Programming Models for Wireless Sensor Networks: Status, Taxonomy, Challenges, and Future Directions

Abrar Alajlan and Khaled Elleithy

Abstract— Wireless sensor networks (WSNs) play an important role in different application areas and have been successfully deployed in different computing environments. However, programming sensor network applications is extremely challenging as the applications becoming more complex. Some of these challenges are due to the sensors' characteristics and others are due to the operating conditions of these sensors. Recently, researchers have proposed diverse programming approaches to mitigate these challenges and make WSN programming more flexible and much easier.
This paper provides an extensive survey of the state-of-art in wireless sensor network programming models, focuses on a classification of programming levels in wireless sensor networks and capturing some likely programming challenges and research future directions.

Index Terms— Sensor network,Wireless sensor network (WSN), Programming approaches, Macroprogramming, Wireless sensor networks taxonomy, Evaluation,Programming challenges.

1 Introduction

Typically, wireless sensor networks are composed of tiny embedded devices, each of which has radio transceiver to send or receive packets, processor to schedule and perform tasks, and power source to provide energy for the sensor [1]. Wireless sensor networks applications contain a large number of sensor nodes used to transmit and forward data between sensing nodes and the sink or base station [2]. Most often, WSN is utilized for the ease of deployment and enhanced flexibility of the network. Furthermore, it supports low cost dense monitoring of hostile environments as well as disaster relief, medical care and military surveillance [3].

The advantage of being able to place remote sensing nodes without having to run wires and the cost related to it is a huge gain. As the size of the circuitry of WSNs is becoming smaller along with the lower cost, the chances of their field of applications are significantly growing [4]. Most sensors, depending on the requirements, are battery powered and hence conserving the energy of these sensors is very crucial. Several programming approaches have been proposed to assist WSNs programming. Two broad classes of WSNs programming models have been explored lately; local behavior and global behavior abstraction [5]. In local behavior abstractions, the application has to be programmed in details at the node-level and the programmers need to synchronize the program flow between the sensing nodes and maintain the routing code manually. In contrast, global behavior abstractions or equivalently "High-level abstraction" has emerged as one of the most important aspects in sensor networks where it is applied to hide the internal operations from system programmers. The main objective behind high-level approach is the ability to treat a group of sensors or the entire network as one single unit rather than programming each node individually [6].

The main contribution of this work is to provide an extensive survey on taxonomy of programming approaches for wireless sensor networks. Our work also captures the programming requirements and uses them to evaluate each of the programming models. This paper also covers some open problems and challenges that need further investigation to make wireless sensor programming reaches its best level of performance and makes it highly usable and efficient.

Section II, identifies the requirements for sensor network programming. Section III, provides a taxonomy on programming approaches for WSNs. An in-depth look on each level of the programming approaches is presented in Section IV, V and VI. Analysis and evaluation of each model is discussed in Section VII. Section VII investigates research challenges and future direction of programming WSNs. Conclusion in provided in Section IX.

2 Requirements for Sensor network programming

It is obvious that sensor networks can be used in multiple applications that can be deployed in diverse environments. Moreover, it is very easy to modify the internal functionality of sensor networks to perform different tasks to support many sensor network applications. In this section we discuss important requirements for sensor network programming.

2.1 Scalability

Many sensor network applications deploy hundreds or even thousands of nodes collaborating to achieve desired goal(s); thus, scalability is one of the major designing attributes in sensor networks applications [7]. A scalable sensor network is representing the ability of the network to maintain its performance even when the network size has changed [8]. In WSNs scalability can be defined in two terms; size and geography. Scalability with respect to size states that if the application works properly with a few nodes, it can perform well with thousands of nodes. On the other hand, the scalability with

respect to geography is defined as the ability to perform correctly in different geographical areas under different environmental conditions [9]. Since we cannot predetermine the location of sensor nodes and we cannot assure the lifetime of sensor nodes, the programming model should help programmers in such a way to design scalable applications that are able to deliver accurate results [8]. The location information of distributed nodes needs to be known so as to exchange the sensed data between sensor nodes [10].

2.2 Localization

In wireless sensor network applications there are hundreds of nodes deployed in some areas such as underwater, in the middle of desert, or in inaccessible terrains, so their locations are random and unknown [11] , [12]. Thus, localization in sensor network, the determination of the geographical locations of sensors, is one of the important aspects for sensor network programming [13]. Many localization techniques have been proposed recently, either by deploying self-localized technique or by installing a Global Positioning System (GPS) device in each node to determine the exact location of the sensor node. Moreover, localizing algorithms can be classified into two groups:

- Range-based algorithms: where each node is equipped with hardware measurements, so the location of each sensor node can be determined by calculating the distance of the selected sensor node with its neighboring nodes [14].
- Range-free algorithms: where each node should determine its estimated location, and the ideal radio range of sensors.

Consequently, range-based algorithms provide more information compared to range-free algorithms; however, it is more expensive since there is some hardware measuring units attached to each sensor [15].

2.3 Failure-Resilience

Failure –resilience or (Fault-tolerance) is one of the most challenging requirements in programming wireless sensor networks [16]. Sensors are usually deployed in inaccessible terrains where people cannot reach the sensor nodes at that place. Some nodes might fail due to the resources limitation, hardware fault or it could be an intrusion from attackers. The failed sensors may lead to inefficient functioning of the network [17].

Thus, the system should keep performing properly even after unreliable communication, node failures, link failures, or unavailability of the network due to misbehaving nodes [18, 19]. Some techniques should be adapted to indicate that the node is not working in a proper way [20]. It could be done by monitoring the status of each node or using the power control technique [17, 19].

It is a very challenging requirement for the programmers to develop a sensor application that is resilient to failures and adaptive to the unexpected environmental changes which is too hard to provide error handling for every failure [18].

2.4 Energy-Efficiency

Energy efficiency is one of the most important issues in designing sensor networks. The overall design of sensor networks should mainly emphasize on enhancing the performance in terms of reduced power consumption. The total lifetime of a battery-powered sensor networks is limited by the non-rechargeable battery's capacity and each sensor node is equipped with a limited computation processor to perform its task [21]. Energy efficiency is very important factor in developing WSNs applications especially for continuous monitoring applications such as disaster monitoring, military surveillance and remote patient monitoring, etc. [22],[23].

2.5 Collaboration

Collaboration is another important characteristic of wireless sensor applications. WSNs applications have been growing recently. These applications vary in size and the number of nodes, from large scale networks to the small ones. All nodes in one application need to communicate in such a way so that the data from these sensors are gathered and analyzed. Thus, collaboration between sensor nodes is essential for these sensors to cooperatively and effectively work together to complete the desired tasks [24] [25].

Most of wireless sensor applications can be classified into two types:

- Data collection type: where all data is collected and sent to the main server such as habitat and environmental monitoring applications.
- Collaborative information processing: where the main task is to convert the data gained from multiple sensor nodes to higher-level information such as a tracking system applications [26].

Collaboration is not an independent requirement, it can support other requirements. For instance, collaboration between sensor nodes may reduce the failure-resilience where the sensing process remains functional even after one node failed. Moreover, collaboration inside each sensor group may reduce data transmission which is in turn will reduce the consumed power [18].

2.6 Time Synchronization

Time-synchronization between nodes is another essential requirement for sensor programming execution. Many WSNs applications such as tracking application and implementation of TDM requires a timer synchronization that is maintained at each sensor node [27].

Clock synchronization is a process used to ensure an accurate scheduling between nodes with no collision [28]. Moreover, WSNs have limited power as discussed earlier; therefore, time- synchronization technique helps to reduce the power consumption by passing some nodes off from time to time [29]. Clock synchronization in sensor nodes is generally required for many reasons such as:

- To support the coordination and collaboration between sensor nodes,
- To manage the sleep and active state for each node [30],
- To avoid collisions between sensor nodes as used in TDMA (Time division multiple access) [31], and
- To reduce the differences between the clocks that is attached to each node at any time [32].

3 Programming Approaches for WSNs: A Taxonomy

In this section we present a taxonomy of the programming approaches for WSNs. Figure 1 depicts the entire taxonomy that categorize the wireless sensor network programming approaches into low-level and high-level programming models. Low-level approach mainly focuses on the use of an existing programming language to provide flexible controls over nodes. TinyOS with nesC as will discuss later, is one of the well-known examples that falls into this subclass [33], [34].

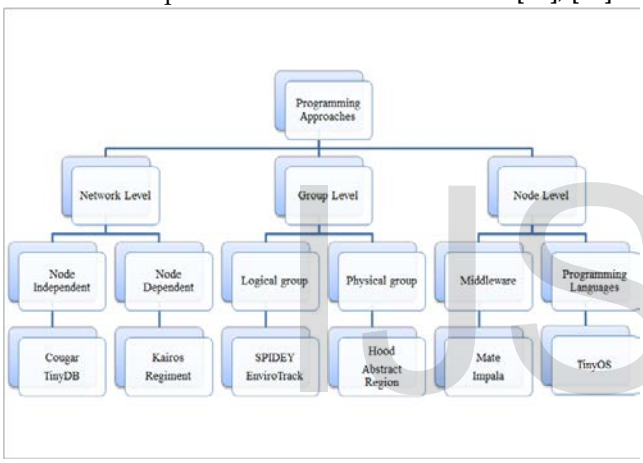


Fig. 1. Taxonomy of programming approaches for WSNs .

The virtual machine that runs on each node is one of the interesting approaches in this subclass. It is responsible for breaking tasks and dynamically distributing them to each node.

High-level programming approach mainly focuses on simplifying the collaboration between sensor nodes. One approach is to divide the whole network into a set of groups and treat each group as a single entity which is called "Group-level abstractions". It helps the programmer to describe collaborative algorithms easily. This approach is further divided into physical groups and logical groups. In physical group, the network can be grouped based on the physical location of the node, whereas the logical group is based on the shared properties among nodes.

The other approach of high-level abstraction is network level abstraction or "macroprogramming abstractions" where the whole network is treated as a single entity. It is an application centric-view, thus, it helps the programmer to focus on the programming logics rather than programming the platforms. Macroprogramming approach is divided into two subcategories; node-dependent and node-independent and we will cover each of them in the next sections.

3.1 Node-Level Abstraction

Node-level programming approach focuses on the use of an existing programming language and abstracting hardware to provide a flexible control over the node.

3.1.1 Programming Languages

Application development at the node level is basically relying on the use of an existing programming language. NesC and C are the most well-known programming languages that are used for tiny embedded systems [18].

NesC

NesC is a C based attached with a programming model that adds some features such as a flexible concurrency and oriented application design. NesC programming language uses a static memory allocation (variables are allocated at the compile time) to simplify the code and obtain an accurate result [34].

TinyOS

TinyOS as in [35], is a simple application specific operating system used in embedded WSNs. TinyOS applications is written in NesC programming language to limit the hardware resources and support operations structures needed by sensors [36]. It is one of the most popular operating systems that support several frameworks applications based on a tiny node connected with the microcontroller and sensors [33]. TinyOS is a graph of the following independent components:

- Module which provides interfaces for configuration,
- Configuration which is used to connect all component together

Each of which has three concepts:

- Commands are basically asking a component to perform some tasks.
- Tasks are performed internally at the component such as initiating a connection or reading data.
- Events are referring to the completion of that task.
-

One example of commands and events when initiating a sensor reading as in getData(). This command will cause a later signal dataReady() when data is obtained. These two concepts – command and event - are used between components, however; the tasks are performed internally at the component [37].

3.1.2 Middleware

The key concept behind using middleware is to support the overall performance of applications and to connect the application layer with hardware and operating systems as shown in figure 2 below. Middleware in WSN supports "reprogrammability" which is the ability to break tasks and distribute these tasks to each node dynamically [38].

Middleware helps applications programmer to focus on the programming logic without caring too much about the implementation details at the lower level. Moreover, middleware provides a reusable code, thus, the programmer can execute a new application without using complex and inefficient methods. Furthermore, it supports system monitoring and integration [39].

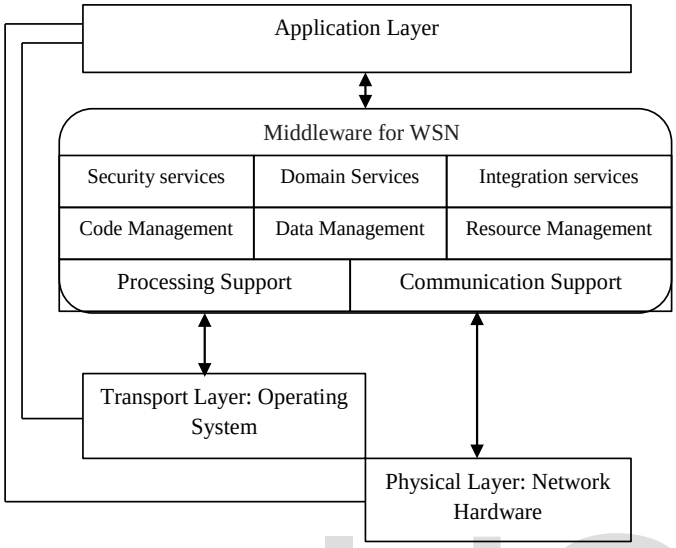


Fig. 2. Reference of wireless sensor networks middleware [38]

Virtual Machine

Virtual machine is one type of middleware where the application is written in small segments and then distributes these segments through the network using tailored algorithms. Therefore, the size of the code transmitted to each node is reduced and the communication amount between the server and each node is minimized as well. [40].

Mate [41], and ASVM [42] are stack oriented virtual machines that run on TinyOS. These interpreter-based virtual machines provide an application specific virtual machine which is employed to enhance the flexibility and offer efficient programming environments [18].

Mate

In Mate, the codes are broken into small fragments that are injected later into the network to build the total program. It has a scheduler to adopt a FIFO based queue of contexts and encloses them with their executions. It performs the execution by fetching the next byte code at the fragments store and attaches it to its corresponding operation. Mate uses a Trickle algorithm to improve the broadcasting speed and to reduce the cost when nodes propagate new data. [41]

Impala

Another example of middleware is Impala [43] which supports modularity, adaptability to rapidly change environments as well as the reprogrammability. This virtual machine is deployed to provide independent execution platforms where the programmers can use them to write their codes [43].

The system architecture of Impala is illustrated in Figure 3, where the lower layer holds ZebraNet application protocols and programs. These application protocols employ different techniques to gather environment information and send it to the base station in peer to peer transmission. The upper layer holds three middleware agents:

- Application Adapter where the application is adjusted for various conditions in order to enhance the overall performance, and energy efficiency.
- Application Updater is used to install the software updates on each node.
- Event Filtering captures events to initiate a sequence of operations.

In Impala there are five different types of events; Timer Event, Packet Event, Send Done Event, Data Event, and Device Event. To eliminate the programming complexity of synchronizing two different handlers, the system will handle them sequentially in case they arrived at the same time. [43].

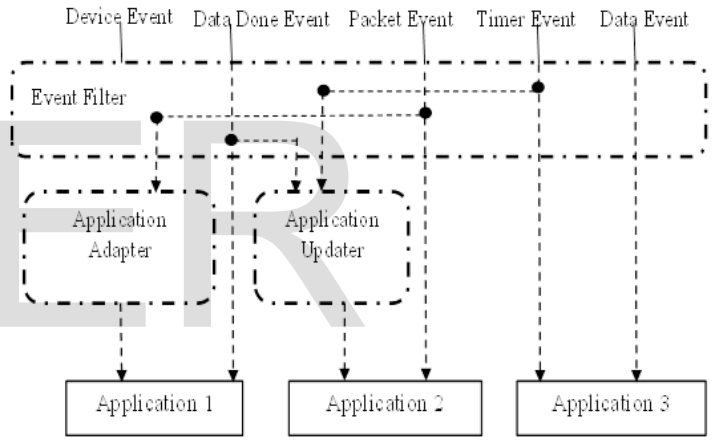


Fig. 3. system architecture of Impala

3.2 Group-Level Abstraction

The main concept behind a group-level abstraction in WSNs is to divide the whole network into small groups and perform computations on those groups instead of dealing with each single node. In a group- level abstraction, the network can be grouped based on the physical locations of the nodes (Neighborhood Based) or it can be grouped logically [44].

3.2.1 Physical Group

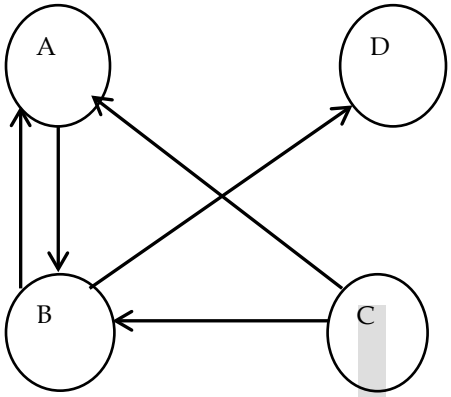
The notion of physical group or " neighborhood based group " is basically a node with its neighbor's without paying any attention to the properties of these nodes [18]. This technique is used to hide the communication details between the nodes and it can be used in "localized algorithms" where the interaction between participating nodes is limited to their neighbors as in [45].

Hood

Hood is one example of a neighborhood- based programming abstraction where a given node is limited to communicate and share data with neighboring nodes only. This physical close-

ness is determined by the physical distance or the number of hops between sensor nodes [46].

In Hood, all nodes in each group have to be in the same network and if one node moves to another network then it is not a member of that group. Figure 4, describes how a node becomes a neighbor of other node. Node A, receive the data reported by node B and C, while it reports its reading to node B. Also, B is a neighbor of D but D is not in B's neighborhood. A node can receive data from its neighbors and its own data is send to its co-neighbor. One node can be assigned to read location over one group, and read the temperature over other group. To manage the complexity of these tasks, Hood provides an interface to read the shared values of each neighbor [47]



Node	*Neighbors*
A	*C ,B*
B	*A,C*
C	*None*
D	*B*

Fig. 4. The Definition of Neighborhood in Hood

Abstract Region

Another example of a neighborhood-based group abstraction is Abstract Region which relies on the concept of grouping the nodes in mesh, spanning tree or could be based on the geographic locations of these nodes [48]. Abstract Regions as in Hood, cannot group nodes from different network. Moreover, this model can be adapted within different network conditions to attain different levels of energy and bandwidth usage as well as the accuracy level of shared operations. Also, each region is separated from other regions and requires a specific implementation. Abstract Regions can be implemented by following these phases:

- **Discovering Neighbors**

In this phase, a node starts to discover its neighbors by either sending a broadcast message or gathering the location of each sensor node in the network. Since sensor nodes might move from one network to another, this phase is a continuous process to update any change in one region. However, a node has the ability to deactivate this process at any given time to reduce the power consumption when sending discovery messages.

- **Enumeration**

This phase is used to address all nodes in one region and return the location of each of the nodes to help the participating nodes in one region to communicate directly.

- **Data sharing**

At this phase, all shared variable between nodes are represented as pair of key and value.

- **Reduction**

This phase is used to cut the shared variables across nodes in one region and it is hidden from programmers [48].

3.2.2 Logical group

A logical group abstraction can be defined as a set of nodes that share the same properties in sensor networks such as node types, sensor inputs, or perform the same tasks [18]. Unlike neighborhood based, the logical group is considered to be a dynamic group since it is based on the shared properties and not limited by the physical location of nodes [49].

Logical group-based, cannot cross multiple networks at the same time which means we cannot reuse the existing sensors without reprogramming them [50].

EnviroTrack

One example of Logical based group is EnviroTrack. It is an application used to track programs where a set of nodes that detect the same event are grouped together [51].

SPIDEY

Another example of logical based group is a SPIDEY language where a set of nodes are grouped based on their shared properties [46]. In SPIDEY language, each node has both static and dynamic attributes which are used to determine the nodes logical neighbors as in [46]. SPIDEY delivers communication APIs, where a broadcasted message is sent to a logical neighborhood instead of nodes that fall in the same communication range. This technique helps programmers to clearly specify the communication range and which nodes to select as a neighbor. All logical neighbors are considered to be one group with functionally related characteristics.

Thus, a group-based abstraction makes programming sensor network model simpler since it performs in group level instead of node level [44]. However, this approach is mainly designed for applications that operate in a single network since we cannot across multiple networks at the same times [50].

3.3 Network-Level Abstraction

Several macro-programming abstractions have been introduced recently. Macro-programming systems or equivalently "networking abstractions" considered to be high-level WSN programming model where the whole sensors network is treated as a single system [18].

This approach helps the programmers to emphasize on improving the semantics of the program instead of studying the characteristics of the programming environments [18].

There are two different major classes of network-level abstractions. One is a node-dependent abstraction which focuses on enabling the programmers to define the global behavior of the system as a collection of nodes that can be treated simultaneously in one program. In contrast, node-independent approach defines the system in independent way as single unit [5]

3.3.1 Node Dependent Approach

Node-dependent approach is intended to deliver more flexibility than node independent. This approach allows programmers to define the global behavior of the computation in terms of nodes and their states [53].

Kairos

Kairos is a node-dependent abstraction where the neighboring nodes can be computed in parallel and communicate using common requests at specific nodes [52]. Kairos has a centralized programming environment which is translated later by the compiler to many executable effective nodal programs [52]. Kairos enhances the use of sensor programming languages by providing three simple mechanisms. First, node abstraction, where the programmer deploys network nodes explicitly and names each node with an integer identifier, yet these integer identifiers do not reflect the structure of the sensor node. Hence, there is no need for the programmer to specify the network structure when using Kairos [52]. Second, is the identification of one-hop neighbors, where the programmers are able to use get neighbors function to support wireless communication between the nodes. When get function is called, a list of neighbors nodes are returned, so the calling node can select which node to communicate with. Third abstraction is accessing data on a remote node which implies the capacity to access variables from selected node since Kairos does not restrict any remote access to the variable nodes [53].
Kairos implements an eventual consistency method; by adopting this feature the program is able to deliver the most accurate result even if an internal node is not assured to be reliable. Thus, Kairos can be used with many well-known programming languages such as python as in [52].

Regiment

Another example of node-dependent abstraction is Regiment, a purely microprogramming functional language that allows the direct use of program state [3]. However, it uses what is called monads; described in more detail elsewhere in [54]. In Regiment, programmers deploy groups of data stream or what is called signals. These signals used to represent the finding of each individual node. Regiment also provides the concept of region as in Abstract Regions [55], which can be used to enhance the logical relationship between the nodes and data sharing between sensor nodes. The compiler at Regiment converts the whole program into a form of simple readily program using token machine technique which is a very simple model to achieve internal sensing and able to receive signals from neighbor nodes [3].

Moreover, Regiment applies a multi-stage programming mechanism to support the use of different programming languages that are not maintained by the given program [55]. Also, Regiment enables the use of generics that qualify the program to pass any data types as in C++. It supports three polymorphic data types:

- Stream which represents the rapid changes in the nodes' states.
- Space which represents the real space with a given value of specific type.
- Event which represents the events that have values and happen at a specific time.

The concept behind streams and event is founded in Functional Reactive Programming (FRP); see [56] for more details. Since Regiment is completely functional language, the values of stream, event and space are treated as formal parameters where they can be returned from function and passed as arguments [3].

3.3.2 Node-Independent Approach

Node-independent approach or equivalently "Database approach" is one type of high-level abstractions for sensor network programming. This approach distributes the nodes in a network using independent way and does not have any obvious abstraction for nodes [53].

TinyDB

TinyDB as in [56], is a query processing system that mainly focuses on improving the energy consumption by controlling the tested data. The network is treated as one database system where users are able to retrieve information by using SQL-Like queries. This approach should adhere to what is called homogeneous network where all nodes must have same capabilities before testing to achieve the desire result. In TinyDB system, the sensed data are actually used as an input of sensor table and system user can access these entries by using SQL-like queries [52].

Cougar

Cougar is another example of node-independent abstractions that has been proposed early at Cornell University[57]. Cougar system is used to test for query processing in sensor networks [58]. Each Cougar system consists of three levels:

- Queryproxy, a tiny database element that runs in sensor nodes to track and perform system queries.
- Frontend element which is used to setup connections between sensor nodes in one network and other nodes in different networks.
- Graphical user interface (GUI) which is used to enable users to perform queries [58].

Cougar drives query to the end nodes and all the computations are achieved at the edge level to reduce the amount of transmitted data. It also helps the system users to retrieve the data and system behavior. However, it is too difficult to deal with complex applications like tracking system using this technique [59].
Although Node-independent abstraction delivers very simple user interface, it is still not suitable for applications that require a lot of control flow.

4 Analysis and Evaluation

In this section, we focus on the most important strategies that are used in each programming model to fulfil the programming requirements discussed earlier. A summary of how

each level of the programming approaches addresses these requirements is shown in the next three tables below. Table 1 below summarizes how a node-level abstraction addresses the programming requirements discussed in section 2. Programmers at this level are able to deploy some features to enhance the scalability of the program by using low-level interfaces. Even though, these interfaces are flexible, they tend to be complex in execution operations [51].

To maintain collaboration and synchronization in TinyOS, two components are used: configuration to connect all components together and module to perform the synchronous method as a FIFO queue. Impala uses timer event signals to manage the collaboration and synchronization between sensing nodes, whereas Mate installs concurrency manager and scheduler to maintain these requirements. Middleware examples listed above, deliver efficient mechanisms for system updates to support dynamic applications and offer a great energy saving [60].

TABLE 1
EVALUATION OF NODE-LEVEL MODEL FOR SENSOR NETWORKS

Evaluation Factor	Node-Level		
	Programming Language	***Middleware***	
	TinyOS/NesC	***Mate***	***Impala***
Scalability	Programmers implement each feature by using low-level interfaces. Flexible but tend to be complex.	Can express a wide range of applications	Can express a wide range of applications
Localization	Variable locations can be statically compiled into the program	Can be extended to perform a localization service.	Static location of communicating nodes
Failure-Resilience	Restrictions allow the nesC compiler to perform whole-program analyses such as data-race detection to improves reliability	Mate is concise programs that are resilient to failure. Ensures the resilient to buggy or malicious capsules.	Adaptation to device failures. Autonomic behavior which increases its fault tolerance
Energy-Efficiency	Restrictions allow the nesC compiler to perform whole-program analyses such as using aggressive function in lining to reduce resource consumption.	Efficient dynamic code update : Small interpreter code	Efficient dynamic code update. Eliminating duplicate components to be transmitted over network.
Collaboration	Use configuration to wire interfaces from several modules together.	Supported shared variables that managed by concurrency manager	
Time Synchronization	Use module, a part of the TinyOS for timer service.	A scheduler component to adopts a FIFO based queue	Timer Event signals attached.

Table 2. shows how the programming requirements are implemented in each programming model at the group level ab-

straction. Since all the programming models listed on table II are group based abstractions, the scalability, collaboration and data aggregation are supported through data sharing. Caching technique is used in several programming models at this level to reduce the communications between sensing nodes and helps to save energy [48, 47, 51].

Caching and abstract region are employed in Hood to improve the communication failures by replacing the failed data with the old cached one.

However, SPIDEY utilizes redundancy mechanism to avoid flooding the whole program and to limit the propagating of information [52]. There are some components or functions attached to each programming model to improve localization: Hood uses mirror to reflect node locations or time synchronization services [47]. In this case, abstract region starts with neighbor discovery where each node initiates the process of discovering the location of its neighbors [48]. As the tracked objects move in EnviroTrack, the location of participating nodes has to be known by using some functions like *Location*: avg (position) [51].

TABLE 2
EVALUATION OF GROUP -LEVEL APBSTRACTIONS FOR SENSOR NETWORKS

Evaluation Factor	**Group-Level**			
	Physical Group		***Logical Group***	
	Hood	***Abstract Region***	***EnviroTrack***	***SPIDEY***
Scalability	Supported through data sharing.	Supported through data sharing	Supported through data sharing etc.	Supported through data sharing etc.
Localization	Use mirrors to reflect location.	At Neighbor discovery stage discovering the location neighboring nodes	For tracking objects. As the tracked objects move, the location of have to be known.	Static physical location of each node should be identified when creating node.
Failure-Resilience	Caching : improves communication failures by substituting the data with old cached data	Caching: improve communication failures by substituting the data with old cached data	Dynamic group management and leader election	Utilize redundancy mechanism
Energy-Efficiency	Power consumption supported through data sharing.	Supported through data sharing Caching low-level control knobs.	through data sharing Caching ("freshness threshold")	Supports aggregation at group level through data sharing etc.
Collaboration	Asymmetric Group definition and operations on group neighbor	Group definition and operations on group	Group definition and operations on group Context label, dynamic group	Group definition and operations on group
Time Synchronization	Use mirror to create some services as time synchronization	Use timeout mechanism, and will fail if not completed within a given time.	Timer handler as an input to executes one iteration per invocation.	Contains a time-period attribute when creating each node.

Table 3. shows the evaluation of macroprogramming approach based on the programming requirements. The main approach to satisfy scalability is to reduce the communication between the sensor nodes. Cougar and TinyDB are the most

well-known examples of node-independent approach. They push the query selection at the edge (nodes) so the transmission data is reduced. Moreover, Cougar and TinyDB extend their SQL so that users can express continuous sensing tasks. Regarding to localization, Regiment provides the ability to divide the tested area to spatial regions to facilitate the localization and communication processes. Also, Regiment is resilient to failure where the master node or (Anchor) in each region is responsible to cover if a node fails or loses connectivity to others [3].

TABLE 3
EVALUATION OF NETWORK-LEVEL MODELS FOR SENSOR NETWORK

Evaluation Factor	**Network-Level**			
	Node-Dependent		***Node-Independent***	
	Kairos	***Regiment***	***TinyDB***	***Cougar***
Scalability	No evidence for support	Purely functional language. Permit the use of fold, map functions.	Network query processing. Queries selection at (nodes) to Reduce transmission data	Network query processing. Queries selection at (nodes). Reduce transmission data
Localization	Each node is only responsible for localizing itself	Use Region for the purpose of localizing sensing	Each node is only responsible for localizing itself	No evidence for support
Failure-Resilience	Eventual consistency	Anchor " leader" is an object persists across node failures	No evidence for support	No evidence for support
Energy-Efficiency	Caching	Purely functional language. Permit the use of fold, map functions	Acquisitional query processor changes sampling rate battery lasts for lifetime.	In network query processing.
Collaboration	Describe a resource access as a variable access. Implicitly express both distributed data flow and control flow.	Region streams Capable of expressing groups of nodes with geographical, and logical relationships	Collaboration can be defined through a query.	Collaboration can be defined through a query.
Time Synchronization	Automatically synchronizes nodes when a checkpoint is taken or restored.	Use signals to represent the finding of each individual node.	Nodes run a simple time synchronization protocol to agree on a global time base .	The data is appended at time intervals specified in the query termed as epochs.

5 Future Research Directions and Programming Challenges for Wireless Sensor Networks

Several programming approaches have been introduced and discussed in the past decades. However, there are many programming challenges still unresolved and need further study to make the WSNs programming valuable and effective ; thus, in this section we list some of them and discus the future direction of programming WSNs.

5.1 Reprogramming

The network programming requirements might change over time, and this change could be parameter changes or reprogram the entire system. Also, wireless sensors might move from one network to another, but the limited resources of these sensors may result in short-lived systems. Thus, sensing nodes should have a dynamic reconfiguration services to keep these sensors functional for a long time [61].

In order to create a useful and effective reprogramming system, some requirements need to be addressed. First, time and space complexity of reprogramming algorithm should correspond to the capacity of sensor node. Second, since the sensing nodes have limited energy resources, the reprogramming system should be energy-efficient. Third, reprograming requires delivering the code entirely even though communications over wireless network are unreliable [62].

5.2 Heterogeneity

In WSNs, the basic form of heterogeneity is deploying multiple different types of sensors in one application, each of which performs different task and has different energy and resources. Heterogeneity in a WSN is used to improve the overall reliability and lifetime of the network [40]. Heterogeneity in WSNs has two forms: physical heterogeneity and logical heterogeneity.

One example of physical heterogeneity is hierarchical architecture, where the upper level sensors are more powerful and have more energy and network resources than the lower ones. Physical heterogeneity in WSNs has three types [62]:

- Computational Heterogeneity: where some nodes have more computational power than others.
- Link Heterogeneity: where some sensors have long distance than others.
- Energy Heterogeneity: where some nodes have more energy resources than other nodes.

In contrast, logical heterogeneity is the case where each sensor has to behave in different way to perform a specific task assigned by the application [18]. One example of the logical heterogeneity is the usage of generic role scheme to assign one task for each sensor node. These roles are stated by a declarative configuration language; described in more details elsewhere in [64].

From programming point view, how to deploy heterogeneous sensors efficiently and how to program the entire system with these sensors are the main concerns in developing WSNs applications.

5.3 Quality of Service

Quality of service is one of the important challenges in designing wireless sensors applications. As stated earlier, wireless sensors are equipped with limited energy resources. Accordingly, system designers need to balance between energy consumed and some quality services such as accuracy and error rates to get efficient results with a satisfying quality. Quality is a very crucial element in designing sensor network application since there are certain actions will be taken according to the sensed result. For example, when detecting vulcanic eruptions or sensing earthquakes before they hit, to change the behavior accordingly or issue an emergency alert, lack of accuracy and large latency would make the application useless. If the information gained from the sensor network is inaccurate, it may ruin the entire application. Thus, the system designers should be able to maintain the overall efficiency level as well as the quality of collected data [18].

The above requirements and the demanding deployment environment of wireless sensors make sensor programming the most challenging task in developing wireless sensors applications. In spite of the considerable effort carried out to let WSN programming model reach its best level of performance, still there are several open problems that need further investigation to make wireless sensor programming highly usable and efficient.

6. Conclusion

In this paper, we have provided taxonomy of different programming levels in wireless sensor networks. Three different levels of programming approaches have been discussed: node level, group level and network level. Several examples have been covered and evaluated based on some programming requirements for each level. Designing efficient programming models for WSNs has many challenges to overcome such as reprogramming, heterogeneity, and quality of service. Still there are missing some qualities and features to let WSNs programming model reach its best level of performance.

REFERENCES

[1] M. Tubaishat and S. Madria, "Sensor Networks: An Overview," *IEEE Potentials*, vol. 22, no. 2, pp. 20–23, April/May 2003.

[2] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci,"A Survey On Sensor Networks," *IEEE Communications Magazine*, vol. 40, no. 8, Aug 2002, pp. 102–114

[3] R. Newton and M.Welsh, "Region Streams: Functional Macroprogramming for Sensor Networks," *Proceeedings of the 1st international workshop on Data management for sensor networks: in conjunction with VLDB*, 2004, pp. 78-87

[4] N.Marriwala and P.Rathee, "An Approach To Increase The Wireless Sensor Network Lifetime," *IEEE on Information and Communication Technologies*, Oct.2012, pp. 495-499

[5] S.Choochaisri, N. Pornprasitsakul, C.Intanagonwiwat, "Logic Macroprogramming for Wireless Sensor Networks," *International Journal of Distributed Sensor Networks*, 2012

[6] U.Bischoff and Kortuem, G., "A State-based Programming Model and System for Wireless Sensor Networks", *IEEE International Conference on Pervasive Computing and Communications*, March, 2007, pp. 19-23

[7] K. Akkaya and M. Younis, "A Survey on Routing Protocols for Wireless Sensor Networks," **Ad Hoc Networks**, vol. 3, no.3, May.2005, pp. 325–349

[8] M. Sousa, A. Kumar, M. Alencar , W. Lopes, "Scalability in An Adaptive Cooperative System for Wireless Sensor Networks," *IEEE International Conference on Ultra-Modern Telecommunications Workshops (ICUMT) ,* Oct. 2009

[9] R. Venkateswarlu and D. Janakiram, "A Simple Model for Evaluating The Scalability in Wireless Sensor Networks," *IEEE International Conference on Intelligent Sensors, Sensor Networks and Information Processing Conference,* Dec. 2005

[10] B.Warneke, et al., "Autonomous Sensing and Communication in A Cubic Millimeter "*IEEE*, vol.34, no.1 Jan 2001, pp. 44 – 51

[11] N. Patwari, J. N. Ash, S. Kyperountas, A. O. Hero, R. L. Moses, and N. S. Correal, "Locating The Nodes: Cooperative Localization in Wireless Sensor Network," *IEEE Signal Processing Magazine,* vol. 22, July. 2005, pp. 54–69

[12] Z. Chaczko, R. Klempous, J. Nikodem, M. Nikodem, J. Rozenblit, An Improvement of Energy Aware Routing in Wireless Sensors Network, *European Modeling and Simulation Symposium*, Barcelona, Oct. 2006

[13] T. Yasuhisa,"Node Localization for Sensor Networks using Self-Organizing Maps", *IEEE Topical Conference on Wireless Sensors and Sensor Networks (WiSNet),* Jan 2011, pp. 61-64

[14] S. Pandey, P. Prasad, P. Sinha and P. Agrawal, "Localization of Sensor Networks Considering Energy Accuracy Tradeoffs ", *IEEE International Conference on Collaborative Computing: Networking, Applications and Work-sharing*, 2005, pp. 1 – 10

[15] M.Rudafshani and S. Datta "Localization in Wireless Sensor Networks," *IEEE International Symposium on Information Processing in Sensor Networks,* April. 2007, pp. 51-60

[16] M.Bellalouna A. Ghabri, "A Priori methods for Fault Tolerance in Wireless Sensor Networks," *IEEE World Congress on Computer and Information Technology (WCCIT),* June. 2013

[17] L.M. Wang, J.F. Ma, C. Wang, and A.C. Kot, "Fault and Intrusion Tolerance of Wireless Sensor Networks," *IEEE International Parallel and Distributed Processing Symposium (IPDPS),* April. 2006

[18] R. Sugihara and R.Gupta, "Programming Models for Sensor Networks: A Survey," *ACM Transactions on Sensor Networks,* vol. 4, no. 2, Article 8, March 2008

[19] D.Geetha, N. Nalini, R. Biradar, "Active Node based Fault Tolerance in Wireless Sensor Network," *Annual IEEE India Conference (INDICON*), Dec. 2012, pp. 404 – 409

[20] V. Potdar, A. Sharif, E. Chang, "Wireless Sensor Networks: A Survey", *IEEE International Conference on Advanced Information Networking and Applications Workshops, AINA*, 2009, pp. 636- 641

[21] A. Alajlan, B. Dasari, Z. Nossire, K. Elleithy and V. Pande, "Topology Management in Wireless Sensor Networks: Multi-State Algorithms," *International Journal of Wireless & Mobile Networks (IJWMN),* vol 4, no. 6, Dec 2012 pp. 17-26

[22] D.Chaudhary, and L. Waghmare, "Energy Efficiency and Latency Improving Protocol for Wireless Sensor Networks," *IEEE International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Aug. 2013 pp. 1303 – 1308

[23] K. Vardhe ,C. Zhou, D. Reynolds, "Energy Efficiency Analysis of Multistage Cooperation in Sensor Network," *IEEE GLOBECOM,* Dec. 2010, pp. 1 - 5

[24] M. Huang, and Y. Hen Hu "Collaborative Sampling in Wireless Sensor Networks."*IEEE GLOBECOM* Dec.2010, pp. 1-5

[25] W. Li, J. Bao, and W. Shen," Collaborative Wireless Sensor Networks: A Survey," *IEEE SMC* Oct. 2011, pp. 2614 –2619

[26] F. Zhao, J. Shin, and J. Reich, "Information Driven Dynamic Sensor Collaboration," *IEEE Signal Processing Magazine*, vol. 19, no. 2. Mar 2002, pp. 61-72

[27] S. Lee, U. Jang and J. Park, "Fast Fault-Tolerant Time Synchronization for Wireless Sensor Networks," *IEEE ISORC* May.2008, pp. 178 – 185

[28] K. Yaguang, Z. Xifang, C. Huakui, " Intelligent Time Synchronization in Sensor Network", *IEEE International Conference on Wireless, Mobile and Multimedia Networks*, Nov. 2006, pp. 1 - 4

[29] S. Lasassmeh and J.Conrad, "Time Synchronization in Wireless Sensor Networks: A Survey," IEEE SoutheastCon, Mar. 2010, pp. 242 – 245

[30] H.Liming, and G.Kuo, "A Novel Time Synchronization Scheme in Wireless Sensor Networks," IEEE Vehicular Technology Conference, vol.2 May. 2006, pp. 568 – 572

[31] Q. Liu, J. Kuang, Z. Bi, H. Wang and N. Wu, "Time Synchronization Performance Analysis and Simulation of a kind of wireless TDMA Network," *IEEE International Frequency Control Symposium and Exposition*, June.2006, pp. 299 – 303

[32] K.Yıldırım and A. Kantarci, "External Gradient Time Synchronization In Wireless Sensor Networks, "*IEEE Transactions on Parallel and Distributed Systems*, vol.PP, no.99, Mar

2013, pp. 1-10

[33] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors," *SIGARCH Comput. Archit. News*, vol. 28, no. 5, Nov. 2000. pp. 93-104

[34] D. Gay, P.Levis, R.Behren, M.Welsh, E.Brewer, and D.Culler, "The NesC Language: A Holistic Approach to Networked Embedded Systems," *ACM SIGPLAN Conference On Programming Language Design And Implementation,* 2003, pp.1-11

[35] TinyOS. http://www.tinyos.net/.

[36] J. Guevara, E. Vargas, F. Brunetti, F. Barrero, L. Aranda, "A Framework for WSN using TinyOS and The IEEE1451 Standard," *IEEE Latin-American Conference on Communications (LATINCOM),* Oct. 2011, pp. 1 – 5

[37] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications," *ACM Conference on Embedded Networked Sensor Systems,* 2003, pp. 126 – 13

[38] K. Romer, O. Kastem, and F. Mattern. "Middleware Challenges for Wireless Sensor Networks,". *ACM SIGMOBILE Mobile Computing and Communication Review (MC2R),* 2002. vol.2 , no.4 pp. 59–61

[39] M. Wang, J. Cao, J. Li and S.Das, "Middleware for Wireless Sensor Networks: A Survey" *Journal of Computer Science and Technology*, vol. 23 no. 3, May 2008 pp. 305-326

[40] B. Rubio, M. Diaz and J. Troya, "Programming Approaches and Challenges for Wireless Sensor Networks," *IEEE Second International Conference on Systems and Networks Communications (ICSNC)*, Aug. 2007

[41] P. Levis and D. Culler, "Mate: A Tiny Virtual Machine for Sensor Networks,". *In Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-X'02), San Jose, CA, USA*, Oct. 2002, pp.85-95

[42] P. Levis, D. Gay, and D. Culler, "Active Sensor Networks," *In Proceedings of the 2nh International Symposium on Networked Systems Design and Implementation (NSDI'05) San Francisco, CA, USA*, Mar. 2005 pp. 29–42

[43] T. Liu and M. Martonosi, "Impala: A Middleware System For Managing Autonomic, Parallel Sensor Systems, " *Proceedings of the ninth ACM SIGPLAN symposium on Principles and practice of parallel programming, San Deigo, CA, USA.* 2003 107–118.

[44] H.Paul and T. *McGregor,* "Wireless sensor networks: a distributed operating systems approach," *Proceedings of New Zealand Computer Science Research Student Conference, NZCSRSC,* 2009

[45] D. Estrin, R.Govindan, J.Heidemann, "Next century challenges: scalable coordination in sensor networks," *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, 1999. pp. 263 270

[46] L. Mottola and G. P. Picco "Logical Neighborhoods: A Programming Abstraction For Wireless Sensor Networks", *Proc. 2nd International Conference on Distributed Computing on Sensor Systems (DCOSS)*, 2006. pp.150 -168

[47] K.Whitehouse, C. Sharp, E. Brewer, D.Culler, "Hood: A Neighborhood Abstraction For Sensor Networks," *Proceedings of the 2nd international conference on Mobile systems, applications, and services*, 2004, pp. 99-110

[48] M.Welsh and G.Mainland, "Programming Sensor Networks Using Abstract Regions," *Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation*, vol.1, 2004

[49] L.Mottola and G. P. Picco ," Programming Wireless Sensor Networks with Logical Neighborhoods, " *In Proceedings of the 1st ACM International Conference on Integrated Internet Ad-hoc and Sensor Networks (INTERSENSE),* May 2006.

[50] P.Vicaire, E.Hoque, Z.Xie," Bundle: A Group-Based Programming Abstraction for Cyber-Physical Systems," *IEEE Transactions on Industrial Informatics*, vol.8, no.2, May 2012, pp. 379 – 392

[51] T. Abdelzaher. et. al. "EnviroTrack: towards an environmental computing paradigm for distributed sensor networks," *Proceedings of the 24th International Conference on Distributed Computing Systems*. 2004, pp. 582–589.

[52] L. Mottola and G. P. Picco, "Programming Wireless Sensor Networks: Fundamental Concepts and State Of The Art," *ACM Computing Surveys*, vol. 43, no. 3, 2011

[53] R. Gummadi, O. Gnawali, and R. Govindan, "Macroprogramming Wireless Sensor Networks using Kairos," *Proceedings of the International Conference on Distributed Computing in Sensor Systems (DCOSS*) June 2006

[54] E. Moggi,"Computational Lambda-Calculus And Monads," *Proceedings. Fourth Annual Symposium on Logic in Computer Science*, June.1989, pp. 14 – 23

[55] R. Newton, G. Morrisett, M. Welsh, "The Regiment Macroprogramming System," *International Symposium on Information Processing in Sensor Networks*, April.2007, pp. 489 – 498

[56] S.Madden, M. Franklin, J.Hellerstein, W.Hong, "TinyDB: an acquisitional query processing system for sensor networks," *Journal ACM Transaction on Database System (TODS)*, vol.30 no.1, Mar. 2005, pp. 122-173

[57] Y. Yao and J.Gehrke, "The Cougar Approach to In-Network Query Processing in Sensor Networks," *ACM SIGMOD Record*, vol.31 no.3, Sep.2002, pp.9-18

[58] W.Fung, D Sun, J. Gehrke, "COUGAR: The Network is the Database," *Proceedings of ACM SIGMOD International Conference on Management of Data*, 2002, pp. 621-621

[59] J.Horey, E.Nelson, A.Maccabe, "Tables: A Spreadsheet-Inspired Programming Model For Sensor Networks," *Proceedings of the 6th IEEE International Conference On Distributed Computing In Sensor Systems*, 2010, pp. 1-14

[60] J.Radhika and S.Malarvizhi, "Middleware Approaches for Wireless Sensor Networks: An Overview," *IJCSI International Journal of Computer Science Issues,* vol. 9, no.3, Nov.2012, pp.224-229

[61] M. Rossi, G. Zanca, L. Stabellini, R. Crepaldi, A. F. H. III, and M. Zorzi, "SYNAPSE: A Network Reprogramming Protocol for Wireless Sensor Networks using Fountain Codes," *in Proc. of IEEE SECON*, June.2008, pp. 188 – 196

[62] Q. Wang, Y. Zhu, and L. Cheng, "Reprogramming Wireless Sensor Networks: Challenges and Approaches," *IEEE Network*, vol. 20, no. 3, May/June 2006. pp. 48-55

[63] M. Yarvis, N. Kushalnagar, H. Singh, A. Rangarajan, Y. Liu, and S. Singh, "Exploiting Heterogeneity In Sensor Networks," *in Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Miami* vol.2, March.2005, pp. 878-890

[64] C.Frank and K.Römer, "Algorithms for Generic Role Assignment In Wireless Sensor Networks," *Proceedings of the 3rd international conference on Embedded networked sensor systems*, 2005, pp.230-242