Data aggregation in wireless sensor networks using the SOAP protocol

2007 J. Phys.: Conf. Ser. 76 012039

(http://iopscience.iop.org/1742-6596/76/1/012039)

Data aggregation in wireless sensor networks using the SOAP protocol

Adil Al-Yasiri and Alan Sunley
School of Computing, Science and Engineering, University of Salford, Greater Manchester, M5 4WT, UK

E-mail: a.al-yasiri@salford.ac.uk

Abstract. Wireless sensor networks (WSN) offer an increasingly attractive method of data gathering in distributed system architectures and dynamic access via wireless connectivity. Wireless sensor networks have physical and resource limitations, this leads to increased complexity for application developers and often results in applications that are closely coupled with network protocols. In this paper, a data aggregation framework using SOAP (Simple Object Access Protocol) on wireless sensor networks is presented. The framework works as a middleware for aggregating data measured by a number of nodes within a network. The aim of the study is to assess the suitability of the protocol in such environments where resources are limited compared to traditional networks.

1. Introduction

In recent years Wireless Sensor Networks (WSN) have emerged as research tool and solution for a number of applications [1] as they incorporate technologies from three different research interests: sensing, communication, and computing [2]. A WSN itself can consist of merely a handful of nodes or potentially thousands that can spread from a few meters in range or cover extremely vast areas, within or surrounding an area of interest. Although the exact positioning of nodes can be preplanned, one of the benefits of WSN is that deployment can be dynamic. For instance it is not always feasible for designated deployment in disaster areas or war zones, where nodes may be dropped from aerial positions. It is for this reason that sensor nodes are designed to be small and easy to deploy in a target area [3]. Additionally, due to the nature of the deployment, nodes are intended to be disposable, so unlike traditional wireless devices such as mobile phones or PDAs the power supply unit cannot always be recharged over its operational lifetime [3]. Thus energy efficiency and preservation is a major research issue for both the design of the physical device itself and the software that runs on it. A node within a WSN is, in most cases, capable of performing sensing, data processing and communication, these are known as “smart sensors” [4]. These sensors are physically composed of electronic sensing circuitry, a processor and a wireless transceiver, plus a power supply unit (battery).

The Simple Object Access Protocol (SOAP) is a standard web service middleware protocol for exchanging messages in distributed applications. It is a protocol based on eXtensible Markup Language (XML), and being purely text-based means it can be easily integrated with different programming languages and platforms. For this reason it is an effective communication protocol for distributed systems. SOAP is however designed for traditional computer networks and the limited

resources of sensor nodes and the ad-hoc nature of such architectures present a problem when developing middleware for WSN. This paper discusses these issues and then detail the design and implementation of a SOAP-based framework on WSN.

2. Data Aggregation and Middleware in Wireless Sensor Networks

Sensor networks are distributed event-based systems that focus on simple data-gathering applications and operate notably differently from that of traditional computer networks [5]. The gathered data can be made accessible to other nodes, including a specialised one (called the *sink*) through a variety of means. One potential method is for the data, collected by individual nodes, to be sent to the sink where it is stored; other nodes can then access that data when required. This is essentially point-to-point communication in which data aggregation is not used, and potential problems arise with efficiency due to the volume of traffic around the sink causing significant power loss [7]. It is also not feasible in large WSN were limited transmission range of nodes mean that multi-hops are required to convey data to a sink. Data aggregation can be used to overcome this problem. Instead of sending each message to a sink directly, a node can delay sending data until it has received messages from surrounding nodes. The node can then perform computation on the aggregated data and send the result to the sink. This distributed system makes efficient use of available resources by spreading computation across the network and only forwarding one message between nodes with aggregated data, reducing communication and energy overheads. Figure 1 below displays an example of a distributed WSN where nodes connected communicate with each other and aggregated data ultimately ends at the sink. Sinks can also act as controlling nodes, or "cluster heads" [8].

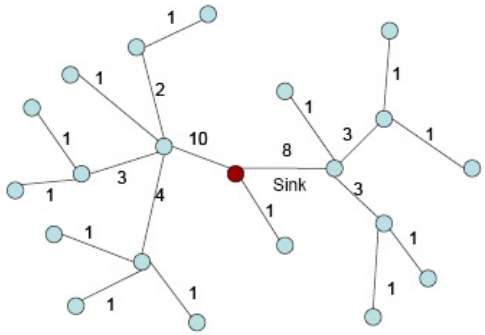

Figure 1. WSN node topology example, from [6].

In a data aggregation model, sensor nodes do not act merely as data sensing and forwarding points. Instead, they make decisions whether the information should be stored for future use, relayed as they are or modified by applying computation and aggregation with other data.

Existing middleware systems that are used in traditional computer networks such as CORBA or LIME could potentially be applied to sensor networks although several problems arise when doing so. Firstly, by providing individual access via objects, energy savings made using data aggregation are not present [9]. Secondly, the actual resources of sensor networks restrict the type of software that can be run across it. Thirdly, as nodes typically have limited resources, the middleware is required to be lightweight [10].

3. A SOAP-based Middleware for Wireless Sensor Networks

Much of the middleware for WSN has been designed for specific applications meaning widespread use is limited. As the deployment of WSN does become a more attractive option there will be a need for sensor nodes of different types and applications to communicate with each other. With a standardized general-purpose platform, multiple autonomous applications will be able to communicate across the same WSN [11]. Such interoperability is achieved in web services where the simplicity, robustness and extensibility of SOAP are exploited [12]. With this in mind it is natural to consider a web services

approach to WSN. On the other hand, SOAP is a protocol that is not explicitly designed for use in WSN and as a text-based protocol comes with a certain amount of overhead. As a result, it consumes network bandwidth and processor time. For these reasons a new framework has been developed to facilitate the use of SOAP in WSN that takes into consideration reducing the overhead on the network bandwidth.

3.1. A SOAP-based Framework

In this framework, there are three basic interactions between the nodes that form the wireless sensor network. The basis for each interaction is a remote procedure call and consists of:

1. A node dispatching a hello message to sinks.
2. A sink sending a Remote Procedure Call (RPC) to registered nodes.
3. Nodes responding to the RPC.

In the first case no return value is expected as the hello message simply invokes an operation within the sink. As it is a notification, reliability is not critical. The RPC's are similarly asynchronous, that is while the sender may expect a reply it does not wait. Meaning the sender can continue working immediately after the invocation is dispatched. This way the sink and node are sufficiently decoupled. Being deployable in an ad-hoc manner a WSN may find itself in conditions that result in inherently unreliable communication links and it is potentially inefficient to block a process while waiting for replies. For this reason this functionality is left out of the middleware, and so is up to application developers to implement.

Using the Observer pattern, there are two main components that make up this framework. A Requestor which constructs a remote invocation from parameters that are passed to it, and an Invoker which reads a request sent to it from a Requestor and de-marshals it to obtain the original parameters. This process is summarised in Figure 2.

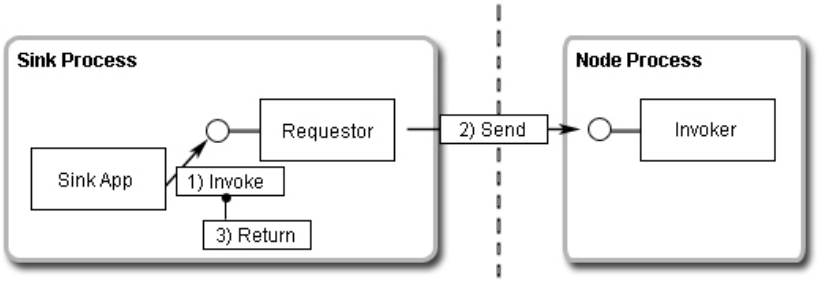

Figure 2. SOAP-Based Framework

The sink application invokes a request (1). When a sink application sends a request a Requestor sends the invocation over the network and immediately returns control to the calling application (2). On the node side of the middleware the Invoker differentiates between those operations that require a response and those that do not. For a synchronous operation a reply is sent even if it has no return value (3). In order to reduce the amount of overhead on the network, the framework allows requests to be modified or filtered upon arrival at the node by applying the following procedure:

1. The invoker node makes several readings and stores the values locally. No data is transmitted on the network at the time of data reading (sensing).
2. The requestor node may choose to select any of the values stored at the invoker node. It may also ask for some pre-defined filtering on the data; such as averaging the data.
3. The result may be returned back to the requestor or stored at the invoker for future reference. This is an adaptive pull strategy as opposed to the traditional push strategy. In our adaptive pull strategy, the requestor may make a request and demand a result (request-pull) or request a calculation with no result needed (request-save).

4. The adaptive pull strategy makes it possible for data aggregation to take place either at the invoker node or at the requester node.

3.2. Simulation of a Data Aggregation Scenario

The scenario used ito test and analyse the SOAP-based framework is that of a fire detection system. Each node within the network will handle four items of data, these are the node ID, along with temperature, humidity and smoke readings. When a node is activated it will register itself with data sinks within the network, after which data sinks periodically send a RPC to registered nodes in order to gather readings. Upon receiving the data the sink applies a formula to each value and is then able to calculate the likelihood of a fire incident:

Temperature (integer):	t = (Temperature + 1) * 0.875
Humidity (integer):	h = 15 - ((Humidity + 1) * 0.15))
Smoke (boolean):	s = If yes +50%, else +0%
Percentage chance of fire:	t + h + s

The higher the temperature and the lower the humidity the greater chance a fire incident is taking place, the detection of smoke, a boolean value, swings the decision be 50%. For instance if the following values were applied to the above formula; Temperature = 20^{o}C, Humidity = 60%, Smoke = 'Yes':

Temperature:	(20 + 1) * 0.875 = 18.375%
Humidity:	15 – ((60 + 1) * 0.15)) = 5.85%
Smoke:	'Yes', thus 50%
Chance of fire:	18.375 + 5.875 + 50 = 74.25%

The scenario has been simulated using the Network Simulator NS2, which offers considerable capability for simulating and analysing wireless networks. Through simulation, the middleware is tested in varying network conditions.

NS2 is flexible enough to specifically simulate a WSN. We simulated an actual phenomenon using a node broadcasting on a separate channel. When a sensor node detects a transmission in this channel it signals to the node that this physical phenomena has occurred in range of the sensor. Essentially it provides an event-driven detection mechanism. The sensor nodes themselves are configured to the physical properties of WAVELan sensor nodes defined and implemented in NS2 by SCADDS [13]. Taken from this are the configuration parameters that enable nodes to act as a 914 Mhz Lucent WaveLAN DSSS interface.

4. Advantages and Challenges of Using SOAP in WSN

The simulation work done in this project has shown that the use of SOAP, as a protocol for data aggregation, places some overhead on the network resources. Firstly, the processing time of a SOAP request delivered from a sink to a node is examined. For this scenario one sink and five nodes were created. Over the 520 second period of the simulation approximately 5000 SOAP messages from the sink where processed by the node. The average processing time was calculated, which revealed that processing times for SOAP messages are unsurprisingly higher, for the most part. Moreover, the processing times of SOAP messages spike more wildly, as shown by the standard deviation in table 1.

Figure 4 shows the round trip delay times of a remote invocation originating from the sink, and ultimately with the response being processed at the sink. This is the simple test with only one sink and one node operating in perfect network conditions. As figure 4 indicates, the round trip times of SOAP messaging are roughly twice as long as transmitting and processing raw, unformatted data.

Table 1 Processing time results

Approx. 5000 messages processed	Average processing Time (microseconds)	Standard Deviation
Raw Data	155.591	319.315
SOAP Messages	810.36	2104.95

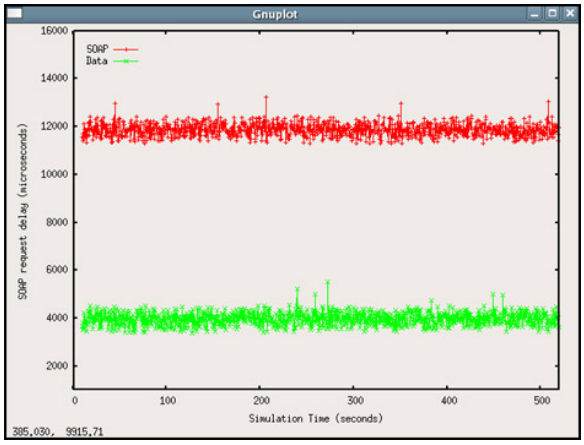

Figure 4. Comparison of SOAP and raw data messages sent to single node

We also measured the dissipated energy with relation to the number of packets sent. Figure 5 displays the average dissipated energy per message being sent. With the average energy dissipation of SOAP messages being roughly three times that of raw data messages, and a greater deviation from the average. The figure shows the total amount of energy dissipated on a single node while sending messages, The results show a steeper rise in dissipated energy when comparing SOAP and data messages.

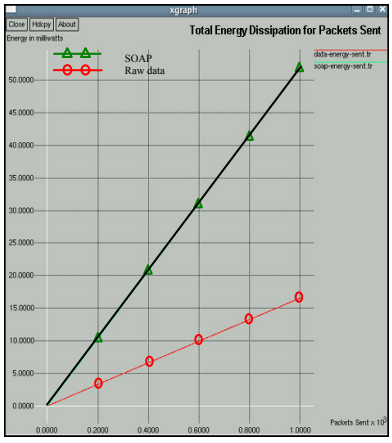

Figure 5. Total energy spent after sending approx. 1000 packets

Clearly, from the above results, it can be concluded that using SOAP, as a protocol for data aggregation, poses some challenges on the design of the framework. It must be noted, though, that these results show the worst case scenario, by comparing the overhead of using SOAP to sending raw data without using any middleware. However, there are some advantages of using SOAP as a data aggregation protocol, which may be summarised as follows:

1. SOAP significantly simplifies the software architecture for data aggregation.
2. It uses a simple textual based protocol which may be used as a framework for aggregating and filtering data at the receiving or sending nodes.

3. The framework reduces the amount of messages transmitted over the network by employing a synchronous protocol; albeit, that a single message has its own overhead. Overall, the network overhead is manageable.
4. The framework allows computation to be distributed among the network nodes, thus allowing better usage of the available resources. This is achieved by allowing data aggregation to be done at the requestor or the invoker nodes.

5. Conclusions

In this paper, a framework was presented for using SOAP as a protocol for data aggregation in wireless sensor networks. The results have shown that SOAP places an amount of overhead on the network while individual messages are sent between the sender and receiver. However, it also shows that employing a synchronous model to the communication between nodes has simplified the protocol and made savings on the overall transmitted messages. Messages are only sent when needed by the network. This is done by employing an adaptive pull strategy as opposed to the traditional push strategy. The adaptive pulling strategy allows results to either be requested and returned (request-pull) or requested and saved (request save).

References

[1] Yu Y, Krishnamachari B and Prasanna V K, 2004, Issues in designing middleware for wireless sensor networks, *IEEE Network Magazine Special Issue, Issue 1, 18.*
[2] Martinez K, Hart J K and Ong R, 2004, Environmental sensor networks, *Comp. 37 pp 50-6.*
[3] Khemapech, Duncan I and Miller A, 2005, A survey of wireless sensor networks technology, in PGNET, *Proceedings of the 6th Annual PostGraduate Symposium on the Convergence of Telecommunications, Networking & Broadcasting, EPSRC.*
[4] Lewis F L 2004 *Wireless sensor networks, smart environments: technologies, protocols, and applications* (John Wiley & Sons).
[5] Krishnamachari B, Estrin D and Wicker S 2002, The Impact of Data Aggregation in Wireless Sensor Networks, *Proc. of the 22nd Int. Conf. on Distributed Computing Systems.*
[6] Akkaya K and Younis M, 2005, A survey on routing protocols for wireless sensor networks, *Ad Hoc Network Journal pp.325-49.*
[7] Li S, Lin Y, Son S H, Stankovic JA and Wei Y 2003, Event detection services using data service middleware in distributed sensor networks, *Proc. of the 2nd Int. Workshop on Information Processing in Sensor Networks.*
[8] Akkaya K and Younis M, 2005, A survey on routing protocols for wireless sensor networks, *Ad Hoc Network Journal* pp 325-49.
[9] Heinzelman W B, Murphy A L, Carvalho H S and. Perillo M A, 2004, Middleware to support sensor network applications, *IEEE Network* 6-14.
[10] Römer K, 2004, Programming paradigms and middleware for sensor networks, *GI/ITG Workshop on Sensor Networks (Karlsruhe)* pp 49-54.
[11] Fok C L, Roman G C and Lu C, 2004, Mobile agent middleware for sensor networks: an application case study, *Technical Report WUCSE-04-73* (Washington University, Department of Computer Science and Engineering, St. Louis).
[12] Chiu K, Govindaraju M and Bramley R, 2002, Investigating the limits of SOAP performance for scientific computing, *Proc. of 11th IEEE Int. Symposium on High Performance Distributed Computing* pp 246-254.
[13] SCADDS, Scalable Coordination Architectures for Deeply Distributed Systems, http://www.isi.edu/scadds/.