This chapter presents a literature survey on routing, data aggregation, dissemination schemes, fault tolerant schemes, MA challenges and issues, mobile agent systems (MAS) in WSNs, and Middleware for WSN. It also further surveyed security requirements, possible attacks and elaborate the work done by other researchers.

Rest of the chapter has been organized as follows. Section 3.1 presents Energy Efficient Routing Protocols in WSNs. Section 3.2 discusses on Data Aggregation and Dissemination Schemes for WSNs. A look on Fault Tolerant Schemes in WSNs is given in Section 3.3. Security Attacks in WSNs are explored in section 3.4 and Security Approach in WSNs are given in Section 3.5. Secure Routing Schemes in WSNs are presented in Section 3.6. Section 3.7 discussed Mobile Agent Paradigm and Section 3.8 gives a look on Mobile Agent System and Sensor Networks. Section 3.9 briefly defines different types of sensor network middlewares and finally chapter is summarized in section 3.10.

3.1 Energy Efficient Routing Protocols in WSNs
According to [84-85] the battery energy of a SN is depleted by: (i) computational processing and, (ii) transmission and reception of data. The network layer controls both of these factors.

Routing protocols have a large scope of research work when implemented in WSNs, because the functioning of these protocols depends upon the type of network structure designed for the application or the network operations carried out using these protocols for a specific application model.

3.1.1 Flat Routing Protocols in WSNs
In flat networks, each SN typically plays the same role and SNs collaborate together to perform the sensing task. Due to the large number of such SNs, it is not
feasible to assign a global identifier to each SN. This consideration has led to data centric routing, where the BS sends queries to certain regions and waits for data from the sensors located in the selected regions. Since data is being requested through queries, attribute-based naming is necessary to specify the properties of data. Early works on data centric routing, viz., Sensor Protocols for Information via Negotiation (SPIN) [47] and Directed Diffusion [86] has minimizes the energy issue through data negotiation and elimination of redundant data. Rumor routing [87] is a variation of directed diffusion and is mainly intended for applications where the geographic routing is not feasible.

### 3.1.2 Hierarchical Routing Protocols in WSNs

Hierarchical or cluster-based routing, are well known techniques with special advantages related to scalability and efficient communication. The concept of hierarchical routing is utilized to perform energy-efficient routing in WSNs. In a hierarchical architecture, higher energy SNs can be used to process and send the information while low energy SNs can be used to perform the sensing in the proximity of the target. Some of routing protocols in this group are: Low-Energy Adaptive Clustering Hierarchy (LEACH) [80], Power-Efficient Gathering in Sensor Information System (PEGASIS) [75], Threshold-Sensitive Energy Efficient TEEN [88] and Adaptive Threshold-Sensitive Energy Efficient (APTEEN) [89].

In LEACH cluster is formed on the basis of received signal strength and local CH is selected to route the data to the BS. The CH is elected randomly and rotated among the SNs in the cluster. The cluster head communicates directly with the BS and neighbouring CH in the network.

Distributed hierarchical approach in LEACH’s makes it scalable. Data aggregation is done by the CHs and then forwarded to the BS which means that compromise or failure of a CH causes a significant problem. Selection of CH in LEACH is a challenging task. Another drawback of LEACH is that it uses single-hop routing where each node can transmit directly to the cluster leader and the BS, and can therefore not be applied to networks deployed in large regions.

In [90], authors introduce CH Election mechanism using Fuzzy (CHEF) logic. They apply the fuzzy if-then rule to the CH election mechanism. By using fuzzy logic, the computational overhead is reduced and the network lifetime is extended.
In addition, the operation of this mechanism is localized. The BS does not collect and elect CHs. The SNs only elect CHs among themselves using the fuzzy logic. The routing protocol [91] based on dynamic clustering protocol which grants a large lifetime for the network. The key idea of this protocol is to reduce transmission in intra-clusters when the objective is to collect the maximum or minimum data values in a region (like temperature, humidity, etc.).

A number of different strategies for multi-hop routing, including minimum distance and minimum-hop routing have been presented in [92]. In [93], authors have proposed efficient integer linear program formulations for assigning SNs to clusters in a two-tiered network where the higher powered relay SNs are used as CHs. The relay SNs can use a single hop model or multi-hop model to send the data to the BS. The objectives in both cases are to maximize the lifetime of the network.

The authors in [94] take a unique look at the CH election problem, specifically concentrating on applications where the maintenance of full network coverage is the main requirement. The approach for cluster-based network organization is based on a set of coverage-aware cost metrics that favor SNs deployed in densely populated network areas as better candidates for CH, active SNs and routers. Compared with traditional energy based selection methods, using coverage-aware selection of CH, active SNs and routers in clustered WSN increases the time during which full coverage of the monitored area can be maintained depending on the application scenario.

In [95], authors analyzing the advantages and disadvantages of conventional hierarchical communication protocols using MATLAB, they have developed Distance-Based Segmentation (DBS), a cluster-based protocol that significantly decreases the energy imbalance in the network by integrating the distance of the SNs from the BS into clustering policies. Furthermore, a Media Access Control (MAC) protocol that eliminates redundant delays in the cluster formation period of conventional protocols is utilized as media access scheme.

The SNs closest to the BS tend to deplete their energy faster than other SNs [96-97], which is known as an energy hole around the BS. No more data can be delivered to the BS after energy hole appears. Therefore, a lot of energy is wasted and the network lifetime ends prematurely.
In [98], authors have proposed a non-uniform SN distribution strategy to achieve the sub-balanced energy depletion. In this strategy if the number of SNs in coronas increases then the network can achieve sub-balanced energy depletion.

In [84], authors have presented an important corona model to maximize the network lifetime by using adjustable transmission range. In this model the maximum transmission range of sensors is divided into several levels. The sensors SNs belong to the same corona have the same range of transmission, whereas different coronas have different transmission ranges. The authors concluded that transmission ranges to all coronas is the effective approach to extend the network lifetime.

In [99], authors have presented a short survey on the main techniques used for energy conservation in WSNs. The main focus is primarily on duty cycle scheme which represent the most suitable technique for energy saving. In [100], authors reviewed the existing definitions of network lifetime as propose in the literature. They discussed about the merits and demerits of the existing definitions, and summarized additional requirements. They have also introduced a number of new performance metrics that have found to be useful in the context of WSN applications.

The authors in [101] have presented a systematic survey and comprehensive taxonomy of the energy saving schemes. They have also introduced mobility as a new energy saving paradigm with the purpose of maximizing the network lifetime. In [102], authors have proposed a new genetic algorithm (GA), for scheduling the data gathering of relay SNs, which significantly extends the network lifetime of a relay SN network. For smaller networks, GA based approach is always finds the optimal solution. This algorithm can easily handle large networks as compared to traditional routing schemes.

The authors have presented a novel CH election problem in [92], specifically designed for applications where the maintenance of full network coverage is the main requirement. This approach is based on a set of coverage-aware cost metrics that favor SNs deployed in densely populated network areas as better candidates for CH SNs, active SNs and routers. Compared with traditional energy-based selection methods, the coverage-aware selection of CH SNs increases the network lifetime depending on the application scenario.
In [103], authors have proposed and evaluated an Unequal Cluster based Routing (UCR) protocol for mitigating the hot spot problem in WSNs. It is designed for long lived, source-driven WSN applications, such as periodical environmental information reporting.

In [104], authors have studied a generic strategy of radioactivity minimization wherein each SN maintains the radio switched on just in the expected packet arrival intervals and guarantees low communication latency. They define a probabilistic model that allows the evaluation of the packet loss probability that results from the reduced radioactivity. This model can be used to optimally choose the radioactivity intervals that achieve a certain probability of successful packet delivery for a specific radioactivity strategy. They also define a cost model that estimates the energy consumption of the proposed strategies, under specific settings.

WSNs attracted lots of researchers because of its potential wide applications and special challenges. For past few years, WSNs mainly focused on technologies based on the homogeneous WSN in which all SNs have same system resource but recently heterogeneous WSN is becoming more and more popular and the results of researches [105-106] show that heterogeneous SNs can prolong network lifetime and improve network reliability without significantly increasing the cost.

A heterogeneous SN is more expensive, and is capable to provide data filtering, fusion and transport. It may posses one or more type of heterogeneous SNs, e.g., enhanced energy capacity or communication capability. They may be line powered, or their batteries may be replaced easily. Compared with the normal SNs, they may be configured with more powerful microprocessor and more memory. They may communicate with the BS via high bandwidth, long-distance network, such as Ethernet. The presence of heterogeneous SNs in a WSN can increase network reliability and lifetime. The main basic and important deployment problem is to decide how many and where heterogeneous SNs should be deployed in the network.

The heterogeneous resources are basically divided into three categories: computational heterogeneity, link heterogeneity, and energy heterogeneity [107].

Computational heterogeneity signifies that the heterogeneous SN has a more powerful microprocessor or microcontroller and more storage memory than the
normal SN. The SNs with more powerful computational resources can provide complex data processing and longer-term storage.

Link heterogeneity means that the heterogeneous SN has high-bandwidth and long haul network transceiver (Ethernet or 802.11) network, than the normal SN. Link heterogeneity can provide more reliable data transmission. Therefore, the reliability of the data transmission will increase by link heterogeneity.

Energy heterogeneity means that the heterogeneous SN is line powered, or its battery is replaceable. Among above three categories of resource heterogeneity, the energy heterogeneity is most important because both computational heterogeneity and link heterogeneity will consume more battery energy resource. If there is no energy heterogeneity, computational heterogeneity and link heterogeneity will bring negative impact to the WSN.

In an application for habitat monitoring [108], authors proposed a tiered system architecture in which data collected at numerous, inexpensive SNs is filtered by local processing on its way through larger, more capable and more expensive SNs. The necessity of heterogeneity and the mechanisms of packet forwarding and processing are demonstrated in [109-110]. However, how to use heterogeneous SNs effectively has not been studied comprehensively.

PEGASIS [75] discusses how to extend the lifetime of WSNs. It is a near optimal chain-based protocol, not a clustering scheme. In PEGASIS, each SN communicates only with an adjacent neighbor and takes turns transmitting to the BS. A number of issues are discussed, viz., reducing the amount of energy spent per processing round, minimizing overall distance between non-leader SNs of the system, and minimizing the number of data transmission to the BS. However, in PEGASIS, SNs die in random locations since the CHs have been chosen without any concern on an overall lifetime of each SN.

To prolong the network lifetime by distributing energy consumption, HEED (Hybrid Energy-Efficient Distributed Clustering) [00000] is a standalone distributed clustering approach in which each SN considers two factors: remaining energy and communication cost before making a decision to join one cluster or the other. In HEED, once selected a CH is maintained for a fixed number of iterations. This is in contrast to some other approaches where the CHs are newly selected in every step. This is to reduce the unnecessary high setup cost associated with the CH selection process.
In [112], authors have studied the impact of heterogeneity of SNs, in terms of their energy and have proposed a heterogeneous – aware protocol to prolong the time interval before the death of the first SN, which is crucial for many applications where the feedback from the WSN must be reliable. In [113], authors have proposed a new distributed energy efficient clustering scheme for heterogeneous WSNs, which are called DEEC. In DEEC, the CHs are elected by a probability based on the ratio between residual energy of each SN and the average energy of the network. The SNs with high initial and residual energy will have more chances to become CHs than the SNs with low energy.

In [114], authors address the deployment problem of heterogeneous WSNs and is supported by an algorithm to decide how many and where heterogeneous SNs should be deployed in the WSN. The core algorithm is based on the locations of all SNs, can optimize placement of heterogeneous SNs in an arbitrary WSN to increase the network lifetime and reliability. In [115], authors propose a distributed election clustering protocol to prolong the stable region of heterogeneous WSNs, which is based on remaining energy and communication cost to elect suitable CH SNs. Compared with classical clustering protocol, it can maintain load balancing of networks, and extremely prolong the stable region and the network lifetime.

In [116], authors have presented an Energy-Efficient Protocol with Static Clustering (EEPSC) which partitions the network into static clusters and utilizes CHs to distribute the energy load among high power SNs for extending the network lifetime. But the authors have not investigated the effect of heterogeneity in the network system.

Energy-Efficient Hierarchical Clustering Algorithm (EEHCA) scheme [117] improves the performance of LEACH and HEED (Hybrid Energy-Efficient Distributed clustering), in terms of network lifetime. EEHCA adopts a new method for CH election, which can avoid the frequent election of CH. In order to improve the performance of the WSN new concept of backup CHs is introduced. Therefore, when SNs finished the communication within their own clusters and the CHs have finished the data aggregation, the head clusters will transmit aggregated data to the BS.

A novel energy efficient centralized clustering algorithm for WSNs is presented in [118]. This algorithm generates a set of possible clustering alternatives, which
helps in finding the optimal clustering and present a performance evaluation of the proposed scheme by using two performance metrics Max-Min and Max-Sum. It is found that Max-Sum improves the system lifetime performance over Low Energy Adaptive Clustering Hierarchy-C (LEACH-C).

In [119], authors presented the Multi-hop Routing Algorithm for Inter CH Communication (MRACHC). The algorithm was multilayer multi-hop routing algorithm, which worked on the principle of divide and conquers, and was performing well in terms of load balance and energy efficiency then LEACH.

3.1.3 Location Based Routing Protocols
In location-based routing, all the SNs are addressed by using their locations. Depending upon the strength of the incoming signals, it is possible to calculate the nearest neighboring SN’s distance. Due to obstacles in the network often the signal strength becomes weaker and SNs are unable to find the nearest neighbor SNs. There are many location-based schemes of which Geographic and Energy aware Routing (GEAR) [120] and Geographic Adaptive Fidelity (GAF) [121] are two important schemes.

3.2 Data Aggregation and Dissemination Schemes in WSNs
The main goal of data aggregation algorithms is to gather and aggregate data in an energy efficient manner so that network lifetime can be enhanced. Data gathering is defined as the systematic collection of sensed data from multiple sensors to be eventually transmitted to the BS for processing. Since SNs are energy constrained, it is inefficient for all the sensors to transmit the data directly to the BS. Data generated from neighboring sensors is often redundant and highly correlated. In addition, the amount of data generated in large WSNs is usually enormous for the BS to process. Hence, we need methods for combining data into high quality information at the sensors or intermediate SNs which can reduce the number of packets transmitted to the BS resulting in conservation of energy and bandwidth. This can be accomplished by data aggregation. It may be defined as the process of aggregating the data from multiple SNs to eliminate redundant transmission and provide fused information to the BS and it usually involves the data fusion at intermediate SNs and transmits the aggregated data to the BS.
Data aggregation attempts to collect the most critical data from the sensors and make it available to the BS in an energy efficient manner with minimum data latency. Data latency is important in many applications such as environment monitoring where the freshness of data is also an important factor. It is critical to develop energy efficient data aggregation algorithms so that network lifetime can be enhanced. There are several factors which determine the energy efficiency of a WSN such as network architecture, the data aggregation mechanism and the routing protocol. The architecture of the WSN plays a vital role in the performance of different data aggregation protocols.

3.2.1 In-Network Aggregation

In a typical WSN scenario, different SN collect data from the environment and then send it to some central SN or BS which analyzes and process the data and then send it to the application. But in many cases, data produced by different SNs can be jointly processed while being forwarded to the BS. In-network aggregation deals with this distributed processing of data within the network.

Data aggregation techniques explore how the data is to be routed in the network and the processing method that are applied on the packets received by a SN. They have a great impact on the energy consumption of SNs and thus on network efficiency by reducing number of transmission or length of packet. In [122], authors define in-network aggregation process “In-network aggregation is the global process of gathering and routing information through a multi-hop network, processing data at intermediate SNs with the objective of reducing resource consumption (in particular energy), thereby increasing network lifetime”.

There are two approaches for in-network aggregation: with size reduction and without size reduction. In-network aggregation with size reduction refers to the process of combining and compressing the data packets received by a SN from its neighbors in order to reduce the packet length to be transmitted or forwarded towards BS. As an example, consider the situation when a SN receives two packets which have a spatial correlated data. In this case it is worthless to send both packets. Instead of that one should apply any function like Average (AVG), Maximum (MAX), and Minimum (MIN) to send a single packet. This approach considerably reduces the amount of bits transmitted in the network and thus saving a lot of energy but on the other hand, it also reduces the precision of data value
received. In-network aggregation without size reduction refers to the process of merging data packets received from different neighbors into a single data packet but without processing the value of data. As an example, two packets may contain different physical quantities (i.e., temperature and humidity) and they can be merged into a single packet by keeping both values intact but keeping a single header. This approach preserves the value of data and transmit more bits in the network but still reduce the overhead by keeping single header.

These two approaches depend on many factors like the type of application, data arrival rate and network characteristics. There is also a trade-off between energy consumption and precision of data for the two approaches.

Most of the work available in literature on in-network aggregation mainly deals with problem of forwarding packets from source to BS, to facilitate aggregation therein. Actually the main idea behind were to enhance existing routing protocols such that they can efficiently aggregate data. Most of the data aggregation techniques fall under three categories - tree-based approaches, multi-path approaches, and cluster-based approaches[123]. There are also some hybrid approaches that combine any of these three techniques. These approaches are described in the coming sections with giving details of some of the main techniques by different authors.

### 3.2.2 Tree Based Approach

The simplest way to aggregate data is to organize the SNs in a hierarchical manner and then select some SNs as the aggregation point or aggregators. The tree-based approach perform aggregation by constructing an aggregation tree, which could be a minimum spanning tree, rooted at BS and source SNs are considered as leaves. Each SN has a parent SN to forward its data. Flow of data starts from leaf SNs up to the BS and therein the aggregation done by parent SNs. The way this approach operates has some drawbacks. In case of packet loss at any level of tree, the data will be lost not only for a single level but for the whole related sub-tree as well. In spite of high cost for maintaining tree structure in dynamic networks and scarce robustness of the system, this approach is suitable for designing optimal energy efficient aggregation technique.

A data-centric protocol is based on aggregation tress, known as Tiny Aggregation (TAG) approach [124]. TAG works in two phases: distribution phase
and collection phase. In distribution phase, TAG organizes SNs into a routing tree rooted at BS. The tree formation starts with broadcasting a message from BS specify level or distance from root. When a SN receive this message it sets its own level to be the level of message plus one and elect parent as SN from which it receives the message. After that, SN rebroadcast this message with its own level. This process continues till all the SNs elect their parents. After tree formation, BS sends queries along structure to all SNs in the network. TAG uses database Structured Query Language (SQL) for selection and aggregation functions. In collection phase, data is forwarded and aggregated from leaf SNs to root. A parent SN has to wait for data from its entire child SN before it can send its aggregate up the tree. Apart from the simple aggregation function provided by SQL (e.g., COUNT, MIN, MAX, SUM, and AVG), TAG also partitions aggregates according to the duplicate sensitivity, exemplary and summary, and monotonic properties. Though TAG periodically refresh tree structure of network but most of the tree-based schemes are inefficient for dynamic network.

In [125], authors proposed a reactive data centric protocol for applications where BS asks some specific information by flooding, known as directed diffusion paradigm. The main idea behind directed diffusion paradigm is to combine data coming from different sources and en-route them by eliminating redundancy, minimizing the number of data transmission, thus maximizing network lifetime. Directed diffusion consists of several elements: interests, data messages, gradients, and reinforcements.

The BS requests data by broadcasting an interest message which contains a description of a sensing task. This interest message propagates through the network hop-by-hop and each SN also broadcast interest message to its neighbor. As interest message propagates throughout the network, gradients are setup by every SN within the network. The gradient direction is set toward the neighboring SN from which the interest is received. This process continues till gradients are setup from source SN to BS. Loops are not checked at this stage but removed at later stage. After these paths of information flow are formed, then best path are reinforced to prevent further flooding according to a local rule. Data aggregation took place on the way of different paths from different sources to BS. The BS periodically refreshes and resends the interest message and it starts to receive data from sources to provide reliability. The problem with directed diffusion is that it
may not be applied to applications (e.g. environmental observation and forecasting[212]) that require continuous data delivery to BS. This is because query driven on demand data model may not help in this regard.

Also matching data to queries might require some extra overhead at the SNs. Mobility of BS SNs can also degrade the performance as path from sources to BS cannot be updated until next interest message is flooded throughout the network. To cope up with above issue if introduce frequent flooding then also too much overhead of bandwidth and battery power will be introduced. Furthermore, exploratory data follow all possible paths in the network following gradients which lead to unnecessary communications overhead.

In [126], authors proposed a new low-control-overhead data dissemination scheme, which they called as Pseudo-Distance Data Dissemination (PDDD), for efficiently disseminating data from all SNs to mobile BS. In the development of PDDD the following assumptions are made: (1) all source SNs maintain routes to mobile BS, (2) no periodical messaging for topological changes due to mobile BS, (3) all links are bi-directional and no control messages will lost, (4) mobile BS SNs will be equipped with unlimited battery power and (5) network partitioning is not considered. Data dissemination process is influenced by Directed Diffusion [86]. Though mobile BS periodically broadcast interest message, SNs do not send exploratory data and do not wait reinforcement message because each SN already has routes to the BS. After getting interest message, adjacent SNs set a parent-child relationship using pseudo-distance of each SN and finally a Partial Ordered Graph (POG) has been build. Optimal data dissemination is achieved in terms of path length by forwarding packets to a parent SN until topology is unchanged. Then each SN is assigned a level for a corresponding BS with pseudo-distance. In order to overcome the shortcoming of POG, author used Totally Ordered Graph (TOG) in place of POG. Due to mobility of BS all SNs have to maintain routes and for any change in topology SNs have to again change route accordingly which led to energy waste.

In [127], authors proposed an energy-aware spanning (ESPAN) tree algorithm for data aggregation. E-Span is a distributed protocol in which source SN which has highest residual energy is chosen as root. Other source SNs choose their parent based on residual energy and distance to the root. The protocol uses configuration message to exchange information of SN, i.e., residual energy and distance to the
root. Each SN performs single-hop broadcast operation to send packets. Single-hop broadcast refers to the operation of sending a packet to all single-hop neighbors [127].

3.2.3 Multi-Path Approach

One of the main drawbacks of tree-based approach is the scarce robustness of the system. To overcome this drawback, many approaches are available in the literature. The theme behind these approaches is that instead of sending partially aggregated data to a single parent SN in aggregation tree, a SN sends data over multiple paths. In this scheme each SN sends data to its possibly multiple neighbors by exploiting the wireless medium characteristic. Thus, data will flow from sources to BS along multiple paths and aggregation can be performed by each intermediate SN. Clearly, schemes using this approach will make the system robust but with some extra overhead. One of the aggregation structures that fit well with this approach is ring topology, where network is divided into concentric circles with defining levels according to the hop distance from BS.

In [128], authors have presented a data aggregation technique using multi-path approach, known as synopsis diffusion. Synopsis diffusion works in two phases: distribution of queries and data retrieval phase. During distribution of queries phase, a SN sends a query in the network. The network SNs then form a set of rings around the querying SN. The SN which is i hop away from querying SN is considered as ring $R_i$. In the second phase, aggregation starts from outermost ring and propagate level by level towards the BS. Here, a source SN may have multiple paths towards BS.

In [129], authors describe a new strategy for data gathering in WSN which considers both issues: energy efficiency and robustness. Authors first say that single path to connect each SN to the BS is simple and energy-saving approach but expose a high risk of disconnection due to SN/link failures. But multi-path approach would require more SNs to participate with consequent waste of energy. Authors present a clever use of multi-path only when there is loss of packet which is implemented by smart caching of data at SNs. Authors also argue that in many practical situation data may be gathered only from a particular region, so they use a different approach that relies on a spanning tree and provides alternative paths only when a malfunctioning is detected. Algorithm adopts a tree-based approach
for forwarding packets through the network. In the ideal situation when no failures occur, this is certainly the best choice, as the minimum numbers of SNs are engaged in the transmission phase. In the presence of link or SN failures, the algorithm discovers an alternative path which ensures the delivery of as many packets as possible within the time constraints. The problem with this approach is that it may cause the arising of hot spots and SNs along preferred paths will consume their energy resources quickly, possibly causing disconnection in the network.

### 3.2.4 Cluster Based Approach

We have discussed about hierarchical organization of the network in tree-based approach. Another important technique to organize the network in hierarchical manner is cluster-based approach. In cluster-based approach, whole network is divided into several clusters. Each cluster has a CH which is selected among cluster members. CHs do the role of aggregator which aggregate data received from cluster members locally and then transmit the result to BS. The advantages and disadvantages of the cluster-based approaches are similar to tree-based approaches.

A Maximum Lifetime Data Aggregation (MLDA) algorithm which finds data gathering schedule provided location of sensors and BS, data packet size, and energy of each sensor has presented in [130]. A data gathering schedule specifies how data packet are collected from sensors and transmitted to the BS for each round. A schedule can be thought of as a collection of aggregation trees.

In [131], authors present a Two-Phase Clustering (TPC) scheme. Phase I of this scheme creates clusters with a CH and each SN within that cluster form a direct link with CH. Phase I of this scheme is similar to various scheme used for clustering. The CH rotation is localized and is done based on the remaining energy level of the SNs which minimizes time variance of sensors and this leads to energy saving from unnecessary CH rotation. In phase II, each SN within the cluster searches for a neighbor closer than CH which is called data relay point and setup up a data relay link. Now the SNs within a cluster either use direct link or data relay link to send their data to CH which is an energy efficient scheme. The data relay point aggregates data at forwarding time to another data relay point or CH. In case of high network density, TPC phase II setups unnecessary data relay link
between neighbors as closely deployed sensors will sense same data and this lead to a waste of energy.

An energy efficient and secure pattern based data aggregation protocol is designed for clustered environment which is presented in [132]. In conventional method data is aggregated at CH and it eliminates redundancy by checking the content of data. This protocol says that instead of sending raw data to CH, the cluster members send corresponding pattern codes to CH for data aggregation. If multiple SNs send the same pattern code then only one of them is finally selected for sending actual data to CH. For pattern matching, authors present a pattern comparison algorithm.

In [133], authors discuss two parts: in the first part, they propose an analytical model to formulate the data aggregation delay at a SN that operates under IEEE 802.15.4 and uses beacon enabled slotted CSMA-CA. Since most of the sensors that are manufactured comply with IEEE 802.15.4, and CSMA-CA is adopted for its simplicity compared with contention free protocols. In the second part, they propose a heuristic algorithm for constructing an energy efficient data aggregation tree under the constraint that the data gathered from the furthest SN will reach the root of the tree within the specified delay bound.

In [134], authors propose LEECF (Low-Energy Event Centric Fusion), an event-centric-based protocol which utilizes the centric SN to aggregate the event data among the triggered sensors in a short delay to decrease the complexity. They also consider a fast data fusion algorithm by designing a confidence matrix for target identity and prove that it attains the same precision as Dempster-Shafer (DS) evidence formula in a much less time.

The proposed scheme begins with a setup phase of level at which the direction and distance of source to BS are organized. When an event occurs, it enters into the search phase of aggregator, when the center SN of source SNs is found, followed by the construction phase of fusion Steiner tree, then the event data converges to the aggregator (tree root) along the fusion tree at which the data fusion phase begins to run and the fusion result is relayed to the BS.

In [135], the authors develop a data fusion algorithm that combines the cluster-based design of WSNs using fuzzy logic methods. The performance of a network is increased by eliminating redundancy and power consumption, ensuring fault-
tolerance between sensors, and managing effectively the available communication bandwidth between network components.

### 3.3 Fault Tolerant Schemes in WSNs

SNs in WSNs are prone to failure due to energy depletion, hardware failure, communication link errors, malicious attack, and so on. Unlike the cellular and ad hoc networks where energy has no limits in BS or batteries can be replaced as needed, SNs in WSNs have very limited energy and their batteries cannot usually be recharged or replaced due to hostile or hazardous environments. So, one important characteristic of WSNs is the stringent power management of the SNs. Two components of a SN, viz., sensing unit and wireless transceiver, usually directly interact with the environment which is subject to variety of physical, chemical, and biological factors. It results in low reliability of performance of the SNs. Even if condition of the hardware is good, the communication between the SNs are affected by many factors, such as signal strength, antenna angle, obstacles, weather conditions, interference.

Several sensor network services, such as tracking, routing, and spatial querying, require continuous maintenance of distributed data structures, such as trees, paths, and clusters, over a large number of SNs. This is a challenging task because message losses and corruptions (due to fading, collisions, and hidden node effects) and node failures (due to software/hardware crashes or energy exhaustion) can drive portions of these large scale structures to be arbitrarily corrupted and hence to become inconsistent with the rest of the structure. For dealing with arbitrary corruptions, we need self healing systems: A self healing system ensures eventual satisfaction of system specifications upon starting from a corrupted state. However, since faults can temporarily violate the program specifications in a self healing system, extra care should be taken for containing the effects of faults: Faults in one part of the system may contaminate the entire system which may result in a high cost, system wide correction. Thus, mechanisms for local containment of faults are needed for continuous and local maintenance of large scale data structures.

Since the complexity of software grows drastically with respect to its size, large scale software systems are extremely error prone and fail frequently. Especially for sensor network applications which are inherently distributed,
reasoning about the system and verification of correctness are more difficult due to the lack of a centralized controller and the lack of a globally shared memory. Again due to their overwhelming complexity, design of fault tolerance for large scale software systems remains a challenging task.

White box approaches for designing fault tolerance, such as exception handling, forward recovery, recovery blocks [136], and application specific fault tolerance methods [137-138], assume that the implementation is fully available, and study the source code for designing fault tolerance. However, they are not applicable for large scale software systems because the task of studying the implementation and designing a fault tolerant version becomes unbearable as the size of the implementation grows.

Black box solutions, such as reset and restart ability approaches’, may be adequate for centralized software systems, however they are inapplicable for massively distributed software since a reset of the software would mean a global reset of the entire network, and would incur a lot of work and down time. Thus, a more efficient and informed approach is needed for achieving scalable design of fault tolerance with respect to software size. Fault control is a highly desirable property for fault tolerance systems, and it has received a lot of attention in the fault tolerance literature. One way of achieving fault control is through design of masking fault tolerance: A system is masking tolerant if in the presence of faults, it always satisfies its safety properties and, when faults stop occurring, it eventually resumes satisfying its liveness properties. Since all faults are masked immediately, a masking fault tolerant system is trivially fault containing.

Masking fault tolerance is a strong property and hence it is applicable for a limited class of faults, such as single node failure, message loss, etc. Also, since masking tolerance is excessively striving, the cost of masking tolerance may be too high a price to pay in WSNs, where energy is a precious commodity, and in real time applications, where the freshness of data is as important as the accuracy of the data. These concerns led researchers to investigate other approaches for fault control.

In [139], the authors show that by adding structure and sacrificing full distribution it is possible to improve the fault control of self stabilizing algorithms. In this article authors investigated an application of the general principle of introducing structure to the area of self stabilizing spanning tree construction. It is
possible to transform an arbitrary self stabilizing spanning tree algorithm into one with increased efficiency and fault control properties. After adding the fixed structure, faults can only lead to perturbations within the algorithm instances in which they happen. This is in contrast to the case where a standard spanning tree algorithm runs in the entire network: a single fault (e.g., of the root) can lead to a global reconfiguration. However, the level of fault control again depends on the distribution of algorithm instances to processing elements: if one processing element participates in all algorithm instances then a failure of this node may also cause global disruption.

In [140] authors identified hierarchical error detection and control framework for a Software Implemented Fault Tolerance (SIFT) layer of a distributed system. The design and implementation of a software based distributed signature monitoring scheme is central to the proposed four level hierarchy: process level, node level, group level, and across groups. Authors report a substantial increase in availability due to the detection framework and help in understanding the tradeoffs between overhead and coverage for different combinations of techniques.

In [141] area of fault control of self stabilizing algorithms has received growing interest. A notion of local correction was suggested in [142] in the context of self stabilization. The meaning of locality there is that each node can act locally to correct a state of an algorithm. However, if the corrected algorithm is global, then the function computed by the corrected algorithm it can be output only after $O(n)$ (number of nodes) or $O(Diameter)$: diameter of the network).

In [143] authors introduce the notion of fault mending, which addresses the issue of relating repair time to fault severity; their method is based on state replication and is self stabilizing only in a synchronous system.

In [144], a broadcast protocol is proposed to contain observable variables in the presence of state corruptions, but the protocol allows for global propagation of internal protocol variables. The notion of fault control within the context of stabilization is formalized first in [145]; algorithms were proposed to contain state corruption of a single node in a stabilizing spanning tree protocol. In [146] authors construct self stabilizing protocols that guarantee safe convergence from states that arise from legitimate states perturbed by limited topology changes. It achieves fault local stabilization in shortest path. To achieve fault control the protocol uses control actions that are a constant time faster than the fault intolerant
program actions. We enable fault control in a hierarchy based manner by suitably varying the speed of actions as per the level of the hierarchy they are executed at.

The services provided by a WSN to a large extent depend on fault tolerance, because it cannot be assumed that all sources of error can be eliminated, even through careful engineering. By considering service availability, we understand the probability with which a request will lead to a valid and useful response. The availability of WSN is defined as:

\[ P(A) = \frac{MTTF}{MTTF + MTTR} \]

where MTTF stands for Mean Time to Failure and MTTR stands for Mean Time to Repair [147].

By analyzing this equation we can conclude that systems that constantly fail and require long repair time will result in systems with very low availability. However, systems that have a high MTTF and can be quickly repaired are considered highly available systems.

Attacks on the availability of SNs are analyzed in [118]. These attacks directly influence the availability of services. Fault tolerant systems can overcome faults and system failures, therefore increasing the MTTF and system availability. Fault tolerance techniques especially crafted for WSNs have not been extensively studied so far. The most important one is replication, which is well suited for WSNs due to their inherent SN redundancy.

Thus, we need to design an intelligent fault tolerant system [148] which must take care of most of the fault arising in a WSN.

### 3.3.1 Fault Tolerance At Different Levels

There are five levels of fault tolerance, viz., physical layer, hardware layer, system software layer, middleware layer, and application layer [121]. Based on the study, we classify fault tolerance in WSNs into four levels from the system point of view. More specifically, fault tolerance in a WSN system may exist at hardware layer, software layer, network communication layer, and application layer.

### 3.3.2 Hardware Layer

Faults at hardware layer can be caused by malfunction of any hardware component of a SN, such as memory, battery, microprocessor, sensing unit, and network interface (wireless radio). There are three main reasons that cause
hardware failure of SNs. The first is that WSNs are usually for commercial use and SNs are cost sensitive. Therefore, design of a SN will not always use the highest quality components. The second is that strict energy constraints which restrict long and reliable performance of SNs. In [149-151] due to stress from the environment and inadequate enclosures, the SNs were exposed to direct contact with water causing short circuits. The report of a large-scale deployment in a potatoes field [152] indicated that the antennas from the SNs were quite fragile and would become loose when inserting the SN into the packaging. When the battery of a SN reaches a certain stage, sensor readings may become incorrect. This has been observed in a field many outlier readings were generated in the network caused by imminent battery failure. For example, sensor readings may become incorrect when the battery of a SN reaches a certain level [153].

The third is that WSNs are often deployed in harsh and hazardous environments which affect normal operation of SNs. The wireless radios of SNs are severely affected by these environment factors.

3.3.3 Software Layer
Software of a SN consists of two components: system software, such as operating system, and middleware, such as communication, routing, and aggregation. An important component of system software is to support distributed and simultaneous execution of localized algorithms. Software bugs are a common source of errors in WSNs. One promising method is through software diversity where each program is implemented in several different versions. Since it is difficult to provide fault tolerance in an economic way at hardware level of a SN, numerous fault-tolerant approaches are expected at the middleware level. The majority of current applications in WSNs are simple. To adapt the real-life applications, there is a need to develop much more complex middleware for WSNs.

In [154], the researchers reported that a software bug caused the longest continuous network outage taking the system offline for three days until the SNs could be reprogrammed manually.
3.3.4 Network Communication Layer

Faults at network communication layer are the faults on wireless communication links. Assuming that there is no error on hardware, link faults in WSNs are usually related to surrounding environments. In addition, link faults can also be caused by radio interference of SNs. For example, SN ‘a’ cannot successfully receive a message from SN ‘b’ if SN ‘a’ is within interference range of other SNs that are transmitting messages at the same time. The standard way to enhance the performance of wireless communication is to use aggressive error correction schemes and retransmission. These two methods may cause further delay of operation. It should be pointed out that there is always a trade-off between fault tolerance and efficiency.

In WSNs, communication links between SNs are highly volatile and are not always yield the same delivery rate of messages in field trials as in lab trials. For instance, in [155] a delivery rate of only 76% of the messages was observed; in [131,156] the instability of the links between SNs leads to constant changes in the routing paths.

In [157], authors present an experimental study on fault tolerant routing for wireless sensor grid networks. A network may be separated into partitions owing to the broken radio links resulted from SN failures. Since the leveling algorithm works effectively for un-partitioned network, a combined technique with extended transmission range is investigated to resolve the network-partitioning problem. In [158], authors propose FTPASC (Fault Tolerant Power Aware protocol with Static Clustering), a static clustering based protocol, which engages high power SNs for power consuming tasks and as a result extends the network lifetime. In classical static clustering, when the CHs energy was depleted all the SNs of that cluster lose their communication with the BS and become essentially dead. In each cluster, FTPASC chooses the SN with maximum energy as the CH. Thus, not only there is always one CH for each cluster, but also the overhead of dynamic clustering is removed.

3.4 Security Attacks in WSNs

In [159] authors have presented detailed attacks in WSNs which are described as follows:
Routing Loops: In WSNs, routing loops attack the information exchanged between SNs. False error messages are generated when an attacker alters and replays the routing information. Routing loops attract or repel the network traffic and increase SN-to-SN latency.

Selective Forwarding: Selective forwarding is a way to influence the network traffic by believing that all the participating SNs in network are reliable to forward the message. In selective forwarding attack, malicious SNs simply drop certain messages instead of forwarding every message. Once a malicious SN cherry picks on the messages, it reduces the latency and deceives the neighboring SNs that they are on a shorter route. Effectiveness of this attack depends on two factors. First is the location of the malicious SN. The closer it is to the BS, the more traffic it will attract. Second is the percentage of messages it drops. When a selective forwarder drops more messages and forwards less, it retains its energy level, thus remaining powerful to trick the neighboring SNs.

Sinkhole Attacks: In sinkhole attacks, an adversary attracts the traffic to a compromised SN. The simplest way of creating a sinkhole is to place a malicious SN where it can attract most of the traffic, possibly closer to the BS or malicious SN itself, which deceptively acts as a BS. One reason for sinkhole attacks is to make selective forwarding possible to attract the traffic toward a compromised SN. The nature of WSNs where all the traffic flows toward one BS makes this type of attack more susceptible.

Sybil Attacks: This is a type of attack where a SN creates multiple illegitimate identities in WSNs either by fabricating or stealing the identities of legitimate SNs. Sybil attacks can be used against routing algorithms and topology maintenance; it reduces the effectiveness of fault-tolerant schemes such as distributed storage and disparity. Another malicious factor is geographic routing where a Sybil SN can appear at more than one place simultaneously.

Wormholes: In wormhole attacks, an adversary positioned closer to the BS can completely disrupt the traffic by tunneling messages over a low-latency link. Here an adversary convinces the SNs which are multi-hop away that they are closer to the BS. This creates a sinkhole because an adversary on the
other side of the sinkhole provides a better route to the BS.

- **Hello Flood Attacks**: This involves broadcasting a message with stronger transmission power and pretending that the HELLO message is coming from the BS. Message receiving SNs assume that the HELLO message sending SN is the closest one and they try to send all their messages through this SN. In this type of attack, all SNs will be responding to HELLO floods and wasting the energies. The real BS will also be broadcasting the similar messages but will have only a few SNs responding to it.

- **DoS Attacks**: Denial of service attacks occur at a physical level, causing radio jamming, interfering with the network protocol, battery exhaustion, and so on.

There are number of security threats in computing. In [160] authors have presented possible security threats in WSNs and are given below:

- **Passive Information Gathering**: An adversary with powerful resources collecting information from WSNs if information is not encrypted.

- **SN Subversion**: Capture of a SN may reveal its information including disclosure of cryptographic keys, hence compromising the whole WSN.

- **False SN**: Addition of a malicious SN by an adversary to inject the malicious data, false SN would be computationally robust to lure other SNs to send data to it.

- **SN Malfunction**: A malfunctioning SN will generate inaccurate data that would jeopardize the integrity of a WSN, especially when that SN is a data aggregating SN for example, a cluster leader.

- **SN Outage**: Means SNs stops functioning. WSNs protocols should be robust enough to mitigate the effects of SN outages by providing an alternate route.

- **Message Corruption**: When contents of a message are modified by an attacker, it compromises the message integrity.

- **Traffic Analysis**: Even if the message transfer is encrypted in WSNs, its still leaves the high probability of analysis of communication patterns and sensor activities revealing enough information to enable an adversary to cause more malicious harm to WSNs.
3.5 Security Approach in WSNs

On the basis of study we have observed that in WSNs security should be implemented at least at following four layers.

3.5.1 Application Layer

Data are collected and managed at application layer therefore it is important to ensure the reliability of data. In [161] author has presented a resilient aggregation scheme that is applicable to a cluster-based network where a cluster leader acts as an aggregator in WSNs. However, this technique is applicable if the aggregating SN is in the range with the entire source SNs and there is no intervening aggregator between the aggregator and source SNs. In the hierarchical clustering approach, a communication channel between the aggregator and BS has potentially limited bandwidth because the cluster leader as an aggregator itself is a SN [161-162]. To prove the validity of the aggregation, cluster leaders use the cryptographic techniques to ensure the data reliability. We will discuss the cryptography in key management section.

3.5.2 Network Layer

The network layer is responsible for routing of messages from SN to SN, SN to cluster leader, cluster leaders to cluster leaders, cluster leaders to the BS, and vice versa.

Routing protocols in WSNs are of two types: (1) ID-based protocols, in which packets are routed to the destination based on the IDs specified in the packets, and (2) data-centric protocols [163] in which packets contain attributes that specify the type of data being provided. In [162] authors have described routing attacks presented in [159], in WSNs as below:

- Packets are dropped completely, or selectively.
- The network is flooded with global broadcasts.
- Some SNs in the network are misguided into believing that SNs are either multiple hops away or do not exist at all in the neighbors.
- A significant proportion of the traffic is tunneled from one place in the network to another distant place of the network depriving other parts of
the network that under normal circumstances would have received the traffic themselves.

- Sometimes traffic is lured to a particular SN or a small group of SNs, depriving other parts of the network that normally would have received the traffic themselves.

Security of routing protocols depends on the location of SNs and the encryption techniques.

### 3.5.3 Data Link Layer

The data link layer does the error detection and correction, as well as encoding of data. The link layer is vulnerable to jamming and Denial of Service attacks. TinySec [164] has introduced link layer encryption, which depends on a key management scheme. However, an attacker having better energy efficiency can still wage an attack. Protocols like LMAC [165] have better anti-jamming properties, which are viable countermeasures at this layer.

### 3.5.4 Physical Layer

The physical layer emphasizes the transmission media between sending and receiving SNs; the data rate, signal strength, and frequency types are also addressed in this layer. Ideally, the FHSS frequency hopping spread spectrum is used in WSNs.

### 3.6 Secure Routing Schemes in WSNs

Secure routing in WSNs is challenging due to the unique characteristics of WSNs as compared to wired and wireless ad hoc networks. Traditional IP-based routing is not a viable solution due to a relatively large number of SNs because the overhead of IP maintenance is very high. The following are the issues that need to be kept in mind while designing a secure routing protocol in WSNs.

- SNs are self-organizing due to the ad hoc deployment, and SNs are left unattended after the deployment.
- In WSNs, most of the time flow of data would be from SNs to cluster leader
and BS.

- Careful route management due to SNs limitations.
- Frequent changes in network topology due to the dynamic nature of WSNs.
- WSNs are application-specific and data-centric.
- Secure location of SNs because Global Posting Systems (GPS) are not suitable for WSNs.

There are very few routing protocols proposed to address the secure routing issues in WSNs. SPINS is a suite of security building blocks proposed by Perig et al. [47], which is a collection of security protocols SNEP (Secure Network Encryption Protocol) and μTESLA, a micro version of TESLA (Time Efficient Streamed Loss-tolerant Authentication). SNEP provides data confidentiality and two-way data authentication with minimal overheads and μTESLA provides authenticated streaming broadcast. SPINS leaves some questions like security of compromised SNs, DoS issues and network traffic analysis issues unaddressed. Furthermore, this protocol assumes a static network topology ignoring the ad hoc and mobile nature of SNs.

INSENS [166-167] tolerates intrusion by bypassing the malicious SNs instead of detecting the intrusion. Even if a malicious SN exists in the network, INSENS mitigates the impact of that intrusion. INSENS does not rely on conventional anomaly-based intrusion detection techniques; instead it uses an intrusion-tolerant mechanism that reduces the harm caused by presence of a small number of undetected intruders in the network by incorporating redundancy in routing.

### 3.7 Mobile Agent Paradigm

A Platform for mobile agent Distribution & Execution (PMADE) [65-67] consists of mainly two components- Agent Host and Agent Submitter. Each host of the network has a server called Agent Host (AH), which accepts and executes incoming agents and a client called Agent Submitter (AS), which submits the agent on behalf of the user to AH.

When a user wants to perform a task, he/she submits the agent, designed to perform that task, to AS on the user system. The AS establishes a connection with the specified AH, where the user is registered, submits the agent and goes offline.
The AH examines the nature of the agent and if required, clones and forwards it to other active AHs in the network. It then goes on to execute one clone.

The execution of the agent depends on its nature and state. It can be transferred from one AH to another whenever required. On completion of execution, it submits its results to the AH, which in turn stores the results, until the AS receives them for the user. AS and AH are discussed in detail in the following sections:

The AS plays a crucial role in formulating and dispatching agents to AH. It acts as an interface between the user and AH. One of its primary tasks is to attach a signature to the invoked agent. It retrieves the static IP address of the host on which it is running and binds it to an agent signature.

The AS also receives replies from the AH for user requests. It keeps track of agents and maintains their profile that it submits to the AH from the user’s system. The architecture allows a user to go offline after submitting its agents and receive results on reconnection to the network. The AS functions are as follows:

- Receives an agent and verifies whether it is listed in its database
- If found, checks the status of AH, connects to it, and sends the agent to the AH
- If connection is unsuccessful sends an appropriate reply to user
- If agent reaches AH successfully, receives acknowledgment from the AH
- Receives results from AH and stores on the local disk for future references by the client

The AH is the key component of PMADE. It consists of the manager modules and the Host Driver. The Host Driver is the basic utility module, which lies below the manager modules and is responsible for driving the AH, by ensuring proper coordination between the various managers and making them work in tandem.

Various manager modules help to perform functions like- agent transfer, execution, communication, etc. Details of the managers and their functions are provided in [65-67]. PMADE provides weak mobility to its agents and allows one-hop, two-hop and multi-hop agents [66]. PMADE has focused on Flexibility [66], Persistence [68-69], Security [10], Collaboration, and Reliability [67].

It implements mechanisms to protect local resources from malicious agents, agents from malicious hosts and for communication among agents across the network and in local host entities. A large variety of mechanisms, policies and tools are available in PMADE to achieve flexible levels of security. These
properties make it suitable for the design and implementation of distributed services in several application areas, viz., mobile computing, distributed database retrieval [70], network management [71-73], and information distribution [70]. Details about the security manager of PMADE are given in [10]. The MA paradigm provides the following key advantages:

- **Saves the network bandwidth and minimizes the latency**: MAs allow efficient and economical use of communication channels, which may have low bandwidth, high latency, and may be error prone. They reduce the network traffic because only shared resultant data is carried over the network and unnecessary intermediate results transmission is reduced. The other conventional distributed computing paradigms require repeated exchange of data over the network links. If a large amount of data is involved, the consumption of network bandwidth can be considerable, with a resultant delay in response time or latency. With MAs, a single serialized object is transmitted over the network, potentially reducing the consumption of network bandwidth.

- **Robust and fault-tolerant**: Since the MA needs only to be transported once between two sites, the possibility of failure due to network faults is reduced and thus MAs are especially appropriate for wireless network links. With proper implementation, an MA may have the intelligence to bypass the faulty hosts or seek temporary shelter on a reliable host. Agent code is not executed on demand but runs continuously and decides for itself when it should perform some activity. MAs encapsulate protocols for special purposes.

- **Enables the use of portable and low cost devices**: MAs enable the use of portable, low-cost, personal communications devices such as PDAs, to perform complex tasks even when the device is disconnected from the network. For example, an MA can migrate off a laptop and roam the Internet to gather information for its user. It can access the needed resources efficiently since it moves to their network location rather than transferring multiple requests and responses across the low-bandwidth laptop connection. Since it is not in continuous contact with the laptop, the agent is not affected
by sudden loss of connection, and can continue its task even if the user powers down or disconnects from the network. When the user reconnects, the agent returns to the laptop with the result of its travels. Conversely, an application that lives in the network can send an MA onto the laptop. The agent acts as the application's surrogate, interacting with the user efficiently and continuing to interact even in the event of long-term disconnection.

- **Autonomous**: Agents have capabilities of task selection, prioritization, goal-directed behaviour, and decision-making without human intervention, i.e., capability of modifying the way in which they achieve their objectives.

- **Dynamic adaptation**: Adaptation implies sensing the environment and reconfiguring in response. This can be achieved through the choice of alternative problem-solving-rules or algorithms, or through the discovery of problem solving strategies.

- **Scalable**: MA paradigm allows an application to scale well, since the number of participating hosts can be increased without any significant impact on the complexity of the application. The parent agent can also clone several child agents to implement concurrent operations, and raise running efficiency.

- **Smart systems**: We can program the agents moving behavior to implement smart systems. These smart systems are programs with intelligence that can help themselves to run smoother in certain conditions [12]. MAAs are naturally heterogeneous and goal oriented in nature. They cooperate, negotiate and collaborate with other agents.

- **Load Balancing by offloading processor and memory intensive tasks**: The agent’s mobility can be used to offload some of the processor and memory intensive tasks of a program to other more powerful and not heavily loaded platforms. This would take a significant load from the CPU and memory of the mobile device and speed up programs [12].

- **Recovering the network failure**: MAs can also be used to create programs which can recover themselves in case of a partial failure in the network. When the agents are running on different machines they are vulnerable. One or more machines running agents can crash or their connection with the other machines can be severed by a network failure. In this case a program
will most likely stop functioning and data can be lost. We can program an agent which may duplicate itself and recreate killed (crashed) agent in case of a failure. This way there is always at least one of each agent type operational so the program will not suffer a failure.

3.8 Mobile Agent System And Sensor Networks

Multiple approaches have been proposed for collecting data being sensed in the WSN in a flexible, reliable, and efficient manner [22]. The most relevant can be classified into multipath approaches, query propagation approaches, and MA approaches. Multipath approaches achieve a high degree of reliability by making use of multiple paths to send information from every SN to the collection point. Query propagation approaches tend to propagate an SQL-like query along a spanning tree that is rooted at the collection point and covers all SNs. Query propagation approaches treat the WSN as a large distributed database [23-26].

While pursuing the goal of collecting the sensed data in a WSN, instead of transmitting the raw data from the sources to the application in the collection point, the application (or a subset of it) is sent to where the data is [27-29]. Thus, MAs carry and aggregate the data being sensed. Moreover, MAs can be aware of network failures. This capability enables them to dynamically decide where to move or clone in the event of an unexpected failure or topology change. Therefore, MAs allow a great degree of flexibility regarding which data is collected and in what manner [30-31]. In terms of reliability, MAs provide a greater degree of fault-tolerance than query propagation and single-path approaches, comparable to multipath approaches. However, the time it takes the MAs to collect all the information (i.e., latency) tends to be larger than in other approaches such as multipath and single-path approaches.

MA is dedicated to reduce the information redundancy and communication overhead in the following ways for prolonging the network lifetime [22, 59-60].

(a) Application Redundancy Eliminating by MA Assisted local Processing With the development of WSN, "one deployment multiple applications" is a trend due to the application-specific nature of sensor networks. Such a trend must require SNs to have various capabilities to handle multiple applications, which is economically infeasible. In general, using memory-constrained
embedded sensors to store every possible application in their local memory is impossible. Thus, a way of dynamically deploying a new application is needed. A sink can assign the processing code (behavior) of MA based on the requirement of a specific application. The processing code carried by the MA packet only requires source nodes of local processing on the raw data as requested by the application. This capability enables a reduction in the amount of data transmission by allowing that only relevant information be extracted and transmitted.

(b) Spatial Redundancy Elimination by MA Assisted Data Aggregation [20]: The degree of sensed data correlation among sensors is closely related to the distance between sensors, so that it is very likely for closely-located sensors to generate redundant sensed data. The MA aggregates individual sensed data when it visits each target source. Though this kind of aggregation technique is typically used in clustering or aggregation tree based data dissemination protocols, the MA assisted aggregation does not need any overhead to construct these special structures.

(c) Communication Overhead Saved by MA Assisted Multiple Tasks’ Data Concatenation: Instead of MA assisted aggregating data of individual SNs right at the point of data sources, we need to propose a packet unification technique that unifies the several short data packets to one longer packet in order to reduce the communication overhead in combined-task level. Due to data concatenation, the duty cycle and communication overhead of intermediate SNs can be reduced so as to increase network lifetime. However, such energy savings can be achieved usually at the cost of the prolonged data latency. To reduce latency, multiple tasks should be executed simultaneously by replicated MAs in the combined-task region. This will decrease the overall execution time. Since replicating MAs incurs additional overhead, we should design carefully how many MAs should be dispatched in combined-task region depending on application's requirements.

MAs have been proposed for efficient data dissemination in sensor networks [14, 61], which differ from client/server based sensor network. In typical
client/server based wireless sensor model, nodes start collecting data as soon as event occurs. The data is then routed towards the sink (BS) based on some appropriate routing algorithm [50-51, 62-64]. Although there are many issues associated with MA such as safety, security, reliability and code caching, their successful applications ranges from ecommerce [14] to military situation awareness [5]. The agents are found extremely useful for data fusion task in distributed sensor network.

In [23-24] authors have adopted an energy-efficient, fault-tolerant approach for collaborative information processing among multiple SNs using an MA based computing model. In this model the sink/base-station deploys MAs that migrate from node to node following a certain itinerary, either pre-determined or determined on-the-fly, and fuse the information/data locally at each node. This way, the intelligence is distributed throughout the network edge and communication cost is reduced to make the sensor network energy-efficient.

In the literature we find that MAs are very useful with the help of WSN in environmental observation and forecasting [212], disaster management, river flow management, and agriculture [213]. In [214-215] authors presented a River water monitoring system for water pollution monitoring using sensing agents.

3.9 Wireless Sensor Network Middleware

WSN has very specific characteristics and requirements that are unique. A system for aiding in the development of sensing agents is often required [187,188,189,190-191]. A system (Middleware) is defined as software that connects software components to applications in order to overcome complex issues in distributed computing. Middleware solutions usually interface with the Operating System and sometimes hardware devices directly and provide a feature rich environment for applications to build from. This homogeneous environment allows agents to run on any node with the middleware, leaving all portability issues to the middleware itself [185-186]. Because middleware solutions have an extensive Application Programming Interface (API), applications need have less specific code and therefore tend to have a smaller footprint. This is especially advantageous in a WSN[183-184]. The smaller the agent application, the easier it fits onto the node or move to other nodes[180-181]. Middleware for WSNs needs to accomplish a variety of different tasks in order to be useful. First, the
middleware must provide a way for agents to communicate with other agents on different nodes within the network. This includes interfacing with the wireless transmitter/receiver, routing packets to other parts of the network, providing discovery techniques, and solving other P2P communication problems. Second, the middleware should try to assist agents to correctly and efficiently use the sensors. This will simplify agent development and allow more time and space for implementing intelligence within the agent itself [182].

**Lime**

LIME [168] is a Java-based middleware tool that provides an API for applications to use in a mobile setting. It is based on shared memory computing model. It is entirely based on Linda—the idea of communicating through reading, writing and deleting data from tuple space. In Linda, this tuple space is assumed to be globally available for all applications. However, this is not the case with Lime due to its distributed nature. Each tuple space is a set of tuples representing messages that are stored either for sending or receiving data on a given node. Each tuple is a sequence of typed fields, which contain the data used for communication. Tuples are either written and added to tuple space using an `out()` operation or read and deleted from tuple space using an `in()` operation. Tuples are inherently anonymous, so pattern-matching techniques are used when searching for tuples such as actuals ("foo") or formals (integer). A few other operations exist such as `rd()` for a non-destructive read, `inp()` and `rdp()` for asynchronous non-blocking access. Because Lime was developed for use in a mobile environment (such as a sensor network), each mobile node has its own tuple space and nodes can address and access other nodes’ tuple space. When connected together they form a federated tuple space, which is used to represent the same global tuple space made popular by Linda. While Lime’s use of tuple spaces is very advantageous to creating mobile applications for a wireless sensor network, Lime itself has a number of shortcomings. Primarily, Lime does not aid in developing intelligent agents in any way. Because of this, developing anything other than a simple reflex agent would require a great deal of work and application overhead, making any agents static to a specific node and unable to be upgraded.

**Agilla**
AGILLA is middleware solution which supports mobile agents in WSNs [169]. It is based on the older Mate middleware [170-171] and it was specifically designed for use on the MICA2 Mote (node) using TinyOS. Agilla and agents work on very limited hardware. It provides support for multiple agents to seamlessly move not only program code but also the current execution state to any node within the WSN. Agilla provides ability to agents to migrate. The sensor network can accomplish mobility logically rather than physically which simplifies the WSN’s requirements. It provides a variety of instructions that allow an agent to move or clone to other nodes. A weak move or clone simply transfers the code where a strong move or clone transfers the entire execution state. However a strong migration can have a significant affect on application overhead and performance do to the added complexity. Communication between different agents is also supported through the use of tuple space, which is set up similar to Lime’s federated tuple space. Agilla also supports read a specific remote tuple space or use multicast to query all one-hop neighbors.

As with any software, Agilla has its downfalls. While the mobility of agents provides a huge step forward, building an intelligent sensor network is extremely difficult. First, Agilla only supports the low-level assembly-like Mate language. It is very inefficient to program large amounts of code such as an expert system, let alone a fuzzy logic based system. Second, the hardware the Agilla software was written for is very slow. While it is designed to last a long time on simple AA batteries, an 8mhz ARM processor leaves much to be desired. Agilla also experiences significant overhead when cloning agents on new nodes, which can be very costly for the low power Mote devices.

Impala
Impala[172] is a middleware system using modular programming approach. The whole architecture includes two level layers: upper layer contains all the necessary protocols and application programs. The lower layer contains middleware agents such as Application Updater, Application Adapter, and Event Filter. Impala can support multiple different applications which located in upper layer by adapting, updating and event filtering data from lower layer. The Impala is original designed in ZebraNet Project, which focus on wildlife tracking in large area with few
communications devices. It has good performance on mobility, lower event processing time and lower application data transmission volume.

**Jade**

JADE[173] is a unique Java based middleware solution for a multiagent system that complies with the official FIPA specification in a streamlined and simplified way. It takes full advantage of Java’s interoperability, uniformity, portability, ease of use, and freedom to provide a rich feature set to develop a wide variety of agents on systems ranging from enterprise servers to low power wireless devices. Jade focuses on providing a communication architecture that is suitable for distributed data fusion. It is designed with the mobile agent paradigm (MAP) in mind, which specifies that all agents act as object and use high level communication mechanisms to interact with other agents within the network. Using the Jade API, some properties of intelligent agents such as autonomy, pro-activeness, cooperative and mobility can be easily cultivated. While Jade is used in a variety of distributed P2P environments, it can excel in a mobile environment such as a WSN. It can also use the LEAP module which can offload some of the resource hungry computational work to a backend container to try to conserve energy and overcome memory and processing power limits. When creating an intelligent agent for a WSN, Jade’s API can provide a wide range of support; however the intelligence of the agent itself is left entirely up to the programmer. Many Jade users have been able to successfully integrate Jess into the system to provide an intelligent solution with minimal effort. This combination has a substantial potential for integrating true intelligent agents in a WSN with powerful rule based expert system with Jade’s agent runtime environment.

**MANNA**

MANNA [174-175] is a policy-based network management system for wireless sensor networks. Depending on the network topology and characteristics (homogeneous vs. heterogeneous), MANNA assigns different roles (network managers or agents) to various SNs. These nodes exchange request or response messages with each other for management purposes. MANNA forms a basis for fault management [57], one of several network management services supported by
this architecture. MANNA network management protocol (MNMP), is a lightweight protocol for managing information exchange among management entities (cluster heads, common nodes, and manager) [175]. Basically, sensor nodes are organised in clusters (sub-network) and send their states to the agent located in the cluster-head. MNMP places management agents on the cluster-heads and each cluster-head acts as a manager for a cluster (local manager). Cluster-heads are responsible for executing local management functions and they aggregate management data received from SNs. Cluster heads forward management data directly to the BS. Furthermore, cluster heads can work cooperatively with other cluster-heads to achieve an overall. A manager is a powerful management entity located outside the WSN responsible for complex management tasks requiring global knowledge of the network. This approach achieves energy efficiency and increases the accuracy of management decisions. Fault management in MANNA mainly relies on the coverage area maintenance service and the failure detection service. Faults are detected in two phases in MANNA. In the installation phase, nodes report their location and energy level to the manager via the agents. The network manager builds coverage and energy models based on the initial information. During the operational phase, nodes update their location or energy whenever there is a change in their state. The network manager periodically performs network auditing by retrieving a node state. If a node which has enough remaining energy according to the energy model does not respond to the auditing, a fault is detected. This scheme has a drawback of possibly providing false debugging diagnostics. For instance, common-nodes may be disconnected from their cluster-head. Random distribution and limited transmission range capability of common nodes and cluster-heads provide no guarantee that every common-node can be connected to a cluster head.

BOSS

In [176] authors proposed a service discovery management architecture for WSNs. The architecture is based on UPnP, the standard service discovery protocol for network management. However, UPnP only runs on devices with high computation power and large memory. Thus, resource-constrained SNs are unable to process the UPnP protocol. Authors address this issue by implementing an UPnP agent in the BS, called Bridge Of the Sensor (BOSS), which provides a
bridge between a managed sensor network and a UPnP network. The proposed system consists of three main components: UPnP control point, BOSS, and non-UPnP SNs. The control point is a powerful logical device with sufficient resources to run the UPnP protocol and manage a sensor network using the services provided by BOSS, e.g. PCs, PDAs, and notebooks. BOSS is a base node that acts as the mediator between non-UPnP sensor nodes and UPnP control point and is implemented in the BS. Each node in a sensor network is a non-UPnP device with limited resources and sensing capability.

The control point and BOSS use UPnP protocol to communicate with each other, while non-UPnP SNs and BOSS use a sensor network proprietary protocol for communication. Thus, the human manager can implement any sensor network protocol to transport data from non-UPnP sensor nodes to the base node. BOSS has three functions. First, it is used to transfer UPnP messages between the sensor network and the control point. Second, since UPnP protocol uses XML message format that is different from a sensor network specific network message format, BOSS is responsible for interpreting transferred UPnP messages. Lastly, BOSS gathers network management information from SNs to provide network management services from each SN to the control point. The base node stores network management services and performs all network management processing. Thus, the base node carries the network management computation burden, rather than the resource-constrained SNs. In the BOSS architecture, the control point can specify which events of non-UPnP sensors it is interested in. BOSS then processes events reported by non-UPnP sensors to the control point.

The network management services provided by the BOSS include basic network information, localization, synchronization, and power management. BOSS can retrieve basic network state information from the sensor network, including sensor node device description, the number of SNs in the network, and the network topology. The localization service gives location information for each sensor in the network. The synchronization service is responsible for performing clock synchronization among nodes in the network. For example, when a node is added in the network, the node sends a message to BOSS and BOSS informs the control point. The central point then synchronizes the new node with other nodes in the network. The power management service allows a
human manager to manage the power of the SNs by checking remaining battery and changing the operation mode of sensors.

The advantage of using BOSS is that different sensor network applications (e.g. Structural monitoring, fire detection, and auto light control) can be managed by multiple UPnP control points (e.g. PCs and PDAs). Furthermore, BOSS allows a sensor network to adapt to topology changes and so supports proactive network management. A drawback of BOSS is that it requires an end-user to observe network states and take management actions accordingly.

**Intelligent Agent-Based Power Management (IABP)**

In [177] authors have designed an intelligent agent based power management system (IABP) using the Belief, Desire and Intention paradigm [178]. In IABP, beliefs represent states of SNs that an agent holds to be true. Commitment rules (or desires) are pre-defined conditions to evaluate beliefs. If beliefs match commitment rules, the corresponding commitment management function (intention) will be executed. This agent-based approach is designed for applications where only a partial view of the state of the network as a whole can be known at any one location or time [177]. IABP agents make power management decisions locally based on requirements of an application [177]. By using agents, information exchange between nodes in a neighborhood in order to make a local decision can be eliminated since agents collect node data and process it to meet a specified goal. The BS could inject a mobile agent into a sensor network to evaluate battery level of sensors in the network. This agent could also command nodes to reduce the sampling rate of sensors if their battery level is low. This scheme allows the BS to assess network states locally rather than gathering SN states to the BS.

Power management is also strongly related to other network attributes such as coverage, accuracy, battery longevity, and latency. The proposed agent-based approach [177] can perform complex decision making for various energy saving strategies. Users can specify desired sampling frequency, transmission range, and node mobility. Agents can be used to redirect traffic or change a link between nodes in the network to ensure a balance between energy conservation and network coverage. When a node’s battery level is critical, the agent finds another nearby node that can forward data. End users can also control sampling frequency
by commanding SNs to transmit only when there is something worth reporting. The energy preserved by reducing transmissions allows a greater sampling rate of SNs, which usually increases the accuracy of sensor data. However, when data polling rates are reduced, there is a risk of missing a crucial event. End users can command that nodes reduce their transmission power in order to conserve power. However, since reducing transmission power reduces communication range, this scheme may compromise network connectivity. The degree of agent mobility freedom allowed in the network can influence the latency of data collected from SNs to the user.

<table>
<thead>
<tr>
<th>Network management system</th>
<th>Main management functionalities</th>
<th>Energy efficiency</th>
<th>Robustness</th>
<th>Adaptability</th>
<th>Memory efficiency</th>
<th>Scalability</th>
<th>Fault tolerant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agilla[169, 179]</td>
<td>Event detection</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MANNA[174, 57,175]</td>
<td>Policy-based management framework, network state retrieval, sampling frequency control, coverage maintenance, and fault detection</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>Yes</td>
</tr>
<tr>
<td>BOSS[176]</td>
<td>Network state retrieval, localization, synchronization, and power management</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>TABP[177-178]</td>
<td>Local power management and sampling frequency control</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Lime[168]</td>
<td>Shared memory computing model</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Impala [172]</td>
<td>Component based Management System</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Jade[173]</td>
<td>Agent Management System</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Table 3.1**: A Comparative Study of WSN Management Middlewares
3.10 Results and Discussion

The systems are characterized by their power consumption, memory consumption, bandwidth consumption, fault tolerance, adaptability, and scalability. Most of the reviewed systems do not provide network health monitoring, fault detection, traffic management, congestion avoidance, power management, and resource management. Furthermore, most of these systems incorporate management functions within application protocols. The development of general purpose network management layer protocols is a challenging problem and remains a largely unexplored area for WSNs. Another significant open problem is the development of management policies and expressive languages or metadata for representing management policies and for representing the information exchanged between SNs, managers, and end users.

3.11 Summary

In this chapter a literature survey on routing, data aggregation, dissemination schemes, fault tolerant schemes, MA challenges and issues, mobile agent systems (MAS) in WSNs, and Middleware for WSN is done. It further surveyed security requirements, possible attacks and elaborate the work done by other researchers. Finally a comparative study of Middleware for WSNs is also done.

In the next chapter we will see architecture of Scalable Agent Pedestal (SAP) for WSN.