Efficient group communication based on Web services for reliable control in wireless automation

Markus Jung and Wolfgang Kastner
Institute of Computer Aided Automation
Vienna University of Technology
Vienna, Austria
e-mail: {mjung,k}@auto.tuwien.ac.at

Abstract—The ongoing demand for more energy efficiency within commercial buildings and homes is a driving motivation for using IT and IP technologies within underlying building automation systems (BAS). Building data and access to automation systems should be made available within internal and external IT systems enabling visionary cloud computing based interaction scenarios. Web services and service-oriented architectures are a promising approach to provide such a communication infrastructure. Communication protocols like CoAP and EXI enable Web service communication on embedded devices but are currently limited to a client-server interaction model with unicast based message exchange. Several existing building automation systems use a group communication approach to exchange process data, which allows to create basic control scenarios without the need of a central control unit. This group communication model for process data is strongly tied to the application layer services and information models of the according BAS. This paper presents a group communication model for CoAP and EXI based Web services, contributes further details on the implementation and provides an evaluation compared to client-server based interaction scenarios.

I. INTRODUCTION

Building automation systems are in the middle of a technology revolution. Driven through the rising energy prices and buildings being responsible for consuming about 23% of the world-wide electricity the trends influencing current research are towards more energy efficiency and integration into IT systems [1].

Interoperability is a key challenge within building automation systems, due to the high number of standards or the lack thereof which is also an inhibitor for the success of home energy management systems [2]. The use of IP at the network layer is one step towards more interoperability. Technologies like 6LoWPAN allow an efficient usage even within constrained wireless low-power and lossy networks (LLNs). Beside IP, the use of Web services and text-based message encodings like XML and JSON are a way to address interoperability at the communication layer. Technologies like the Constrained Application Protocol (CoAP) [3] and efficient XML interchange (EXI) [4] allow a lightweight variant of a Web service communication stack [5] [6]. Compared to RESTful Web services based on HTTP and following a strict request-response interaction based on a client-server communication model, CoAP already provides more interaction patterns by offering non-confirmed and asynchronous communication patterns and further opens the possibility to group communication by using UDP for packet exchange. However, the current standard does not contain details on group communication which is addressed by a separate IETF draft [7]. In our previous work, we have identified the need of normalized application layer services and information models in order to provide a complete and interoperable communication stack and presented a concept based on the Open Building Information Exchange (oBIX) [8] standard together with an efficient CoAP/EXI protocol binding. We further showed how a gateway can be setup enabling such a communication stack for existing BAS technologies like KNX, BACnet and ZigBee [9]. A short-coming of this Web service stack compared to existing BAS is the need of a central controller that executes control logic and acts as a client to all CoAP endpoints offered by the devices. Although using the asynchronous communication capabilities of CoAP and scripting engines support an efficient and flexible realization of such control logic containers, still a single-point of failure is introduced. Also think about a cloud-based interaction scenario where likewise appliances are controlled from a remote control center. In this paper we present how our existing oBIX/CoAP/EXI protocol stack can be extended with group communication facilities that enable decentralized control scenarios and interactions between sensors (e.g. push button) and actuators (e.g. light switch actuator) without the need of a central control unit or the excessive use of unicast messaging. Furthermore, an API for group communication management based on oBIX is shown and once more illustrates the necessity of normalized application layer services and information models. The use of oBIX provides the required data point semantics to realize such group communication services. In parallel with EXI, it is possible to make the message encoding and processing as efficient as binary encodings of state-of-the-art building automation systems without any performance penalty through the use of XML based messages. Together with the presented gateway approach this communication model allows the seamless integration of existing BAS systems together with IPv6 and CoAP based devices providing interoperability between non-IP based BAS systems. The structure of the paper is as follows. Section II outlines related work and alternative solutions to the presented concept. The group communication model together with the Web service stack is presented in Section III,
which illustrates the different possible interaction scenarios and provides an overview of how IPv6 CoAP devices can use the group communication model for process data exchange and how this communication model can be used together with a gateway for non-IP BAS devices. The implementation is described in Section V and an experimental evaluation of different interaction scenarios is sketched in Section VI.

II. RELATED WORK

In [10], a RESTful runtime script container for IoT applications is presented. The concept is based on a JavaScript execution engine that is extended with CoAP capabilities to perform asynchronous communication. For creating control logic, the runtime container acts in a client-server model with the CoAP enabled devices. For control logic that involves the interaction between two different devices, all the information has to be transported to the application server from the one device and back to the other device. Although it is shown that using asynchronous communication this can be done quite efficiently, still a single-point of failure is introduced. The group communication model presented in this paper avoids this single point of failure and the additional round trip of information.

[7] provides a first standard draft on using IPv6 multicast for CoAP group communication. The document guides how to use CoAP in the context of group communication. Furthermore, it includes different use cases and protocol flows that illustrate the group communication. The standard proposes the use of a specific group communication resource that should be offered by a CoAP endpoint. A group may consist of one or more CoAP endpoints. A CoAP endpoint can join multiple groups. We identify this as a misconception that limits the use of group communication scenarios, because in our concept a group is not based on CoAP endpoints instead it is based on data points, where one CoAP endpoint can provide multiple data points. Having CoAP as message exchange protocol does not provide a standardized information model of resources – no normalized data point semantics are specified. This is covered by the application layer service specified in our stack through oBIX, enhanced with a novel group communication extension.

The use of embedded Web services in smart metering applications is investigated by [11]. It outlines the interoperability problem of heterogeneous standards found in the area of smart metering and the need of a Web service based integration layer. It presents a smart meter profile based on DPWS and SOAP Web services using an efficient binding to CoAP and EXI. In contrast, our work uses oBIX on top of such an efficient protocol stack, since we identified the RESTful paradigm of oBIX to be more suitable to the domain of home and building automation systems. The group communication model presented in our paper could complement their solution and also be incorporated into DPWS based SOAP Web services if a CoAP binding is used.

A resource-based integration middleware for building automation systems is presented in [12]. This approach can be compared to our approach of a transparent IPv6 multi-protocol gateway. In contrast to our proposal, the integration middleware is based on SOAP and the efficiency of the service interfaces is not addressed. The middleware adheres to a custom application layer following the paradigms of associative memory, production rules and distributed transactions.

III. IoT PROTOCOL STACK

This section presents an overview of our already proposed communication stack based on IPv6, the constrained application protocol (CoAP), efficient XML interchange (EXI) and Open Building Information eXchange (oBIX) [9] [13] which is enhanced with a novel group communication model in the following section. The protocol stack follows a waist-line architecture demanding IPv6 as common network layer. This requires each device to be addressed using IPv6. Further, we demand that even each data point should be addressable using IPv6.

On the lower layers of the protocol stack 6LoWPAN allows the use of IPv6 even in LLNs operated on IEEE 802.15.4. For our proposed concept, there is no requirement for a specific link, instead typical links found in LLNs require the upper layers of the protocol stack to be designed carefully. Fragmentation needs to be avoided. Otherwise the energy consumption for constrained nodes is too high. Therefore, the payload has to be as small as possible. On the upper layers different protocol bindings and message encodings for oBIX are provided, which can be combined in multiple ways.

For constrained devices only the CoAP and EXI protocol binding is most relevant, since it also enables efficient group communication. HTTP based Web services like RESTful Web services or SOAP Web services are too resource demanding and require a strict request-response interaction within a client-server model. Asynchronous communication is not possible due to the use of HTTP. Group communication is not possible due to the use of TCP. To clarify the presented group communication model only a brief overview for oBIX is provided. For details on IEEE 802.15.4, 6LoWPAN, CoAP and EXI please refer to the according specifications.

![IoT protocol stack](image-url)

**Fig. 1:** IoT protocol stack
A. Open Building Information Exchange (oBIX)

oBIX attempts to provide a standard XML syntax for representing machine-to-machine (M2M) information provided by embedded sensors and actuators. Its goals are to use enterprise friendly technologies like XML, HTTP and URIs for M2M communication. Although oBIX was designed to work on embedded devices, HTTP and XML are not the best technology choice for LLNs. To address this issue, oBIX specified in the 1.1 specification working draft a custom binary protocol designed for the use in 6LoWPANs. A further goal is to provide a standardized representation for common M2M features like data points, histories and alarms extensible for custom enhancements.

**oBIX object model.** For representing M2M information, oBIX specifies a generic and simple meta-model for information modelling represented through the oBIX object model. It defines 17 base object types, including 10 value types, and further comes with the concept of contracts. Contracts allow to define a classification of certain types of oBIX objects. They consist of a template model and allow the generation of platform specific types (e.g. Java or .NET classes). Contracts by themselves are also expressed as oBIX objects and allow to convey semantics for human developers with default values for objects adhering to these contracts. In this way, flexible type inheritance is guaranteed for oBIX objects. Objects are based on a set of standard attributes but can also be extended with custom attributes (facets) and further can contain other objects as children (aggregation) or reference other objects (composition). Objects can also expose functionality as oBIX operations which can be invoked by a client.

Within our protocol stack, contracts are used to model standard device types found in building automation systems and wireless sensor networks. These so called IoT contracts provide a generic device representation that can be used on CoAP enabled devices or at gateways for existing non-IP technologies directly. Contracts also define standardized service interfaces. The oBIX core library comes with a set of standard contracts including a watch service, histories and alarming.

Below are two simple example contracts of a light switch actuator that closes and opens an electrical circuit and a push button that acts as a sensor and might be linked to the actuator.

**Listing 1:** Light switching actuator contract

```xml
<obj href="iot:LightSwitchActuator" is="iot:Actuator">
  <bool name="value" href="value" val="false" writable="true"/>
</obj>
```

**Listing 2:** Push button contract

```xml
<obj href="iot:PushButton" is="iot:Sensor">
  <bool name="value" href="value" val="false"/>
</obj>
```

**Networking.** For networking, oBIX defines the client-server communication model and further adheres to the request-response interaction of HTTP. Three request types are specified: i) **read** for accessing the current state of an object, ii) **write** for updating the current state, and iii) **invoke** for execution of an operation. For HTTP, the protocol binding is straight forward to the **get**, **put** and **post** protocol verbs.

**Data points.** Within automation systems data points are a widely used term to identify I/O-signals provided by a sensor or actuator (**hard point**) or a configuration variable like the setpoint for a controller (**soft point**). oBIX provides a contract to mark basic value objects as data points and further differs between writable and read-only data points.

IV. IoT GROUP COMMUNICATION

This section presents how the presented IoT stack can be enhanced with highly efficient group communication capabilities. Use cases for this group communication are provided in the context of embedded CoAP devices in LLNs and completed with interaction scenarios with non-IP building automation systems (e.g. KNX, BACnet, Wireless M-Bus) that are integrated transparently using our IPv6 multi-protocol gateway [9] which offers per-device CoAP endpoints.

The group communication concept is inspired by existing BAS (e.g. KNX) that assign group addresses to data points. The same approach can be applied to oBIX if a CoAP protocol binding is used and by assigning IPv6 multicast addresses to each group of data points that should interact with each other.

A. IPv6 multicasting

The proposed group communication model takes advantage of the IPv6 multicast mechanism that allows to transmit an IPv6 packet to multiple hosts. Multicast addresses have the prefix FF02::/16 followed by two 4 bits groups for defining flags and a scope as illustrated in Table I.

<table>
<thead>
<tr>
<th>Bits</th>
<th>8</th>
<th>4</th>
<th>4</th>
<th>112</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field</td>
<td>prefix</td>
<td>flags</td>
<td>scope</td>
<td>group ID</td>
</tr>
</tbody>
</table>

**TABLE I: IPv6 multicast address format**
In order to avoid conflicts with well-known addresses\(^1\) the transient flag must be set. In that case it is possible to use the 48 or 32 bits representation with an address space of either 32 (cf. Table II) or 16 bits for group communication relationships within local networks.

### B. Group communication object

The IPv6 multicast addresses can be assigned to basic oBIX value object types that are adhering to the obix:Point contract. Only basic value object types (e.g. int, real, bool) can provide this data point semantics. For the group communication, a new contract iot:groupCommContract is introduced as illustrated in Listing 3.

**Listing 3: Group Communication Contract**

```xml
<obj name="groupComm" href="iot:groupComm">
  <list name="groups" href="groups" of="obix:str"/>
  <op name="joinGroup" href="joinGroup" in="obix:str" out="obix:list"/>
  <op name="leaveGroup" href="leaveGroup" in="obix:str" out="obix:list"/>
</obj>
```

The contract is kept simple. It supports to query the current groups of a data point and two operations which allow to join or leave a group. The operations take as input an IPv6 multicast address as obix:str object and return the current list of groups. For invoking the operation, the post method of CoAP or HTTP needs to be used.

The group communication object allows managing the assigned addresses at runtime. Where some nodes might be configured statically with certain multicast addresses it is possible to add or remove IPv6 multicast addresses on certain data points at runtime. The feature of administrating the assigned group communication relationship at runtime depends on the platform and the available OS socket API. Section V describes a possible realization, which is part of the presented proof of concept implementation.

### C. Group communication semantics

To have meaningful group communication, only data points of the same value type or a value type that can easily be transformed to another value type (e.g. int to real and vice versa) should be put into a single group. Consider a sensor like a push button and an actuator like a switch actuator. Both have a value point represented by a bool object. Assume that both points joined the group represented through the IPv6 multicast address FF12::1. Once a human person presses the button, a physical signal is detected and the according oBIX object is updated. If the oBIX point is an output value, which is determined by not having the writable attribute set to true, the updated value is sent out to all registered group communication IPv6 multicast addresses. The switching actuator receives the multicast request and detects that the address is assigned to the local data point. The oBIX value is updated accordingly and triggers the physical signal for the switching actuator.

### D. Group communication request format

The group communication solely relies on IPv6 and non-confirmed CoAP put requests as recommended in [7] in order to avoid congestion. This limits the application of the current solution to non-safety-critical use cases or use cases in which a human user is in the control loop and compensates any message failures. The IPv6 multicast group communication request is encapsulated into a UDP packet consisting of a CoAP header and the payload. The CoAP header takes 4 bytes. Additional 2 bytes are required to provide the content-format information. Since the IPv6 multicast address is directly linked to the value of a data point no further location path or query needs to be transmitted. For simple switching actuation the UDP payload is below 10 bytes. In the case of LLNs and 6LoWPANs according to [14], as a rule of thumb, the UDP application payload has to be below 50-60 bytes in order to avoid fragmentation. The presented group communication request format stays well below this limit. For the group communication, the payload can be standard oBIX XML, but preferable it should be binary EXI encoding using the oBIX schema information. In this way, the payload can be further reduced from tens of bytes required to encode basic value objects to a few bytes as outlined in Table III for a selected number of value object types.

### E. Group communication interaction scenarios and use cases

For the group communication multiple interaction scenarios and use cases can be identified (cf. Figure 2).

---

\(^1\)http://www.iana.org/assignments/ipv6-multicast-addresses/ipv6-multicast-addresses.xml

### TABLE II: IPv6 multicast address format

<table>
<thead>
<tr>
<th>Field</th>
<th>8</th>
<th>4</th>
<th>4</th>
<th>32</th>
</tr>
</thead>
<tbody>
<tr>
<td>scope</td>
<td>prefix</td>
<td>flags (transient)</td>
<td>group ID</td>
<td></td>
</tr>
</tbody>
</table>

### TABLE III: Group communication payload

<table>
<thead>
<tr>
<th>Object type</th>
<th>CoAP payload ex.</th>
<th>XML bytes</th>
<th>EXI bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bool</td>
<td>&lt;bool val=&quot;false&quot;/&gt;</td>
<td>20</td>
<td>3</td>
</tr>
<tr>
<td>Int</td>
<td>&lt;int val=&quot;58&quot;/&gt;</td>
<td>15</td>
<td>4</td>
</tr>
<tr>
<td>Real</td>
<td>&lt;real val=&quot;58.12&quot;/&gt;</td>
<td>19</td>
<td>6</td>
</tr>
<tr>
<td>Str</td>
<td>&lt;str val=&quot;hello world&quot;/&gt;</td>
<td>24</td>
<td>15</td>
</tr>
</tbody>
</table>

---

Fig. 2: Group communication interaction scenarios
**Grouping similar devices.** First of all, similar devices can be grouped together. For example, light switching actuators of rooms, floors and buildings can be arranged in groups aligned to the building topology and assigned to a separate IPv6 multicast address (e.g. FF12::1). Similarly a functional grouping including all devices covering different domains (e.g. lighting/shading, HVAC) available at the local site can be done.

**Process communication.** Beside sending requests to groups of similar devices it is also possible to use group communication for exchanging process data between sensors or actuators. This interaction pattern can be observed in state-of-the-art BAS like for example KNX. The example provided above shows how a sensor can be linked to an actuator based on an IPv6 multicast group address (e.g. FF12::1). Together with the highly efficient EXI encoding this interaction pattern is a reasonable choice for automation systems based on LLNs. Furthermore, it provides the advantage of not having a centralized controller that might fail or congestion if decentralized control is based on unicast messaging.

**Integration with existing BAS technology.** Another use case for the group communication can be identified together with the gateway approach, in which BAS devices of non-IP technologies (e.g. KNX, BACnet) are represented through a unique IPv6 per-device CoAP interface at the gateway. In this case, the group communication model can be used to interconnect native IPv6 and CoAP devices but also just as integration middleware between KNX and BACnet for example. Furthermore, one use case for the integration with legacy technologies is the seamless integration of LLNs based sensors into existing HVAC control setups, in which IPv6 based CoAP devices can be used within BACnet control logic. As a simple case study, think of a BACnet control logic operating on hard points like binary input/output and analog input/output objects and soft points like a set point for a temperature control usually represented through binary or analog value objects. While the current room temperature might be represented through an analog input object this can be replaced through a soft point represented through an analog value object which is in the same multicast group like a sensor in LLNs (e.g. FF15::3 with site-local scope). In this way, LLN sensors can complement existing automation systems and be seamlessly integrated into existing control logic.

**V. IMPLEMENTATION**

The presented group communication model for oBIX is based on CoAP and EXI is implemented based on Java within our transparent IPv6 multi-protocol gateway, which is provided as open source project. The group communication capability is also already open source and can be easily configured for devices and their value object types.

**A. CoAP multicast**

For implementing the group communication model, the Java CoAP library Californium [15] needed to be modified, since no multicast communication was supported at the time being. The modified library is also available as open source. For the implementation, the problem of identifying the target IPv6 multicast address in Java needed to be solved. If a socket joins multiple multicast addresses it is not possible to identify the original target address. In order to solve this issue a dedicated datagram socket for each IPv6 multicast address is created. Once a request is received the socket uses a thread context variable to mark the request as multicast request and further stores the single multicast address to which the socket has joined. This thread context information is used by the server for further processing of the multicast requests.

**B. Group communication service**

For managing the relationship between oBIX data points and IPv6 multicast addresses, a group communication service keeps track of the relationship. If new IPv6 multicast addresses are registered, sockets are dynamically opened and if all relationships to a multicast IPv6 address are closed the according socket is closed. Furthermore, the message processing pipeline is modified, if IPv6 multicast requests arrive. The group communication service takes over the processing of requests and bypasses the oBIX server. The group communication service operates directly on the oBIX objects building the abstraction layer among heterogeneous technologies. If multicast requests are received the according objects are updated. If an update concerning an object is observed the updated value is sent to all assigned multicast addresses.

**C. Group communication administration**

For the administration of the assigned group communication addresses, an HTML5 interface is provided at the gateway. It uses the HTTP binding together with JSON encoding to provide a generic oBIX client that allows basic interaction with oBIX primitive data points, like for example to toggle boolean data points or to provide input fields for numerical values. A proxy mechanism is used to enable the administration of CoAP nodes within this interface. The HTML5 interface uses Javascript to retrieve the available oBIX objects which are discovered using the oBIX lobby. The object representation is parsed and according input fields are shown. If a basic data point is equipped with a group communication object, identified through the oBIX contract, according input elements are shown that allow to group multiple data points of different objects. The group addresses are numbered consecutively in the available address range.

**D. oBIX EXI encoder/decoder**

For encoding and decoding oBIX objects, the oBIX toolkit has been extended to provide direct EXI encoding and decoding without any XML based conversion. Based on a SAX stream parser for EXI this can be done without any performance penalty.
VI. Evaluation

For the evaluation, three scenarios of interacting with IoT devices have been identified. Based on a previous evaluation [13] the i) cloud-based control scenario and ii) local gateway control scenario have been extended with iii) the group communication control scenario.

![Fig. 3: Control application logic scenarios](image)

A. Delay evaluation

For the delay evaluation, the IPv6 multi-protocol gateway is deployed on a Raspberry Pi board (700 Mhz ARM CPU, 512 MB memory) that provides a standard hardware platform. The introduced delay through the different control logic scenarios is measured by using a KNX test bed. Instead of linking a push button to a switching actuator based on KNX group communication those CoAP endpoints are used that are provided for the devices at the gateway.

For the local and cloud control scenario, a Java application operates in a typical client-server communication style with the different CoAP endpoints using XML encoded messages. With the CoAP observe capability it is possible to asynchronously push updates on changed data points to clients. For the group communication, no separate application process needs to be started. Solely the group communication address table is used. The group communication uses the efficient schema informed EXI encoding together with the direct EXI-to-oBIX encoder/decoder for basic value object types. On the gateway for the local and group communication address the unicast and multicast CoAP communication is handled using the local network interface.

The group communication approach out-performs clearly the other control approaches as illustrated in Figure 4. Although even a cloud-based control scenario would allow a delay in the subsecond area, for control scenarios with humans in the loop the group communication based approach would be most applicable.

![Fig. 4: Delay evaluation](image)

As evaluation of the different decoders used for the group communication a benchmark on the Raspberry PI platform has been performed using messages based on the basic value types bool, int, real and str (with 10 characters).

![Fig. 5: oBIX Decoder evaluation](image)

As shown in Figure 5, the decoding approaches that require a transformation over XML perform poorly. The direct EXI decoder created within the implementation is more efficient than the XML decoder and far more efficient than the other decoders except the oBIX binary decoder, which still provides the best performance but at the price of requiring a custom implementation for each platform. Performance gains can be expected, once the used EXI library becomes more mature.

B. Reliability and performance discussion

The group communication model solves severe reliability and performance problems of client-server based control scenarios. Since no central control entity is required several potential failure sources could be excluded. If CoAP devices within an LLN support this group communication mode only a device failure or a transmission failure within the LLN would lead to a failure in the control logic. The probability of a collision with the LLN can be reduced through the use of the efficient EXI encoding leading to a small payload size.

In contrast for a cloud-based solution where the server in the cloud may fail, there might be a connectivity issue with Internet connection between the control center and the devices. Even for a local control application the additional process executing the application logic introduces another source of failure.
Regarding the performance, the client-server model requires always the round-trip between the sensor to the controller and the actuator. Although this can be done quite efficiently using the asynchronous communication facilities of CoAP, still further communication is performed that negatively affects the performance of the LLN and increases the probability for collisions. With a group communication model, this round-trip can be avoided. Furthermore, the CPU demand on control entities can be reduced. Also for the gateway approach, the group communication model dramatically increases the efficiency allowing to support the access to more devices by means of the same hardware. However, the use of group communication model and the client/server model can coexist.

It can be thought of combining both approaches, e.g. having the group communication model for local sensor-actuator interaction probably with a human being in the control loop, and the cloud-based control just to interact with soft points in order to optimize the energy consumption of a building.

C. 6LoWPAN and IPv6 multicasting discussion

Efficient IPv6 multicasting in LLNs is still a hot research topic [16]. Especially for CoAP, group communication is still in a draft state [7]. In general two-ways are identified to realize multicasting: One-way is to use mechanisms where a node joins a specific multicast group from which it wants to receive messages and for the second approach group joining is not required. The multicast listener discovery (MLD) protocol or the IPv6 routing protocol of low-power and lossy networks (RPL) provide a way for group joining. Alternatively, the multicast protocol for low power and lossy networks (MPL) based on the trickle algorithm does not require a group joining and is designed to efficiently realize multicasting within LLNs.

VII. CONCLUSION

This paper presented an efficient group communication model based on Web services applicable in wireless automation systems. The communication model is incorporated as a custom oBIX service which is based on our proposed oBIX binding to IPv6, CoAP and EXI. Using the schema of the standard oBIX object model and direct EXI encoders/decoders for according platforms (e.g. Java), the resulting group communication model is as efficient as custom binary protocols found in nowadays building automation systems making it even attractive for the use in low-power and lossy wireless networks.

The proposed group communication model has been implemented in Java and published as open source in order to show the feasibility and to evaluate the performance compared to the client/server model. Furthermore, the reliability of the group communication model compared to the client/server model has been discussed for cloud-based and local control scenarios showing the increased reliability of the group communication model but leaving the possibility for a coexistence of both interaction scenario open.

As further work, we consider an implementation for Contiki and a thorough evaluation of the large scale effects of the group communication model using simulation models. Also, incorporating confirmed group communication facilities is planned.

ACKNOWLEDGMENT

Authors express their acknowledgement to the consortium of the project IoT6 (www.iot6.eu). The IoT6 project is supported by funding under the Seventh Research Framework Program of the European Union, with the grant agreement FP7-ICT-2011-7-288445.

REFERENCES