2008

An architectural framework for developing advanced integrated environmental monitoring systems

Xitao Xing
University of Windsor

AN ARCHITECTURAL FRAMEWORK FOR DEVELOPING ADVANCED INTEGRATED ENVIRONMENTAL MONITORING SYSTEMS

By

Xitao Xing

A Thesis
Submitted to the Faculty of Graduate Studies
through the Department of Earth and Environmental Sciences
in Partial Fulfillment of the Requirements for
the Degree of Master of Science at the
University of Windsor

Windsor, Ontario, Canada

2008

Your file Votre référence
ISBN: 978-0-494-47065-7
Our file Notre référence
ISBN: 978-0-494-47065-7

Author's Declaration of Originality

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

Abstract

The environment has become a topic of great public and academic concern. Monitoring studies of environmental variables are needed to allow control strategies and policy-/decision-making to be applied effectively. This thesis presents an overview of the main issues and requirements for the capabilities of integration, flexibility and scalability for complex environmental monitoring applications. The scope and depth of the topics are considerable and are developing rapidly in line with new technology and computational techniques. The author proposes and designs an architectural framework to develop advanced integrated environmental monitoring systems (A-ITEMS), which feeds the requirements of complex environmental monitoring systems. Afterwards, in terms of the theoretical and technical investigation on the A-ITEMS, this thesis demonstrates the key ideas by implementing an Integrated Watershed Telemetry (IWT) system. The practical design and simulation implementation of the IWT system does show that the A-ITEMS has the significant flexibility and capability to adapt to complex environmental monitoring applications.

Acknowledgements

I would firstly like to express my thanks to my supervisor Dr. Phil A. Graniero for his valuable professional guidance, thoughtful insight and financial support throughout this thesis. Dr. Graniero is also acknowledged for giving me an opportunity to study in an excellent lab, The Multi-purpose Environmental Modelling Facility (MEMF Lab) at University of Windsor. His excellent supervision, encouragement and support are greatly appreciated. I firmly believe his guidance and advice will benefit me beyond my study into my future career and life.

I would like to thank my parents, who have supported and inspired me to pursue all of my education. I am also indebted to my parents-in-law for their untiring support.

My warmest thanks go to my dearest wife, Yong Ge, for her continuous encouragement and support. Her love gives me the motivation to live and study in a new country. And my sweetest thanks to my son, Andrew Z. Xing. His birth gives me much more pleasure and strength.

I also thank Ontario Centres of Excellence (OCE), and the Geomatics for Informed Decisions (GEOIDE) NCE for providing the financial support through my graduate studies, and Solinst Canada Ltd. for providing funding, equipment, access to their technical staff, and work space at their head office during this research.

Last but not least, special thanks to my colleagues, Nafaâ Jabeur, James McCarthy and Dan D'Alimonte, for their kindness and tireless help. I can always find sparkles in our professional discussions.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

A-ITEMS	Advanced Integrated Environmental Monitoring System
BS	Base Station
DB	Database
DBMS	Database Management System
DPC	Data Processing and Control
EM	Environmental Monitoring
EMP	Environmental Monitoring Program
EMS	Environmental Monitoring System
FS	Field Station
GIS	Geographical Information System
GUI	Graphical User Interface
HS	Home Station
IDE	Integrated Development Environment
IP	Internet Protocol
ISO	International Organization for Standardization
IWT	Integrated Watershed Telemetry
LRM	Long Range Modem
MVC	Model-View-Control
ODBC	Open Database Connectivity
OOP	Oriented Object Programming
OSI	Open Systems Interconnection
RDA	Real-time Data Acquisition
RDO	Remote Data Objects
RF	Radio Frequency
RRL	Remote Radio Link
RTU	Remote Terminal Unit
SRM	Short Range Modem
STS	Solinst Telemetry System
TCP	Transmission Control Protocol

Chapter 1 Introduction

In order to appropriately respond to environmental risks which are diversified and difficult to be realized, it is required to comprehensively grasp information about factors, which have influence over the natural environment and their interrelationship, and establish an environmental monitoring (EM) to reflect the knowledge in administrative measures and activities. EM is one of the keys to effective management of environmental situation. It has a good track record in most stages of environmental and resource protection-from initially identifying environmental problems to finally providing direct evidence in enforcement actions. Decisions based upon monitoring results can be far-reaching, requiring a comprehensive data base that is both accurate and reliable. Data from EM may be into valuable information for many applications, e.g. the continuing assessment of the effect of pollutants and pollution on human, the natural and the modified environment; planning resources used and product and process changes in order to minimize environmental impacts, etc.

Conceptually speaking, the fundamental EMS is a system platform which has the capability to design monitoring networks, receive, process, manage, analyze, assess and report monitoring data. In the environmental domain, the EMS is thought of as a versatile, flexible and cost-effective data collection means to accumulate unattended data from multiple sensors at remote locations over an extended period of time (Colbert et al., 1971). It focuses on data acquisition and transmission. The use of EMS can make early detection of problems and implementation of effective measures possible. It is usually important to detect environmental changes and their causes promptly, as late realization of the actual conditions and slow taking measures would increase bad influence to the environment and the cost for the recovery.

Many specific EMSs have been developed for various environmental topics, such as Air, temperature, water, land, ecology, noise, radiation, etc. For example, in 1984, the Canadian Forest Service (CFS) established the Acid Rain National Early Warning System (ARNEWS), to detect early signs of air pollution damage to Canada's forests. Since that time, more than 150 ARNEWS plots have been established across Canada to monitor changes in forest vegetation and soils caused by air pollution and environmental change (D'Eon et al., 1994). In 1995, Global Climate Observing System (GCOS) was developed by the World Meteorological organization to integrate numerous in-plate monitoring programs, and measure global climate changes and to determine their causes and ecological and sociological consequences. Global Environmental Monitoring System (GEMS)/Water is a UN program on global water quality that was initiated in 1976 by United Nations Environment Programme (Brydges, 2004).

1.1 Environmental monitoring systems

1.1.1 Primary functions

In general, an EMS normally serves five primary functions (Lenzerini, 2002).

Data acquisition

It is the most fundamental capability for an EMS to acquire data via sensor networks. Usually, data acquisition includes some particular devices, e.g. sensors, stations, etc and the corresponding software which supports the work of devices. One so-called two-ways data flow may be adopted to meet data acquisition. One way is that data are pushed into one monitoring centre once the monitoring infrastructure is formed. In order to implement this way, all monitoring components have to be correctly set up in advance, and may need some filters to get the required data because users are unable to control data flow. The other way gives users the chance to control data acquisition. If users need data, they can send some specific commands to sensor networks, and then monitoring field devices can be activated to transfer data to the monitoring centre. Without users' commands, the devices will cache data internally or stay idle or even sleeping. Sometimes, these two ways may be combined by setting particular schedules for the devices. Once the devices reach the schedules, they will be activated to transfer data.

Alarm recognizing and processing

Alarms are from critical situations, and their recognition gains one of the most important concerns in the environmental domain. Once an alarm happens, it should be sent to the monitoring centre. It always has the most prior level to transfer. People may utilize some special communication lines, e.g. land line, to transfer alarm data. Generally speaking, alarms are detected by triggers, most of which normally deposit in the devices. However, some alarms may come from normal data sets and be recognized by triggers in the monitoring centre, e.g. tendency. Alarm recognition depends upon the alarm rules, and could be more complicated if it is related to other phenomena. So, usually, one subsystem is developed to process alarm situation.

Data management

Data are generally archived for current and future data analysis. It is essential that a data management system, which is reliable, practical, efficient, and may incorporate GIS capabilities, be adopted. Standardized metadata need to be designed. Data management is also viewed in a broad sense and the

system incorporates more than just quantitative information. For example, it is important to keep thorough records of all management interventions as part of the data management exercise. If data is produced and stored across a number of agencies or organizations, a strong data management system can be a valuable tool for driving inter-agency collaboration and ensuring that the most is made of monitoring data.

Because of data types and formats involved and associated knowledge in a flexible and accessible format to a broad user community, environmental data management faces the considerable challenge of providing integrated information systems for managing and presenting a diversity of environmental monitoring data. The emerging field of environmental informatics broadens the scope of data management, viewing it as an integrated component of the monitoring system of transforming raw environmental data into higher-grade knowledge suitable for decision making (Lane et al., 2004).

Data assessment and analysis

An EMS assesses data to know whether the acquired data meet the monitoring requirements, especially on data quality and quantity. Through data assessment, users can know if monitoring area is completely covered, or the data sampling rate is feasible, etc. Data with unknown quality are in effect unreliable and have little confidence to be placed in their analysis and interpretation. In order to fulfill data assessment, some particular standards would be adopted, such as ISO 9000 series, which emphasize standardizing procedures for quality management. And meantime, data assessment is always designed as an integral component of the data quality subsystem in the EMS. It may work together with data analysis. If data is assessed successfully and data quality is met, then data analysis can be implemented. Model design is the key of data analysis. And different models may lead to definitely different results, which can affect final decision making.

Data / information reporting

Monitoring information need to be reported to users or collaborators to present the situation of monitored environmental objects, which is one of the basic capabilities on the application level of the EMS. There is a clear distinction between data and information, where data is a measurement that can be directly from monitoring networks and disorganized, whereas information is the result of processing, manipulating and organizing data in some way. In other words, people can think of information as one kind of knowledge that comes from data. In the EMS, both of data and information could be published to meet various applications. Usually, the reporting component is developed to present information. Data sharing is also important and may face some critical technical

and business issues, for example, data security, data fusion, data standards, etc. In all, the reporting component is the "window" demonstrating the objectives of environmental monitoring systems and the current situation of monitored environmental objects.

1.1.2 Primary components

Generally, an EMS has three fundamental activities: acquiring data, assessment and reporting. Acquiring data is the systematic collection of data for the purpose of monitoring the environment, as opposed to collection of field data primarily to support a scientific study. Assessment is the process of analyzing and evaluating the resulting monitoring data, together with other scientific evidence, to support policy-making (Messer, 2004). Reporting is to share the monitoring results with users.

Based upon these three activities and capabilities of an EMS, an EMS has five primary components presented in Figure 1.1. Three components, Front Communication Controller, Data processing Unit and Data Management, constitute the Real-time Data Acquisition (RDA) subsystem. Monitoring networks in field detect the environment and create measurements. Data are received through Front Communication Controller, processed by Data Processing Unit, and eventually stored and managed in Data Management component. Once users gain data, they can analyze data and evaluate the environmental situation. Analysis and Assessment component is supported by some predefined monitoring rules and environmental models. There are many schemes for Reporting component to work well, e.g. publishing data or information via internet.

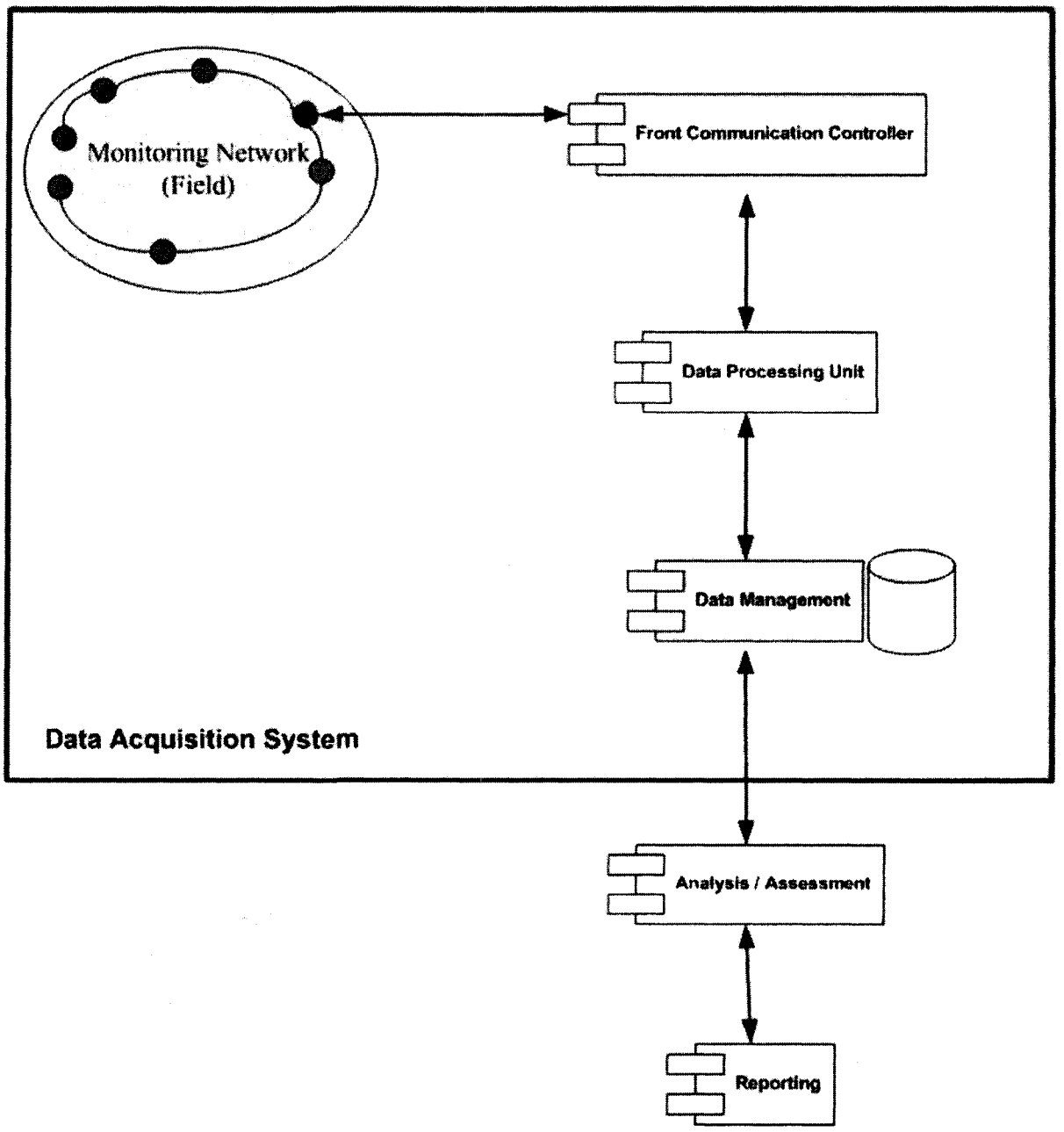

Figure 1.1 Components of General EMS

Front Communication Controller provides the front pipe to transfer messages. It has the capability to transfer any message and avoid loss of messages. Concurrent data transaction and high sampling rate may become the challenge for this component. Concurrent data transaction implies that various data sources could arrive at the same time. Putting all data in an order and then processing them one by one would become more and more burdensome and may finally crash the system. High sampling rate brings a number of data and requires a quite highly efficient way to avoid possible data loss.

Data Processing Unit is the core of an EMS and processes messages. It accomplishes extracting incoming messages, building outgoing messages, processing alarm messages and outputting data to data management. Front Communication Controller provides the data pipe to transfer messages to and from Data Processing Unit.

All acquired data, either real-time or historical, need to be saved somewhere. A complete data storage component not only deposits data but also manages data. **Data Management** usually consists of many features, such as metadata, data quality, data table, security rules, etc., each of which establishes one specific job.

Data Analysis / Assessment is often necessary to conduct some exploratory data analysis early on a monitoring study to ensure data integrity and reconcile the collected data with expectations. Some data preparation, such as removal of obvious erroneous observations or data transformation, is often necessary prior to analysis. The appropriate statistical and mathematical analyses may be decided in the design phase, and well before the data is collected, because carefully considered data analysis may have implications for the spatio-temporal monitoring design and help ensure that users maximize the opportunity to address the objectives. That said, statistical and mathematical modeling are often an iterative process and the analysis should be adapted to best suit the data and collection characteristics. It is recommended that statistical diagnostics be examined to check assumptions and ensure the most appropriate analysis is used (Henderson et al., 2005). Environmental monitoring data usually have both of time and spatial properties, which can present the change with time or the spatial distribution of monitoring environmental objects. The environmental situation of one definite location may be affected by the combination of chemical physical, biological, or ecological facts. Data from these facts are fused to discover the cause-and-effect relationships.

Reporting publishes the information for the public. Data Analysis / Assessment component creates valuable information for users to understand the current situation as well as the future tendency of environmental change. The results may be reported to the public by various ways. For example, the geospatial system has become a vital way to report environmental data or information.

1.1.3 New technology, approaches, and opportunities

With advances in hardware and wireless network technologies, low-cost, low-power, multifunctional miniature sensor devices have been created. Each sensor node is equipped with five units: sensing hardware, a limited memory, a radio transceiver or other wireless communications device, a small microcontroller, and a power source, usually a battery. It has wireless communication capacity and

sufficient intelligence for signal processing and for disseminating the data. It is becoming possible to have hundreds or thousands of ad hoc tiny sensor nodes spread across a geographical area.

These sensor nodes can collaborate among themselves to establish an ad hoc sensing network, or sensor web. A sensor web can provide access to information anytime, anywhere by collecting, processing, analyzing and disseminating data (Tubaishat et al., 2004). The idea of sensor networks (Culler et al., 2004) has been made popular by UC Berkeley who also developed a series of sensor nodes called mica nodes (Hill et al., 2002). Sensor webs can figure out the interconnection and cooperation within intelligent sensors, and data gathered by a particular node on such a network can influence the behavior of another node (Delin et al., 2001). They include some basic requirements: large number of sensors, low energy use, efficient use of the small memory, data aggregation, network organization, collaborative signal processing and query ability.

Sensor webs promise to revolutionize sensing in a wide range of application domains, which is because of their reliability, accuracy, flexibility, cost effectiveness and ease of deployment (Tilak et al., 2002). Smart sensors can offer vigilant surveillance and can detect and collect data concerning any sign of machine(s) failure, earthquakes, floods, and even a terrorist attack. Sensor networks enable: 1) information gathering, 2) information processing, and 3) reliable monitoring of a variety of environments for both civil and military applications (Tubaishat et al., 2004). Many initial sensor webs have been deployed for environmental monitoring, which involves collecting readings over time across a volume of space large enough to exhibit significant internal variation. Researchers are using sensor webs to monitor nesting seabird habitats and microclimate chaparral transects and to conduct analogous studies of contaminant propagation, building comfort, and intrusion detection. One example, monitoring the microclimate throughout the volume of redwood trees, helps form a sample of entire forests (Culler et al., 2004).

1.1.4 Current problems

Current EMSs usually need to couple several technologies in an environmental monitoring system, e.g. communication protocols & transmission platforms. If the underlying technology in some components change, the system may not support the new technology and not only key components but the entire system architecture may have to be developed again.

Current EMSs lack open mechanism to support new monitoring hardware and data sources. Some new monitoring hardware could be added into current sensor networks. They may be from various vendors and have different firmware and data protocols. New data sources have different data format

and specifications. In order to recognize these new hardware and data sources, current EMSs usually have to develop some new corresponding components, which make the system have more and more redundant codes.

Some systems need central processing, and some need distributed processing. Central processing and distributed processing have individual advantages. Central processing has the better performance on processing and managing data, system maintenance and data security, whereas distributed processing can assuage the workload of the monitoring centre and has more flexible to deploy the system. Different EMSs may adopt various processing approaches in terms of specific requirements, but current EMSs are not able to support both approaches.

It is difficult to remotely modify behaviours of sensors and stations. In an EMS, the normal operation is that users set up all monitoring hardware, e.g. sensors and stations, and then deploy them in the field. If users want to modify the behaviours of the hardware, e.g. changing the sampling rate of one sensor, they usually go to field. Such operation would need much time and cost, and sometimes may lose some important data.

1.2 Thesis objectives

The goal of this thesis was to design a general architectural framework that can be used to guide the development of new, innovative environmental monitoring technologies. This was accomplished by completing the following three objectives:

1. Identify the functional and structural requirements for integrated, flexible, cost-effective technologies that can address complex environmental monitoring problems;

2. Develop the architectural framework for an advanced integrated environmental monitoring system (A-ITEMS) that can support the above requirements; and

3. Design the software for a real environmental monitoring system and a station simulator, using the features of the A-ITEMS architectural framework, in order to demonstrate and test the framework's utility.

1.3 Research benefits

The architectural framework resulting from this thesis would help developers build scalable, integrated monitoring system components. The framework presents some fundamental domain object

functional object models, which are generic representations of primary structures and features for EMS. Developers can easily inherit these models and create some new objects to build their own system components. The loosely coupled relationships among these models would allow developers to easily extend their monitoring systems, or reconfigure them into different product variations for different applications.

Simulated components such as field stations can be blended with real, operational environmental monitoring systems to give users more choice in optimizing and verifying an EMS design. Simulation mechanism can help users design an effective monitoring system before they deploy a live field system. An EMS usually needs to be correctly set up and then can be activated to acquire data. However, in order to get ideal data, some parameters may need to be adjusted in terms of the performance of the monitoring system. Though users can remotely modify some of them, they will still spend much time and money on testing a real live system. Simulation mechanism would be convenient to test the system before the system is deployed in the field. For example, the case study shows how a station simulator and the simulation mechanism can be used to test performance effects of different sampling rates and reporting rates.

Furthermore, prototype or operational EMS software structured with the A-ITEMS framework can be extended, or components can be replaced by alternate components that fulfill the responsibilities in a different fashion. This allows the software, and EMS as a whole, to be used as an experimental platform for developing new tools, methods, and approaches for distributed sensor networks and environmental monitoring. For example, the design trade-offs between central processing at a Home station versus distributed processing at higher-capacity (and higher-cost) field stations can be investigated.

1.4 Thesis outline

In Chapter 2, the requirements for an advanced integrated environmental monitoring system are gathered. In Chapter 3, the resulting A-ITEMS architectural model is presented. Chapter 4 contains a case study in which the A-ITEMS architectural framework was used to design the 'home station' software that manages a new environmental monitoring system under commercial development. Some basic performance testing examples are presented to illustrate the use of the framework for software design and evaluation. Finally, in Chapter 5, some main conclusions are drawn.

Chapter 2 Requirements Analysis

From October 2006 through to March 2007 I spent an average of three weeks out of each month conducting my research in the Research and Development department at Solinst's main office in Georgetown, Ontario. During this time I examined their existing sensor and telemetry products and their capabilities by studying their hardware and their software source code, and interviewing the R&D staff. I also participated in monthly meetings with Dr. Graniero, Solinst's executives, and their R&D staff, where customers' environmental monitoring requirements and challenges were discussed, and the requirements for a new generation of environmental monitoring hardware and software were identified. During the other weeks I conducted my research in MEMF Lab at the University of Windsor. I reviewed literature and learned background relevant to designing an environmental monitoring framework, and I developed the architecture and design approach with Dr. Graniero.

I used the information gathered from the literature review, design and planning meetings, and investigation at Solinst to produce a list of requirements for an advanced integrated environmental monitoring system, which is presented in this chapter.

2.1 Functional requirements

2.1.1 Modular design for extending hardware and communication support

The EMS would support concurrent operations in a distributed infrastructure and avoid the bottleneck of specific technologies, e.g. transmission protocols, data repository, hardware firmware of different vendors, and so on. In such system, heterogeneous monitoring networks can work independently and be flexibly adjusted with the change of monitoring requirements.

In most cases, we can not completely forecast what functions will be required in a system. What we usually do is to meet the current requirements, which unfortunately could make one possible limit in future. For example, with the change of system operation environment such as new hardware, communication, users' requirements, etc., we will have to spend more and more time on the maintenance of the system, and furthermore could have to rebuild the entire system.

Different users could adopt various technical schemes, such as firmware, protocol, communication, data warehouse, etc. In order to flexibly support different technical schemes, modular design is obviously required. For example, users may utilize different databases, such as Microsoft Access, DBase, Oracle, etc. These databases have their specific access and operation strategies. We can

design database or data warehouse as an independent module and develop some common interfaces to feed various conditions. This module can be changed with particular applications; however it does not affect other modules. Modular design is not only helpful to maintain the system, but also flexible to support different applications.

The general EMS can only recognize its predefined data sources. However, for some reason, new data sources may be needed by the monitoring program to improve the monitoring performance. These new data sources could bring some unknown data, which may be rather important and can not be ignored. For example, watershed monitoring may need some temperature data from other agencies or directly add some temperature sensors. But these data sources may not be predefined in the system. Once these data arrive, the system can not recognize them and leave them as garbage. So how to feed new data sources need be studied and is quite challenging to the EMS. In the A-ITEMS, we will propose some schemes to solve this requirement later.

2.1.2 Feed new data sources by data integration technology

The registration mechanism is adopted to feed new data sources. In terms of different data sources, the registration mechanism has various requirements. In general, there are three kinds of new data sources. i) The new data source directly comes from other monitoring networks. As presented in Figure 2.1, data from network B can be transferred into network A. In Figure 2.1, the solid line represents the previous data pathway and the dash line means the new or changed data pathway. In order to register network B in DPC (A), the configure file, which completely describes network B, must be provided. Before network B transfers its data to DPC (A), DPC (B) first sends the configure file to DPC (A) and finishes the registration of network B. ii) The new data source is from other data repositories. In Figure 2.1, DPC (A) acquires data from data depositories (databases, files etc.) in DPC (B). One specific metadata of DPC (B) must be formed and sent to DPC (A) for registration. iii) New sensors or field stations create another new data source for DPC (A). They must build their configure files and accomplish their registration in DPC (A). The configure files can be provided by some way, e.g. new field station can make its configure file. Once it is added into one network, it will first send the configure file to DPC. It cannot transfer data until it is done its registration in DPC. Based on the registration mechanism, the system can conveniently support new types of field stations or sensors, which are from different vendors.

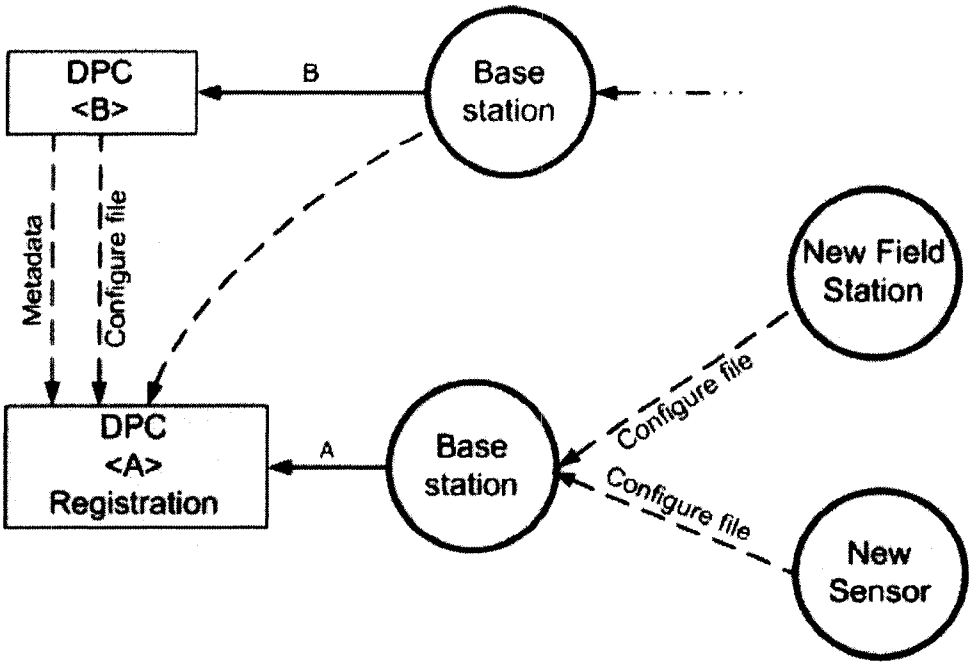

Figure 2.1 Registration of new data sources

Except the import of configure files, the specific data protocols of new data sources are also critical. However, because of the modular design of data protocol library, the new data protocol is quite convenient to add in. Of course, the configure files appoint the corresponding data protocol for each data source.

2.1.3 Transience, movability and mobility

The station power is limited and very important because one station needs enough power to work. In order to save power and normally transmit data, the schedule mechanism is used to adjust the work of the station. One station may be idle or sleeping at some particular time or in an interval. For some reasons, e.g. saving power, updating, etc., some stations may stop working. Then users need to temporarily deactivate stations or add some new stations. These conditions may lead to the topological change of monitoring networks. Though sometimes the condition exists transiently, the EMS still has to adjust the structure of monitoring networks in terms of possible logical relationships and spatial information of stations.

The position of stations and connection relationships among stations may be modified from time to time. Some stations may need to move to more effective locations, perhaps to improve radio coverage or to better capture environmental dynamcis. An EMS should be able to keep track of where a station is located now as well as where it was in the past. It is also desirable for the EMS to determine

whether some stations should move, where they move and also how to adjust the topology of the monitoring network.

Compared to the movability, the mobility is more challenging to handle, but far more efficient to acquire and transmit data. Some stations or special devices may be mobile to acquire data (e.g. Graniero and Miller, 2003). The location is not just a simple attribute of sensors and therefore the measurement data they produce, but also a critical factor which influences the topology of communication networks.

2.1.4 Expandable and adjustable monitoring networks

The EMS may need to support multiple monitoring sub-networks, each of which has a base station or bridging station directly communicating with a DPC. With the support of communication and appropriate scheduling control, the monitoring network theoretically can be scalable. However, the bigger the monitoring network becomes, the more latency it may have. Therefore, the monitoring network is usually designed upon specific applications. When users set up the schedules to control the behaviour of a station, they should try to avoid possible conflicts because of some facts, e.g. communication, protocol, etc.

2.1.5 Support different transmission platforms and communication protocols

Communication is the heart of EMS. The future of transmission platforms profoundly affects the future of EMSs. Basically, the features of communication include bearer, bandwidth and transport protocol, where bearer is the medium over which communication travels and can be divided into two broad categories: landline and wireless. Bandwidth is quite important because it can determine how many data can be transferred within a period of time. In the environmental monitoring domain, numerous real-time data must be timely sent to users. Some particular cases could produce several measurements every one second. Sometimes, the communication bandwidth can become the bottleneck because the less data would influence the result of assessment and analysis. The transport protocol is an agreed set of rules to allow devices to communicate with each other. Typically, it will have error detection techniques and addressing so that the messages can go to the correct place and be extracted correctly. The big issue of transport protocols is whether they are open or proprietary, which is quite related to professional standardization such as local standards, national standards and international standards. Because of some particular reasons, the system may be required to support some kind of special transport protocol. The communication coverage is also quite important, especially for the routing design of monitoring networks.

The basic wireless communication is radio frequency, which provides the communication between RTUs and the master station. Currently, cellular communication has been used more because it is capable of carrying much more data and has good coverage. More and more modems are to use IP or Internet protocol addresses. Certainly, satellite communication is also one of primary transmission platforms and normally adopted in some special applications because of its expensive cost.

2.1.5.1 Transmission platforms

In general, the use of communication monitoring networks (wireless in particular) is increasing and this is a trend expected to continue in the future. Communications for environmental monitoring purposes is expected to follow the mainstream trends and make use of the more advanced communication networks and equipment that will inevitably become available over the next ten years. The major feature will be the conflict between bandwidth and coverage. Most modern systems opt for increased bandwidth at the expense of coverage. The most effective wireless communication technologies for environmental data include cellular, radio and satellite. However, the landline could also be used to meet some specific situations because of its stability and possible high speed though its use is quite limited and expensive.

2.1.5.2 Communication Protocol Model

In order to support different connection protocols in the A-ITEMS, the international standard for open systems should be utilized to provide a common interface, say, Open Systems Interconnection Basic Reference Model (OSI Reference Model, for short), which is a layered, abstract description for communications and computer network protocol design to describe the internal behavior of real open systems (ISO, 1994). It divides the functions of a protocol into a series of layers and therefore is also called the OSI seven layer model shown in Figure 2.2. This logical separation of layers makes reasoning about the behavior of protocol stacks much easier, allowing the design of elaborate but highly reliable protocol stacks. Each layer only uses the functions of the lower layer, and only exports functionality to the upper layer. For example, a layer that provides error-free communications across a network provides the path needed by applications above it, while it calls the next lower layer to send and receive packets that make up the contents of the path.

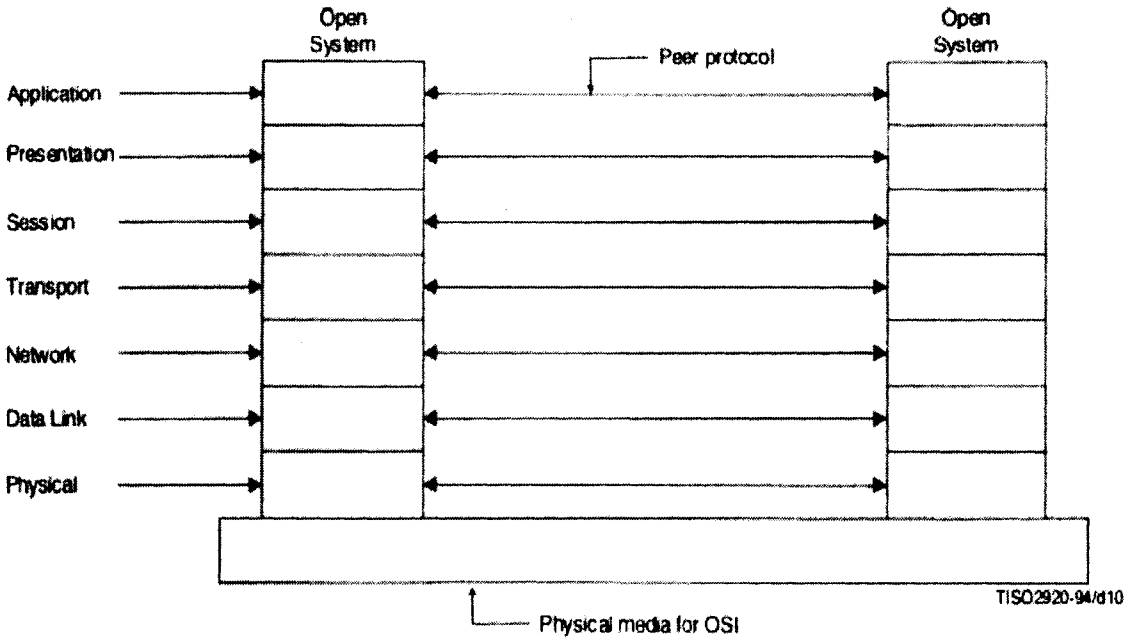

Figure 2.2 OSI reference model (ISO, 1994)

A simplification of the OSI reference model, the four-layer communication reference model (Braden, 1989) (Figure 2.3), is more appropriate. It consists of four layers: application layer, transport layer, network layer and data link layer. The three top layers in the OSI model - the application layer, the presentation layer and the session layer - usually are lumped into application layer in this model.

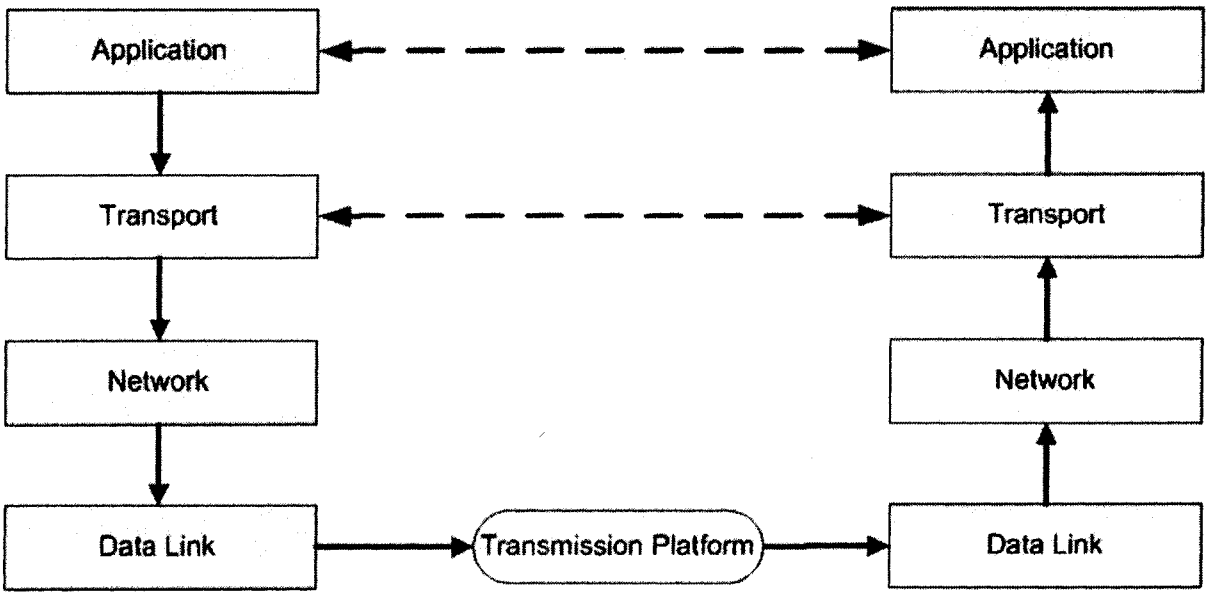

Figure 2.3 Four-layer reference model (modified from Braden, 1989)

The network layer uses encapsulation to provide abstraction of various protocols and services. Generally, a protocol at a higher level uses a protocol at a lower level to help accomplish its aims. The transport layer consists of a set of rules and the protocol, which are used with the network layer, to send data in a form of message units between terminals over the network. The network layer takes care of handling the actual delivery of the data. The transport layer focuses on keeping track of the individual units of data packets that a message is divided into for efficient routing through the network. The data link layer provides the functional and procedural means to transfer data across the physical link between network entities no matter what transmission platform is.

2.1.6 Packet mechanism

Packet mechanism facilitates the transmission of monitoring data from source to user in a standardized and highly automated manner, and implements common transport data structures and protocols, which can enhance the development and operation of the A-ITEMS (CCSDS, 1987). Packet mechanism defines two data structures: source packets and transfer frames, and a multiplexing process to interleave source packets from various application processes into transfer frames. The source packet is a data structure generated by an application process in a way that is responsive to the needs of that process. It can be processed at fixed or variable intervals and may be fixed or variable in length, which usually depends on the application process. It includes a packet header that identifies the source and characteristics of the packet, and the internal data content which is completely under the requirements of the application process (CCSDS, 1987). Compared to the source packet, the transfer frame is a different data structure that provides an akin envelope for transmitting packetised data over a transmission platform. It carries information in the transfer frame primary header that tells data processing and control unit how to route the transfer frames to their intended destination. It can have a secondary header and operational control field which meet other particular needs, for example, the specific data recording the activities of sources. Normally, the transfer frame is of fixed length.

In the A-ITEMS, packet mechanism is rather helpful to feed various monitoring networks which have different monitoring indicators and data protocols. It provides the end-to-end transport for monitoring data sets from source application processes located in monitoring networks to distributed user application processes that the A-ITEMS can support. Meantime, it makes it highly efficient to share data on the transmission level rather than database level, which may be quite useful for some particular monitoring applications with high sampling rates.

2.1.7 Routing

Routing has gained more and more concern on monitoring networks. It is generally used to solve the limitation of communication. For example, the limited bandwidth can cause data jam on the data pathway. The poor coverage can let a field station, which is an in-field work station or a so-called RTU, become an island and result in the failure of data transmission. The routing may be required if the mobility exists in field, e.g. a field station may be mobile. In the environmental monitoring domain, the monitoring infrastructure usually stays stationary. But if the monitoring area is changed or some field stations cannot work normally, the data pathway has to be adjusted. Otherwise some data would be lost.

Depending on the capability of the communication hardware, the EMS may need to determine a new network topology and calculate new routes between stations using some effective algorithms such as Location-Aided Routing (Ko and Vaidya, 2000), A Distance Routing Effect Algorithm for Mobility (Basagni et al, 1998), or Location-Based Multicast Algorithm (Jiang and Camp, 2002). The mobility can not only be processed at some central location in the EMS, but also by stations if stations can share data with each other.

Because some stations may be movable or mobile, and the bandwidth and coverage of specific communications may be limited, the A-ITEMS needs to consider possible routing cases and design some routing schemes. There are two routing schemes, which may be used by the A-ITEMS in terms of specific applications.

2.1.7.1 Self-adaptive routing

Self-adaptive routing includes central self-adaptive routing implemented by DPC and station self-adaptive routing implemented by stations. In DPC, mobility processing component predefines the routing rules and algorithms. Once it gathers required topologic and status information of monitoring networks, it will create a new routing. If the previous routing needs to be updated, the new routing information will be transferred to update the routing parameters in corresponding stations. As presented in Figure 2.4, mobility processing component has routing rules and algorithms, and can create the new routing. It can send the routing information to stations whose routing needs to be changed. In Figure 2.4, the solid arrow line is the previous routing, the dash arrow line means the new routing, and the two-point dash line indicates that mobility processing component transfers the routing to specific stations. The dash circle represents the stations whose routing needs to be adjusted.

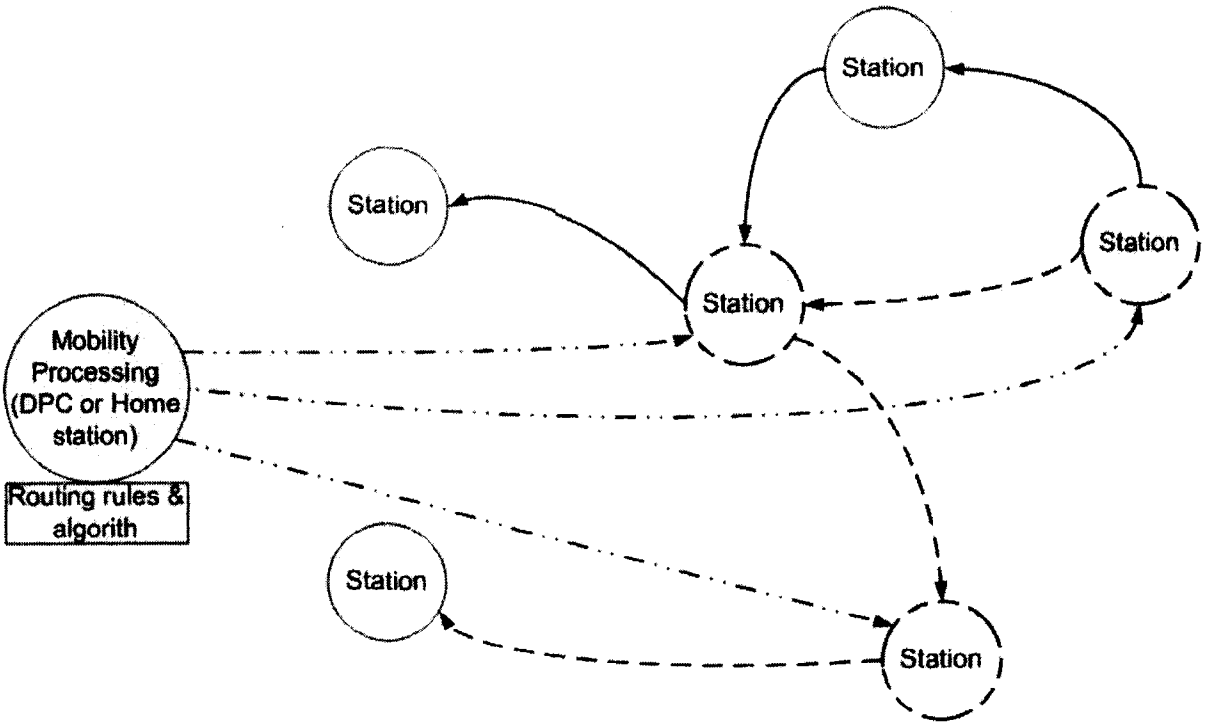

Figure 2.4 Central self-adaptive routing

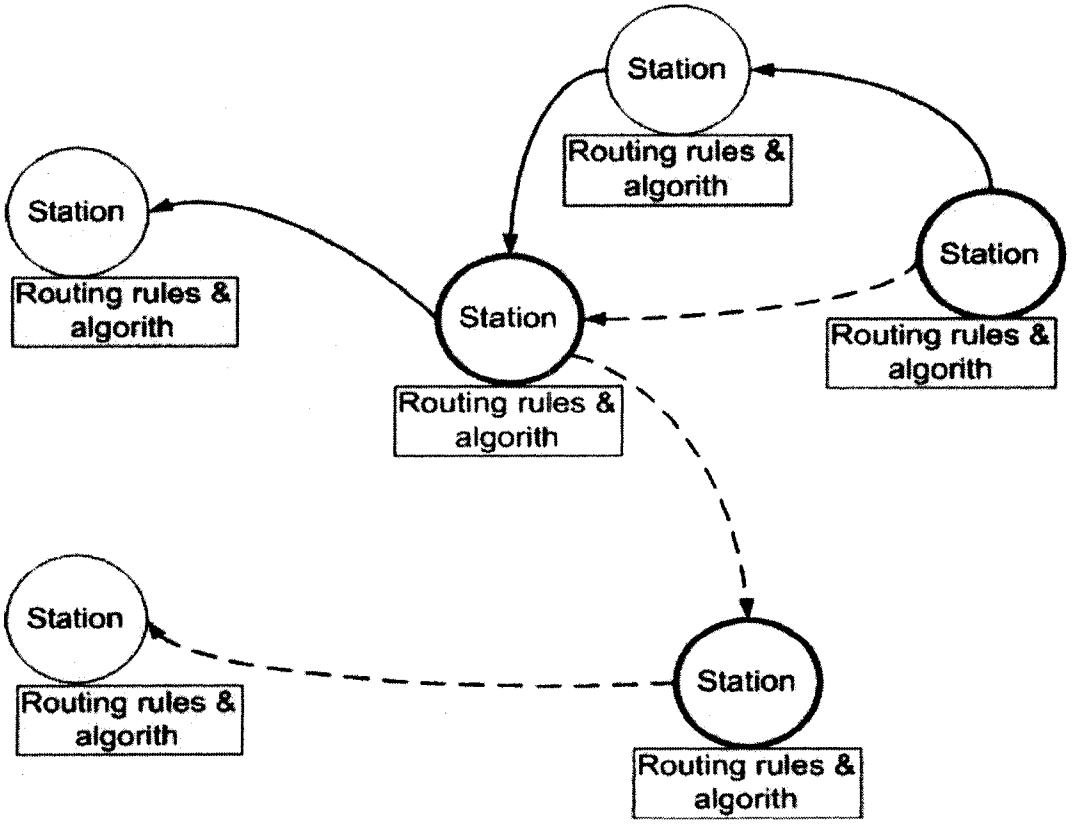

Figure 2.5 Station self-adaptive routing

The basic idea of station self-adaptive routing is that the predefined routing rules and algorithms reside in stations, which can communicate with each other. Once the station gains enough information about its neighbours, it can make one local routing or global routing, which depends on the algorithms. As Figure 2.5 shows, each station owns the routing rules and algorithm. It broadcasts its basic status information and data. Once it has the sense to modify the routing, it would cooperate with other relative stations. In Figure 2.5, the solid line means the previous routing, the dash line is the new routing and the dash circle denotes the stations with the changed routing. In terms of the trigger mechanism, the routing rules may be designed as specific routing triggers. Once the routing needs to be updated, the corresponding routing trigger would happen.

2.1.7.2 Manual routing

Manual routing is performed by users, and means that the new routing is created manually and then used to update the routing configuration in stations. This scheme is quite useful to build the new routing because of the possible change of monitoring requirements. If self-adaptive routing scheme can not work, manual routing is the easy way to adjust the routing, but it may need to develop some specific tools to create the new routing. For example, using GIS can let users know the geospatial distribution of stations. The diagnostic tool can check the validation of the new routing.

2.1.8 Remote configuration

Telemetry, conceptually, provides remote configuration to control the work of devices. In the environmental monitoring domain, a large number of monitoring devices might be distributed around. The in-field configuration consumes much time, cost, staff and tools, which obviously is not an ideal way that people favor. Remote configuration is capable to remotely set up the in-field devices while users work in office and manipulate the devices to detect monitored objects. For example, we may want to change the sampling rate to get more data or adjust monitoring networks to more efficiently transfer data because of some reasons.

2.1.9 Firmware update

In the A-ITEMS, it is one of the main objectives to remotely update the firmware of in-field devices. Usually, two models may be utilized: Dynamic Memory Exchange (DME) and File Transfer Protocol (FTP). DME provides the capability for data exchange via communication between virtual devices in Home Station and real in-field devices. The primary function of DME is to keep the same configuration data between virtual devices and real in-field devices. That means if the virtual device is changed by users, once such changes are confirmed, the change will be transferred through DME to

update the firmware of corresponding in-field devices. Meantime, the firmware of real in-field devices could be modified by some way, e.g. trigger events, the modification is also able to reach the virtual devices in the A-ITEMS through DME.

FTP also may be used to transfer the configuration data from the virtual device in the A-ITEMS to the real in-field device through the monitoring network. If the connection is built between the in-field device and its virtual device in the A-ITEMS, the in-field device would check the firmware version of its virtual device. If the version is outdated, it would upload its new version to the A-ITEMS. However, if the version is new, it would download the new version to update its firmware. Before FTP is adopted, some factors have to be taken into account, e.g. communication capacity, computing capacity of in-field device, etc.

DME is more appropriate for A-ITEMS. It gets the specifications of devices and creates general domain object models. Then the properties of these domain object models can be utilized to generate the representation of the corresponding in-field devices. The A-ITEMS calls such representation the virtual device. It uses such virtual device to synchronize the real in-field devices. Once the virtual firmware is changed, the specific in-field device would be modified afterwards.

2.1.10 Adaptive Graphical User Interface (GUI)

GUI has become a familiar part of the software landscape, both as users and as developers. The basic thing is how to create the adaptive GUI, which is seperated with its content and then can be dynamically updated. Model-View-Control (MVC) is a fundamental architecture shown in Figure 2.6 that separates an application into three distinct components, Model, View, and Controller, so that modifications of one component can be made with minimal impact to the others. Model manages information and notifies observers when the information changes. It contains only data and functionality that are related by a common purpose. View is responsible for mapping graphics onto a device. It attaches to a model and renders its contents to the display surface. Controller processes and responds to events, typically user actions, and invokes changes on the model and perhaps the view.

MVC is often thought of as a software design pattern. In a broad term, constructing an application using a MVC architecture involves defining these three modules. MVC decouples views and models by establishing a subscribe/modify protocol between them. A view must ensure that its appearance reflects tha state of the model. Whenever the model's data change, the model notifies views that depend upon it. Each view can update itself. The good thing is that this approach can give us the change to attach multiple views to a model to provide different presentations. Again, using MVC,

views can be nested. For example, a control panel of buttons might be implemented as a complex view containing nested button views (Gamma et al., 1994).

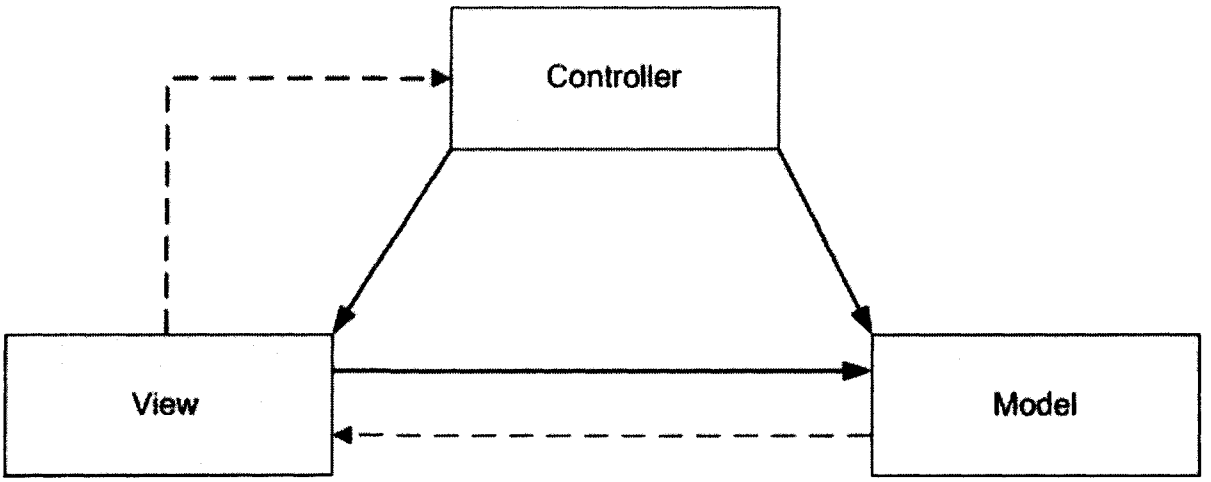

Figure 2.6 Concept of MVC

The A-ITEMS defines multiple domain object models, which describe the structure of monitoring networks. When users design monitoring networks, the A-ITEMS creates the specific concrete domain objects, which inherit corresponding domain object models. These concrete domain objects retrieve data from databases using system function models. Users only need to view these concrete domain objects no matter how they are created. For example, the A-ITEMS can use station object model to create some concrete stations, which may have different properties and methods. Based on the principle of MVC, the A-ITEMS pushes these concrete stations to dynamically update the design view.

2.1.11 Triggering for alarms and adaptive behaviour

Trigger activities happen when some particular condition arises. In the environmental monitoring domain, the trigger scheme is crucial for the monitoring success and can be designed as one subsystem. Once the monitoring indicator reaches the predefined threshold, some kind of trigger would take place and notifies users what happened. For example, if water level of a river rises to specific value, the flood alarm will be triggered. The trigger scheme becomes one of the fundamental features in environmental monitoring systems and quite helpful to avoid the possible disaster by catching alarms. Most of triggers reside in stations or sensors. In the A-ITEMS, the triggers can be conveniently designed to catch some exceptional conditions.

Trigger mechanism is used to solve some exceptional conditions or so-called alarms from measurements. There are two types of trigger scenarios. The general scenario is the alarm activated by single measurement; the other is the alarm from the change trend of measurements. The general scenario usually catches the alarms on the site and transfers alarm messages to DPC through a prior pathway.

One trigger designer may be developed to construct complex behavioral rules based on observations made by stations. For example, if rainfall is detected in the headwaters of a watershed, water level may be read more frequently downstream, and in two hours the downstream water chemistry probes (expensive with respect to power and maintenance) begin collecting observations. An event may be scheduled for one hour later to check the mid-stream water level gauges; if the water level is rising faster than expected, the water chemistry probes can be adjusted to begin collecting sooner than originally ordered. The rules are decomposed into simple trigger components which are sent to the observing stations. When the triggering event occurs, the station simply notifies the DPC, which then uses its advanced computing power to evaluate and resolve all triggers and may send new configuration adjustments to the station. This gives considerable flexibility in a conventional system. For example, the assessment and analysis subsystem may be used to evaluate the rules and determine required outcomes (including asking for more data), and also negotiate the necessary changes in terms of current network organization. Eventually, the EMS may send the configuration changes to the specific stations.

2.2 Structural requirements

2.2.1 Distributed architecture

In order to build a better collaboration, the system may adopt the distributed architecture, which is helpful to divide a complicated environmental monitoring program into units and assuage the overload of central processing unit. One unit could depend upon others or be independent as well. For example, the EMS can be designed by one component application while its operation may be managed by another component application. The fundamental principle of the A-ITEMS is to divide the complicated EMS into multiple primary components and subsystems, each of which takes one specific job and is able to cooperate with each other. The conventional EMS usually tightly couples their businesses with the system, and is difficult to expand and maintain. The A-ITEMS provides a decoupled structure and can flexibly integrate other systems and build particular monitoring systems in terms of business requirements. Because the same system conceptual structure is shared, various systems will be convenient to integrate with each other.

2.2.2 Collaborative data and message management

Data from monitoring networks are transmitted to the monitoring operator via some kind of transmission platform. The monitoring operator processes and stores these data in the data depository managed by data server. Because the alarm events are the crucial factor in the EMS, if data are of alarms, they would reach trigger processing, a special component in the A-ITEMS, which deals with the alarm events. The monitoring operator also has functions to access in-field monitoring networks and is able to control the work of the monitoring devices.

Data server stores and manages the data and information by databases, files, etc. In the EMS, we can separate data into multiple types, such as monitoring network information, system configure information, measurement data, and so on. The design depends on the application. For example, it may use a distributed structure and deal with concurrent transactions. However, it should provide the unified interface for other subsystems to retrieve data.

Aside from alarms and status changes, data messages are the most common. Assess/analysis subsystem processes data and outputs some valuable information. It can use some specific techniques, which depend on applications. For example, GIS is able to present the geospatial relationship among monitored objects. Reporting subsystem not only publishes the information to users, but also provides the channel to share data and information with other systems.

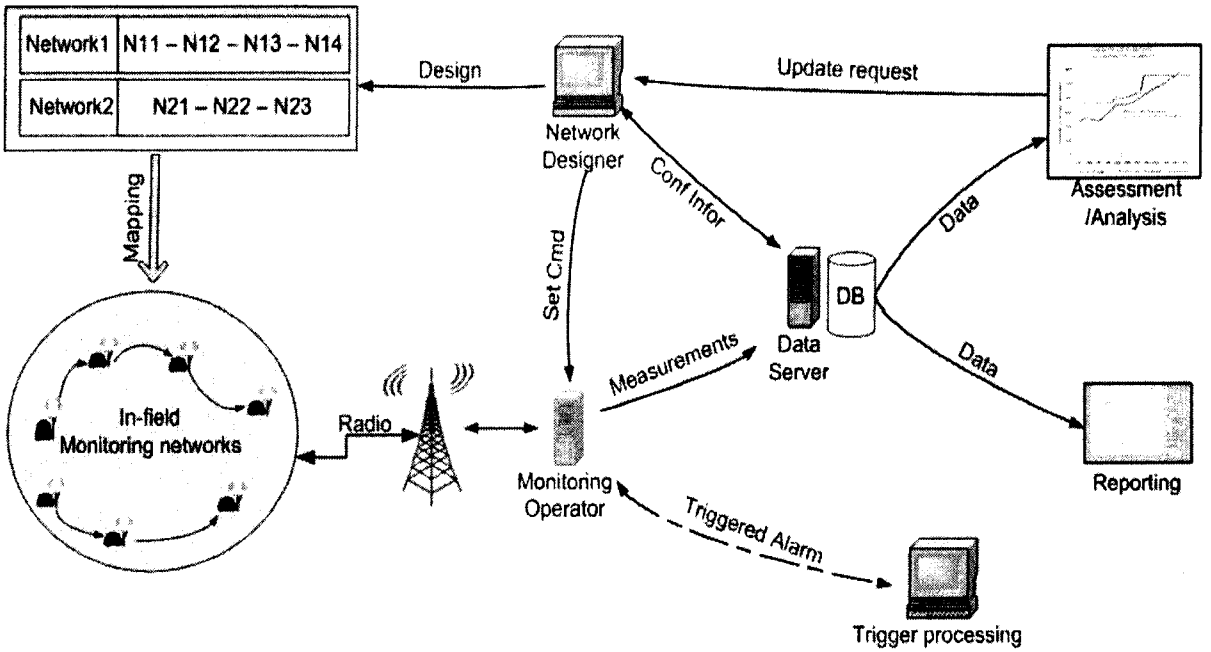

Figure 2.7 System concept

Chapter 3 A-ITEMS Architectural Framework

3.1 Conceptual Approaches

3.1.1 General system architecture model

In general, a computer-based system usually can be modeled as an abstract core using an input-processing-output architecture as shown in Figure 3.1. Around this core, other additional system features in terms of specific requirements can be developed, such as GUI, maintenance, etc. The subsystems and the information flow among them can be specified for subsequent engineering work. The architecture diagram is usually required to definitely present each subsystem and its information flow.

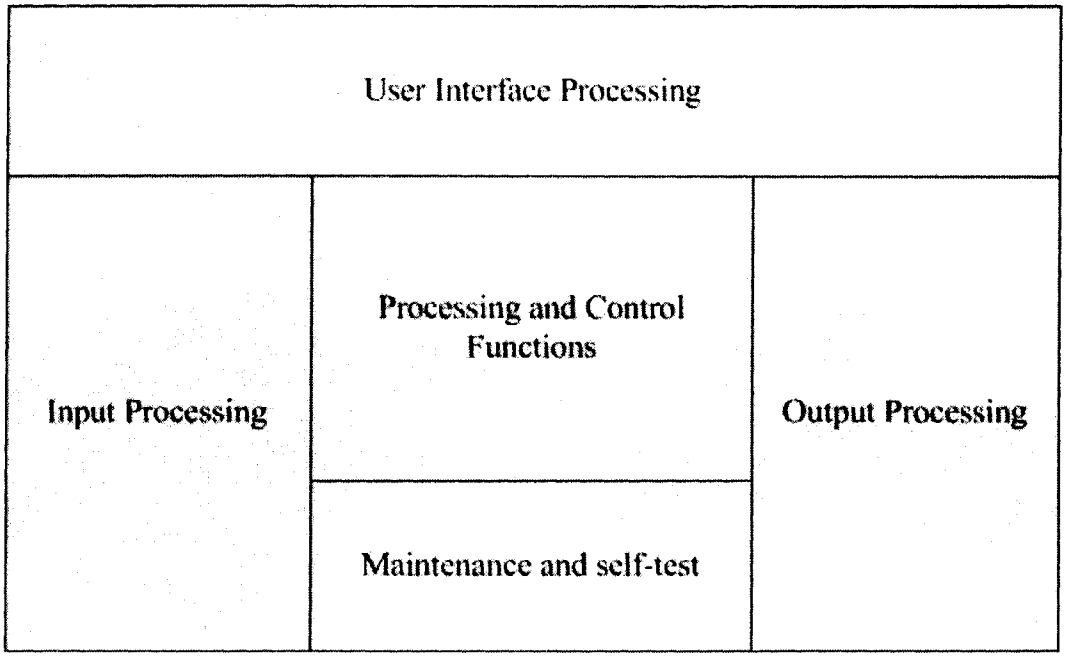

Figure 3.1 General system architecture model (Pressman, 1992).

3.1.2 Integration technology

In an EMS, the integration technology may be used to construct the whole system or accept multiple data sources. In general, there are two kinds of categories of integration: system integration and data integration.

A system is an aggregation of cooperating subsystems so that it is able to deliver the over-arching functionality. **System integration** is regarded as one strategy to bring together of the component

subsystems into one system and ensure that the subsystems function together as a system. In information technology, system integration is the process of linking together different computing systems and software applications physically or functionally[1]. The system integrator brings together discrete systems utilizing a variety of techniques such as computer networking, enterprise application integration, business process management or manual programming. The subsystems have interfaces. Integration involves joining the subsystems together by "gluing" their interfaces together.

Data integration is the process of combining data residing at different sources and providing the user with a unified view of these data (Lenzerini, 2002). With the increase on the volume and need to share existing data, data integration has taken an important role in data/information-based systems.

Some of the current work in data integration research concerns the Semantic Integration problem (Ziegler et al, 2004). This problem is not about how to structure the architecture of the integration, but how to resolve semantic conflicts between heterogeneous data sources. A common strategy for the resolution of such problems is the use of ontologies which explicitly define schema terms and thus help to resolve semantic conflicts. This approach is also called ontology based data integration.

A recent trend in data integration has been to loosen the coupling between data. Here the idea is to provide a uniform query interface over a mediated schema shown in Figure 3.2. This query is then transformed into specialized queries over the original databases. This process can also be called as view based query answering because people can consider each of the data sources to be a view over the (nonexistent) mediated schema. Formally such an approach is called Local As View (LAV) — where "Local" refers to the local sources/databases. An alternate model of integration is one where the mediated schema is designed to be a view over the sources. This approach called Global As View (GAV) — where "Global" refers to the global (mediated) schema — is often used due to the simplicity involved in answering queries issued over the mediated schema. However, the obvious drawback is the need to rewrite the view for mediated schema whenever a new source is to be integrated and/or an existing source changes its schema.

3.1.3 Object-oriented design

Objects are autonomous entities within a software system that are composed of both the data that describe the state of the object and the methods that define valid operations on the state of

---

[1] http://www2.cis.gsu.edu/cis/program/ Last Accessed December, 2007

the object (Booch, 1993). Objects are the fundamental elements in object-oriented design, and basically serve to unify the ideas of algorithm and data abstraction. In general, there are four major elements to object-oriented design: abstraction, encapsulation, modularity, and hierarchy, which are introduced briefly below.

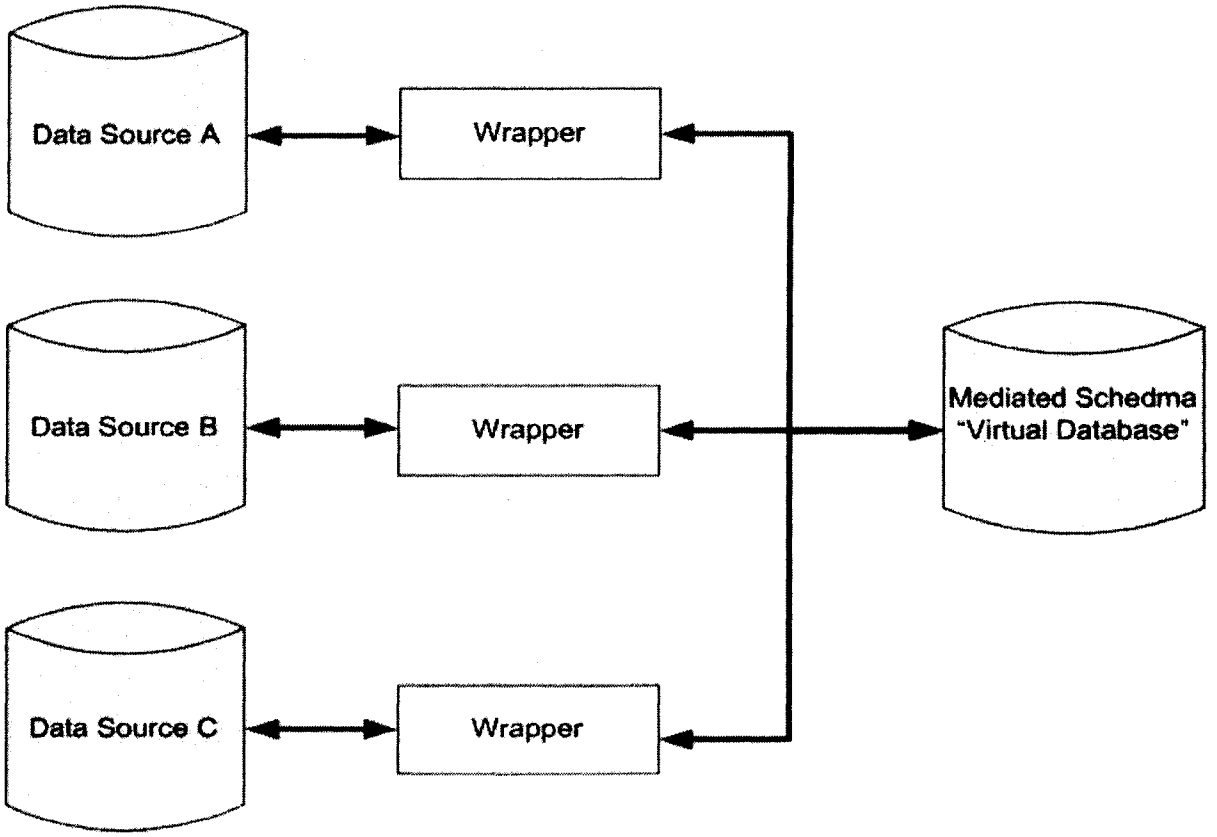

Figure 3.2 Simple schematic for a data integration solution (Lenzerini, 2002)

Abstraction is the concept of identifying the essential characteristics of an object that distinguish it from all other kinds of objects, and thus provide crisply defined conceptual elements of a system (Booch, 1993). Abstraction focuses on the outside view of an object, and so serves to separate an object's essential behavior from its implementation. Commonly, deciding upon the right set of abstractions for a given domain is the central problem in object-oriented design.

An abstraction of an object is also called an interface to provide essential behaviors or member functions, each of which defines preconditions (invariants assumed by the behavior) and postconditions (invariant satisfied by the behavior). The entire behaviors constitute the protocol of an

abstraction, which provides the way for an object to act and react the outside view. For example, a Modem object is expected to take a data message and send it to a remote target. The exact way in which the Modem object structures the message and any other necessary parameters into a byte stream and sends it to the associated physical device is not important at the abstract level; all that is important is that the Modem object is able to carry out this responsibility, and that other objects can rely on the Modem object to fulfill this job.

Encapsulation is the concept of hiding the specific details of an abstraction that make up its structure and behavior, which separates the interface of an abstraction and its implementation (Booch, 1993). Continuing the Modem example, the implementation for each specific type of Modem will do the encoding and transmission in a way that is appropriate for the specific device, and the requestor does not have to be aware of how it is done.

In all, abstraction focuses upon the observable behavior and responsibilities of an object, whereas encapsulation focuses on the implementation that gives rise to this behavior.

Modularity is the concept that a system can be decomposed into a set of cohesive and loosely coupled modules (Booch, 1993). Modular design means trying to subdivide an assembly into smaller parts (modules) that can be easily interchanged. For example, one Modem implementation that represents one manufacturer's device can be replaced with a different Modem implementation that represents another manufacturer's device. Although they may encode and transmit messages in completely different ways, and the system behaviour changes because of the switch, the rest of the system does not have to be aware of the switch because the expected abstract responsibilities are still carried out by the new hardware.

In general, modularity is characterized by: i) Functional partitioning into discrete scalable, reusable modules consisting of isolated, self-contained functional elements; ii) Rigorous use of well defined modular interfaces, including object-oriented descriptions of module functionality; and iii) Ease of change to achieve technology transparency and, to the extent possible, make use of industry standards for key interfaces.

Hierarchy is a ranking or ordering of abstractions. There is more to object-oriented design than simply encapsulating in an object some data and the procedures for manipulating those data. Object-oriented methods also deal with the classification of objects and address the relationships between different classes of objects.

The primary facility for expressing relationships between classes of objects is derivation. That is, new classes can be derived from existing classes. What makes derivation so useful is the notion of inheritance. Derived classes inherit the characteristics of the classes from which they are derived. In addition, inherited functionality can be overridden and additional functionality can be defined in a derived class. An advantage of inheritance is that modules with sufficiently similar interfaces can share a lot of code, reducing the complexity of the program. Inheritance therefore has another view, a dual, called polymorphism, which is the ability of objects belonging to different data types to respond to method calls of methods of the same name, each one according to an appropriate type-specific behavior (Meyer, 1997).

Polymorphism allows client programs to be written based only on the abstract interfaces of the objects which will be manipulated. This means that future extension in the form of new types of objects is easy, if the new objects conform to the original interface. In particular, with object-oriented polymorphism, the original client program does not even need to be recompiled in order to make use of new types exhibiting new (but interface-conformant) behaviour.

3.2 Architectural Structure

The A-ITEMS is modeled to consist of independent modules or subsystems, each of which has specific interfaces to cooperate with other modules. Object-oriented design is used to represent system domain objects and functional objects, which form the fundamental units in the A-ITEMS.

The A-ITEMS adopts an open design mechanism to support broad system integration. It divides the system into multiple components, such as database management, monitoring network design, operation controller, etc. (Figure 3.3) and is organized with reference to the general system architecture model (Figure 3.1). Each component is developed as one separate software module maintains its own internal operations, and has particular interfaces to communicate with other modules. For example, Database Management Module provides the basic functions for other modules to access. It does not care about whatever kind of connection approach users may utilize, e.g. ODBC, MAADO, RDO, etc. The connection process resides in the module. Users can freely change the connection approach, but would not affect other modules at all. Once data enter Database Management Module, they will be processed and stored by this module. For instance, if Assessment/Analysis module needs some data, it will send the request to Database Management Module, and then wait until it gets data. It does not need to know how Database Management Module works. Such mechanism can be used to help users to conveniently maintain and reorganize one

concrete EMS. It is used by each module in the A-ITEMS and can let the A-ITEMS integrate other incoming modules.

Since the components independently carry out distinct duties within the monitoring system and their only interaction is through 'broker' components, they can be distributed across one or more computers. For example the configuration controller can run on several network administrator's desktop computers, the operation controller can run on a server in a managed server facility, and they can all access the common network description stored on a remote database management system running on yet another server. This improves the scalable flexibility of A-ITEMS to support small networks of a few stations through to enterprise-style, multi-user, multi-network systems.

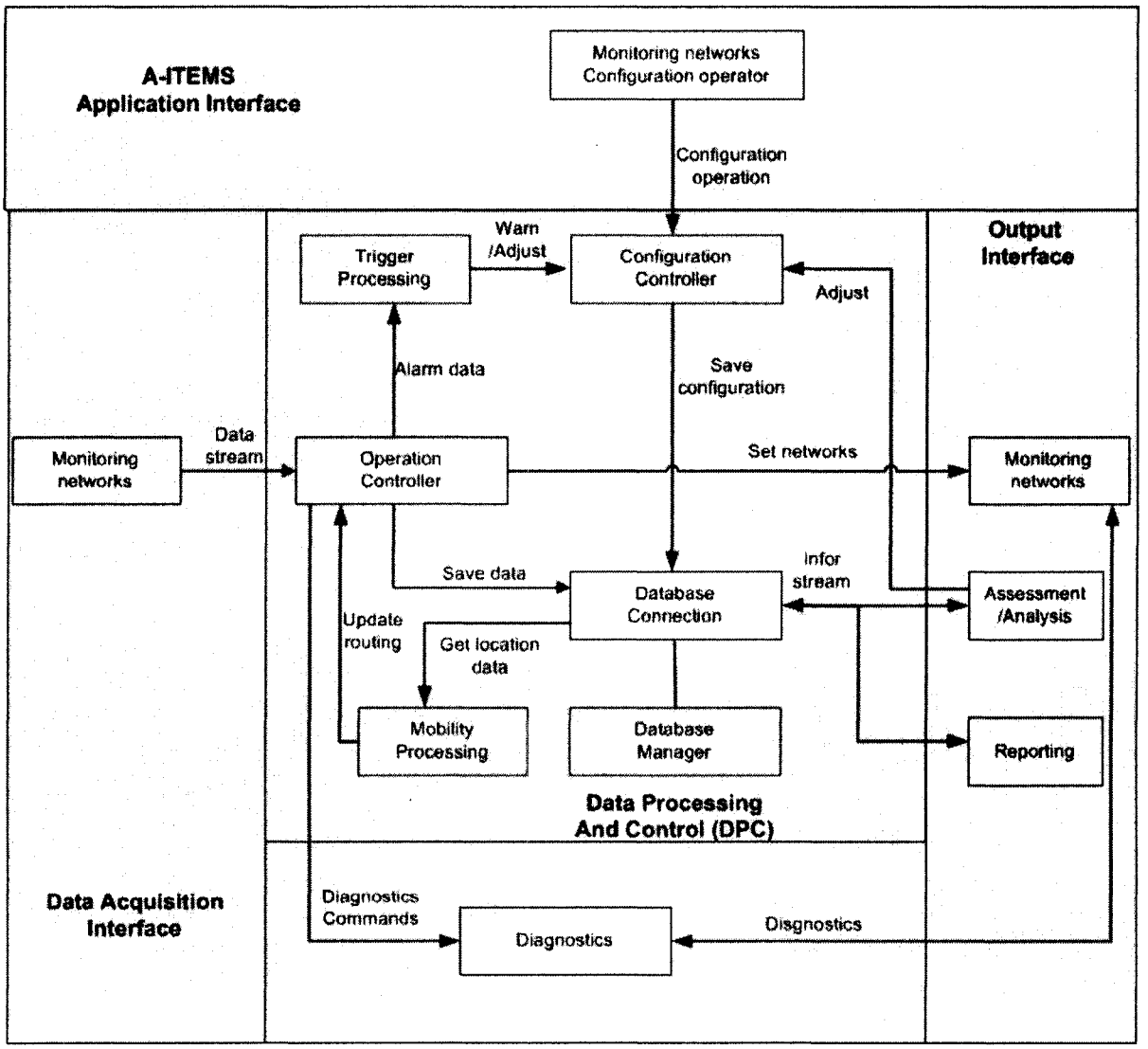

Figure 3.3 System modeling architecture of the A-ITEMS. Compare to Figure 3.1.

Data acquisition interface takes the job to build monitoring networks and create data stream for data processing and control. In fact, these monitoring networks adopt the virtual mechanism and map real in-field monitoring networks. With such scheme, the A-ITEMS can simulate and test real monitoring networks before they are deployed in field. Once the simulated monitoring network can work, the corresponding real monitoring network should work as well.

Data processing and control (DPC) is not only the engine of the A-ITEMS, but also builds the platform to integrate other subsystems. As long as one monitoring project begins, it will be executed on the background. Furthermore, if the A-ITEMS is deployed by one distributed mode, DPC may work in another computer. It includes five subsystems: Operation controller, configuration controller, database management, trigger processing and mobility processing. Operation controller sends, receives and parse messages through port domain objects and provides the only interface talking to monitoring networks. Configuration controller processes design operations of monitoring networks, and parses the monitoring commands. For example, if a new station is created, configuration controller would provide one station template in terms of the type of station. When the properties of this station are set up, configuration controller would check their validation and deposit them in the corresponding databases. If one project is activated, configuration controller would have necessary information to set up monitoring networks. Database management is supported by data server which consists of database connection and database manager, and provides data depositories to store the system configuration, specifications of monitoring networks and measurements. It is the data and information centre where other systems can access and query data. It consists of database connection which provides the unified interface to connect any kind of database, and database manager which deposits and maintains data. In order to conveniently process complicated and crucial alarms, trigger processing is designed as one individual subsystem. Mobility processing could be one subsystem or thread. It monitors the topological changes of stations in monitoring networks, which could be mobile or movable, and forms the new topology in terms of specific algorithms if necessary. Once a new topology is built, mobility processing notifies operation controller to update the routing structure in monitoring networks.

The **monitoring network designer** describes and manages the network's components and behaviours, and is implemented by **configuration controller**. The structure is highly modular and extensible; all domain descriptions are stored in an internal database. Based on the particular protocol for each device, appropriate commands can be generated to set up the device. Then the A-ITEMS can control behaviours of monitoring networks by adjusting the firmware of devices. Furthermore, the monitoring network may be transformed through a formatter plug-in and published as a set of SensorML

documents, for example. If the A-ITEMS is integrated into a larger infrastructure, this capability can provide initial configuration information to one so-called virtual monitoring network (VSN).

Diagnostics is necessary in the A-ITEMS. Though the design of monitoring networks is accomplished, the question is how to assure that these monitoring networks can work correctly. Diagnostics defines specific diagnostic rules and provides particular tools to examine the design of the A-ITEMS. For example, it can try to access a specific station and retrieve the state of one particular attribute. It also can set up one special sampling rate to test the work of operation controller.

The output interface is quite flexible and usually depends upon specific requirements. In order to clarify data flow of the A-ITEMS, monitoring networks are put here to present that they can receive requests from operation controller and update the firmware of monitoring devices. Other two primary subsystems are listed: Assess/analysis and information reporting. GIS platforms may be used to analyze and report information because measurements have the basic geospatial information.

3.3 Domain object model

In general, domain objects represent a problem domain's logical entities. Herein, they are adopted to model the real-world entities and perform specific operations on data attributes and also provide data for presentation. One domain object is highly abstracted to describe the behaviors and states of the real entities and encapsulate persistent data. In the A-ITEMS, domain object model is to represent the primary entities in monitoring networks, such as sensor, station, modem, port, etc.

The monitoring network consists of multiple modeled domain objects shown in Figure 3.4, which map the real in-field devices and their networking relationships. In order to clearly present the monitoring network, the base station is taken out of field stations because only it directly connects to the DPC in each monitoring network. The base station is quite different with field stations, and may have two modems. One connects to other field stations, and the other directly connects to DPC.

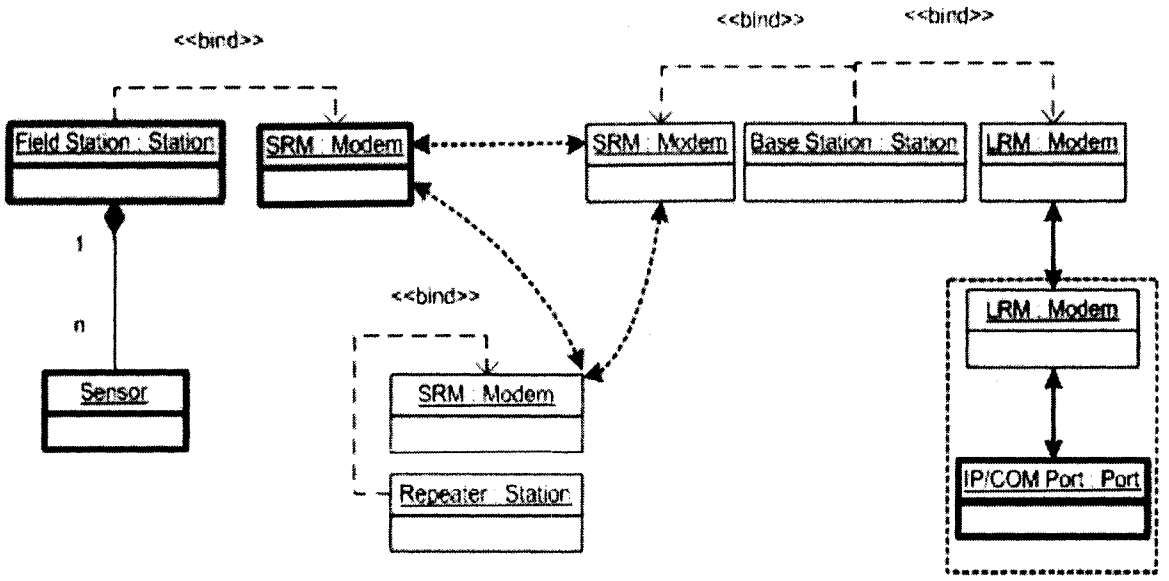

Figure 3.4 Monitoring network

3.3.1 Sensor object model

The sensor is in charge of making environmental measurements. It is usually attached to one station and sends measurements to the station. One sensor has some specific properties and methods presented in Figure 3.5. In order to uniquely identify one sensor in monitoring networks, each sensor must have one ID, which may be defined by some particular rules. Sampling rate is one of critical properties for one sensor and represents how often measurements are created. It normally depends upon specific applications. For example, in order to catch a moving behaviour, sampling rate may be set as a high value. In the environmental monitoring domain, it is very important and may influence the performance of an EMS. In general, measurements need be stored in its definite repository.

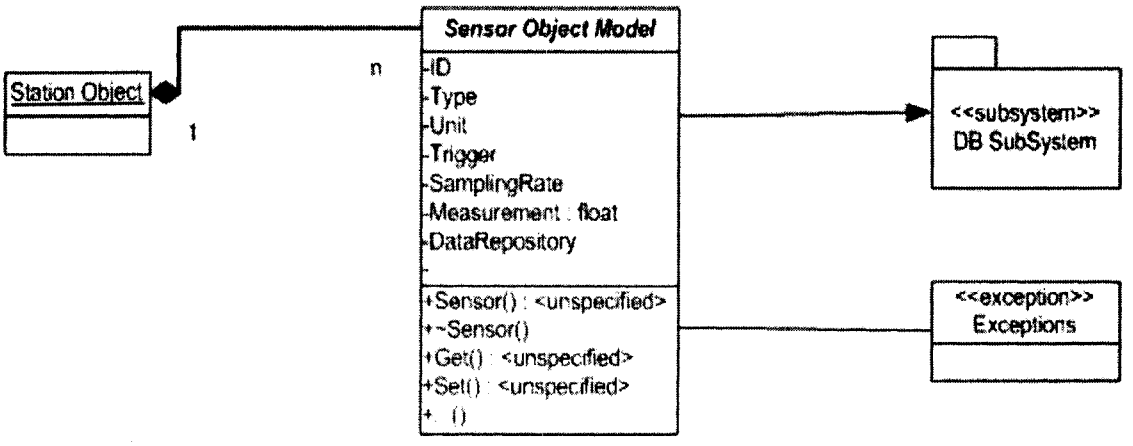

Figure 3.5 Sensor object model

Trigger is another important property in the sensor and may be quite complicated. It defines one or more special conditions. Once the sensor reaches one condition, trigger would be fired. For example, we can set one trigger as one threshold, maximum value. As long as one measurement is more than this threshold, the trigger will happen. And then one warning message would be shown.

3.3.2 Schedule object model

Time-based or scheduled activities happen at a particular time. They are placed on a schedule as schedule items and are invoked at the specific time according to the system reference clock. There are two types of scheduled activities. One is one-time scheduled activity, which happens at a specific time, e.g. 4:15pm on Friday, November 3, 2006. Once the activity happens, the schedule item is removed from the schedule. The other is interval-based scheduled activity happening regularly according to some regular cycle, e.g. 5 times per second, every minute, every 10 minutes, every hour, and every second day. It can be scheduled to begin either: a) at a specific time; or b) immediately. And it also can be scheduled to end on the last activation, according to the scheduled interval, which occurs at or before a specific time, or never end.

In terms of the activities of schedule object, one schedule object model is presented in Figure 3.6, where one station or modem object may own a schedule object. The property of ScheduleQueue defines one container, which keeps all schedule items with the same schedule structure. The schedule object can create a new schedule item, or remove a schedule item once it has happened. The method, GetNextSchedule, acquires the coming schedule activity.

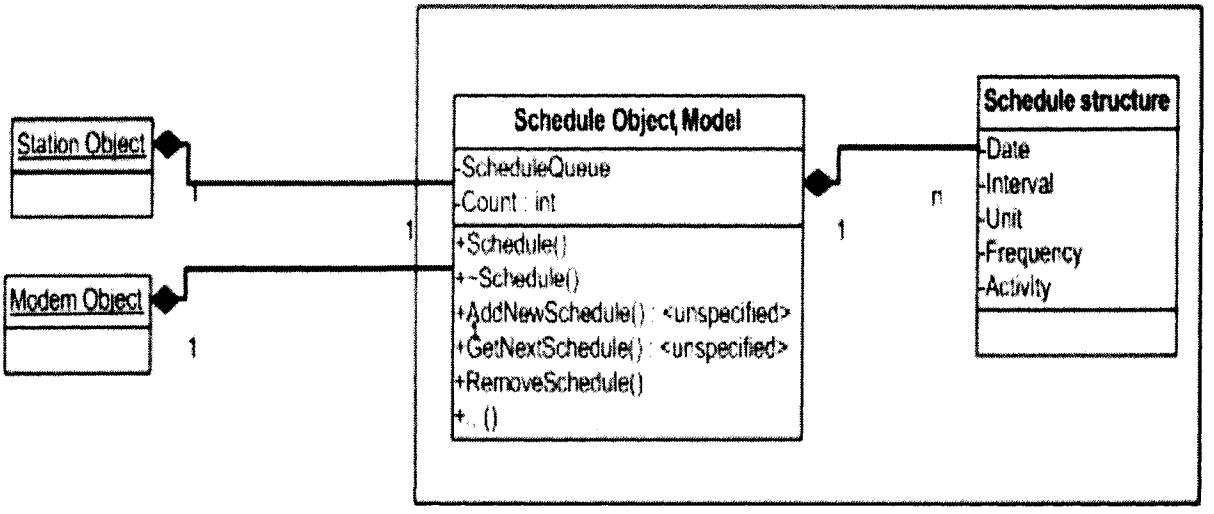

Figure 3.6 Schedule object model

3.3.3 Modem object model

The modem is used to communicate among stations. It has the specific firmware which can be configured via its internal AT commands. In the general situation, the A-ITEMS uses one home modem connecting to multiple field modems, each of which is attached by one field station. One message must have its destination, which could be multiple, and source. There are two ways to send a message here. One way is that the home modem broadcasts messages, and all field modems can receive them. The stations will determine whether messages should be processed or ignored. The other way is the home modem need change its destination before one message is transmitted. Therefore, the message will reach its specific field modem rather than others. Based on the work of the modem, we can simply solve the routing problem as well.

Modem object model shown in Figure 3.7 has one serial number, which is the unique identification in the A-ITEMS. Source address and destination address can support the routing. Baudrate determines how quick the message is transmitted. One modem can be idle, sleeping, or even shut down in terms of activity schedules.

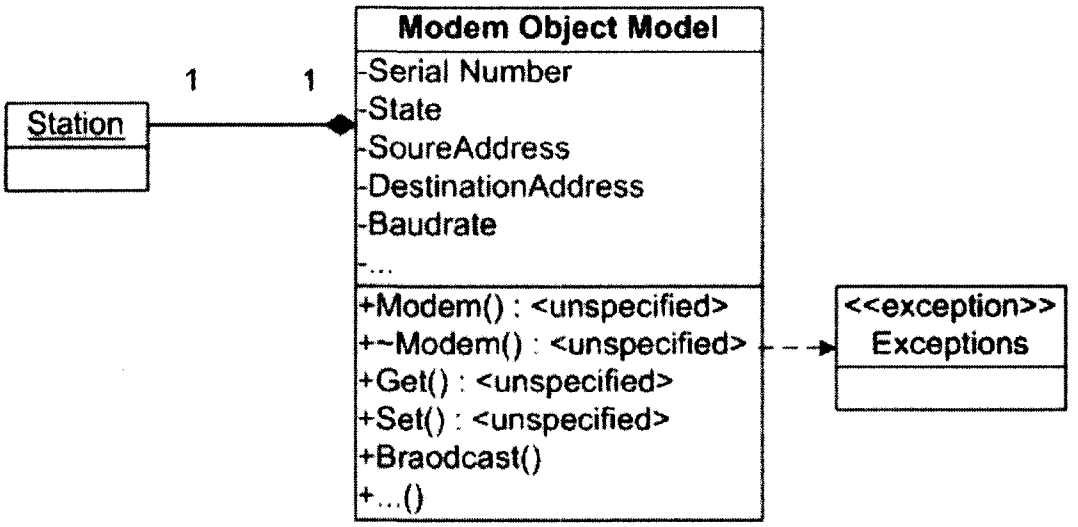

Figure 3.7 Modem object model

3.3.4 Station object model

The station is of a super data acquisition unit in monitoring networks, and sometimes also called RTU (Remote Telemetry Unit). The station generally consists of one controller and one modem. The controller contains the firmware including one memory map and some powerful capacities on computing and storage. In terms of the role of one station in the A-ITEMS, it can be categorized into two fundamental types: base station and field station. Base station is the connection point between

monitoring networks and DPC, and also able to attach some sensors through its IO board while field stations are the primary data acquisition units in monitoring networks.

Figure 3.8 presents one station object model. The model lists some primary properties and methods, and presents the relationships between this model and other models. Some particular exceptions could happen. The serial number provides the unique identification of one station in monitoring networks. Registered property indicates that the station has already been configured in the A-ITEMS, and then is able to be accessed. Otherwise, it would be free and can not be utilized yet. The properties of Memory map and protocol usually cooperate with each other to present the structure and operation commands of the station's firmware. The activities of a station can be controlled by specific schedules managed by the schedule object. In some cases, the schedule is necessary and important because of the limits of stations' power and requirements of an application. Based on the schedule object, the station can be activated or deactivated. One station may be attached by multiple sensors and one or two modems, which is implemented by *attach* and *bind* methods.

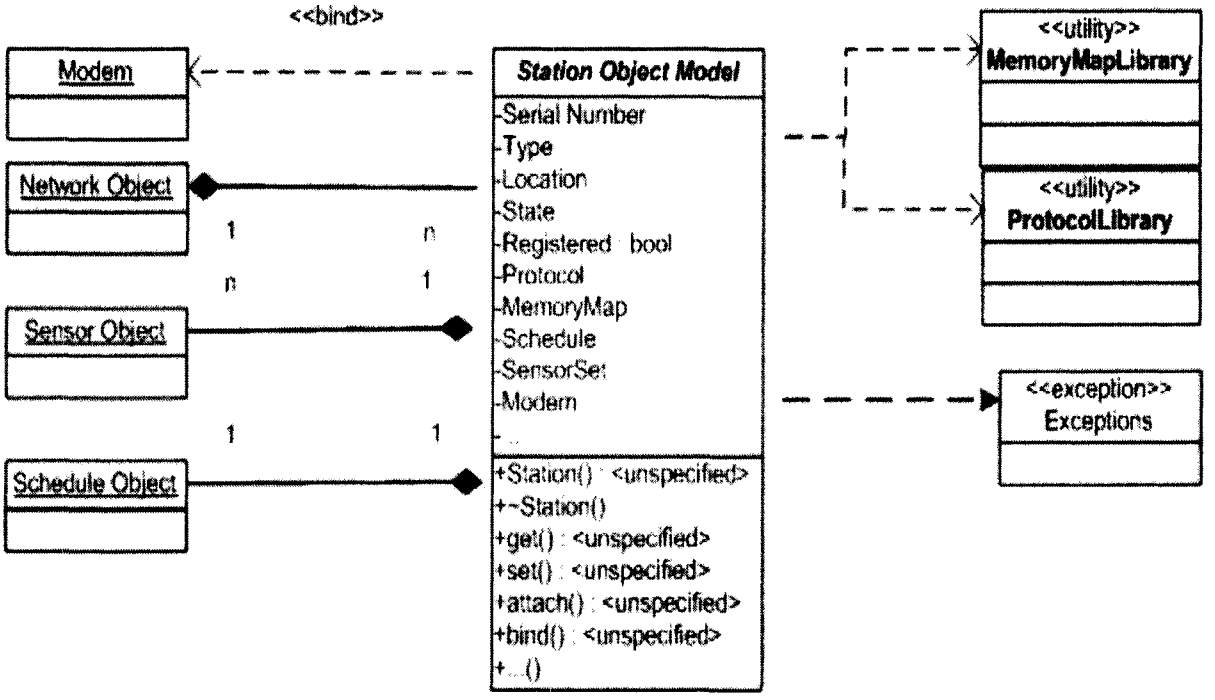

Figure 3.8 Station object model

3.3.5 Port object model

Port object model is designed as one unified interface to solve different communication paradigms. Generally, there are two kinds of communication paradigms: packet switching and circuit switching.

In packet switching paradigm, packets, basic units of information carriage, are routed between nodes over data links shared with other traffic. This paradigm is adopted to optimize the use of the channel capacity available in a network, to minimize the transmission latency and to increase robustness of communication. The well-known typical use of packet switching is the internet, which uses the internet protocol suite over a variety of data link layer protocols and local area networks. In the A-ITEMS, packet switching paradigm can be utilized to deal with the long-distance communication. Each TCP connection between DPC and monitoring networks is based upon specific IP addresses, and has an associated 16-bit unsigned port number (1 to 65535) reserved by the sending or receiving application, but some of TCP ports are predefined or registered. Circuit switching paradigm establishes a dedicated circuit or channel between nodes and terminals before users can communicate. Each dedicated circuit cannot be used by other callers until the circuit is released and a new connection is set up. However packet switching paradigm does not require a circuit to be established and allows many pairs of nodes to communicate almost simultaneously over the same channel. Each packet is individually addressed precluding the need for a dedicated path to help the packet find its way to its destination.

In terms of the characteristics of these two communication paradigms, we can abstract them as one common object model shown in Figure 3.9, where both COM port object model and TCP port object model inherit from port object model.

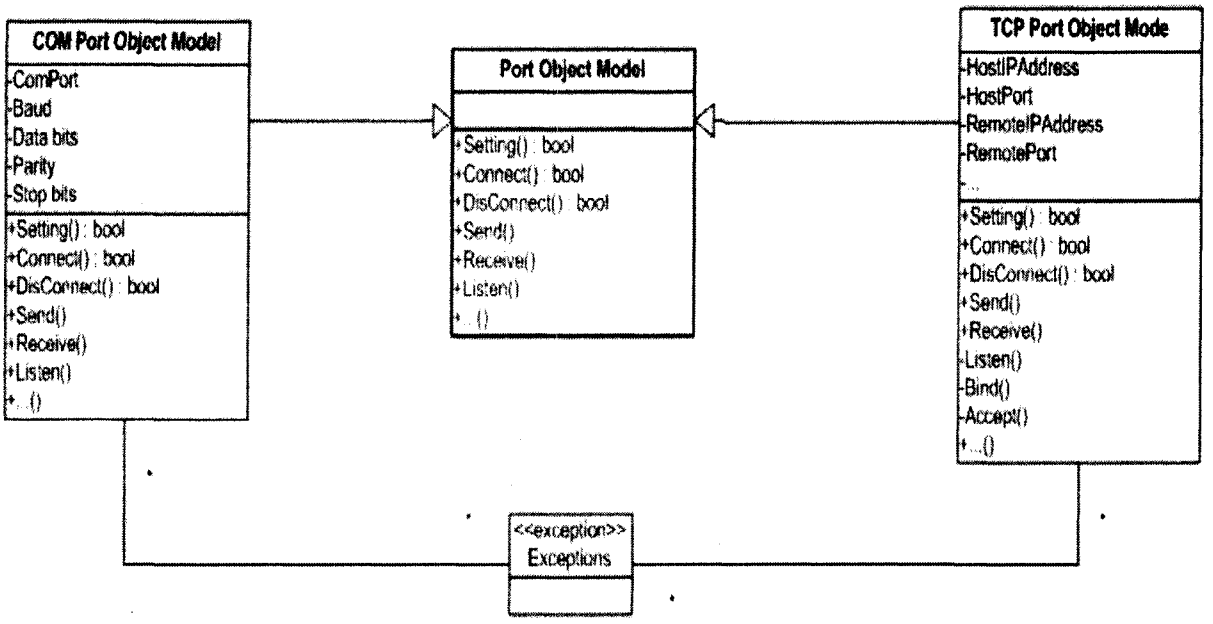

Figure 3.9 Port object model

3.4 Functional object model

3.4.1 Database connection model

Applications that make use of databases often need to frequently obtain connections to the database. Opening and maintaining a database connection for each request is costly and wastes resources. For example, a popular website that is serving information from a back-end database may need to obtain a database connection for each client who is requesting a page with their browser.

In the A-ITEMS, some components may try to access the same database. To ensure the system is capable of responding to each request fast enough, we need to profile the time spent on performing each task. Generally speaking, one of the most expensive tasks involving accessing databases is the initial creation of the connection. Once the connection has been made, the transaction usually takes place very quickly. The connection pooling technique can be used to improve the system performance by retaining a pool of already-opened connections, so the system can simply grab one as necessary, use it, and then hand it back, without the long wait for the initial creation of the connection. Based on this idea, the dynamic database connection model is presented in Figure 3.10.

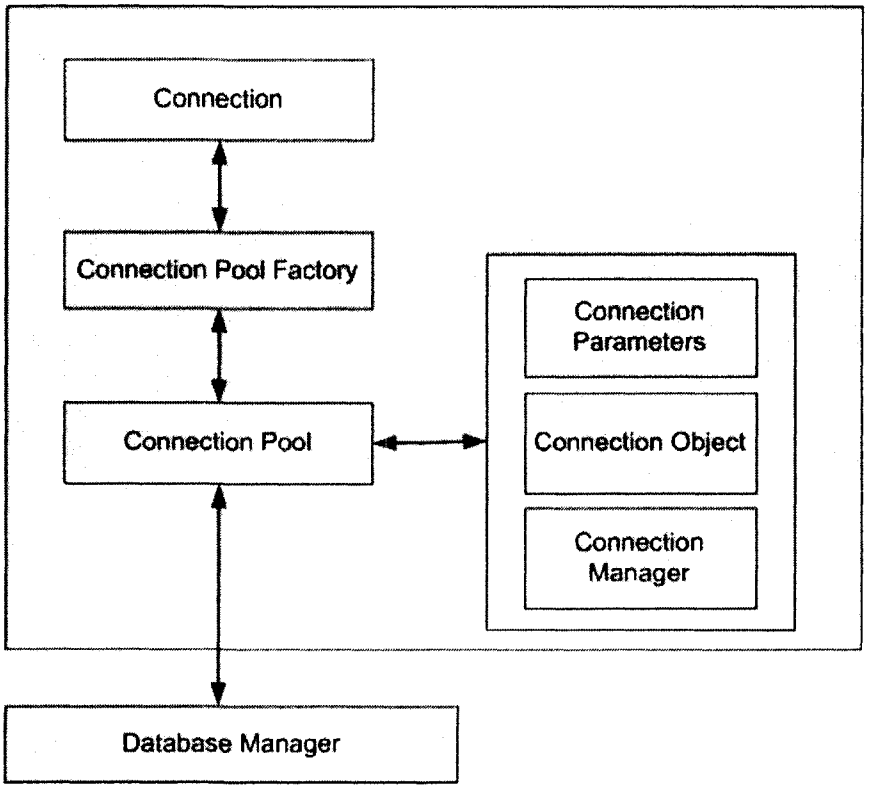

Figure 3.10 Dynamic database connection model (modified from Falkner et al., 2002)

Once a connection is requested, connection pool factory will check the connection pools. If the connection does exist, one connection object will be returned. Otherwise, connection pool factory creates a new connection pool or chooses one connection pool to hold the new connection, which depends on various parameters such as number of minimum connections and maximum connections. Connection parameters contain the necessary properties about specific connection approaches, e.g. ODBC, ADO, etc. In order to pass through possible check-in, user name and password may be provided by connection parameters for each connection request.

3.4.2 Database management model

The A-ITEMS can provide not only local data sources, but also remote data sources. The data sharing is one of the system characteristics. Meantime, it also needs to flexibly feed some new data sources by some way, such as registration mechanism.

Database management model is designed as three layers shown in Figure 3.11: application layer, middleware layer and data layer. The bold dashed line represents the division of different layers. Application layer creates requests to retrieve data or register new data sources. Middleware layer parses requests and forms corresponding SQL commands. But it does not know where data reside, either locally or remotely, so it needs to search them. As to this point of view, there are many strategies to implement searching task. For example, we can create the metadata, which describes databases as well as data. If data cannot be found locally, a request would be sent to remote data sources. If no data source owns such data, application layer would receive one fail message. Of course, database management model may have other jobs. For example, maintaining the consistency of data, harmonizing the concurrent operation of multiple users, and checking the validation of coming data, etc. Herein, these particular techniques are out of our research of this paper, so we will not discuss them. The lowest is data layer, which includes the entire data and its description files, e.g. metadata, data directory, etc.

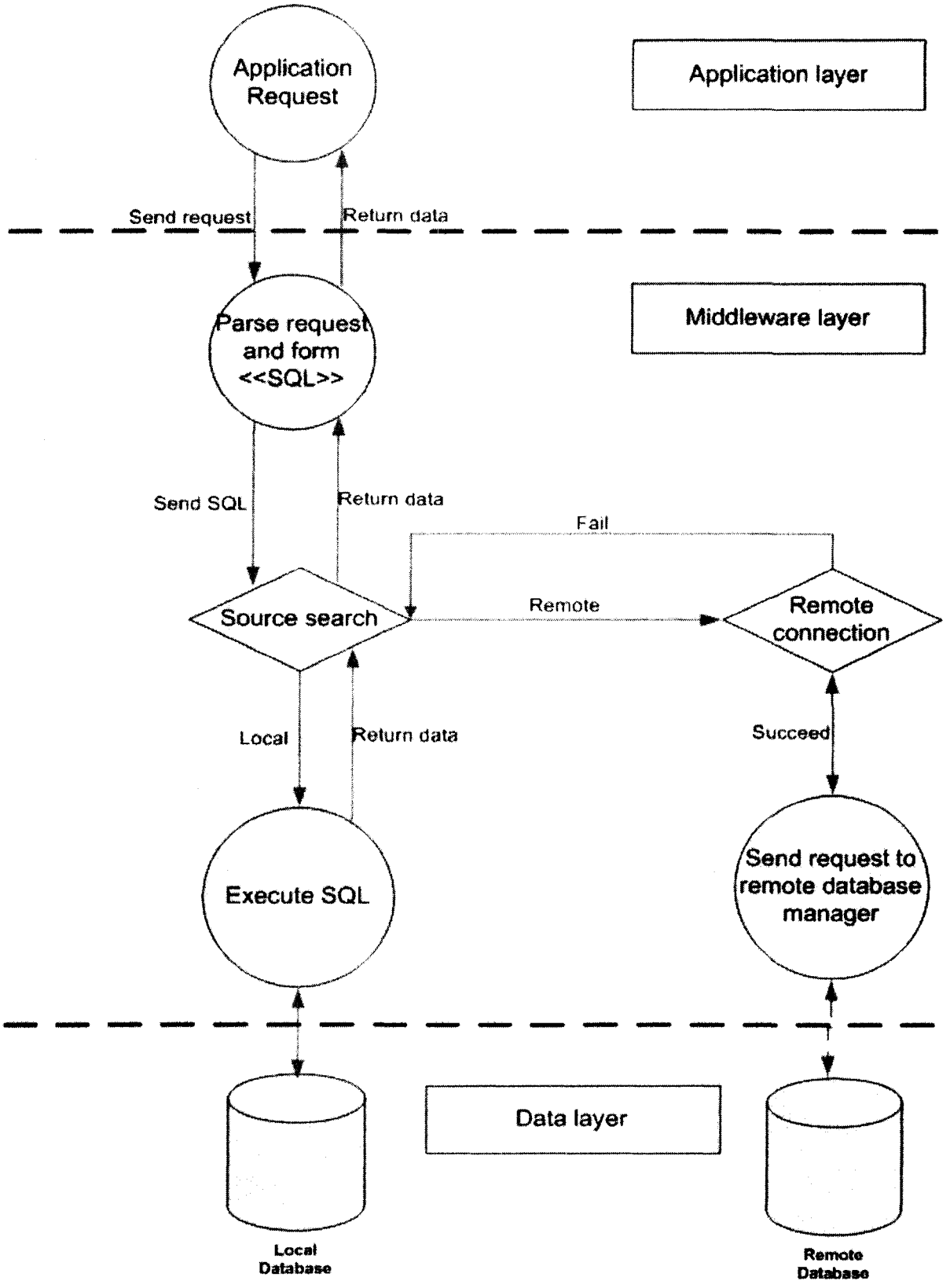

Figure 3.11 Database management model

3.4.3 Communication model

Communication component is one of the key features in the A-ITEMS, and provides the pathway to transfer data between DPC and monitoring networks. DPC needs support multiple connections, and each connection taking either packet switching or circuit switching is built through one specific port as presented in Figure 3.12. In order to manage connections, one connection pool is designed and includes connection parameters, connection object and connection manager. If one connection belongs to circuit switching, it will be in charge of asking the corresponding port object to release the port once the transmission is done. Any outgoing transmission will request the specific connection from connection pool, and meantime any incoming transmission will apply for a new connection from connection pool. In order to deal with concurrent transmissions, multithreading strategy can be adopted. Each connection will bind one thread created by thread factory, and then no conflicts would happen among connections.

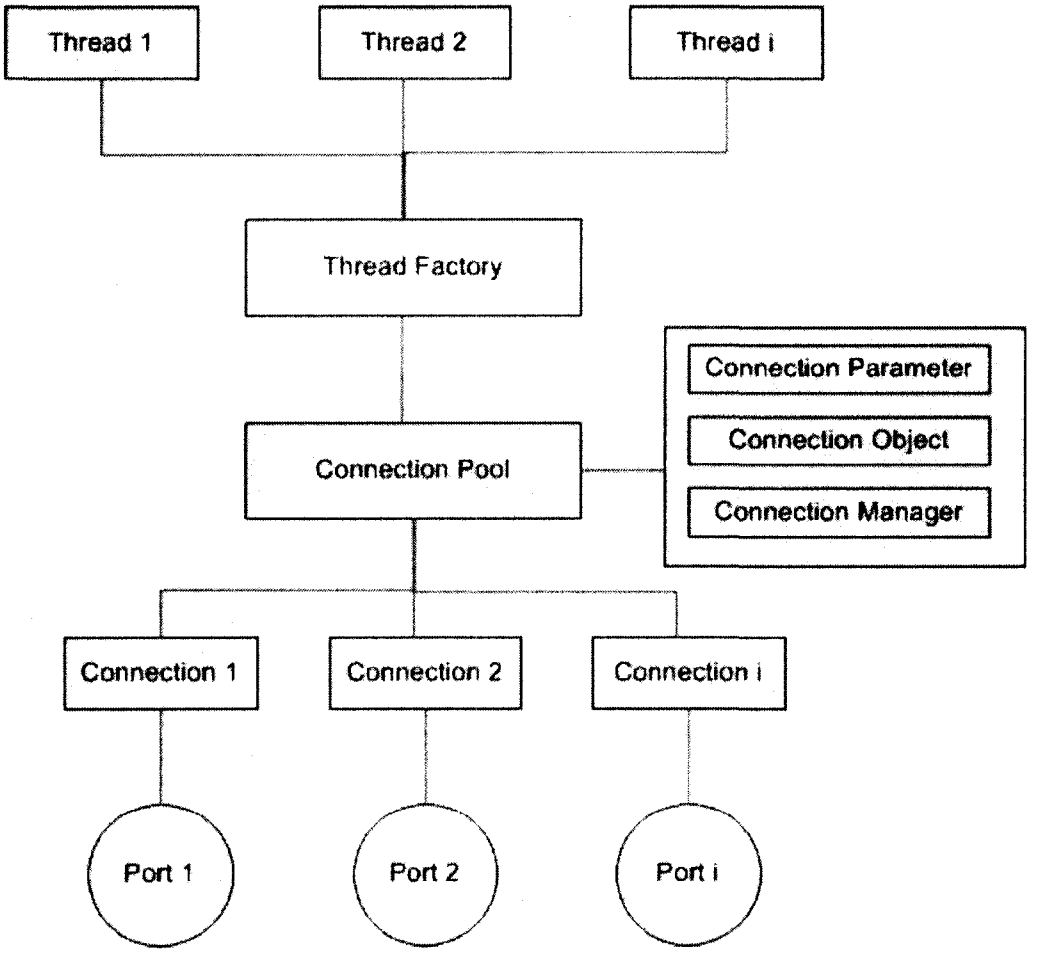

Figure 3.12 Communication model

3.4.4 Message scheduling model

Owing to the latency and possible failure of transmission, outgoing messages could be sent out again. Obviously, such activities are driven by time. In a predefined interval, if an outgoing message does not receive its acknowledgement, it would be retransmitted automatically. If it is retransmitted more than the predefined times, the A-ITEMS will post a system message to notify users that the related request failed. In the A-ITEMS, such time-related activities are performed by message scheduling model shown in Figure 3.13, where the MessageSchedule and the Clock are the primary objects. The Clock object provides the current time while the MessageSchedule keeps track of all pending events and decides which event should occur next, and then triggers the event. The MessageSchedule is usually designed as one thread, and will sleep between every two sequential events.

The MessageSchedule maintains a list of pending EventItems on a PriorityQueue. The EventItem is made up of trigger time, precedence and command. The precedence is used to prioritize events scheduled for the same time and set up specific applications. If more than one EventItem appears on the MessageSchedule with the same trigger time and precedence, they are considered to be simultaneous actions and there is no guarantee as to which order they will be executed.

An object implementing the command interface is supplied by the object requesting an addition to the MessageSchedule. When the MessageSchedule triggers the action of an EventItem, it simply calls the *execute* method of the command object. Any conceivable action may be contained by a command object since the programmer is not limited as to what code goes into this method.

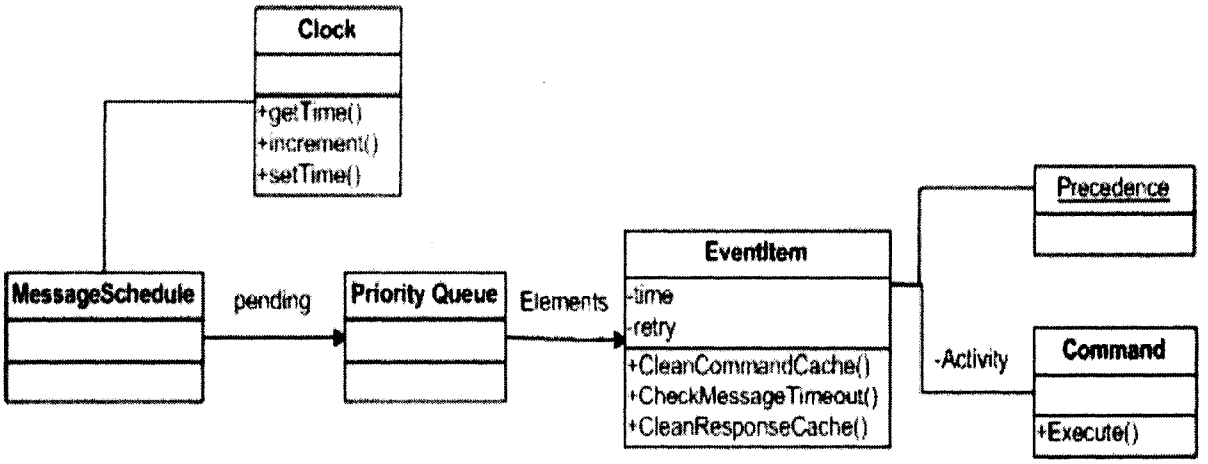

Figure 3.13 Message scheduling model (modified from Graniero, 2001)

3.4.5 Message handling model

In the A-ITEMS, the message is the fundamental data unit and has the specific structure based on protocols. Message handling model is designed to process the message and meet two objectives: the message can be successfully sent to its destination and can be extracted correctly and saved in the proper depositories. Basically, there are five steps to handle messages: 1) build the message; 2) send the message out; 3) receive the message; 4) extract the message; 5) deposit the message. In order to accomplish such five steps, some objects and models need be designed as presented in Figure 3.14. The application sends definite requests to message handler, and then message handler builds corresponding messages in terms of specific protocols. Afterwards, the messages will be sent out. However, because of the complexity of communication, message handler can not guarantee that the messages can successfully reach their destinations. Thus one message pool is designed to keep message tasks which are not finished yet. Herein, one "message task" represents a process of message handling. For example, message handler sends a message to field station A, and then it receives an acknowledgement for this message from field station A. Now we can say that such message task is done. Generally, message tasks are in order by time. Once one message task is done, its related messages would be removed or the state of messages be changed to "Finished". If an acknowledgement can not find its source, it would be ignored. One message could be sent again if the message task is not finished in a predefined interval. Such time events are triggered by message scheduling model. Finally, message handler model calls message parser to extract incoming messages.

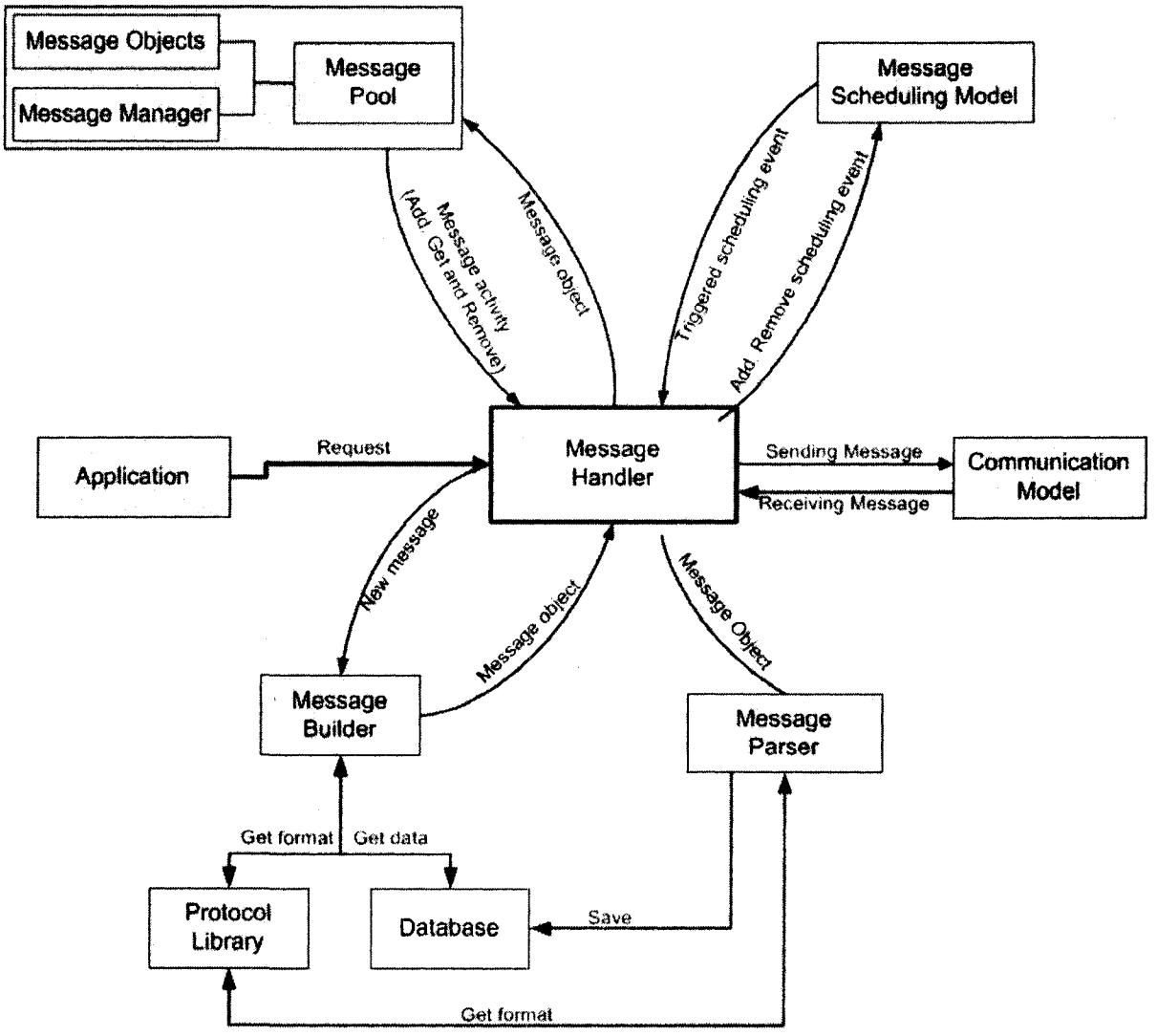

Figure 3.14 Message handling model

Chapter 4 Case Study: Integrated Watershed Telemetry System

Solinst Canada Ltd.[2] had historically developed two telemetry systems to support environmental monitoring. One was the Remote Radio Link (RRL) system, which was used for basic, small-area applications using radio modems to wirelessly connect up to ten remote instruments to a single base station. The other was the Solinst Telemetry System (STS) which was designed for more complex, large-area applications with dozens (or in one case, hundreds) of monitoring stations. The underlying communication infrastructure followed standard SCADA system design and relied on older-generation, circuit-switched cellular modems. However, Solinst found this to be a limitation for their strategic vision of more powerful monitoring systems: remote programming and adjustment, mixed sensor and communication platforms, two-way data flow, and interrupt-driven alerting.

MEMF Lab, University of Windsor and Solinst conceived of an R&D project to: a) evolve the RRL and STS stations into new 'Gold' products (RRLG and STSG) that used next-generation technology and incorporated more advanced system features; and b) develop new Integrated Watershed Telemetry (IWT) stations for complex environmental monitoring applications. The new system would enhance the current technology in several respects by the introduction of an advanced "Home station" into the system. The Home station is an integrated collection of software tools and services operating on one or multiple remote workstations or servers, wirelessly communicating with deployed field stations.

The IWT system would include the following hardware features:

- Easy combination of radio, cellular, and satellite communication;
- Sensors from several vendors working together in one monitoring system;
- Sensor stations independently "pushing" measurements to the home station; and
- Two-way message flow including remote station diagnosis and adjustment.

The inclusion of a more advanced Home station software suite would crate the opportunity to develop the following software features:

---

[2] http://www.solinst.com Last Accessed December 20, 2007

- System-level design, configuration, and management rather than simply programming individual stations;
- Advanced features and behaviours for field stations beyond the actual limited capability of the hardware by emulating the behaviour on a 'virtual' station managed by the Home station; including
- Triggers that adjust behaviours at one or more stations based on events at another station.

These features will open the opportunity to advance the state and practice of watershed monitoring, analysis, and decision support. In particular, the IWT system provides great chances for automated, adaptive sampling and analysis based on hydrological events observed in real time, especially with respect to water quality monitoring.

In this project, MEMFLab[3], University of Windsor, took the role of designing and developing the Home station software. Solinst was responsible for designing and producing the station hardware and firmware, including support for their sensors and selected wireless communication. From October 2006 to August 2007, Dr. Graniero and I designed the first version of the basic Home station. After August 2007, James McCarthy and Dan D'Alimonte joined the project and developed the following version of the Home station software. While McCarthy and D'Alimonte continued with Home station development, I designed and implemented a simple station simulator to mimic the behaviour of Solinst field station hardware.

4.1 Relationship between A-ITEMS and Home station

The Home station is a concrete monitoring system design and operation application that uses the architecture and models of the A-ITEMS in its design. It takes the responsibility of data acquisition and does not care about how data are used. However, it provides the interface for external systems to access data repositories and retrieve data, and also can accept the response from external systems to modify the monitoring configurations to meet some particular requirement (Figure 4.1). For example, if the assessment/analysis subsystem needs more data, it can request the Home station to change the sampling rates of specific sensors. Based on the modular design of the A-ITEMS, the Home station

[3] http://matrix.memf.uwindsor.ca/ Accessed January 11, 2008

can be flexibly embedded into other EMSs and also integrate with other external systems and applications (Jabeur et al., in press).

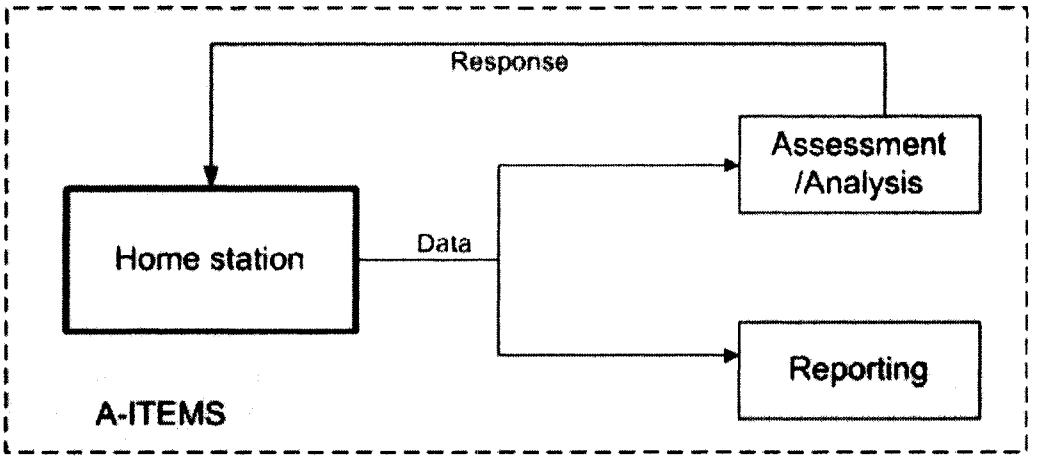

Figure 4.1 Relationship between A-ITEMS and Home station

4.2 IWT equipment overview

The IWT system is constituted by some devices shown in Figure 4.2, where it has only one Home station and multiple IWT networks. Each IWT network has one IWT Bridging station with an IP address except that one standalone sensor station may become one special IWT network.

4.2.1 Field station

The field station is the fundamental data acquisition unit to cache data from its attached sensors and working in field. It is designed and produced by Solinst. The field station can own at most 4 IO boards, each of which can attach 64 different sensors. Because of the limit of memory capacity, one field station can cache at most 10837 measurement records. Each measurement record has 255 measurements no matter how many IO boards or sensors one field station has in fact. For example, if one field station only has two IO boards and each of IO board has 4 sensors. One measurement record still consists of 255 measurements, some of which are equal to 0. This design would waste much more memory space in one field station, and may be improved later.

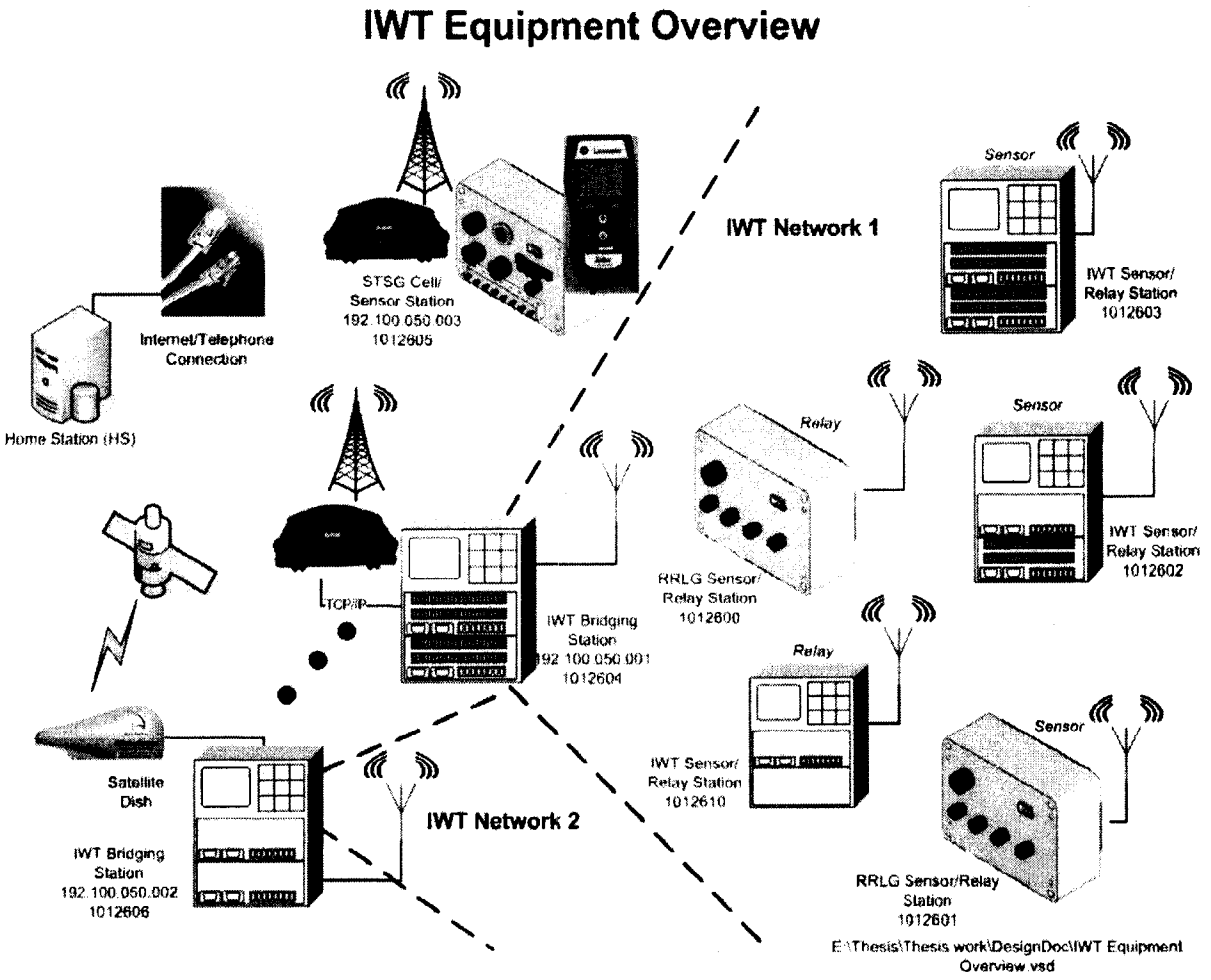

Figure 4.2 Overview of the IWT Equipment (From Solinst Canada Ltd.)

In terms of their specific functions and operations, the field stations can be classified to IWT bridging station, sensor/relay station and cell/sensor station. IWT bridging station directly connects to the Home station and is the only outlet of the monitoring network. It has at least one radio modem which communicates with the modems of other stations, and may own another radio modem with IP address connecting to the Home station. Sensor/relay station takes the responsibility of caching data from sensors or only works as a repeater to transfer data. If it is a repeater, it will work all time. Otherwise, in order to save its power, it has some particular schedules to control its state, such as idle, sleeping and shutdown. Cell/sensor station is also named standalone station and only communicates with the Home station.

4.2.2 Wireless transmission platforms

Figure 4.2 presents three wireless transmission platforms: cellular, satellite, and radio frequency (although landline could be used as well). In the IWT system, different transmission platforms

may be used and need to be switched. Normally, cellular platform is used to connect the Home station and IWT bridging stations. Radio frequency platform solves the communication among field stations and may also be utilized between the Home station and IWT bridging stations.

In general, wireless communication uses high frequency (100 MHz – 5 GHz) carrier waves to transmit information from one site to another. Typically, as technologies advance, the frequencies of the carrier waves increase. The lower frequency spectrum becomes fully occupied whereas the higher the carrier frequency the more bandwidth is available. However, higher carrier frequencies do not provide the coverage that lower frequencies do and the transmitters typically have much lower power. Satellite communications provides excellent coverage and is quite reliable but it is quite expensive.

The wireless transmission platforms are divided into two types: short range modem (SRM) and long range modem (LRM). SRM is constrained by transmission distance (Figure 4.3) and does not rely on a 3rd party carrier service. It typically adopts radio frequency (RF) technology, but may be not limited to this. The SRM's addressing and communication protocol is determined by the manufacturer. Message routing may or may not be automatically handled by the SRM hardware.

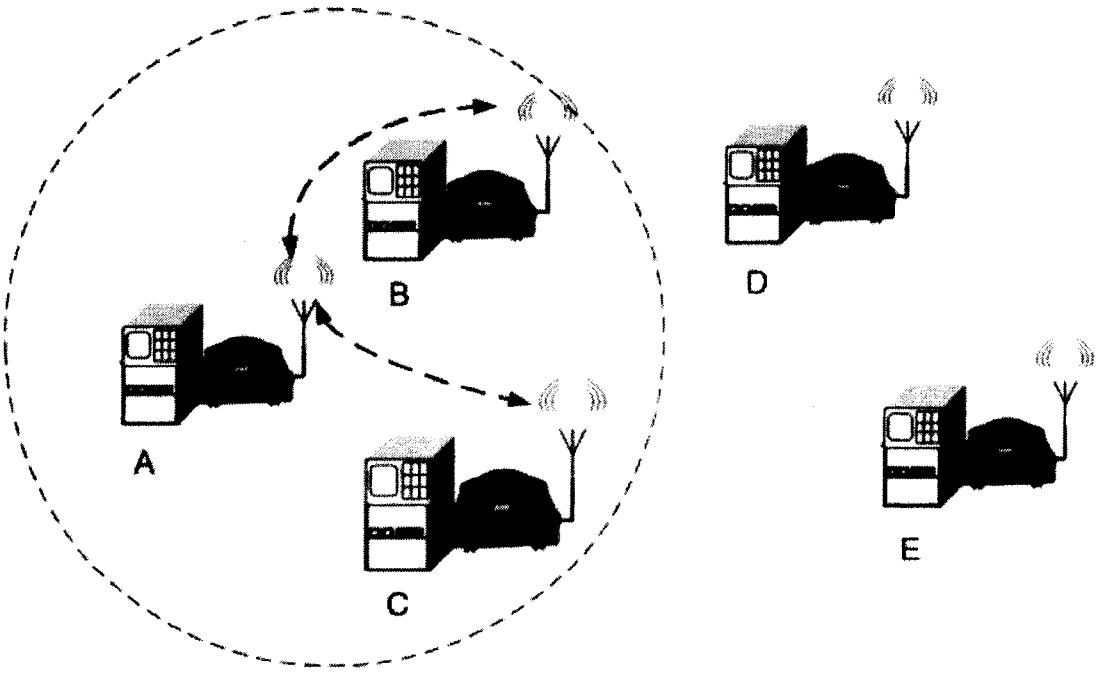

Figure 4.3 Short range modem (SRM)

In Figure 4.3, the dashed circle represents the valid transmission distance. There are five stations,

each of which has one SRM. Only station B and station C are within the transmission scope of station A, which means station A can communicate with station B and station C rather than station D and station E. Of course, station A may only transmit its data to station B based on particular routing mechanism.

The Digi 9XTend RS-232/485 RF Modem, which is produced by MaxStream Inc[4]., was adopted in the project. This modem provides outstanding range (up to 40 miles with outdoor line-of-sight range) and security in a low-cost wireless solution, and is coupled with a DIP switchable RS-232 / RS-422 / RS-485 interface board. It builds a RS-232/RS-485 interface and supports advanced networking & low-power modes.

LRM, compared to SRM, is not constrained by transmission distance (Figure 4.4). LRM typically utilizes cellular or satellite technology, but is not limited to these. It typically relies on a 3rd party carrier service for transmission. Its addressing and communication protocol is also determined by the 3rd party carrier service. Routing is handled by the LRM and the 3rd party carrier service.

In Figure 4.4, the dashed rectangle presents the LRM connection. The coverage of LRM depends on the cellular tower distribution which is supported by the carrier service. The LRM usually utilizes Internet and TCP/IP protocols to transmit data.

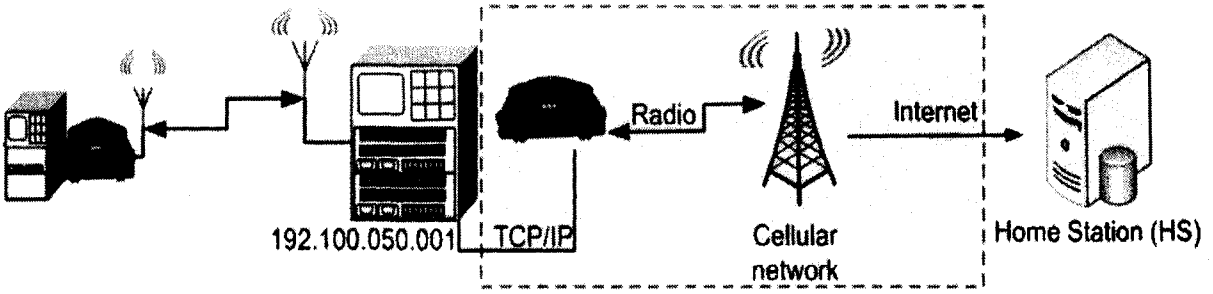

Figure 4.4 Long range modem (LRM)

The LRM usually uses cellular technologies such as GSM, CDMA, GPRS, etc. For example, the

[4] http://www.maxstream.net/ Accessed January 11, 2008

BlueTree 4200 CDMA produced by BlueTree Wireless Data Inc.[5] is a smart, rugged LRM built to provide simple and reliable communications over the cellular CDMA (1xRTT) data network, and the wireless service in Canada can be provided by Bell Canada. Its static IP address is from Bell Canada. Before it works, it must be activated and gain one specific IP address from Bell Canada.

4.3 High level architecture of Home station

With reference to the A-ITEMS architecture (Figure 3.3), a high level architecture was designed for the Home station (Figure 4.5). In order to conveniently manage data, two databases are defined: measurement database, which stores measurements and configuration database, which keeps the setting information of monitoring networks and may or may not contain the system information of the Home station. Certainly, there may be another database to manage the system execution information of the Home station. Each database has one specific database manager which controls the external access and maintains the internal data. Before one operation tries to access databases, it must apply for the authorization from database connection manager. The configuration controller manipulates the design of monitoring networks and responds to and validates the external or internal design operations. The external design operations are directly from users' input, whereas the internal design operations are created by some particular functions, e.g. trigger functions, analysis functions, and so on. The operation controller maintains two-way communication, receives data and sends messages out. When messages are received, the operation controller confers with monitoring network representation to properly parse the message.

[5] http://www.bluetreewireless.com/ Accessed January 13, 2008

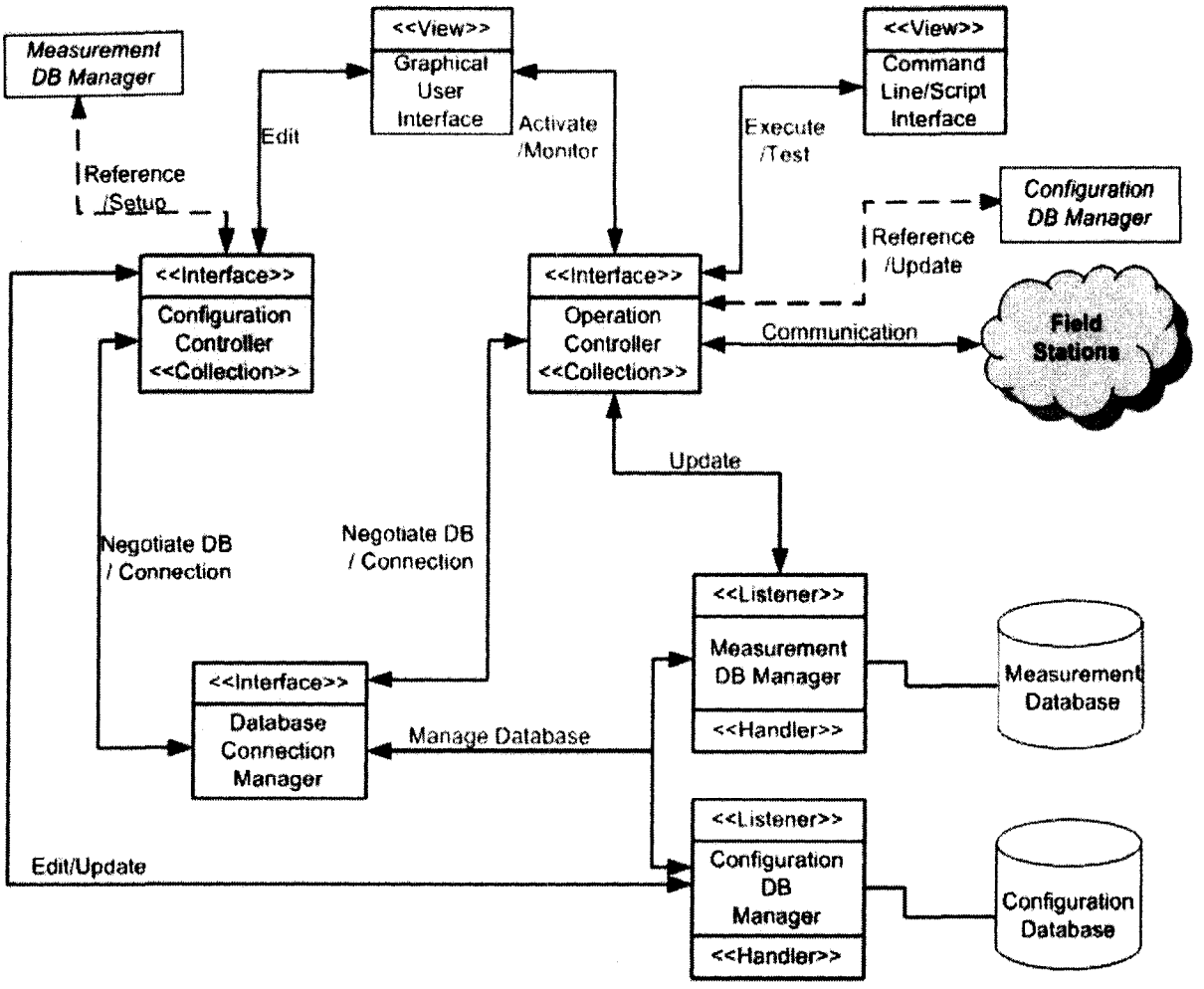

Figure 4.5 High level architecture of Home station. Compare to Figure 3.3.

4.3.1 Development platform

At the foundation of all system software, an operating system performs such basic tasks as controlling and allocating memory, prioritizing system requests, controlling input and output devices, facilitating networking and managing file systems[6].

As of 2007, Microsoft Windows held a large amount on the worldwide desktop market share. The most widely used version of the Microsoft Windows family is Microsoft Windows XP, released on October 25, 2001. Though in November 2006, Microsoft released Windows Vista, a major new

[6] http://en.wikipedia.org/wiki/Operating_system Last Accessed January 10, 2008

version of Microsoft Windows which contains a large number of new features and architectural changes, Microsoft Windows XP currently is utilized by most clients. Even though some clients would use Windows Vista, the Home station can easily be operated except some limited updates. Therefore, Microsoft Windows XP is chosen as the operating system platform to develop the Home station.

The Home station is developed upon the application level, and also needs the support of low level functions, e.g. message handling, communication handling, and so forth. C++ is regarded as a mid-level language and comprises a combination of both high-level and low-level language features (Herbert, 1998). Therefore, C++ was adopted to develop the Home station. Microsoft Visual Studio was used for development, and Microsoft's .NET Framework[7] was used to support several important operating system mechanisms like serial port control and multi-threaded concurrent processing.

Multithreading is becoming an increasingly important part of modern programming. One reason is that multithreading enables a program to make the best use of available CPU cycles and thus allow very efficient programs to be written. Another reason is that multithreading is a natural choice for handling event-driven code, which is so common in today's highly distributed, networked, and GUI-based environments.

The Home station is required to support multiple IWT networks, which may communicate with the Home station concurrently. The Home station needs multiple ports to transfer messages, and also processes different messages concurrently. Multithreading mechanism provides such capability for the Home station. Each port is abstracted as one individual thread, which processes the messages to and from this port.

4.4 Station simulation

As part of the project I developed a station simulator as a stand-alone application. It can mimic a real field station to sample data, report data, update hardware firmware, and generate alarms.

A station can be simulated in software so that a real field station need not be deployed in the earlier stages of EMS design. The station simulator can be utilized to meet the following tasks: i) It can test the design of large sensor networks without large cost of deployment of real devices. This is quite

[7] http://www.microsoft.com Last Accessed January 10, 2008

necessary in the environmental monitoring domain because the environmental monitoring program may have a large monitoring area and a large number of stations, and installation, adjustment, and re-deployment costs can quickly grow; ii) Home station software development and enhancements can be conveniently tested against a station simulator during design and implementation in order to detect errors and validate correct behaviour; iii) As described earlier, the Home station may communicate with some field stations. Multiple artificial station simulators can mimic these real field stations and cooperate with the Home station in the software application domain, which is critical to test the design of the Home station as well as users' applications; iv) In order to acquire ideal data for a particular monitoring problem, or to avoid data loss in a particularly difficult deployment environment, some particular parameters may need to be experimented with before they are set correctly, e.g. sampling rate, reporting rate, etc.

Figure 4.6 presents the general structure of a station simulator. The Home station edits the specifications of each station, and exports its specifications to one configuration file. The station simulator utilizes the configuration file to initialize the simulated station. It can communicate with the Home station as if it were a real station. Multiple stations can be simulated to concurrently communicate with the Home station.

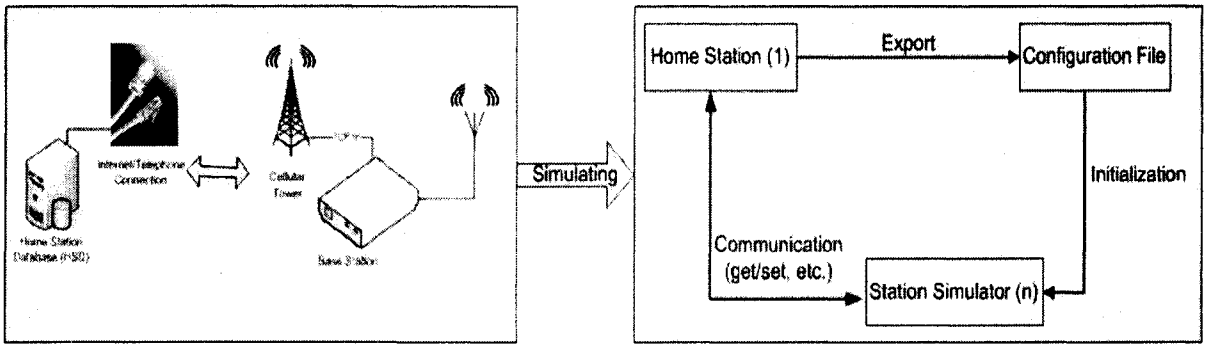

Figure 4.6 General station simulator structure

4.4.1 High level architecture

The station simulator consists of several primary components, which are presented by a high level architecture in Figure 4.7. The simulated station is initialized by one configuration file, which contains the fundamental specifications of one specific station. The synchronization needs to be accomplished to guarantee that the simulator is using the same clock system as the Home station. The function of request update is to get the current firmware version of the station and make sure that the simulator is using the same version as the Home station. The scheduling procedure is designed to control the timing of sampling and reporting. The sampling procedure can get measurements from

multiple sensors. Once the reporting procedure reaches the schedule, it can report measurements via the communication port procedure. The protocols take the role of encoding and decoding messages. In order to simulate multiple stations, a separate instance of the simulator is executed for each station. The multiple simulators may be executed on a single machine or multiple machines, which depends upon valid communication ports.

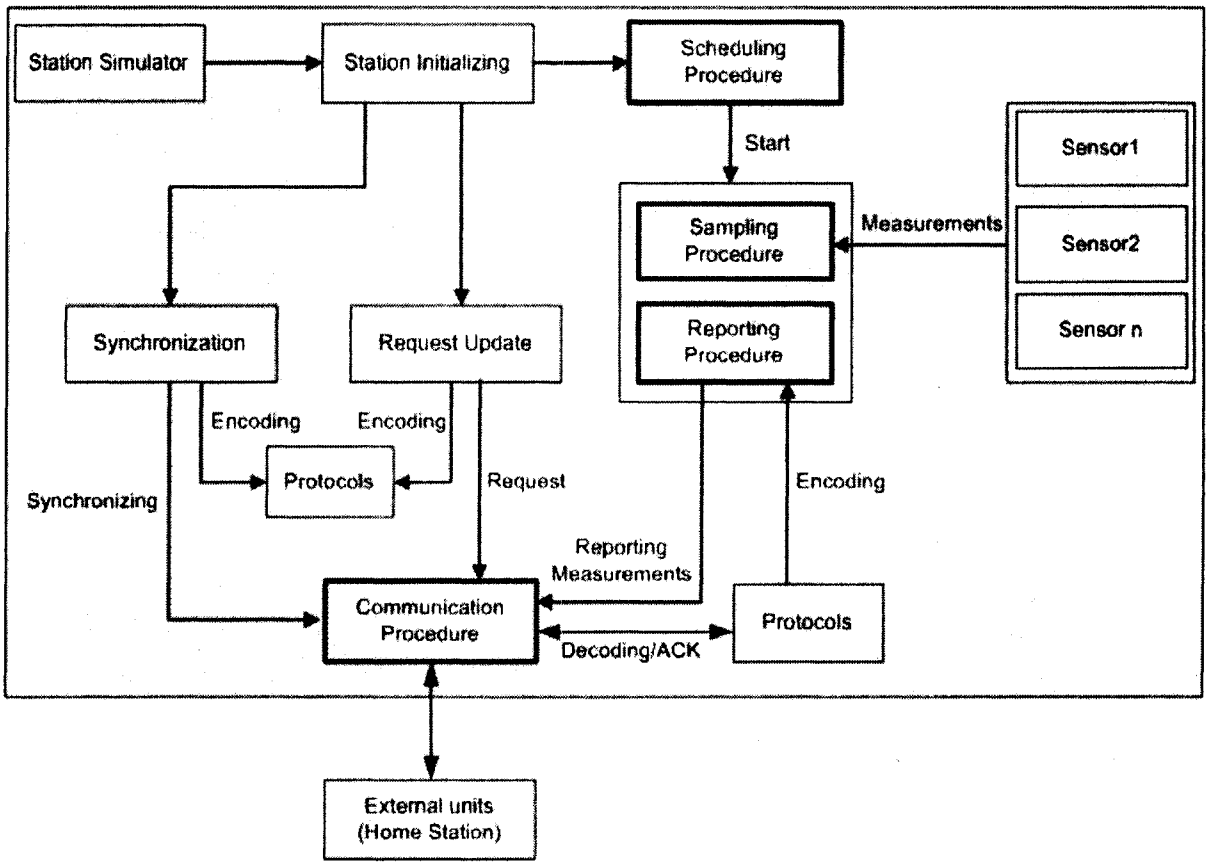

Figure 4.7 High level architecture of station simulator

4.4.2 Simulation environment

The simulation is implemented under Microsoft Windows XP and contains some basic components:

- One Home station. It designs monitoring networks, and can export the specifications of each station. It runs on one machine and communicates with monitoring networks via some specific communication modes. For current simulation, the serial port is utilized.

- One or multiple station simulators. The simulator uses the specifications of one station to create corresponding simulated stations. Each simulator receives and submits messages via its fixed serial port. Currently, the simulator is used to simulate Solinst field stations.

- Specific communication modes. Two kinds of communication modes are used. One is to utilize MaxStream radio modems[8]. The Home station uses one MaxStream radio modem to communicate with each simulator, which also has one MaxStream radio modem. The other is to utilize virtual serial port, which is a redirector without network software support and usually used to create a pair of back-to-back virtual COM ports on the same computer. The simulator and the Home station can then communicate using virtual serial ports instead of conventional inter-process communication mechanisms such as named pipes. Such a virtual serial port is capable of emulating all serial port functionality. For example, COM1 and COM2 can be created, and then directly used without one real cable. The mechanism is presented by Figure 4.8.

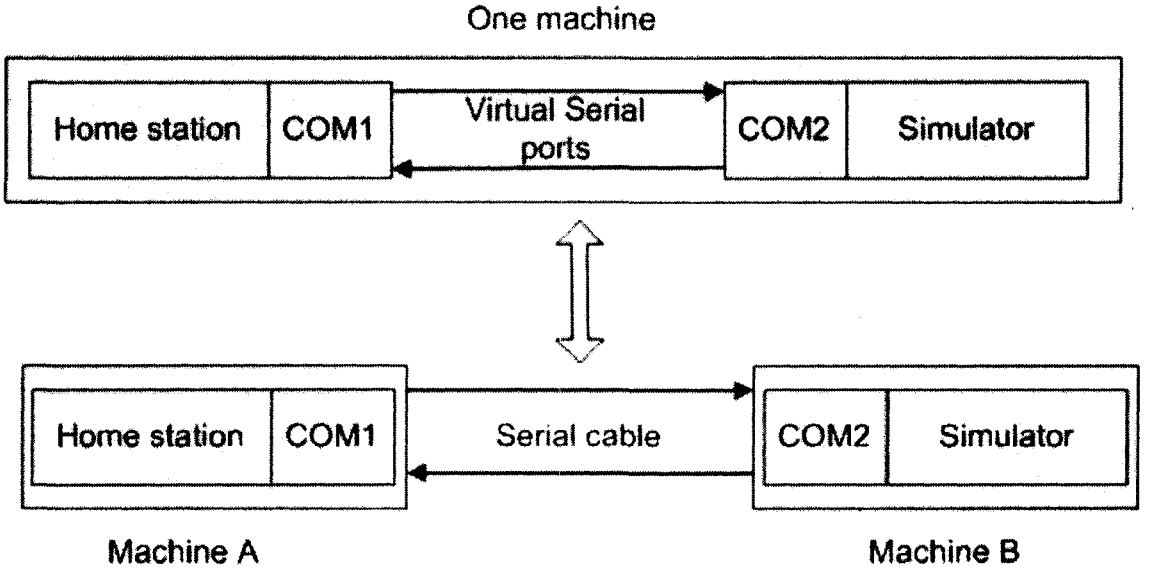

Figure 4.8 Mechanism of virtual serial port

- Some basic predefined parameters. Retry number is set as 3 and means how many times one message transmission loop can be repeated if it fails. Delay time uses 3 minutes and represents how long the simulator can wait for the response of one message. If the simulator does not receive the response, it needs to submit the message again. IO boards and sensors are listed in Table 4.1, where the types of measurements are designed only to conveniently debug the simulator. They can be replaced by some real types of measurements.

[8] http:// www.maxstream.net Last Accessed October 10, 2007

Table 4.1 Settings of IO boards and sensors

IO Board	Sensor	Type	Measurement
1	1	1	3.0, fixed value
	2	2	Current second of system clock
2	1	1	3.0, fixed value
	2	2	Current second of system clock

4.5 Prototype tests

4.5.1 Remote configuration

The Home station would control the behaviours of stations through remote configuration. Here, one test shows that the Home station communicates with one simulator and enquires what the current reporting rate is as well as modifies the rate as one new value.

In Figure 4.9, the Home station sends the simulator one request to gain the reporting rate. The simulator receives the request and sends the reporting rate back. The Home station gets 1400. In Figure 4.10, the Home station sets up the reporting rate as 1600 and sends the update to the simulator. The simulator receives the update and modifies its firmware, which shows "ReportingRate is changed as 1600".

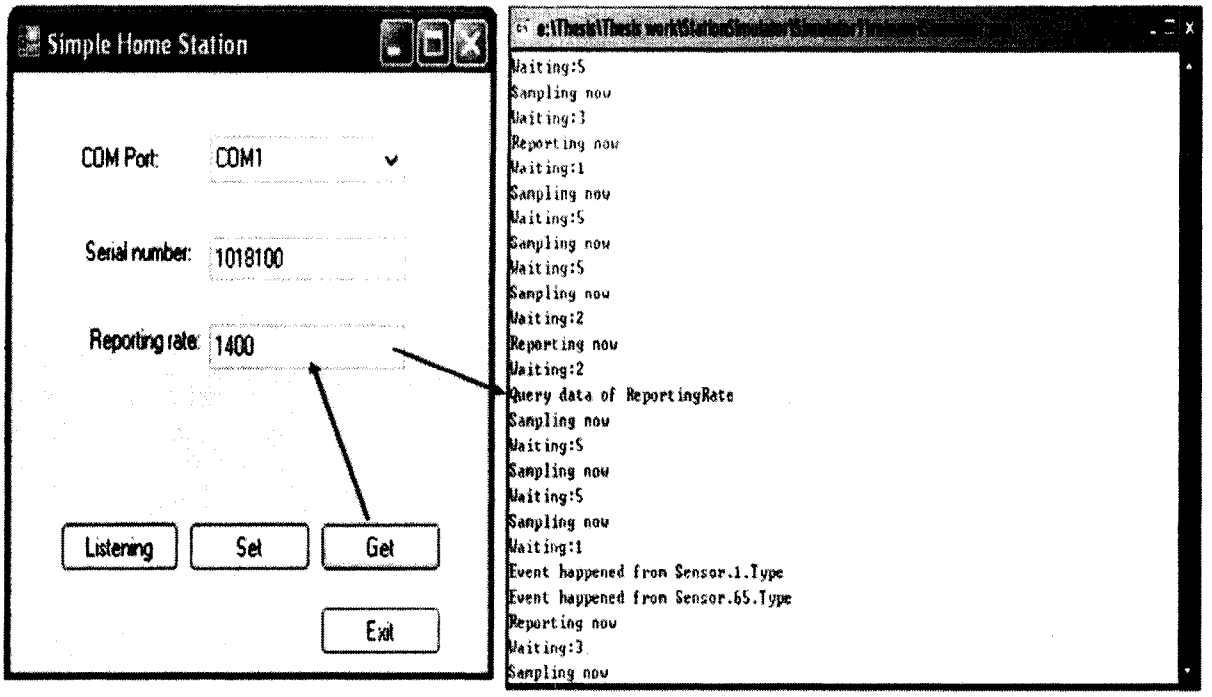

Figure 4.9 Get value

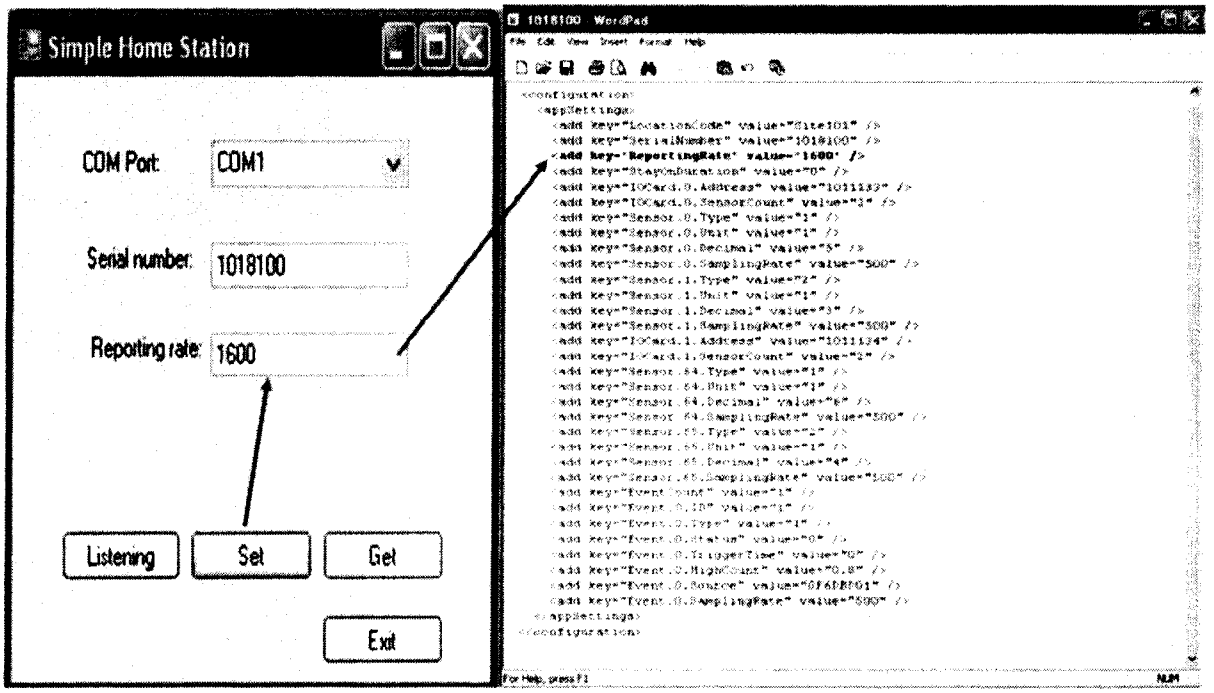

Figure 4.10 Set value

4.5.2 Support multiple stations

In terms of the design of the IWT system, multiple stations would concurrently connect to the Home station. Here, one test of two stations is implemented. The test infrastructure is shown in Figure 4.11. Each station simulator has one unique serial number, which is contained in its messages, and one radio modem with definite source address and destination address. The Home station needs to know the destination address in terms of the serial number of one station. It also has one radio modem with source address and initial destination address. When the Home station sends messages to one specific station, it first changes the destination address of its radio modem as the source address of the radio modem of this station using AT command. For example, if the Home station submits one message to the station 1018100, it needs to change the destination address "n" as "1", and then the message can reach the station 1018100.

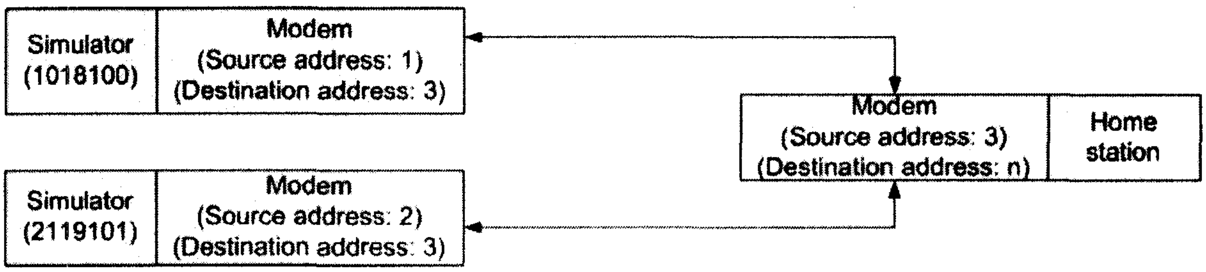

Figure 4.11 Support multiple stations

In order to implement this test, the responses are logged for each station. If two stations can receive their responses respectively, then it would say the Home station can operate these two stations. The partial test result is presented in Table 4.2. Each simulator submits different messages except L command, so most of the responses must be different. From Table 4.2, the performance does show this difference. Based on the same mechanism, the Home station would support multiple stations. But at most how many stations the Home station can support depends on communication and computing capacities

Table 4.2 Responses of two stations from Home station

Simulator: 1018100	Simulator: 2119100
2E-31-33-2F-30-37-2F-32-30-30-36-01-30-38- 3A-32-36-3A-31-34-7E-80 2D-0F-C0 AA-00-00-00-00-00-00-00-00-00-00-00-00-00- 00-00-00-00-00-00-00-00-00-00-00-00-00-00- 00-00-00-00-00-00-00-00-00-00-00-00-00-00- 00-00-00-00-00-00-00-00-00-00-00-00-00-00-	A1-31-33-2F-30-37-2F-32-30-30-36-01-30-38- 3A-32-36-3A-31-34-3A-59 2D-0F-C0 44-00-00-00-00-00-00-00-00-00-00-00-00-00- 00-00-00-00-00-00-00-00-00-00-00-00-00-00- 00-00-00-00-00-00-00-00-00-00-00-00-00-00- 00-00-00-00-00-00-00-00-00-00-00-00-00-00-

00-00-00-00-00-00-00-7A-E2 2D-0F-C0 2D-0F-C0-2D-0F-C0 2D-0F-C0 39-08-C0 2D-0F-C0 2D-0F-C0 2D-0F-C0-3A-F7-80 2D-0F-C0-2D-0F-C0 D8-2A-00 2D-0F-C0 08-96-01 2D-0F-C0 2D-0F-C0-2D-0F-C0 2D-0F-C0 ...	00-00-00-00-00-00-00-5B-95 2D-0F-C0 2D-0F-C0 8A-A3-81 2D-0F-C0 2D-0F-C0 2D-0F-C0-86-5E-81 2D-0F-C0 2D-0F-C0 2D-0F-C0-18-FA-00 2D-0F-C0-2D-0F-C0 2B-F1-40 2D-0F-C0 2D-0F-C0 B5-55-C1 2D-0F-C0 ...

4.5.3 Message transmission performance

Message Transmission is a kind of loop operation, , which is presented in Figure 4.12 and means that the station sends one message out and afterwards expectedly receives its specific acknowledgement.

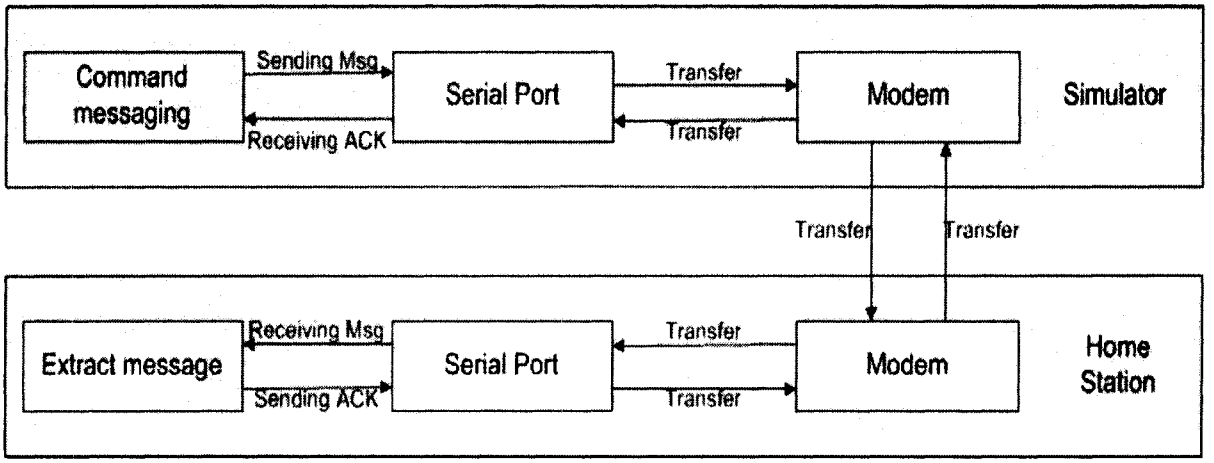

Figure 4.12 Message transmission loop

In order to show the performance of message transmission with the possible influence of different baud rates, one testing infrastructure is designed in Figure 4.13. Because of the design of Solinst protocol, the data field in each message has at most 256 bytes in the IWT system. Plus other extra fields, one message has at most 263 bytes. The following message is retrieved from the measurement message in the station simulator. As introduced above, one measurement record of the Solinst station consists of 255 measurements even though the station may not have 256 sensors. Therefore, some

measurements may be 0. Here, using one message with many bytes would demonstrate the performance of message transmission better. Certainly, other messages may be used as well.

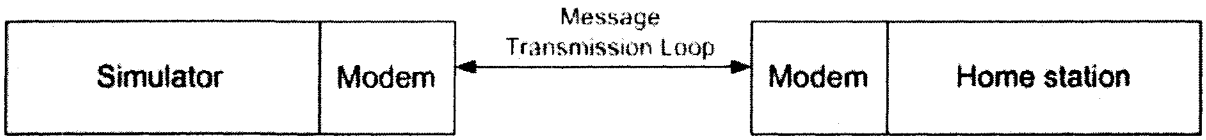

Figure 4.13 Testing infrastructure for message transmission with different baud rates

The message consists of 261 bytes: 00-44-0F-88-F4-F9-00-55-65-47-77-BC-74-61-A3-00-80-00-61-A3-00-80-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-F1-50.

The baud rate is changed to 9600, 19200 and 115200 bps, respectively. The results are shown in Table 4.3, where the consumed time is the mean value of 10 times message transmissions. These consumed time values are pretty close and the maximum difference is approximate 2 seconds. For the particular EMSs, which require second resolution for sampling and reporting, such difference would be considered. However, the mean consumed time is more than 20 seconds, which would be a serious issue in some particular EMSs. The optimization may be required to improve the message transmission in the future.

Table 4.3 Test for message transmission with different baud rates

Baud rate (bps)	Consumed time (second)
9600	25.324
19200	24.356
115200	22.612

4.5.4 Relationship between sampling rate and reporting rate

Sampling rate represents how quickly stations get measurements from sensors. Reporting rate implies how often stations send measurements to the Home station or other stations. In the IWT system, one fundamental rule is that once the reporting event happens, all current measurements should be successfully reported, which can make sure that the space in one station is enough to keep all incoming measurements and no measurements would be unexpectedly overlapped. So, it is quite important to correctly set the sampling rate and reporting rate. The following issues would be tested and provide some clues.

- If the sampling is fast or high, more measurements will be gained. Then the reporting task should need more time.
- The more stations there are, the more time the reporting needs

4.5.4.1 Theoretical relationship

The following steps present the quantity relationship between sampling rate and reporting rate based on the fundamental rule: if one reporting event happens, all the current measurements must be reported before the next reporting event.

1) Calculate the time for message unit: T_{Unit}

$T_{Unit} = T_S + T_C + T_I$, T_S is the processing time of station simulator, T_C means the communication time, and T_I represents the processing time of the Home station. For Solinst protocol, each complete reporting unit contains two basic message units: L command and D command. Therefore, they need such time as:

$T_{Unit-L} = T_{SL} + T_{CL} + T_{IL}$ and $T_{Unit-D} = T_{SD} + T_{CD} + T_{ID}$ Then $T_{Unit} = T_{Unit-L} + T_{Unit-D}$

2) Calculate the time for one measurement record T_{Rec}

Each measurement record has 769 bytes, and each time the maximum reporting bytes are 252. Each measurement record consists of 4 message units. If the difference of consumed time among four message units is ignored, then,

$T_{Rec} = 4 \times T_{Unit}$

3) Calculate how many records need to be reported: n

The sampling rate is $T_{Sampling}$ and the reporting rate is T_{Report}, then

$$n = \left\lceil \frac{T_{Report}}{T_{Sampling}} \right\rceil$$

4) Relationship based on the fundamental rule:

$$T_{Report} \geq n \times T_{Rec} = \left\lceil \frac{T_{Report}}{T_{Sampling}} \right\rceil \times T_{Rec}$$

$$\Rightarrow T_{Sampling} \geq T_{Rec}$$

For one specific communication mode and execution environment, T_{Rec} would be one constant. n can be predefined in terms that how many records need to be cached. Besides $T_{Sampling} \geq T_{Rec}$, the sampling rate and reporting rate would have the following relationship:

If $n = 1$, *then* $T_{Sampling} \leq T_{Report} < 2T_{Sampling}$, which means that only one cached record needs to be reported.

If $n = 2$, *then* $2T_{Sampling} \leq T_{Report} < 3T_{Sampling}$

.....

And so on until n reaches the maximum quantity of measurement records in the station. If n reaches the maximum quantity, the earliest records will be replaced.

4.5.4.2 One station: reporting rate and sampling rate

In order to conveniently do this test, one pair of virtual serial ports is utilized. The baud rate is 9600 bps. As the test of message transmission performance shows, one message transmission would consume more than 20 seconds. Then one measurement record would need more than 80 seconds. According to the theoretical relationship between sampling rate and reporting rate, only one measurement record would be cached and reported. Then the sampling rate and reporting rate are set as 120 seconds and 130 seconds, respectively. The test is implemented for 1 hours and 15 minutes. The consumed time is logged in Table 4.4.

Table 4.4 Consumed time of reporting operation with one station

Consumed time (ms)	Record	Consumed time(ms)	Record	Consumed time(ms)	Record
118907	1	124829	1	229735	2
129094	1	231781	2	228016	2
130938	1	227672	2	330921	3
121094	1	235907	2	229734	2
133172	1	229688	2	223656	2
127063	1	223640	2	223672	2
124797	1	318500	3	215734	2
136860	1	221671	2		

The mean consumed time is 118.42 seconds, the minimum is 106.166 seconds and the maximum is 136.86 seconds. In Table 4.4, the first logged records have only one measurement record to report, but the last logged records have 2 or 3 measurement records. This would happen because the consumed time of one measurement record may be affected by some facts, for example, the computer became slow sometime. The first logged records show that the consumed time of one measurement record is not constant. However, the entire performance still meets the theoretical relationship between sampling rate and reporting rate.

The test result is also plotted in Figure 4.14, where s/r means seconds per measurement record. From the plot, record count and consumed time become an approximately linear relationship. That means that the more measurement records the simulator needs to report, the more time it consumes. The linear relationship also meets the above theoretical analysis.

4.5.4.3 Two stations: reporting rate and sampling rate

Theoretically, tow stations should consume more time than one station because the Home station has to change the destination of its modem and does other operations, e.g. querying the information of different stations, creating new station objects, etc. The baud rate, the sampling rate and reporting rate are not changed. The test is implemented for 1 hours and 2 minutes. The consumed time is logged in Table 4.5.

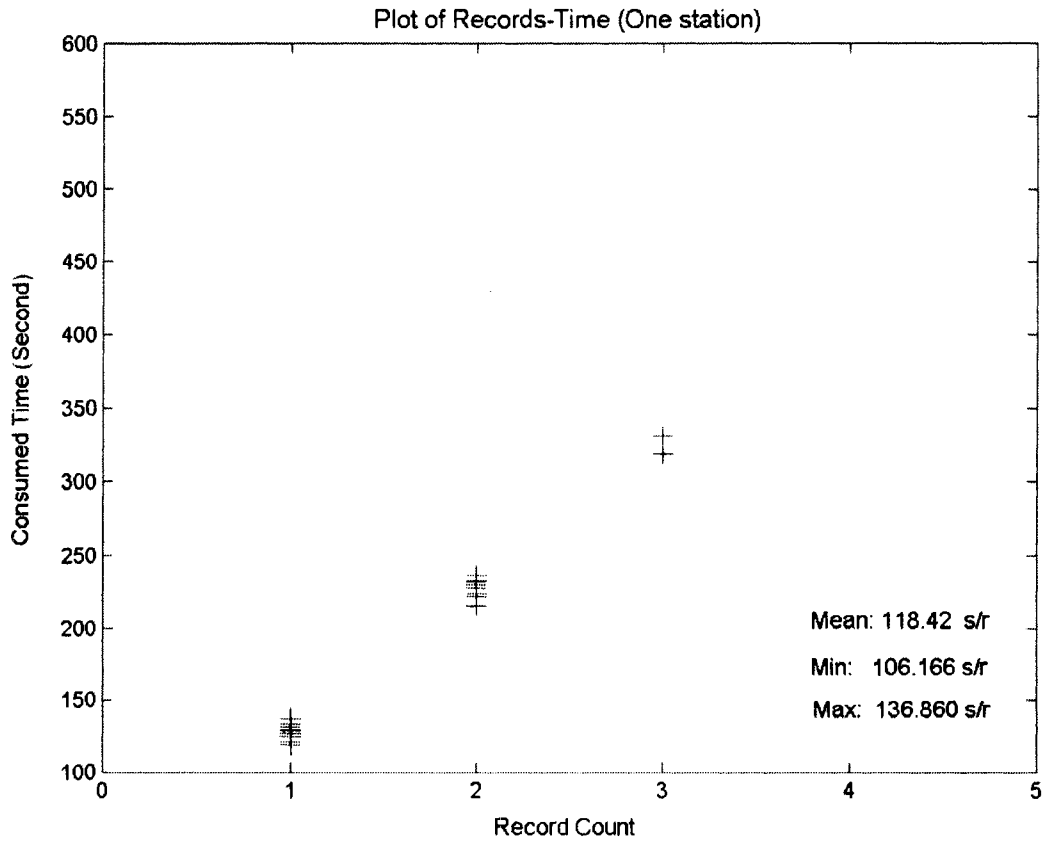

Figure 4.14 Consumed time of reporting with one station

The mean consumed time is 145.56 seconds, the minimum is 126.844 seconds and the maximum is 183.63 seconds. In Table 4.5, the first logged records have only one measurement record to report, but the last logged records have 2 or 3 measurement records. It would have the same reason as the test of one station. It is obvious that two stations consume more time than one station.

Table 4.5 Consumed time of reporting operation with two stations

Consumed time (ms)	Record	Consumed time(ms)	Record	Consumed time(ms)	Record
136812	1	142985	1	279766	2
134875	1	181031	1	285968	2
138844	1	126844	1	550891	3
147062	1	279734	2	430797	3
132781	1	267844	2		
162922	1	281937	2		

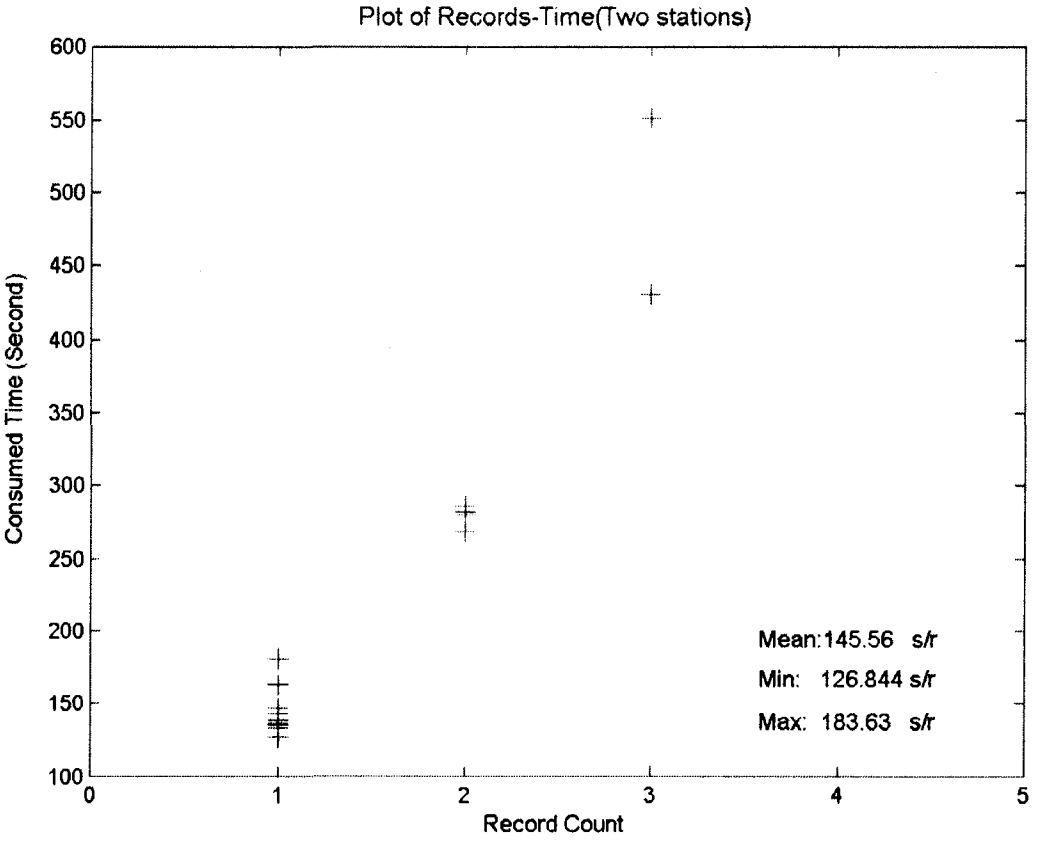

Figure 4.15 Consumed time of reporting with two stations

Figure 4.15 plots the test result. From the plot, record count and consumed time have one approximately linear relationship though the test of one station has a more significant linear relationship. That means that the more measurement records the simulator needs to report, the more time it consumes. The linear relationship also meets the above theoretical analysis.

4.5.5 Trigger mechanism

The trigger mechanism may be quite complex. For example, one trigger may change some specifications of other stations rather than only show some alarm messages in the Home station. Here, one test is implemented for the basic trigger mechanism. One high threshold (50) is set for two sensors of the second IO board. As long as the measurement of these sensors is more than 50, this trigger will happen as shown in Figure 4.16. The type of this trigger is predefined. Once it happens, the simulator shows one notification, "Trigger type 1 happened from Sensor.1.SensorType" on the screen and also sends the alarm message to the Home station.

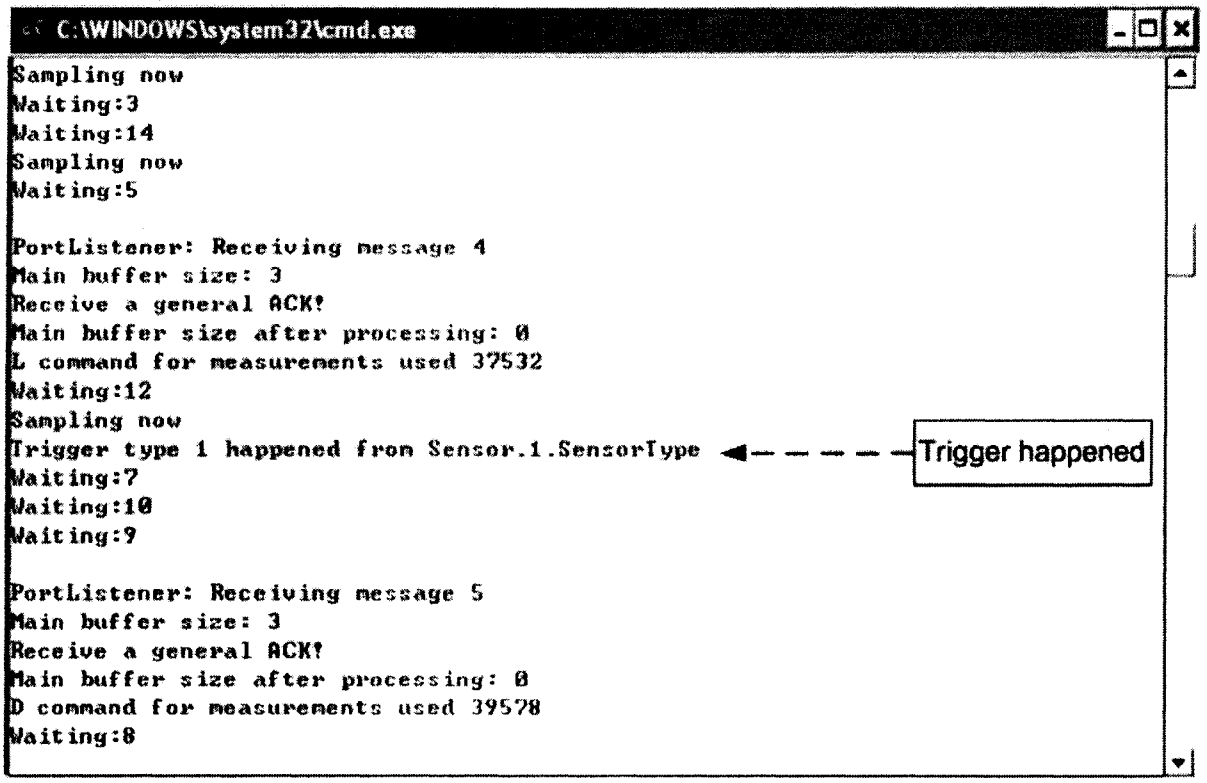

Figure 4.16 Trigger mechanism in station simulator

Chapter 5 Summary

The technology of environmental sensors, wireless communication, and rugged in-field computing are advancing rapidly. It is important that EMS are designed to take advantage of new generations of hardware and software. In this thesis, an architectural framework for developing advanced, integrated EMS was presented. The architectural design was based on requirements developed from direct interaction with a company that develops hydrological sensors and telemetry systems, using their knowledge and customer experience as well as literature reviews. The resulting A-ITEMS architecture captures those requirements, and the modular and integrated architecture gives EMSs designed with the A-ITEMS architecture more flexibility for implementing some or all of those requirements. Monitoring network software may be rapidly designed or modified by following the A-ITEMS framework. One or more system designers can share or integrate components, making software development and system design more flexible and fast. Different EMS that are based on the A-ITEMS architecture and its abstraction and interface design, but may be developed by different vendors, can more likely interact and cooperate with each other, achieving a degree of interoperability which is a design goal for many technology areas including sensor networks and environmental monitoring.

The modular architecture of A-ITEMS, which allows substitution of algorithms or use of simulated components, is also important for EMS software design, development, and testing. By simulating real hardware components within the EMS, bugs in the data flow and message manipulation can be found. The system's performance can be tested under scenarios that are difficult to create in the real world, such as message processing speed when communicating with hundreds of field stations, or fault tolerance and recovery when hardware or communication errors occur at the field stations.

Although the current IWT Home station and station simulator implementations only accommodate Solinst stations, it provides the ability through the A-ITEMS architecture to quickly add support for other hardware. Just as the station configuration can generate either memory map commands or a simulator configuration file, additional generator objects can be written to configure other hardware. MEMF Lab is currently creating station configuration and message protocol objects to support Crossbow mica mote hardware operating with the SWL Sensor Web Language (Nickerson and Lu, 2004).

Some specific acquisition parameters, such as sampling rate, reporting rate, etc., are critical for an EMS and need be set correctly to avoid loss of data or costly battery drain. For example, the higher

reporting rate is, the more messages that must be transmitted and processed, without necessarily increasing the quality of information about environmental dynamics. Sampling rate and staying-on duration are also quite related to the system performance. So before monitoring networks are deployed, these acquisition parameters should be carefully balanced. A simulation approach can test the system performance without deploying real devices. The case study demonstrated that, by using a modular architecture such as A-ITEMS, an EMS can substitute simulated components directly into the operational field system for experimentation. Based on simulation experiments that test the system performance, these parameters can be optimized before deployment, increasing confidence that the system will operate in an effective manner without expensive adjustments in the field.

Some basic triggers can be designed and processed, such as high-value threshold, low-value threshold, etc. The A-ITEMS designs the component, trigger processing, to process the trigger-related messages. In the thesis, one basic trigger with high-value threshold was demonstrated by the IWT system. Once one trigger happened, an alarm message will be built and sent to users. The A-ITEMS architecture gives the capability that in the future, the basic trigger manager components can be with more advanced components that can gather more information from the sensor network configuration database and can represent more sophisticated trigger conditions and actions. This will make the basic field stations with limited capabilities seem as if they are much more powerful, and more subtle and important environmental events can be detected, which is important for hazard detection and resource management.

References

Basagni, S., Chlmtac, I., Syrotiuk, V. and Woodward, E. 1998. A distance routing effect algorithm for mobility (DREAM). *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, Dallas, TX. 76-84.

Booch, G. 1993. *Objected-Oriented Analysis and Design with Applications* (2nd ed.). Addison-Wesley. 608pp.

Braden, R. (ed.) 1989. *Requirements for Internet Hosts—Communication Layers.* Internet Engineering Task Force RFC 1122. Information Sciences Institute (ISI). 116pp.

Brydges, T. 2004. Basic concepts and applications of environmental monitoring. In:Wiersma, G.B. (ed.) *Environmental Monitoring:* CRC Press. 83-109.

Colbert, D.C., and Carder, K. 1971. Environmental data acquisition telemetry system. *OCEANS*, 3:373-376.

Consultative Committee for Space Data Systems. 1987. *Telemetry Summary of Concept and Rationale.* CCSDS Green Book 100.0-G-1. Green Book. Washington, D.C.

Consultative Committee for Space Data Systems. 1995. *Recommendation for Space Data System Standards: Packet Telemetry.* CCSDS Blue Book 102.0-B-2. Washington, DC. 41pp.

Culler, D., Estrin, D. and Srivastava, M. 2004. Overview of sensor networks. *IEEE Computer*, 37(8):41–49.

Delin, K.A. and Jackson, S.P. 2001. The Sensor Web: A new instrument concept. *SPIE Symposium on Integrated Optics*, San Jose, CA. 9pp.

Delin, K.A, Jackson, S.P., Johnson, D.W., Burleigh, S.C., Woodrow, R.R., McAuley, M., Britton, J.T., Dohm, J.M., Ferre, T.P.A., Ip, F., Bucker, D.F., and Baker, V.R. 2004. Sensor Web for spatio-temporal monitoring of a hydrological environment. *35th Lunar and Planetary Science Conference*, League City, TX. 2pp.

Delin, K.A, Jackson, S.P., Johnson, D.W., Burleigh, S.C., Woodrow, R.R., McAuley, M., Britton, J.T., Dohm, J.M., Ferre, T.P.A., Ip, F., Bucker, D.F., and Baker, V.R. 2005. Environmental studies with the Sensor Web: principles and pratice. *Sensors*, 5:103-117.

D'Eon, S.P., Magasi, L.P., Lachance, D. and DesRochers, P. 1994. *Canada's National Forest Health Monitoring Plot Network Manual on Plot Establishment and Monitoring.* Information Report PI-X-117, Petawawa National Forestry Institute. Canadian Forest Service. http://www.eman-rese.ca/eman/reports/publications/arnews/arnews.html

Falkner, J., Timney, J. and Panduranga, S.N. 2001. *Beginning JSP Web Development.* Wrox Press. 831pp.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J.M. 1994. *Design Patterns: Elements of Resuable Object-Oriented Software.* Addison-Wesley. 416pp.

Graniero, P.A. 2001. *The Effect of Spatiotemporal Sampling Strategies and Data Acquisition Accuracy on the Characterization of Dynamic Ecological Systems and Their Behaviours.* Unpublished PhD Dissertation, University of Toronto. 181pp.

Graniero, P.A. 2002. Project proposal on "Development of a mobile geospatial data acquisition/fusion system and real-time, wireless integration with a spatial database infrastructure"

Graniero, P.A. and Miller, H.S. 2003. Real-time, wireless field data acquisition for spatial data infrastructures. *Proceedings, GeoTec Event 2003*, Vancouver, BC. 9pp.

Graniero, P.A. 2006. Project proposal: An advanced telemetry system for complex environmental monitoring applications

Henderson, B., Dobbie, M., and Harch, B. 2005. *An Integrated Environmental Monitoring Program for Adelaide's Coastal Waters.* Technical Report No. 19, Stage 2 Research Program 2003-2005, Adelaide Coastal Waterls Study Task EMP I. South Australian Environment Protection Authority. 100pp.

Hill, J. and Culler, D. 2002. Mica: A wireless platform for deeply embedded networks. *IEEE Micro*, 22(6):12–24.

ISO. 1994. *Open Systems Interconnection Basic Reference Model: The Basic Model.* ISO/IEC International Standard 7498-1. International Organization for Standardization.

Jabeur, N., McCarthy, J.D., Xing, X. and Graniero, P.A. In press. A knowledge-oriented meta-framework for integrating sensor network infrastructures. *Computers & Geosciences.* 11pp. doi:10.1016/j.cageo.2008.04.006.

Jiang, X. and Camp, T. 2002. A review of geocasting protocols for a mobile ad hoc network. *Proceedings of the Grace Hopper Celebration of Women in Computing*. Vancouver, BC. 5pp.

Ko, Y.H. and Vaidya, N.H. 2000. Location-aided routing (LAR) in mobile ad hoc networks. *Wireless Networking*, 6:307-321.

Lane, A.M.J., Rennie, S.C., and Watkins, J.W. 2004. Integrated data management for environmental monitoring programs. In: Wiersma, G.B. (ed.) *Environmental Monitoring:* CRC Press. 37-82.

Lenzerini, M. 2002. Data integration: a theoretical perspective. *Symposium on Principles of Database Systems,* Madison WI. 243-246.

Messer, J.J. 2004. *Monitoring, Assessment, and Environmental Policy*. CRC Press.

Meyer, B. 1997. *Object-Oriented Software Construction*. Prentice Hall.

National Academy of Sciences. 1977. *A Report to the U.S. Environmental Protection Agency from the Study Group on Environmental Monitoring, Volume IV,* Washington, D.C.

Nickerson, B.G. and Lu, J. 2004. A language for wireless sensor webs. *Proceedings of the Second Annual Conference on Communication Networks and Services Research,* 8pp.

Parr, S.S. and Hirst, D.J. 1999. The UK Environmental Change Network and the Internet: their role in detecting and interpreting environmental change. In: Pykh, Y.A. (ed.) *Environmental Indices: System Analysis Approach*, EOLSS Publishers, Oxford, UK. 223-236.

Pressman, R.S. 1992. *Software Engineering: A Practitioner's Approach* (3rd ed.). McGraw-Hill.

Russinovich, M.E. and Solomon,D.A. 2005. *Microsoft Windows Internals: Microsoft Windows Server 2003, Windows XP, and Windows 2000* (4th ed.). Microsoft Press.

Schildt, H. 1998. *C++: The Complete Reference* (3rd ed.). McGraw-Hill Osborne.

Schildt, H. 2004. *The Art of C++*. McGraw-Hill Osborne.

Schindler, D.W. and Fee, E.J. 1974. Experimental Lakes Area: whole-lake experiments in eutrophication. *Journal of Fisheries Research*, 31(5): 937-953.

Tilak, S., Abu-Ghazaleh, N. and Heinzelman, W. 2002. A taxonomy of wireless micro-sensor network models. *ACM Mobile Computing and Communications Review*, 6(2):28-36.

Tubaishat, M. and Madria, S. 2003. Sensor networks: an overview. *IEEE Potentials*, 22(2):20- 23.

Ziegler, P. and Dittrich, K.R. 2004. Three decades of data integration - all problems solved? In: Jacquart, R. (ed.) *Building the Information Society: IFIP 18th World Computer Congress.* Springer. 3-12.

Vita Auctoris

Xitao Xing was born in 1970 in China. In 1991, he began studies in Surveying Engineering at the Xi'an College of Geology and gained a Bachelor of Engineering degree in 1995. Afterwards, he continued his study at the Wuhan Technical University of Surveying and Mapping, and was granted a Master of Engineering degree in 1998. He graduated with a Master of Science degree in Earth Sciences from the Department of Earth and Environmental Sciences at the University of Windsor in 2008.