Development Platform for Heterogeneous Wireless Sensor Networks

Xu Chong

Information Technology Department, Far East Horizon Limited,
36 Floor, Jin Mao Tower, No 88 Century Avenue, Shanghai, China
zephyr.xu@gmail.com

Abstract. A universal platform is developed to provide several interfaces that every sensor can register or communicate with each other and platform (environment). To understand the operation of the platform, sensor nodes simulated by the Blackfin processor and PCs are used. This report presents the performance of the DSN based on an experiment in which different sensor nodes are connected to the platform, send/get data to/from platform and also can communication with each other. The results of this study demonstrate the principles and implementation of distributed sensor networks which can be extended to a complete and operational sensor management environment.

1 Introduction

Smart environments represent the next evolutionary development step in building, utilities, industrial, home, shipboard, and transportation systems automation. Like any sentient organism, the smart environment relies first and foremost on sensory data from the real world. Sensory data comes from multiple sensors of different modalities in distributed locations. The smart environment needs information about its surroundings as well as about its internal workings.

The challenges in the hierarchy of detecting the relevant quantities, monitoring and collecting the data, assessing and evaluating the information, formulating meaningful user displays, and performing decision-making and alarm functions are enormous [1].The information needed by smart environments is provided by Distributed Wireless Sensor Networks, which are responsible for sensing as well as for the first stages of the processing hierarchy. The importance of sensor networks is highlighted by the number of recent funding initiatives, including the DARPA SENSIT program, military programs, and NSF Program Announcements.

Y. Wang and T. Li (Eds.): Knowledge Engineering and Management, AISC 123, pp. 113–119.
springerlink.com

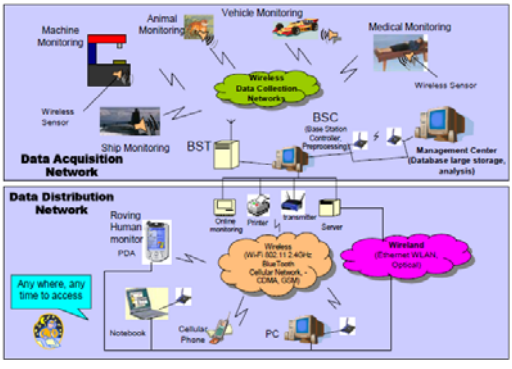

Fig. 1. Distributed sensor network [2]

The complexity of wireless sensor networks, which generally consist of a data acquisition network and a data distribution network, monitored and controlled by a management center.

2 Design and Implement Platform

Sensor networks are the key to gathering the information needed by smart environments, whether in buildings, utilities, industrial, home, shipboard, transportation systems automation, or elsewhere. To cite an example, recent terrorist and guerilla warfare countermeasures require distributed networks of sensors that can be deployed using, e.g. aircraft, and have self-organizing capabilities. In such applications, running wires or cabling is usually impractical. A sensor network is required that is fast and easy to install and maintain.

2.1 Basic Architecture

From a holistic perspective, a service oriented architecture (SOA)-based system is a network of independent services, machines, the people who operate, affect, use, and govern those services as well as the suppliers of equipment and personnel to these people and services. This includes any entity, animate or inanimate, that may affect or be affected by the system. With a system that large, it is clear that nobody is really "in control" or "in charge" of the whole ecosystem; although there are definite stakeholders involved, each of whom has some control and influence over the community [4, 5].

Instead of visualizing a SOA as a single complex machine, it is perhaps more productive to think of it as an ecosystem: a space where people, machines and services inhabit in order to further both their own objectives and the objectives of the larger community. In certain situations this may be a difficult psychological step for owners of so-called enterprise systems to take: after all, such owners may rightly believe that since they own the system they should also have complete control of it.

This view of SOA as ecosystem has been a consistent guide to the development of this architecture.

Taking an ecosystems perspective often means taking a step back: for example, instead of specifying an application hierarchy, we model the system as a network of

peer-like entities; instead of specifying a hierarchy of control, we specify rules for the interactions between participants [6]. The three key principles that inform our approach to a SOA ecosystem are:

• An SOA is a medium for exchange of value between independently acting participants;
• Participants (and stakeholders in general) have legitimate claims to ownership of resources that are made available via the SOA;
• The behavior and performance of the participants is subject to rules of engagement which are captured in a series of policies and contracts.

Summing up the above, the distributed sensor platform is implemented in service oriented architecture, the platform is considered as a service, a contractually defined behavior that can be implemented and provided by a component for use by another component.

2.2 Smart Sensors

Wireless sensor networks satisfy these requirements. Desirable functions for sensor nodes include: ease of installation, self-identification, self-diagnosis, reliability, time awareness for coordination with other nodes, some software functions and DSP, and standard control protocols and network interfaces [IEEE 1451 Expo, 2001].

There are many sensor manufacturers and many networks on the market today. It is too costly for manufacturers to make special transducers for every network on the market. Different components made by different manufacturers should be compatible. Therefore, in 1993 the IEEE and the National Institute of Standards and Technology (NIST) began work on a standard for Smart Sensor Networks. IEEE 1451, the Standard for Smart Sensor Networks was the result. The objective of this standard is to make it easier for different manufacturers to develop smart sensors and to interface those devices to networks.

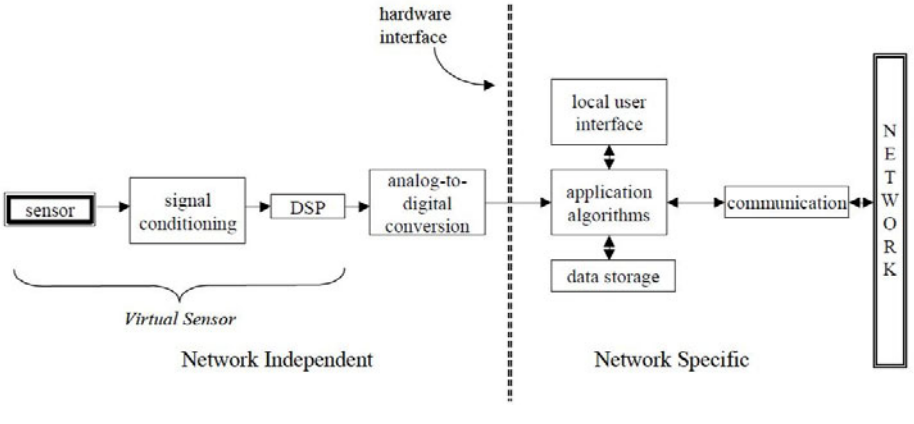

Fig. 2. A general model of smart sensor

2.3 Sensors for Smart Environments

Many vendors now produce commercially available sensors of many types that are suitable for wireless network applications. See for instance the websites of SUNX Sensors, Schaevitz, Keyence, Turck, Pepperl & Fuchs, National Instruments, UE Systems (ultrasonic), Leake (IR), CSI (vibration). The table below shows which physical principles may be used to measure various quantities. MEMS sensors are by now available for most of these measured.

Table 1. Measurements for Wireless Sensor Networks

	Measurand	Transduction Principle
Physical Properties	Pressure	Piezoresistive, capacitive
	Temperature	Thermistor, thermo-mechanical,
Motion Properties	Position	E-mag, GPS, contact sensor
	Velocity	Doppler, Hall effect, optoelectronic
Contact Properties	Strain	Piezoresistive
	Force	Piezoelectric, piezoresistive
	Torque	Piezoresistive, optoelectronic
Presence	Tactile/contact	Contact switch, capacitive
	Proximity	Hall effect, capacitive, magnetic, seismic,
Biochemical	Biochemical agents	Biochemical transduction
Identification	Personal features	Vision
	Personal ID	Fingerprints, retinal scan, voice, heat

2.4 Sensor Simulation

The distributed sensor network platform is aimed to connect enormous sensors and let them work together for the users. If two different kinds of sensors can work correctly on the platform, it is usually the case that more sensors can use the platform together, and we'll discuss the scalability of the platform below.

In this platform, sensor has several attributes:

ID: the id assigned by platform, NAME: the name of sensor,

DESCRIPTION: the description of the sensor,

COORDINATORID: id of the sensor's coordinator,

STATUS: show the status of sensor,

INITIALBATTERYLEVEL: the initial battery level of the sensor,

CURRENTBATTERYLEVEL: the current battery level of the sensor,

REGTIME: the time sensors start up.

For the purpose of simulating sensors, there are two sensors connect to the platform. One is called "Temperature_P" simulated by PC, as a temperature, it sends its data (temperature) to the platform every 20 seconds, at the same time, it want knows the temperature in other places which was collected by one another sensor called "Temperature_B" simulated by Blackfin-537., It get data of "Temperature_B" from platform every 10 seconds. On the contrary, the sensor on Blackfin will send data every 10 seconds, and get data from "Temperatrue_P" every 20 seconds.

"Temperature_P" Sensor:

When the sensor starts up, it will register to the platform and describe itself: "I work on PC, I send my collected data each 20 seconds, and get data each 10 seconds."

"Temperature_B" Sensor:

At the same time, also starts up the sensor on Blackfin, it does the same jobs as the PC sensor does. Its description is: "I work on Blackfin, I collect data each 10 seconds, and get data each 20 seconds."

3 Scalability Measurement

The scalability of the platform was tested by creating multiple threads used to stimulate multi-sensors connected to platform. The performance wasmeasured when the number of sensors is 1, 10, 20, 40…and so on and respectively calculated the time when the sensor does some jobs with the platform.

For instances, one sensor registers to the platform cost about 280ms, and 10 sensors register at same time, the minimum time is 310ms, and maximum time is 368ms, and the average time of ten sensors is 340ms.

This figure blew show the result of measurement that the transformation of time when sensors messaging and register platform with the increase of number. Here, the x axle is the number of sensors; y axle is the average time that the sensor costs when it finish the operation, unit is millisecond (ms). The pink line with circular point is the messaging line, and the purple line with rectangle point is the registration line.

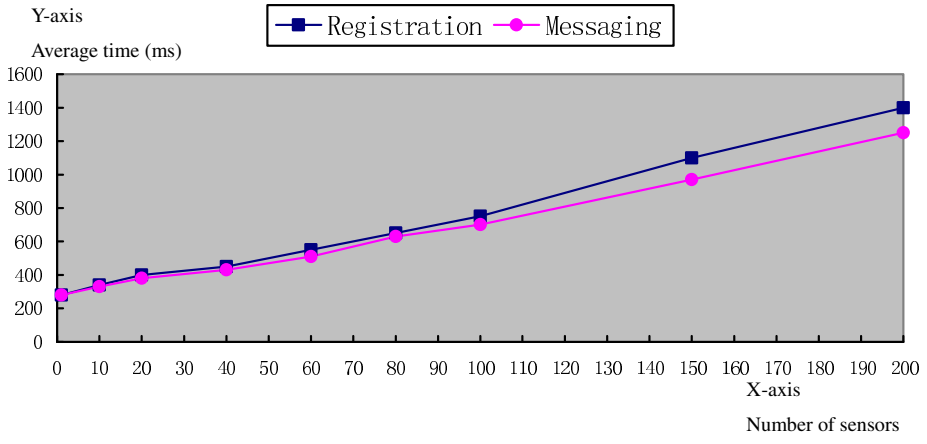

Fig. 3. Scalability Measurement

4 Performances Analysis

From the section 4.3, the performances of the platform can be seen that the time of communication between sensor and platform will increase linearly with the connected sensors enhanced when the number of sensors is not particularly large.

As to the web services, the platform process incoming requests by a process thread pool which within a group of threads. When the request comes, an idle thread in the pool will serve it. The process thread pool will not create an unlimited number of threads to handle a large volume of requests. Therefore, the communication time with the platform of the multi-sensor is larger than single sensors. It depends on the ability of server's CPU that deals with multi-threads jobs.

Is that possible to connect an infinite number of sensors to the platform? The answer is negative. The maximum number of sensors is approximately 500. In the test, when the sensors exceed 500, a few sensors will fail to communicate with platform, when the number is more than 600 or even much more than this, the most of sensors will announce that they can't invoke the service on the platform.

According the response time, from the figure 3, we know that platform will response sensor within 300ms if only a single sensor connected to the platform, but, with the growing number of connected nodes, time has also increased. When the number reaches 1000, the average response time will excess 6500ms, meanwhile, nearly half nodes doesn't get service.

If we used coordinator to manage a series of sensors, in order to send request sequentially, instead of every sensors send request to platform at same time, the system will more stable and efficiently. For example, a Blackfiin connect 1000 sensors as their coordinator, then the Blackfin sends all the registration request of sensors to the platform every 500ms, the platform response each request within the 400ms. That's the

advantage of using coordinator to manage sensors, so that we'll discuss coordinator in the next chapter.

Theoretically, every operation of sensor will cost same time because every service in the platform is equal. However, in our implement, it can be seen in the figure 3, the registration takes a little more than the operation of send message. This is due to the database operation, obviously, the register operation in the platform does more work than the sensor send message to the platform.

If server creates threads infinitely, all resources on the server can only be used to manage these threads. By limiting the number of threads can be created, we can make thread management overhead to maintain in a controlled level. If a request arrives for all threads in the thread pool are occupied, then the request will be queued up, only after busy thread completes his task, the idle threads can process the new request. This method is actually more effective than switching to a new thread, because you do not need to be carried over from request to request threads switching. But the problem is, if the thread efficiency is not high (especially in busy server), the waiting request queue will become large.

5 Recommendations for Further Work

Due to the limitation of the time of research, this project has limitations that require some improvement or need to be solved in the future. For instance, the platform should be extended to connect more types of sensors which are implemented by other hardware and other environment. Research on the security aspect of data transmission between the sensors and the platform is also needed.

As discussed in section 3, the number of nodes that connected to the platform has a limitation. Therefore, it is efficient that use coordinator nodes to manage number of sensors, it will improve the performance of platform greatly.

References

1. Altman, E., Basar, T., Jimenez, T., Shimkin, N.: Competitive routing in networks with polynomial costs. IEEE Trans. Automat. Control 47(1), 92–96 (2002)
2. Lewis, F.L.: Smart Environments: Technologies, Protocols, and Applications. In: Cook, D.J., Das, S.K. (eds.). John Wiley, New York (2004)
3. Bronson, R., Naadimuthu, G.: Operations Research, 2nd edn. Schaum's Outlines. McGraw Hill, New York (1997)
4. Bulusu, N., Heidemann, J., Estrin, D., Tran, T.: Self-configuring localization systems: design and experimental evaluation. In: ACM TECS Special Issue on Networked Embedded Computing, pp. 1–31 (August 2002)
5. Cao, J., Zhang, F.: Optimal configuration in hierarchical network routing. In: Proc. Canadian Conf. Elect. and Comp. Eng., Canada, pp. 249–254 (1999)
6. Chen, T.-S., Chang, C.-Y., Sheu, J.-P.: Efficient path-based multicast in wormhole-routed mesh networks. J. Sys. Architecture 46, 919–930 (2000)