ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 2, February 2015

A Review on Run-Time Reconfigurations and Code Update Mechanisms in Wireless Sensor Networks

Rekha.K.S.[1], Dr.T.H.Sreenivas[2]

Assistant Professor, Department of Computer Science & Engineering, The National Institute of Engineering, Mysore,, India[1]

Professor, Department of Information Science & Engineering, The National Institute of Engineering, Mysore, India[2]

ABSTRACT: The design of wireless sensor networks for distributed applications is the most challenging task due to the factors such as limited storage on nodes, variable data arrival rate and high energy cost of communication. In most Wireless Sensor Networks(WSN), the battery is the sole energy source of the sensor node. The Sensor nodes are expected to work on batteries for several months to a few years without replenishing. Thus, energy efficiency becomes a major issue in WSNs**.** Nowadays, it is desired to use the WSN in heterogeneous platform for multiple monitoring. The Software development for the sensor node is a tedious task due to the change in network topology. The present state of art has come up with different solutions to overcome the performance issues such as Memory overhead, Heterogeneity, Portability, Scalability, Cost and Quality of Service. An adaptive framework should minimize the resource consumption and provide an optimal solution for run-time Reconfiguration.

This paper presents a detailed review on current state-of-the art in the Run-time Reconfigurations of Wireless Sensor Networks. An Analysis is made on the existing middleware approaches and an evaluation is done based on the parameters like Application Openness, Scalability, Heterogeneity, User Friendly Interface, Mobility and Power Efficiency. The paper identifies a few open research issues that must be addressed during the Run-Time Reconfigurations of Wireless Sensor Networks.

KEY WORDS: Wireless Sensor Networks, Reconfiguration, Heterogeneity, Scalability, Portability, Quality of Service.

I. INTRODUCTION

The Wireless Sensor networks have greatly influenced our life from an Environment Monitoring to Battle field monitoring. The design of a sensor network involves many disciplines such as signal processing, networking, Embedded System, Software Engineering and distributed algorithms. Due the factors like easy deployment, self-organization, self-healing the Monitoring System could be set up in an industry for the process control/Technology, Agriculture (water harvesting), Health(patient Monitoring),Society(Fire Fighters, Pervasive computing).A wireless sensor network consists of a number of nodes spread across a geographical area. In [1], three important functional requirements were identified: the need for dynamic modifications, the need for heterogeneous node support, and the need to integrate new software versions into a running system. Remote software updating is a core component in improving ease of use by providing flexibility in a deployed network, and directly supporting the maintenance function.

The nodes are typically fitted with sensors (for example light, temperature, pressure, and audio sensors) for monitoring their environment and have wireless communication capability. The nodes also have some degree of intelligence for processing the data gathered by their sensors and the capacity to do so. The network is deployed over a geographical area in an ad hoc manner. Some nodes within the network could be placed at known locations, but by and large, the nodes figure out their positions themselves using various localization algorithms [49], [50], [51]. Reconfiguration provides an efficient solution to the changing needs of the various applications in the WSN. It is

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 2, February 2015

required to reprogram the sensor nodes whenever a new functionalities need to be sent through wireless links. The Code Dissemination mechanism would propagate the code to the intended node. The highly dynamic application should be self-configured to meet the changing needs of the network

Wireless nodes may operate in a heterogeneous environment where environmental conditions and protocols vary with place and time. This suggests need for nodes to sense the environment and reconfigure platform software as necessary. Such reconfiguration may involve dynamically loading and unloading appropriate software modules or tuning parameter settings to achieve desired performance[2]. Reconfigurations can be performed either by Centralized method or Distributed method. The Centralized Reconfiguration monitors the Quality of Service at the base station and reprogram the sensor nodes when necessary. The Distributed Reconfiguration method locally monitor the context and perform reconfiguration on each node .

In Section II, the Classification of Wireless Sensor Applications based on Sensor Node, Network and Application is given . The Section III presents the Software Update Mechanisms in WSN. The Section IV classifies the Middleware Approaches for WSN. The Section V identifies a few open research problems during Run-Time Reconfiguration of WSN. The the theare Update Mechanism for Run-time Reconfiguration and Concluding Remarks.

II. CLASSIFICATION OF WIRELESS SENSOR NETWORK APPLICATIONS

For the development of application software for sensor networks, the use of a component based framework is desirable. The components of the framework provide the functionality of single sensors, sensor nodes, and the whole sensor network. According to these components, applications are classified into sensor applications, node applications and network application[5]. The Figure 1 shows the Architecture of Sensor Networks.

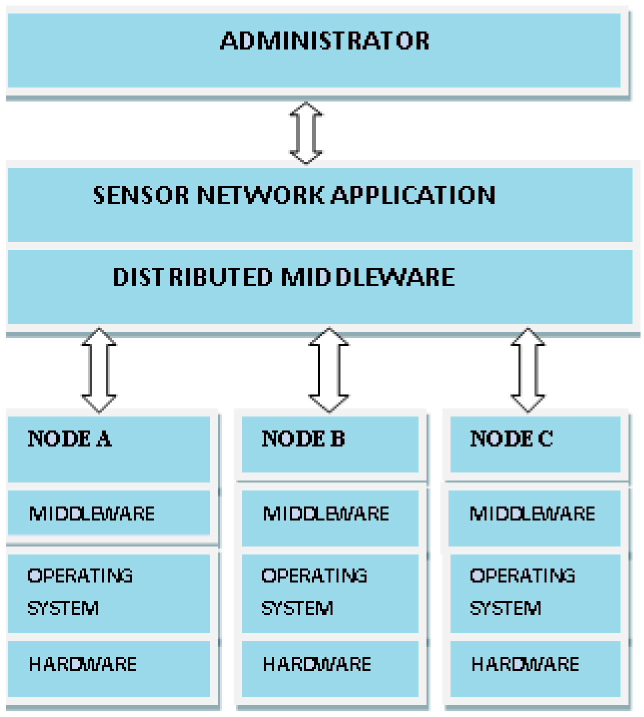

Fig 1. Architecture of Sensor Network

A. Reconfiguration on Node Properties

The *node application* contains all application specific tasks and functions of the middleware to build up and maintain the network, e.g., routing, looking for nodes, discovering services, and self localization[6][53].

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 2, February 2015

The WSN's are deployed for dynamic applications, the topology of the network changes continuously due to the factors like limited Battery. There are several topologies like Star, Mesh, Cluster Based Topology. The Reconfiguration of node properties is required in dynamic applications like weather forecasting where data has to be gathered for a range query posed by the user. The reconfiguration options are adapting topology for efficient dissemination, concentrating only on a part of target nodes to achieve high coverage and performing role assignment so that all functions are divided.

In [8], a comprehensive model LACON is proposed for monitoring and reconfiguring sensor networks. The LACON provides measures to mitigate network impairments. The results confirm that the proposed self configuring model adopts optimized design parameters and maintains QoS even for highly ad hoc networks. In [9], a new Probabilistic Geographic Routing protocol (PGR) is introduced which is a decentralized energy aware routing protocol for wireless ad hoc and sensor networks. PGR uses geographical location a long with residual energy and link reliability information to make routing decisions. Instead of deterministically choosing the next hop, PGR assigns probabilities to the candidate next hop nodes. The probability assigned to each node is a function of its residual energy and the corresponding link reliability estimation. Using the residual energy in the cost function ensures that nodes with more reliable links are not drained out of energy too quickly. This will in turn increase the lifetime of the network .In [10], a novel programming abstraction called generic role assignment, which allows the automatic assignment of roles to sensor nodes based on properties of sensor nodes and their respective network neighbourhood. A distributed algorithm is proposed for role assignment and two variations that perform probabilistic initialization is also defined.

B. Network Reconfiguration

The *sensor network application* describes the main tasks and required services of the entire network without assigning any tasks or services to individual nodes. It represents an interface to the administrator to evaluate the network results**.** Network reconfiguration refers to node discovery, role assignment, changing the whole network inclusive its architecture and its protocols[53].

In[2] , a distributed mechanism is presented that enables individual sensors to follow locally selfish strategies. This in turn, result in the self-organization of a routing network with desirable global properties. The Directed Diffusion [3], is a well-known data dissemination mechanism for Sensor Networks, whose main aspects include data-centric routing, in-network aggregation and attribute-based data naming. In [4], two new variants were proposed, Push Diffusion, optimized for many receivers and few senders and One-Phase Pull, designed for the reverse case.

C. A sensor application

A sensor application contains the readout of a sensor as well as the local storage of data. It has full access to the hardware and is able to access the operating system directly. The sensor application provides essential basic functions of the local sensor node, which may be used by the node application[53].

In[49], the process of Sensor Reconfiguration by which instruction given to the sensor network/ some sensors in the sensor network to change this configuration data. Configuration data is stored on non-volatile storage space i.e. EEPROM of the mote. The sensor node is supposed to read it and take appropriate action.

III. SOFTWARE UPDATE MECHANISMS IN WSN

WSN's software update research is described under three categories : the sensor node as **execution environment**, the protocols for **disseminating the update**, and **size reduction** of transmitted update at the Host[1].

In [30], the widely used mechanisms for reprogramming the sensor nodes : full image replacement and binary difference image replacement are proposed. In[32],Dynamic TinyOS is proposed for Modular and Transparent Incremental Code-Updates.

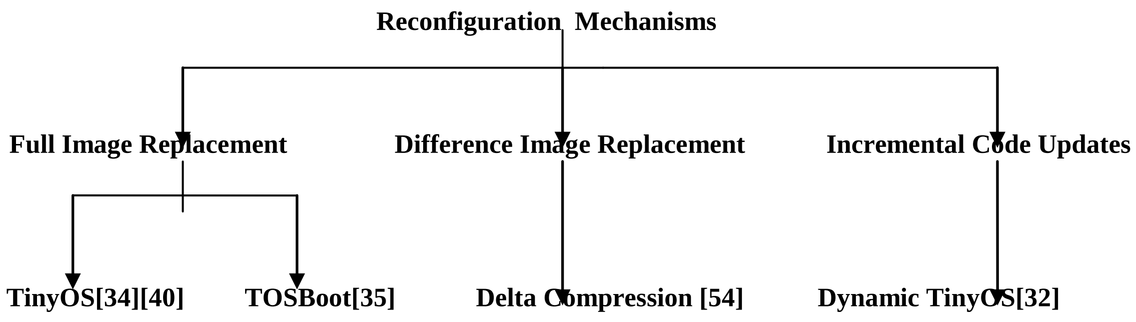

Fig 2. Reconfiguration Mechanisms

A. Full Image Replacement

The most common way to update software in embedded systems and sensor networks is to compile a complete new binary image of the software together with the operating system and overwrite the existing system image of the sensor node. The full image replacement does not require any additional processing of the loaded system image before it is loaded into the system, since the loaded image resides at the same, known, physical memory address as the previous system image[30].

The most popular way to update software in sensor networks is to compile a complete new binary image of the application software together with the system code and overwrite the existing system image of the sensor node. Since the image is compiled and linked afresh in every iteration, these solutions offer a very fine-grained control over the possible reconfigurations. However, these approaches result in bandwidth overhead as unchanged parts of an application need to be re-disseminated in the network[34].

- **TinyOS** : TinyOS [34[,[40] is a popular operating system for sensor nodes that generates a monolithic binary image of the entire application. Deluge [8] is a networked boot loader and dissemination protocol that performs full image upgrades of TinyOS applications.

- **TOSBoot** : TOSBoot [35], is an exceptionally flexible, albeit low complexity approach. As TOSBoot is a complete binary-image replacement mechanism, any system functionality may be changed including bug-fixes, driver updates, or even an entirely new operating system if so desired. TOSBoot provides additional flexibility by having the ability to handle multiple binary images. This allows the node to be imaged with any of the stored images at boot-time meaning an older version of an image does not have to be re-uploaded over the network if it should be required again at a later date. TOSBoot is inflexible however in the fact that large binary images must be sent over the network, this mechanism may be too slow for a large amount of nodes. TOSBoot also requires that nodes have access to external storage, such as flash, to store uploaded images, something that may not be present on all systems.

B. Incremental Updates

- **Dynamic TinyOS[32]** is introduced to enable the dynamic exchange of software components and thus incrementally update the operating system and its applications. The core idea is to preserve the modularity of TinyOS, i.e. its componentization, which is lost during the normal compilation process, and enable runtime composition of TinyOS components on the sensor node. The proposed solution integrates seamlessly into the system architecture of TinyOS. It does not require any changes to the programming model of TinyOS and existing components can be reused transparently.

Evaluation : Dynamic TinyOS incurs a low performance overhead while keeping a smaller - up to one third - memory footprint than other comparable solutions.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 2, February 2015

C. Diff-based Approaches

Often a small update in the code of the system, such as a bug_x, will cause only minor differences between in the new and old system image. Instead of distributing a new full system image the binary differences, deltas, between the modified and original binary can be distributed. This reduces the amount of data that needs to be transferred [30].Delta algorithms compress data by encoding one file in terms of another. This type of compression is useful in a number of situations: storing multiple versions of data, displaying differences, merging changes, distributing updates, storing backups, transmitting video sequences, and other[55]. In [54], an efficient code update mechanism for sensor networks based on differential compression is presented. The delta algorithm is pursuing a greedy strategy resulting in minimal delta file sizes. The algorithm operates on binary data without any prior knowledge of the program code structure. Performance evaluations show that update size reductions in the range of 30% for major upgrades to 99% for small changes are achieved.

IV. MIDDLEWARE APPROACHES FOR WSN

The use of middleware can bridge the gap between applications and low-level constructs. The main purpose of middleware in Wireless Sensor Network is to support network hardware, network stack , application installation and data maintenance activities. The middleware is responsible for execution among heterogeneous nodes.

The middleware platforms are designed with abstractions that can offer consistent and general mechanisms to deploy, reconfigure, both system and application level software. The middleware collects the information from the application and network protocols and decides how to support the applications and at the same time adjust network protocol parameters. The design of a successful middleware layer must address many challenges to meet the application needs. The General middleware Architecture for Wireless Sensor networks is shown below in Figure 3.

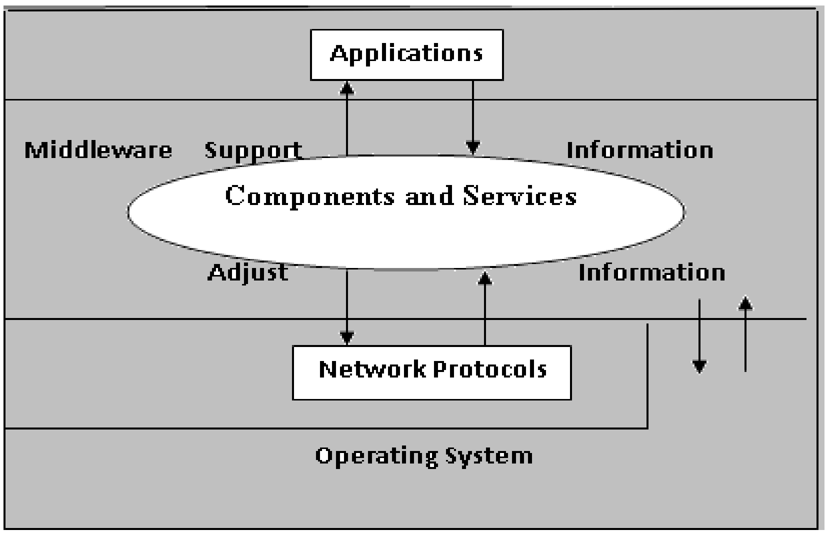

Fig 3. General Middleware Architecture for Wireless Sensor Networks

- **Scalability**

If new nodes are added to the existing application, the network performance should not be affected.

- **Heterogeneity**

Middleware should provide low-level programming models to bridge the gap between hardware and networks.

- **Ease of Use**

The level of abstraction Middleware defines user friendly approach.

- **Application Openness**

Application Openness is the ability to extend and modify the existing system, whenever there is a change in requirements.

- **Mobility**

It is middleware ability to keep connectivity with mobile nodes in a network.

- **Power Efficiency**

The middleware should be designed for effective utilization of processor and memory by enabling lower-power communication.

4.1. Classification of Middleware Approaches for WSN

The term *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. The primary objective of the middleware layer is to hide the complexity of the network environment by isolating the application from protocol handling, memory management, network functionality and parallelism [52].

In[12], the two main classes of Programming sensor networks : *programming support* and *programming abstractions* are proposed. The first is concerned with providing systems, services, and runtime mechanisms such as reliable code distribution, safe code execution, and application-specific services. The second provides the concepts and abstractions of sensor nodes and sensor data. The programming abstractions class has two main subclasses . The first focuses on the global behaviour of a distributed sensor network as a whole also called macroprogramming. The second deals with the local behaviour of the sensor network nodes in a distributed computation.

In [11], the author has proposed the five main subclasses under the programming support class namely: virtual machine based, modular programming based, database based, application driven, and EventBased middleware . The Figure 4. Shows the programming models for Wireless Sensor Networks.

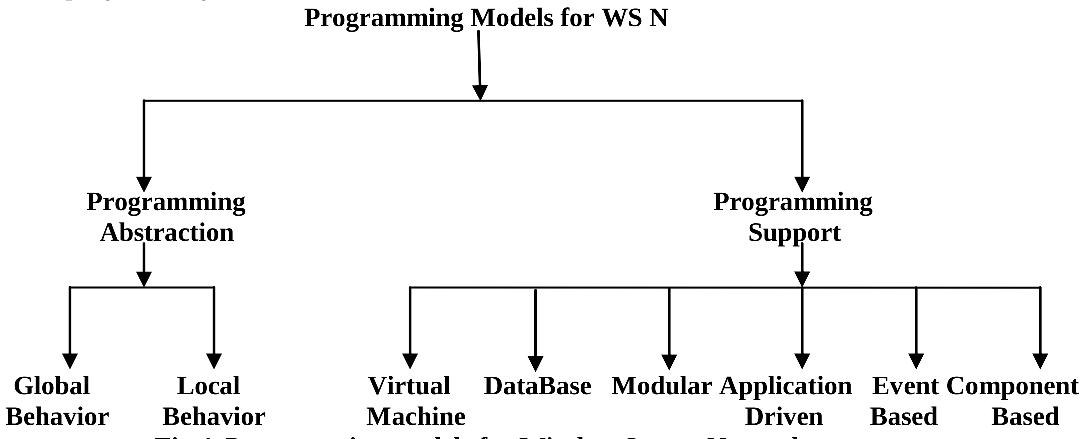

Fig 4. Programming models for Wireless Sensor Networks

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 2, February 2015

4.2 Programming Support

A. Virtual Machine

Virtual machine middleware approach is used to decrease overall power and resource consumption .The system consists of virtual machines and interpreters. Developers write applications into small modules, The system injects and distributes the modules through the network[12].

1) Mate:

Mate [14] is a byte code interpreter that runs on TinyOS. It is a single TinyOS component that sits on top of several system components, including sensors, the network stack, and nonvolatile storage (the "logger"). Code is broken in **capsules** of 24 instructions, each of which is a single byte long; larger programs can be composed of multiple capsules. In addition to byte codes, capsules contain identifying and version information. Mate avoids message buffering and large storage. The synchronous model makes application level programming simpler and far less prone to bugs than dealing with asynchronous event notifications. Another Mate functionality is infection or network updates achieved by adding a version number to the capsule. So, comparison takes place at the neighbors, followed by installation of the new version. This process cascades with hop-to-hop communication.

Mate has a stack-based architecture [15] with three execution contexts – clock, send, and receive Although Mate has a small, concise, resilient, and simple programming model, its energy consumption is high for long running programs. Mate's virtual machine architecture increases security. But its programming model is not flexible enough to support wide range of applications.

Evaluation: Mate program is small and suitable for sleepy application, where energy consumption is very less. Mate supports adapting features to changes in the sensor networks. For complex application much energy is required due to the interpretation overhead.

2) MagnetOS

In [26], a distributed, power-aware, adaptive operating system, called MagnetOS, is proposed. MagnetOS specifically targets ad hoc and sensor networks. MagnetOS provides a single system image of a unified Java Virtual Machine(JVM) across the nodes that comprise an ad hoc network. The abstraction lets the whole network appear as a single, unified Java VM. Following the JVM pattern, the system comprises dynamic and static components. The static component is responsible for rewriting regular java applications in the form of objects or modules, which explains MagnetOS's object-oriented nature. Then the component injects them into the network with special instructions to keep the semantics. At this point, a dynamic runtime component on each node monitors the object's creation, invocation, and migration, providing different services for applications[12].

Evaluation : dividing the applications into several component and placing these on components reduces energy consumption, avoids hotspots and increases system longevity.

B. Database Inspired Middleware

This approach views the whole network as a virtual database system. It provides an easy-to-use interface that lets the user issue queries to the sensor network to extract the data of interest. However, the approach provides only approximate results, and it lacks the support of real-time applications that need the detection of spatio-temporal relationships between events [12].

1) TinyDB

TinyDB[16] is a distributed query processor that runs on each of the nodes in a sensor network. TinyDB runs on the Berkeley *mote* platform, on top of the TinyOS [Hill et al. 2000] operating system. Queries in TinyDB, as in SQL, consist of a SELECT-FROM-WHERE-GROUPBY clause supporting selection, join, projection, and aggregation.The semantics of SELECT, FROM, WHERE, and GROUP BY clauses are as in SQL. The FROM clause may refer to both the sensors table as well as stored tables, which are called materialization points. Materialization points are created through special logging queries. They provide basic support for sub-queries and windowed stream operations .Tuples

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 2, February 2015

are produced at well-defined sample intervals that are a parameter of the query. The period of time between the start of each sample period is known as an epoch. Epochs provide a convenient mechanism for structuring computation to minimize power consumption.

Evaluation: Multiple queries are supported by TinyDB, The TinyDB supports network software scalability by Augmenting the new code to the standard TinyDB code.

2) Cougar

Cougar is another middleware applying database pattern in sensor network. In Cougar system, there are two types of data: stored data and sensor data. Signal processing functions in each sensor node generate the sensor data, and date are communicated or stored in local as relations in database system[13] .

C. Event Based Middleware

Another approach to WSN middleware is based on the notion of events. Here, the application specifies interest in certain state changes of the real world (basic events). Upon detecting such an event, a sensor node sends a so-called event notification towards interested applications. The application can also specify certain patterns of events (compound events), such that the application is only notified if occurred events match these patterns[17].

1) WMOS

In [18] , the WMOS (Wireless-Middleware Publish Subscriber System) middleware is proposed which uses the Publish/Subscribe Communication Generics. It is developed on the TinyOS and realizes the content/topic self-adaptive double-mode based on the XML matching. It has grade QoS and provides data gathering services such as Pub / Sub middleware, and it also provides a set of API interfaces for application layer. WMOS can not only provide developers with a convenient interface, but also reduce the energy consumption of each sensor node effectively

Evaluation: WMOS will provide the interface which can receive the information and does the matching with corresponding sensor node. The communication link will be established only when the information collected by the sensor node matches with the query. Publisher/Subscriber bridges the between application layer and protocol layer and reduces the energy consumption.WMOS will increase the security, resources exploration function gradually.

2) SensorBus:

In [23], SensorBus, a message-oriented middleware (MOM) model that employs the publish-subscribe paradigm has been proposed. In this approach, a component that generates events (producer) publishes the types of events that will be available to other components (consumers).The consumer interested in a determined event “subscribes” to this event, receiving from this moment on notifications about the event “subscribed” to. These notifications are sent asynchronously from producers to all interested consumers. The MOM performs the functions of collecting producer’s messages, filtering and transforming such messages (when necessary) and routing them to the appropriate consumers.

3) DSWare(Data Service Middleware)

DSWare[56][57] provides a novel event-detection mechanism that is reliable and energy efficient. resides between the application and network layers, integrates various real-time data services and provides a database like abstraction to applications. In DsWare the data are replicated in multiple physical nodes mapped to a single logical node using hash based mapping.

4) Mires

Mires proposes an adaptation of a message-oriented middleware for traditional fixed distributed systems. Mires provides an asynchronous communication model that's suitable for WSN applications, which are event driven in most cases, and has more advantages over the traditional request-reply model. Mires is built on TinyOS using NesC. It adopts a component-based programming model using active messages to implement its publish-subscribe-based communication infrastructure[45][12].

D. Component Based Middleware

The component model introduces a new plugin to the existing components in a software. The component model helps for the light weight reconfiguration for Wireless Sensor Networks, as it only updates the required portion of the module rather than changing the entire modules.

In[19], ReWise a new component based middleware is proposed to Home monitoring application to enable reconfiguration of WSN. In this application, five different types of sensor node including temperature, smoke detector, occupancy detector, in-bed detector and humidity are deployed. As different sensor types are likely to have their own software components for running relevant application logic, at network deployment time each sensor is programmed with the core middleware services atop operation system. After successfully deployment of sensors, according to the type of each sensor the other necessary application components should be transferred remotely to each node. During the application running time, variant scenarios of component reconfigurations might be happened.

Evaluation: The ReWise component based reconfiguration is limited to the TinyComponents, But, if there is a need to replace the main component then the issues of State preservation and safety checking is not addressed.

1) RUNES MIDDLEWARE

In[20], the RUNES a component-based approach to network embedded system. The RUNES is proposed for a tunnel fire scenario. where the tunnel is instrumented with sensor nodes that report to a central controller when possible, but that can dynamically adapt their behaviour to report to fire-fighters entering the tunnel in groups to rescue the situation. The middleware needs to allow communication among different devices, and must allow adaptation of behavior in a context aware manner
.

Evaluation : The Runes middleware overcomes the problem of heterogeneity, software reconfiguration and topology reconfiguration in tunnel fire scenario.

2) WISEKIT

In [21] , a novel distributed middleware approach named WiSeKit is presented for addressing the dynamicity of Context-aware home. WiSeKit provides an abstract layer accelerating development of adaptive WSN applications. Using this middleware, the developer focuses only on application-level requirements for adaptivity, while the underlying middleware services expose off-the-shelf APIs to formalize the process of adaptive WSN application development and hide the complexity of the technical aspects of adaptation. The basic design concept of WISEKIT are Adaptation Time, Adaptation Scope, Adaptation Policy, Fine-grained Reconfiguration and Hierarchical Adaptation.

Evaluation: In this paper adaptation of only few application running on sensor node are addressed. The application which has to be deployed on cluster head and sink has to be designed further.

In [25] a novel distributed component-based middleware approach called WISEKIT proposed for addressing the dynamic applications. WISEKIT provides an abstract layer accelerating development of adaptive WSN applications. Using this middleware, the developer focuses only on application-level requirements for adaptivity, while the underlying middleware services expose off-the-shelf APIs to formalize the process of adaptive WSN application development and to hide the complexity of the technical aspects of adaptation. An Application scenario in the area of Home Monitoring is implemented with WISEKIT.

Evaluation: WISEKIT supports distributed adoption and hides the technical complexities of adoption.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 2, February 2015

3) **FIGARO :**

In[22] FIGARO, a component represents a single unit of functionality and deployment. The services provided by a component are described by its interface. For instance, Components must provide the code for all the operations in the interface declaration. The programming model of Figaro has two core constituents: – the **component model** defines constructs for structuring the code on the single nodes. It is designed with reconfiguration in mind, thus providing dedicated constructs to deal with component dependencies and versions, and to simplify the reconfiguration process. – the **distribution model** defines constructs to restrict component dissemination only to a given subset of nodes—the reconfiguration target—based on programmer specified characteristics of the nodes or their current software configuration.

In FIGARO, programmers do not need to manage the reconfiguration manually, Instead, the underlying run-time automatically and transparently manages the reconfiguration process, based on dependencies and component versions. When components are instantiated at startup, the run-time keeps track of their version, the interface they implement, and their dependencies. In FIGARO, dependencies are explicitly declared by the programmer using the **DECLARE DEPENDENCY** macro. The first parameter of this macro is a receptacle, the dual of an interface. An interface specifies a set of operations provided by a component to others, while a receptacle specifies the set of interfaces a component requires from others. Using FIGARO, the programmer can deal explicitly with component dependencies and version constraints, as well as select precisely the subset of nodes targeted by reconfiguration, leaving the others unaltered. The run-time support imposes a very limited processing and memory overhead. The communication overhead lies within 9% of the theoretical optimum[16].

Another relevant work in the context of component-based reconfiguration for WSN has recently been reported under the name of the FIGARO framework [20] as an approach for WSN reconfiguration in the RUNES project [22]. The main contribution of FIGARO is to present an approach for determining *what* should be reconfigured *and* where the reconfiguration should take place. The former one is related to runtime component replacement, and the latter is concerned with which nodes in the network should receive an updated code. The component model of FIGARO also fails to support lightweight reconfiguration, which is addressed in an approach by REWISE component model.

4) REMOWARE the Reconfiguration Middleware

In [24] , REMOWARE a new middleware solution is presented for Home monitoring application. The Removare addresses the crucial issue by paying particular considerations to the resource and energy overheads induced by the reconfiguration process. RemoWare includes a set of optimized reconfiguration services deployed on the sensor nodes,which consistently update the required pieces of code. These services include binary update preparation, code distribution, run-time linking, dynamic memory allocation and loading, and system state preservation. RemoWare can be exploited on any system software written in the C language.

Evaluation : The fine-grained reconfiguration process is achieved with component-based approach. In SOS's reconfiguration model relies on kernel features, thereby kernel upgrades require replacement of the entire kernel image. In RemoWare the key reconfiguration tasks are implemented independently of the underlying system software. The memory allocation model is one of the most important innovation of RemoWare compared to other proposed systems except FiGaRo. which is limited to Contiki's capabilities for dynamic reconfiguration.

E. Modular Programming Approach

The main idea of this approach is to make applications as modular as possible to facilitate their injection and distribution through the overall network by mobile code. Also less energy is consumed to transmit small modules than whole applications[33].

1) Impala:

Impala is a middleware designed based on an event based programming model with code modularity, ease of application adaptability and update, fault-tolerance, energy efficiency, and long deployment time in focus[23]. Impala

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 2, February 2015

doesn't support heterogeneity in terms of hardware platforms, since it's being destined to run only on Hewlett-Packard/Compaq iPAQ Pocket PC handhelds running Linux. So, its applications are limited [14].

2) SOS

SOS - A Modular Sensor Network OS, wiith a structured architecture based on a small kernel that is installed on all the nodes in the network. The rest of the system and the application functionality are implemented as a set of dynamically loadable binary modules. This modularity forms the basis of the SOS architecture, as it defines distinct boundaries and allows modules to be loaded and unloaded at runtime.SOS provides an event-driven execution model, with each module implementing a handler that is invoked by the OS scheduler to dispatch messages to destination modules[36].

3) FlexCup

A FlexCup ("FLEXible Code UPdates"), a code update mechanism that enables on the fly reinstallation of software components in TinyOS-based sensor nodes in an efficient way. FlexCup needs to be involved in the process of compiling the components on the base station, and installing the code update on the sensor nodes: During the code generation process, FlexCup generates meta-data that describes the compiled components. FlexCup then uses this meta-data during a code update to place the new component inside the running application, relink function calls to the appropriate locations and perform address binding of data objects. Using this method, FlexCup is able to reconfigure, exchange or reinstall parts of an application running on sensor nodes without having to retransmit the whole program image[7].

F. Application driven

This approach introduces a new dimension in middleware design by supplementing an architecture that reaches the network protocol stack. This will let programmers finetune the network on the basis of application requirements that is, applications will dictate network operations management, providing a QoS advantage. However, the tight coupling with applications might result in specialized, not general purpose, middleware[12].

1) MiLAN :

MiLan for WSNs means Middleware Linking Applications and Networks, it provides the solution that specific applications are allowed to affect the performance of entire network. MiLAN contains network protocol stack to configuration and manage network. It use graph based approach to allow application to know how it performs collected data from different combinations of low level* components, and how to choose combination of sensors to satisfy its quality of service requirements. MiLAN is originally designed for medical advising and monitoring, so it have good performance on application reliable[38][13].

Evaluation: While conventional middleware operates above the networking layer, for applications that rely on multiple and varying low-level data sources, it is not a viable approach to manage the network completely independent from the needs of the application. The core of the research presented in this paper is the integration between the needs of the application and the management of the network into a single, unified middleware system called Milan. Through this tight coupling, Milan can tradeoff between application performance and network cost, while still retaining the separation between the policy specifying how to react to a dynamic environment and the mechanisms to implement the policy.

2) Finger:

Finger [46] is a WSN policy system developed at Imperial College and completely implemented on top of the TinyOS [40] operating system. Finger operates on a reduced subset of the Ponder2 policy specification language.

3) ESCAPE

ESCAPE [47], is a policy-based WSN middleware implemented on TinyOS [40]. Applications in ESCAPE are implemented using nesC-components [2] and adopt a similar pub/sub interaction model .

V. RELATED WORKS

- **MiSense**

In [45] , MiSense acomponent-based service-oriented middleware Architecture is proposed. It provides an abstraction layer between applications and the underlying network infrastructure. MiSense promotes a service-oriented middleware component framework that can reduce complexity by imposing structure on top of the component model in the form of composability restrictions and by offering well defined, service-specific interfaces to the rest of the system. MiSense aims at fixing the service interface at a level of abstraction that will maximize the gains in productivity, while keeping those parts of the architecture with significant impact on the performance flexible enough to be able to benefit from domain-specific optimization. MiSense provides a well-defined content based publish/subscribe service, but allows the application designer to adapt the service by making orthogonal choices about the communication components for subscription and notification delivery, the supported data attributes, and a set of service extension components.

Evaluation : MiSense reduces the complexity by offering well-defined service- specific interfaces. MiSense breaks up the middleware design into interacting component so as to meet requirements such as optimization flexibility and reusability. MiSense meet the application requirements and increases the network global time.

- **MIREA(Middleware for real-time wireless embedded systems)**

In [28] A component-based middleware called MIREA has been proposed to build the context-aware framework. MIREA [29] is specifically targeted at real-time embedded systems. MIREA is light-weight, component-based, and supports flexible reusability of software components. The aim of building the context-aware framework on top of the component-based middleware in is to lend adaptability at the middleware level that in turn will lead to adaptation at the application level.

This framework provides integration between the component software and context-awareness technologies for the WSNs. This design architecture has several advantages in terms of ease of programming, software updates and reconfiguration, dynamic application development etc. This architecture is lightweight in nature and suitable for sensor node level context-aware processing.

Evaluation : An experiment was carried out to compare the Contiki image size with context-aware framework implementation as a single software module (monolithic context-aware framework) vs. Contiki image with MIREA implementation. In case of monolithic implementation, once the context-aware framework is programmed on the node, it is difficult to reconfigure and update the software program. Any software reconfiguration would require communicating full application image of ~23K bytes to the node. On the contrary, in case of the component-based implementation, once the node is programmed with MIREA middleware that occupies ~33K bytes, less than ~1K bytes need to be transmitted to the node to enable component runtime reconfiguration or reprogramming.

- **VAPRES**

In [41],an architectural frame work called VAPRES has been proposed for situation-based reconfiguration in WSN. VAPRES (pronounced “vapors”, the Virtual Architecture for Partially Reconfigurable Embedded Systems) is a virtual architecture developed for Partial Reconfigurable-capable FPGAs to provide a flexible and dynamic module communication layer. The VAPRES system control region includes a soft processor to serve as the central controlling agent, a flash controller core to read and store module partial bitstreams, and numerous peripherals for external device communication. VAPRES and the underlying communication architecture provide an essential enabling methodology for dynami ,situation-based PR.

Evaluation: The architecture of VAPRES is evaluated using three metrics: processing performance, power consumption, and resource utilization. The use of VAPERS provide increased WSN flexibility and computational resources, resulting in superior power consumption and performance compared to a microprocessor capable of satisfying similar demands**.** These benefits include power consumption reductions, consistent performance when tracking multiple targets, and significant cost savings.

- ***µDDS:***

µDDS is a publish/subscribe middleware for real-time wireless embedded systems based on Data Distribution Service specification and implements a subset of standard interfaces for event subscriptions and publication to be used by applications. There are three main elements of the development model, µ[54] DS Middleware and µDDS library, user application and PaRTiKle's Kernel . Applications implemented on top of µDDS can disseminate and collect data through a publish/subscribe interface provided by the middleware. Different routing protocols can be used to implement the overlay network; the middleware is currently implemented on 802.15.4 standard devices which can support the star, tree and mesh topologies [28]

Evaluations: results demonstrate that µDDS is lightweight and efficient, and the use of the µDDS middleware simplifies the development process of real-time wireless embedded publish/subscribe applications.

- **EnviroTrack (Data centric)**

EnviroTrack (Data centric) [42], is well suited for embedded tracking application. It adopts a *data centric* programming paradigm called attributed based naming through "context labels", where the routing and addressing are based on the content of the requested data rather than the identity of the target sensor node. As most projects, it is also built on top of TinyOS using compiled NesC [35] programs. Its contribution stems from its convenient robust interface to the application developer geared towards tracking the physical environment. The attributed based naming is applied by associating user-defined entities (context label) to real physical targets. With this network abstraction layer the programmer declares the environmental characteristics which define the context label of the object to be tracked. Based on this, all sensor nodes that sense the same declared characteristics (object) are aggregated to track that physical target such as a car or a fire. With powerful network management mechanisms such as lightweight group management and group leader election, it supports the dynamic behavior of the tracked targets such as mobility. Thus the presence of any moving target is detected and reported, very useful for environmental watch applications and military applications.

***Evaluation*:** The proposed work is a well distributed programming support for environmental tracking. However its performance is based only on very small scale implantation and it is at its early stage of development. More work need to be done in terms of self-organization and autonomic system approach

- **REED-Rule Based Middleware**

Fei, and Magill (2008) proposed a middleware solution that supports rule execution and event distribution (REED). REED supports both the distribution of rules and the events that trigger them. REED employs a rule-based paradigm to allow sensor networks to be programmed at run time. This provides a flexible environment where applications and users can program the sensor nodes to allow their behaviors to be changed at run time. Furthermore, the behaviors of the WSN's are described using descriptive rules and thus no knowledge of code programming is needed for the developers[43].

- **MoMi**

MoMi uses a rule-based system to detect faulty nodes in a wireless sensor network. MoMi uses a Model-Based Diagnosis (MBD) framework to present the likely causes of system abnormalities to an administrator. Observations about local and surrounding nodes are compared with each other based on predefined rules, after which conflicting observations are sent to a gateway. The gateway in turn generates a prediction of possibly faulty nodes[44].

- **STONE(STructural health mOnitoring NEtwork**)

STONE[48] is generic and allows for many application-specific customizations. Moreover, the software design allows flexible coupling of node software and the backend application with an embedded RPC mechanism. STONE exploits concepts for system reconfiguration and application partitioning between the sensor network and the backend software. The reconfiguration technique developed in STONE, allows to overcome the limitations of control message loss, latency and safe updates of variables and RPC execution.

- **Macro programming**

Kairos :

Kairos allows programmer to programming whole sensor network, make a high level specification, and dealing with low-level concerns at each node in network. At beginning, developers write a centralized program for whole application, Kairos preprocessor divided it into subprograms and Kairos compiler compiled them into annotated binary codes. After that the binary codes are distributed to sensor nodes. Annotated binary code is node-specific version contains code to control behavior of each single node. A set of nodes work together to express as a macro level program abstraction. Kairos can decide to use loose node synchronization or tight node synchronization base on programmer's purpose[39][13].

In[31], TinyOS is extended to allow dynamic exchange of components in WSN applications by conserving their modularity during the compilation process. This generates the possibility of incremental adaptation of sensor nodes' behavior through partial code replacement. The designed system does not require any alterations in the existing user interfaces, remaining transparent to the user.

Evaluation: This approach imposes almost no performance overhead for loaded application while keeping a smaller memory footprint than other comparable solutions.

- **KSPot+**

In [37], an architectural design of KSpot+ is presented. KSpot+ is a distributed middleware framework that introduces network-awareness to the data acquisition process by combining three cooperating components:

i) The Tree Balancing Module, which balances the workload incurred on each sensor node by constructing efficient network topologies;

ii) The Workload Balancing Module, which minimizes data reception inefficiencies by synchronizing the waking windows of each sensor node.

iii) The Query Processing Module, which manages query execution and additionally employs a ranking mechanism that unveils only the k-highest ranked answers thus further minimizing energy consumption.

Evaluation : KSpot+ provides mechanisms for measuring and improving the efficiency of the network topology in order to decrease the cost of data acquisition. KSpot+ employs a ranking mechanism that unveils only the k-highest ranked answers for minimizing energy consumption.

VI. SURVEY FINDINGS

From our survey, we have found that an existing mechanisms of Runtime-reconfigurations need to address various challenges such as Portability, Application Knowledge, Security, Scalability, QoS and Heterogeneity. We have given a comparative study of existing middleware's based on their features, performance and limitations in Table 1. Due to these constraints the proposed mechanisms works for heavy weight reconfigurations. Although the Reconfiguration challenges are addressed by existing code update mechanisms, their support is not totally satisfactory. A few proposed solutions assume that sensor nodes are deployed in homogeneous and resource constrained environments.

At the time of Emergency, the measurement of sensor nodes may vary as compared to the normal scenario. In such cases, the middleware's should be designed to tune at different operational level. The Runes Middleware is designed to adjust to the different operational level during the Emergency. In ReWiSe, , a component is reconfigured at the behaviour-level instead of at the component-level model for achieving *lightweight* and *fine-grained* software reconfiguration in WSNs. KSpot+ is an energy-efficient data-centric middleware architecture for measuring and improving the efficiency of the network.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 2, February 2015

Though many of the proposed systems offers excellent services, there is always a trade-off between Energy overhead and Reprogramming Rate. They try to overcome the aforementioned challenges, it is not possible for all of them to be available simultaneously on the relatively resource constrained environment. The Reconfiguration mechanism should be designed to avoid reboot of system each time when new updates are received. There is a need to explore light weight algorithms for Localization, Routing and Configure Management in WSN.

Table 1. Comparison of Design issues of Middleware Approaches

Approaches	**Application Openness**	**Scalability**	**Heterogeneity**	**Mobility**	**Ease of Use**	**Power Efficiency**
Virtual Machine Approach[12]						
Mate[14]	Weak	Strong	Medium	Strong	Weak	High
MagnetOS[26][12]	Strong	Strong	Medium	Strong	Weak	High
Data Base Inspired Middleware[12]						
TinyDB[16]	Medium	Medium	Medium	Medium	Strong	High
Cougar[13]	Weak	Weak	Weak	Weak	Strong	Medium
Event Based Middleware [17]						
MIRES[57][12]	Strong	Strong	Medium	Medium	Strong	High
WMOS[18]	Medium	Strong	Strong	Strong	Strong	High
DSWare[56]	Weak	Strong	Strong	Weak	Strong	High
Application Driven[12]						
MILAN[38][13]	Medium	Strong	Weak	Weak	Strong	Medium
Modular Programming[33]						
IMPALA[23]	Strong	Strong	Weak	Strong	Strong	High
FlexCup[7]	Medium	Medium	Weak	Strong	Weak	Medium
Componenet Based Middleware						
Runes[20]	Medium	Strong	Strong	Strong	Strong	Medium
Rewise[19]	Weak	Medium	Strong	Medium	Medium	Medium
Wisekit[25]	Strong	Strong	Strong	Strong	Strong	High
FIGARO[22]	Medium	Strong	Strong	Medium	Medium	Medium
REMOWARE[24]	Strong	Medium	Strong	Medium	Strong	High
Distributed Middleware						
KSpot+[37]	Strong	Strong	Strong	Strong	Strong	High

VII. CONCLUDING REMARKS

Due to resource constraints, the sensor nodes are originally deployed for limited applications. But, There is a need for self configurable framework which works consistently in both resource constrained and resource rich environment. In this paper we have summarized research in Run-time Reconfiguration techniques for sensor networks, focusing on

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 2, February 2015

representative approaches. We started with a taxonomy of Code Update mechanism for sensor nodes. We then presented a table giving a comparative study of existing middleware's based on their features, performance and limitations. From our survey, we have found that existing approaches of Runtime-reconfigurations are not totally satisfactory as they are designed for specific requirements. Several methodologies and platforms may be chosen depending on the type of application and organization of sensor network.

Our future research activities concentrate on the realization of the light weight algorithms for Run-Time Reconfigurations in WSN. The Self configurable model should be designed so as to simplify the development of sensor application, node application , and sensor network applications. The other possible future work is exploiting the Software Reconfigurations in an embedded system environments.

REFERENCES

1. S. Brown, C.J. Sreenan, "Updating Software in Wireless Sensor Networks: A Survey", Technical Report UCCCS20061307, Ireland 2006.
2. Alex Rogers, Esther David, and Nicholas R. Jenning," Self-Organized Routing for Wireless Microsensor Networks", IEEE TRANSACTIONS
3. ON SYSTEMS, MAN, AND CYBERNETICS—PART A: SYSTEMS AND HUMANS, VOL. 35, NO. 3, MAY 2005.
4. J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan, "Building Efficient Wireless Sensor Networks with Low-Level Naming". In SOSP 2001, Lake Louise, Ban_, Canada, October 2001.
5. John Heidemann, Fabio Silva, and Deborah Estrin ".Matching Data Dissemination Algorithms to Application Requirements". Technical Report ISI-TR-571, USC/Information Sciences Institute, April 2003.
6. Jan Blumenthal, Matthias Handy, Frank Colatowski, Marc Haase, Dirk Timmermann, **"**Wireless Sensor Networks - New Challenges in Software Engineering", 0-7803-7937-3/03 02003 IEEE.
7. Pankaj Nagar,Kripa Shanker Chaturvedi, " Study of Wireless Sensor Networks as a Software Engineering Approach", Proceedings of the World Congress on Engineering 2011 Vol II, WCE 2011, July 6 - 8, 2011, London, U.K.
8. Marr´on, M. Gauger, A. Lachenmann, D. Minder, O. Saukh, and K. Rothermel. "Flexcup: A flexible and efficient code update mechanism for sensor networks". In Proceedings of the Third European Workshop, EWSN 2006, Zurich, Switzerland, 13–15 February 2006; pp. 212–227.
9. Iqbal, M.M., Gondal, I. Dooley, L".LACON: localized autonomic configuration in ervasive sensor networks", *Dec 2004 (ISSNIP04), Melbourne Australia*, pp 31-36.
10. Tanya Roosta, Mike Menzo, Professor Shankar Sastry, "Probabilistic Geographic Routing Protocol for Ad Hoc and Sensor Networks", Ad-hoc Networks, 2005 - robotics.eecs.berkeley.edu.
11. Christian Frank, Kay R¨omer, **"**Algorithms for Generic Role Assignment in Wireless Sensor Networks"**,** *SenSys'05,* November 2–4, 2005, San Diego, California, USA. Copyright 2005 ACM 1-59593-054-X/05/0011**.**
12. J.Radhika 1 and S.Malarvizhi **"**Middleware approaches for Wireless Sensor Networks: An overview", IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 6, No 3, November 2012, ISSN (Online): 1694-0814.
13. Salem Hadim, Nader Mohamed , "Middleware: Middleware Challenges and Approaches for Wireless Sensor Networks", IEEE DISTRIBUTED SYSTEMS ONLINE 1541-4922 © 2006 Published by the IEEE Computer Society Vol. 7, No. 3; March 2006.
14. Shuai Tong, "An Evaluation Framework for middleware approaches on Wireless Sensor Networks". Helsinki University National Conference on Emerging Trends on Engineering and Management ISBN: 978-81-926416-0-7 Page 7 of Technology, TKK T-110.5190 Seminar on Internetworking 2009-04-27.
15. P. Levis, and D. E. Culler. "Maté: a Tiny Virtual Machine For Sensor Networks," *Architectural Support for Programming Languages and Operating Systems*, 2002. URL: http://www.cs.berkeley.edu/~pal/pubs/mate.pdf (accessed in February 2006).

16. P. J. Koopman. "Modern Stack Computer Architecture," System Design and Network Architecture Conference, 1990. URL:
17. http://www.ece.cmu.edu/~koopman/forth/sdnc90b.pdf (accessed in February 2006).

18. SAMUEL R. MADDEN, "TinyDB: An Acquisitional Query Processing --System for Sensor Networks",ACM Transactions on Database Systems, Vol. V, No. N, Month 20YY,
19. Bartolom´e Rubio, Manuel D´ıaz and Jos´e M. Troya, "Programming Approaches and Challenges forWireless Sensor Networks" , Second International Conference on Systems and Networks Communications (ICSNC 2007) 0-7695- 2938-0/07 IEEE.
20. Lifang Zhai1, Chunyuan Li2, Liping Sun, "Research on the Message-Oriented Middleware for Wireless Sensor Networks", JOURNAL OF COMPUTERS, VOL. 6, NO. 5, MAY 2011.
21. Amirhosein Taherkordi, Frank Eliassen, Romain Rouvoy, and Quan Le-Trung, "ReWiSe: A New Component Model for Lightweight Software Reconfiguration in Wireless Sensor Networks",OTM 2008 Workshops,LNCS 5333,pp-415-525,2008,Springer-Verlog,Berlin-Heidelberg,2008
22. Geoff Coulson#, Richard Gold†, Manish Lad†, Cecilia Mascolo†, Luca Mottola_, Gian Pietro Picco_, and Stefanos Zachariadis," Dynamic Reconfiguration in the RUNES Middleware", *Public Demonstration in Proceedings of the 3rd IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS06),* Vancouver (Canada), October 2006.
23. Amirhosein Taherkordi1, Quan Le-Trung1, Romain Rouvoy1,2, and Frank Eliassen1, **"**WiSeKit: A Distributed Middleware to Support Application-Level Adaptation in Sensor Networks", The 9th IFIP international conference on Distributed Applications and Interoperable Systems (DAIS'09), LNCS vol. 5523, 44-58, Lisbon, Portugal, June 9-12, 2009.
24. Mottola, L., Picco, G.,and Sheikh, A. "FiGaRo: finegrained software reconfiguration for wireless sensor Networks". In: Proc. of the 5th European Conference on Wireless Sensor Networks (EWSN), (2008),— Bologna, Italy, LNCS vol. 4913, 286-304.
25. Admilson R. L, Fabio C.S. Silva, Lilian C. Freitas, João Crisóstomo, Carlos R. Francês," SensorBus: A Middleware Model for Wireless Sensor Networks"**,** Proceedings of the 3rd international IFIP/ACM,2005.

26. AMIRHOSEIN TAHERKORDI,FREDERIC LOIRET,ROMAIN ROUVOY,INRIA Lille and FRANK ELIASSEN , "Optimizing Sensor Network Reprogramming via In-situ Reconfigurable Components", ACM Journal Name, Vol. 9, No. 2, 05 2013, Pages 1–37.
27. Amirhosein Taherkordi1, Romain Rouvoy1,2, Quan Le-Trung1, and Frank Eliassen1, **"**Supporting Lightweight Adaptations in Context-aware Wireless Sensor Networks, Author manuscript, published in ",1st International COMSWARE Workshop on Context-Aware Middleware and Services (CAMS) 385 (2009)", *CAMS 2009*, June 16, Dublin, Ireland Copyright © 2009 ACM 978-1-60558-525-3/09/06... DOI : 10.1145/1554233.1554244.
28. R. Barr , et al., "On the Need for System-Level Support for Ad hoc and Sensor Networks," *Operating Systems Rev.*,vol. 36, no. 2, 2002, pp. 1-5.
29. Kavi Kumar Khedo† and R.K. Subramanian††, "A Service-Oriented Component-Based Middleware Architecture for Wireless Sensor Networks", IJCSNS International Journal of Computer Science and Network Security, VOL.9 No.3, March 2009.
30. Apolinar González, Walter Mata, Luis Villaseñor, Raúl Aquino, Magaly Chávez, Alfons Crespo, (2009). ―"µDDS: A Middleware for Real-Time Wireless Embedded Systems‖", UNESCO code 120325;330417;120314;120305.
31. Jagun Kwon, Stephen Hailes," MIREA: Component-based middleware for reconfigurable, embedded control applications", In *Proceedings of the IEEE International Symposium on Intelligent Control* (ISIC,2010).
32. Adam Dunkels, Niclas Finne, Joakim Eriksson, Thiemo Voigt, "Run-Time Dynamic Linking for Reprogramming Wireless Sensor Networks"**,** SenSys'06, November 1.3, 2006, Boulder, Colorado, USA. Copyright 2006 ACM 1-59593-343-3/06/0011.
33. Waqaas Munawar, Olaf Landsiedel, Muhammad Hamad Alizai, Klaus Wehrle, "Remote Incremental Adaptation of Sensor Network Applications", Proceedings of the 8th GI/ITG KuVS Fachgespräch Wireless Sensor Networks, 2009.
34. Waqaas Munawar*‡, Muhammad Hamad Alizai†, Olaf Landsiedel†, Klaus Wehrle†," Dynamic TinyOS: Modular and Transparent Incremental Code-Updates for Sensor Networks", This work was conducted while the author was working with the Distributed Systems Group at RWTH Aachen University.
35. Md.Atiqur Rahman, "Middleware for wireless sensor networks :Challenges and Approaches", Helsinki University of Technology,marahma1@cc.hut.fi, TKK T-110.5190 Seminar on Internetworking 2009-04-27.
36. J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. "System architecture directions for networked sensors". In Proceedings of the ninth internation conference on Architectural support for programming languages and operating systems, 2000.
37. J. W. Hui and D. Culler, "The dynamic behavior of a data dissemination protocol for network programming at scale," in Proceedings of the 2nd international conference on Embedded networked sensor systems. ACM, 2004, pp. 81–94.
38. Mohammad M. Molla and Sheikh Iqbal Ahamed ," A Survey of Middleware for Sensor Network and Challenges", Proceedings of the 2006 International Conference on Parallel Processing Workshops,IEEE.
39. Panayiotis G. Andreou, Demetrios ZeinalipourYazti,George Samaras, Panos K. Chrysanthis,," Towards a Networkaware Middleware for Wireless Sensor Networks"**,** *8th International Workshop on Data Management for Sensor Networks (DMSN 2011)* Copyright 2011.
    A. Murphy and W. Heinzelman." Milan: Middleware linking applications and networks", Technical report #795, January 7, 2003 2002.
40. S. Hadim and N. Mohamed. "Macro-programming wireless sensor networks using kairos", In International Conference on Distributed Computing in Sensor Networks, 2005.
41. P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler, **"**TinyOS: An Operating System for Sensor Networks", to appear in Ambient Intelligence, Jan Rabaey ed.
42. Rafael Garcia, Ann Gordon-Ross, and Alan D. George," Exploiting Partially Reconfigurable FPGAs for Situation-Based Reconfiguration in Wireless Sensor Networks", Conference: Field-Programmable Custom Computing Machines - FCCM , pp. 243-246, 2009.
43. T. Abdelzaher , B. Blum and Q. Cao and D. Evans and J. George , S. George, T. He and L. Luo, S. Son, R. Stoleru,, J. Stankovic and A. Wood., "EnviroTrack: Towards an Environmental Computing Paradigm for Distributed Sensor Networks", ICDCS, 2004.
44. Xiang Fei*, Zixi Yu, "Development of a Rule Based Wireless Sensor Network Middleware", Proceedings of the 16th International Conference on Automation & Computing, University of Birmingham, Birmingham, UK, 11 September 2010.
    A. De Jong, M. Woehrle, and K. Langendoen. "MoMi: model-based diagnosis middleware for sensor networks". In Proceedings of the 4th International Workshop on Middleware Tools, Services and Run-Time Support for Sensor Networks, pages 19–24. ACM, 2009.
45. E. Souto , et al., "A Message-Oriented Middleware for Sensor Networks,", *Proc. 2nd Int'l Workshop Middleware for Pervasive and Ad-Hoc Computing* (MPAC 04), ACM Press, 2004, pp. 127-134.
46. Y. Zhu, S. L. Keoh, M. Sloman, and E. C. Lupu, "A lightweight policy system for body sensor networks", IEEE Transactions on Network and Service Management, vol. 6, no. 3, pp. 137–148, 2009.
47. G. Russello, L. Mostarda, and N. Dulay, "A policy-based publish/subscribe middleware for sense-and-react applications," J. Syst. Softw., vol. 84, pp. 638–654, April 2011.
48. Kallirroi Flouri*1, Olga Saukh2, Robert Sauter3, Khash Erdene Jalsan,Reinhard Bischoff , Jonas Meyer and Glauco Feltrin, A versatile software architecture for civil structure monitoring with wireless sensor networks, Smart Structures and Systems, Vol. 10, No. 3 (2012) 209-228.
49. Bulusu N., Heidemann J., Estrin D., "GPS-less Low Cost Outdoor Localization for Very Small Devices. IEEE Personal Communications Magazine," IEEE Personal Communications Magazine, Vol. 7, No. 5, pp 28-34. October, 2000.
50. Bulusu N., Bychkovskiy V., Estrin D., Heidemann J., "Scalable, Ad Hoc Deployable RF-based Localization," Proceedings of the Grace Hopper Celebration of Women in Computing Conference 2002, Vancouver, British Columbia, Canada. October 2002.

52. Savarese C., Rabaey J. M., Beutel J., "Locationing in Distributed Ad-Hoc Wireless Sensor Networks," Proceedings of IEEE International conference on Acoustics and Speech and Signal Processing ICASSP,Salt Lake City, Utah, USA(Volume 4,pp2037-2040, May 2001.
53.] K. Geihs, „Middleware Challenges Ahead", IEEE Computer, Juni/2001, S. 24-31. http://www.di.inf.pucrio.br/~rcerq//semGSD/sugestoes/-r6024.pdf . [Accessed Jan. 2008]
54. Frank Golatowski, Jan Blumenthal, Matthias Handy, Marc Haase," Service-Oriented Software Architecture for Sensor Networks"**,** Rostock, Germany: Institute of Applied Microelectronics and Computer Science, University of Rostock.
55. Pascal von Rickenbach and Roger Wattenhofer, Decoding Code on a Sensor Node, *J. ACM, no. 21, pp. 168–173,* USA, 2007
56. JAMES J. HUNT,KIEM-PHONG VO,WALTER F. TICHY," Delta Algorithms: An Empirical Analysis" ACM Transactions on Software Engineering and Methodology, Vol. 7, No. 2, April 1998, Pages 192–214.

57. S.Li, S.H. J.Stankovic, "Event Detection Services using Data Services Middleware in Distributed Sensor Networks" Proceedings of the Workshop on Information Processing in Sensor Networks(IPSN'03),Palo Alto,CA,Apr.2003.
58.] Shuoqi Li, Ying Lin, Sang H. Son, John A. Stankovic, and Yuan Wei, Event Detection Services Using Data Service Middleware in Distributed Sensor Networks, Telecommunication Systems, 2004 – Springer.

BIOGRAPHY

Rekha.K.S. holds Master's degree (2007) in Software Engineering, from Visvesvaraya Technological University, Belgaum. Currently she is working as an Assistant Professor in the Department of CS&E, The National Institute of Engineering, Mysore. She is pursuing her Ph.D. Her research interests include Wireless Sensor Networks, Middleware design, Software Engineering ,Software Architecture, Embedded Systems and Distributed Systems.

Dr.T.H.Sreenivas holds a Master's Degree(1986) from IIT,Khanpur and Ph.D (1999) from IIT,Madras. Currently he is working as a Professor in the Department of Information Science & Engineering, The National Institute of Engineering, India. His research interest includes Operating systems, kernel development and programming, real-time kernels, Real-time operating systems and Wireless Sensor Networks.