International Journal of Computational Intelligence Techniques
ISSN: 0976-0466 & E-ISSN: 0976-0474, Volume 3, Issue 2, 2012, pp.-90-98.
Available online at http://www.bioinfopublication.org/jouarchive.php?opt=&jouid=BPJ0000221

A SCALABLE AGENT PEDESTAL FOR WIRELESS SENSOR NETWORKS

PATEL R.B.[1]* AND JAIN D.[2]
[1]Department of Computer Science and Engineering, G.B. Pant Engineering College, Pauri-Garhwal-246194, Uttarakhand, India.
[2]Department of Computer Engineering, M.M. Engineering College, Ambala-134003, Haryana, India.
*Corresponding Author: Email- patel_r_b@yahoo.com

Received: October 25, 2012; Accepted: November 06, 2012

Abstract- Wireless sensor network (WSN) is envisioned as an economically viable paradigm and a promising technology because of its ability to provide a variety of services, such as intrusion detection, weather monitoring, security, tactical surveillance, and disaster management. In this article we present a Scalable Agent Pedestal (SAP) for WSNs. In SAP mainly mobile agent (MAs) are used to manage the WSNs. It provides common solution to WSNs fault tolerance, load balancing, energy & end-to-end network delay problems. SAP gives true distributed computing and communication environment with the help of MAs. It supports code mobility over the mobile/fixed base station (BS) and sensor node (SNs). A comparative study is also made.

Keywords- WSN, MA, SN, SAP, AMS

Citation: Patel R.B. and Jain D. (2012) A Scalable Agent Pedestal for Wireless Sensor Networks. International Journal of Computational Intelligence Techniques, ISSN: 0976-0466 & E-ISSN: 0976-0474, Volume 3, Issue 2, pp.-90-98.

Issues

Distributed Wireless sensor Networks (DWSN) has become a very popular research topic due to its wide application spanning across civilian and military domain, including environmental monitoring (e.g. temperature sensing), generic object tracking (e.g. people or object locator), and surveillance in a large building, disaster area or battlefield [16,19]. The advances in sensor technology and ad hoc wireless networking have brought the study of DWSN to a new stage the emergence and spurs of Wireless sensor Networks (WSNs). It is economically feasible today to implement WSNs, but there are several technical challenges that must be overcome before they can be used for the increasingly complex information gathering tasks. These tasks, such as battlefield surveillance, remote sensing, global awareness, etc., are usually time critical, cover a large geographical area, and require reliable delivery of accurate information for their completion [17,18].

A mobile agent (MA) is an autonomous transportable program that can migrate under its own or processing element (PE) control from one PE to another in a heterogeneous network [13,14]. In other words, the program running at a PE can suspend its execution at an arbitrary point, transfer itself to another PE (or request the PE to transfer it to its next destination) and resume execution from the point of suspension. Once an agent is launched it can continue to function even if the user is disconnected from the network. They implement a computational metaphor that is analogous to how most people conduct business in their daily lives: visit a place, use a service, and then move on. It behaves like a human agent, working for clients in pursuit of its own agenda. When an agent reaches a server, it is delivered to an agent execution environment. If it agent possesses necessary authentication credentials, its executable parts are started. To accomplish its task, the MA can transport itself to another server in search of the needed resource/service, spawn new agents, or interact with other stationary agents. Upon completion, the MA delivers the results to the sending client or to another server.

It has been found that MAs are especially suitable for structuring and coordinating wide area networks and distributed services that require intensive remote real time interactions. 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 [25-26]. 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. 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 [15].

The new challenges brought to the study of WSN include [19]:

- Data volumes being integrated are much larger due to the increasing amount of sensors being deployed;
- The communication bandwidth for wireless network is much lower;
- The environment is more unreliable, causing unreliable network connection and increasing the likelihood of input data to be in faulty; and
- Fixed routing is impossible.

In addition of balancing the load across networks for the above defined issues, the system should provide Fault Tolerance, Optimal Resource Utilization, Route and Resource Discovery, Path Maintenance, Service Discovery, and Security.

Rest of the paper is organized as follows. Issues are explored in Section 1. Section 2 highlights on related works. System model is given Section 3. Section 4 explores on the architecture of Scalable Agent Pedestal (SAP) and Agent/SN namespace is given in Section 5. Advantages of SAP are discussed in Section 6. Section 7 presents results and discussion and finally article is concluded in Section 8.

Related Works

LIME [1] 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 and 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 do 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. 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 is middleware solution which supports mobile agents in WSNs [2]. It is based on the older Mate middleware [3-4] 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. 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. Agilla also experiences significant overhead when cloning agents on new nodes, which can be very costly for the low power Mote devices.

Impala [5] 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 Zebra Net 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[6] 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 [7-8] 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 [12], 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) [8]. Basically, sensor nodes are organized 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 re-

ceived 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.

In [9] 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 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.

In [10] authors have designed an intelligent agent based power management system (IABP) using the Belief, Desire and Intention paradigm [11]. In IABP, beliefs represent states of SNs that an agent holds to be true. Commitment rules (or desires) are predefined 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 [10]. IABP agents make power management decisions locally based on requirements of an application. 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.

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.

The current available solutions for WSNs have advantages and severe limitations with regard to performance issues. A better WSNs performance can be achieved with a more flexible and intuitive architecture which should be well-researched [21-24]. Thus, we need to design an adequate WSNs management system which should meet requirements/challenges for fulfilling need of an application.

System Model

A communication area is divided into different regions. A high energy node will be the member of a region. This high energy node in a region will work like backbone node (BN) and maintains information about other members of the region in the form of database. Each region would have a backbone node (BN). All the routing is carried out through BNs only. Thus, a BN acts as a centralized control for a region. BN is a node with maximum energy and maximum node degree in a particular region. All the nodes (SNs/BNs) are distributed randomly in communication area. In the model, distance table is assign to each node with an entry in terms of hop distance from every other node in the network. The distance is calculated using Euclidian formula.

MAXEN=32% of Etotal

MINEN=30% of Etotal

Euclidian formula to calculate distance table is:

$$D[i][j]=\text{sqrt}(((xi-xj)*(xi-xj))+((yi-yj)*(yi-yj)))$$

Each node sends request to every other node in their region containing information of their node degree and energy. The node with maximum number of node degree and maximum energy is selected as a BN of that region. Once the BN is selected, all the nodes in that region are notified. All BNs of different regions should be in transmission range of each other.

We are required to develop a computing/communication system for WSNs that fulfills most of the above the challenges. The developed system should enable the fast and cost-efficient deployment of self-managed computing/communication sensing devices with high overall management cost, but with low management cost at each

SNs. With developed system one should be able to deploy large scale computing/ communication systems without the need of cost-intensive distributed sensing infrastructure to monitor sensitive area with long life network. This system should facilitate to improve the performance and incorporate new ideas.

System Architecture

When a BN/BS wants to search some information it requests to BN for members information (viz. ID, energy level, node degree, etc.). If the BN is not aware about availability of the type of services a BN/BS is interested and presence of the same in the region then it guides the same to the BN/BS. Then BN/BS uses a scalable agent pedestal (SAP) for fault tolerance, load balancing, energy efficient and efficient end-to-end packet delivery across the network for WSNs and creates a MA to perform its desired task in the present region.

Keeping in view of the above defined issues, we have designed and implemented A Platform for mobile agent Distribution & Execution (PMADE) [13-15] based a Scalable Agent Pedestal (SAP) for WSNs [Fig-1]. The main components of the system are as follows- Policy Manager, Resource Manager, MAs, Interface, and Agent-Agent communication layer.

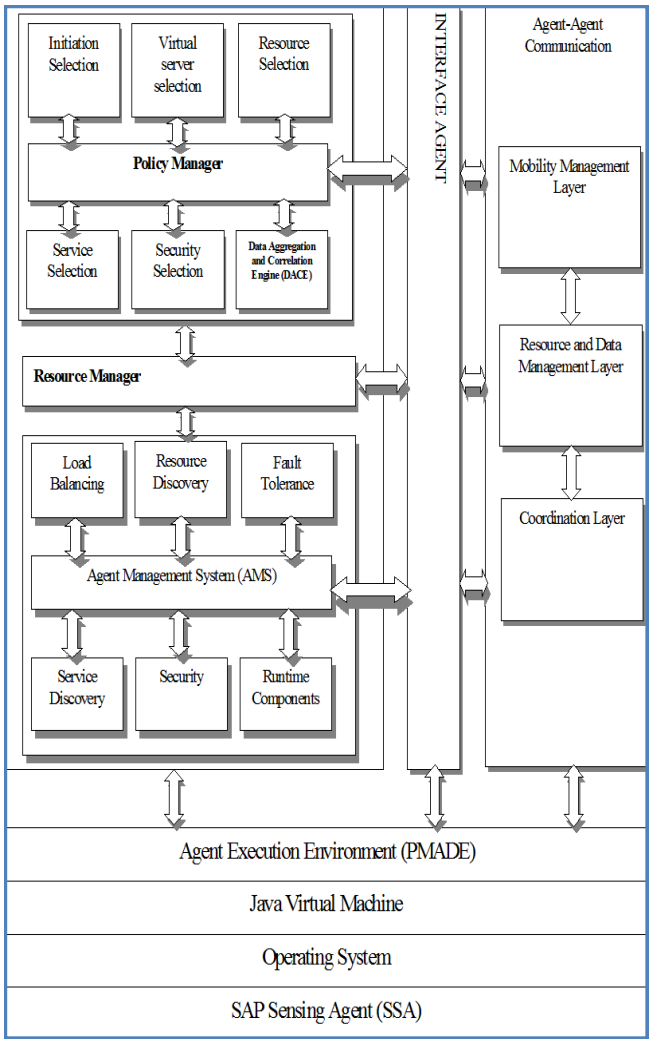

Fig. 1- Scalable Agent Pedestal (SAP) for WSNs

This system implements the agents for Load Balancing, Sensing Agents, Application Agents, Interface Agents, etc. depending upon their roles. These agents execute the predefined policies for network management and share the valuable information with each other through different layers using interface for the communication. The various components of the system are illustrated in the following sections:

Policy Manager

This manager comprises of different supporting policies and a policy is selected as per requirement of an application.

Initiation Selection Policies

It selects one of the predefined policies for executing the MAs. The application selection plays a major role for categorizing and grouping the policies. The policies defined in the development of SAP may be subcategorized into the following:

The load balancing process can be started by under-loaded or the overloaded BN. The overloaded BN is responsible for finding other BNs to share its workload (BS Initiative (BS-I)). If a lightly loaded BN is the initiator then it is called receiver initiative (R-I). When above two policies are integrated together, i.e., mixture of first two (BS-I and R-I). It is a hybrid initiative called Symmetrical Initiative (Sy-I).

Virtual Server Selection

This policy is further subcategorized into the following: first policy is based on the one-to-one mechanism, where two BNs are picked at random. A virtual server transfer is initiated if one of the BNs is heavily and the other is lightly loaded. Unlike the first scheme, this scheme allows a heavily loaded BN to consider more than one lightly BN at the time of transfer of load. Third scheme is a logical extension of the first two schemes (one-to-one, one-to-many). While in the first scheme we match one heavily loaded BN to a lightly loaded BN and in the second scheme, we match one heavily loaded BN to many lightly loaded BNs, in this scheme we match many heavily loaded BNs to many lightly loaded BNs.

Resource Selection

There are various resources available in DWSN. These resources are either available at local site (at the sensing unit, i.e., SN) or at global site (BN/BS). To achieve quick response time and high throughput, proper resource scheduling is always necessary. The resource selection is further subcategorized into the following: (a) Route discovery- This policy is used to select the path from the available SN/BN disjoint paths. (b) Topology update- DWSN is a dynamic network where SNs dead frequently. In this network, topology changes quickly. Hence this policy is used to update the topology of the network. (c) Cost computation- There are various node disjoint paths available from source (SN) to destination (BS). The best path from the available paths is that which is most reliable in terms of data transmission. This policy is used to compute the cost of the available path in terms of reliability.

Service Selection

For the sake of simplicity, we have selected three different types of policies for service selection namely multicasting, topology update, and broadcasting. This mechanism is used to avoid delay in trans-

mission of sensitive information from SNs. Topology changes occur frequently in WSN, so to face the challenges due to frequent topological changes each SN sends a hello message to its neighbors.

Security Selection

Security policies are based on the PMADE on the security framework [27].

Data Aggregation and Correlation Engine (DACE)

It is necessary for the DACE to aggregate and correlate the different detection results before further transmission. The local DACE for a mobile BS (MBS) are capable of operating in a standalone mode and detect attacks against the SN/BN. Since WSNs are constrained by bandwidth, energy consumption, and process capability, it is desirable to correlate the alert information on the local SNs first, before transmitting every alert across the network.

The functionality of the DACE depends on the node type: if the node is a BN, its DACE utilizes the aggregation and correlation to combine the detection results from the intrazone SNs in the same region and neighboring gateway BNs. If the node is an intrazone SN, the functionality of the local DACE is to distribute the outputs to all the gateway BNs in the same region.

Agent Management System (AMS)

The main task of AMS is management of agents and nodes (SNs/ BNs/BSs), i.e., registration, authentication, security and mobility as per PMADE [13-15,27]. AMS can also be extended to support clustering and teaming.

AMS selects the agents as per their requirement. These agents are not fixed but vary according to their role. Agents are divided into different groups and they are picked by AMS depending upon their requirement. Each agent has a role defined and executes the predefined policies. The generalized architecture of the developed SAP consists of the following groups of agents which are used in heterogeneous networks. This list of agent is not only limited to following groups also.

At present AMS contains nine SAP agents- SAP Mapping Agent (SMAPA), SAP Route Estimating Agent (SREA), SAP Migration Planning Agent (SMPA), SAP Code Container Agent (SCCA) and SAP Result Container agent (SRCA), SAP Sensor Agent (SSA), SAP Broker Facilitator Agent (SBFA), SAP Application Agent (SAA), and SAP Interface Agent (SIA) in future number of agents may be increased as per need of the applications, means developed system is adaptable in nature. These agents are called SAP agents because at any moment of time as per requirement of the applications algorithm/protocols associated with these agents are changed/updated or new amendment can be made. SRCA and SCCA facilitates distributed environment for adapting the nature of the network bandwidth. These agents are identified as single entity known as agent management system (AMS). AMS is named because of its nature to accommodate any kind of changes occurring in the system. Other component of AMS is Network manager (NM) which is responsible to identify the topology of the network with assistantship of SMAPA. NM provides global identification to mobile devices and MAs. AMS supports code mobility over the mobile/ fixed peer device. For balancing the load over the network these agents work together as a SAP multiagent system.

Load Balancing

For balancing the load across the network, we have developed a set of agents which are briefly introduced as under.

Load Index Agent (LIA)- This agent calculates the load index (LI) of each resource (Processing Unit: PU, Memory, I/O) on a particular node (SN/BN). LI of PU is the sum of remaining PU lifetimes of the tasks running on a node. LI of memory is the sum of page fault processing time of tasks on a node (SN/BN). Similarly LI of I/O is the sum of I/O processing time of tasks on a node (SN/BN). The sum of load index of each resource is the total value function for a node (SN/BN) and is used for load transfer by the respective agent. Normally this cared by BNs, because we are balancing the load on BNs only. If model will be used at global network (Internet) level then BS is also included in the list for managing the load.

Load Transfer Agent (LTA)- This agent is used for migration of task from heavily loaded BN to lightly loaded one. It executes two predefined policies namely–local and global. The policy is chosen according to the response time of task submitted for execution. If the response time of a task at the local site is less than the global site then local policy is executed otherwise global policy.

Resource Management Agent (RMA)- This agent is responsible for gathering information about each node's (SN/BN) resource requirement and passing this information to Resource Manager (RM)[14], which makes its entry in resource database and allocates the appropriate resource to the requesting node.

Routing Agent (RA)- It is a stationary agent responsible for updating the routing table that resides at each node (SN/BN). RA carries a route vector table containing the communication cost from the assigned node (SN/BN) to other nodes (SN/BN) in the network. This table has a lifetime measured by the number of hops. It plays an important role in informing each node (SN/BN) in the network about the addresses of other nodes (SN/BN) and if failure of link for a particular node (SN/BN) is detected, RA spreads route failure information over the network by flooding the updated table.

Load Computation Agent (LCA)- This is a set of MA. It is responsible for information gathering. It travels around the BNs and collects the load information, and propagates this information to other BNs.

Directory Agent (AD)- This agent is activated whenever an overloaded situation arises on a BN. AD finds the suitable receiver partner for the overloaded BN that launched it.

Resource Discovery

For route discovery and balancing the load across the network, a set of agents are developed. A brief introduction about these agents is as follows:

Route Discovery Agent (RDA)- This agent keeps a record of SN disjoint paths which are not deemed failed yet. As soon as the rating of a path falls below a given threshold, the path is discarded from the Active Path Set (APS) (set of SN disjoint paths) and accordingly a new path is added for future references. This agent executes route discovery policy.

Topology updates Agent (TUA)- This agent keeps record of one hop neighbors using the hello message technique. As in WSN,

SNs died the network in a random interval of time, so every SN/ BN should have knowledge about its neighbors. This agent executes topology update policy. For simplicity BN maintains information about SNs in its region.

Cost computation Agent (CCA)- This agent computes the cost of sending the messages to another SN in the network. The cost is measured by the number of hops traveled by the message and bandwidth lost due to the presence of selfish SN in between the paths. As the battery power of SN is a sacred resource in WSN, this agent plays an important role in cost computation in terms of bandwidth loss. This agent executes the cost computation policy.

Fault Tolerance

For fault tolerance across different types of networks, a set of agents are developed. They are briefly introduced as under-

Process monitor agent- This agent monitors the state and starvation of a process in a task queue. It classifies a process state into a processing state, a stop state, a silent state, and an unknown state.

Processor monitor agent- This agent monitors the crash state of a node (SN/BN) (shutdown, power value) and the normal state of a node (SN/BN). During the normal execution of a processor, this collects the used and the available node (SN/BN) processing power utilization.

Network monitor agent- This agent monitors communication bandwidth, communication latency time, network disconnection, and partition between its own node (SN/BN) and connected nodes (SN/BN).

Fault decision agent- This agent decides the occurrence of a failure by analyzing state information of each resource and identifies a process failure, a node failure or a network failure.

Rescheduling agent- This agent evaluates the performance benefits that can be obtained due to task migration and decides whether task migration occurs or not. This agent also decides a new resource allocation for tasks.

State display agent- This agent shows the state of each resource and the type of failures occurred. Also it decides whether task migration occurs or not. If this agent receives a rescheduling result for migration from the rescheduling agent, it requests to allocate new selected resources and restarts execution.

Service Discovery

For selection of service across the network, we have developed a set of agents which are briefly introduced as under.

Advertising Agent

This agent actively broadcasts service descriptions already registered. The Policy Manager controls the rate of advertisements. Various policies are employed to adjust the rate of advertisement. For example, if the network is fairly static, then the advertisement rate can be slowed down. Also policy is event driven (Events represent the availability of the paths from source (SN) to destination (BS)). Advertisements can also be assigned different priorities.

Forwarding Agent- This agent receives service advertisements and requests for service messages. Then it decides whether to drop or to propagate the advertisement based on the policy. To prevent broadcast storms, this agent uses multicast tree for selectively forwarding service advertisements. For example, this uses to forward advertisements to more active or resource rich SNs in the network.

Cache Agent- This agent is responsible for handling remote advertisements, storing remote advertisements of services, handling requests to match services present in the cache. The Forwarding Agent, on receiving an advertisement might also decide to forward it to other SNs/BNs or broadcast the advertisement to all other BNs and from it to SNs. Each advertisement contains a lifetime. When a new advertisement is received by a Cache Agent, the agent decides to either accept it or reject it. An advertisement is accepted only when there is sufficient space in the cache to hold this advertisement or when an old advertisement is removed from the cache based on the policy chosen.

Security

A set of agents are developed for secure data transmission across the network. These agents are briefly introduced as under:

Multi path Secure Routing Agent- This agent executes the multipath policy which has multiple paths to combat the frequent topological change and link instability problem in WSN, since the use of multiple paths could diminish the effect of possible link failures.

Secret Sharing Agent- This agent executes secret sharing policy. In this policy, the secret message is divided into *N* pieces such that in order to get message, the adversary must compromise at least *T* shares. With fewer than *T* shares, the enemy cannot learn anything about the message and has no better chance to recover the secret than an outsider who knows nothing about the message. This gives the desirable security properties.

Share Allocation Agent- This agent executes share allocation policy with the objective of maximizing the message security. It chooses a SN disjoint path for secret allocation so that adversary can never get the message.

Resource Manager (RM)

Resource Manager (RM) manages the resources in the network. Each task (agent) in execution has its own resource requirement, which is provided by RM. It keeps track of which resource is available at which node (SN/BN). Resources may be at local site or at global site. As soon as the demand of resources comes from the tasks (agents) in execution, these are provided by the RM. MAs execute predefined policies to provide the desired resources to the demanding task (agent). In this process they also consume certain resource like memory, processor time, etc. So, to keep track of the all these resources, RM is included in the architecture which fulfills the demand of resources as per the requirement.

Agent-Agent Communication

SAP includes P2P type communication between agents. It supports both local & remote communication and uses wireless transmission as the transport mechanism for remote communication. Agent Communication Channel (ACC) is a message routing agent integrated in the SAP, which delivers messages as requested by their senders. It supports both local as well as remote communica-

tion. The agent-to-agent communication, also known as the agent-based messaging paradigm, and is uses message queue processor [29] for making the different type of communications viz. federated communication, direct communication, etc[28-29]. Agents can communicate directly or indirectly through the ACC. The agents can be located at the same node (SN/BN/BS) or communicate wirelessly between remote nodes (SN/BN/BS).

For information processing in WSN applications, SAP supports both types of computational models, namely, remote communication between agents located at different nodes (SN/BN/BS), as well as agent mobility with local communication at the same node (SN/BN/BS). In the CSCP, the SAP Sensing Agents (SSAs) located at the constituent SNs communicate remotely with the agents in BNs. In the MACP, the mobile SAAs move from SN-to-SN and communicate locally with the SSAs, residing on the same SNs. This layer provides the facility for agent-agent communication. There are three layers for agent- to-agent communication.

Resource and Data Management Layer- RM which is the key component of the system architecture operates on this layer. As discussed earlier it is responsible for managing the resources consumed by MAs when it executes on a particular site. On each layer the respective MA operates.

Coordination Layer- MAs communicate and coordinate using communication and coordination layers. The request an agent receives from the communication layer is submitted to the coordination layer for further processing. Agents communicate by exchanging messages using mobile group approach through reliable communications channels. This is normally happen between BS-BS, BN-BN, BS-BN and SN-BN.

Mobility Manager Layer- SAP supports agent mobility. It provides ability to agent to move from one node (SN/BN/BS) to another in the network. Migration of an agent from source (BS) to destination (SN) and vice-versa may trigger updates for the service provider agents.

Among the different components of the MA, the most important is the MA's itinerary. Itinerary can be determined either statically, i.e., it can be calculated either before the agent is dispatched or while the agent is migrating. Dynamic itinerary planning is more flexible, and can adapt to environmental changing (sensor ups and downs) in real time. However, since the itinerary is calculated on the fly, it also consumes more computation time and more power of the local sensor. In the SAP this job is done by BNs and exchange between BN and BS which saves energy of SNs. Computation-efficiency, power-efficiency, and flexibility are three conflicting objectives that cannot be satisfied at the same time.

A simplified sub-optimal solution that determines the agent itinerary on the fly based on three parameters obtained in real time, namely, the remaining energy on board the SN, the signal energy sensed at the current location, and the geographical distance with possible neighbors. When the packet size is the same, the communication cost is proportional to the distance between the source and the destination, which can be calculated using the longitude and latitude information exchanged between nodes(SN/BN). The SAP Broker Facilitator Agent (SBFA) on each SN is responsible for exchanging information through ACC when the SN is initially activated and when dramatic changes have occurred to any of these three parameters. Therefore, the SBFA on each SN is able to calculate the cost for migrating to each neighboring SN.

Before the MA migrates to the next destination, the SBF compares the cost and directs the SAP Application Agent (SAA) to migrate to the one with the lowest cost, that is, the SN with the high remaining energy, sensed with high signal energy, and very close to the current SN.

Agent and SN Namespace

This system maintains a local ID at each layer. These ID's are down-streamed at the boot up time, i.e., the ID of system high up in the hierarchy will be sent to all the lower layers. The layer low in this hierarchy will prefix the parent layer ID to its own local ID and this combination forms the new ID of the layer. The lowest layer (layer 7^{th}) will be at the node/PE level. Each agent/SN will have a 12-digit Hexadecimal ID. Format of ID is shown in [Table-1]. In this format we have 1-digits to represent a country code, 1-digit for state provincial, 1-digit for sub-state provincial, 1 Digit for next sub-provincial, 1 digit for next sub-region, 2 digits for next sub-region, 2 digit for smallest region beyond that area will not be divided and 3 digits for representing the agent/SN identification number (ID). Out of 3 bytes- 1 byte is used by representing BNs and remaining 2 Bytes for SNs, i.e., 64 K SNs/BN.

Table 1- Format of 12 digit hexadecimal id

Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6	Layer 7		
Country Code	State Provincial	Sub-State Provincial	Sub-Provincial	Sub-Region	Sub-Region	Smallest Region	Agent ID	Source of Data
1 - 256	1 - 256	1 - 256	1 - 256	1 - 256	64 K	64 K	64K	Maximum range
1 BYTE	1 BYTE	1 BYTE	1 BYTE	1 BYTE	2 BYTE	2 BYTE	3 BYTE	ID in Hexadecimal

This format enables a total population of SNs 16M nodes in a smallest area where deployment of agent may be done. Thus, by using this assumption 2^{88} agents/SNs can be uniquely identified across a country and 2^{96} SNs/agents across the world which is quite a good assumption to manage a whole world using agents and SNs. The first time registration will be only through the layer 7. This process will append the agent/SN ID with the system ID thus giving a unique ID to each SN/agent. For example, see [Table-1] where we are getting "111111111111" for SN X belonging to country India (Layer 1), Haryana State (Layer 2), Ambala District (Layer 3), Barara tehsil (Layer 4), Narayangarh block (Layer 5), Panjlasa panchayat (Layer 6) and Seembla Village (Layer 7). Here "1" is a code of India and personal ID of SN X is "1111". This SN will be under BN 1 in the said area.

This ID will be up-streamed towards the root system via the branch of the hierarchical tree, thus making identity available to each of its parental ancestor layer. This way every SN/agent has a unique ID and will communicate to any ancestor layer via its local layer. Any information may be collected or disseminated across the whole network or particular region, etc. In case of certain failure, maliciousness the node/agent can directly communicate with its next ancestor higher in the branch because of P2P nature of this system. This system also prevents any unauthorized access by a SN/agent to any other system to which it dose not belong. This is because the ID of that SN/agent will be supported only by the branch

to which it belongs. In case of roaming of mobile BS, the system may be enhanced to provide only limited privileges to the agents like query, etc.

Advantages

The SAP saves network bandwidth, because only results travel on the network and not the whole agent. Further, SAP agents regularly reduce their size by removing processed methods which are no longer required during their itineraries. When there are several clones of an agent running, saving in bandwidth is quite substantial. It also saves processing power, secondary and primary storage devices at the remote node.

From the security point of view, agents are required to report results to the BS/BN in a secure manner, so that further tampering of the result is not possible by the BN. In SAP, the agent provides the result in encrypted form to the BN.

If an agent is running on a machine which is disconnected from the network, its results remain on the machine for a fixed period of time. When the machine is reconnected the results are sent to the BS/BN. However, if this duration is large, the BS/BN removes the entries of the pending agents and the result is lost. The system administrator can set this duration as per the application requirements.

SAP is component oriented, both in system environment and in agent construction. The basic component of the system is a MA from which both the system components and user agents are built. Component orientation allows the application developer to adopt a systematic design which is extensible and in which ideas such as transport protocols or communication mechanisms from different systems can be imported in the form of plug-in components.

A useful aspect of the design of SAP is modularization to facilitate experimentation. It is possible to redesign and re-implement a single module within SAP, without affecting any others.

Major advantages of remote agent creation of the SAP are that user does not have to install the agent platform on the mobile device. The system also reduces communication over wireless links to overcome low bandwidth and network disconnection. SAP enhances service functionality by operating without constant user input. SAP is platform independent. One of its important features is its flexibility and extendibility.

Results and Discussion

WSNs have continually proven their usefulness in many different fields ranging from temperature regulation to traffic monitoring. Intelligent agents are allowing such WSNs to be more usable, productive, and available and they will continue to do so as the technology continues to develop. Because of hardware constraints, using intelligent agents on WSNs can be a very difficult task, which is why middleware technologies are needed. Middleware software can reduce the memory footprint, provide mobility to agents, help to maintain agents in the network and significantly reduces development time.

There are different middleware software solutions for WSNs available today, each with their own advantages and disadvantages. When MAs are desired on severely limited hardware such as a Mote, SAP may be the good solution because of its small footprint and great support for agent mobility. Agilla is also good and support mobility to agents. If there is more powerful hardware is available, Jade, along with Jess may be a viable solution because of Jade's feature rich API and possible expert system shell. If dynamic updates are needed, SAP and Impala may be considered.

It is essential to evaluate each situation and possible choice carefully to identify the best solution. The final solution depends upon the existing requirements of the implemented WSN coupled with the best possible middleware to create the desired outcomes.

A comparative of SAP with some existing WSN management middleware are shown in [Table-2]. From the table it is clear that SAP is agent and component based system which is Energy Efficient, Robust, Adaptive, Scalable and Fault tolerant. It is not memory efficient because of agent framework size.

Table 2- A Comparative Study of SAP with some Existing WSN Management Middleware

Network Management System	Main Management functionalities	Energy efficiency	Robustness	Adaptability	Memory efficiency	Scalability	Fault Tolerant
Agilla	Event detection	Yes	No	Yes	Yes	No	No
MANNA	Policy based management framework	NA	NA	NA	NA	NA	Yes
BOSS	Network state retrieval, localization, synchronization, and power management	Yes	Yes	Yes	Yes	No	No
IABP	Local power management and sampling frequency control	Yes	No	Yes	Yes	No	No
Lime	shared memory computing model	Yes	No	No	Yes	No	No
Impala	Component based Management System	Yes	Yes	No	Yes	Yes	No
Jade	Agent Management System	No	Yes	Yes	No	Yes	Yes
SAP	Agent and Component based Management System	Yes	Yes	Yes	No	Yes	Yes

Conclusion

In this article we have presented a Scalable Agent Pedestal (SAP) for WSNs. In this system mainly MAs are used to manage the network. SAP is capable of balancing the load across network. It uses set of MAs which are executing predefined policies for finding the load and resource requirement status at each node (SN/BN/BS). Due to the support for mobility of agents in heterogeneous networks, it is able to locate the resource and service in WSNs. Inter-agent communication is supported by the system to get the updated load information on each SN. It generates less message transfer complexity and overhead compared to other existing systems developed earlier. Also it has a major impact on the efficiency of the network with the following performance measurement metrics-response time, energy, throughput, fault tolerance and end-to-end network delay. We are in the process of studying and design fault tolerant and mobility management of the SAP.

References

[1] Murphy A., Picco G.P. and Gruia-Catalin R. (2006) *ACM Trans-*

actions on Software Engineering and Methodology, 15, 279-328.

[2] Fok C., Roman G. and Lu C. (2005) *Proc. IEEE ICDCS Conf*.

[3] Levis P. and Culler D.E. (2002) *Architectural Support for Programming Languages and Operating* Systems.

[4] Blum B. (2006) *Mate-VM for Sensor Nets*.

[5] Ting L. and Martonosi M. (2003) *ACM SIGPLAN Symp. Principles and Practice of Parallel Programming*, Princeton University.

[6] Bellifemine F., Caire G., Poggi A. and Rimassa G. (2003) *Introduction JADE a White Paper*.

[7] Ruiz L.B., Nogueira J.M. and Loureiro A.A.F. (2003) *IEEE Communications Magazine*, 41(2), 116-125.

[8] Ruiz L.B. (2003) *MANNA: A Management Architecture for Wireless Sensor Networks*, Ph.D. dissertation, Federal Univ. of Minas Gerais, Belo Horizonte, MG, Brazil.

[9] Song H., Kim D., Lee K. and Sung J. (2005) *Proc. ICMU Conf*.

[10] Zhang H. and Hou J.C. (2004) *Proc. NSF TAWN Conf*.

[11] Tynan R., Marsh D., OKane D. and OHare G.M.P. (2005) *IEEE ICPPW Conf*.

[12] Ruiz L.B., Siqueira I.G., e Oliveria L.B., Wong H.C. Nogueira J.M.S., Loureiro A.A.F. (2004) *International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems*, Venice, Italy, ACM Press.

[13] Patel R.B. and Garg K. (2001) 5*th World Multi Conference on Systemics, Cybernetics and Informatics* (SCI) and *7th International Conference on Information System Analysis and Synthesis* (ISAS), Orlando, Florida, USA, 4, 287-293.

[14] Patel R.B. and Garg K. (2004) *WSEAS Transaction on Computers*, 1(3), 57-64.

[15] Patel R.B. and Mastorakis N. (2005) *WSEAS Transactions on Computers*, 3, 4, 287-314.

[16] Akyildiz I.F., Su W., Sankarasubramaniam Y., Cayirci E. *IEEE Communications Magazine*, 40(8), 102-116.

[17] Culler D., Estrin D., Srivastava M. (2004) *IEEE Computer*, 37 (8), 41-49.

[18] Mhatre V., Rosenberg C., Kofman D., Azumdar R. and Shroff N. (2005) *IEEE Transactions on Mobile Computing*, 4(1), 4-15.

[19] Yu-Chee Tseng, Sheng-Po Kuo, Hung-Wei Lee and Chi-Fu Huang (2004) *The Computer Journal*, 47(4), 448-460.

[20] Chee-Yee Chong, Kumar S.P. (2003) *IEEE*, 91(8), 1247-1256.

[21] Georgoulas D., Blow K. (2008) *Fourth Advanced International Conference on Telecommunications*, 8-13, 95-100.

[22] Yongzhong Li, Jing Xu, Bo Zhao, Ge Yang (2008) 3*rd IEEE Conference on Industrial Electronics and Applications*, 1562-1565.

[23] Xiaofeng Han, Xiang Cao, Lloyd E.L., Chien-Chung Shen (2007) *IEEE INFOCOM,* Anchorage, Alaska, 1667-1675.

[24] Cui Yanrong and Cao Jiaheng (2007) *International Conference on Wireless Communications, Networking and Mobile Computing*, 2372-2375.

[25] Biswas P.K., Xu Y., Qi H. (2008) *Information Fusion*, 9(3), 399-411.

[26] Wu Q., Rao N.S.V. and Barhen J. (2004) *IEEE Transactions on Knowledge and Data Engineering*, 16(6), 740-753.

[27] Patel R.B. and Garg K. (2005) *Control and Intelligent Systems*, 33(3), 175-183.

[28] Patel R.B., Kumar N. (2010) *International Journal of Mobile Computing and Multimedia Communications*, 2(3), 34-46.

[29] Patel R.B., Garg K. (2002) *First International ICSC Congress on Autonomous Intelligent Systems*, Deakin University Waterfront Campus, Geelong, Australia, 107.