A3ME – Generic Middleware for Information Exchange in Heterogeneous Environments

Arthur Herzog
Dept. of Computer Science
Technische Universität Darmstadt
D-64283 Darmstadt, Germany
Email: aherzog[at]dvs.tu-darmstadt.de

Alejandro Buchmann
Dept. of Computer Science
Technische Universität Darmstadt
D-64283 Darmstadt, Germany
Email: buchmann[at]dvs.tu-darmstadt.de

***Abstract*—This paper presents a proof of concept for a generic middleware for heterogeneous sensor/actuator networks, which enables ad-hoc discovery, self-description exchange and basic interactions between heterogeneous devices. The main approach is to represent each device by a device-agent which knows its capabilities, constraints and policies, and is able to describe those to other devices on request.**

***Index Terms*—Middleware, ad hoc networks, semantics.**

I. Introduction

We present a proof of concept for a new middleware to enable information exchange and interactions among devices in heterogeneous environments. Heterogeneous environments does not mean just different device types, but also devices with vastly different capabilities. The dimensions of heterogeneity include computing capabilities, communication, software, degree of mobility and availability. Such environments are for example areas where conventional computing, Wireless Sensor Networks (WSNs), Wireless Sensor and Actuator Networks (WSANs), Ubiquitous Computing, Mobile Computing and Robotics or a subset of those come together. All these devices have their specific capabilities, properties, policies and constraints. They 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. Regarding the mobility different devices can be static or mobile, but also many mobility levels in between are possible, like a notebook which is static usually, but alternates its position between the working place and home of a person. A sensor attached to a tram is another example, here the sensor is mobile all the time but, its motion follows a defined path namely the route of the tram. Considering the software of the devices, they differ vastly: the devices use different operating systems, software, programming languages, content representation languages, application programming interfaces, etc.

Today, application developers must deal with this heterogeneity when developing a new application for heterogeneous networks. Ad-hoc discovery and interaction with new nodes is only possible for specialized solutions, but not in general. Usually 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 a heterogeneous environment without the need of adjustments each time new hardware is introduced. The agent-based approach offers an abstraction for the different devices: it sees all the different nodes in the network as independent entities – device-agents (DAs). Each device-agent knows its capabilities, properties, policies and constraints. Depending on its capabilities, a device-agent can offer services to other agents and can perform tasks, sometimes using the services of other agents. Basic ideas for this middleware were also described in [9].

II. Related Work

For each of the before mentioned areas there exist domain specific middleware solutions, but to simplify the interaction between those we need a more generic solution with small footprint. In *conventional computing* there are established solutions like Common Object Request Broker Architecture (CORBA) [4], Enterprise JavaBeans (EJB) [6], Web Services and their software stack. For many kinds of devices these solutions are too heavy weight w.r.t. communication, computing and storage requirements. These solutions also are basically centralized solutions, relying on one or multiple central servers needed to enable middleware functionalities.This also limits or even prevents ad-hoc interactions when the central instances are not reachable.

In the *WSN* area most research has been done in homogeneous WSNs using the same kind of device for all nodes in the network or in combination with a second more powerful kind of device that is used as a gateway and data sink. One kind of middleware used here is TinyDB [13], which represents a

Supported by the DFG Research Group 1362, Cooperative, Adaptive and Responsive Monitoring in Mixed Mode Environments and by the LOEWE Priority Program Cocoon.

978-1-4673-1786-3/12/$31.00

homogeneous network of sensors as a relation on which you can execute SQL like queries. Another type of middleware in WSN is Agilla [8]. Here executable code can move between devices and continue its execution on different sensors for example to compute average temperature. The WSN operating systems like TinyOS [12] and Contiki [7] also can be seen as a kind of middleware for WSN since they simplify the programming and interactions within the WSN. In contrast to it our solution is designed to enable interactions also beyond the own WSN.

In *WSANs* an additional type of nodes – actuators – is used. They are capable of interacting with the physical world (e.g. automatic watering of plants controlled by soil humidity sensors). This type of networks have usually low number of different device types.

In *ubiquitous computing* multiple devices in an individual's surroundings are performing tasks. Hereby the user doesn't necessarily have to interact consciously with the devices. Additionally to simple sensors and actuators a broad variety of devices is involved here like media devices, mobile phones, light and temperature control devices, etc [11].

III. Goals

In the development of the A3ME Middleware we focused on following goals:

a) Ad-hoc Interactions: The primary goal is to provide on-the-fly interactions with different devices. This means devices must be able to discover each other, without individual manual adjustment, exchange their descriptions and offer/use services to/from each other.

b) Decentralized Solution: A centrally controlled heterogeneous network with thousands of devices doesn't scale well. Therefore our goal is to enable these ad-hoc interactions in a decentralized manner without a central instance needed to enable the interaction.

c) Technology Independence: Information technology evolves very fast and many specific technologies that are standard today will be obsolete in a few years. Therefore, it is important to design a solution that abstracts from the specifics of individual technologies and makes it possible to replace them or use different technologies in parallel.

IV. Device-agent based Middleware

We will present our middleware approach in a top-down manner.

A. Nodes Represented by Device-agents

The idea is to see each node in the network as an agent, which knows its capabilities and might have tasks it tries to perform by communicating and cooperating with other agents in the network. To distinguish our agents from the different kind of agents used in computer science, we call the agents in our approach *device-agents*. A device-agent is special software, which resides on a specific device and knows the device's specific capabilities, constraints, policies and services (Figure 1). Each of the nodes represented as device-agent is an independent entity, which is also able to function on its own. These entities interact with each other to build a network and to enable higher level services and capabilities. An agent-based approach facilitates the self organization and adaptability of the system.

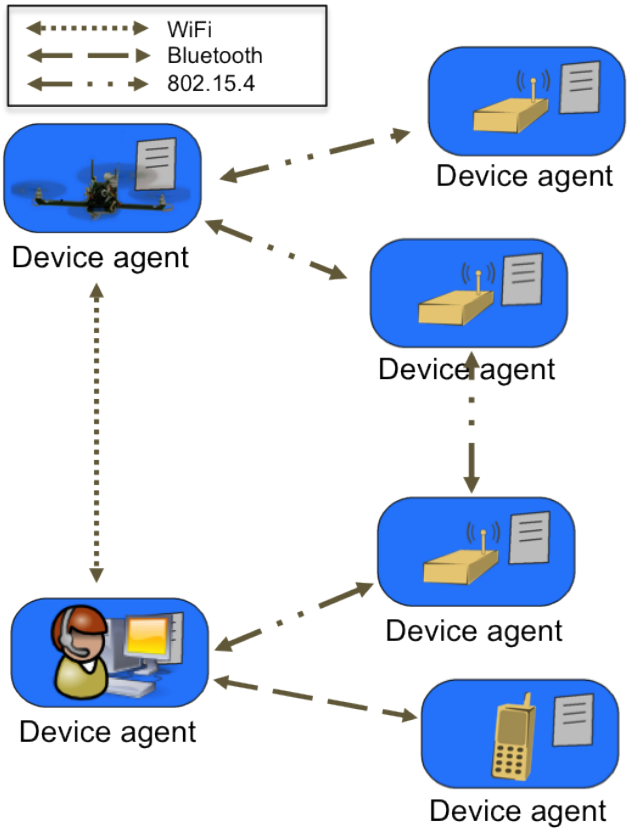

Figure 1. A3ME device-agents

Each device-agent is seen as a black box. This means we want to hide the hardware and software the node is built of, and just show its capabilities and services to the outside. To interact with each other, device-agents have to support a basic set of messages and interaction protocols. This will be described in more detail in the following sections.

B. Device-Agent Interface

Interaction Interface defines how device agents interact with each other. All interactions happen through exchange of messages. The structure of these messages is described in IV-D. Different interactions are composed of interaction primitives described in IV-C. In section IV-J we describe how the content of the A3ME messages is built.

C. Interaction Protocols

Here a minimal set of interactions is defined, which are enabled through A3ME between different devices through device-agents. These protocolls specify the flow of message types to be exchanged for each type of interaction.

1) Inform Interaction Protocol: The Inform Interaction Protocol is used to introduce a device-agent to others. This is usually done when a device is switched on, enters a new area while moving or periodically. In this interaction just one Inform message is sent through all available communication channels of the device and does not require other device-agents to react to it.

2) Request Interaction Protocol: The Request Interaction Protocol is the protocol used for all kinds of requests. Request Interaction is initiated by a $Request$ message. Every receiver answers the request with an $Inform$ message containing the requested information. An information request might be rejected when answering it violates the policy of the DA.

3) Call Interaction Protocol: The Call Interaction Protocol is used to call a service offered by the device. The command needs either the service ID or the capability to which it is related. In the second case the default service for the given capability will be called, e.g. for an LED the toggle LED will be called.

D. Common Message Structure

This section describes the structure of the message, which will be transported by the different communication technologies as payload. This means the A3ME message will be put inside a communication technology specific message body. The communication technology specific message might contain other information required by the protocol used, like sender address, receiver address etc. For example, if the communication protocol used is TCP/IP, the addresses would be IP-addresses.

The A3ME Message contains parameters defined in FIPA ACL Message Structure Specification [2]. The only mandatory parameter is the performative. Usually it also contains sender and destination addresses for the message, it also contains their IDs and eventually other kind of addresses valid for the sender and/or receiver on another communication interface.

E. Message Types

For interactions between nodes messages of predefined types are used. FIPA ACL performatives [3] are used as types. These define 22 performatives, whereof we currently use four: inform, request, cancel and not understood.

F. Addressing of Individual Devices

In heterogeneous environments a device might have different communication capabilities and for each of those it has a different address, often even of different kind. For example a device can have an IP-address for the Ethernet connection, a second IP for the WIFI connection and a com-port to communicate with a sensor node base station, which on its side has an address inside the WSN.

In A3ME we decided to identify each device by a device-agent identifier (DAID). A DAID is composed of textual device-agent name and of a set of addresses, where each address also has an address type assigned. This solution was inspired by agent identifiers (AIDs) used in JADE [5] Java Agent DEvelopment Framework. The IDs of the agents in JADE are composed of a name and the platform on which they reside.

G. Group Addressing

In requests it is also possible to address whole groups of devices. In A3ME we don't focus on a specific solution, but allow the use of any existing group-addressing technology. A simple type of group-addressing would be to address only devices of a specific device type or all nodes using a specific communication technology (e.g. Bluetooth). But it is also possible to use more sophisticated groupings. This could be for example Logical Neighborhoods [14] or Scopes [10], where the groups are defined dynamically depending on device's static or dynamic properties.

H. Message/Data Encoding

All messages and data structures in A3ME are defined in ASN.1. For transferring this data on top of any communication technology it is encoded and afterwards decoded using the aligned ASN.1 Packed Encoding Rules (PER) [1].

I. Predefined Extensible Classification

For classification of devices, capabilities, properties, services, etc. we developed a predefined simple ontology. This ontology does not pretend to be complete. Instead it is extensible with base-classes.

The advantage of using an ontology with base-classes is that it enables interactions with before unknown devices and eliminates the need to agree on a complete ontology to classify each other.

J. Query Language

Devices exchange and interpret messages defined in ASN.1 syntax, but for humans it is not very comfortable to define queries using ASN.1 directly. Therefore we define also a SQL like query language (A3ME-QL) to simplify the entering of queries by humans. These queries are then translated into ASN.1 and forwarded to the A3ME middleware. When presented to the user ASN.1 queries are translated back into the A3ME-QL.

V. Device-agent Realization Levels

The realization of device-agents representing, for example, a small sensor node and a smart phone are quite different. The DA for sensors might only offer a static description of the sensing capabilities and basic services to get the sensor readings. The smartphone-DA can have in addition to its sensing capabilities the ability to communicate via different communication channels and offers an interaction interface to a human user. Therefore we classify the DA realizations into two levels: core and basic A3ME DA realizations.

The core A3ME DA is the smallest version of a device-agent realization. It provides only the self-description information about the device it represents. In the simplest case it is only the static information which is programmed into the device on deployment. An enhanced version additionally can contain

dynamic info which is either collected, measured or calculated during runtime.

The Basic A3ME DA extends the Core functionality with the basic functionalities of the represented device as services. These are in the case of a sensor device the services which offer the sensor readings of the connected sensors.

VI. Example Walk trough

In this section we will describe the activities in the A3ME framework using a simple example. A mobile device has the task to discover temperature sensors in its proximity and collect temperature readings.

a) Query in A3ME QL: We can think of a local application which is built on top of A3ME middleware and implements the described task. To get the needed Information we can formulate a query as follows:

```
1 REQUEST device.name, temperature.data
2 FROM ALL
3 WHERE IS–A mote
4 PERIOD 1 minute DURATION 5 minute
5 RANGE 3 hop
```

In line 1 of the query we list the data-descriptors we are interested in: device.name, temperature.data. The left part of the data-descriptors is always an entry of the A3ME ontology (here: device and temperature). The right part of the data-descriptor describes the information-type we are interested in. The possible values here are: type-code, type-name, name, description, id, data and m2m-description. $FROM ALL$ specifies from whom we are querying the information: ALL means it is addressed to all who get the message. Line 3 adds a capability existence condition: here only devices which have the capability mote, meaning they are of type mote, shall answer the query. Line 4 specifies that we are interested to get the answer every minute for the duration of five minutes. Line 5 limits the forwarding range of the query to 3 hops.

b) Encode Message: The query is encoded using ASN.1 unaligned ASN.1 Packed Encoding Rules (PER). The length of the example query described above is encoded to a byte array of 19 bytes.

c) Message Transport: The encoded query is put into a bluetooth message and sent to the connected workstation. The workstation forwards the bluetooth message to other bluetooth devices and additionally extracts the payload from the bluetooth message, puts it into a message for sensor nodes and broadcasts it through the connected sensor-node-basestation to the sensor nodes. Now the workstation decodes the payload and evaluates the contained query. Since the workstation does not satisfy the condition in the WHERE part, it does not have to send any answer.

The sensor nodes in range get the message forwarded by the workstation. Since the forwarding range is not exceeded yet (3 hops), they first forward the message, than decode the payload. Now they evaluate the contained query and set up a repeated answering of the query every minute for the duration of 5 minutes. An answer message is then sent to the requester periodically and contains a result set with one row with the name of the device followed by the current value of the temperature. The workstation gets the individual answers from the sensor nodes and forwards them to the requesting mobile device via bluetooth.

VII. Conclusions and Future Work

The realization of A3ME enables basic interaction between heterogeneous devices in heterogeneous environments. It extends existing solutions with more interoperability yet is simple enough to be usable on resource restricted nodes directly. The use of a predefined but extensible ontology enables interaction with a priori unknown nodes. Basic interactions offered by A3ME device-agents enables ad-hoc device, capabilities and service discovery. ASN.1 based messages and data definition allows to generate program code for these structures, thereby simplifying and accelerating the development of DAs and applications for new devices.

As our future work we would like to build new applications on top of this basic middleware, which now can use and benefit from the specialized solutions from the different research areas. One such application would be a generic graphical user interface (SMARTUI). Such a user interface would enable a user to discover and query sensor-measurements from nearby sensors, discover devices in a smart home and interact with them, discover a robot and control it, discover and allow interactions with other users via their devices. The information about discovered devices, capabilities, users etc. can be completed and looked up via semantic web mechanisms and visualized on a map.

References

[1] ASN.1 Encoding Rules-Specification of Packed Encoding Rules (PER). Recommendation x.691, ITU-T, 2002.

[2] FIPA ACL Message Structure Specification. Standard, FIPA, 2002.

[3] FIPA Communicative Act Library Specification. Standard, FIPA, 2002.

[4] Common Object Request Broker Architecture (CORBA) Part 1: CORBA Interfaces, 2008.

[5] F. Bellifemine, G. Caire, A. Poggi, and G. Rimassa. JADE A White Paper. Technical Report September, Telecomitalialab, 2003.

[6] L. DeMichiel and M. Keith. Enterprise JavaBeans TM Version 3.0 Simplified API. Technical report, Sun Microsystems, 2006.

[7] A. Dunkels, B. Grönvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *First IEEE Workshop on Embedded Networked Sensors (Emnets-I)*. IEEE CS, 2004.

[8] C.-L. Fok, G.-C. Roman, and C. Lu. Agilla: A Mobile Agent Middleware for Sensor Networks. Technical Report WUCSE-2006-16, Washington University, St. Louis, 2006.

[9] A. Herzog, D. Jacobi, and A. Buchmann. A3ME - An Agent-Based Middleware Approach for Mixed Mode Environments. In *2008 The Second International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*, pages 191–196. IEEE, Sept. 2008.

[10] D. Jacobi, P. E. Guerrero, I. Petrov, and A. Buchmann. Structuring Sensor Networks with Scopes. In *3rd IEEE European Conference on Smart Sensing and Context (EuroSSC)*. IEEE CS, 2008.

[11] K. Kreuzer. Openhab – Open Home Automation Bus. http://openhab.org.

[12] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. TinyOS: An operating system for sensor networks. In *Ambient intelligence*. Springer Verlag, 2004.

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

[14] L. Mottola and G. P. Picco. Programming wireless sensor networks with logical neighborhoods. In *1st international conference on Integrated internet ad hoc and sensor networks - InterSense '06*. ACM, May 2006.