor network.
- The processing time $\{t2–t1\}$ was measured for the same sensor networks (Snid=1, Snid=2, and Snid=3).
- $T_{total\_sensing}$= query creation time at COSMOS + transmission delay + query processing time at SN simulators + report creation time at SN simulator + transmission delay + report processing time at COSMOS = $\{T_{cmp} \times$ the number of sensor networks operating + T_{td} + T_{amp} + $T_{amp} \times$ the number of sensor nodes operating + T_{td} + $T_{cmp} \times$ the number of sensor networks operating network $\times$ the number of sensor nodes operating
- $t1$ is the time when the COSMOS sends an *InstantCmd* message to each sensor network.
- $t2$ is the time when all reports from the sensor networks are received.

Figure 13 shows the processing times of Snid=1, Snid=2, and Snid=3 as the number of running sensor networks was increased gradually. Some peaks are experimental exceptions and can therefore be ignored. The processing times within the COSMOS are almost the same, even when the number of sensor networks increases to 50. The COSMOS is intended to support integrated query processing over multiple heterogeneous sensor network environments. In implementing sensor network common interface processing, any of a variety of sensor network types can be handled by the COSMOS at the same time. Figure 13 shows that the query processing performance of the COSMOS over 50 sensor networks with 100 sensor nodes, each equipped with two sensors, is tolerable around 400 ms. Moreover, the number of sensor networks processing simultaneously does not significantly degrade the performance of the COSMOS. Technically, the whole query processing time is dependent on the processing time within each sensor network, but the pure performance of the COSMOS can be analyzed separately.

VI. Conclusion

In this paper, we presented a new solution for integrated sensor data processing over heterogeneous sensor networks. The COSMOS handles heterogeneous sensor networks and provides common functions, such as query processing, metadata definition/management, and so on, for various monitoring applications. We demonstrated by simulation that the COSMOS provides integrated sensor data to multiple applications by using sensor data from multiple sensor networks with reasonable performance.

This has great potential for applications in remote monitoring fields. By using the COSMOS, remote monitoring applications can check the current environment status and control the environment without human effort. The advantage of the COSMOS over other interfaces defined by IEEE1451, SensorML, and GSN is that current non-standardized sensor networks can be used in many applications without any modification. They only need to implement appropriate adaptors to communicate with the host. This will encourage the active proliferation of sensor-related industries.

There are still some unresolved issues. They include the handling mobile sensor nodes, access to devices as sensor-node units for nodes which have IP addresses, expansion of sensing types, and the interaction between the COSMOS and the COSMOS for global scale service provision. In our future research, we will deal with these issues.

References

[1] C. Karlof and D. Wagner, "Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures," *ScienceDirect Online*, vol.1, 2003, pp. 293-315.

[2] Y. Wei, J.Heidemann, and D. Estrin, "An Energy-Efficient MAC Protocol for Wireless Sensor Networks," *Proc. 21th Annual Joint Conf. the IEEE Computer and Comm. Societies*, vol. 3, 2002, pp. 1567-1567.

[3] Y. Yao and J. Gehrke, "The Cougar Approach to In-Network Query Processing in Sensor Networks," *ACM SIGMOD Record*, vol. 31, 2002, pp. 9-18.

[4] M. Botts et al., *OGC Sensor Web Enablement: Overview and High Level Architecture*, OpenGIS White Paper, 2007.

[5] K. Aberer, M. Hauswirth, and A. Salehi, "A Middleware for Fast and Flexible Sensor Network Deployment," *VLDB*, 2006, pp. 1199-1202.

[6] Y.B. Kim, M. Kim, and Y.J. Lee, "COSMOS: A Middleware Platform for Sensor Networks and a u-Healthcare Service," *Proc. 23rd ACM Symp. Applied Computing*, 2008, pp. 512-513.

[7] T.H.H. Vu et al., "Spatiotemporal Pattern Mining Technique for Location-Based Service System," *ETRI Journal*, vol. 30, no. 3, June 2008, pp. 421-431.

[8] M. Sgroi et al., "A Service-Based Universal Application Interface for Ad Hoc Wireless Sensor and Actuator Networks," *Ambient Intelligence*, Springer-Verlag, 2005, pp.149-172.

[9] Y. Yao and J. Gehrke, "The Cougar Approach to In-Network Query Processing in Sensor Networks," thesis, Cornell University, 2002.

[10] C.C. Shen, C. Srisathapornphat, and C. Jaikaeo, "Sensor Information Networking Architecture and Applications," *IEEE Personal Comm. Magazine*, 2002. pp.52-59.

[11] G. Fedak et al., "XtremWeb: A Generic Global Computing System," *Proc. 1st Int'l Symp. Cluster Computing and the Grid*, 2001, pp. 582-587.

[12] M. Botts and A. Robin, *Sensor Model Language (SensorML) Implementation Specification*, OpenGIS Implementation Specification, 2007.

[13] IEEE Instrumentation and Measurement Society, *IEEE Standard for a Smart Transducer Interface for Sensors and Actuators-Common Functions, Communication Protocols, and Transducer Electronic Data Sheet (TEDS) Formats,* IEEE Std 1451.0, 2007.

[14] Y. Doh, "USN Applications & Technologies," *RFID/USN Korea*, Oct. 2005.

[15] M. Kim, and Y.J. Lee, *The Standard Interface for Heterogeneous Sensor Networks*, TTA, TTAS.KO-06.0169, 2007.

[16] M. Kim, and S. Yoo, *Proposal for New Draft Recommendation on USN Middleware Reference Model*, ITU-T AVD3121, 2007.

[17] M. Kim, J.W. Lee, and S. You, *Proposal for a Draft Recommendation on USN Middleware Reference Architecture*, ITU-T AVD 3370, 2008.

[18] M. Kim and S. Yoo, *Service Description and Requirements for USN Middleware*, ITU-T F.usn-mw, 2008.

Marie Kim received the BS and MS degrees in computer science from Sogang University, Rep. of Korea, in 1996 and 1998, respectively. She is currently with the ETRI, Rep. of Korea. She was a research member of the IMT 2000 project with Samsung. Her research interests include the development and standardization of mobile RFID middleware and USN middleware.

Jun Wook Lee received the MS and PhD degrees in electrical and computer engineering from Chungbuk National University, South Korea, in 1997 and 2003, respectively. He is currently working at ETRI, South Korea. His major research interests include sensor data mining, sptatio-temporal data mining, location-based services, context awareness in the ubiquitous computing environment, and the development of a USN middleware platform.

Yong Joon Lee received the MS degree from Yonsei University, Korea, in 1988, and the PhD degree in computer science from Chungbuk National University, Korea, in 2001. He is currently working at ETRI, Korea. His major research interests include RFID and sensor data management, sensor data mining, context awareness in ubiquitous computing environment, and the development of sensor network middleware.

Jae-Cheol Ryou received the BS degree in industrial engineering from Hanyang University in 1985, the MS degree in computer science from Iowa State University in 1988, and the PhD degree in electrical engineering and computer science from Northwestern University in 1990. He joined the faculty of the Department of Computer Science at Chungnam National University, Korea, in 1991. His research interests are Internet security and electronic payment systems. He is currently with the Internet Intrusion Response Technology Research Center (IIRTRC) of Chungnam National University, Korea.