jet INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2012, VOL. 58, NO. 1, PP. 55–62
Manuscript received December 31, 2011; revised March 2012. DOI: 10.2478/v10177-012-0008-4

Sensor-Actor Network Solution for Scalable Ad-hoc Sensor Networks

Zenon Chaczko, Christopher Chiu, Shahrzad Aslanzadeh, and Toby Dune

***Abstract*—Architects of ad-hoc wireless Sensor-Actor Networks (SANETS) face various problems and challenges. The main limitations relate to aspects such as the number of sensor nodes involved, low bandwidth, management of resources and issues related to energy management. In order for these networks to be functionally proficient, the underlying software system must be able to effectively handle unreliable and dynamic distributed communication, power constraints of wireless devices, failure of hardware devices in hostile environments and the remote allocation of distributed processing tasks throughout the wireless network. The solution must be solved in a highly scalable manner. This paper provides the requirements analysis and presents the design of a software system middleware that provides a scalable solution for ad-hoc sensor network infrastructure made of both stationary and mobile sensors and actuators.**

***Keywords*—Sensor-Actor Networks (SANETS), Ad-hoc Wireless Sensor Networks (WSNS), middleware, distributed software services.**

I. Introduction and Background

THE paper examines a sensor network made up of multiple individual devices that function as a single survivable entity. The devices must cooperate with each other to provide processing for a single purpose. Managing the failure of devices and connection links on the network is a key concern to ensure survival of the key network processing tasks, as individual devices can fail. With a distributed network of portable, wireless sensing and actuating devices to be commanded, scattered throughout the area of interest there is an ability to gain an awareness of the situation not previously possible. Information is power; with more information at hand about a particular situation, more informed decisions can be made with minimum manual intervention. Improved techniques to obtain information and manage hardware devices in an environment, in the form of sensory data and devices/resource controls, bring about changes in the way users perceive and interact with the environment. Greater informational gathering capabilities and more flexible resource control can provide a variety of benefits to a vast range of applications such as:

- Monitoring for research in various application domains such as habitat monitoring and control for bio-diversity and bio-complexity studies;
- Military situational awareness, toxin and radiation detection, monitoring and possible neutralization of hostile movements;
- Management of remote network infrastructure;
- Security monitoring and active protection of assets; and
- Management and monitoring of stress/seismic activity effects in civil infrastructures (i.e. bridges, roads, buildings).

The Distributed SANET (DSAN) software infrastructure (middleware) solution presented in this paper is intended to provide a platform that addresses the challenges involved in realizing an ad-hoc wireless Sensor-Actor Network (SANET). The DSAN is to be used in the development of real-world sensor and actuator networks and the trialing of new research concepts in the SANET field. Further developments in sensor networks can utilize the functionality provided in the middleware solution, allowing researchers to be more focused at their specific problem area. The main aim of this project is to reduce the development time of future research from being exhausted by redesigning or redeveloping the underlying network management functions that are provided by the presented middleware solution.

II. SANET Challenges

SANETS or Wireless Sensor-Actuator Networks (WSANS), as for all ad-hoc based networks (Fig. 1), present some serious challenges. With WSANS made of many disparate nodes, managing the information that needs to be disseminated across the entire network is critical. In this section, an investigation is made about the important challenges associated with WSANS that use ZigBee-based sensor nodes (motes) in particular.

a) **Range:** The maximum throughput for ZigBee is 250 kilobytes per second. With ZigBee wireless technology,

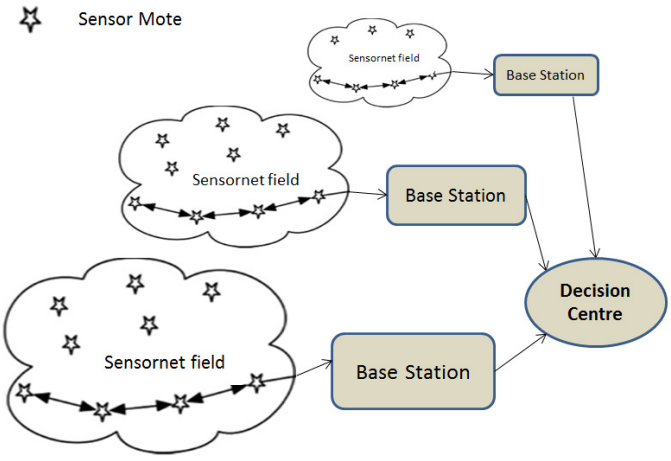

Fig. 1. Wireless Sensor-Actor Network mesh networking.

Z. Chaczko, C. Chiu, S. Aslanzadeh, and T. Dune are with the Faculty of Engineering & IT, University of Technology, Sydney, Australia (e-mails: zenon.chaczko@uts.edu.au, christopher.chiu@uts.edu.au, shahrzad.aslanzadeh@student.it.uts.edu.au, toby.dune@alumni.uts.edu.au).

packet-based networking is used and instead of using access points, gateways are applied which can provide connectivity from distributed nodes back to the coordinator. The maximum signal range for ZigBee is 1,000 meters.

b) **Power:** Using wireless technology for devices such as mobile phones and laptops, the battery lifetime will last for about 1 to 2 days; however, ZigBee is an efficient wireless technology that can be used in energy-constrained environments as it has a battery lifetime of 3 to 5 years.

c) **Cost:** Deployment and maintenance of the nodes is becoming inexpensive over time; varying on the underlying technology.

III. Middleware for SANET Technology

Traditionally, component middleware systems [1], [2] is a type of middleware that enables reusable service elements [3], [4] to be composed, configured, adapted, tested, integrated and installed to build software applications reliably and inexpensively, while adhering to requirements of distributed shared memory across disparate environments. Data space concerns can be addressed through Tuple-Space implementations as supported in modern component based [2] software-system middleware, following the multi-layer concepts of a core entity of representing the structures and interconnections between internal entities. This provides users with a specific set of capabilities (Fig. 2) listed below:

- **Connector Facilities within Components:** This includes Remote Procedural Calls (RPC), Remote Method Invocation (RMI) or message passing mechanisms;
- **Horizontal Models of Infrastructure Services:** Request brokers or publish-subscribe mechanisms between components within the same platform; and
- **Vertical Models of Domain Paradigms:** Common semantics and context awareness, and high-level services spanning from transaction and lease support, to multilayer security and privacy for multiple platforms.

Recent advancements in the miniaturisation of sensing devices including Micro-electromechanical systems (MEMS), embedded processors such as System-on-a-Chip (SoC) and wireless communications provide the hardware capability necessary to control devices embedded in the environment, collect environmental data and report it. With the availability of the hardware and various information processing and management systems, including commercial of-the-shelf (COTS) technologies, the construction of effective SANET solutions becomes feasible. Products such as Sun's Remote Method Invocation (RMI), Microsoft's .NET Remoting and Object Management Group's Common Object Request Broker (CORBA) have dramatically matured and become de-facto standards in the ICT industry. At present, these solutions are being used to reduce the Software Development Life Cycle and improve the effectiveness of building systems by reducing costs (time, work efforts and resources) mostly in business domains. Whilst commercial middleware solutions have traditionally been used in business, including enterprise management resource planning, stock control and asset management systems, e-commerce reservation systems and many other applications [5]; the technology can be transformed for distributed wireless networking solutions.

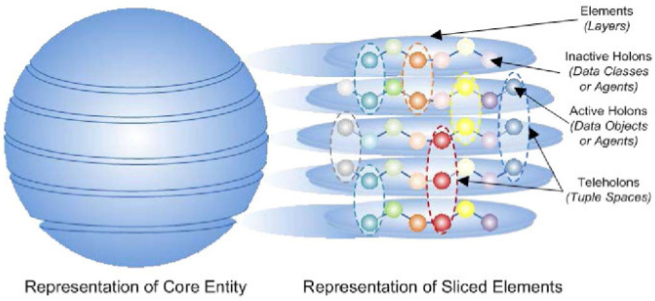

Fig. 2. SANET component architecture paradigm.

A distributed middleware for SANET systems can be built on evolvable, autonomic [6] and ad-hoc networks with actuation and control. This encapsulates events monitoring and control processes, operations, networks and hardware systems in civil and environmental engineering, computing and telecommunications, medicine, defense, manufacturing and infrastructure industries. SANET applications possess distinct characteristics relating to its mission critical aspects and time constraints. Time criticality and strict deadlines are essential, as the correct data response that is delivered beyond a given threshold can result in unpredictable or catastrophic consequences. Therefore, SANET middleware models need to meet stringent Quality of Service (QoS) qualitative requirements such as scalability, robustness, usability, security, efficiency, latency, privacy and trust [3], [4], [7]–[10].

For all application domains, the ultimate goal of infrastructure oriented software systems such as middleware is to support the process of software intensive system development by facilitating integration of components and protecting engineers from inherent and accidental complexities related to heterogeneous computing environments, management of resources, security and fault tolerance. The important issue for component middleware systems is being able to alleviate the compositional complexity and management of distributed SANET systems. Reducing the Software Development Life Cycle (SDLC) shortens the time-to-market delivery that is essential in modern engineering industry. As the majority of developer roles are to assemble distributed networked systems by selecting a combination of custom made components and compatible COTS frameworks [3], [5], [11], the process of selection is an important focus of this research. The construction of an effective system requires components to possess compatible application programming interfaces, semantics, context and protocols which make the analytical process of selection and development of a compatible set of software components a challenging task. Problems are exacerbated by the availability of various vendor-driven strategies for configuring and deploying the underlying software middleware to leverage dedicated hardware and software features.

IV. SANET CHALLENGES

"The sheer number of sensor nodes and the dynamics of their operating environments (e.g. limited battery power and hostile physical environment) pose unique challenges in the design of sensor networks and their applications" [1].

With the introduction of this technology and emergence of wireless sensor networks, a new set of technical challenges arise [7], [12]. These challenges form the basis of the middleware design.

A. Scalability of System and Communications

There is a need to manage the complexity of dealing with an immense number of sensors and a large volume of information contained within sensor and actuator networks so *"...existing distributed system scaling techniques are not directly applicable given the extreme conditions under which our target systems must operate"* [13]. The vast number of devices on these networks prevents the ability to manually configure and repair devices individually within the system; thus the number of electronic devices also increases probability of failure. The software must automatically configure devices and robustly handle failed devices.

B. Dynamic and Hostile Environment

The devices that make up the sensor network will be deployed to monitor hostile environments. Devices in the sensor network have a high coupling with the physical environment that they are deployed in. The dynamics of such an environment poses complex design challenges regarding as to how to manage the changing availability of resources and communication links within a large network. The sensor network must respond robustly to the dynamic environment in which it is situated.

C. Power Utilisation

As quoted from Zhong, *"Power consumption is crucial to wireless sensor network applications"* [14]. The lifetime of a sensor network is a function of energy consumption [15]. To improve overall network life, avoidance is necessary in key parts of the network from being over-utilized and drained quickly. Methods for distributing the processing and communication tasks evenly over the network are needed; the SANET system must be power aware.

D. Processing Resources

Embedded sensor devices throughout the network will have limited memory and processing resources. All network management must conform to the limitations of this target hardware [12].

E. Diverse Range of Applications and Uses

The many uses of the SANET network and continuing research and development in the SANET field compels the system to be expandable and maintainable [7]. To be able to utilize this emerging technology effectively and efficiently, smart systems are needed to manage the issue inherent to such networks. The software has to be able to manage the vast number of small autonomous devices in a way that allows for the effective combination of all of available resources. All devices must be able to interact and work together to support a common goal. The middleware that can solve these problems will facilitate the creation of new sensor network systems easily. A sensornet specific middleware will streamline the development of customized SANET sensors for the client's specific monitoring needs. There is a shortage of middleware solutions that are able to adequately handle the range of the domain concerns and constraints that can be encountered within the SANET context.

V. ENGINEERING DSAN MIDDLEWARE

Infrastructure-system software such as middleware is required to provide a set of services designed specifically to manage the complexities that exist within the field of distributed sensor and actuator networks [3], [11], [15]. This covers a suite of functional and non-functional requirements that need to be addressed when modeling the middleware and designing software components specifically to support the operations in distributed SANETS [5], [16]. The DSAN middleware aims to provide a base for a number of communication and management services to reliably enable distributed environmental monitoring and control, actuator management, in-situ (i.e. ONE-WIRE and CANBUS) and wireless processing (i.e. Bluetooth IEEE 802.15.1 and ZigBee IEEE 802.15.4), and reporting to a centralized datacenter (Fig. 3).

A. Functional Requirements

The set of high-level functional requirements that the DSAN middleware must address for operability concerns include the following aspects:

- The communication interfaces with embedded sensor and actuator devices;
- Automated health monitoring of available resources within the SANET system;
- Automated configuration management for the SANET network;
- Lightweight communication infrastructure for distributing events and configuration throughout the middleware system;
- Persistent storage for recording of notifications, alarms and alerts as well as and configuration reports; and
- A user interfaces for viewing system state, logging events and warnings as well as configuring the processing tasks on the devices within the sensor network.

The TINI (*Tiny Inter-Net Interface*) device [10] has been chosen as it provides the reference implementation of the distributed embedded sensor devices. The standard Java Mobile

Runtime Environment, with the addition of JINI [17] services was selected to provide a platform for the scalable centralized services required.

B. Non-Functional Requirements

In order to solve a very diverse set of problems characterized within DSANS, the middleware must possess a range of architectural qualities. Therefore, when building middleware for SANETS, among the most important non-functional requirements are considered as follows:

- **Lightweight Implementation:** The code solution must efficiently apply and use the resources available on the small embedded computing devices used in SANET networks;
- **Robustness:** The middleware must gracefully handle failure of wireless and in-situ components registered in the network;
- **Scalability:** The middleware for sensor networks must be inherently scalable, so it is able to support the massive number of devices contained within the SANET system. Additionally, the distribution and scale of the SANET environment means potential geographic spread among various infrastructure interconnections and software interface, this is resolved with Remote Method Invocation (RMI);
- **Adaptability:** The middleware must gracefully adapt to the continually changing health status of the SANET network, while also maintaining the user's processing needs; and
- **Flexibility:** A diverse range of situations the middleware can be applied to, along with the immaturity of the technology means the middleware has to be very flexible. To cater for new research, the system must allow for components implementing specific functionalities to be updated or replaced without impacting on the rest of the middleware.

VI. Middleware Services

Service repositories have been designed to provide an initial set of solutions for each of the diverse challenges involved in the SANET application. The services provided by the middleware are divided into a layered model (Fig. 4). The middleware services must be implemented over different hardware architectures; with a set of distributed lightweight services interacting with highly scalable central services. All services are integrated via the integration bus layer.

A. Integration Layer

The integration layer forms a solid base for all other services to be built upon. This layer provides a flexible set of communication services for lightweight and device-independent communication. Interfaces are provided to handle the low-level functions of the operating system in a hardware-independent manner.

1) Distributed Communication Service

The object of the Distributed Communication service is to provide an interface that will enable the distributed nodes throughout the system to communicate. The distributed communication service has been designed to provide support for the following features:

- Minimal resource utilization to run on embedded devices;
- Platform/media layer independent addressing mechanism;
- Transmission media independence;
- Robust communication error handling;
- Synchronous Communication (Lightweight RMI) for device interaction, device control, and agent activation;
- Asynchronous Communication for availability 'heart-beating' and the sensor interface; and
- Mobile Code for Agent Distribution.

The design of Distributed Communication Services is based on lightweight client-to-server communication. These services are designed to be independent of the communication media

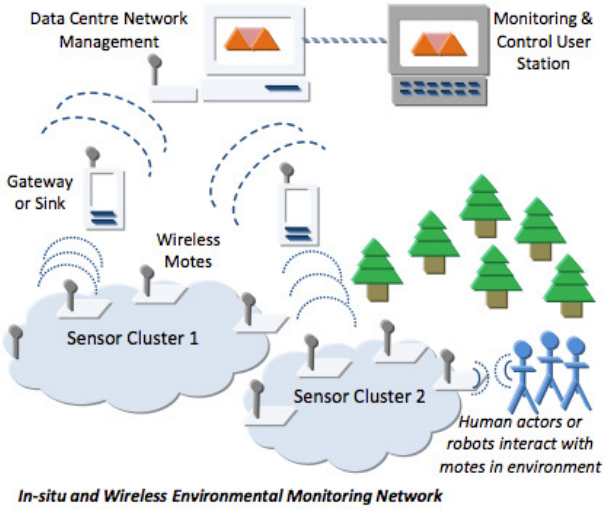

Fig. 3. Overview of DSAN middleware.

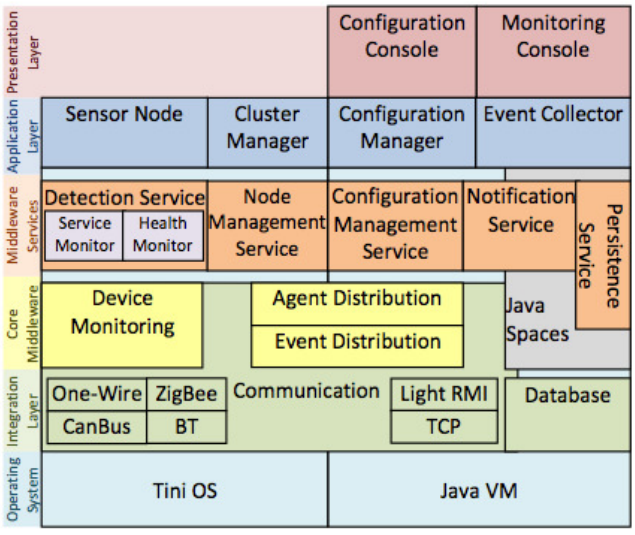

Fig. 4. Architectural model of DSAN middleware.

being used. Generic interfaces are provided for establishing and tearing down of connections regardless of the media-type used.

B. Core Middleware Layer

The core middleware layer services provide high level interfaces to the integration layer. The services provided cater for higher-level data distribution and routing functionality.

1) Agent Distribution Service

The objective of the Agent Distribution Service is to enable the coordination of the distributed processing required by the sensor network. This is provided by the distribution and execution of mobile agents throughout the system. This service provides the underlying mechanisms for adaptability of the SANET system. Work, in the context of this system, is defined as a specific task for a remote agent to perform; it includes some agent configuration parameters and the required sensors that should be monitored.

2) Event Distribution Service

The objective of the Event Distribution Service is to provide a standard reliable system for events to be generated by devices. Events are routed throughout the network to reach their destination. In order to reduce the amount of data sent through the network, any device is capable of intercepting events and providing local processing and actions, instead of forwarding them to the central event collector.

3) Device Monitoring Service

The Device Monitoring service provides for data input to the system. A framework for implementing custom drivers is used to provide support for a range of monitoring scenarios. Lightweight dynamic driver loading and unloading mechanisms are provided to enable run-time reconfiguration of data collection, with minimal processing overhead.

C. Middleware Services Layer

The middleware services layer provides high-level functions for managing the distributed SANET. Management of the networks' distributed processing includes such components as:

1) Configuration Management Service

The Configuration Management Service provides a centralized configurable model of the processing needs within the sensor network. The Configuration Manager is an automated system responsible for ensuring the optimum level of service in utilizing the available resources. The Configuration Manager contains models of both the target environment (environmental model) and the Sensor Network (system model). The environmental processing model determines what physical properties of the environment should be monitored and how this should be done. This includes a set of work distribution rules. The system model is a record of the current state of the sensor/actuator nodes within the system: which sensor nodes are ready to check the environment and which sensor/actuator devices are available.

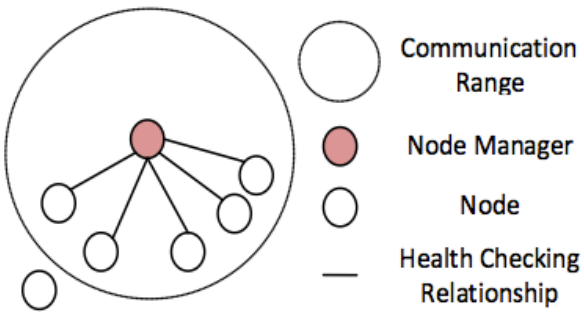

Fig. 5. Centralized health checking in DSAN middleware.

The system model is dynamically updated when nodes enter and leave the system. The system model is automatically updated to reflect the current state of the sensor network. The model of system state is based upon received system reports from the event collector indicating that resources are entering the system or are no longer available. User interaction with the models contained within the Manager is via a Configuration Console which allows users to view the individual node's activity, as well as how many nodes are active and what environmental properties they are actually monitoring.

2) Node Management Service

The node manager is responsible for managing a group of nodes within its local coverage area. It provides in-situ management of network resources. The Node Management Service works in cooperation with the Configuration Management Service to provide for agent distribution. A cache of device work allocations, within the node manager, is used to manage the distributed processing requirements of nearby processing nodes.

This reduces the communication load to the remote configuration management. Processing nodes use the Node Management Service to announce themselves, to publish their sense collection and processing capabilities and receive processing tasks. The Node Management Service is also responsible for tracking the health of processing nodes within its coverage area. Health checking agents are distributed to other nodes within the system to enable nearby peer nodes to monitor each other's health. The remote configurability of the System

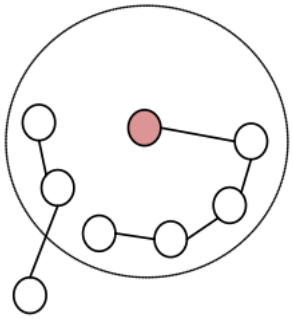

Fig. 6. Decentralized health checking in DSAN middleware.

Health Service allows for health checking to be decentralized and distributed throughout the network. Greater resource utilization can be achieved if the nodes manage themselves, provided that all health-checking tasks and power drain is not somehow centered on the node manager (cluster-head) which may attribute to possible premature failure. As illustrated in Fig. 5, the node manager needs to continually talk to all nodes within its maximum communication range. Distributing the task of health checking to individual nodes within the network ensures power utilization will be more evenly drained.

Algorithms can be validated to determine node proximities and configure nodes accordingly in order to check the health of the closest devices. This would reduce the power output required to perform the same amount of health checking. Figure 6 gives an example of how the same node configuration as discussed above could be more efficiently health-monitored. The Node Manager needs only to continually health check one node. The nodes perform health checking on neighboring nodes; communication distance is potentially closer than that of the distance to the node manager. Note also the ability for the node manager to manage the health of nodes outside its direct communication range, as routing would need to be performed to do the initial setup for agent and event distribution.

3) System Health Monitoring Service

The System Health Monitoring Service enables sensor/actuators nodes to check the health status of their nearby peers. The service consists of a set of health checking agents that are remotely managed to provide optimal health checking coverage and reporting. System health information is used for automated management of how the available processing resources are utilized. The sensor network will adapt to the reduction in available resources over time, allowing the network to degrade gracefully. The current implementation of the Service uses status heart-beating to detect device abnormalities, while alternate methods, such as leasing, can be used as replacement or in combination with this method.

4) Detection Service

The Detection Service enables the distributed processing capability for the business operations of the SANET. The service is remotely managed to enable dynamic re-configuration for optimum processing in accordance with specified processing allocations. Processing and data aggregation algorithms are implemented as agents within the service. These agents utilize collected data input from the Device Monitoring Service and provide output in the form of events via the Event Distribution Service.

5) Persistence Service

The Persistence Service provides a central store for logging events and maintaining active models of the network. The service is built upon the Java Space service of JINI. In this release of the DSAN middleware system, the reference implementation of Java Spaces as provided by JINI is used. The DSAN middleware relies on the expandability of the Java Spaces model, future releases of the DSAN software may have to use a more scalable and capable implementation of the Java Spaces service.

VII. Case Study

Location tracking has long been an area of interest where research into SANETS and Wireless Sensor Actuator Networks is concerned. It is useful for the tracking and location of objects; such as vehicles on a road or people. For this case study, it is important to create a basic prototype for a tracking system that involves both sensors and mobile robots. The specific objective is to design a WSAN solution that is capable of detecting heat within a specified temperature range in order to determine whether a fire is likely to be in or nearby an area. The proposed hardware architecture involves a temperature sensor connected to a Texas Instruments CC2530ZNP board attached to a battery board which broadcasts the local temperature over IEEE 8.2.15.4 (ZigBee) Protocol, to a ZigBee network of routers (other CC2530ZNP boards) which include multiple coordinator nodes and the Stellaris Evalbot (Fig. 9) via the Texas Instruments CC2530EM board interfaced to the Evalbot's EM socket.

The Evalbot will obtain the local temperature via reading the temperature sensor on its Analog-to-Digital Converter (ADC) and use this temperature as well as the temperature broadcast by nearby nodes and the coordinator node(s) to calculate areas of high temperature. Then, the onboard motors will be powered to navigate towards the high temperature areas to investigate whether a fire is present, and then broadcast an alert over router nodes to a coordinator node, ultimately reaching a main computer. The model has been depicted in Fig. 7. Implementing a full user-interface is beyond the scope of this case study and not required for a proof-of-concept; all that is required is a conceptual design. The basic idea is to have the data received by the coordinator node connected to the PC via USB (virtual COM port) collated by time received and location and broadcast over TCP/IP to a centralized system.

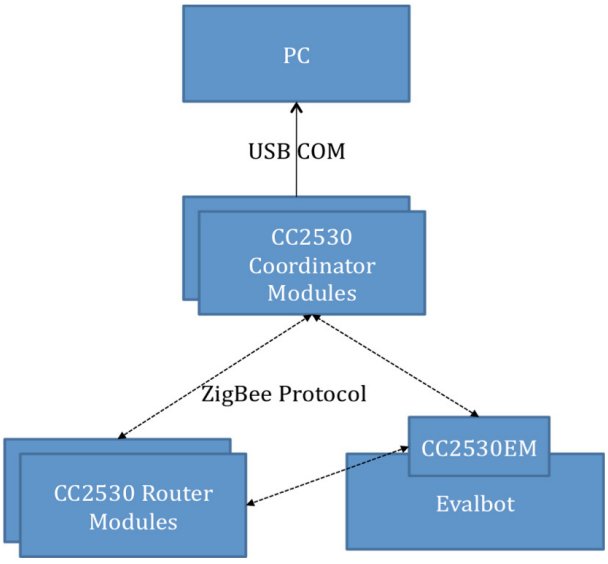

Fig. 7. Model diagram of coordinator and router nodes.

Fig. 8. TI CC2530ZNP ZigBee sensor motes.

This system will create a map of locations and show temperatures at these locations, displaying a warning if the temperature is greater than a set threshold. The important features of this software will be to be able to set the location of the coordinator nodes manually so the location of router nodes can be calculated. This is along with the ability to display the temperature and warning message, and to send out alerts via e-mail, SMS or another medium to emergency services.

A. Hardware Components

1) Texas Instruments CC2530ZNP Mini-Kit

The CC2530ZNP is a board designed to introduce computer engineers to the ZigBee Network Processor (Fig. 8) without a significant amount of hardware and software setup work required. This makes it suitable for developing a quick prototype to test the design while using the protocol and on-board chip (CC2530) to be deployed in a final product.

2) Texas Instruments LM3S9B92 Stellaris Evalbot

The Stellaris Evalbot is a useful prototyping and evaluation tool and has been chosen to simulate an automated investigation and response vehicle. The Evalbot has an EM socket that can be used for attaching an RF module; in this case this is ZigBee (Fig. 9). DC motors to provide movement and bumper-switch sensors allow for obstacle detection and avoidance.

3) Texas Instruments CC2530EM Module

In the Stellaris Evalbot (robot) that is used for the prototype of the mobile sensor mote, the main method of wireless communication with other sensor and actuator devices is

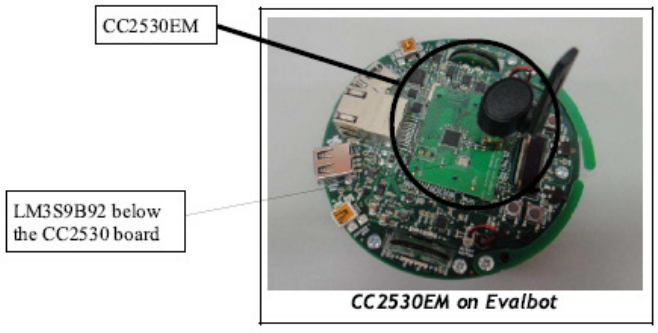

Fig. 9. TI Stellaris Evalbot with CC2530EM as a mobile node.

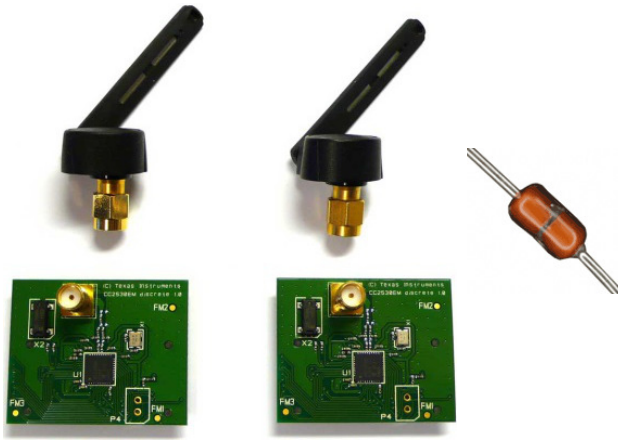

Fig. 10. a) TI CC2530EM wireless module (left), b) Honeywell 135-102DAG-J01 thermostat (right).

through the EM socket on the Evalbot that allows the installation of an onboard RF unit, the CC2530EMK module (Fig. 9 and Fig. 10a). The CC2530EMK module uses the ZigBee Network Protocol to communicate sensor data and receive control to/from the Base Station/decision center. There are significant programming benefits in terms of firmware development using the CC2530EMK module that use the same on-board chips as the sensor nodes for the rest of the sensor network.

4) Honeywell 135-102DAG-101 Temperature Sensor

The Honeywell 135-102DAG-J01 glass encapsulated NTC thermostat is used (Fig. 10b). Although any low-power temperature-measuring sensor will suffice, this one was chosen due to its low power, affordability and high availability.

B. Implementation

The initial design proved to be ambitious to develop in the timeframe; so with the resources available for this project, some concessions were made. As an insufficient number of nodes were constructed to allow for location sensing (upwards of five nodes of a known location would be ideal), so a practical solution was implemented with a minimum of two nodes. The CC2530ZNP Mini-kit was used to test communication between modules and to establish whether the ZigBee protocol was effective in this case. There were also issues of communication with the CC2530EM and the Evalbot. It was decided that this would be acceptable due to the fact that the 2530EM uses the same chip as the CC2530ZNP Mini-kit boards, so the same code should work on both. Hence, it was a matter of hardware interfacing between Evalbot and EM socket. The final system developed ended up as follows:

- CC2530ZNP modules communicate with each other sending the light sensor value, along with the temperature sensor via the designated coordinator and router nodes;
- The CC2530ZNP coordinator collates information for post-processing on middleware server; and
- The Evalbot reads the temperature and light sensor values from ADC and uses the following simple algorithm:

```
Do {
    Read ADC Temp/Value
    Move random distance in random direction
    Read ADC Temp/Value
    If (New Temp > Old Temp/Val)
        Continue Move
    Else
        Move back to previous location
} While (No Error)
```

The main problem encountered with the progress of the initial design was the technical expertise in the development team and the time required for studying the technology. With the experience gained from embedded microprocessor programming along with knowledge into wireless protocol design, the concept of the system was essentially realized. Further work into swarming algorithms with multiple nodes will be investigated for collaborative swarm intelligence in SANETS, along with the efficient optimization of peer-to-peer communication between wireless nodes, gateways and the coordinator.

VIII. Conclusion

The DSAN middleware environment achieves the goal of enabling the end-user to interact effectively in SANET contexts. Further outcomes in terms of the infrastructure design and implementation have established the main outcomes:

- **Project Management:** The design and implementation of the SANET middleware system was used in conjunction with the TRAC e-wiki tool, facilitating the practice of formal software engineering standards.
- **Configuration Management:** The development of the SANET middleware system was achieved with Subversion Configuration Management to commit code changes and integration branches to the main code trunk.

The domain of wireless sensor and actuator networks is at its early stages, with much development work being done. Many aspects of the domain are currently in early research and development stages. This means that new developments are continually being made and are open to being incorporated into a middleware solution. The DSAN is designed specifically with the future of SANET applications in mind. It is the intention that future research in the sensor and actuator network field is able to build upon and extend the DSAN. The DSAN layered architecture promotes the use of strong encapsulation of services with concise interfaces. Sound design principles enable expandability of the middleware by future investigation; each service designed within the middleware provides functions required in a different area of research. With developments any in area, a service in the middleware can be upgraded or replaced, leaving the rest untouched. Researchers need only look at the specific set of problems that relate directly to their field and let the middleware take care of the rest of the concerns. Wireless communication can consume a lot of power, so new developments in power-aware algorithms and design principles are needed to maximize the utilisation of energy throughout a network as a whole. These new developments can be built into DSAN services to enable them to be tested in real-world scenarios.

References

[1] C. Shen and C. Srisathapornphat, "Sensor Information Networking Architecture and Applications, University of Delaware," in *IEEE Personal Communications*, August 2001, p. 52.

[2] C. Szyperski, "Emerging component software technologies – A Strategic Comparison," *Software Concepts and Tools*, vol. 19, no. 1, pp. 2–10, 1998.

[3] F. Golatowski, J. Blumenthal, M. Handy, M. Haase, H. Burchardt, and D. Timmermann, "Service-Oriented Software Architecture for Sensor Networks," in *Proceedings of International Workshop on Mobile Computing (IMC'03)*, Rockstock, Germany, June 2003, pp. 93–98.

[4] A. Rezgui and M. Eltoweissy, "Service-Oriented Sensor-Actuator Networks," *IEEE Communications Magazine*, vol. 45, no. 12, pp. 92–100, 2007.

[5] X. Chu and R. Buyya, *Sensor Network and Configuration: Fundamentals, Standards, Platforms, and Applications*. Germany: Springer-Verlag, January 2007, ch. Service Oriented Sensor Web, pp. 51–74.

[6] A. G. Ganek and T. A. Corbi, "The dawning of the Autonomic Computing Era," *IBM Systems Journal*, vol. 42, no. 1, pp. 5–18, 2003.

[7] I. F. Akyildiz and I. H. Kasimoglu, "Wireless Sensor and Actor Networks: Research Challenges," *Ad-hoc Nets*, vol. 2, no. 4, pp. 351–367, 2004.

[8] Z. Chaczko and R. Klempous, "Anticipatory Biomimetic Middleware," in *Journal of American Institute of Physics (AIP), Casys 2009*, Liege, Belgium, August 2009.

[9] Z. Chaczko, R. Kohli, R. Klempous, and J. Nikodem, "Middleware Integration Model for Smart Hospital System Using the Open Group Architecture Framework (TOGAF)," in *14th International Conference On Intelligent Engineering Systems, INES 2010*, Las Palmas of Gran Canaria, Spain, May 5–7 2010.

[10] TINI Website and Development, (2010) Tiny InterNet Interface, http://www.ibutton.com/TINI/index.html, last visited June 2010.

[11] E. C. H. Ngai, M. R. Lyu, and J. Liu, "A Real-Time Communication Framework for Wireless Sensor-Actuator Networks," in *Proceedings of IEEE Aerospace Conference, Big Sky*, Montana, U.S.A., March 2006.

[12] C. Y. Chong and S. P. Kumar, "Sensor Networks: Evolution, Opportunities, and Challenges," *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1247–1256, 2003.

[13] D. Estrin, *Center for Embedded Network Sensing*. Los Angeles: Computer Science Department, University of California, 2001.

[14] C. Zhong, *Pico Radios: What does it take to design a link between them?* Department of EECS, UC Berkeley, 2004.

[15] R. Vidhyapriya and P. T. Vanathi, "Conserving Energy in Wireless Sensor Networks," *IEEE Potentials*, vol. 26, no. 5, pp. 37–42, 2007.

[16] F. Xia, Y. C. Tian, Y. J. Li, and Y. X. Sun, "Wireless Sensor/Actuator Network Design for Mobile Control Applications," *Sensors*, vol. 7, no. 10, pp. 2157–2173, 2007.

[17] JINI Website, (2010), http://www.jini.org/, last visited May 2010.

[18] Z. Chaczko and G. Resconi, "Organising Software Infrastructures: EgoMorphic BIM Model, Conscious Brain and Education – Mind and Living Systems, Risk Management, Economical Systems, and Social Models, Applied Mathematics, Programming, and Biomimetic Tools," in *Partial Proceedings of the Eighth International Conference CASYS'07 on Computing Anticipatory Systems*, D. M. Dubois, Ed., Liege, Belgium, August 6–11 2008, application of Biomimetic Design Methods in Infrastructure Systems. Chaos, Lie?ge, Belgium, Vol.21. pp.372–385.

[19] I. Gorton and S. Motwani, "Issues in co-operative software engineering using globally distributed teams," *Information and Software Technology, Elsevier Science*, vol. 38, pp. 647–655, 1996.

[20] E. C. H. Ngai, Y. Zhou, M. R. Lyu, and J. Liu, "Reliable Reporting of Delay-Sensitive Events in Wireless Sensor-Actuator Networks," in *Proceedings of the 3rd IEEE International Conf.erence on Mobile Ad-Hoc and Sensor Systems (MASS'06)*, Vancouver, Canada, October 2006.

[21] F. Xia, W. H. Zhao, Y. X. Sun, and Y. C. Tian, "Fuzzy Logic Control Based QoS Management in Wireless Sensor/Actuator Networks," *Sensors*, vol. 7, no. 12, pp. 3179–3191, 2007.