# MiTS: Middleware for Travelling Sensor Nodes

Bart Elen, Wouter Joosen, Sam Michiels and Pierre Verbaeten
Distrinet - Department of Computer Science - K.U.Leuven
3000 Leuven, Belgium
Bart.Elen@cs.kuleuven.be

## Abstract

*In this paper, we consider a sensor network application as a composition of environments (physical and optionally logical clusters of sensor nodes) that each have a relatively static topology and sensor nodes that migrate through these environments. We call the latter travelling sensor nodes.*

*We illustrate why travelling sensor nodes are needed in many challenging sensor node applications. We sketch the requirements for the system software of these travelling sensor nodes and we outline our approach to architecting an adaptive middleware layer for such sensor nodes.*

*This paper reports on a starting project in which we start from the observation that the current operating system and virtual machines for sensor nodes do not contain all the needed support for highly mobile nodes. A specific middleware solution therefore is required. We mainly present a first inventory of the requirements and a high-level architecture of such middleware.*

## Keywords

Wireless sensor networks, middleware, mobility, adaptability and distributed networks

## 1. Introduction

Sensor networks are a relatively new technology that makes it possible to monitor a wide area, a large collection of goods or a population of people or animals. Sensor networks are built with an often large number of cooperating wireless sensor nodes. Those nodes can operate unattended and are very limited in power, computational capacity and memory. A typical sensor node is built around a low-power microcontroller running at a few MIPS with a few kilobytes of RAM. Sensor nodes can use multi hop communication to save energy and are easily utilized as an ensemble [1][2].

We define 'travelling sensor nodes' in this paper as sensor nodes that are able to leave their local environment (typically a sensor sub-network) and to join other environments, also and even if the other sensor networks are managed by a different entity.

We believe that travelling sensor nodes are required by network applications in major sectors such as the transportation sector, the automotive sector, agriculture, health care etc. We clarify this briefly.

- Imagine an application in agriculture**:** Wireless sensor nodes will be implanted in animals to collect information about their growth and health. Each time an animal is sold it may become part of the sensor network of its new owner.
- Imagine an application in health care**:** Chronically ill people who are discharged from the hospital after an operation or after a serious medical crisis will wear smart clothing that will monitor their health, that will monitor how they react on their medication and that can call for help when and if needed. Each time such a monitored patient enters a doctor's practice, for instance in a hospital, the sensor based smart clothing may become part of a local sensor network in the hospital.

An example for the transportation sector will be given in section 2.

Sensor networks that dynamically integrate travelling sensor nodes must have special properties. The environment sensor network will be one autonomic system, managed by a network manager who has to make all required network management decisions. But since the infrastructure of this network is dynamically extended by integrating sensor nodes with different owners who have put different software on these nodes, some new challenges must be addressed: travelling sensor nodes are facing both network heterogeneity, application heterogeneity and network service heterogeneity.

We further demonstrate the challenge of enabling cooperation between travelling sensor nodes and the heterogeneous environments they operate within. These sensor networks differ from each other in terms of network size, node mobility, node density, connection stability and other properties. A wide variety of applications is deployed, each requiring network services such as position determination and time synchronisation. A lot of service implementations are available for most of these services. In a short example we will further illustrate that the network service implementations required by the travelling sensor node do depend on its actual environment. This environment is determined by both the network properties of the available sensor network, the service requirements of the applications (both on the hosting network as well as on the travelling sensor node) and by the availability of network service implementations for each service.

The sensor nodes on state-of-the-art sensor networks do not face the same heterogeneity because they only have to function in one specific environment. Such sensor nodes can be programmed with all service implementations that are needed for that specific environment. This is different for travelling sensor nodes that will join multiple sensor networks and will be confronted with a high level of heterogeneity. The actual state-of practice would require travelling nodes to be reprogrammed when migrating into another environment. There are two problems with this current (state-of-the-art) solution. First, reprogramming requires user interaction. Secondly, a sensor network ad-

ministrator will often not be authorized to program visiting travelling sensor nodes. In our opinion, a much better solution would be based on travelling sensor node that self-adapt to the changing environment.

We therefore argue that a middleware solution is needed to adapt the network services on the travelling sensor node to the current environment of the node. This middleware will hide the heterogeneity of the environment for the applications. This approach enables cooperation between the sensor nodes from different networks. We are working on a new, adaptive middleware called MiTS (Middleware for Travelling Sensor nodes) that fulfils these requirements. In section 2 of this paper, we present the example mentioned above. This way, we elaborate on the requirements for MiTS. Section 3 enumerates the major functionalities of MiTS. In section 4, we compare our findings with the state-of-the-art in sensor node system software and we overview a number of other open problems that we need to tackle moving forward. We summarize in section 5.

## 2. Requirements from a real-world example

Large containers that are used for global transport will be equipped with sensor nodes which collect information about the container: location, content, status etc. In a multi-model transportation environment, containers will migrate between harbours, ships, trains and trucks. This clearly exemplifies the concept of a changing environment – which we assume to be equipped with sensor network technology as well. We will use the container transport application to clarify the key properties and challenges that do arise when supporting travelling sensor nodes. In the next paragraphs, we briefly discuss the kind of mobility of a travelling sensor node, the heterogeneity of the environments it must deal with and the adaptability it requires from the middleware.

### Mobility

The large containers that are used for global transport encounter warehouses, ports, trucks, boats and trains while travelling to their destination. All those locations can be equipped with a local sensor network to collect information about the containers. Each time the container arrives at a new location, its sensor nodes migrate to a new environment.

### Heterogeneity

Travelling sensor nodes do encounter a lot of heterogeneity. The various applications, the available network services and the network properties can differ from sensor network to sensor network.

#### *Network heterogeneity*

In the container sensor network case is there the sensor network of a port which can contain tens of thousands of containers while the sensor network of a train will involve substantially less containers. In warehouses, containers will be moved, while containers typically stay immobile on a boat. Another example is the node density which may vary with the environment.

#### *Application heterogeneity*

A wide variety of applications run on sensor nodes. One application may identify the products inside the container with the help of RFID [3][4] while another application can be used to localize 'lost' containers in ports [5]. Other examples of applications for container sensor networks are protecting containers against unauthorized access [6] and checking that the container did not become too hot, cold or humid and whether is has been dropped or bumped against [3].

A lot of those applications use network services such as:

- Route discovery: to discover the correct paths on the sensor network.
- Position determination service: to find the current location of the container.
- Flooding service: to flood alarm messages and queries over the network.
- Time synchronization service: to make it possible to order events on the sensor network.
- Election service: for electing the sensor node that has to do a task or receives a certain responsibility.

#### *Network service heterogeneity*

A lot of different implementations are available for most network services on sensor networks. Examples of this are LEECH and TEEN for route discovery [7], simple flooding and SBA for flooding [8], APS and SPA for location determination [9], and RBS and GPS for time synchronization [10]. There is currently no standardization for those network services on sensor networks and our experience with fixed networks and ad hoc networks indicates that there will always be multiple network service implementations for those network services.

### Service implementation selection

Which network service implementations are best fitted for installation on the sensor node depends heavily on the current environment of the node. As an example will we take a closer look at the dependency of the routing discovery services on the sensor networks.

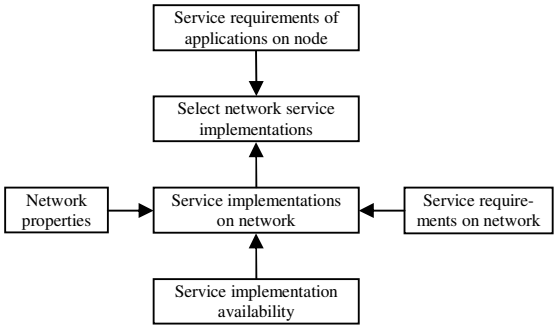


**Figure 1. Service implementation selection**

On the small sensor network of a train, a simple route discovery service such as DSDV [11] can be used. The much larger sensor networks of ports require a better scalable route discovery service which, as stated in [12], can be realised with a hierarchical route discovery service such as LEACH [13]. If an application is deployed on the port's sensor network that does require QoS, a route dis-

covery service with QoS support such as SPEED [14] will be preferred. The network service that is used on the network will only be installed on the travelling sensor node when its applications require it.

The middleware must be self-adaptive. The service implementations that it contains must be adapted to its environment This environment is determined by both the service requirements of the applications on the node and by the service implementations that are used on the network (Figure 1). The service implementations on the network are determined by both the network properties, the service requirements of the applications on the network and by the availability of service implementations.

## 3. Platform Support

We argue that cooperation can be made possible between travelling sensor nodes and environments of different sensor networks by hiding the heterogeneity of the environment for the applications with adaptive middleware. We outline our approach to architecting the MiTS middleware which must address the following deployment tasks:

- First, it will detect the **service requirements** of its applications.
- It will **discover** the **network service implementations** used on the local sensor network.
- Then will it **adapt** the **network services** running on the nodes to its current environment by installing, removing or replacing services implementations on the node.
- Fourth will it **map** the **abstract services** that the applications are using on the best fitted service implementations.

Figure 2 gives an overview of the high-level architecture of the MiTS middleware. The middleware is component-based to support adaptability. The key tasks will be described along with the presentation of the main components of the middleware.

Applications must be able to operate using different service implementations. Therefore they must be programmed independently from specific services. The applications will therefore use **abstract services** which will be mapped at runtime on concrete service implementations. The middleware architecture contains a number of network services and a middleware core which consists of two network managers (figure 2): the service manager that is an interoperability layer between application logic and environment specific networks services, and the service component manager, that enables dynamic replacement of service implementations on a travelling sensor node.

### Service manager

The service manager has several responsibilities. The first is that it must collect information about the service requirements of the applications on the node. The second responsibility is that it must gather information about the services offered by the service implementations on the sensor nodes of the environment. The third key responsibility is mapping the abstract services that the applications are using on the most appropriate service implementations that are available. All gathered information is stored in the **Service DB.** The service manager can be configured itself at runtime by adjusting the **Service selector policy**.

### Service component manager

The service component manager is responsible for removing, installing and replacing network service implementations on the node and for detecting which network service implementations are used on the local sensor network. It needs the support of two services: the Service discovery and the Component exchange service (Figure 2).

The behaviour of the Services component manager can be

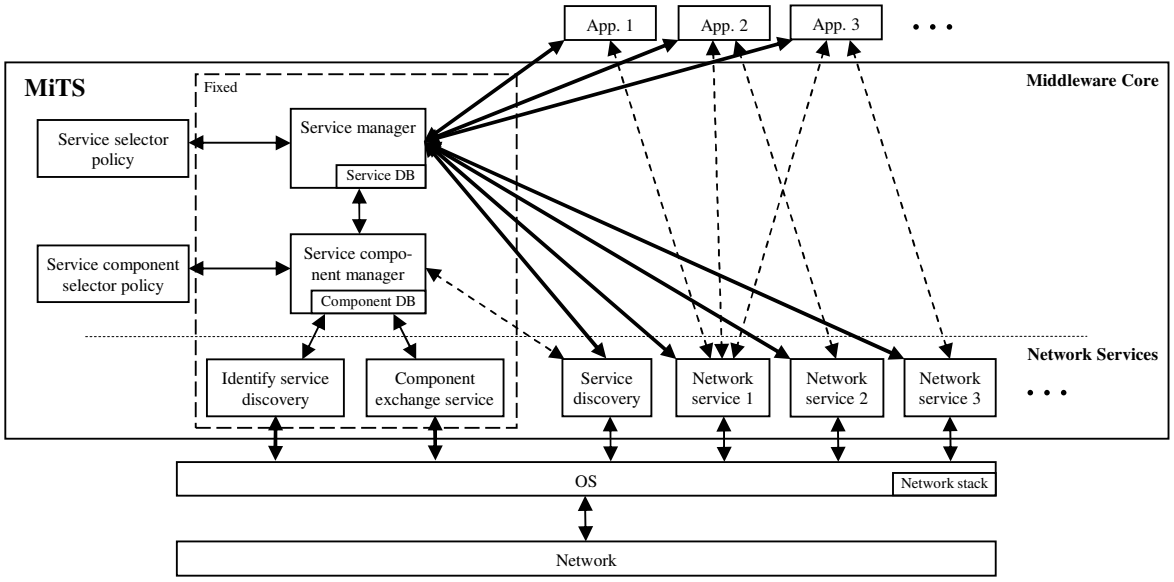


**Figure 2. High level architecture**

changed at runtime by replacing the **Service component selector policy.** The service component manager will store all gathered information about the network services on the travelling node in the **Component DB.**

### Service discovery

The service discovery consists of two components. The first component is responsible for initially **identifying** the **service discovery** used on the network. This can typically be achieved by simply asking any neighbouring node since all nodes on the network have to be equipped with this network discovery service.

The second component is the **Service discovery** component which has the task of discovering all the required service implementations on the network

### Component exchange service

The number of services which can be stored on a sensor node is relatively limited because of the very restricted hardware of sensor nodes. Whenever a sensor node requires a network service which is currently not available on the node, it will use the component exchange service to download the necessary functionality from another sensor node in its neighbourhood. The number of hops involved must be minimized. The proposed solution requires that the sensor nodes on the network are equipped with exactly the same component exchange service.

### Standardization

A certain amount of conventions for interoperation are required to enable the migration of travelling sensor between different environments. But, all aspects of the middleware that are standardized by such conventions can no longer be changed for the purpose of creating new applications. This may limit the room for innovation and the flexibility that the middleware offers. Hence we have limited such conventions to the minimal but sufficient. The only standardized aspects of our middleware are the interfaces of the network protocols of the Identify service discovery and the Component exchange service, as well as the abstract service definitions that the applications and middleware are using and the network services are implementing.

## 4. Related Work

In this section will we explain why the state-of-the-art operating systems, virtual machines, and middleware do not offer the support for travelling sensor nodes as discussed in this paper.

### Operating Systems

So far, sensor node operating systems have been focused on achieving high performance at minimal cost.

#### *TinyOS*

TinyOS [15] is currently the most wide-spread operating system for wireless sensor nodes. TinyOS is designed for high performance. It is not possible to execute multiple applications simultaneous on this OS. In that sense is TinyOS very static, it does not support adding or removing a service at runtime.

#### *SOS*

SOS [16] is a more dynamic operating system for sensor nodes. It does support the simultaneous execution of multiple applications and it allows the installation and removal of applications at runtime. SOS does even contain a module exchange service which installs all new software of the sensor nodes in the neighbourhood on the node. However, SOS does not support a self-managed modification of service software as targeted by MiTS.

### Virtual machines

Virtual machines have been created with objectives that are comparable to the design objectives of operating systems: for instance to limit resource consumption.

Sending an application over a wireless connection costs a lot of energy. Sending a single bit can consume the same energy as executing 1000 instruction [17]. Recent research has been addressing the support for creating applications with a small footprint, e.g. by building an application specific virtual machine with for instance Maté [17].

### Middleware

The former paragraphs show that the community addressing system software for sensor nodes has not (yet) been addressing the flexibility we are aiming for. Obviously, research on flexible middleware can offer relevant solutions to guide our research. We cannot be exhaustive in enumerating relevant work in this space. We briefly discuss two key research results that share some of our objectives.

#### *ReMMoC*

ReMMoC [18] is a middleware platform that dynamically adapts both its discovery and binding protocol (e.g. RMI or publish-subscribe) to allow interoperation with heterogeneous services in mobile environments. This middleware uses reflection and WSDL which are well fitted for PDA's but are too heavy to be used in sensor node software.

#### *INDISS*

INDISS [19] is middleware that does deliver service protocol interoperability in the mobile environment. This middleware makes it possible to exchange service discovery information between applications that have been developed for different service discovery protocols. It realizes this by translating all incoming service discovery packets to the protocol the application does support. Each application running on INDISS must contain its own service discovery implementation. This makes the usage of existing applications possible but it also makes the software system unneeded large.

### More Open Problems

Before a solution for travelling sensor nodes can be deployed in the real world, several remaining management and security challenges need to be solved as well. We give a short overview of the most important additional challenges that we anticipate.

Management challenges:

- **One entity:** It must still be possible to approach the complete sensor network as one entity.
- **Border detection:** The sensor node must be able to detect when it crosses the border of a sensor network. If a container is for instance loaded on a boat then it will have to detect that it must leave the port's sensor network, even if it still can have connectivity with it.

Security challenges:

- **Network identification:** The node must be able to verify whether it has joined the right sensor network.
- **Trusted network services:** The sensor node must be able to trust each network service that it installs.
- **Authentication:** Only authorized nodes may be able to collect confidential information.

Our current focus is on refining the architecture introduced in section 3, while targeting a scalable implementation with acceptable footprint.

## 5. Summary

In this paper, we discuss sensor network applications that are a composition of (1) environments (of sensor nodes) that each have a relatively static topology and (2) travelling sensor nodes that migrate through these environments.

We have used a container transport application case study to illustrate the system software requirements of travelling sensor nodes as these face a lot of heterogeneity in their environment. This heterogeneity is caused by the network properties, the application requirements and the available service implementations. We have shown that the combination of the mobility of the travelling sensor node and the heterogeneity of its environment make it difficult for the travelling sensor node to cooperate with sensor nodes in the environment.

We have argued that cooperation can be made possible between the sensor nodes of different networks by hiding the heterogeneity of the environment of the node with self-adaptive middleware. We are working on a new middleware called MiTS that will allow sensor nodes to travel to different sensor networks by hiding all heterogeneity for the applications on the node.

Travelling sensor nodes – as defined here – currently are a relatively new research area. The middleware solution sketched in this paper is making the first step towards allowing sensor nodes to migrate. This enables sensor nodes from different sensor networks to cooperate in a heterogeneous environment.

## REFERENCES


[1] I. F. Akyildiz and W. Su and Y. Sankarasubramaniam and E. Cayirci, Wireless sensor networks: a survey, Computer Networks, vol. 38, no. 4, March 2002, pp. 393-422

[2] D. Culler and W. Hong, Wireless sensor networks, Communications of the ACM, vol. 47, no. 6, June 2004, pp 32-33

[3] D. Culler and H. Mulder, Sensor Nets / RFID, http://www.intel.com/research/exploratory/smartnetworks.htm, 2004

[4] Bulldog Technologies Releases RFID and Sensor Network Products, Frontline, http://www.frontlinetoday.com/frontline/article/articleDetail.jsp?id=153264, March 2005

[5] E. H. Callaway, Wireless sensor networks, ISBN 0-8493-1823-8, 2004

[6] http://www.bulldog-tech.com

[7] K. Akkaya and M. Younis, A survey on routing protocols for wireless sensor networks, Elsevier Ad Hoc Networks, vol. 3, no. 3, 2005, pp. 325-349

[8] B. Williams and T. Camp, Comparison of broadcasting techniques for mobile ad hoc networks, Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC), 2002, pp. 194-205

[9] D. Niculescu, Positioning in ad hoc sensor networks, IEEE Network, July/August 2004

[10] J. Elson, K. Römer, Wireless sensor networks: a new regime for time synchronization, ACM SIGCOMM Computer Communication Review, vol. 33, no. 1, January 2003, pp. 149–154

[11] C. E. Perkins and P. Bhagwat, Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers, SIGCOMM '94, August 1994, pp. 234-244

[12] K. Akkaya and M. Younis, A survey on routing protocols for wireless sensor networks, Elsevier: Ad Hoc Networks, 2004

[13] W. Heinzelman, A. Chandrakasan, H. Balakrishnan, Energy-efficient communication protocol for wireless sensor networks, Proceedings of the Hawaii International Conference on System Sciences, January 2000

[14] T. He, J. Stankovic, C. Lu and T. Abdelzaher, SPEED: a stateless protocol for real-time communication in sensor networks, Proceedings of International Conference on Distributed Computing Systems, Providence, RI, May 2003

[15] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler and K. S. J. Pister, System architecture directions for networked sensors, Architectural Support for Programming Languages and Operating Systems, 2000, pp. 93-104

[16] C. Han, R. K. Rengaswamy, R. Shea, E. Kohler and M. Srivastava, SOS*:* A dynamic operating system for sensor networks, Proceedings of the Third International Conference on Mobile Systems, Applications and Services (Mobisys), 2005

[17] P. Levis and D. Culler, Maté: a tiny virtual machine for sensor networks, Architectural Support for Programming languages and Operating Systems, San Jose, October 2002

[18] P. Grace, G. S. Blair and S Samuel, ReMMoC: a reflective middleware to support mobile client interoperability, Proceedings of International Symposium on Distributed Objects and Applications(DOA), November 2003, pp. 1170-1187

[19] Y. Bromberg and V. Issarny, Service discovery protocol interoperability in the mobile environment, SEM, 2004, pp. 64-77