INTRODUCTION & ARCHITECTURE

Ioannis Chatzigiannakis
Research Academic Computer Technology Institute
Patras, Greece
ichatz@cti.gr

Paul Spirakis
Research Academic Computer Technology Institute
Patras, Greece
spirakis@cti.gr

1. Introduction

The last decade we witnessed a tremendous progress towards the interconnection of the digital and physical domains, as is also evidenced by the outstanding activity in the wireless sensor networking research area and the continuous integration of sensing devices in multiple application domains. Very recently we started looking on how to expand this eco-system of embedded devices by fully integrating them to the web. The coexistence and cooperation of embedded systems with our social life is unveiling a brand new era of exciting possibilities.

Such systems are large networks of technical and techno-social nature. They comprise myriads of units/nodes with individual properties, objectives and actions. Each such artefact is unimpressive: small, with limited sensing, signal processing, and communication capabilities, and usually of limited energy. They are organized in large societies of networked tiny artefacts to accomplish tasks that are difficult or beyond the capabilities of todays conventional centralized systems. Such systems are open, in that nodes may enter or leave the collective at any time. Decision-making is distributed and possibly highly dispersed, and interaction between the units may lead to the emergence of unexpected phenomena.

These systems or societies have particular ways to achieve an appropriate level of organization and integration. This organization is achieved seamlessly and with appropriate levels of flexibility, in order to be able to accomplish their global goals and objectives. And they do this in a proactive way to meet the current or anticipated needs of their "users". For this reason, they adapt to the changes in their environment and change their internal organization by communicating, cooperating, and forming goal-driven sub-organizations.

Our envisioned systems have an identified purpose (which depends on the application). Adaptation serves this purpose. This means that sudden variations of external service requests or environmental physical conditions or of motion of network nodes does not stop the system from serving its goal. Instead the system continues to operate in a set of desired states with maintained, or gracefully degraded or even improved quality of service. Clearly, technology expects such systems to be dependable and adaptive: to the user needs, sudden changes of the environment, specific applications characteristics.

2. Features and System Overview

The unified software system presented in this book gives birth to a distributed self-organized society, a notion a bit weaker than a distributed operating system, but more suitable for populations of tiny artefacts.

Our distributed system follows basic principles of System Science. The networks are (rather complex) "systems" of small and restricted parts, that may be organized via local communication. Dynamicity (and change) is a main characteristic, of both the network and its environment. Any computation is locally restricted, and no central control exists. Thus, the networks (systems) considered are *distributed* and cooperation among the tiny parts is not automatically guaranteed. Rather, it *has to be established* (in ad-hoc situations) and *maintained*, while adapting to dynamically changing external situations.

The resulting distributed self-organized society fulfills all advantageous characteristics of distributed systems:

- *Resource sharing*. The system allows the sharing of hardware and software resources (storage, computation, sensing, actuation, bandwidth) which are associated with different artefacts on the network. The unified system provides access of shared resources to the entities requesting them. This service is operational also for devices that are note continuously connected to the rest of the network (e.g., because they apply some duty cycling mechanism, or due to their mobility patterns, etc.).

- *Openness*. The system provides efficient collaboration of hardware and software from different (non-proprietary) vendors. The new components are added and integrated in the system easily and result to an analogous increase of the system capabilities. Possible differences in data representation of interface types on different sensors (of different vendors) are suitably manipulated so that the new entities to be efficiently integrated in the system. This is mainly achieved via the generic layer for abstracting software and hardware interfaces. The system manages to incorporate together artefacts of increased computational and power capabilities with light computational entities of very limited energy power towards the accomplishment of various important common tasks.

- *Concurrency*. Several processes can operate at the same time on different artefacts on the network and they may (or may not) communicate with each other. Furthermore, a shared resource can be simultaneously accessed by more than one artefact. The system guarantees that any object that represents a shared resource is responsible for ensuring that it operates correctly in a concurrent environment. In order such an object to be safe, its operations are synchronized in such a way that its data remains consistent.

- *Scalability*. The capabilities of the system are increased by adding new resources to cope with new demands. On the other hand, a significant increase of the users demands or a decrease of available resources does not result to disanalogous decrease of system's efficiency; that is, the cost of a resource removal is reasonable, the performance loss due to the increased number of users and demands is controlled, etc. Furthermore, all important distributed, computational, communication algorithms available in the system exhibit the desirable scalable properties. They are designed exploiting suitable structures, such as hierarchical ones, so that to keep being efficient when applied in small or large scale tasks.

- *Fault tolerance*. A degraded, but of acceptable quality, service is provided when failures occur. The applications running in the system can be completed successfully, although with possibly reduced performance, but without loss of data or information, despite possible failures in certain components of the system.

  Failure events appear more frequently in this dynamic, global computing distributed environment than in common distributed systems. This is inherited by the dynamic nature of the system; users, processes, artefacts are moving, connecting and disconnecting from the system. Such events are viewed by the system as failures and their suitably managed so that the related entities or tasks are successfully and efficiently completed.

- *Transparency*. The users have completely transparent access to the resources and have no need to know anything about the distribution of the system. More analytically, the system provides transparency in various aspects: (i) *access transparency*, that is, local and remote resources are accessed using identical operations (ii) *location transparency*: users cannot tell where the hardware and software resources (CPUs, sensors, data) are located; the name of the resource does not encode the location of the resource and (iii) *migration (mobility) transparency*: resources are free to move from one location to another without having their names changed.

  All these kinds of transparencies are critical in order the system to be able to provide quality of service to the mobile and dynamic entities that are activated on it; mobile users and resources are able to perform distributed, computational, communication tasks while moving without being disturbed.

In the following section we describe how these desired properties are achieved via various services and mechanisms supported by the system.

3. System Architecture

The distributed self-organized society follows a *distributed object* architecture. Based on this architecture, there is no distinction between the *client* and the *server* processes. Each distributable entity is an object that provides services to other objects and receives services from other objects. Object communication is accomplished through a *Radio* interfaces provided by the hardware abstraction layer (see Chapter 2.0).

We employ the distributed object architecture due the important advantages that it provides. We refer to some basic of them (a) it allows the system designer to delay decisions on where and how services should be provided, (b) it is a very open system architecture that allows new resources to be added to the system as required, (c) it allows the system flexibility and scalability and (d) allows the efficient system reconfiguration during the *dynamic* migration of the objects across the network.

The system architecture proposed decomposes the global, complex and dynamic system into three complementary *layers*. This architecture is innovative in the sense that it incorporates main aspects of future networking technologies, whose combination does not necessarily exist in today's computing paradigms. It can be described as follows:

1 *Higher layer*. This is the layer where the applications and services take place. It includes the interaction between the autonomous entities that run in the system. These mechanism coordinate and manipulate the various autonomous, selfish entities (procedures) so that the system performance is maintained in the desirable level. For example, these environments determine suitable scheduling procedures for the use of shared resources by a set of competitive and selfish processes so that the system performance is maintained high.

2 *Service layer*: The intermediate layer provides a range of services that are required for the stable and fault-tolerant implementation of application at the higher layer. These mechanisms are strongly coupled with the system's lower layer's performance and availability.

This layer takes care of reaction of the system to some main external challenges that can be dynamically appearing: namely (a) track mobile objects that belong and do not belong to the society, (b) reaction to sensed data streams and (c) security threats to parts of the system.

3 *Adaptive layer*. It is the basic layer in which the system "comes together", organizes continuously, and is always ready to react. This level includes the networking infrastructures (wireless, mobile and opportunistic) and the communication protocols employed. Also, it includes the operating system(s) of the distributed system. Note that the distributed self-organized society might consists of more than on operating systems. Therefore this layer consists of a collection of components that together control the operation of the society.

The layer provides special treatment for the efficient management of mobile nodes, nodes that are duty cycling etc. Given the dynamics of the environment it adapts the topology by establishing local leaders in in neighborhoods, and dynamically creating cluster of devices (i.e., societies of tiny artefacts). Moreover it constructs an adaptive hierarchy of societies that enable scalable control of large networks of tiny artifacts.

End-to-end communication that takes advantage of the hierarchy of the nodes to deliver messages efficiently and scale to large societies and also provides delay-tolerant "connectivity", i.e., guaranteed communication between any pair of system nodes.

4. Available Services and their Functionality

The advantageous distributed characteristics of the system are achieved via a series of functionalities and mechanisms which are activated by the system as a response to various kinds of events, processes, actions, applications that take place on it. We present them in a high level. We focus on the most novel and innovative functionalities that the integrated software system is composed from:

Neighborhood Discovery. Devices periodically collect information of local topology (up to a certain distance). This component does not depend on any other component.

Leader Election & Clustering. Several nodes elect themselves as leaders based on a the available topology information. A non-leader processor that does not identify a leader within a certain hop distance k, tries to

convert itself to a leader. Leaders within k hops from each other are eliminated until there are no leaders that are within k hops or less from each other. Based on the local leaders elected, clusters of nodes are constructed.

This component maintains a central role in the integrated system and **depends only** on the NEIGHBORHOOD DISCOVERY component in order to detect local changes and adapt the infrastructure.

Hierarchy Construction. Achieved by repeated application of the Clustering algorithm. Creating an overlay network between the leaders of each cluster ("highways" construction).

This component sits on top of the LEADER ELECTION & CLUSTERING component in order to interconnect the cluster heads and establish the hierarchy in the society.

End-to-end Communication. Uses the clusters and the hierarchy of clusters to provide efficient end-to-end communication. It also takes advantage of the mobile elements of the network (e.g., roomba devices) to offer delay-tolerant and reliable connectivity.

This component depends (i) on the NEIGHBORHOOD DISCOVERY component to establish communication with a neighboring node, (ii) on the LEADER ELECTION & CLUSTERING component to establish communication with nodes that are part of the same cluster, (iii) on the HIERARCHY CONSTRUCTION component to establish communication with a distant node (i.e., nodes that is part of another cluster), and (iv) on the MOTION PLANING component in order to take advantage of mobile nodes in order to provide connectivity and establish communication with a disconnected node.

Data Aggregation. The end user can issue queries to the network that require the nodes to aggregate sensed data based on various attributes (e.g., location, group/color etc.).

This component depends (i) on the LEADER ELECTION & CLUSTERING component so that sensor values are aggregated within the cluster and (ii) on the END-TO-END COMMUNICATION component in order to propagate aggregated data throughout the society.

Group Key Establishment. A security mechanism utilizing shared symmetric keys that are updated on timely basis within each cluster or across each color.

This component depends on (i) the LEADER ELECTION & CLUSTERING component in order to establish a common key among the members of the cluster and (ii) the END-TO-END COMMUNICATION component in order to exchange messages with the other nodes.

Tracking & Private Tracking. Tracking of the location of the mobile elements that are not controlled by the network. The tracking component can be extended to offer its services in a private way. The resulting private tracking component implements security techniques that prevents tracking by outsiders.

This component depends (i) on the NEIGHBORHOOD DISCOVERY component to detect changes on the local neighborhood and track objects of interest and (ii) on the END-TO-END COMMUNICATION component in order to exchange messages with the other nodes of the society so that location information is propagated throughout the society.

Motion Planing. Provides motion strategies for navigating mobile elements that are controlled by the network to facilitate connectivity, data collection and/or affect topology.

This component **depends only** on the END-TO-END COMMUNICATION component in order to exchange messages in case it is required to coordinate the navigation of multiple mobile elements.

The various services of the system communicate and interact to each other, in order to exchange necessary information for the successful accomplishment of each of them. The communication between users, applications, processes and the various system performance and operational services is mainly accomplished via message passing. In Fig. 1 we illustrate the most representative interactions/interfaces that take place among system's services (in the figure "Ixx" stands for interface xx). Figure 2 depicts the conceptual interaction of the components to support applications (in the figure the pointed arrow $A \rightarrow B$ indicates a dependence of component A to component B).

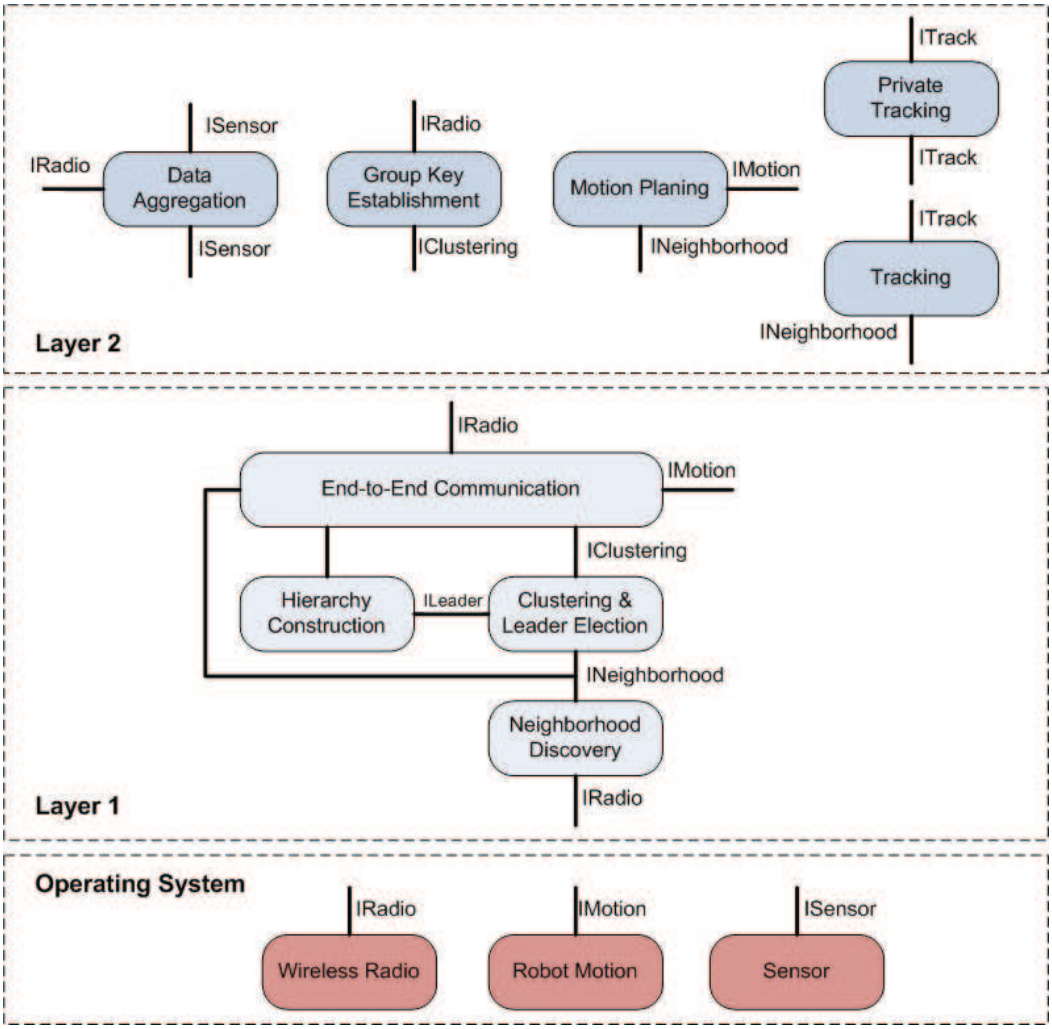

Figure 1: Relation of Components and Interfaces

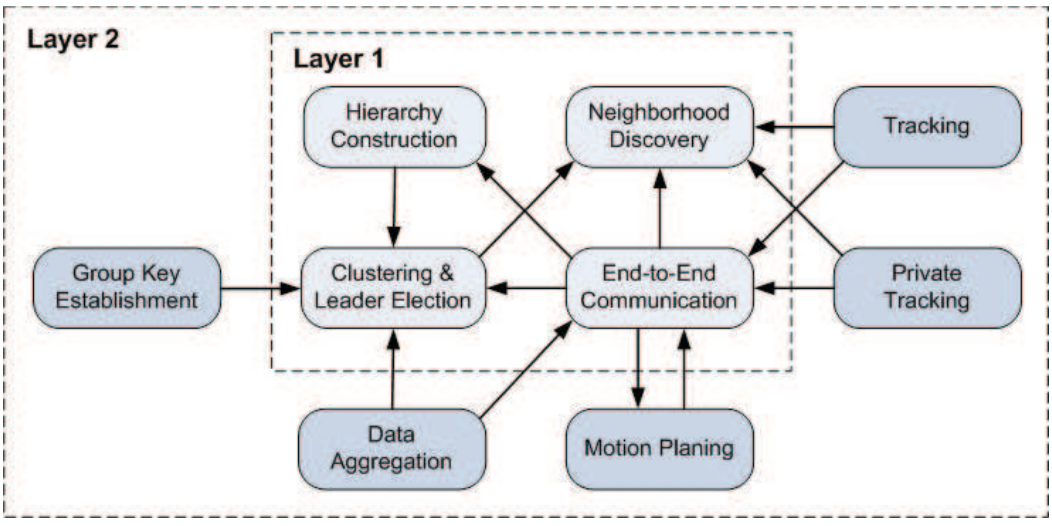

Figure 2: Component Interaction to Support Applications

The description of each component is presented in the following chapters including performance evaluations based on experiments conducted in real-world testbed deployments. We pay special attention to those characteristics of them that do not appear in the corresponding functionalities of most existing distributed systems.

5. Applications support

We now examine a few examples in order to demonstrate how the services offered by the distributed self-organized society can be combined to implement a wide spectrum of applications:

Monitoring systems of industry is a very important application for the Internet of Things. One of the most evident examples are video monitoring systems, where data gathered by quite expensive systems are not processed electronically but only occasionally viewed by human observers, who tend to overlook important events. Such systems serves more for documentation of events and are not very useful for their prevention.

The situation can be changed by systems based on small electronic artefacts. Monitoring (and tracking) *missing objects*, *misbehavior of some employee*, and *movement of people on premises* can be implemented using the TRACKING and/or PRIVATE TRACKING components.

One can also gather valuable data concerning utilization of resources and optimize the processes by identifying bottlenecks in a fast way. This also helps to avoid unnecessary redundancy of equipment. *Gathering valuable data* concerning utilization of resources and also *continuous monitoring* of real medical data of workers can be implemented using the DATA AGGREGATION component.

Monitoring traffic. The use of sensor devices on vehicles (e.g., cars, buses, tracks) allow the development of applications that offer a higher degree of accident prevention through improved driver-warning strategies, hazard detection, actuation and sensing including sensor fusion and sensor networks. It will also aim at the integration of independent safety systems and their interaction with the driver.

In cases of accidents, the network can to react to the changing physical environment by allowing the vehicles to cooperate and establish a virtual traffic light system. This distributed cooperation allows to automatically recover from the traffic jam caused by the accident and overcome the problem. This requires that the nodes are able to adapt the internal structure of the network and assign roles to particular nodes.

In such mobile networks, nodes will have to cooperate with totally untrusted partners over insecure communication mediums. The public key infrastructure must be able to react and reconfigure as the neighborhood of the node changes. At the same time, since the sensor will be monitoring the persons driving the cars, this constitute a major privacy threat as they can be potentially traced and thus be used for user profiling.

Detection of accidents and hazards can be implemented using the DATA AGGREGATION component. Propagating warnings can be implementing using the END-TO-END COMMUNICATION component. Establishing a virtual traffic light system can be implemented using the LEADER ELECTION & CLUSTERING component. Establishing cooperation of nodes with totally untrusted partners over insecure communication mediums can be implemented using the GROUP KEY ESTABLISHMENT component.

MUlti Radio Pedestrian Energy Scavenging Sensor. In [CCK+10] we examine a unifying scenario that utilizes tiny battery powered device equipped with heterogeneous sensors, that can be embedded in shoes soles and can scavenge energy from pedestrians' steps. The resulting networks of tiny artefacts can be exploited to offer innovative services. Such services can be implemented using the LEADER ELECTION & CLUSTERING, HIERARCHY CONSTRUCTION and PRIVATE TRACKING components.

Wildlife monitoring. In [BBDC+10] we present a scenario for monitoring animals in their natural habitat. A sensor network application for such a dynamic environment can be done implemented using the END-TO-END COMMUNICATION, DATA AGGREGATION, TRACKING and MOTION PLANING components.

Precision agriculture. A "classic" application scenario for wireless sensor networks can be implemented using the DATA AGGREGATION component. Can also be developed using the very modern approach of deploying one or more mobile sinks based on the MOTION PLANING component. For the cases where the future farm includes actuators for controlling the agriculture, the END-TO-END COMMUNICATION component can be used to operate the actuators.