A DATA HANDLING INTEGRATION BUS FOR WIRELESS SENSOR NETWORKS

Bo Dong and Eric Schendel
Faculty Advisor: Dr. Ahmed Mahdy
Department of Computing Sciences
Texas A&M University–Corpus Christi
6300 Ocean Drive, Corpus Christi, Texas 78412-2795
bdong@islander.tamucc.edu, eschendel@islander.tamucc.edu

ABSTRACT

This paper proposes a novel data-handling pattern for Wireless Sensor Networks. The proposed pattern uses the Enterprise Service Bus (ESB) structure to develop a new integration bus that handles a large amount of different data formats provided by the sensor devices. The proposed integration bus offers many benefits to a wide variety of wireless sensor network applications.

Keywords: Wireless sensor network, Enterprise Service Bus, Data handling

INTRODUCTION

The research on Wireless Sensor Networks (WSN) has been growing steadily. WSNs are widely used in environment monitoring, system tracking as well as underwater exploration. As WSN applications continue to be developed, new array of challenges are being introduced. Current research on WSN mainly focuses on the design of efficient communication and security protocols such as low-power secure routing algorithms. As WSN reaches new territories, the need for large-scale data handling cannot be underestimated.

In wireless sensor networks, the central issue revolves around the data provided by sensor nodes. WSN can be viewed as a two-layer system; a sensor device layer which can be deployed in different environments to gather data, and a client application layer which processes the collected data and interacts with the end user. At the sensor device layer efficient protocols, low-energy algorithms, effective architectures as well as other technologies are used to give an effective, efficient and accurate data gathering. At the client application layer graphical user interfaces and parallel computing algorithms have been used to provide the user with a powerful and convenient way to access sensor device data. However, the number of sensor nodes increases as the size of the network increases. This results in an increase in the amount of collected data and raises the need for efficient data handling techniques.

The remainder of this paper is organized in two sections. The first section discusses major data handling challenges. The next section illustrates the technology of an integration bus, and shows how it can be used in WSNs to resolve data handling challenges.

CHALLENGES IN DATA HANDLING

As mentioned earlier, data handling is very imperative to the proper operation of Wireless Sensor Networks. The bottleneck between the sensor device layer and client application layer limits the speed and capability of data transmission and handling. The major challenges are:

Challenge one: How to efficiently handle a large amount of data

In large-scale WSNs, client applications may receive massive data to reach their computing requirements. For example, in sensor location systems, SemiDefinite Programming (SDP) [1], a technique used to give accurate position information even in the presence of noisy backgrounds, is used to estimate the sensor nodes' position according to the distance between different nodes. In this algorithm, each node should provide at least three distances to neighbor nodes. If there are 3000 sensor nodes in the application, there will be at least 9000 position information which do not include other information that should be transferred to the client application layer.

Challenge two: How to furnish required data to different client applications

In large WSN applications it may be necessary to filter and minimize the amount of transferred data in order to keep the network load within adequate limits. It is more efficient if the sensors only transmit necessary data to client applications. For example, in monitoring system, there may be different client interfaces that focus on different aspects of the environment. In this case, the performance improves only if useful data is processed and communicated.

Challenge three: How to handle different data formats

The data gathered by sensor nodes is transferred to different client servers or applications. These different sensor severs might use different data formats and protocols, such as Extensible Markup Language / Hypertext Transfer Protocol (XML/HTTP) or Java Message System (JMS). Therefore, transferring the data in different formats to client applications presents a major challenge. Not only that but also how to provide the right data format to corresponding client applications is not straightforward.

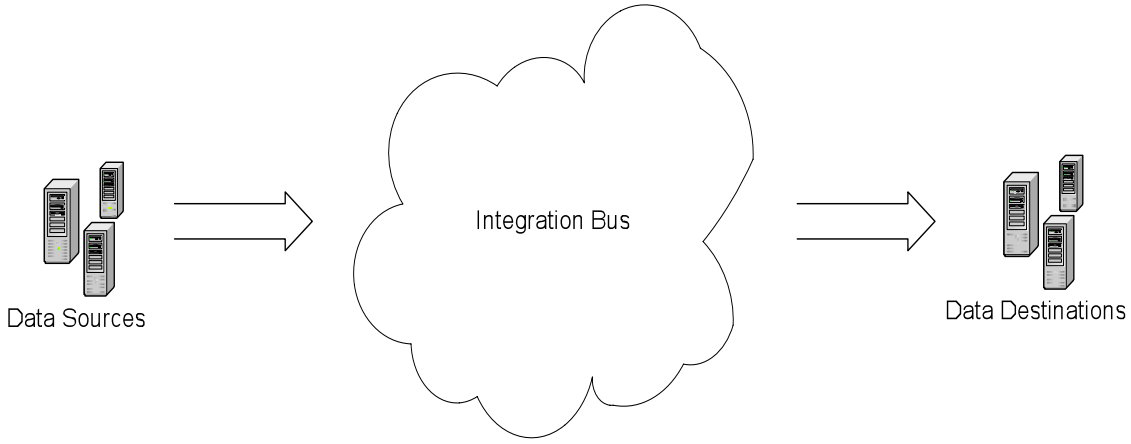
DATA HANDLING SOLUTION

There are three ways to solve these problems. First, it can be resolved by using heavy logic and computation circuits on the sensor node. Second, the communication process part maybe altered to provide a solution. Thirdly, data handling models can be added to each client application to resolve the data handling issues.

In most WSN applications power consumption is a vital design issue. If using the first method, logic decision and heavy computing will lead to more stored power consumption. Hence, it is not a suitable option. Alternatively, adding data handling models to each client application layer is not feasible either. This is because it is very expensive to add several models to many client applications that the system might be using. Also, each client application will have its own data requirements. The third option of adding data models to clients does not support that flexibility. Therefore, option two in which data is handled while it is being transferred from the device nodes to client applications is most feasible.

Integration Bus

The second method requires virtual alteration to the communication link between the sensor devices and client applications in order to handle the data. The integration bus presents the most viable option. The integration bus can be defined as a Message-Oriented Middleware (MOM) solution providing for a componentized scalable architecture. It utilizes an Enterprise Service Bus (ESB) architecture that provides integration capabilities based on a library of middleware services and components. All clients and services connect to this bus via standard protocols such as JMS/HTTP. Hence, a messaging based integration bus provides a cloud like abstraction to end applications by handling incoming messages via the various service entities it encapsulates. MOM also makes it possible to add intermediate services to filter, translate, and route data.



Enterprise Service Bus

An Enterprise Service Bus uses Message-Oriented Middleware (MOM) and standards such as JMS and Java Business Integration (JBI) to provide a peer-to-peer architecture where communication is accomplished through named message queues, which can be distributed across multiple hosts. Clients and services can be connected to the bus without any knowledge of the other clients and services. They merely send and receive messages.

An ESB comprises of a variety of services but is depicted as a single entity because the clients and services need not be aware of the details. Messages needed by a particular service are routed to that service and translated to the service's format by the bus. This is possible because an ESB is:

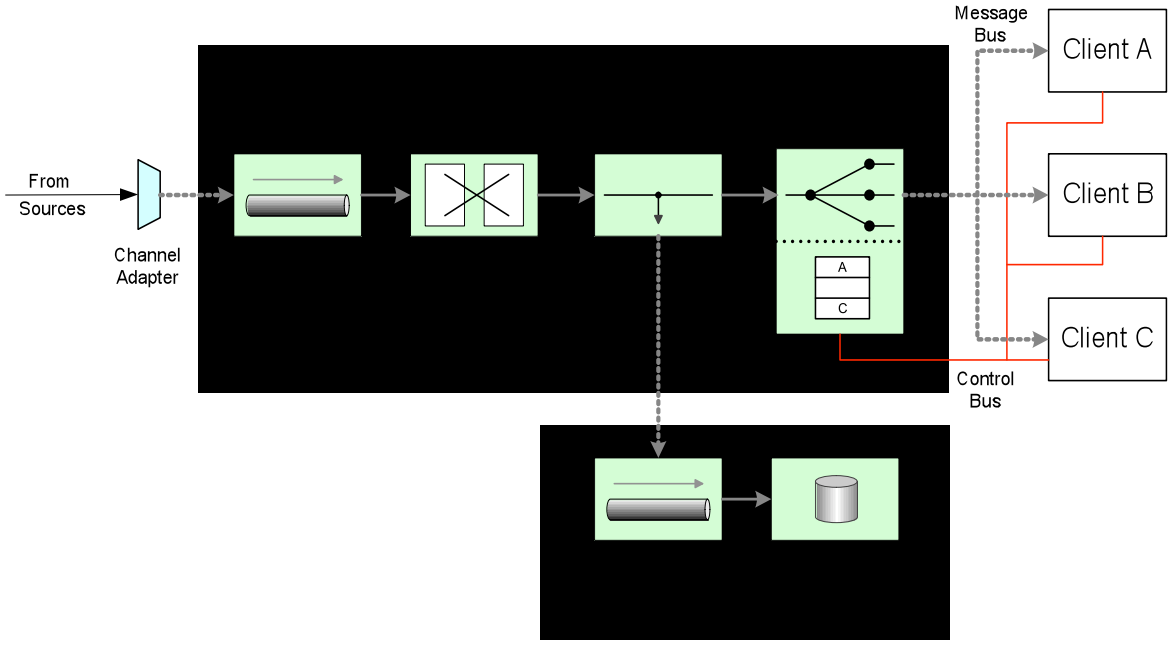
- *Standards-base*: such as JBI provides reusability and flexibility and XML allows easy data manipulation and routing.
- *Highly distributed*: loosely coupled services provide integration functionality such as data transformation services.
- *Remotely configured and managed*: automatic population of updates and configuration to services.

- *Extensible*: services require little coding due to many ready-to-use components such as routing, translation (e.g. Extensible Stylesheet Language Translator - XSLT), and data flow control (e.g. Business Process Execution Language - BPEL).
- *Scalable*: services can be load balanced across multiple servers.
- *Robust*: services can failover to a backup server.

For an ESB to support multiple producer and consumer systems, a canonical XML schema helps conform all the passing data on the bus. Data translators are used to transform data between a client or service format and the canonical format allowing clients to act just like a service since they will be almost completely decoupled.

How to Resolve Those Challenges Using Integration Bus

The integration bus is based on ESB for allowing remote clients to view and analyze the input from the connected sensor servers. The following figure describes the services necessary for the integration bus to support a large number of input sources along with disparate client application connections at any given moment. Also it provides the ability to archive messages for history discovery.



Enterprise Integration Patterns (EIP) illustrates the Integration and Message Store Services as they translate comfortably to ESB components. The components are as follows:

- *Channel Adapter*: allows producers to communicate with the ESB protocol such as XML/HTTP to JMS.
- *Translator*: converts external message formats to the canonical bus format.

- *Wiretap*: broadcasts traversing message to a secondary channel.
- *Dynamic Recipient List*: a control bus registers services for receiving data broadcasts via the message bus.
- *Message Store*: allows archiving mechanism for messages.

With this infrastructure, remote clients can connect or disconnect to the integration service at anytime for a real-time data stream from sensor devices. Also, the client can access archived messages via the message store service.

Message channel can efficiently handle large amounts of data because it "is a set of connections that enables applications to communicate by transmitting information in predetermined, predictable ways." [2] In this way, the large amount of data is divided into small amounts according to different kinds of information. Data requirements of each client are resolved by a dynamic recipient list. All the incoming messages are broadcasted to the client applications which are in the control list of the dynamic recipient list module. Therefore, the client application just changes the dynamic recipient list via the control bus when it requires a specific data stream. The Canonical Translator can handle different data format since it converts external message formats to the canonical bus format.

FUTURE WORK

Even though the integration bus is very beneficial to sensor data handling and controlling, it may bring other problems. Future work involves implementation of the discussed design to gauge if it is the best infrastructure. Also, this pattern needs to consider more factors of WSN applications. It should not only focus on large-scale long-term applications, but also consider real-time short-term applications. Another consideration is how WSN applications can provide an efficient way to develop client applications to control corresponding sensors. This will introduce more challenges than data handling. However, it still can be resolved using the same methodology. The objective of this research is to generate a uniform middleware for the data handling in WSN applications.

CONCLUSIONS

Wireless Sensor Network (WSN) applications are widely used in different kinds of situations and settings. Ability to process large amounts of data is essential for WSNs. This document described the different components of Enterprise Service Bus (ESB) which provides an architectural foundation for an integration bus. The afore described WSN Integration Bus provides a scalable and generic method to handle both, large amounts of data and different data formats. Applications of ESB defining an Integration Bus are not limited to WSNs. ESB can be applied to other areas of computing where data transference requires a multi-tier framework.

REFERENCES

[1] Biswas, P., Ye, Y.Y., Semidefinite programming for ad hoc wireless sensor network localization, *Proceedings of the third international symposium on Information processing in sensor networks*, (46-54), April 2004.

[2] Hohpe, G., Woolf, B., *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*, Addison Wesley, October 10, 2003.

[3]Griffith, S., Richards, N., *JBoss: A Developer's Notebook*, O'Reilly, July, 2005.

[4] Tatibana, C. Y., Montez, C., Oliverira, R. S., Real-time Dynamic Guarantee in Component-based Middleware, *Object and Component-Oriented Real-Time Distributed Computing, 2007. ISORC apos;07. 10th IEEE International Symposium*, (7-9), 214 – 221, May 2007.

[5] Srinivasan, K., Dutta, P., Tavakoli, A., Levis, P., Some Implications of Low-Power Wireless to IP Routing, *Proceedings of the Fifth Workshop on Hot Topics in Networks (HotNets V)*, 2006.

[6] Fowler, M., Rice, D., Foemmel, M., Hieatt, E., Mee, R., Stafford, R., *Patterns of Enterprise Application Architecture*, Addison Wesley, Nov. 05, 2002.