CK Tham, "Sensor-Grid Computing and SensorGrid architecture for Event Detection, Classification and Decision-Making", book chapter in "*Sensor Network and Configuration*: Fundamentals, Techniques, Platforms, and Experiments", ed. NP Mahalik, Springer-Verlag, Germany, June/July 2006

Sensor-Grid Computing and SensorGrid architecture for Event Detection, Classification and Decision-Making

Chen-Khong THAM
Dept of Electrical & Computer Engineering
National University of Singapore
E-mail: eletck@nus.edu.sg

Summary

Integrating sensor networks and grid computing in *sensor-grid computing* on a *SensorGrid* architecture is like giving 'eyes' and 'ears' to the computational grid. Real-time information about phenomena in the physical world can be processed, modelled, correlated and mined to permit 'on-the-fly' decisions and actions to be determined on a large scale. Examples include surveillance for homeland security, business process optimization, environment monitoring with prediction and early warning of natural disasters, and threat management (e.g. missile detection, tracking and interception). In this chapter, we describe the sensor-grid computing concept, the SensorGrid architecture and present our work on information fusion, event detection and classification, and distributed autonomous decision-making on SensorGrid, as examples of what can be achieved. Finally, we discuss several research challenges that need to be overcome before such a concept can become reality.

Keywords: sensor networks, grid computing, information fusion, detection, classification

1. Introduction

Recent advances in electronic circuit miniaturization and micro-electromechanical systems (MEMS) have led to the creation of small sensor nodes which integrate several kinds of sensors, a central processing unit (CPU), memory and a wireless transceiver. A collection of these sensor nodes forms a *sensor network* which is easily deployable to provide a high degree of visibility into real-world physical processes as they happen, thus benefitting a variety of applications such as environmental monitoring, surveillance and target tracking. Some of these sensor nodes may also incorporate actuators such as buzzers and switches which can affect the environment directly. We shall simply use the generic term 'sensor node' to refer to these sensor-actuator nodes as well.

A parallel development in the technology landscape is *grid computing*, which is essentially the federation of heterogeneous computational servers connected by high-speed network connections. Middleware technologies such as Globus (2006) and Gridbus (2005) enable secure and convenient sharing of resources such as CPU, memory, storage, content and databases by users and applications. This has caused grid computing to be referred to as 'computing on tap', utility computing and IBM's mantra, 'on demand' computing. Many countries have recognized the importance of grid computing for 'eScience' and the grid has a number of success stories from the fields of bioinformatics, drug design, engineering design, business, manufacturing and logistics.

There is some existing work on the intersecting fields of sensor networks and grid computing which can be broadly grouped into three categories: (1) 'sensorwebs', (2) sensors to grid, and (3) sensor networks to grid. In the first category of 'sensorwebs', many different kinds of sensors are connected together through middleware to enable timely and secure access to sensor readings. Examples are the SensorNet effort by the Oak Ridge National Laboratories (ORNL) and NIST (USA) which aims to collect sensor data from different places to facilitate the operations of the emergency services; the IrisNet effort by Intel to create the 'seeing' Internet by enabling data collection and storage, and the support of rich queries over the Internet; and the Department of Defense (USA) ForceNet which integrates many kinds of sensor data to support air, sea and land military operations. In the second category of sensors to grid, the aim is to connect sensors and instruments to the grid to facilitate

collaborative scientific research and visualization (Chiu and Frey 2005). Examples are the efforts by the Internet2 and eScience communities in areas such as the collaborative design of aircraft engines and environment monitoring; DiscoveryNet (Imperial College UK) which aims to perform knowledge discovery and air pollution monitoring; and the earthquake science efforts by the CrisisGrid team (Indiana University USA) and iSERVO (International Solid Earth Research Virtual Observatory). Finally, in the third category of sensor networks to grid, the aim is mainly to use grid web services to integrate sensor networks and enable queries on 'live' data. Examples are efforts by Gaynor et al (2004) to facilitate quicker medical response and supply chain management; and the SPRING framework proposed by Lim et al (2005).

Our focus and approach, which we refer to as *sensor-grid computing* executing on an integrated *sensor-grid architecture* or simply '*SensorGrid*' for short (Tham and Buyya 2005) - see Fig. x.1 - builds on the three categories of existing work described above and aims to achieve more by exploiting the complementary strengths and characteristics of sensor networks and grid computing. Essentially, sensor-grid computing combines the real-time acquisition and processing of data about the environment by sensor networks with intensive distributed computations on the grid. This enables the construction of real-time models and databases of the environment and physical processes as they unfold, from which high-value computations such as analytics, data mining, decision-making, optimization and prediction can be carried out to generate 'on-the-fly' results. This powerful combination would enable, for example, effective early warning of threats (such as missiles) and natural disasters (such as tornados and tsunamis), and real-time business process optimization.

One other key aspect of sensor-grid computing is the emphasis on distributed and hierarchical *in-network processing* at several levels of the SensorGrid architecture. As will be explained later, the sensor-grid computing approach is more robust and scalable compared to other approaches in which computations are mainly performed on the grid itself. The sensor-grid computing approach together with the SensorGrid architecture enable more timely responses to be achieved and useful results to be available even in the presence of failures in some parts of the architecture.

The organization of this chapter is as follows. In Section 2, we describe a simple *centralized* approach to realize sensor-grid computing. We then point out its weaknesses and propose a *distributed* approach. In Section 3, we describe two applications of distributed sensor-grid computing which we have implemented. In Section 4, several challenges and research issues related to sensor-grid computing are discussed. Finally, we conclude in Section 5.

2. Approaches to Sensor-Grid Computing

One simple way to achieve sensor-grid computing is to simply connect and interface sensors and sensor networks to the grid and let all computations take place there. The grid will then issue commands to the appropriate actuators. In this case, all that is needed are high-speed communication links between the sensor-actuator nodes and the grid. We refer to this as the *centralized sensor-grid computing* approach executing on a centralized SensorGrid architecture.

However, the centralized approach has several serious drawbacks. Firstly, it leads to excessive communications in the sensor network which rapidly depletes its batteries since long range wireless communications is expensive compared to local computation. It also does not take advantage of the in-network processing capability of sensor networks which permits computations to be carried out close to the source of the sensed data. In the event of communication failure, such as when wireless communication in the sensor network is unavailable due to jamming, the entire system collapses. Other drawbacks include long latencies before results are available on the field since they have to be communicated back from the grid, and possible overloading of the grid (although this is unlikely).

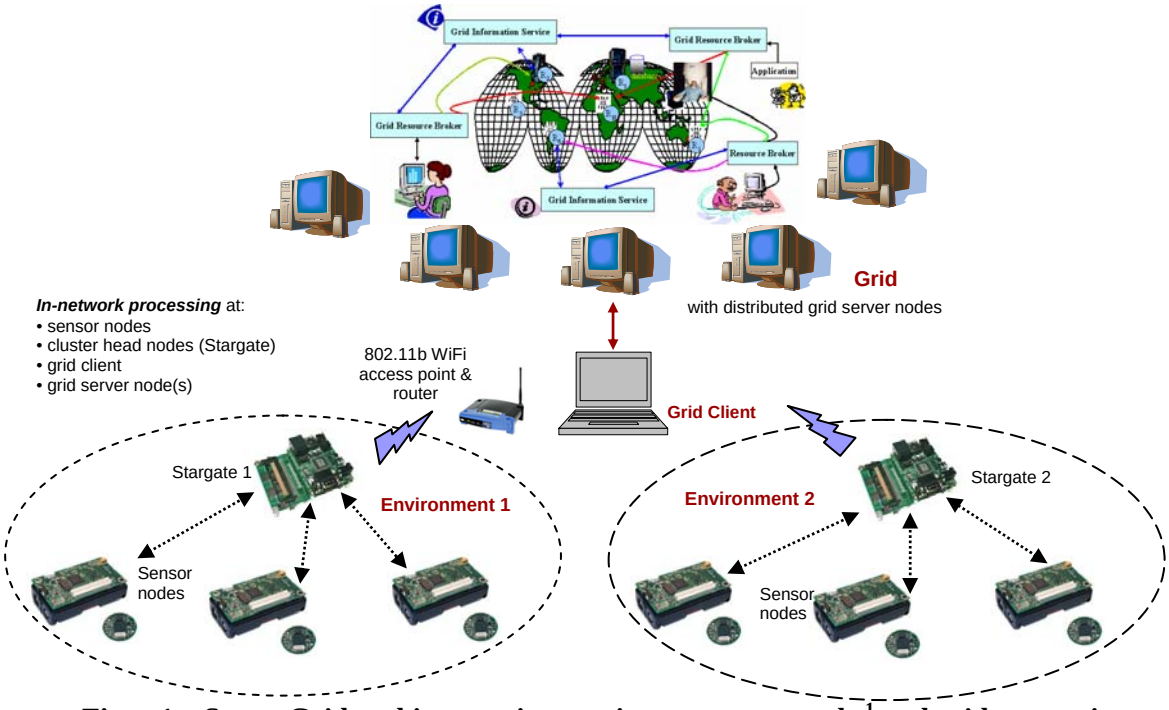


Fig. x.1 – SensorGrid architecture integrating sensor networks[1] and grid computing

The more robust and efficient alternative is the decentralized or *distributed sensor-grid computing* approach which executes on a distributed *SensorGrid* architecture and alleviates most of the drawbacks of the centralized approach. The distributed sensor-grid computing approach involves in-network processing within the sensor network and at various levels of the SensorGrid architecture. We present specific realizations of the SensorGrid architecture with sensor nodes which are Crossbow motes, Stargate nodes, Hewlett Packard iPAQ nodes, grid clients and the grid comprising a number of grid server nodes[2]. Fig. x.2 shows several possible configurations of the SensorGrid architecture. The role and characteristics of each of the components and the features of the different configurations will be discussed later in this chapter.

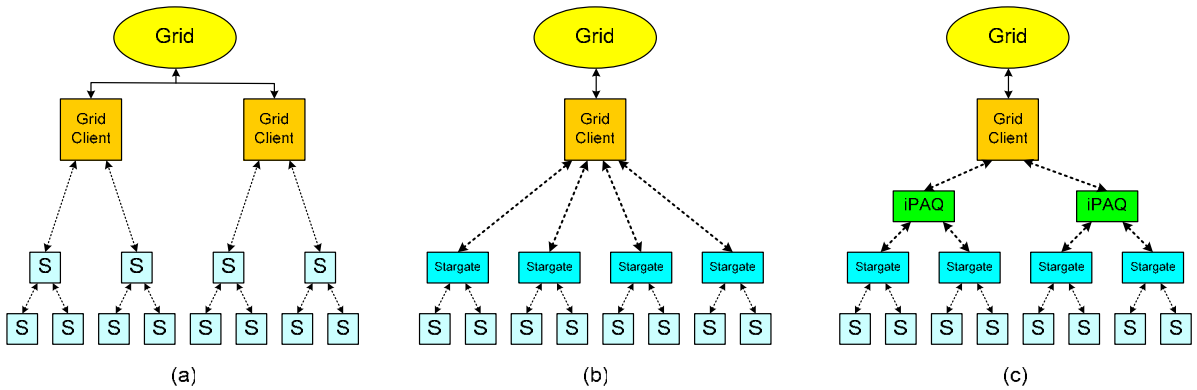


Fig. x.2 – Several possible configurations of the SensorGrid architecture
(*Note*: The squares with 'S' denote sensor nodes. Wired links are shown by solid lines and wireless links by dotted lines.)

[1] The Stargate and sensor nodes shown are products by Crossbow – see http://www.xbow.com
[2] The SensorGrid architecture can be constructed using other components with similar characteristics.

3. Distributed Sensor-Grid Computing Applications

The distributed sensor-grid computing approach is highly suitable for a number of distributed applications such as analytics, data mining, optimization and prediction. In this chapter, we present a distributed information fusion, event detection and classification application, and a distributed autonomous decision-making application, as two examples of sensor-grid computing on the SensorGrid architecture.

3.1 Distributed information fusion, event detection and classification

Since the nodes in a sensor network are independently sensing the environment, this gives rise to a high degree of redundant information. However, due to the severely resource-constrained nature of sensor nodes, some of these readings may be inaccurate. Information fusion algorithms, which compute the most probable sensor readings, have been studied extensively over the years, especially in relation to target detection and tracking.

There are two methods for performing information fusion in the context of classification (Brooks et al 2003): (i) *decision fusion*, which is done when the different measurements are statistically independent, and (ii) *data* or *value fusion*, which is done when different measurements yield correlated information. In the case of decision fusion, classification is done at individual sensor nodes or sensing modalities and the classification results are combined or fused at the *fusion center* in a number of ways, such as by applying the product or sum rule on the likelihood estimates. In the case of data fusion, feature vectors from individual sensors are concatenated and a classification algorithm is applied on the concatenated feature vector at the fusion center. Although data fusion is likely to yield better classification performance, it is more expensive to implement from the communication and computational points of view. If there are *M* sensing modalities at each sensor node, *K* sensor nodes and *n* dimensions in each measurement, data fusion typically requires the transmission of (*K*-1)*Mn* values to the fusion center (which is one of the *K* sensor nodes) and the classification algorithm there would need to operate on a *KMn*-dimensional concatenated feature vector. In contrast, decision fusion requires a classification operation to be applied on either an *n* or *Mn*-dimensional feature vector at each sensor node, followed by the communication of (*K*-1) decisions to the fusion center and the application of a decision fusion algorithm on the *K* component decisions there. In the subsequent parts of this sub-section, we will describe the design and implementation of a distributed decision fusion system for event detection and classification using the SensorGrid architecture and evaluate its performance.

3.1.1 *Distributed Decision Fusion*

To begin, we review basic statistical pattern classification theory and describe the *optimal decision fusion* (ODF) algorithm proposed by Duarte and Hu (2003).

Maximum a posteriori (MAP) classification

In statistical pattern classification, a feature vector *x* is assumed to be drawn from a probability distribution with probability density function $p(x)$. Due to a natural assignment process beyond our control, each sample *x* is assigned a class label C_n and we say that $x \in C_n$. The probability that any sample *x* belongs to a class C_n, denoted by $P(x \in C_n) = P(C_n)$, is known as the *prior probability*. The *likelihood* that a sample will assume a specific feature vector *x* given that it is drawn from a class C_n is denoted by the conditional probability $p(x \mid C_n) = p(x \mid x \in C_n)$. Using Bayes' rule, the probability that a particular sample belongs to class C_n, given that the sample assumes the value of *x*, is denoted by the *a posteriori* probability

$$p\left(x \in C_n \mid x\right) = p\left(C_n \mid x\right) = \frac{p\left(x \mid C_n\right) P\left(C_n\right)}{p\left(x\right)} \tag{0.1}$$

A *decision rule* or *classifier* $d(x)$ maps a measured feature vector *x* to a class label C_n which is an element of the set of *N* class labels $\mathbf{C} = \{C_1, C_2, \ldots, C_N\}$, i.e. $d(x) \in \mathbf{C}$. If $x \in C_n$ and $d\left(x\right) = C_n$, then a correct classification has been made, otherwise it is a misclassification. The *maximum a posteriori* (MAP) *classifier* chooses the class label among the *N* class labels that yields the largest maximum a posteriori probability, i.e.

$$d\left(x\right) = \arg\max_n P\left(C_n \mid x\right) \tag{0.2}$$

where $d(x) = n$ means $d\left(x\right) = C_n$.

Optimal decision fusion (ODF)

Let us consider a sensor network or SensorGrid configuration that consists of a fusion center and *K* distributed sensor nodes. Each sensor node observes feature vector *x* and applies a classification algorithm, such as the MAP classifier described above, to arrive at a local decision $d(x) \in \mathbf{C}$. The *K* sensor nodes forward their local decisions $d_k \in \mathbf{C}$ to the fusion center which forms a $K \times 1$ decision vector $\mathbf{d}(x) = \begin{bmatrix} d_1 & d_2 & \cdots & d_K \end{bmatrix}$.

Duarte and Hu (2003) devised a decision fusion method which they referred to as *optimal decision fusion* (ODF). The decision fusion algorithm is itself a decision rule $l(\mathbf{d}(x)) \in \mathbf{C}$ that maps a feature vector $\mathbf{d}(x) = \begin{bmatrix} d_1 & d_2 & \cdots & d_K \end{bmatrix}$ to one of the class labels. The sample space of the decision fusion classifier is finite and countable: since each decision d_k has *N* possible values, the combined decision vector $\mathbf{d}(x)$ can have at most N^K different combinations.

For each feature vector *x*, the set of *K* sensor nodes provide a unique decision vector $\mathbf{d}(x)$. The set of N^K decision vectors partition the feature space into N^K disjoint regions, denoted by $\left\{r_m; 1 \le m \le N^K\right\}$. Furthermore, let us denote the unique decision vector that every $x \in r_m$ maps to as $\mathbf{d}(m)$. Following the MAP principle at the fusion center, we have

$$l(\mathbf{d}(m)) = C_{n^*} \text{ if } P(C_{n^*} \mid \mathbf{d}(m)) > P(C_n \mid \mathbf{d}(m)) \tag{0.3}$$

for $n \ne n^*$. Using Bayes' rule, if $P(x \in r_m) \ne 0$, then

$$\begin{aligned} P(C_n \mid \mathbf{d}(m)) &= \frac{P(\mathbf{d} = \mathbf{d}(m) \mid x \in C_n) \cdot P(x \in C_n)}{P(x \in r_m)} \\ &= \frac{P(\mathbf{d} = \mathbf{d}(m); x \in C_n)}{P(x \in r_m)} \\ &= \frac{\left|\left\{x \mid x \in r_m \cap C_n\right\}\right|}{\left|\left\{x \mid x \in r_m\right\}\right|} \end{aligned} \tag{0.4}$$

where $\left|\{\ldots\}\right|$ is the cardinal number of the set. Hence, the MAP classification label C_{n^*} for r_m can be determined by

$$n^* = \arg\max_n \left|\left\{x \mid x \in r_m \cap C_n\right\}\right| \tag{0.5}$$

when the feature space is discrete. Essentially, this means that the class label of $\mathbf{d}(m)$ should be assigned according to a majority vote of class labels among all $x \in r_m$.

The ODF decision fusion algorithm is robust and produces high classification accuracy in the final classification even in the presence of faulty or noisy sensors or sensor node failures.

3.1.2 Event detection and classification on SensorGrid

The classification and decision fusion algorithms described above are implemented on the SensorGrid configuration shown in Fig. x.2(a) which consists of leaf sensor nodes and several levels of fusion centers implemented in sensor nodes, one or more grid client(s) and one or more grid server node(s). The actual system can be seen in Fig. x.3.

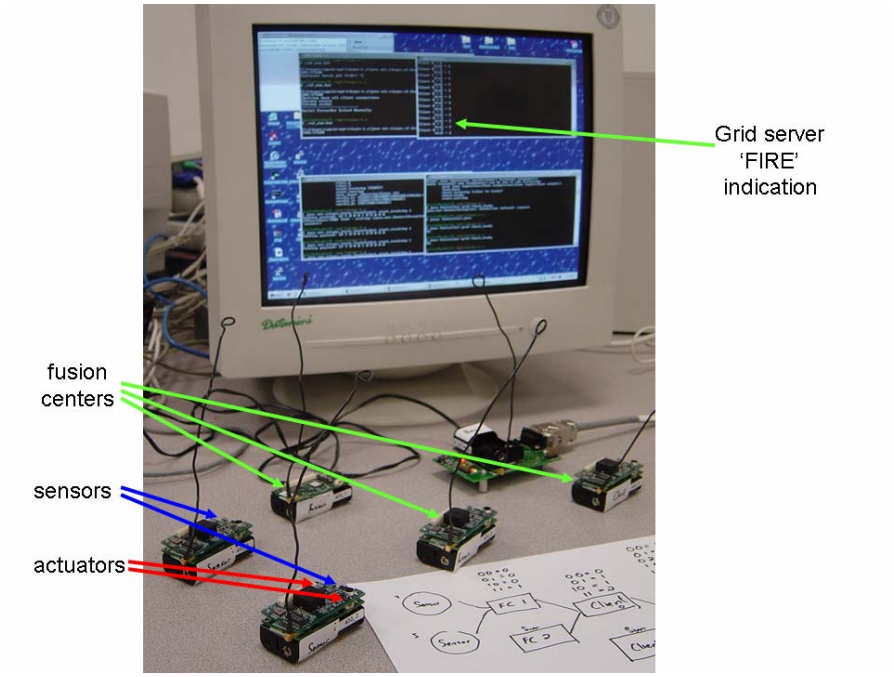


Fig. x.3 – SensorGrid for decision fusion

Sensor nodes

The sensor nodes are Crossbow MPR410 MICA2 nodes, each having a basic CPU, 433 MHz radio board and some SRAM, flash and non-volatile memory. The nodes run the TinyOS operating system which has a component-based architecture that allows modules to be 'wired' together for rapid development of applications, and an event-driven execution model designed for effective power management. The MTS310 sensor add-on board has sensing modalities such as 2-axis accelerometer, 2-axis magnetometer, light, temperature, acoustic and a sounder. We use the light and temperature sensing modalities in our application.

Grid client and grid server nodes

Components from the *Globus Toolkit* (Globus 2006) are implemented on the grid client and grid server nodes. The Globus Toolkit is an open source middleware platform which allows the sharing of computational resources and the development of grid applications. There are several primary components which make up the Globus Toolkit:
(a) *Resource Management*
Globus Resource Allocation Manager (GRAM) provides a standardized interface for resource allocation to all local resource management tools and the Resource Specification Language (RSL) is used to exchange information between the resource management components.

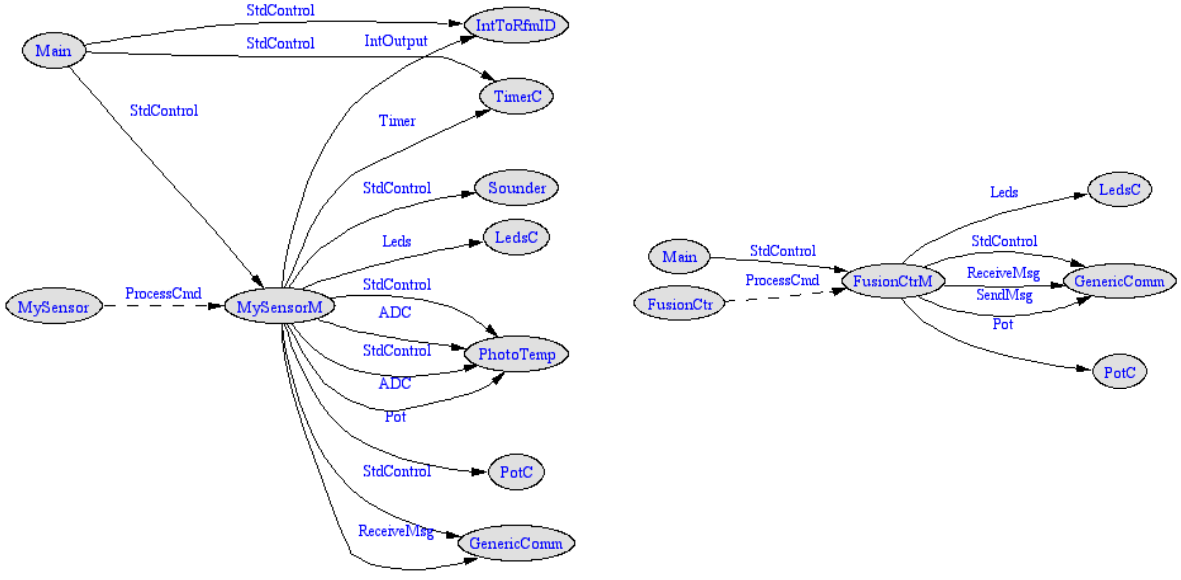
(b) *Information Services*
The Globus Metacomputing Directory Service (MDS) makes use of the Lightweight Directory Access Protocol (LDAP) as a means to query information from system components. The Grid Resources Information Service (GRIS) also provides a standardized method of querying nodes in the grid for information on their configuration, capabilities and status.
(c) *Data Management*
GridFTP, based on the popular Internet-based File Transfer Protocol (FTP), is used to perform transfer of data or allow data access.
(d) *Grid Security Infrastructure (GSI)*
Underlying the three above-mentioned modules is the GSI security protocol which provides secure communications and a 'single sign on' feature for users who use multiple resources.

The Open Grid Services Architecture (OGSA) in current versions of the Globus Toolkit enable the computational, data and storage resources on the grid to be packaged as services which are easily discovered and accessed by various users (see Section 4.1).

Implementation on SensorGrid

As mentioned earlier, our proposed sensor-grid computing approach involves more than simply connecting a sensor network and a grid computing system together. The judicious exploitation of the in-network processing capability of the SensorGrid architecture will lead to more efficient implementations of distributed algorithms, such as those for event detection and classification, in terms of the amount of processing and data transmission that needs to be carried out and the timeliness of the computed response.

The MAP classifier of equations (x.1) and (x.2) is implemented on all the leaf sensor nodes and the optimal decision fusion (ODF) algorithm of equations (x.4) and (x.5) is implemented on the sensor nodes which take on the role of fusion centers, grid clients and a grid server node.



(a) (b)

Fig. x.4 – (a) Sensor node component 'wiring'. (b) Fusion center component 'wiring'.

The 'wiring' of TinyOS components in sensor nodes performing light and temperature sensing is shown in Fig. x.4(a), while the 'wiring' of components which perform decision fusion at the sensor nodes which take on the role of fusion center is shown in Fig. x.4(b).

3.1.3 Training phase and Operation phase

Before the SensorGrid architecture can be used for event detection and classification, the MAP classifiers and fusion centers running the ODF algorithm need to be trained.

We developed a user client application which initiates the training and operation phases, as well as specifies key parameters such as number of classes, details of the particular SensorGrid architecture configuration, number of samples for each class etc.

Training is carried out in a natural manner one class at a time where a number of training samples of sensor readings under a variety of conditions corresponding to the same class are collected. A convenient method of finding the likelihood $p(x \mid C_n) = p(x \mid x \in C_n)$ is to use the Gaussian probability density function and determine the mean and covariance of the training feature vectors belonging to the same class. By using the 'equal priors' assumption, the MAP classifier shown in equation (x.2) becomes the Maximum Likelihood (ML) classifier where the decision rule is given by

$$d(x) = \arg\max_n P(x \mid C_n) \tag{0.6}$$

Equation (x.6) is implemented at every leaf sensor node to provide the local classification result.

The local classification results for every sample of each known class, i.e. with the same class label C_n, are sent to the fusion center at the next higher level which then forms the decision vector $\mathbf{d}(x) = \begin{bmatrix} d_1 & d_2 & \cdots & d_K \end{bmatrix}$ defining the region r_m, where *K* is the number of sensor nodes or lower level fusion centers reporting to a particular fusion center. As mentioned earlier, the fusion center can be a sensor node, grid client or grid server node. The counter corresponding to $\left|\{x \mid x \in r_m \cap C_n\}\right|$ is incremented for each sample presented. This process is repeated for every class at various levels of the decision fusion system until the highest level, i.e. grid server node, is reached.

At the end of the training phase, the decision fusion system is switched to the operation phase during which 'live' sensor measurements are made at regular intervals and a process similar to the training phase takes place, except that the counter corresponding to $\left|\{x \mid x \in r_m \cap C_n\}\right|$ is not incremented, but instead, equation (x.5) is used to determine the classification outcome C_{n^*} of the entire decision fusion system.

3.1.4 Experimental results

This sub-section describes a few experiments which were conducted to validate the effectiveness of the decision fusion-based event detection and classification system implemented on the SensorGrid architecture.

The environment is divided into four regions, each having a sensor node which takes on the role of a fusion center, as shown in Fig. x.2(a). The task is to accurately detect and classify different types and severities of fire occurring in the environment, as shown in Table x.1.

Class	Description
1	No fire
2	Fire in one region
3	Fire in two regions
4	Fire in all four regions

Table x.1 – 'Fire' event detection and classification

It is common practice to tabulate the results of classification experiments in an $N \times N$ *confusion matrix* whose elements $[\mathbf{CM}]_{ij} = n_{ij}$ are the number of feature vectors from class C_i classified as class C_j. Two performance metrics can be computed from the confusion matrix: (1) the probability of correct detection PD_m for class *m* is given by

$$PD_m = \frac{n_{mm}}{\sum_{j=1}^{M} n_{mj}}$$

and (2) the probability of false alarm PFA_m for class *m*, which is the probability that an event is classified as class *m* when the true underlying class is different, is given by

$$PFA_m = \frac{\sum_{k=1,k\neq m}^{M} n_{km}}{\sum_{k=1,k\neq m}^{M}\sum_{j=1}^{M} n_{kj}}$$

Class	Dec = 1	Dec = 2	Dec = 3	Dec = 4
Class = 1	20	0	0	0
Class = 2	2	18	0	0
Class = 3	2	2	16	0
Class = 4	0	4	0	16

Table x.2 – Classification results of the ODF decision fusion method on SensorGrid

From Table x.2, we see that the ODF decision fusion method yields PD = 1.0, 0.9, 0.8, 0.8 (average PD = 0.875) and PFA = 0.067, 0.1, 0.0, 0.0 (average PFA = 0.042) for the four classes.

The performance of the ODF decision fusion method is compared with the majority voting decision fusion method in which the final classification outcome is simply the highest frequency class label from among the classification decisions of local classifiers.

Class	Dec = 1	Dec = 2	Dec = 3	Dec = 4
Class = 1	20	0	0	0
Class = 2	3	17	0	0
Class = 3	1	3	16	0
Class = 4	0	3	2	15

Table x.3 – Classification results of the majority voting decision fusion method on SensorGrid

From Table x.3, we see that the majority voting method yields PD = 1.0, 0.85, 0.8, 0.75 (average PD = 0.850) and PFA = 0.067, 0.1, 0.033, 0.0 (average PFA = 0.050) for the four classes. Hence, the ODF decision fusion algorithm outperforms the majority voting method, and we have also validated the correct implementation of these algorithms on the SensorGrid architecture.

3.1.5 Other possible SensorGrid configurations

In the previous section, the SensorGrid configuration shown in Fig. x.2(a) have been used extensively. The drawback of this configuration is that, due to the limited radio range of the sensor nodes, the grid client has to be placed fairly close to the fusion centers and clusters of sensor nodes. Although multi-hop data forwarding over a series of sensor nodes can be done to extend the range of data communication, the latency between the occurrence of an event and its measurement reaching the 'sink', which is the grid and user applications in this case, is likely to be significant (Intanagonwiwat et al 2000). Furthermore, due to the limited energy available on sensor nodes, multi-hop data forwarding in a sensor network is likely to be unreliable with a high data loss rate.

To overcome these difficulties and to facilitate more computationally intensive in-network processing close to the source of the sensor data, we introduce more powerful processing and communication nodes into the SensorGrid architecture, such as: (1) the Crossbow Stargate SPB400 single board computer, which is essentially a 400 MHz Linux computer with an Intel XScale processor with CompactFlash, PCMCIA, Ethernet and USB connectivity options (e.g. a CompactFlash WiFi network adapter can be connected for high speed wireless communications), and (2) Hewlett Packard iPAQ personal digital assistants (PDAs) with WiFi connectivity. The resulting configurations of the SensorGrid architecture are shown in Fig. x.2(b) and (c).

In the SensorGrid configuration of Fig. x.2(c), two additional levels in the form of Stargate and iPAQ cluster heads have been added between the sensor nodes and grid client levels to form a *hierarchical* SensorGrid architecture in which computational and communication resources, as well as the degree of data aggregation, increases when we move up the hierarchy. An implementation of the resulting system is shown in Fig. x.5. This configuration also reduces the communication distances between the different levels, thus conserving power since radio communications over long distances consumes substantial amounts of energy in sensor networks. Instead of executing complex algorithms only at the grid, the presence of a larger amount of computational resources for in-network processing close to the source of sensor data improves the timeliness of the local and regional responses to sensed events.

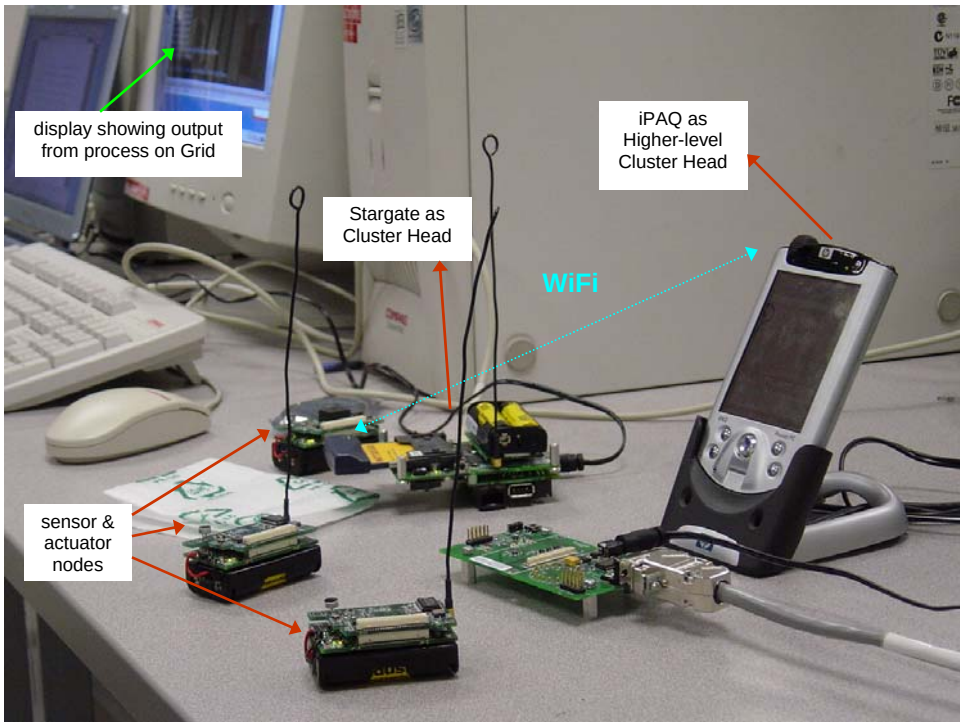


Figure x.5 – Hierarchical SensorGrid architecture

3.2 Distributed autonomous decision-making

There are many cases in which a response is needed from the sensor-grid computing system, but the best action to take in different situations or *state*s is not known in advance. This can be determined through an adaptive learning process, such as the Markov Decision Process (MDP) or reinforcement learning (RL) (Sutton and Barto 1998) approach. MDP problems can be solved off-line using methods such as policy- or value-iteration, or on-line using RL or neuro-dynamic programming (NDP) (Bersekas and Tsitsiklis 1996) methods.

We developed a multi-level distributed autonomous decision-making system and implemented it on the hierarchical SensorGrid architecture shown in Figure x.5. Basic NDP agents were implemented in Crossbow motes (Tham and Renaud 2005) at the local or ground level, and more complex NDP agents at grid server nodes were implemented at the core of the grid[3]. Each NDP agent is able to act autonomously and most parts of the sensor-grid computing system remain responsive even in the presence of communication failures due to radio jamming, router failures etc. between some components.

4. Research Issues

Sensor networks is a relatively recent field and there are many research issues pertaining to sensor networks such as energy management, coverage, localization, medium access control, routing and transport, security, as well as distributed information processing algorithms for target tracking, information fusion, inference, optimization and data aggregation.

Grid computing has been in existence longer, but nevertheless, still has a number of research challenges such as fair and efficient resource (i.e. CPU, network, storage) allocation to achieve quality of service (QoS) and high resource utilization, workflow management, the development of grid and web services for ease of discovery and access of services on the grid, and security. Resource allocation itself involves a number of aspects such as scheduling at the grid and cluster or node-levels, Service Level Agreements (SLAs) and market-based mechanisms such as pricing.

Apart from the afore-mentioned research issues in sensor networks and grid computing, sensor-grid computing and the SensorGrid architecture give rise to additional research challenges, especially when it is used in mission-critical situations. These research challenges are: web services and service discovery which work across both sensor networks and the grid, interconnection and networking, coordinated QoS mechanisms, robust and scalable distributed and hierarchical algorithms, efficient querying and self-organization and adaptation. Each of these areas will be discussed in greater detail in the following sub-sections.

4.1 Web services on grid and sensor networks

The grid is rapidly advancing towards a utility computing paradigm and is increasingly based on web services standards. The Service-Oriented Architecture (SOA) approach has become a cornerstone in many recent grid efforts. It makes sense to adopt an SOA-approach as it enables the discovery, access and sharing of the services, data, computational and communication resources in the grid by many different users.

On the SensorGrid architecture, the grid computing components are implemented as grid services using the Globus Toolkit 3 (GT3) (Globus 2006) shown in Fig. x.6, which conforms to the Open Grid Services Infrastructure (OGSI) standard.

[3] The NDP agents in the grid are implemented as grid services which are described in Section 4.1.

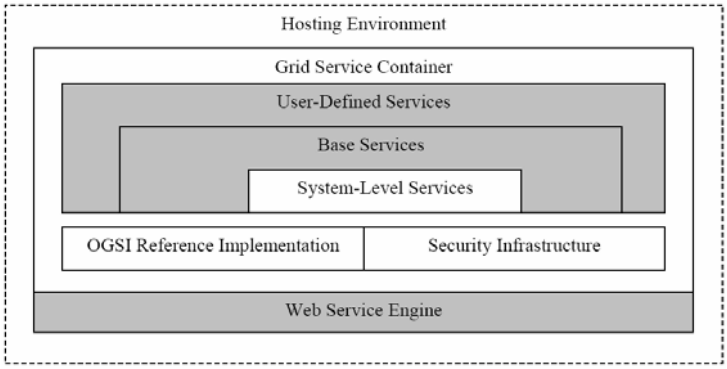


Fig. x.6 - GT3 core architecture

A grid service is basically a form of web service that can be invoked remotely for Internet-based applications with loosely coupled clients and servers, but with improved characteristics suitable for grid-based applications. In particular, a grid service is a *stateful* web service, which remembers the state of operations invoked by grid clients. This is usually the case when the grid client has a chain of operations, where the result of one operation needs to be sent to the next operation.

Another major improvement of grid services is the capability of being *transient*, as compared to normal *persistent* web services. Web services are referred to as *persistent* because their lifetime is bound to the web services container. After one client has finished using a web service, all the information stored in the web service can be accessed by subsequent clients. In fact, while one client is using the web service, another client can access the same service and potentially mix up the first client's operations. Grid services solve such problems by allowing programs to use a *factory* or *instance* approach to web services. When a client needs to create a new instance, it will communicate with a factory. These instances are *transient* since they have a limited lifetime which is not bound to the lifetime of the container of the grid services. In other words, one can create and destroy instances at will whenever they are needed, instead of having one persistent service permanently available in the case of normal web services. The actual lifecycle of an instance can vary depending on the nature of the application.

In GT3, at the server-side, the main architecture components include:
(a) Web services engine. This is provided by the Apache AXIS framework and is used to deal with normal web services behaviours, SOAP message processing, JAX-RPC (Java API for XML-based Remote Procedure Call) handlers and Web Services configuration.
(b) Globus Container framework, which provides a software container to manage the stateful web service through a unique instance handle, instance repository and life-cycle management, including service activation/passivation and soft-state management.

As shown in Fig. x.7, the GT3 container provides a pivot handler to the AXIS framework to pass the request messages to the Globus container. This container architecture is used to manage the stateful nature of web services and their life cycles.

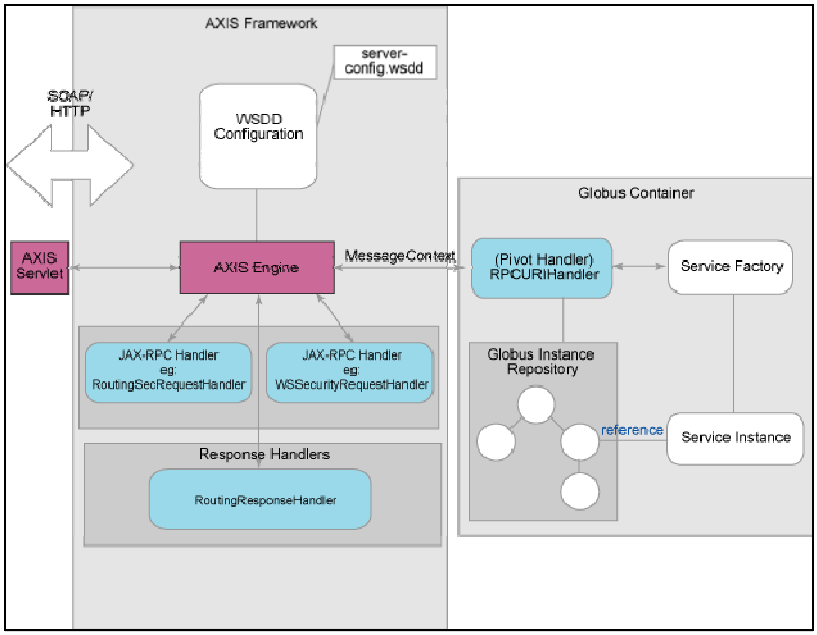


Fig. x.7 - Grid server-side framework in GT3

Referring to Fig. x.7, once the service factory creates a grid service instance, the framework creates a unique grid service handle (GSH) for that instance, and that instance is registered with the container repository. This repository holds all of the stateful service instances and is contacted by the other framework components and handlers to perform services such as: (a) identifying services and invoke methods, (b) getting/setting service properties (i.e. GSH and Grid Service Reference or GSR), (c) activating/passivating service, and (d) resolving grid service handles to reference and persist the service.

At the client-side, Globus uses the normal JAX-RPC client-side programming model and AXIS client-side framework grid service clients. Essentially, the client AXIS engine communicates to the server via SOAP messages. In addition to the normal JAX-RPC programming model, Globus provides a number of helper classes at the client side to hide the details of the OGSI client-side programming model. Fig. x.8 shows the client-side software framework used by Globus.

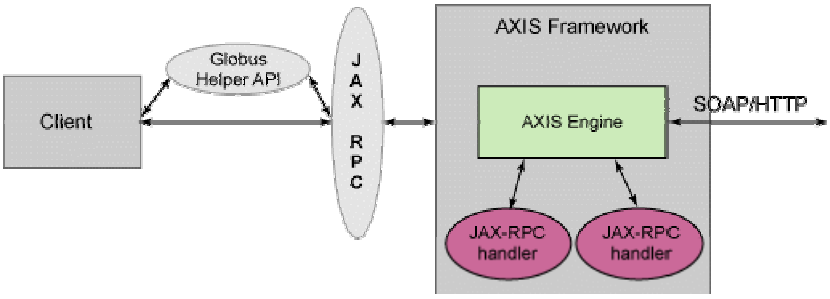


Fig. x.8 - Grid client-side framework in GT3

On the SensorGrid architecture, such as the configuration shown in Fig. x.2(b), the grid client forms the feature vector using the processed sensor readings from the Stargate embedded device. The Stargate acts as a gateway between the sensor network and the grid client and preprocesses the raw sensor readings from the sensor nodes. After that, the grid client invokes the grid service and provides it with the feature vector formed from the sensor network readings. The computationally intensive operations mentioned in Sections 1 and 3 are then performed on the grid.

Likewise, in sensor networks, it makes sense to share the sensor-actuator infrastructure among a number of different applications and users so that the environment is not swamped with an excessive number of sensor nodes, especially since these nodes are likely to interfere with one another when they communicate over the shared wireless medium and decrease the effectiveness of each node, and actuators may also take conflicting actions.

There has been some recent work on adopting an SOA and web services approach to sensors and sensor networks. The OpenGeospatial Consortium's Sensor Model Language (SensorML) standard (OGC 2006) provides the XML schema for defining the geometric, dynamic and observational characteristics of sensors. We are currently developing a web services interface between the Stargate and grid client which will enable sensor network services to be discovered and accessed by grid applications as well as by user applications running on user client devices.

4.2 Interconnection and networking

The communications and networking conditions in sensor networks and grid computing are worlds apart. In sensor networks, the emphasis is on low power wireless communications which has limited bandwidth and time-varying channel characteristics, while in grid computing, high-speed optical network interconnects are common. Thus, communications protocols for sensor-grids will have to be designed take into account this wide disparity.

ZigBee has emerged as one of the first standards-based low power wireless communications technologies for sensor networks, e.g. Crossbow MICAz motes. Furthermore, a machine-to-machine (M2M) interface between ZigBee and GPRS has recently been announced, thus enabling sensor networks to be connected to the cellular network infrastructure. One other promising development is low-rate Ultra-Wide Band (UWB) wireless technology which has several characteristics suitable for sensor networks, i.e. extremely low power consumption, reasonable communication range, and integration with UWB-based localization and positioning technology.

4.3 Coordinated QoS in large distributed system

The timeliness and correctness of computations have been studied extensively in the real-time systems community, while performance guarantees in terms of delay, loss, jitter and throughput in communication networks have been studied extensively by the networking research community. We shall refer to these as application-level and network-level QoS, respectively.

A number of QoS control mechanisms such as scheduling, admission control, buffer management and traffic regulation or shaping have been developed to achieve application-level and network-level QoS. However, all these QoS mechanisms usually relate to a particular attribute such as delay or loss, or operate at a particular router or server in the system. In order to bring about the desired system-level outcome such as meeting an end-to-end computational and communication delay requirement, these QoS mechanisms need to be coordinated instead of operating independently. We have developed a combined grid and network simulator (Sulistio et al 2005) to study these issues.

There are several methods to achieve coordinated QoS. For example, coordinated QoS can be viewed as a multi-agent Markov Decision Process (MDP) problem which can be solved using online stochastic optimal control techniques. Tham and Liu (2005) have shown that this technique can achieve end-to-end QoS in a multi-domain Differentiated Services network with multiple resource managers in a cost effective manner.

4.4 Robust and scalable distributed and hierarchical algorithms

In Section 3, we described the design and implementation of distributed information fusion and distributed autonomous decision-making algorithms on the SensorGrid architecture. Generally, it is more difficult to guarantee the optimality, correctness and convergence properties of distributed

algorithms compared to their centralized counterparts, although the distributed versions are usually more attractive from an implementation point of view.

Apart from distributed information fusion and decision-making, other current research efforts on distributed and hierarchical algorithms which are relevant to sensor-grid computing are distributed hierarchical target-tracking (Yeow et al 2005), distributed control and distributed optimization (Rabbat and Nowak 2004).

4.5 Efficient querying and data consistency

Another key area in sensor-grid computing is efficient querying of real-time information in sensor networks from grid applications and querying of grid databases by sensor network programs. It is expected that databases will be distributed and replicated at a number of places in the sensor-grid architecture to facilitate efficient storage and retrieval. Hence, the usual challenges of ensuring data consistency in distributed caches and databases would be present, with the added complexity of having to deal with a large amount of possibly redundant or erroneous real-time data from sensor networks.

4.6 Self-organization, adaptation and self-optimization

The sensor-grid computing paradigm involves close interaction with the physical world which is highly dynamic and event-driven. In addition, components in the SensorGrid architecture such as the sensor nodes are prone to failures arising from energy depletion, radio jamming and physical destruction due to bad weather conditions or tampering by people or animals.

Self-organizing and adaptive techniques are required to cope with these kind of failures as well as to respond to different unpredictable events, e.g. the computation and communication requirements at a certain part of the SensorGrid architecture may increase dramatically if an event such as a fire or an explosion happens in that region and the user demands higher resolution sensing. Since the dynamics of phenomena in the physical world are rarely known in advance, self-optimization techniques are also needed to better manage system resources and improve application-level and platform-level QoS. Techniques from the field of *autonomic computing* address some of these issues and can be applied in the SensorGrid architecture.

5. Conclusion

In this chapter, we have provided an in-depth view of the potential and challenges in sensor-grid computing and described how it can be implemented on different configurations of the SensorGrid architecture. In the near future, as sensor systems, sentient computing and ambient intelligence applications become more widespread, we expect that the need for sensor-grid computing will grow as it offers an effective and scalable method for utilizing and processing widespread sensor data in order to provide value in a number of domain areas.

Last but not least, the success of the sensor-grid computing approach will depend on the ability of the sensor network and grid computing research communities to work together to tackle the research issues highlighted above, and to ensure compatibility in the techniques, algorithms and protocols that will be developed in these areas.

Acknowledgements
The author gratefully acknowledges the contributions of Daniel B. Yagan, Wai-Leong Yeow, Leslie Tan and Jean-Christophe Renaud to the work described in this chapter.

References

Bertsekas DP, Tsitsiklis JN (1996) Neuro-Dynamic Programming, Athena Scientific, USA

Brooks R, Ramanathan P, Sayeed AK (2003) Distributed Target Classification and Tracking in Sensor Networks, In: Proceedings of IEEE, vol. 91, no. 8

Chiu K, Frey J (2005) International Workshop on Scientific Instruments and Sensors on the Grid, In conjunction with e-Science 2005, Melbourne, Australia

Duarte MF, Hu YH (2003) Optimal decision fusion for sensor network applications, In: Proceedings of First ACM Conference on Embedded Networked Sensor Systems 2003 (SenSys 2003)

Gaynor M, Moulton S, Welsh M, LaCombe E, Rowan A, Wynne J (2004) Integrating wireless sensor networks with the Grid, IEEE Internet Computing, July-August 2004

Globus (2006) The Globus Alliance, Online: http://www.globus.org

Gridbus (2005) The Gridbus Project, Online: http://www.gridbus.org

Intanagonwiwat C, Govindan R, Estrin D (2000) Directed diffusion: A scalable and robust communication paradigm for sensor networks, In: Proceedings of MOBICOM 2000

Lim HB, Teo YM, Mukherjee P, Lam VT, Wong WF, See S (2005), Sensor Grid: Integration of Wireless Sensor Networks and the Grid, In: Proceedings of IEEE Conference on Local Computer Networks (LCN'05), November 2005

OpenGeospatial Consortium (OGC) (2006) Sensor Model Language (SensorML), Online: http://vast.nsstc.uah.edu/SensorML/

Rabbat M, Nowak R (2004) Distributed optimization in sensor networks, In: Proceedings of Information Processing in Sensor Networks (IPSN)

Sulistio A, Poduval G, Buyya R, Tham CK (2005) Constructing A Grid Simulation with Differentiated Network Service Using GridSim, In: ICOMP'05 - The 2005 International Conference on Internet Computing

Sutton R, Barto A (1998) Reinforcement Learning: An Introduction, MIT Press

Tham CK, Buyya R (2005), SensorGrid: Integrating Sensor Networks and Grid Computing, Invited Paper in: CSI Communications, Special Issue on Grid Computing, Computer Society of India, July 2005

Tham CK, Liu Y (2005) Assured end-to-end QoS through adaptive marking in multi-domain differentiated services networks, Computer Communications, Special Issue on Current Areas of Interest in End-to-End QoS, Vol. 28, Issue 18, Elsevier, pp. 2009-2019.

Tham CK, Renaud JC (2005) Multi-Agent Systems on Sensor Networks: A Distributed Reinforcement Learning Approach, Invited Paper, In: Proceedings of 2nd International Conference on Intelligent Sensors, Sensor Networks and Information Processing - 2005 (ISSNIP 2005), Melbourne, Australia

Yeow WL, Tham CK, Wong LWC (2005) A Novel Target Movement Model and Efficient Tracking in Sensor Networks, In: Proceedings of IEEE VTC2005-Spring, Sweden, 29 May-1 June 2005

Biography of Author

Chen-Khong THAM is an Associate Professor at the Department of Electrical and Computer Engineering (ECE) of the National University of Singapore (NUS). He holds a joint appointment as a Lead Scientist at the Institute for Infocomm Research (I^2R) Singapore. His research interests are in coordinated quality of service (QoS) management in wired and wireless computer networks and distributed systems, and distributed decision-making and machine learning. He lectures courses in computer networks, sensor networks and real-time systems, and is the supervisor of the Computer Networks and Distributed Systems (CNDS) Laboratory at the Department of ECE, NUS. Dr Tham obtained his M.A. and Ph.D. degrees in Electrical and Information Sciences Engineering from the University of Cambridge, United Kingdom, in 1990 and 1994, respectively. He held a 2004/05 Edward Clarence Dyason Universitas21 Fellowship at the University of Melbourne, Australia.