A3ME - an Agent-based Middleware Approach for Mixed Mode Environments

Arthur Herzog, Daniel Jacobi, Alejandro Buchmann
Technische Universität Darmstadt, Germany
{aherzog, jacobi, buchmann}@dvs.tu-darmstadt.de

Abstract

In this paper we introduce a new agent-based approach for middleware in mixed mode environments. Mixed mode environments have different dimensions of heterogeneity: devices with widely differing capabilities, different software, and different communication technologies. To deal with these, each node in the network is seen as an independent entity: a device agent. These device agents are abstractions of the network nodes and they offer services corresponding to nodes' capabilities and also use services offered by other agents. Interoperability is achieved through an agent interaction interface. For the description of device types, capabilities and services a predefined but extensible ontology is used. This approach combines and adopts results from the areas of multi agent systems and heterogeneous ad hoc networks.

1 Introduction

The term Mixed Mode Environments (MME) describes environments with different dimensions of heterogeneity. First of all, MME refers to networks composed of very different kinds of devices. These are distributed among various physical environments and communicate with each other using different communication technologies. Individual nodes in the network can be sensors, actuators, robots, Unmanned Vehicles (UV), Human Interface Devices (HID), mission control stations or large servers. All these devices have their specific capabilities and constraints, are manufactured by different companies and use different software and operating systems. Some nodes do not even have an operating system. The physical environment can also be heterogeneous like indoors, outdoors, underground, underwater, etc. These devices can communicate with each other in different ways: by wire, radio, infrared, light, sound or through other media. For each of these communication media, many different communication technologies exist, which use different protocols, frequencies, encoding schemes, etc.

In MME different areas are merging together: Wireless Sensor Networks (WSNs), Wireless Sensor and Actor Networks (WSANs) and Ubiquitous Computing. Most research in the WSN area up to now is done in homogeneous WSNs using the same kind of device for all nodes in the network or with a more powerful device that is used as a gateway and data sink. In WSANs an additional type of nodes - actors - is used. They are capable of interacting with the physical world. In ubiquitous computing multiple devices in people's surroundings are performing tasks without people necessarily interacting with the devices. Here a broad variety of devices are involved like media devices, mobile phones, light and temperature control devices, etc.

Today, application developers must deal with this heterogeneity when developing a new application for heterogeneous networks. Each time a new kind of node appears in the network, the applications have to be adjusted to deal with the new hardware. Middleware is a way to avoid this direct interaction of applications with the hardware and software of the devices, and to enable and simplify the interoperability among devices. Middleware abstracts over all the devices and communication technologies, and offers the applications well defined interfaces to interact with other nodes.

Our aim is to enable interoperability among different nodes in MME without the need of adjustments each time new hardware is introduced. The agent-based approach offers an abstraction for the different hardware: it sees all the different nodes in the network as independent entities, which we call device agents. Each device agent knows its capabilities (and constraints). Depending on its capabilities, a device agent can offer services to other agents and can also perform tasks, sometimes using the services of other agents. The complexity of agents representing, for example, a small sensor node or an UV can vary considerably. Thus, a sensor agent might be only capable of measuring the current temperature and sending it to a receiver interested in this data,

[0]Supported by the DFG Research Group 1362, Cooperative, Adaptive and Responsive Monitoring in Mixed Mode Environments

whereas the UV agent can not only move through the environment, but also collect the data from the sensor agents, aggregate and evaluate the collected data, and use this information further in its decision making process. Communication between different nodes is enabled by using a uniform message structure and by defining basic interaction mechanisms.

In Section 2 we discuss the challenges we have to deal with in heterogeneous networks and in 3 we give a short overview of related work. In Section 4 we present how we intend to solve the described challenges using an agent-based approach. Section 5 contains conclusions and our directions for future work.

2 Challenges for Middleware in Mixed Mode Environments

2.1 Device Heterogeneity

The heterogeneity of devices available in MME introduces additional complexity to the middleware. The middleware needs an abstraction for all the different devices. This abstraction must be able to describe itself and its capabilities to other nodes in the network. Since, theoretically, there can be an unlimited number of different devices in MME, a predefined basic classification of devices is needed, to which all the devices can be assigned, in order to allow a classification based device discovery and recognition. Capabilities of the devices can be offered as services. To handle the huge variety of capabilities and services basic, i.e. minimalistic, classifications must be defined.

Once we have a description language and a classification for devices and their services, we also need mechanisms for their discovery and use. Since a network in MME can not rely on the constant availability of individual nodes, all these mechanisms must work in a distributed way without a central coordinating instance.

2.2 Communication Heterogeneity

The second kind of heterogeneity in MME is reflected by the variety of communication technologies used. This means there are different protocols and standards for different transmission media. Consequently, the properties of communication are quite diverse in terms of bandwidth, reliability, communication range, etc.

To deal with this communication heterogeneity, abstract mechanisms for communication must be introduced which are independent of the individual communication technologies. Here we can distinguish between message oriented and stream oriented communication. The message and stream structures and encoding schemes should be specified independently of the underlying communication media and technologies.

2.3 Dynamic Multi-hop Network

The physical environment in MME can be harsh, so some nodes in the network may fail or lose their connectivity. Additionally, nodes can join and leave the network and some nodes may be mobile within the network. Therefore, the devices as nodes in a network can appear and disappear at any time. Furthermore, the presence of individual nodes, their positions, and capabilities are not always known in advance. This means that mechanisms for ad hoc network setup, self healing and self configuration must be defined.

3 Related Work

A variety of different middleware solutions already exists. Henricksen et al. [10] give an overview of the different approaches for middleware for WSNs. They consider 4 different categories: database-inspired approaches (e.g. TinyDB [12]), tuple-space approaches (e.g. TinyLIME [4]), event-based approaches, and service discovery based approaches. Not included in this categorization are virtual machine approaches like Mate [11] and Agilla [5]. Another overview of middleware in WSNs is given in [9]. These two survey papers provide an overview over the existing middleware approaches in WSN, but to the best of our knowledge none of these is designed to work in MME with its different dimensions of heterogeneity.

Mundo architecture [13] is an approach in Ubiquitous Computing which deals with heterogeneous devices. It classifies the devices into five groups, according to their roles. All communication here is based on publish/subscribe scheme and is bundled into MundoCore communication middleware [1].

BASE [2] is a middleware which uses a minimal kernel. In BASE different transport protocols, services and hardware capabilities can be added as plug-ins. All intra and inter node interactions a represented as invocation objects which can be transmitted over different communication channels.

4 Agent-Based Middleware Approach

The Agent-based Middleware approach for Mixed Mode Environments (A3ME) aims at offering a solution to deal with the challenges described above. The idea is to introduce a new abstraction for the different devices used and consider each of them just as a separate agent which knows its capabilities, and offers and uses services. The capabilities depend on the hardware and software environment of

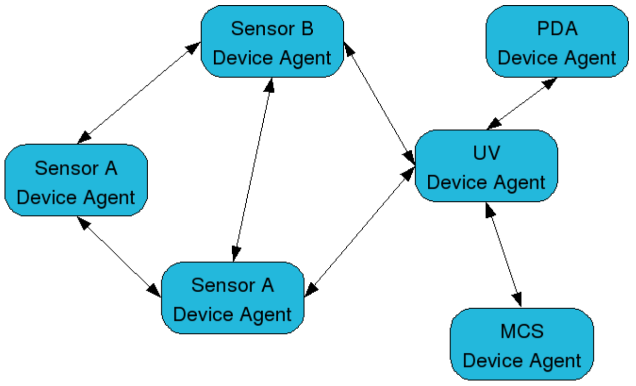

Figure 1. Network nodes as device agents.

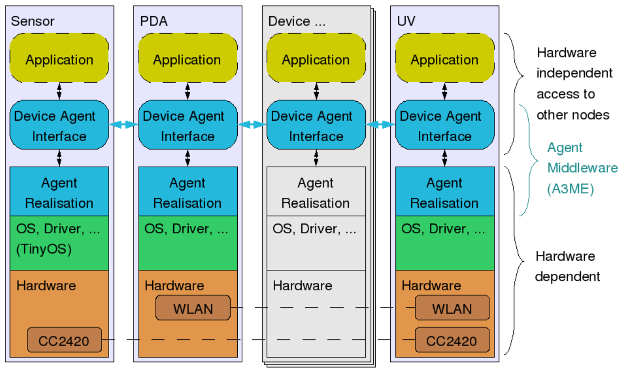

Figure 2. A3ME Diagram.

the individual nodes (Figure 1). Some of the nodes can also perform tasks by communicating and cooperating with other agents in the network.

To explicitly distinguish our agents from the different kind of agents used in Computer Science, we call the agents in this approach *device agents*. Each of the nodes is seen as a black box, in the sense of hiding the hardware and software the node is built of, and just showing its capabilities to the outside instead. To interact with each other, device agents have to support a basic set of messages and interaction protocols, which will be described in more detail later.

In contrast to existing approaches, especially in the area of WSN, we treat each node as an independent entity, which knows its capabilities and is capable to function on its own. These entities - represented by device agents - interact with each other to build a network and to enable higher level services and capabilities. This agent-based approach facilitates the self organization and adaptability of the system.

When we speak of agents, we mean representations for physical entities such as single devices which communicate with the network in some way. To do so, a device agent may have to cooperate with other device agents. Our understanding of an agent differs significantly from that used in current WSN projects like Agilla, described in [5]. In Agilla, an agent represents code which can move between devices and continue its execution. In our paper, agent means an abstraction of a network node.

Because of the nature of an agent-based system this approach facilitates a self organizing and adaptive architecture: Device agents representing physical nodes discover their environment, find other device agents and start interacting with them when needed. This architecture is adaptive because when the neighborhood changes, either after the disappearance of nodes or while the node is moving through the environment, the device agents just have to rediscover the environment by requesting corresponding services.

Before describing the system let us look at a simple example: in the case of fire alert in an office building, firefighters would like to know where in the building the fire is located. The office building contains a number of sensors. Here at least two kinds of nodes would be available: temperature sensors spread throughout the building and Personal Digital Assistant devices (PDAs) used by firefighters to obtain the data. Device agents representing the sensors (`sensor-agents`) have the capability to sense and send the temperature at their location on demand. Second type of device agent is running on the PDAs (`pda-agent`). This `pda-agent` has to collect temperature measurements in an area and show these to their users. To do so, PDA-agents have to discover other agents which can measure temperature, and request them to send the temperature data. As soon as the temperature is received, `pda-agents` visualize it to their users.

4.1 Middleware Overview

In the general case we have many different kinds of device agents. All these agents must provide an Agent Interaction Interface (AII). The AII defines a basic set of interaction protocols and the message structure to use independent of the hardware and communication technology. The implementation of AII may be different for each type of device, but it should provide all basic hardware dependent capabilities of the current device and offer them as services. Additionally each device can have one or multiple applications running on it. Applications interact with other nodes in the network through the AII. An application can also define tasks for the agent and additional hardware independent services like aggregation functions. These hardware independent services are also offered through AII (Figure 2). The services offered can be publicly accessible by other agents, restricted for use by only some agents or by authentication or they can be private, what means usable by local applications only.

Let's use the firefighter example to explain it. In our example we introduced two kinds of devices: sensors and

PDAs. For sensors we can for example use TinyOS to implement the AII and to offer all the hardware capabilities this sensor has as services. In this case the capabilities are temperature sensing and radio communication. For temperature sensing, a service `getTemperature` is offered, which sends the requester a message with sensors coordinates and the current sensor value. For the radio communication capability `sendMsg` and `forwardMsg` services are offered. The `sendMsg` service is offered as private service. The PDA has output capability for the user and also the radio communication capability. These capabilities are implemented as private services only to enable the firefighter application sending and receiving messages and accessing the display to show the collected information.

To realize the AII, first of all agents must be able to communicate with each other. For this we use an *Agent Communication Language* (ACL). To enable agents to publish, discover and use *services*, AII defines the basic mechanisms (services). For describing these services, the data and the tasks we also need a *content representation* schema.

4.2 Agent Interaction Interface

The first part of the AII is an *Agent Communication Module* (ACM). It is responsible for the abstraction of the physical communication. It offers the communication interfaces to other higher level modules of the device agent. Two basic kinds of communication are: via messages or via streams. In this paper we concentrate on message based communication, but we keep in mind to allow stream based communication too.

Since we have to deal with very different devices regarding resource constrains and communication capabilities, we have to orientate on the most constrained devices - small sensors - when designing the communication language. Other less constrained devices then still will be able to use this language. This doesn't prevent non resource constrained devices to still use the more powerful communication languages among each other.

The different nodes in the network must be able to interact with each other. For this they must be able to understand each other - what means here they must be able to understand the messages exchanged. This can be achieved by using a Common Message Structure (CMS). Additionally we have to define basic *Agent Interaction Protocols* (AIP) to allow a node to announce itself when it joins the network, to discover other devices, to search and request services, etc. The ACM, CMS and AIP together define an *Agent Interaction Interface* for devices in MME.

Back to our example; when the fireman activates his PDA, the `pda-agent` sends a message requesting the devices in a specified range for sending their capabilities description. As answers to this request arrive, the available

Table 1. FIPA ACL Message structure

```
(MessageType
    :sender AgentIdentifier
    :receiver AgentIdentifierSet
    :content String
    :reply-with Expression
    :reply-by DateTime
    :in-reply-to Expression
    :reply-to AgentIdentifierSet
    :language Expression
    :encoding Expression
    :ontology Expression
    :protocol Word
    :conversation-id Expression
    :UserDefinedParameter Expression
)
```

devices in range are displayed to the fireman. As the fireman sees there are temperature sensors in range he can order the `pda-agent` to request the temperature values. Once the `pda-agent` gets the requested temperature values it can display them on a map using the location coordinates, which the `sensor-agents` provided it with.

4.2.1 Common Message Structure

The different nodes must use the same message structure and encoding among different nodes and communication techniques, which we call Common Message Structure. The Foundation for Intelligent Physical Agents (FIPA) [8] has defined such a generic message structure in [7], shown in Table 1. To keep the messages small, we don't use all the parameters specified for FIPA ACL Messages, but only a few of them: message type, sender, receiver and content.

For the message type we use FIPA Communicative Acts (CA). In FIPA ACL each message is seen as a communicative act which has a specified intention. FIPA ACL defines 22 CAs - generic types of messages (see Table 2). Usage of these generic message types gives the advantage to know the general intention the message is representing, even if the content of the message can not be understood. So a node can for example answer with a `NOT UNDERSTOOD` message when ever it can not understand the message content or can't deal with this kind of message. The FIPA ACL defines also a bit efficient coding for messages we can use here [6].

For identification of sender and receiver we use a combination of two IDs: a local ID and a group ID. These two together would build the global ID. Group ID can be used for an area, a cluster or just for a group of agents. By setting local ID to zero a whole group can be addressed. The content part of the message will be discussed in 4.3.

Table 2. FIPA Communicative Acts

1	Accept Proposal	12	Propagate
2	Agree	13	Propose
3	Cancel	14	Proxy
4	Call for Proposal	15	Query If
5	Confirm	16	Query Ref
6	Disconfirm	17	Refuse
7	Failure	18	Reject Proposal
8	Inform	19	Request
9	Inform If	20	Request When
10	Inform Ref	21	Request Whenever
11	Not Understood	22	Subscribe

4.2.2 Agent Interaction Protocol

Additionally to the CMS we have to define a basic set of interaction protocols to allow agents to introduce themselves in a new network, to request the other nodes in the network to describe them, to publish, search and request services, etc.

Let's look at our example introduced before: When the firefighter arrives near the burning forest he switches on his PDA. The `pda-agent` controlling the PDA sends a message introducing itself. This message is of type INFORM and describes the kind of device it represents and a rough description of its capabilities: here it would be a 'human interface device' with capabilities to forward messages and to display them to the fireman. Since the fireman is interested in the information of how far the fire has spread already, the `pda-agent` sends a request message for agents capable to measure temperature. Sensors in the specified range capable to measure temperature reply with a short description of their temperature measuring services: these can deliver the temperature per request or by a subscription to deliver it for example every 10 minutes for the next two hours.

In the general case we have to define the following general interactions:

- *description request*: This is a message requesting descriptions of other devices, capabilities or services.
- *description*: This is a message contains the description of the device, capability or service. It can be used to answer a *description request*, to introduce itself when turned on or entering a new area or to inform others about some change allowing them to react to it.
- *service request*: This is a message requesting a specified service.
- *data*: This is a message containing the data like an event, the requested sensor readings or the answer of a requested service.

As we described in 4.2.1 we are going to use FIPA Communicative Acts to specify message types. For the interactions described above we use Communicative Acts: `QUERY_REF`, `INFORM` and `REQUEST`. Additionally we use `NOT_UNDERSTOOD` messages to react to unknown messages.

4.2.3 Heterogeneous Communication

To exchange messages between devices using different communication technologies in the same network: WLAN, Bluetooth, Zigbee, etc., devices capable to communicate by two or more communication technologies would have to bridge the messages between those if needed. This bridging represents a special capability of a node and is also visible as a service. This bridging service is in general not used directly, but depending on the recipients specified in an incoming message and depending on device agents bridging policy.

4.3 Content Representation

In order to search for specific services or information we need a description language for device agents, their capabilities, provided services and for the information exchanged. The content can be described using a simple predefined ontology like in this incomplete example:

- device type: tag, mote, mobile, workstation, vehicle, server, set of devices, other;
- capability type: sensor, actor, HID, energy, communication, CPU, storage, protocol, communication language, other;
- service type: start, stop, invoke, invoke periodically, function, other;
- sensor type: switch, temperature, light, humidity, acceleration, voltage, chemical, positioning, ultrasonic, vision, other;
- actor type: light, switch, motion, manipulator, tool, other;
- HID type: input, output, inputoutput, other;
- energy type: not limited, battery, renewable, passive, other;

This predefined ontology should be kept small, hence it can be coded efficiently. To address a temperature sensing capability we could use the ontology coding for `capability.sensor.temperature`. For each capability the dimension, measuring unit and probably a short description should be given, so those can be provided on request. The used ontology doesn't have to be known completely at each node. It is enough that each node knows the part it is concerned with. The predefined ontology can be extended with more detailed subclassifications and with branches not covered (using the corresponding 'other' branch of the predefined ontology). For other nodes which don't know the extended version of the ontology, the deepest corresponding entry of the predefined ontology can be provided.

4.4 Other Protocols and Languages

A variety of protocols and languages optimized for specific technologies and tasks exist. Nodes capable to use specific protocols and languages model this as their capabilities and can use A3ME to discover other nodes with related capabilities. After discovery of a node with corresponding capability these nodes can be set up to use this capability independently of A3ME. For example, some nodes can be connected to the Internet. After discovering this through A3ME mechanisms, those nodes can establish a connection with each other through the Internet and don't have to communicate through a WSN and avoid wasting energy of battery powered nodes in the network.

For description there already exist more precise and established languages like SensorML [14] or Standards for a Smart Transducer Interface for Sensors and Actuators [3]. Nodes which don't have resource constraints can use these description languages to get more precise descriptions. Here A3ME still can be used in the first step to discover other nodes, supporting these higher level description languages.

5 Conclusion and Future Work

In this paper we introduced the term Mixed Mode Environment and discussed special challenges for middleware in MME. As a possible solution we presented an agent based approach, which has the advantage of being interoperable, since it is designed to work on different devices which have different software platforms and use different communication technologies.

In our future work we will specify the individual components of the presented middleware in detail and enhance the corresponding prototypes. Since the middleware is designed to work in a decentralized manner, it would be reasonable to apply peer to peer mechanisms here. Since there exist a lot of multi-agent-based simulation tools and in A3ME each node is already represented as an agent, it would be reasonable to introduce automatic generation of models for simulation out of the description of device agents.

References

[1] E. Aitenbichler. System Support for Ubiquitous Computing. In *Advances in Pervasive Computing (submission to the Doctoral Colloquium at Pervasive 2004)*, pages 1–6. Austrian Computer Society (OCG), Austrian Computer Society (OCG), 2004.

[2] C. Becker, G. Schiele, H. Gubbels, and K. Rothermel. BASE - a micro-broker-based middleware for pervasive computing. In *PERCOM '03: Proceedings of the First IEEE International Conference on Pervasive Computing and Communications*, page 443, Washington, DC, USA, 2003. IEEE Computer Society.

[3] Committee on Sensor Technology of the IEEE Instrumentation and Measurement Society. *IEEE STD 1451 Standards for a Smart Transducer Interface for Sensors and Actuators*, 2000.

[4] C. Curino, M. Giani, M. Giorgetta, A. Giusti, A. L. Murphy, and G. P. Picco. TinyLIME: Bridging mobile and sensor networks through middleware. In *PERCOM '05: Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications*, pages 61–72, Washington, DC, USA, 2005. IEEE Computer Society.

[5] C.-L. Fok, G.-C. Roman, and C. Lu. Agilla: A Mobile Agent Middleware for Sensor Networks. Technical Report WUCSE-2006-16, Washington University in St. Louis, Department of Computer Science and Engineering, March 2006.

[6] Foundation for Intelligent Physical Agents. *FIPA ACL Message Representation in Bit-Efficient Encoding Specification*, 2002.

[7] Foundation for Intelligent Physical Agents. *FIPA ACL Message Representation in String Specification*, 2002.

[8] Foundation for Intelligent Physical Agents. *FIPA Communicative Act Library Specification*, 2002.

[9] S. Hadim and N. Mohamed. Middleware: Middleware challenges and approaches for wireless sensor networks. In *IEEE distributed systems online*, volume 7. IEEE Computer Society, March 2006.

[10] K. Henricksen and R. Robinson. A Survey of Middleware for Sensor Networks: State-of-the-Art and Future Directions. In *International Workshop on Middleware for Sensor Networks*, pages 60–65, New York, NY, USA, 2006. ACM Press.

[11] P. Levis and D. E. Culler. Maté: A Tiny Virtual Machine for Sensor Networks. In *10th International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 85–95, New York, NY, USA, October 2002. ACM Press.

[12] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TinyDB: an acquisitional query processing system for sensor networks. *ACM Trans. Database Syst.*, 30(1):122–173, 2005.

[13] M. Mühlhäuser, E. Aitenbichler, G. Austaller, A. Hartl, A. Heinemann, and C. Trompler. Towards Personalized Ubiquitous Computing Services. Technical Report TK-01/02, Fachbereich Informatik, TU Darmstadt, 2002.

[14] Open Geospatial Consortium Inc. *OpenGIS Sensor Model Language (SensorML)*, July 2007.