Software Infrastructure for Wireless Sensor and Actuator Networks

Zenon Chaczko
Faculty of Engineering and IT
University of Technology (UTS)
Sydney, NSW, Australia
zenon.chaczko@uts.edu.au

Christopher Chiu
Faculty of Engineering and IT
University of Technology (UTS)
Sydney, NSW, Australia
christopher.chiu@uts.edu.au

Shahrzad Aslanzadeh
Faculty of Engineering and IT
University of Technology (UTS)
Sydney, NSW, Australia
shahrzad.aslanzadeh@student.it.uts.edu.au

Toby Dune
Faculty of Engineering and IT
University of Technology (UTS)
Sydney, NSW, Australia
toby.dune@alumni.uts.edu.au

***Abstract*—In the development of large ad-hoc Wireless Sensor and Actuator Agent Networks (SANETS), a multitude of disparate problems are faced. In order for these networks to function, software must be able to effectively manage: unreliable dynamic distributed communication, the power constraints of un-wired devices, failure of hardware devices in hostile environments and the remote allocation of distributed processing tasks throughout the network. The solutions to these problems must be solved in a highly scalable manner. The paper describes the process of analysis of the requirements and presents a design of a service-oriented software infrastructure (middleware) solution for scalable ad-hoc networks, in a context of a system made of mobile sensors and actuators.**

Keywords*– SANETS, *Middleware, Distributed System Software Services.

I. BACKGROUND

The aim of this work is to create a network made up of many individual devices that can function as a single survivable entity. The individual devices must cooperate with each other to provide processing for a single purpose. Managing failure of devices and links on the network is a key concern in providing for the survival of the key processing tasks of the network, as individual devices fail. With a distributed network of miniature, wireless sensing and actuating devices at our command, scattered throughout the area of interest we have the ability to gain an awareness of the situation not previously possible. Information is power: with more information at hand about a particular situation, it is possible to increase our power to make the right decisions at a right time. Improved techniques of both obtaining information and managing hardware devices in the environment, in the form of sensory data and devices/resource controls, bring about changes in the way we perceive and interact with the environment. Greater informational gathering capabilities and more flexible resource control can provide huge 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 realising a Sensor and Actuator Agent Network (SANET). The DSAN is to be used in the development of real-world sensor and actuator networks and the trialling of new research concepts in the SANET field. Further developments in sensor networks can utilise the functionality provided by the proposed solution of middleware, allowing researchers to be more focussed at their specific problem area. It is aim of this project to prevent development time of future research from being wasted by having to reinvent, redesign or redevelop all the underlying network management functions that are provided by the presented middleware solution.

II. MIDDLEWARE FOR SANET TECHNOLOGY

In the traditional sense, component middleware systems [15, 16] is a type of middleware that enables reusable service elements [8, 14] to be composed, configured, adapted, tested, integrated and installed to build software applications reliably and at a low cost, 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 [16] software systems 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 (Figure 1):

- **Connector Facilities within Components**
  Includes remote procedure calls, remote method invocation 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.

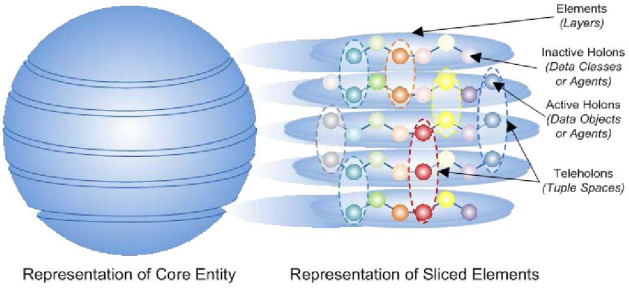

Figure 1. SANET Component Architecture Paradigm

Recent advancements in miniaturisation of sensing devices including Micro-electromechanical systems (MEMS), embedded processors such as Systems on a Chip (SoC) and in wireless communication have provided 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 software information processing and management systems, including commercial of-the-shelf (COTS) technologies, the construction of effective SANET solutions becomes a reality. Products, such as Sun's Remote Method Invocation (RMI), Microsoft's .NET Remoting and Object Management Group's (OMG) 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, resource and materials) 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 [6]; they rapidly have become in dominant use for SANET systems that are built on evolvable, autonomic [7] and ad-hoc networks with actuation and control. This encapsulates monitoring and control processes, operations, networks and hardware systems in civil and environmental engineering, computing and telecommunications, medicine, defence, 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, the need for SANET middleware models to meet stringent Quality of Service (QoS) qualitative requirements such as scalability, robustness, usability, security, efficiency, latency, privacy and trust [1, 2, 3, 8, 14, 17]. For all application domains, the ultimate goal of infrastructure oriented software system 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), thus shortening the time-to-market is essential in modern engineering and business concerns. As the majority of developer roles are to assemble distributed networked systems by selecting a combination of custom made components and compatible COTS frameworks [6, 8, 12] 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 (APIs), semantics, context and protocols which makes 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.

III. 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*" [15]. With the introduction of this new technology and emergence of wireless sensor networks, a new set of technical challenges arise [1, 5]. These challenges form the basis of the middleware.

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."*[10]. The vast number of devices on these networks prevents the ability to manually configure and repair devices individually within the system; the number of electronic and mechanical devices also increases probability of failure. A software system must automatically configure devices and must robustly handle failed devices.

B. Dynamic Hostile Environment

The devices that make up the sensor network will be deployed in the hostile environments that they need to monitor. 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 how to manage the changing availability of resources and communications links within a large network. The sensor network must respond robustly to the continually changing environment in which it is situated.

C. Power Utilisation

As quoted from Zhong, "*Power consumption is crucial to wireless sensor network applications*" [21]. The lifetime of a sensor network is a function of energy consumption [18]. To improve overall network life we must avoid key parts of the network from being over utilised and drained too quickly. Methods for distributing the processing and communication tasks evenly over the network are needed. The sensor network design 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 [5].

E. Diverse Range of Applications and Uses

The many uses of the sensor and actuator network and continuing research and development in the SANET field compels the system to be expandable and maintainable [1].

To be able to utilise this emerging technology efficiently, on an ever increasing scale, smart software systems are needed to manage the problems inherent to such networks. Software must provide the ability to manage the vast number of small independent devices in a way that allows for the effective combination of all of the available resources. All devices must be able to interact and work together to support a common goal. A software system middleware that can solve these problems will facilitate the easy creation of new sensor network systems. A sensor network specific middleware will streamline the development of customised sensor networks 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 could be encountered within the SANET context.

IV. 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 [8, 12, 18]. This covers a suite of functional and non-functional requirements that need to be addressed when modelling the middleware and designing software components specifically to support variety of operations in distributed SANETS [6, 20]. 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. OneWire and CanBus) and wireless processing (i.e. Bluetooth IEEE 802.15.1 and ZigBee IEEE 802.15.4), and reporting to a centralised data centre, as depicted in Figure 2.

A. Functional Requirements

The set of high level functional requirements that the DSAN middleware must address includes the following system requirements:

- The communication interfaces with embedded sensor and actuator devices;
- Automated health monitoring of available resources within the SANET system;
- Automated configuration management for the sensor and actuator 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 interface for:
  - Viewing system state and warnings; and
  - Configuring the processing tasks on the devices within the sensor network

The TINI (*Tiny Inter-Net Interface*) [17] device has been chosen to provide for the reference implementation of the distributed embedded sensor devices. The standard Java runtime, with the addition of Jini [11] services, was chosen to provide a platform for the scalable centralised services required.

B. Non-Functional Requirements

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

- **Lightweight Implementation**
  The code solution must efficiently utilise the resources available on the small embedded computing devices used in sensor/actuator networks;
- **Robustness**
  The middleware must gracefully handle failure of wireless and in-situ components registered in the network;
- **Scalability**
  The nature of middleware must be inherently scalable, so it is able to support the massive number of devices contained within the SANET system. Furthermore, the distribution and scale of the SANET environment means potential geographic spread among various infrastructure interconnections and software interfaces; this is solved with Java's Remote Method Invocation (RMI);
- **Adaptability**
  The middleware must gracefully adapt to the continually changing health status of the sensor network, while also maintaining the processing needs of the user(s); and
- **Flexibility**
  The diverse range of situations that the middleware can be applied to and the young age of the technology mean the middleware must be flexible. To cater for new research, the system must allow for components implementing specific functionality to be updated or replaced without affecting the rest of the middleware.

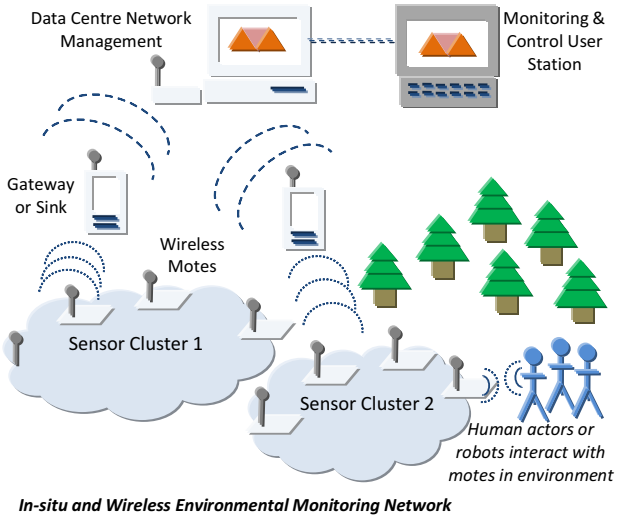

Figure 2. Overview of DSAN Middleware

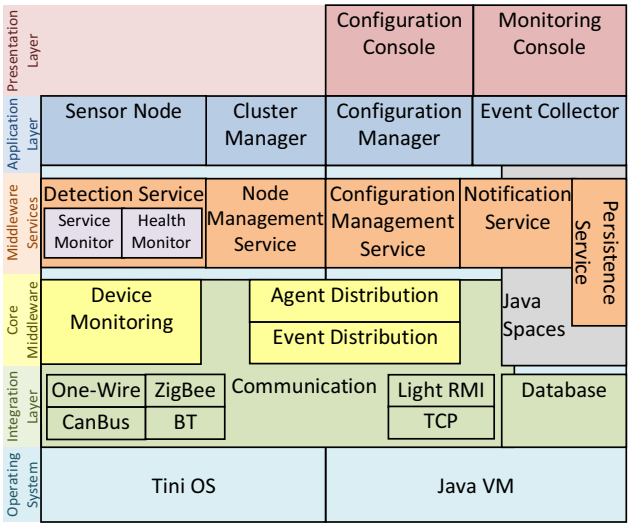

Figure 3. Architectural Model of DSAN Middleware

V. MIDDLEWARE SERVICES

A set of services have been designed to provide an initial set of solutions for each of the diverse challenges involved in the sensor network application. The services provided by the middleware are divided into a layered model (Figure 3). The middleware services must be implemented over different hardware architectures; a set of distributed lightweight services must interact 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 utilisation to run on embedded devices;
- Platform/media layer independent addressing mechanism;
- Transmission media independence;
- Synchronous Communication (Lightweight RMI) for device interaction, device control, and agent activation;
- Asynchronous Communication for availability 'heart beating' and the sensor interface;
- Mobile Code for Agent Distribution; and
- Robust Communication Error handling

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 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 on 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 centralised 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 utilising 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 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. 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. Users view the node's state of activeness, how many are active and what environmental properties they are 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 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 peers nodes to monitor health each other. The remote configurability of the System Health Service allows for health checking to be decentralised and distributed throughout the network. Traditionally, if all health checking tasks were provided by the node managers themselves, a greater utilisation and power drain would be centred on the node manager, causing premature failure. As illustrated in Figure 4, the node manager needs to continually talk with all five nodes near its maximum communication range. Distributing the task of health checking to individual nodes within the network, power utilisation will be more evenly drained.

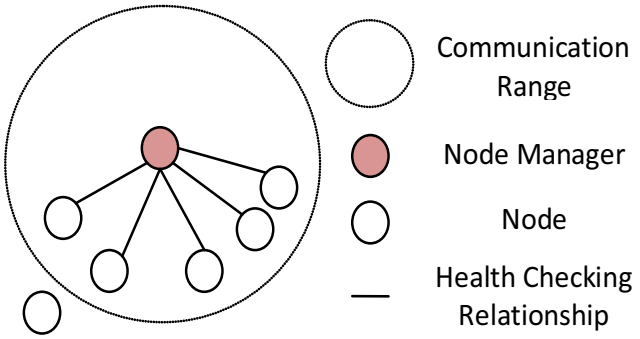

Figure 4. Centralised Health Checking in DSAN Middleware

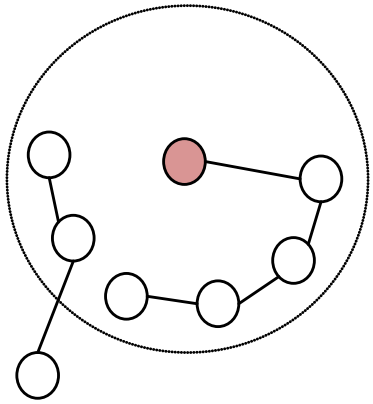

Figure 5. Decentralised Health Checking in DSAN Middleware

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 5 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 neighbouring 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; 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 utilised. 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 optimal processing in accordance with specified processing allocations. Processing and data aggregation algorithms are implemented as agents within the service. These agents utilise 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 JavaSpace service of JINI. In this release of the DSAN middleware system, the reference implementation of JavaSpaces as provided by Jini is used. The DSAN middleware relies on the expandability of the JavaSpaces model, future releases of the DSAN software may have to use a more scalable and capable implementation of the JavaSpaces service.

VI. MAIN CONCLUSIONS

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 e-wiki tool in TRAC, 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 young and a lot of work is being done. Many aspects of the domain are currently in early research and development stages. This means that many new developments are being made and are open to being incorporated into a middleware solution. The DSAN is designed specifically with the future of the 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. The use of sound design principles enables expandability of the middleware by future research. 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. Wireless communication can consume a lot of power, so new developments in power-aware algorithms and design principles are needed to maximise 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 some real-world situations.

REFERENCES

[1] Akyildiz, I. F., Kasimoglu, I. H., (2004) *Wireless Sensor and Actor Networks: Research Challenges*, Ad Hoc Networks, 2(4), pp.351-367

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

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

[4] Chaczko, Z., Resconi, G., (2008) *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*, Partial Proceedings of the Eighth International Conference CASYS'07 on Computing Anticipatory Systems, Liège, Belgium, August 6-11, 2007, D. M. Dubois (Ed.), Application of Biomimetic Design Methods in Infrastructure Systems. Chaos, Liège, Belgium, Vol.21. pp.372-385, Publ. by Chaos, 2008, ISSN 1373-5411 ISBN 2-930396-08-3

[5] Chong, C.-Y., Kumar, S.P., (2003) *Sensor Networks: Evolution, Opportunities, and Challenges*. Proceedings of the IEEE 2003, 91(8), pp.1247-1256

[6] Chu, X., Buyya, R., (2007) Service Oriented Sensor Web. In: Mahalik, N. P. (ed), *Sensor Network and Configuration: Fundamentals, Standards, Platforms, and Applications*. Springer-Verlag, ISBN: 978-3-540-37364-3, Germany, Jan. 2007, pp.51-74

[7] Ganek, A. G., Corbi, T. A., (2003) *The dawning of the Autonomic Computing Era*, IBM Systems Journal 2003, 42(1), pp.5-18

[8] Golatowski, F., Blumenthal, J., Handy, M., Haase, M., Burchardt, H., Timmermann, D., (2003) *Service-Oriented Software Architecture for Sensor Networks*, In Proc. Int. Workshop on Mobile Computing (IMC'03), Rockstock, Germany, June, pp.93-98

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

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

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

[12] Ngai, E. C.H., Lyu, M.R., Liu, J., (2006) *A Real-Time Communication Framework for Wireless Sensor-Actuator Networks*. In Proc. IEEE Aerospace Conf., Big Sky, Montana, U.S.A., March 2006

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

[14] Rezgui, A., Eltoweissy, M., (2007) *Service-Oriented Sensor-Actuator Networks*. IEEE Communications Magazine 2007, 45(12), pp.92-100

[15] Shen, C., Srisathapornphat, C., (2001) *Sensor Information Networking Architecture and Applications*, University of Delaware, In IEEE Personal Communications, August 2001 p.52

[16] Szyperski, C., (1998) *Emerging component software technologies - A Strategic Comparison*, Software Concepts and Tools, Vol 19, No 1, pp.2-10

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

[18] Vidhyapriya R., Vanathi, P.T., (2007) *Conserving Energy in Wireless Sensor Networks*. IEEE Potentials 2007, 26(5), pp.37-42

[19] Xia, F., Zhao, W.H., Sun, Y.X., Tian, Y.C., (2007) *Fuzzy Logic Control Based QoS Management in Wireless Sensor/Actuator Networks*. Sensors 2007, 7(12), pp.3179-3191

[20] Xia, F., Tian, Y.C, Li, Y.J., Sun, Y.X., (2007) *Wireless Sensor/Actuator Network Design for Mobile Control Applications*. Sensors 2007, 7(10), pp.2157-2173

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