ISSN 2047-3338

A Survey of Wireless Sensor Network- Software Architecture Design Issues

Zaheer Aslam, Nauman Qamar, Noor Khan, Shehzad Rizwan, Kamal Ahmad, Noor Zaman, Shafiullah and Rashid Zubair
Gandhara University of Sciences, Peshawar, Pakistan

***Abstract*– Since Wireless Sensor Networks (WSN) formulate the recent research into some advances to enable progression in the field of software and hardware technologies. Hence, in this research work, we study the performance and behavior of wireless sensor networks (WSN) in terms of effectiveness of system requirements on the resulting software architecture. As WSN currently faces a challenge due to rapid development in software technology and thus requires some new techniques for programming and tools.**

***Index Terms*– WSN, Architecture, Issues, Analysis and Software**

I. INTRODUCTION

WIRELESS sensor networks (WSNs) are just one of the presently/currently hot/active research topics. Both noncombatant/civilian and military oriented institutions have sidetracked/diverted a prodigious contract/deal of funding/subsidy to research on WSNs as advances in software and hardware deliver the means to make WSNs. WSN,s are masses of various small sensor nodes. Each node can send/transmit messages through the network to the information sink – or decisive controlling device. The nodes can also forward messages from other nodes, execute network organization tasks, and an assortment of other functions. The applications of WSNs vary widely. WSNs could be secondhand in industrial settings for machine control and environment monitoring. Other applications could be medical – monitoring a patient's health from a variety of perspectives. The military is highly interested in sensor networks for intelligence gathering, while WSNs have possible applications in aerospace for the structural integrity of planes. A great deal of research has already been performed on WSNs, and a number of possible implementations and architectures suggested. The University of California–Berkley has remained at the forefront of research, creating smart-dust, WEbS, and PicoRadio [1]. Through all prior research, it has been shown that WSNs require unique software architecture to solve inherent difficulties [1].

II. WSN CHARACTERISTICS

The idea of wireless sensor networks infers a number of WSN characteristics/features which severely sway the software architecture. Specifically, WSNs must be self organizing, execute cooperative processing, energy optimized, and modular. These four requirements/needs in particular influence/impact seriously on the form of the software architecture.

A. Self-Organization

The outsized number of nodes in a WSN renders direct handling/manipulation by a user for network organization impractical [1]. The user might not go through thousands of nodes leading the network configuration and clustering. Subsequently, the nodes must be proficient of organizing the network and partitioning it for competent operation given the environment and network attributes [1]. Additionally, the nodes of a sensor network must be robust [2]. The aggregate formed by the nodes must have a high up time. The large number of nodes in a network along with unattended operation complicates any attempt at a fault tolerant design [2]. Sensor networks with wired connections do not necessarily rely on other nodes to transmit data. This reduces the need for redundancy and the robustness of individual nodes. In contrast, wireless sensor network nodes transmit information from node to node with a small amount of processing in between [2]. Consequently individual nodes must be highly robust, while the organization of the network must tolerate individual device failure [2]. Variations in the network topology can affect the degree of network vulnerability to failures, necessitating complex routines to implement fault tolerance [4].

B. Cooperative/Obliging Processing, Concurrency

Nodes in a network mostly direct information flow/stream through the network to several data-sinks- the points to which data from the network is served/fed [2]. Every sensor node might posses a restricted quantity of memory, so that the cushioning/buffering of data is unfeasible [2]. Moreover, the node performs/executes a number of concurrent operations: processing, capturing, and transmitting/sending sensor data, whilst concurrently forwarding/sending data from further nodes in bridging or multi-hop situations [2]. Wireless Sensor Networks also deliver/offers a distinctive chance for obliging processing. Cooperative/obliging processing can reduce/diminish network traffic through data aggregation/accumulation and preprocessing [1]; e.g., the establishment of a wireless network might encompass the triangulation of a firsthand node when it joins a network to inaugurate the node's position.

C. Energy Efficiency

Wired sensor network have the luxury of external power sources such as power over Ethernet. The nodes of wireless networks have no practical way of utilizing an external energy source, which would in any case be contrary to the point of a

WSN. A sensor network may also be distributed in hostile or remote environments [1]. Energy efficiency dictates the minimization of communication between nodes. Therefore the choice of protocols and network configuration are key in terms of network lifespan [1]. Protocol related energy savings are directly related to the physical, link, and network layers [1]. Additional power savings come from an operation system (OS) for the nodes which supports advanced power management and lower power task scheduling [1]. Power sensitive task scheduling can minimize power use though non linear battery effects [1]. Advanced power management would put any hardware not in use to sleep, minimizing power consumption [2].

D. Modularity

Sensor nodes in a network tend to be specific, and therefore contain only the hardware needed for the application [2]. The range of possible applications dictates a large variance in the hardware required for sensor nodes [2]. Accordingly, the software for the nodes must exhibit a high degree of modularity [2].

III. SOFTWARE ARCHITECTURE COMPONENTS

The nature of a sensor network lends itself to a service oriented component based framework. Applications split into sensor, node, and network applications, providing the basis for fundamental application layers in a sensor network [2]. Sensor applications interface with the sensors, local data, and hardware on a node, along with the operating system [2]. Sensor applications form the base layer and provide the basic functions of a sensor node [2]. Node applications use the basic functions provided by sensor applications to perform middleware tasks for network buildup, maintenance, and localization [2]. Network applications deal with the services and tasks of the network as a whole [2]. Network applications thereby act as an interface to the layer administrating to the network [2].

A. Middleware

Middleware refers “to the software layer between operating system and sensor application on the one hand and the distributed application which interacts over the network on the other hand” [2]. Opinions vary on the actual granularity limitations. Some maintain that middleware, in various forms, exists in a sensor network hierarchy all the way down to individual nodes [2]. Others favor the cluster as the basic unit of middleware [3]. Regardless of the granularity, the design of middleware aims to be scalable, adaptive, generic, and reflective. Scalable middleware performs optimization based on resource constraints at runtime [2]. The nature of wireless sensor networks calls for lightweight middleware, or middleware which has low communication and computations requirements [3]. By performing optimizations at runtime the interfaces of middleware are customized [2]. The sensor network changes as nodes move, necessitating runtime adaptations of the middleware to exchange and run components as needed by the application [2].

Localized algorithms can be used to enhance system scalability and robustness in the face of interactions between sensor nodes [3]. These algorithms can also provide reflective middleware, which changes the behavior of layers on the fly instead of exchanging them [2]. Generic middleware attempts to reduce overhead imposed by using generic interfaces for middleware components [2]. This implies the customization of the application interfaces and features, allowing for interpretation by middleware and compile time optimization [2], [3]. Generic interfaces also allow for the standardization of system services to diverse application [3]. Conversely, while middleware interfaces may be generic, the interfaces of application component on a specific sensor node are anything but. In short, middleware acts as an abstraction layer to help hide software specifics from the application layer.

IV. SOFTWARE ARCHITECTURE

The requirements and characteristics of a wireless sensor network mentioned in the previous sections call for a service based architecture. The services provided naturally divide into layers which vary depending on the exact topology of the network. Sections A and B examine two possible software architectures for a WSN detailed in [1], [2], [3], and [4], accompanied by the underlying reasoning for each architecture.

A. A Basic Service Oriented Architecture

1) Architectural Description: A simple use case can help determine the software architecture. An example use case in [1] provides illustration. The client application requests data from the network about surface conditions in a certain area. The client first sends a request to a surrogate proxy for the desired information. The proxy communicates with the appropriate nodes, which in turn then determine the surface conditions in the area using cooperative algorithms [1]. The proxy takes the information returned from the nodes, translates it, and sends it back to the client [1]. Fig. 1 illustrates the use case. The use case, along with the requirements and characteristics specified in previous sections of this paper, calls for flexible software architecture. Such architecture can be realized using the node application structure shown in Fig. 2, along with the sensor network architecture shown in Fig. 3.

Fig. 2 illustrates the division of node applications into three layers. The lowest layer handles hardware specifics, such as hardware and sensor drivers. The node operating system acts as a buffer layer between the hardware specifics and the host middleware application layer. The operating system layer handles the processes which relate strictly to the node operation, while the host middleware handles processes concerning the services offered by the node to the network [1]. The middleware is comprised of four different components which are called as needed, with the option to add additional modules for security or routing [1]. The VM or Virtual machine component enables platform independent program execution, while algorithms define the behavior of modules [1].

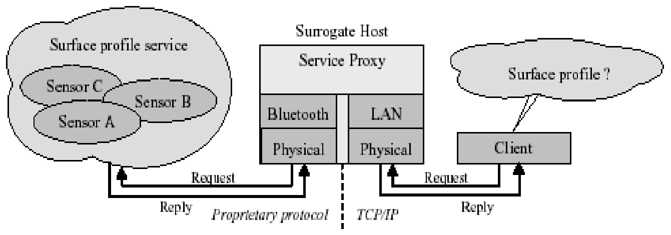

Fig. 1: Surrogate Architecture in Sensor Networks [1]

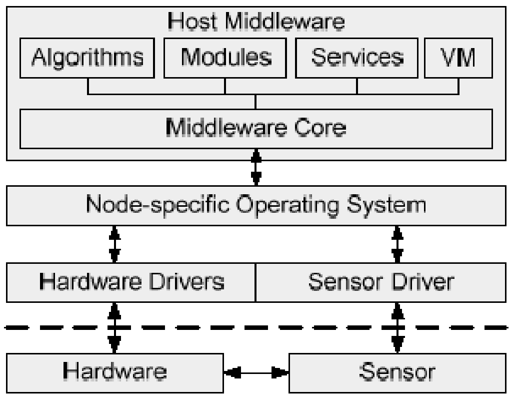

Fig. 2: Node Application Structure [1]

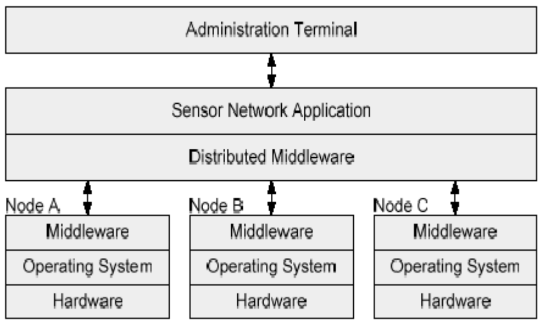

Fig. 3: Sensor Network Software Architecture [1]

The general overall software architecture of the sensor net is shown in Fig. 3. The individual nodes interact with the distributed middleware layer to perform the functions dictated by the sensor network application. The administration terminal is a connection point independent external actor which evaluates results from the sensor network application [1]. The diagram specifically illustrates the behavior of the sensor network application, which cannot assign tasks to individual nodes. Instead the layer abstract shown indicates that the distributed middleware handles tasks for the entire network and acts as network service coordinator [1].

B. Architectural Issues

While the architecture presented certainly presents a solid and basic design for a sensor network, it does not reflect some of the requirements such as energy efficiency which can have a significant effect of software architecture. Network topology control in particular proves effective to extending network life and increasing network capacity [5]. Additionally, the amount of energy available to each sensor cannot support long range communication, necessitating a tiered network structure [4]. The need for such a network structure then influences the software architecture design. Finally, the use of a single tier architecture increases network load on nodes surrounding the command node [4]. The increases traffic on these key nodes decreases their lifetime, and in turn shortens the lifetime of the entire network. The adaptation of a multi-tiered network structure can reduce the energy consumption imbalance if the network supports sufficient fault-tolerance [4].

1) A Cluster Based Service Oriented Architecture: The factors mention above can influence the software architecture of a wireless sensor network when taken into account. A second architecture, proposed in [3], uses clustering to handle the factors mentioned, and provide for application Quality of Service (QoS) management.

2) Architectural Description: One of the more favored network architectures for wireless sensor networks involves clustering. A cluster is a set of adjacent sensors which are grouped together and interface with the rest of the network through a gateway, or cluster head [4]. Gateways are higher energy nodes which maintain the network in the cluster, perform data aggregation, and organize sensors into subsets [4]. Clusters exhibit dynamic behavior. Clusters form and are modified on the fly depending on conditions and node availability [4]. During cluster formation, one node is elected as the gateway. It is important to note, that while clusters can overlap spatially, one node cannot belong to multiple clusters [3], [4].

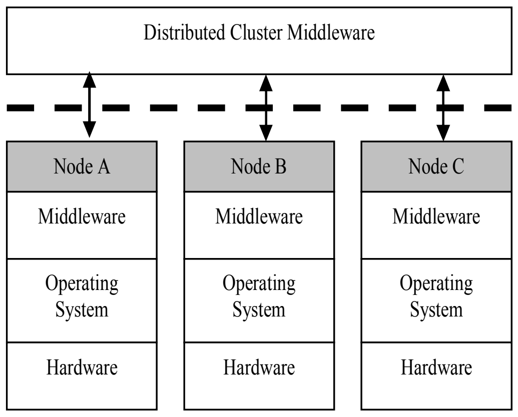

Fig. 4: Cluster Software Base Architecture

Given the presence of clustering, a cluster can be regarded as the base unit for the software architecture. Given such, data collection would be performed in a distributed manner [3]. However, in order to dynamically manage these clusters, the architecture of the middleware needs be fairly complex.

Fig. 4 displays the software architecture at the cluster level. Fig. 5 shows a cluster based middleware architecture proposed in [3]. As shown in Fig. 5, the architecture proposed contains an abstraction modeled as a Virtual Machine, similar to the architecture conceived in Section IV.A of this paper. The Virtual Machine provides the same service as in the first architecture– that of hardware independent program execution [3]. However, this Virtual Machine splits down into two additional layers: the resource management layer and the cluster layer. The cluster layer encompasses the distributed cluster middleware illustrated in Fig. 4. This software layer forms clusters from the collection of sensor nodes surrounding the target area [3]. The exact factors controlling the initial formation of clusters can vary depending on the application of the sensor network, and do not have significant impact on the software architecture.

The resource management layer controls resource allocation and adaptation to meet QoS requirements for the sensor network application [3]. Resource management is an important part of QoS in distributed application such as a wireless sensor network. Environmental and system changes can affect the amount of available resources, requiring the middleware to reallocate resources on the fly to accomplish the tasks given by the sensor network application [3].

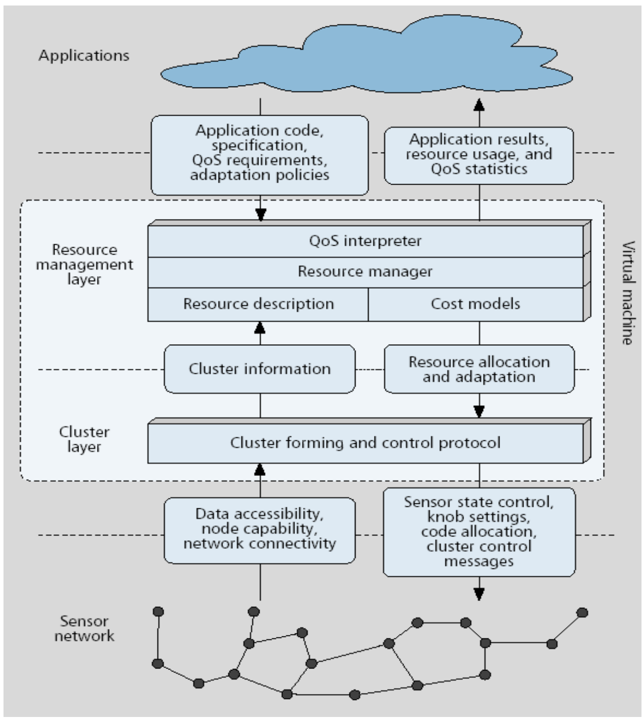
Applications
Application code, specification, QoS requirements, adaptation policies
Application results, resource usage, and QoS statistics
Resource management layer
QoS interpreter
Resource manager
Resource description
Cost models
Virtual machine
Cluster information
Resource allocation and adaptation
Cluster layer
Cluster forming and control protocol
Data accessibility, node capability, network connectivity
Sensor state control, knob settings, code allocation, cluster control messages
Sensor network

Fig. 5: Middleware architecture for a cluster based WSN [3]

C. Architectural Issues

The cluster based architecture proposed faces a number of challenges inherent in its design. The more complicated network topology incurs higher overhead costs for forming and maintaining clusters [3]. Specifically, clusters must be formed dynamically in order to track moving phenomena [3].

Therefore nodes would be changing cluster membership depending on the speed of the target phenomena. Using clusters also increases the vulnerability of the network to faults [4]. If a gateway fails, the members of the cluster must quickly deal with the fault, by electing a new cluster head or resorting to ad hoc networking to members of other clusters [4]. The more time is spent in network reconfiguration the more data can be lost due to memory limitations at the effected nodes [4]. This behavior implies an increased network vulnerability to the algorithms used for network formation. Overhead also comes from resource management layer. In order to effectively manage the resources of the network, the layer must gather and update information on node energy levels, network connectivity, cluster loads, and a number of other statistics [3]. Such polling can result in a dramatic increase to overhead if it is not handled correctly. Obviously the amount of overhead depends on the exact implementation of the resource management.

V. NODE SOFTWARE DEVELOPMENT

A. Software Development for Sensor Nodes

Sensor node software in particular lends itself well to an iterative form of development. As with any real-time embedded system, the code must be optimized to perform within certain parameters. Due to the sensitivity of the performance, efficiency, and lifetime of a WSN to the algorithms controlling network configuration, an iterative design, implementation, and test phase is required. Since the applications of WSNs vary so greatly, the algorithms involved in a specific type of WSN must be optimized through a large amount of calculation and design. The development pattern for node applications suggested in [1] favors the highly iterative design pattern show in Fig. 6. The design method closely follows standard software engineering practices. Of note, component interface optimization is performed during the design stage as mentioned in Section III.A of this paper [1]. Evaluation through the monitoring of results leads to additional iterations. The development pattern results in a specific application for the node comprised of specially tailed parts. Due to the presumption of generic middleware interfaces mentioned in Section III.A of this paper, the design process of the distributed middleware for either of the architectures mentioned conforms to the applicable processes used in software engineering.

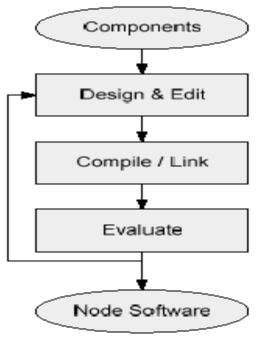

Fig. 6: Proposed Node Software Development Process [1]

VI. CONCLUSION

As described above, wireless sensor networks possess the potential for many applications. The advance of technology enabled the creation of prototype WSNs, but the hardware and software both have a ways to go before WSNs are practical, cost-effective, and usefully. Numerous software difficulties must be solved before wireless sensor networks may be considered a mature technology. Chief among these problems stands the formation, creation, and testing of a robust, efficient software architecture that can fulfill all of the goals and requirements needed. Despite (or because of) the work still to be done, wireless sensor networks are a great example of the unique challenges in software engineering produced by the advance of technology.

REFERENCES

[1] Blumenthal, Jan; Handy, Matthias; Golatowski, Frank; Hasse, Marc; Timmermann, Dirk. Wireless Sensor Networks – New Challenges in Software Engineering. Emerging Technologies and Factory Automation, 2003. Proceedings. ETFA '03. IEEE Conference , Volume: 1 , 16-19 Sept. 2003

[2] Hill, Jason; Szewczyk, Robert; Woo, Alec; Hollar, Seth; Culler, David; Pister; Kristofer. System Architecture Directions for Networked Sensors. ASPLOS 2000.

[3] Yu, Yang; Krishnamachari, Bhaskar; Prasanna, Viktor K. Issues in Designing Middleware for Wireless Sensor Networks. IEEE Network, Volume: 18, Issue: 1, Jan/Feb 2004 Pages: 15 – 21. Copyright 2004 IEEE.

[4] Gupta, G.; Younis, M. Fault-Tolerant Clustering of Wireless Sensor Networks Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE, Volume: 3, 16-20 March 2003. Pages: 1579 - 1584 vol. Copyright 2003 IEEE.

[5] Liu, J.; Li, B., Distributed Topology Control in Wireless Sensor Networks with Asymmetric Links, Global Telecommunications Conference, 2003. GLOBECOM '03. IEEE, Volume: 3, 1-5 Dec. 2003, Pages: 1257 - 1262 vol.3, Copyright IEEE 2003.