Cooperative Task Allocation in Utility-Based Clustered Wireless Sensor Networks

Mo Haghighi

***Abstract*—Wireless Sensor Networks (WSNs), due to their many features and capabilities, have become the mainstream for monitoring various phenomena of interest, ranging from monitoring environmental conditions to structural health monitoring, medical and military applications. However, they suffer from a number of resource constrains, which affect their operational range and lifetime longevity. Auction-based techniques, although mainly used for economic applications, in the recent years have shown effective results for modeling competitive environments in various engineering fields, especially wireless networks. In previous studies, we have shown how market-based techniques can be integrated on the node-level for efficiently executing the multi-paradigm applications. In this paper, we will investigate how these techniques may be utilized on the cluster and network levels for saving energy and prolonging network lifetime.**

***Index Terms*—Agent-based, market-based, utility, sensomax, multi-operational, multi-paradigm, concurrency, resource, allocation, clustered.**

I. Introduction

Wireless Sensor Network (WSN) consists of a number of spatially distributed sensor devices with wireless communication capabilities, which have been spread across an environment, in order to gather sensory data on various parameters of interest. WSN has become the mainstream technology for many engineering fields, due to their small size and lack of infrastructure. WSNs have wide variety of applications, ranging from environmental and health monitoring to military and agriculture. Due to the miniature size of sensor nodes and their expected long operational periods, WSNs suffer from a number of resources limitations in order to maintain reasonable network lifetime and meet the application requirements. Therefore most WSNs are very resource-constraint in terms of processor and memory. Middleware is a software layer, which intermediates between the applications and WSN hardware resources, in order to regulate resources' utilization by various applications and prolong the network lifetime. Sensomax [1], [2] is a WSN middleware, which supports multiple concurrent applications in a clustered fashion by decentralizing the network management from a single sink point (gateway) to multiple cluster-heads. Sensomax has been written in Java and having benefitted from the core Java APIs, it can provide a number of functionalities, such as threading, sockets and garbage collection, which ease the process of running soft computational algorithms. In fact, one of the key strength of Sensomax is its capability to apply computational algorithms in a distributed fashion on multiple data sources within the network, without imposing heavy overheads on its underlying resources.

Sensomax can execute multiple concurrent applications by making distinction between different types of application requirements, which can be categorized into multiple operational paradigms, including Query-driven; Data-driven; Time-driven and Event-driven.

Applications are submitted to the network in a raw form; in which case, several operational paradigms might potentially exist in a single application. Sensomax identifies and characterizes those requirements, and processes them individually or collectively based on a number of factors. Examples of such influential factors can be described as the number of concurrent applications and the amount of processing at different network levels. Therefore Sensomax needs to balance out the amount of processing amongst the available resources equally.

As was pointed out, applications are executed in a clustered fashion, in which, multiple requirements are split amongst cluster-members according to their operational paradigm. Such balancing task needs to be performed by each cluster-head in a way that it carefully migrates excessive processing on to members in idle mode. Using market-based algorithm is one of the effective techniques, which accomplishes the aforementioned task efficiently.

Market-based techniques, although being primarily used in economy, have recently become quite popular in wireless communications due to their simplicity, decentralization and scalability. In a simplest way, such techniques consider the network as a market or an auction, in which a seller offers a single task to a number of buyers. Every buyer places its bid by announcing its price to the seller, and the buyer with the highest bid will then receive the task. The same scenario can be applied in WSNs, in which a task is offered to multiple nodes for execution. Nodes estimate the amount of energy required for executing the task and place their bids by announcing their estimated costs to the seller (cluster-head). However, since reducing energy consumption is the main objective of WSNs, in this case buyer with the lowest price receives the task.

In [3], we have introduced a number of cost equations, which are used for calculating the cost associated with multi-paradigm application on the node level. In this paper however, we will investigate the feasibility of applying market-based techniques on cluster and network levels, in order to simplify the aforementioned complexities and

Manuscript received May 30, 2013; revised September 24, 2013.

Mo Haghighi is with the Department of Computer Science, University of Bristol, Bristol, BS8 1UB, UK and Large-Scale Complex IT Systems (LSCITS) (e-mail: Mo.Haghighi@bristol.ac.uk).

DOI: 10.7763/IJIEE.2013.V3.396

implement an autonomous resource-allocation mechanism in which the following objectives can be achieved:

1) Maximizing network utility (global utility) whilst minimizing the cost of meeting the pre-deployed applications requirements;
2) Maximizing cluster utility (local utility) whilst minimizing the costs of in-node processing of their cluster-members.

II. Existing Approaches

Existing works including [4]-[9] have considered Market-based techniques for various purposes in WSNs. Most of the existing research in the exploitation of computational algorithms in WSNs, whether intended at distributing processing, regulating memory allocation or minimizing energy consumption, can be narrowed down under a major category of resource allocation. In most cases such issues need to be dealt with at runtime, where a number of applications may change their requirements dynamically. In the case of market-based algorithms, the basic idea is that, just as human societies, market mechanism can facilitate decentralized resource allocation involving complex supply/demand trade-offs of goods and services in computational systems [4]. It continues to conclude that two major factors impact the increasing usage of market-based techniques including the movement towards data-centric and service-based operations; and advances in e-commerce, which has increased the effectiveness of such algorithms. [4] Also states that it is highly viable to abstract the whole network operation as a competitive market, where objective problems can be simplified into a single profit maximization problem. It is worth noting that exploiting computational methods in WSNs requires compatible middleware architecture, where parallel distributed processing is supported. There have been many middleware designed and implemented for WSNs, however they mostly lack dynamicity, interoperability, multi-tasking and autonomy. Lack of such features often result in a highly application-specific behavior, in which associated hardware are often tightly coupled to the middleware. New advances in microelectronics have brought a number of benefits to WSNs including inexpensive and energy-efficient processors, memories and miniature-sized sensor devices. Exploiting such proportionally powerful, yet energy-efficient, hardware components has facilitated development environments relatively on par with those available in the conventional systems, where high-level and object-oriented programming languages such as Java can be used.

A thorough study has been conducted on the existing market-based approaches in WSNs with particular focus on their methodologies. [5] Proposes a market-based scheme, in both centralized and distributed fashions; and an algorithm by which tasks can be priced in order to establish a fair energy balance. The centralized approach requires all nodes (sellers) to calculate and transmit their cost for performing a given task to the consumer (buyer), whereas the distributed approach requires the node to delay their responses in ratio to their prices. This way, only the winner with the lowest price will communicate with the buyer and the overhead gets significantly reduced. We implemented their pricing scheme based on Sensomax architecture and the distributed approach proved to be very efficient for small-scale networks; however as the number of nodes increases, it is more likely for multiple nodes to calculate the same price, which in that case multiple winners will communicate with the buyer simultaneously. In a real-world application, the propagation delay will improve this situation, however due to lack of such delay in large-scale simulation, the chances of packet collision and selecting amongst sellers are very high. [8] Investigated task allocation in multi-sensor surveillance systems by benchmarking their proposed market-based algorithms against three different algorithms, in both centralized and hierarchical (fixed and dynamic-tree) fashions. In this approach, mobile robots are the main targets for distributed task allocation, which offers considerable adaptability and dynamicity due to high mobility nature of the targets. Based on their experimental results, dynamic-tree allocation in a hierarchical fashion, which shares a number of similarities with Sensomax's communication model, proves to be more cost-efficient.

III. Architecture

Sensomax is an agent-based middleware, which has been developed in Java and is capable of running multiple concurrent applications in a multi-paradigm mechanism. Sensomax was originally developed in Java ME to run on a network of Sun Spot devices [11]. However it was later converted into Java SE and SE-embedded to interoperate on various Java-enabled devices.

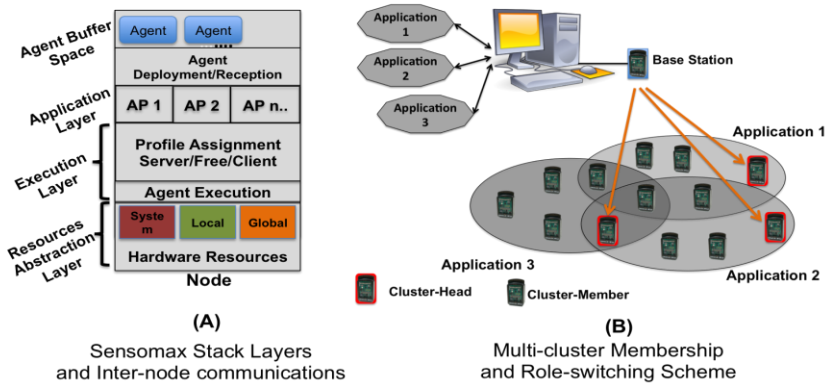

Fig. 1. Sensomax stack layers and multi-cluster execution model.

One of the most notable features of Sensomax is the possibility of dynamically injecting and executing computational methods into the middleware. End-users may write their own computational methods in Java, with respect to the components' rules that are defined by the system APIs, and Sensomax will package them into a standalone module and inject it into the network.

Sesnsomax is also a component-based middleware, in which, the overall functionalities of the system are broken down into multiple modules, each with a set of unique functions. The interaction of those modules creates a dynamic execution environment in which existing or new modules may leave or join the system freely without affecting others' operations. Data aggregation algorithms, energy management strategies, statistical analysis of captured data, and any arithmetic operations on the captured data can be integrated into the existing modules or initiated as new independent modules and added to the system. The activities of the modules are highly embedded and take place within the module, therefore no interference is imposed on others. However, to maximize the performance and energy efficiency, each operation needs to be authorized by the main kernel. Full details of Sensomax are given in [1], [2].

Application can be submitted either onsite to the base station, or remotely via the web. Each application is translated into a number of tasks/subtasks of event, timing and data conditions, and bundled up as an agent. Sensomax can accommodate multiple agents and applications concurrently and, as Fig. 1 shows, nodes can interact through their transport layers where multiple agents are received and queued for processing. Each agent is then translated back into an application and transferred into the application layer. The object-orientation and multi-threading nature of Java facilitate a number of novel possibilities in Sensomax, where each application runs in an isolated process whilst being minimally controlled by the main kernel (root process). Sensomax allocates a cluster of nodes to each application, each containing a variable number of nodes, based on the distribution of the task. The application itself resides on the cluster-head and coordinates with its members to accomplish the integrated tasks. As Fig. 1(B) shows, in order to minimize the number of nodes in the network and the overheads of multiple applications on a single cluster-head, Sensomax uses a hierarchical task allocation in which it allows nodes to role-switch between a head of one cluster and a member of another, dynamically. Such a feature not only promotes reusability of the nodes and resources, but also facilitates greater multitasking. This feature is known as the Profile Assignment, which is shown in Fig. 1(A).

As Fig. 1(A) shows, in the execution layer, each application can maintain a different profile in each node and be executed with respect to that specific profile it assumes.

Such diverse execution environment with multiple concurrent applications and agents brings a number of challenges to the developers to establish a balanced trade-off between performance and energy management, whilst satisfying the applications requirements. In Sensomax, these challenges are dealt with, by a number of customizable computational agents on multiple levels of node, cluster and network. In previous publications [12], [13], we have experimented the utilization of soft computational algorithms for detecting sudden change points in several data streams, and to automate memory allocation by estimating the required data storage for multiple concurrent applications respectively.

In this paper we use market-based techniques to accomplish the following objectives, whilst maintaining efficient energy management, appropriate performance and meeting the applications requirements:

3) Autonomous decision-making on assigning applications to cluster-heads;
4) Keeping balance between the number of submitted applications and allocated clusters;
5) Making trade-offs between assigning applications to an existing cluster or creating a new one;
6) Symmetrical distribution of subtask to cluster members;
7) Prioritizing received agents for sequential processing;
8) Queuing applications for execution on the node, network and cluster levels;
9) Locating and collecting specific captured variables;
10) Regulating access to shared resources by multiple applications.

Based on the four proposed algorithms in [3], the total cost of each application per node, can be calculated by adding up all the costs contained in each application in ratio to the total number concurrent access, to the power of Z, which indicates the number of shared resources, as shown in Equation 1.

$$C_{Application} = \frac{1}{SH^{Z}}(P_Q + P_D + P_T + P_E) \quad (1)$$

$$\left| SH = \text{Shared Resources}, Z = \text{Multiple Access} \right.$$

Since the total impact of each concurrent application has already been taken into account in Equation 1-4 in [3], the total utility of node is shown in Equation 2. It is worth noting that since every node can belong to different clusters concurrently, the cost of each application is therefore reduced based on total utility of the node in that specific role. R stands for the collection of all concurrent applications and W stands for the whole collection of roles a node can assume. Therefore the total node utility (idle time) can be described as the deduction of all applications cost from the total node lifetime.

$$U_{Node} = M - \sum_{A=1}^{A=S^W} \sum_{j=0}^{j=S^R} C_{Application_j} \cdot \frac{1}{(U_{Role_A})^{S^R}} \quad (2)$$

$$\left| Application_j \in R, Role_A \in W \right.$$

As Equation 3 shows, the same concept applies to the clusters utilities. Utility of every cluster can be calculated as the summation of all members' utilities plus the ratio of the cluster-head's utility to the cluster density.

$$U_{Cluster} = [\sum_{i=1}^{i=S^{Cluster}} U_{Node_i}] + \frac{U_{Cluster-Head}}{D_C} \quad (3)$$

$$\left| Node_i \in Cluster, D_C = \text{Cluster Density} \right.$$

Sensomax's component-based architecture makes it seamless to dynamically integrate and apply various soft computations, be it at the cluster-head, the end nodes, or both.

Since Sensomax's architecture executes applications in the cluster-heads, Equations 1-5 in [3] are integrated into every node, whereas the aforementioned equations 2 and 3 are dynamically loaded into the relevant cluster-heads, in which, every node and cluster's utilities are calculated. Every equation, expressed algorithmically, is translated into a computational component and lies in the computational layer of the relevant resources. As we mentioned before, resources and their associated agents are split into three categories of Global, Local and System. In [2] we have shown that system agents are applied and executed much quicker due to their isolated processing. For this reason and in order to avoid interfering with the normal operations of the nodes, all market-based communications and processing are performed within system agents and resources. So far we have described the process of calculating the costs and utilities of applications, nodes and clusters. However, advertising tasks, communicating bidding prices and selection of winners are the most crucial operations of market-based schemes.

Above-mentioned operations require an extensive message-exchanging between the seller and buyers. For simplicity, we have used the same distributed message exchanging technique as in [5], where a task is advertised to the network by an end-user; each node calculates its cost for executing the task, and delays its response in proportion to the calculated cost. Therefore the winner is the first one communicating with the buyer. This process is performed on three levels: base station advertising each application to the cluster-heads; cluster-heads advertising subtask to their members; and nodes advertising the sub-subtasks to their concurrent applications. However it is worth mentioning that the initial advertising happens in a centralized fashion, where a task agent is broadcasted to all. For the reasons mentioned earlier, we have added a centralized acknowledgement message with the winner ID sent by the seller to all the nodes in order to inform them about the winner. This is to prevent the losers to forward their bidding price after winner has been selected. Also in case of having multiple nodes sending the same bid, the seller is either required selecting the winner randomly or necessitates further communications with the node, in order to request other properties. To resolve such situations, nodes (buyers) are required to advertise their remaining lifetime along with their bidding prices. As we said such problem only exists in the simulation environments where propagation is delay is missing and their bidding prices might arrive simultaneously. However, as the remaining lifetime of all nodes are all different, this will ensure, a second criteria exists for selecting the winner. Due to the length restriction of this paper, a number of details on the equations' derivations have been omitted.

IV. Evaluation

Our first experiment was conducted on 20 Sun Spot devices. In this experiment, four simple tasks involving query, data, time and event requirements, which demand two variables (Temperature and Light), were submitted to the base station. The base station advertises the tasks to the cluster-heads and requests their bids. This experiment was conducted in two different phases; in the first one, as shown in Fig. 2, cluster-heads V, X, Y and Z are instructed to advertise query-driven subtasks to node A and data-driven, time-driven and event-driven subtasks to node B, C and D respectively. In the second phase one however, as shown in Fig. 3, there is no limitation on how to distribute subtask requirements. The first phase was conducted to reassure that cluster-members of different clusters expose relatively coherent costs, in order to validate the functionalities of the aforementioned paradigm-based equations.

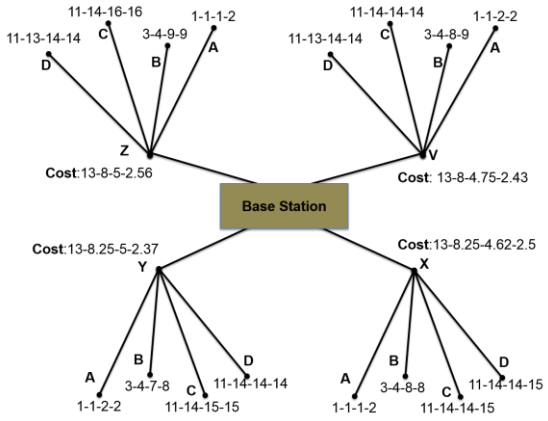

Fig. 2. Single-paradigm cost distribution.

Fig. 2 demonstrates how different cluster-heads display relatively close bids for executing the same applications. It is worth mentioning that no selection is performed by the cluster-head in this phase and all cluster-heads are required to execute the tasks regardless of the bids received from the their members. Also, the slight variation of cost estimation shown by each node is due to different remaining energy levels. One interesting point to note here is the constant decrement in the prices announced by the cluster-heads. As we mentioned there are two variables in this scenario SH and Z. According to equation 5, SH = 2 and with more concurrent applications joining the cluster head (which in this case demands the same variables) Z, which stands for multiple access, also increases by one unit per application.

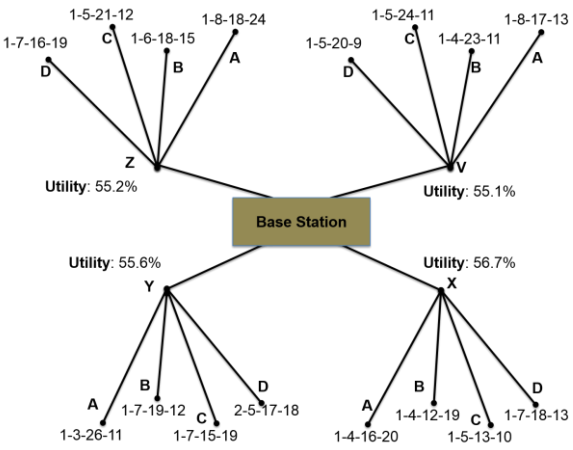

Fig. 3. Mixed-paradigm cost and utility distribution.

In the second part of this experiment, as shown in Fig. 3, four tasks are sent to the cluster-heads, in which they are required to advertise the subtask of each task to their members, where nodes with the lowest estimated costs, receive the subtasks. Every node can execute multiple

subtasks concurrently, and based on the type and number of concurrent subtasks (as we have shown in equations 1-3) every node's bid for the next subtask will be affected. In this experiment, cluster X received the highest utility of 56.7%. It is worth mentioning that the overall utilities of all nodes were nearly 70%. However this approach although being efficient in this experiment, is not appropriate for large-scale networks with variable cluster densities, as the amount of communications for coordinating the requirements of the same task and failure rates amongst nodes will be considerably higher.

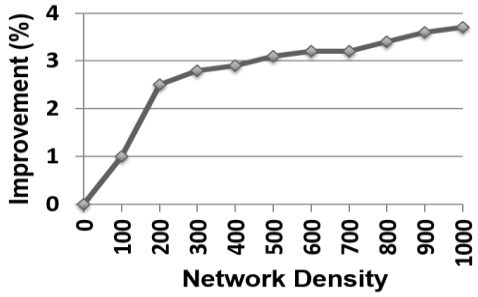

Fig. 4. Network lifetime improvement vs. network density.

Fig. 4 shows the overall impact of market-based techniques on the network lifetime based on network density, which was conducted using SensomaX Companion Simulator (SXCS) [10], on 1000 virtual nodes. According to this figure, network lifetime is improved by nearly 4% with a density of 1000 nodes. On average the network shows 3% increase in lifetime longevity with 200-700 nodes. Finally, Fig 5 shows the improvement in the task distribution process of (in the same experiment as in Fig. 4), by selecting the most suitable node (by the cluster-head) based on node's remaining energy and the number of concurrent applications.

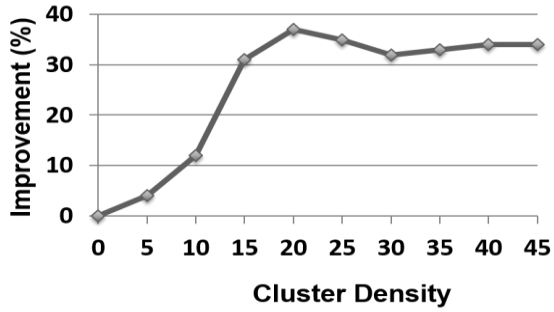

Fig. 5. Network lifetime improvement vs. cluster density.

V. Conclusion

In this paper we have shown how market-based techniques can be exploited to calculate the utilities of the clusters and networks, in order to distribute tasks on cluster and network levels. Sensomax has immensely benefited from market-based scheme in terms of regulating the energy consumption. The effectiveness of market-based techniques has proven to be proportionate to the network and cluster density, due to the growing number of competitive players. Apart from the additional overhead imposed by the calculations involved in the cost estimation phase, and the extra communications between the buyers and the sellers, such a scheme can certainly optimize the intra-cluster decision-makings on the network level.

References

[1] M. Haghighi and D. Cliff, "Sensomax: An agent-based middleware for decentralized dynamic data-gathering in wireless sensor networks," presented at The 2013 International Conference on Collaboration Technologies and Systems, CTS 2013, May 2013, San Diego, USA.

[2] M. Haghighi and D. Cliff, "Multi-Agent support for multiple concurrent applications and dynamic data-gathering in wireless sensor networks," presented at The Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, IMIS-2013, July 2013, Taiwan.

[3] M. Haghighi, "Market-based resource allocation for energy-efficient execution of multiple concurrent applications in wireless sensor networks," presented at the International Conference on Ubiquitous context-awareness and Wireless Sensor Networks, Jeju, Korea, July 2013.

[4] A. T Zimmerman, J. P Lynch, and F. T Ferrese, "Market-based computational task assignment within autonomous wireless sensor networks," in *Proc. IEEE International Conference on Electro/Information Technology*, pp. 23-28, 2009.

[5] N. Edalat, X. Wendong, T. Chen-Khong, E. Keikha, and O. Lee-Ling, "A price-based adaptive task allocation for wireless sensor network," in *Proc. IEEE 6th International Conference on Mobile Adhoc and Sensor Systems*, pp. 888-893, 2009.

[6] T. Mullen, V. Avasarala, and D. L. Hall, "Customer-driven sensor management," *IEEE Journal of Intelligent Systems*, vol. 21, pp. 41-49, March 2006.

[7] W. Cheng, L. Shengbin, L. Wei, Y. Zongki, and X. Kanru, "A price-based distributed algorithm for optimal utility-energy trade-off in wireless sensor networks," in *Proc. 66th IEEE Vehicular Technology Conference*, pp.143-147, 2007.

[8] A. M Elmogy, A. M. Khamis, and F. O. Karray, "Dynamic complex task allocation in multisensor surveillance systems," in *Proc. 3rd International Conference on Signals, Circuits and Systems* (SCS), pp. 1-6, 2009.

[9] M. I. Khan, B. Rinner, and C. S Regazzoni, "Resource coordination in wireless Sensor Networks by combinatorial auction based method," in *Proc. 3rd IEEE International Conference on Networked Embedded Systems for Every Application*, pp. 1-6, 2012.

[10] M. Haghighi, "An agent-based multi-model tool for simulating multiple concurrent applications in WSNs," *Journal of Advances in Computer Networks (JACN)*, June 2013, Malaysia.

[11] Oracle, "Sun Spot Programmer's manual," Release v6.0, Sun Labs, Oracle, 2010.

[12] M. Haghighi and C. J. Musselle, "Dynamic collaborative change point detection in wireless sensor networks," presented at the International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, October 2013.

[13] M. Haghighi, "Dynamic data storage estimation for multiple concurrent applications using probability distribution modeling in WSNs," *Journal of Advances in Computer Networks* (JACN), June 2013, Malaysia.

Mo Haghighi is a doctoral researcher at the University of Bristol, UK. He is currently pursuing his research in the area of "Decentralized Agent-based Adaptive Dynamic Data Gathering in Large-scale Wireless Sensor Networks". He has obtained a BEng in Electronic and Telecommunications engineering followed by an MSc in Wireless Sensor Networks. He began his research in a joint collaboration between the University of Bristol, the BAE Systems and Large-Scale Complex IT Systems (LSCITS). Prior to his PhD, he had worked for Sun Microsystems/Oracle for over two years, primarily involved in academic projects. As a member of LSCITS, his research has broadened to include complexity science, Cloud computing, multi-agent systems and quantitative data analysis for large-scale complex systems. Mo has extensive programming experience in Java, C++ and Assembly. He also specializes in designing embedded systems (ARM, Freescale, Microchip and Intel), large-scale distributed systems, network security, machine learning, LoWPANs and Microwave communications.