Embedded Networked Sensing – EmNetS

Panneer Muthukumaran[1], Rostislav Spinar[1], Ken Murray[1], Dirk Pesch[1], Zheng Liu[2], Weiping Song[2], Duong N. B. Ta[2], Cormac J. Sreenan[2]

[1] Centre for Adaptive Wireless System,
Cork Institute of Technology, Ireland
{panneer.muthukumran, rostislav.spinar, ken.murray, dirk.pesch}@cit.ie

[2] Mobile and Internet Systems Laboratory,
University College Cork, Ireland
{zl3, wps2, taduong, cjs}@cs.ucc.ie

Abstract

This paper presents the work under investigation within the Embedded Networked Sensing (EmNetS) project funded by Enterprise Ireland under the WISen industry/academia consortium. The project addresses four main research areas within the embedded networked sensing space, namely, protocol stack development, middleware development, sensor network management and live test bed implementation. This paper will provide an overview of the research questions being addressed within EmNetS, the motivation for the work and the proposed solutions under development.

Keywords: Wireless Sensor Networking, Protocol Stacks, Testbed, Middleware, Network Management.

1 Introduction

The WISen Industry/Academia Consortium has identified wireless sensor networks as a medium term target for Irish Industry and a range of application domains focusing on Utilities and Resource Management in the first instance. Market forecasts indicate that the global wireless sensor network market could be worth $8.2Bn by 2010. The market is currently US-led but there is growing demand in Europe in particular for applications in the utilities, health-care, and environmental monitoring domains. The EmNetS project aims to advance research in Ireland in the areas of wireless sensor networking software and live test bed implementation [1]. The project addresses the need to network a range of low power heterogeneous sensor devices to be used in the utilities and responsive building environments. Firstly the project aims at developing a network protocol stack for individual sensor nodes and for cluster controller/base station type nodes that need to inter-work with other networking technologies such as local area networks and mobile networks, e.g. GSM. The protocol stack will be based on the industrial adopted IEEE 802.15.4/Zigbee stack for low power sensing [2]. It is the aim of the protocol stack development to overcome some of the limitations of IEEE 802.15.4/Zigbee stack such as scalability, energy efficiency in large scale mesh networks, dynamic address assignment and energy efficient routing. It is envisaged within the environmental monitoring and responsive building environments the number of sensing devices can run to the order of hundreds to thousands. The current sensor networking standards are unable to support such large scale energy efficient deployments. A further part of the software infrastructure consists of a simple middleware layer that provides application programmers interfaces to allow rapid development of applications. The development of such a software platform will ease product development of sensor network applications. Sensor networks that provide a mission critical role require remote management facilities in order to monitor the correct operation of the network, query the status of individual nodes, and provide means to upload software updates. A remote management system is current being developed that will integrate with the protocol stack within a live system deployment. This test bed deployment

will provide for test and validation of networking protocols, as well as scalability and internetworking trials within the EmNetS team and the sensor network research community at a National level. This paper provides a technical overview of the current state of research and development activity within the EmNetS team. The challenges and proposed solutions under investigation will be presented in each of the aforementioned areas of research.

2 Protocol Stack Development

Wireless sensing within the responsive building environment has been highlighted as the target application domain for the EmNetS project. In such sensing environments the area of deployment can be relatively large, for example the control of HVAC systems in multi-story buildings based on user location/density. To facilitate such large scale deployments, the sensing devices must cooperate to efficiently route data from source to a destination data sink which can be many hundreds of meters apart and contain multiple intermediate nodes. Mesh networking topologies can provide high redundancy for failed data links, provide scalable network topologies and provide the dynamic selection of alternative routes for high priority traffic. Within a mesh topology, data can be routed to fulfil requirements of energy efficiency, throughput, and quality of service (QoS). The deployment of energy efficient mesh wireless sensor networks is therefore desirable in the provisioning of services over large sensor fields. To enable energy efficient data transmission over sensor networks requires the use of energy efficient protocol stacks. Energy efficient algorithms should be present at each layer in the stack, in particular the MAC and NWK layers and cross layer interaction used to optimise performance. Techniques in the literature employ the transmission of beacon packets between transmitter and receiver to facilitate low duty cycle, energy efficient channel access in which devices transmissions are coordinated. With this strategy, devices can sleep between the coordinated transmissions, which results in energy efficiency and prolonged network lifetimes. Beacon scheduling is an important mechanism in multi-hop mesh networks to enable multiple beacon enabled devices function whilst avoiding beacon and data transmission collisions.

2.1 Distributed Beacon Synchronisation

The IEEE 802.15.4 MAC standard for low duty cycle, low data rate devices is the most significant commercially adopted MAC protocol to date [2]. The standard however does not specify techniques by which the synchronisation of beacon packets is to be achieved to enable low duty cycle functionality. Furthermore, the standard specifies that to enable mesh topologies, the router devices within the network need to be line-powered and engage in idle listening. Recent proposals exist in the literature for low duty cycle MAC protocols are based on the channel polling, low power listening technique [3, 4]. These schemes however suffer from long and variable preambles at the transmitter side and are best suited for bit streaming transceiver chipsets. The latest trend is toward packetized radios in which the preambles are a fixed length such as the TI CC2420. A collision-free beacon scheduling algorithm for IEEE 802.15.4/Zigbee Cluster-Tree Networks is presented in [5]. The approach called Superframe Duration Scheduling (SDS) builds upon the requirement for beacon scheduling outlined in the Zigbee specification for Cluster-Tree multi-hop topologies. The SDS algorithm functions within the coordinator. Although centralised control reduces the computational overhead and information flow between distributed devices, it can result in excessively data flow of control traffic toward the coordinator, which results in devices close to the coordinator being excessively overloaded in relaying this data. A distributed approach may be more attractive. The IEEE 802.15.5 Task Group 5 is discussing the proposal for beaconing scheduling for mesh topologies [6]. The proposal involves making fundamental changes to the superframe structure on the MAC to provide a beacon-only timeslot in which beacons of neighbouring devices will be transmitted. This proposal however involves changing the MAC superframe structure, which affect the interoperability with the current MAC standard. In order to overcome the limitations outlined above, the EmNetS team propose a distributed beaconing scheduling strategy, in which the coordinator device does not participate in the beacon scheduling process. The proposed algorithm is depicted in Figure 1. Local decisions are made at each mesh router device based on information received during the beacon scan. In this way information is not required to be sent to the coordinator each time a new device requests a

beacon schedule time, hence reducing the control traffic overhead toward the coordinator. When a node initially starts, it associates with a beacon enabled device, based on for example, the strongest signal strength (network layer function). If the node is required to transmit beacons it must build a list of its neighbours and neighbours' neighbours. It does this by listening for its neighbour's beacons (obtain neighbour list) and records the beacon transmit time of each in a Beacon Schedule Table (BST). The device will then request the neighbours' neighbour list in the CAP of each neighbour [2]. The list will contain the beacon offset time of the two-hop neighbours relative to the one-hop neighbour sending the list. This data is also added to the BST. Upon completion of this step the node will have a complete list of its neighbours and neighbours' neighbours in absolute values (one-hop neighbours) and offset values (two-hop neighbours). The device can now determine its own schedule period. The new device shall remain scanning for beacons to calculate the transmission offset values between its scheduled beacon time and that of the received beacons and notify each neighbour of this offset. This may also facilitate the reception of any unheard beacons in the first beacon scan. The new device will at this stage have bi-directional non-interfering connectivity between it and all one-hop neighbours in the PAN.

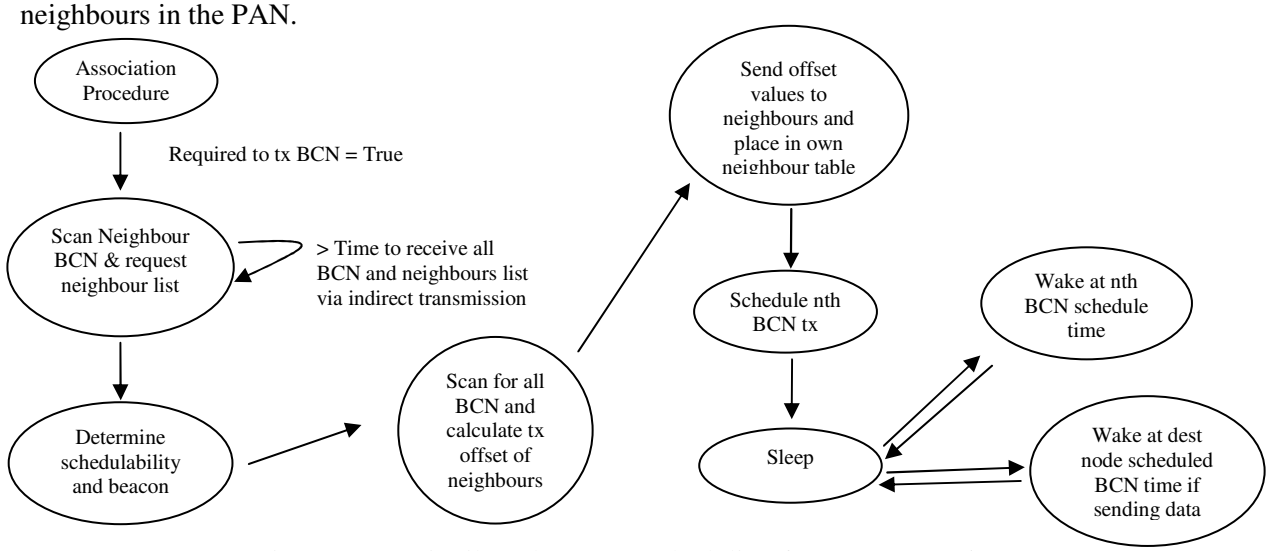

Figure 1 Distributed Beacon Scheduling for IEEE 802.15.4

2.2 Two level Zone based Routing

The communication scenarios in wireless sensor networks can be classified into few-to-many data dissemination, many-to-one tree based routing, and any-to-any routing topologies [7]. Many of the routing algorithms in wireless sensor networks are based on network-wide dissemination and the collection of data from the interested nodes. Tree based routing is used for forwarding data to a common destination at the tree root. These scenarios make the sink node or root a fixed node and there is no support for communication between any two independent devices. To implement dynamic backbone networking inside a wireless sensor network, any-to-any routing would be useful. However limited memory in wireless sensor networks makes it impossible to maintain routes to every node in the network. The Zigbee standard uses a variant of the AODV routing algorithm to route packets in which each node has to maintain a routing table with the entries of destination and next hop in that route. This method is not suitable for larger networks. For example when a node tries to send to multiple destination nodes, it requires multiple route entries along the same path. To combat the limited resources of sensing devices in terms memory and computational complexity, we have defined a framework for network routing in wireless sensor networks similar to the Zone Routing Protocol ZRP [8]. In this strategy we divide the network into clusters or zones. A hybrid of proactive and reactive routing is used for the intra-cluster level and a reactive approach is used at the inter-cluster level. Each node in a cluster always maintains routes to the destination cluster, rather than to maintain the entire route for a destination node. This framework works on the basis of "Think Global and Act Local". Each node maintains two types of routing tables to perform routing at the two levels. To perform inter-cluster routing, nodes on the current cluster do not care about the destination node, instead they try to route the packet to the next cluster which is en-route to the destination cluster. If the destination node belongs to the current cluster, it routes using the intra-cluster algorithm based on

AODV. The concept is illustrated in Figure 2. The nodes on the edge of the cluster that have neighbours in other clusters are called Gateway nodes. These gateway nodes forward the data packet to their neighbour clusters. Gateway nodes broadcast (proactively) their next cluster information to all nodes inside the cluster. This broadcast enables the nodes within a cluster to build routes to appropriate gateway nodes depending on the destination cluster. When a data packet is transmitted, it needs to be supplied with destination cluster address and node address. Nodes send the Inter-Cluster routing request to all gateway nodes, unless it knows which gateway has a route to the destination cluster. As the network evolves it is possible that nodes will learn which gateways have paths to the most recent destined clusters. They forward the route request packet to neighbouring clusters via neighbour gateway nodes. When this packet reaches a gateway in the destination cluster or a gateway that has knowledge of the remaining path, a route reply packet is sent back to the source node. The route reply packet reaches the source node with the next cluster or entire cluster list (optionally). Each gateway nodes along the path caches the cluster level route. This completes the inter-cluster routing procedure. The data packet can now be sent to the destination cluster. Inside the cluster, it finds the destination node using intra-cluster reactive routing. The simplified version of AODV may be used as a basis of the reactive routing algorithm.

3 Middleware Development

Emnets is developing a simple middleware platform to provide application developers interfaces to facilitate rapid development of applications in the utilities space; this platform will ease product development of wireless sensor network. The goals of the middleware are: (1) to develop services oriented towards rapid applications development; (2) to develop a composition tool for middleware synthesis; (3) to develop a resource-aware deployment tool for middleware mapping.

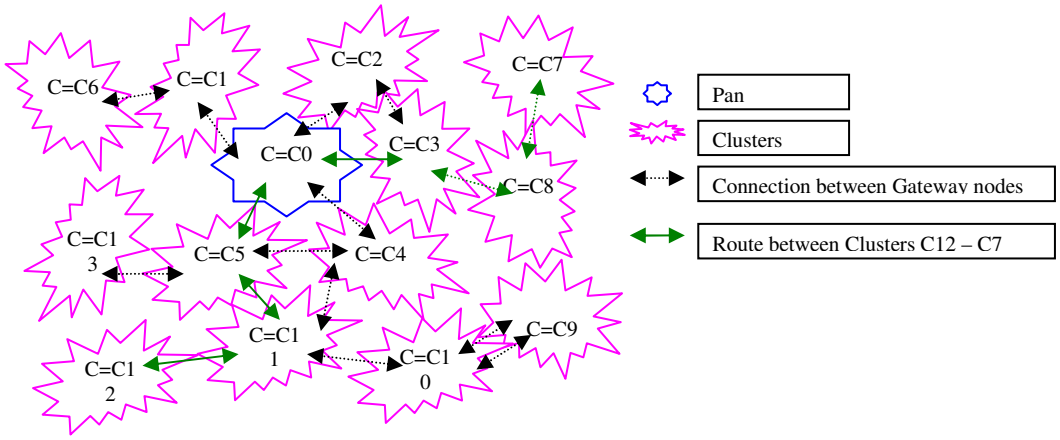

Figure 2 Concept of cluster based routing in wireless sensor networks

The middleware will employ a VM-based approach. There are two main advantages of employing a VM-based approach in the middleware system. Firstly, middleware services do not need to be rewritten for different platforms as they run transparently over the varied platforms. Secondly, a virtual machine provides a well-designed instruction set. This enables rapid prototyping of highly compact application binaries which may result in low energy overheads when they are distributed in the network [9]. As shown in Figure 3, the middleware system will be decomposed into two layers, virtual machine layer and services layer. Virtual machine layer resides on top of operating system and network stack, it utilises a virtual machine to provide APIs to abstract different hardware and/or system platforms. Services layer resides on top of virtual machine layer and consists of middleware services. All services provided by the middleware will be implemented in this layer. The middleware services can be tailored for each sensor node, which is based upon the facts that (1) services can be realised in different ways based upon hardware components, hardware resources, user requirements, optimisation criteria, etc. (2) services are required differently by applications running on top of it, the services running on any specific sensor node do not need to reflect the full requirement specification. The middleware services may include, just to name a few, service discovery, aggregation, localisation, synchronisation, adaptation, update, security. New services can be added into the system if they use

certain interfaces. There are a number of algorithms and/or models for different services in literature, thus, instead of developing new algorithms and new models for the services, the proposed middleware services will be realised using the ones in existence. Similarly, the virtual machine used in the middleware system will be based upon one of the existing virtual machines for wireless sensor network (e.g., Mate, Agilla, SensorWare, etc.), and further modification will be applied if necessary. The middleware system will be built by defining a set of components, dependencies, and the specific components which can be selected under a certain condition. A key design goal is to develop a tool capable of selecting components, capturing relationships among components and composing specific components. For the purposes of selection and composition, each component will have enough information as its attributes; such information may include functional properties, non-functional properties, required and/or provided guarantees upon qualities, etc. In order to compose a middleware system, the following information will be needed by the composition tool: platform description, middleware services, constraints, and quality criteria. The platform description specifies hardware components and their resources; middleware services specify the services which run on top of a particular device; constraints specify the resource constraints of each component; and quality criteria specify user's non-functional requirements which may include reliability, usability, performance to name a few. After reading the information, the composition tool builds a dependency diagram by satisfying the dependencies of a start component which can be any one of the components selected by the user. During this process, based upon resource constraints and quality criteria, the composer selects the most suitable components from a set of different possible components, and also produces additional components required and necessary glue code to hook all components together [9, 10]. Finally, the system will also provide a mechanism to map the middleware images into the entire network. Figure 4 depicts the processes of composition and deployment.

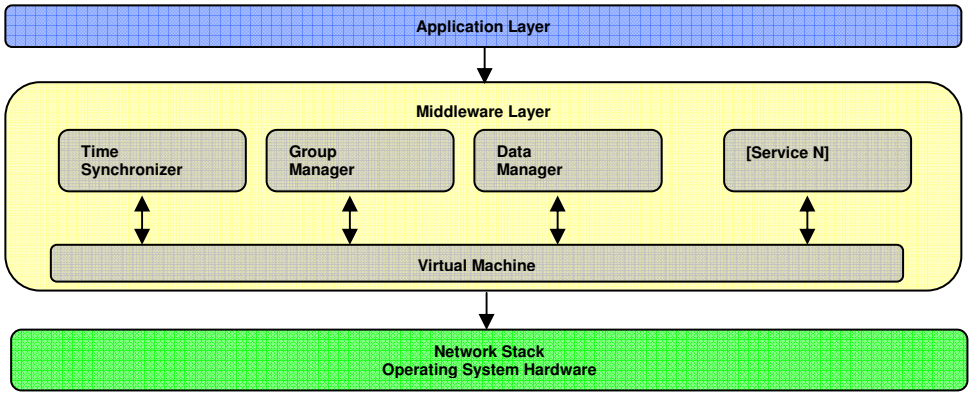

Figure 3 Proposed Middleware Architecture

4 Sensor Network Management

The EmNetS project is developing a general purpose remote management system to monitor, manage and control the behaviors of wireless sensor networks. To date, there has been very little research on network management and performance debugging for wireless sensor networks. This is mainly because the original vision for such networks envisaged extremely dense, random deployments of very inexpensive nodes, operating fully autonomously to solve or avoid faults and performance problems. However in reality, it is not the case. Many real-world applications, including those for utilities, require carefully planned deployment in specific locations, and nodes actually are not very inexpensive. As a result, autonomous approaches will need to be complemented by traditional network management approaches, but using new algorithms that are cognizant of the severe resource constraints which characterize sensor nodes.

Some relevant work in this area includes [11], which proposes two simple application-independent protocols for collecting health data from and disseminating management messages to the sensor networks. It is limited as a passive monitoring tool only, i.e., it requires a human manager to issue queries and perform analysis on collected data. In contrast, [12] proposes to reuse the main sensing application's tree routing protocol to deliver monitoring traffic. As a result, it might be non-trivial to

adapt the current monitoring mechanism to different classes of applications. In [13], the authors have surveyed some existing work in sensor network management. They found that currently there's no generalized solution for sensor network management.

Part of the EmNetS project has been targeted to fill this gap in sensor network research. Our goal is to develop a simple yet efficient, general purpose, policy-based sensor network management system which should exhibit the following important characteristics: low management overhead, strong fault tolerance, adaptive to network conditions, autonomous and scalable.

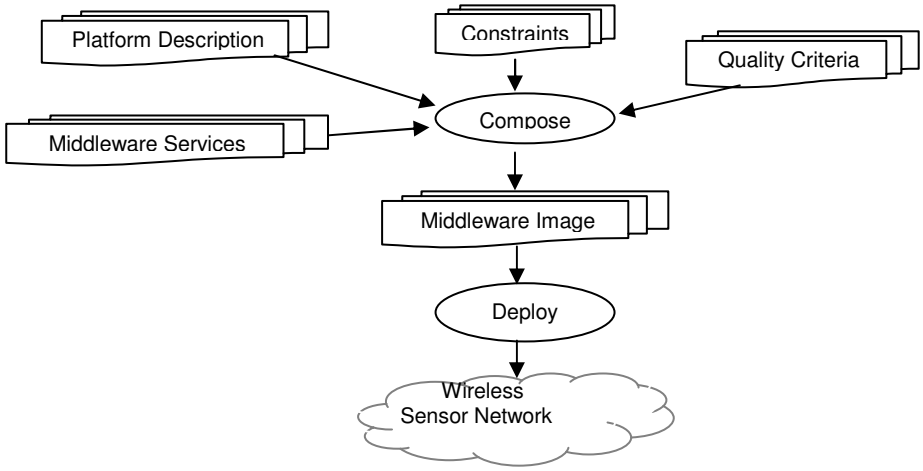

Figure 4 Processes of Composition and Deployment

4.1 EmNetS's Network Management System

To strike a good balance between scalability and complexity, a hierarchical network management system would be an appropriate solution. We propose to use several layers of management, in which the managers in the lowest layer directly manage the sensor nodes in their part of the network. Each manager passes collected health data to its higher-level manager and at the same time disseminates commands from the higher-level manager to the nodes it manages. Typically, a management layer of the proposed EmNetS's Network Management System (ENMS) consists of the following components (see Figure 5):

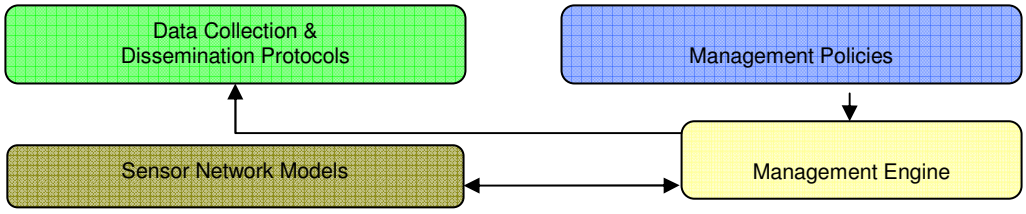

Figure 5 EmNetS's Network Management System Components

(1) Sensor network models: The models are to depict the actual states of the sensor networks. There are various possible sensor network models to be captured in our management system, for example link quality map, network topology map, energy map, etc. An important requirement is that the network models must be extensible to easily accommodate future classes of sensing applications.

(2) Data collection and dissemination protocols: To collect health data from sensor networks, we are exploring a combination of energy-efficient application-dependent and application-independent data collection protocols. The former protocol would use the main sensing application's tree routing protocol to deliver health data from the sensor networks. One way to implement this approach is to piggy-back the health data into the real application data packets. The latter has the advantage of being independent from the sensing applications, thus can be easily adapted to be used in different applications. Moreover, when the application fails, the latter protocol can still continue functioning. We envisage a combined protocol in which the former approach would be used to report network health data periodically, while the latter is for active node probing and sending management messages when required.

(3) Management policies: ENMS's management policies will specify tasks to be executed if certain system health conditions are met, e.g., battery level of node A is now 10%, so node A should go to sleep mode.

(4) Management execution engine: The management engine uses the data collection/dissemination protocol to update the sensor network models, and to send commands to sensor nodes. Based on the collected health data, and the management policies, it will then automatically analyze the current situation and execute the right management tasks, e.g., re-configuring a network route in case of congestion. Together with well-defined management policies, an intelligent management engine would help to achieve the desirable level of autonomy for our ENMS, thus minimizing the need of human managers.

5 Test bed Implementation

One of the key activities within the EmNetS project is the development of a live system test bed. The objective of the test bed is twofold. Firstly it provides a hardware platform to validate protocols and network architectures developed within EmNetS and to execute experiments demonstrating the suitability of the developed platform for utilities and responsive building applications. Secondly, the test bed provides network management functionality via a backbone USB/WiFi network for topology control, network element status indication, performance/fault analysis, configuration and remote programming of individual devices. The test bed architecture, depicted in Figure 6, is based on the Re-Mote test bed architecture developed at the University of Copenhagen [14].

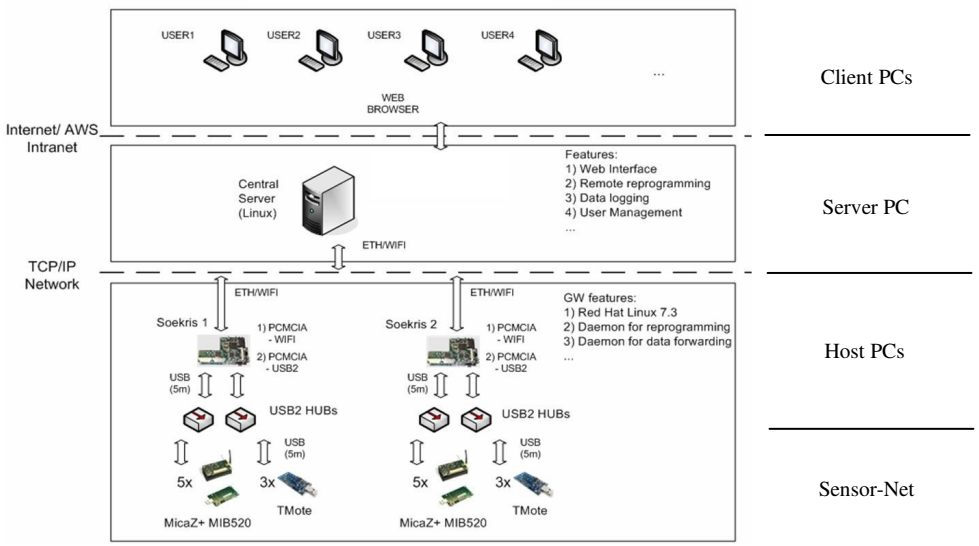

Figure 6 EmNetS Test bed

The test bed consists of the following layers –

The *Sensor Net* layer consists of the sensor devices. The Xbow MICAz and Moteiv TMote are currently supported. The test bed includes support for software stacks developed in TinyOS versions 1 and 2. Future work in this layer includes the additional support for the Contiki operating system and the provisioning of more advanced components for testing and debugging. The *Hosts PCs* consist of Linux based embedded PC platforms with USB2 connectivity to the sensor nodes in the layer below. The host PCs contains the mote control host daemon to facilitate mote discovery and issuing mote commands start, stop and reset for specific network topology control and remote power management. This layer also contains the Bootloaders for the sensor devices within the testbed. Connectivity to the upper IP based Server layer is via an Ethernet/WiFi link. Future work in this layer will focus on the implementation of wireless USB and the provision of application data logging. The *Server PC* layer contains the mote control server daemon to bridge the communication between the clients in the upper layer to motes in layer 1. It also contains an MYSQL based database for central storage of all system information and a TOMCAT information server to provide client system information, user authentication and mote information. Connectivity to the top layer is via WiFi link. Further extensions

in this layer are focused on development of an administrative interface to manage the network users, reservation of test bed resources and mote information. The *Client PC* layer forms the top layer in the sensor test bed in which the user can interact with the system. Each client PC contains a java graphical user interface which lists the available motes and provides services such as start, stop and reset of the individual devices. Individual motes can also be reprogrammed with a console window available for each device. Future directions in this layer include the development of an interactive map showing the node deployment, a reservation system by which users can reserve network resources for a particular test and a data logging facility. It is envisaged that the Sensor-Net test bed architecture will evolve across multiple domains and institutes providing a tool for remote access and the deployment of networking protocols to further advance wireless sensor network research.

6 Conclusion

The EmNetS project is current undertaking a research programme in the area of embedded wireless sensor networks and is advancing the current state of the art in energy efficient, scalable networking protocols, middleware, network management and testbed development. The application domain under investigation includes the responsive building environment which provides a number of key research challenges in terms of energy efficiency and scalability. This paper has provided an overview of current research activities within the programme and highlighted the challenges and solutions under development.

References

[1] http://www.cs.ucc.ie/emnets/

[2] IEEE 802.15.4 Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs), 2006

[3] A. El-Hoiyi, J.-D. Decotignie, and J. Hernandez. Low power MAC protocols for infrastructure wireless sensor networks. In Proceedings of the Fifth European Wireless Conference, Feb. 2004.

[4] Joseph Polastre, Jason Hill & David Culler, "Versatile Low Power Media Access for Wireless Sensor Networks", Proc. Embedded Networked Sensor Systems, 2004, pp. 95 – 107

[5] Anis Koubaa, Mellek Attia, "Collision-Free Beacon Scheduling Mechanisms for IEEE 802.15.4/Zigbee Cluster-Tree Wireless Sensor Networks", Technical Report, Version 1.0, Nov. 2006, http://www.open-zb.net/

[6] Ho-In Jeon, Yeonsoo Kim, "BOP Location Considerations and Beaconing Scheduling for Backward Compatibility to Legacy IEEE 802.15.4 Devices", submitted to IEEE 802.15.5 Task Group 5.

[7] A Holistic approach to multi-hop routing in sensor networks, Alec Lik Chuen Woo, Thesis, University of California, Berkeley.

[8] The Haas, Z. J., and Peatlman, M. R., "The zone routing protocol (ZRP) for ad hoe networks", Internet Draft -- Mobile Ad hoc NETworking (MANET) Working Group of the Internet Engineering Task Force (IETF), November 1997.

[9] Joel Koshy, Raju Pandey, "VM*: Synthesizing Scalable Runtime Environment for Sensor Networks", *SenSys'05*, San Diego, California, USA, 2-4 November 2005.

[10] Peter Graubmann, Mikhail Roshchin, "Semantic Annotation of Software Components", EUROMICRO-SEAA'06, Cavtat/Dubrovnik (Croatia), August 28 – September 1, 2006.

[11] G. Tolle, and D. Culler, "Design of an Application-Cooperative Management System for Wireless Sensor Networks", European Workshop on Wireless Sensor Networks, Istanbul, Turkey, Jan 2005.

[12] S. Rost, and H. Balakrishnan, "Memento: A Health Monitoring System for Wireless Sensor Networks", IEEE SECON, Reston, VA, Sep 2006

[13] W. L. Lee, A. Datta, and R. Cardell-Oliver, "WinMS: wireless sensor network-management system, an adaptive policy-based management for wireless sensor networks", Tech. Rep. UWA-CSSE-06-001, The University of Western Australia, June 2006.

[14] http://www.distlab.dk/sensornet