Agent based System Architecture for Wireless Sensor Networks

Sajid Hussain, Elhadi Shakshuki, Abdul Wasey Matin  
Jodrey School of Computer Science  
Acadia University  
Wolfville, Nova Scotia, Canada B4P 2R6  
{Sajid.Hussain; Elhadi.Shakshuki; 073720m}@acadiau.ca

Abstract
This paper reports on an ongoing research on developing an agent based system architecture for wireless sensor networks. The main objective of this system is to facilitate the design, implementation, and maintaining applications for sensor networks that consist of wireless sensor nodes. The proposed system architecture contains the following four types of agents: interface, regional, cluster, and query agents. The interface agents receive queries from the users and return the results back to them. The regional agent is responsible for sending query packets to the cluster agents. The cluster agents disseminate query into the network and provide efficient in-network processing. Finally, the query agents process the query in the sensor node. The delegation of specific tasks to different software agents facilitates the development and maintenance of sensor network applications.

Keywords: Agents, wireless sensor networks, ubiquitous.

1. Introduction

Wireless sensor networks (WSN) enable pervasive, ubiquitous, and seamless communication with the physical world. A few common applications are military, security, habitat monitoring, industrial automation, and agriculture. WSN comprises numerous sensor devices, commonly known as motes that monitor the physical entities such as temperature, light, motion, metallic objects, and humidity [1]. The sensor motes are spatially scattered over a large area. Since the deployment environment could be harsh, hostile, inaccessible, and remote, the network must possess easy deployment and maintenance free strategies.

Although wireless sensor networks appears similar to traditional wireless networks, because of low battery constraints and the large size of WSNs, different mechanisms for routing, transport, and end-to-end communication are needed [2]. Since a sensor mote acts as a source of data, WSNs can leverage from the principles of distributed databases [3, 4, 5]. Moreover, as each mote has sufficient computing, storage, and communication resources, motes can be treated as autonomous agents that can create self-organized clusters by coordinating their activities and collectively work together for achieving their goals [7]. Thus it would be useful if the key roles are identified and assigned to different types of agents based on their functionalities. Hence, an agent-based architecture will assist in understanding and designing the application specific system.

As the precise awareness of mote characteristics would be beneficial in appreciating the architecture and design of WSN, this paper illustrates the commonly used mote. Currently, most of the motes are developed using open source initiatives; one of the commonly used mote platform is mica [6].

In mica platform, mote, sensors, and programming board are designed on three distinct boards. Motes (e.g, Crossbow’s mica2, mica2dot, and micaz) are programmed by a programming board, MIB510 or MIB600, which uses RS-232 serial interface. The sensor data is obtained by attaching a sensor board, MIB300 or MIB310, to the mote. Since MIB600 is an Ethernet gateway and implements TCP/IP protocol, it can be a bridge between a sensor network and the Internet.

Since the ratio of sleep power to active power is only 1:200, the battery life can be increased by designing protocols and applications that have a small duty cycle. For radio transmission, the older motes, mica2 and mica2dot, operate under non-standard protocols at 315/433/916 MHz.

2. Agent-based Architecture

The proposed agent based system architecture is based on hierarchical software agents and classified into different types of agents based on their functionalities [8], as shown in Figure 1. The sensor network is divided
into several regions due to spatial, topological, and deployment conditions. Moreover, each region is divided into several clusters based on queries, motes hardware, and data acquisition mechanisms.

The architecture is based on a four layer hierarchy of software agents. At the interface layer of the architecture, an interface agent interacts with the sensor network. At the regional layer, the regional agent acts as a gateway between the sensor network and the traditional networks. At the cluster layer, the cluster agent performs query dissemination and efficient in-network processing. Finally at the query layer, the query agents which are located on the data collecting sensor nodes perform data acquisition and local computation.

2.1 Interface Agent

The interface agent is front-end of the system and responsible for accepting the user requests and processing them. The interface agent provides the interface or services that are used to query the sensor network. These agents are located on traditional devices such as desktops and PDAs. The design of the interface agent does not depend on the size and complexity of the sensor network, but it is based on the application and user profile.

A user starts the interaction with the system by submitting a form-based query. When the interface agent receives it, the agent maintains the connection with the user (i.e. maintaining the HTTP connection with the user’s web browser).

The interface agent contains three components: receiving (RCV), request handler (RQH), and service (SRV) component. Figure 1 shows an interface agent that contains RCV and SRV components, and r number of RQH components. The receiving component is a daemon process that receives all user requests at a well-known TCP/IP port. The request handler component processes the user requests. It also formats the query result for the user. For each incoming user request, an instance of a request handler is created. The request handler component is the process that maintains the TCP/IP connection with the user application. The service component interacts with the regional agents. The service component sends the queries to the desired regional agents; the results received from the regional agents are combined and forwarded to the appropriate request handler.

It is possible that the design of interface agent can have different forms based on the application. For example, applications where queries are submitted by different types of users, an XML based interface agent may be useful.

2.2 Regional Agent

A sensor network consists of numerous sensor nodes that could be distributed in a vast region like a forest, a farm, a parking lot, or a university campus. Depending on the geographical conditions and sensor node density, a sensor network is divided into several regions and each region is managed by a regional agent. For example, a sensor network of n sensor nodes, \{S_1, S_2, ..., S_n\}, is divided into m disjoint regions, \{R_1, R_2, ..., R_m\}. The regional agent is equipped with a rich power supply or a direct power connection. The node of the regional agent is commonly known as a base station.

The regional agents are statically assigned based on network topology and deployment conditions. The regional agent consists of two components: external and internal. As shown in Figure 1, each regional agent contains external and internal components. The external component interacts with the traditional networks using TCP/IP and the internal component interacts with the sensor network using radio based active messages. The communication between external and internal components is achieved by serial communication.

The regional agent can be deployed by several techniques. For example, a PC that is attached with a mote, an Ethernet-programming board, MIB600, or a
2.3 Cluster Agent

For query dissemination and efficient in-network processing, a region of a sensor network is divided into several clusters. Each cluster is managed by a cluster agent.

A cluster agent's main functionality is to receive query packets from the external component of the region and retransmit the query packets to the desired sensor nodes. The cluster agent keeps track of the types of sensors available in each node and the current energy levels of the associated nodes. The query dissemination plan is generated based on the type of query, available sensors, and energy levels of the associated sensor nodes.

Each sensor node must be registered with a cluster agent; moreover, each sensor node belongs to exactly one cluster. When a new sensor node is added, the new sensor node broadcasts a message to register with a cluster agent. One or more cluster agents broadcast a confirmation message with their latency and energy levels. A cluster agent that has the optimum latency and energy level, it announces itself as the cluster agent for the new incoming sensor node. The new sensor node stores the address of the associated cluster agent.

A sensor network of \( n \) sensor nodes, \( \{S_1, S_2, \ldots, S_n\} \), is divided into \( m \) regions, \( \{R_1, R_2, \ldots, R_m\} \), and each region is divided into \( p \) clusters, \( \{C_{i_1}, C_{i_2}, \ldots, C_{i_p}\} \). Figure 1 shows that each region contains \( p \) number of cluster agents.

Cluster agents are appointed based on the energy consumption. For example, if the energy level of the current cluster agent falls below a certain threshold value, the role of a cluster agent will be delegated to another sensor node. Meanwhile, when the energy falls below the threshold value a broadcast message for new cluster agent is transmitted. In this case, the sensor nodes capable of becoming the cluster agent and then broadcast their energy levels, and the sensor node that has the most optimum energy level becomes the cluster agent. Thus, the new cluster agent broadcasts the declaration message to confirm its designation as new cluster agent. Finally, the sensor nodes associated with the older cluster agent update the address of their associated cluster agent.

It should be noted that the threshold value must be large enough to accommodate the broadcasts that are required to transfer the role of the cluster agent. In each cluster, although not every sensor node is capable of becoming a cluster agent, there can be several candidates for a cluster agent.

2.4 Query Agent

Each sensor node has a query agent that is responsible for the following tasks: acquisition of sensor data, filtering of inaccurate and unwanted data, aggregation and processing of useful data, and transmission of the desired results. Figure 1 shows that each cluster contains \( z \) number of query agents.

The query agent processes the query packet that is transmitted by the associated cluster agent; the query packets received from other cluster agents are immediately rejected.

The sensor data is collected depending on the query. The query agent is responsible to optimize the query processing for the sensor node. The sampling interval can be adjusted according to the expected accuracy, current energy level, type of query, and type of application.

3. Design Processes

The proposed agent-based architecture is used to design a sensor application that monitors temperature, light, and mobility in different buildings of the campus at Acadia University, Wolfville, Nova Scotia.

The sensors are placed in four buildings: Carnegie Hall (CAR), Horton Hall (HOR), Huggins Science Hall (HSH), and K. C. Irving Environmental Science Centre (IRV). Figure 2 shows the four buildings as rectangular boxes; for vividness, the sensors are not shown in the picture.

The four buildings, where sensors are deployed, are divided into two regions: region R1 includes CAR and IRV, and region R2 includes HOR and HSH; the regional agents are deployed in CAR and HSH. The interface agent is deployed on a PC in CAR, although it can be deployed in any building. The receiving component (RCV) of the interface agent is managed by a web server.
implemented in Java. For each incoming user request, an instance of RequestHandler (RQH) component is created in a separate thread. Finally, the service component (SRV) is implemented as a Java RMI server; for more details please refer to [9]. Although the web server is deployed on a PC in CAR, the data being served is not a local data; the data is obtained from the sensors.

Figure 2: A deployment of a wireless sensor network in Acadia University.

Figure 3 shows different types of messages that are exchanged between different entities in the environment. As the web server is a multi-threaded concurrent server, for each incoming user request, an instance of a request handler is created; the request handler sends the user request to the service component (SRV). The request handler remains in a block state until the query result arrives from SRV. When the response arrives from SRV, the message is formatted for the user application and the result is returned to the user.

The service component (SRV) forwards the user request to all regional agents. The request can be forwarded by using any standard distributed computing paradigm, such as, RMI, SOAP, or sockets. In our work, Java RMI service is used to forward the requests from the interface agent to the regional agents. The service component (SRV) of the interface agent is deployed as an RMI server; the service component is available at rmi://131.162.135.56:9090. Each regional agent contains an instance of a callable RMI client. The RMI clients are registered with the RMI server, interface agent.

Figure 3 shows that the service component invokes the query function at both regional agents. The request handler component of the interface agent combines the partial results obtained from the regional agents and the query result is sent to the user application (i.e. a web browser).

In each region, mica2 is connected to a PC which is used to deploy the service of a regional agent. A regional agent consists of an external and internal component, as shown in Figure 1. The external component allows the regional agent to interact with the service component of the interface agent and the internal component allows the regional agent to interact with the sensor network. In our case, the external component is a callable RMI client and the internal component is a TinyOS component that is programmed on the mica2 mote. Since the interface agent sends the user request to all the regional agents, the user request is asynchronously sent to two external components. As mica2 mote is used, the external and the internal components pass the message back and forth using the serial forwarder mechanism provided in the TinyOS environment.
If Ethernet gateway programming board, MIB600, is available, the implementation of a regional agent can be transferred from a PC to MIB600. In that case, a PC is not required to deploy a regional agent. Moreover, if Crossbow’s stargate gateway is available, both the interface agent and regional agent can be transferred to a stargate gateway. In other words, a complete sensor network application can be deployed using a stargate gateway and a few MIB600 Ethernet gateway programming boards.

The query packets created by a regional agent are disseminated into the corresponding region of the sensor network. For example, the query packets created by R1 are sent to the sensors located in CAR and IRV. In each region, cluster agents are used to disseminate the query and every sensor is registered with a cluster agent. In our experiment, seven cluster agents are deployed in region 1, as shown in Figure 4. Four cluster agents are deployed in CAR, one for each floor of the building. For IRV, three cluster agents are deployed on the main floor; the agents are deployed in lobby, library, and botanical gardens.

The cluster agent is assigned the responsibility of retrieving the query results from the associated sensors. The partial results received from all the cluster agents are combined using the internal component. When all the necessary results have arrived, the internal component combines them and submits the result to the external component, which accordingly sends the result to the interface agent. Each cluster agent creates a query execution plan for its associated sensors. Depending on the type of query and the desired accuracy level, the cluster agent creates a query execution plan.

4. Conclusion

This paper presented our proposed approach to agent based system architecture, which identifies the important software agents that facilitate in design and implementation of sensor network applications. The architecture reduced the overall message communication; thus the energy consumption is reduced.

For the future work, our plans involve completing the implementations and investigating different techniques for the election of cluster agent and compare them. We are also planning to incorporate the energy level of the cluster agents in the query dissemination plan.

References: