MOCAP: Towards the Semantic Web of Things

Kristina Sahlmann
HTW Berlin, University of Applied Sciences
Wilhelminenhofstraße 75A, 12459 Berlin, Germany
+49 30 5019 3626
kristina.sahlmann@htw-berlin.de

Thomas Schwotzer
HTW Berlin, University of Applied Sciences
Wilhelminenhofstraße 75A, 12459 Berlin, Germany
+49 30 5019 2604
thomas.schwotzer@htw-berlin.de

ABSTRACT

The original idea of the internet had a decentralized approach: every internet host sends data packages along to its neighbors if they are not addressed to itself. Every host is a sender and a receiver at the same time. Nowadays we are moving towards Internet of Things (IoT) and again all the small devices and sensors get decentralized nature. They exchange data with their environment and neighbors next to them. Data exchange requires a mutual understanding of exchanged data. Semantic approaches, namely a vocabulary help which leads to a Semantic Web of Things. This paper describes a proposal for the Micro-Ontology Context-Aware Protocol (MOCAP). Sensors can use micro-ontologies which are always context-aware and send this semantic information to other devices. The receivers gain valuable information using the semantic description and data about micro-ontology. Implementing M2M protocols on limited devices like sensors is challenging. We introduce an extension to MQTT and CoAP that exchanges data based on Micro-Ontologies. That leads to semantic interoperability even on the level of sensor grids.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols – *routing protocols*.

General Terms

Design, Experimentation, Standardization, Theory

Keywords

Internet of Things, Semantic Web, Ontology, M2M Protocol, MQTT, CoAP, Sensors

1. INTRODUCTION

At the 12. Google I/O conference 2015 Google introduced the new operation system named Brillo for the smart home and the Internet of Things (IoT). Google is also working on the Weave API which should be used for communication in the IoT. This API can be used cross-platform even on top of the existing stack. The Google's solution for the interoperability is furthermore a core set of schemas[1] for data exchange. Developers can extend this schema in terms of certified program which should guaranty that devices can exchange data seamlessly. This should improve the user experience. Every Android device recognize automatically any Brillo OS or Weave API based device. Users can choose a device, set it up and use it immediately.

That seamless user experience is also the idea developed by the Physical Web Project[2]. Every smart device has an URL. Users ask for devices nearby and get the list. Then they can use the URL in order to get more information. The project uses the URI Beacons schema.

This paper describes the decentralized approach for data exchange between small devices on the IoT using the low level M2M protocols. We combine the common technologies like RDF with constrained devices and their networks. The transported data is enriched by semantic description. We introduce the common M2M protocols and apply the idea of context-aware micro-ontology. M2M protocols has to follow some rules and restrictions.

2. RELATED WORK

There are several works on standardization of ontologies among them for sensors, too. The SSN ontology [2] by W3C describes high-level model for sensors, their capabilities, platform, observation etc. on behalf of OWL. But it does not address the units of measurement or other specific domain knowledge. There should be simple semantic definition for the sensor data exchange.

There is a draft for SenML [6] providing the lightweight protocol describing the media type i.e. for temperature sensor in protocols like HTTP or CoAP. They introduce several representations JavaScript Object Notation (JSON), eXtensible Markup Language (XML) and Efficient XML Interchange (EXI). But the markup is limited on sensors.

Other work in [12] proposes "a method of transforming sensor data into the RDF conforming to SSN ontology". They introduce an XML-based language for annotation of sensors.

The work presented in [11] proposes the design for lightweight ontology, linked IoT data, distributed semantic data storage and semantic service discovery and ranking. It shows some concepts to connect the sensors with the web and also use common web practices like linked data and web services. But it does not describe a M2M protocol binding or any semantic enhancements.

The researcher group developed a semantic engine for IoT [4]. They address some challenges for IoT and want to provide interoperability and integrate semantic web technologies among others. They introduce a semantic-based M2M architectures for ETSI M2M and oneM2M.

There is another work [5] analyzed the semantic usage on the IoT. They see the Semantic Web of Things (SWoT) as the next step after IoT and Web of Things. The goal of SWoT is to connect the physical and the digital world, and semantic is the key. They analyze the protocols for the SWoT application, and see the

[1] https://developers.google.com/brillo/

[2] http://google.github.io/physical-web/

MQTT still in the telemetry market, and CoAP is more applicable for IoT and higher level standards.

3. SCENARIO

The growth of the Internet of Things (IoT) has led to the spread of various small devices, including wearables, beacons, sensors etc. Every device can be seen as peer. Devices usually communicate directly – in a direct, a P2P manner. Devices don't have to be connected with Internet. Information exchange is usually performed between nearest devices with near-field protocols.

Devices can be i.e. a couple sensors and a smartphone. Sensors are constrained devices. They have small processors and little storage. But they have an operation system and network access. And they are built and used for the certain purpose (e.g. measure temperature). That purpose is the context and the devices are context-aware. In our scenario, sensors can measure different environment data. The smartphone next to it should collect and combine these information and present it users. This can occur as well in the push as in the pull way. Either users scan for the nearest sensor or will be notified. Sensors measure the temperature, the humidity and the brightness in the house and environment etc. Users get the information on smartphone but also the automatic blinds for the windows in their home get these information and close or open automatically. Users can control the blinds with the smartphone, too. Devices need a common understanding of exchanged data to work in that way. This scenario is outlined in the Figure 1.

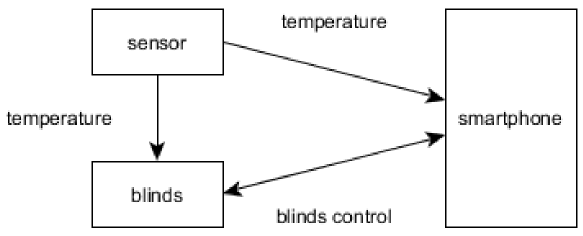

Figure 1. Context-aware data exchange.

Sensors have only less context-aware information (e.g. position, measured values). Thus, even small devices have to describe and exchange their context. In our model, information are only merged if their context matches. Each information is enriched with a micro-ontology data that describes that context. This is a controlled vocabulary. It can be a part of the top-level ontology.

Back to our scenario the sensor knows only the micro-ontology for its own measurement. Let's assume it measures the temperature. The smartphone knows this temperature micro-ontology either because users added it before or the smartphone has downloaded it from Internet in order to process the information. Now they can communicate.

In the home scenario the sensors measurement should tell what they measure and in which unit of measurement the results are. The micro-ontology of the sensor is a subset of the environment ontology. Thus the window blinds need to know this micro-ontology for the measurement but also for its control. The smartphone has knowledge about the environment ontology and can process data from the other devices like temperature, humidity or brightness sensors. And the smartphone knows the micro-ontology for the blinds control as a subset of home control ontology.

The way how these devices are going to exchange context-aware data based on micro-ontology we are calling MOCAP – micro-ontology context-aware protocol.

4. M2M PROTOCOLS

In these scenarios we have a machine to machine (M2M) communication. There are two M2M protocols specified for the Internet of Things (IoT): the Message Queuing Telemetry Transport (MQTT) and the Constrained Application Protocol (CoAP). They both work on level four of the Open Systems Interconnection model (OSI Model), which makes them better suitable for constrained environments than HTTP. Both protocols are open standards: MQTT v. 3.1.1 is an OASIS Standard [1] and CoAP is specified as Request for Comments (RFC) 7252 [9] by Internet Engineering Task Force (IETF). The MQTT has an extension for the sensor networks the MQTT-SN [10].

These protocols have different architecture and message formats. They have a mutual understanding of the M2M data exchange and constrained devices as well as IoT, though. We want apply a context-aware data based on micro-ontology on M2M protocols. Because it's difficult to say which protocol may win by the end, we take a look on all three protocols and analyze advantages and disadvantages.

We use the Resource Description Framework (RDF)[3] as a data format for communication. This is a standard for Semantic Web. Alternative we could use JSON-LD[4] as a standard for Linked Data, but for our home scenario is seems to be oversized. RDF use the URI for resource and properties identification. On the other hand RDF has several data representations among them N-Triples and Turtle. N-Triples are used i.e. by DBpedia. Turtle format seems to be more appropriated for the small devices because it separates the data representation into two parts: a list of prefixes and a list of the triples as shown in the Table 1. The data representation is more compact.

Table 1. RDF Turtle example.

@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> . @prefix sens: <http://environmentdata.com/sensor/> . @prefix meas: <http://xmlns.com/sensor/> .

sens:mysensor rdf:type temperature;
meas:units "degree Celsius";
meas:short "°C";
rdf:value "20.5" .

4.1 MQTT

First we take a look on MQTT. This protocol is designed for publish and subscribe messaging. It uses TCP/IP or "other network protocols that provide ordered, lossless, bidirectional connections" [1] for the transport. The protocol relies on client/server architecture paradigm.

The MQTT protocol defines fourteen types of Control Packets. Two of them are suitable for transporting semantic information: PUBLISH and SUBSCRIBE.

The SUBSCRIBE Control Packet contains the client subscription for one or more Topics. This packet is sent from the client to the server. Topics have Topic Names which can be structured by topic level separator as the forward slash ('/') and provide the hierarchical structure. In our case the smartphone and the blinds

[3] http://www.w3.org/standards/techs/rdf#

[4] http://json-ld.org/

are subscriber by the temperature sensor. The sensor must recognize the subscription Topic for micro-ontology. The subscription Topics should be the namespaces of the ontology and the micro-ontology. The PUBLISH packet may contain only one Topic. Therefore the sensor publishes the message twice: one with the ontology and one with the micro-ontology namespace. Our scenario with MQTT is shown in the Figure 2.

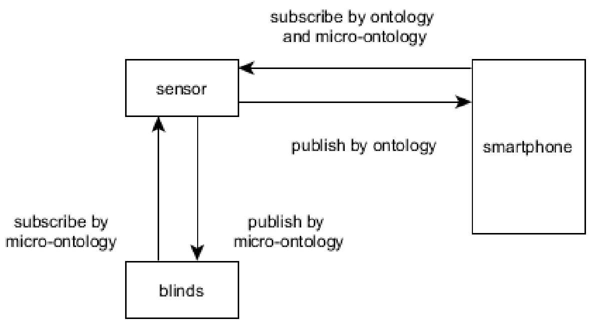
subscribe by ontology and micro-ontology
sensor
publish by ontology
smartphone
subscribe by micro-ontology
publish by micro-ontology
blinds

Figure 2. MOCAP based on MQTT protocol.

The PUBLISH Control Packet transports an Application Message. This packet can be send from a client to server or from a server to a client. The Application Message contains among others a variable header and a payload. The Topic Name is one field from the variable header. It identifies the topic of the payload. The payload contains the publish message. The MQTT Control Packet can have a size up to 256 MB [1] due to variable length encoding scheme. The payload in RDF format can contain N-Triples or Turtle. For constrained devices and networks the message payload should be reduced to the Turtle and consists only the triples. Using prefixes allows sending an ontology within a single package.

4.2 MQTT-SN

The MQTT-SN is specified by IBM. The protocol is intended for Wireless Sensor Networks (WSNs) and constrained devices with a limited processing and power capacity. This protocol addresses the lower transmission rate. In case of WSN based on IEEE 802.15.4 the packet length is limited to 128 bytes [10] for the whole message. Subtracting the overhead for security, etc. there will be left only the half-length for the payload. MQTT-SN doesn't necessary need the TCP/IP layer. The protocol can work on any network which supports "a bi-directional data transfer service between any node and a particular one (a gateway)" supporting the protocol. The protocol already supports UDP and Zigbee. The MQTT-SN extends the MQTT protocol optimizing it for constrained devices and networks.

Due to this limited capacities the Topic is replaced by a short topic id of two bytes. Clients must register their Topics with the gateway to get the corresponding topic id. Gateways mediate between MQTT-SN and MQTT. Furthermore pre-defined topic ids and short topics are introduced which don't require registration. The pre-defined topic is two byte long. The short topics have a fixed length of two octets. Both the pre-defined and short topics are used by PUBLISH message. We propose to use that structure to transmit prefixes of the micro-ontology namespaces in RDF Turtle format. The publish message contains only the triples of Turtle. MQTT-SN is more suitable for sensors and the semantic data can be reduced by using the RDF Turtle.

4.3 CoAP

The Constrained Application Protocol (CoAP) was developed for use with constrained networks and nodes [3]. The protocol "provides a request/response interaction model between application endpoints" [9]. It uses UDP or other datagram-oriented protocol like 6LoWPAN. The protocol can easily be integrated with Web over HTTP under some circumstances. CoAP-HTTP Proxy and HTTP-CoAP Proxy must be implemented. Service and resource discovery is supported beside multicast clients.

The Constrained RESTful Environments (CoRE) [8] uses the REST architecture paradigm. Clients and servers exchange data by means of GET, PUT POST and DELETE requests. CoAP endpoints can be both, client and server. Resources are identified by an URI with coap-prefix, e.g. "coap://server/temperature". Because of the underlying datagram-oriented transport and constrained network, the size of the request/response is limited to the datagram and IP packet size without fragmentation. For the UDP it results in 1024 bytes for the payload size [9]. In case of 6LoWPAN L2 the packets are limited to 127 bytes including overhead.

For our scenario the sensor have a server role, and the smartphone and blinds have the client role to succeed the GET request for the temperature, see Figure 3.

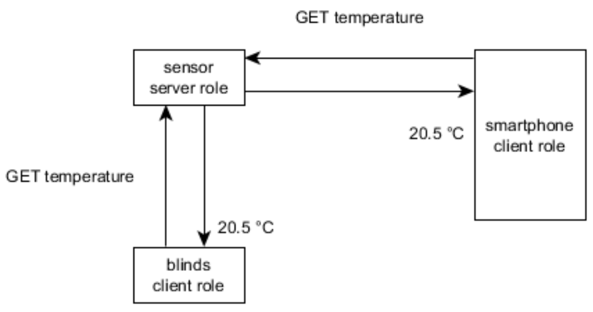

Figure 3. Scenario with CoAP.

The most common request is the GET request. It retrieves information from the current resources. On success a 2.05 (Content) response code should be presented in the response. The payload content has to indicate the content-format of the payload in order to simplify the message processing. There is a sub-registry for the subset of Internet media types which can be used by CoAP as a numeric identifier. For an example "application/xml" has the identifier "41" [9]. The payload itself has a very limited size for transporting RDF enriched semantic data. As we already have seen, the Turtle could be divided into two parts: the prefix definitions and the triples. Thus we could transfer only the triples assuming that clients know those prefixes.

There are two ways in CoAP how the endpoints get connected: either by service discovery or by multicast. In case of service discovery the client knows (or learns) the server's address. The resource discovery offered by the CoAP endpoint proceeds in machine-to-machine way. For more interoperability the endpoints should support the Constrained RESTful Environments (CoRE) Link Format [8] of resources.

Using the entry point clients get the response with a payload in the CoRE Link Format. It consists of resources hosted by the server, i.e. a list of environment sensors i.e. for temperature, humidity, etc. There are several examples described in the

RFC6690 [8]. They consider a server with two resources: for temperature and humidity. The GET request returns a list of these resources, see Table 2.

Table 2. CoAP GET request and response.

REQ: GET /.well-known/core
RES: 2.05 Content </sensors>;ct=40;title="Sensor Index", </sensors/temp>;rt="temperature-c";if="sensor", </sensors/light>;rt="light-lux";if="sensor"

The resource URI could be the namespace of the micro-ontology which is known by the sensor. The attribute "rt" describes the resource type. In this case it is the unit measurement. The attribute "if" describes the interface of the resource which is sensor. The client can process this semantic data and match it to its known ontologies.

In the multicast CoAP the endpoints listen on the default CoAP port in order to offer services to multicast endpoints. This process is described by RFC 7390 [7]. After they have received a multicast request they can process the message or ignore it. The message can contain the semantic information about micro-ontology of the client. The endpoint matches these information to its known ontologies and process it. Every message is identified by Message ID used to detect duplicate messages. The request may include further options and among them the client URI.

The CoAP protocol is more adapted for constrained nodes and networks. The MOCAP protocol can be setup on top of it because the nodes are context-aware and have limited vocabulary. The real challenge is to reduce the semantic data for the limited capabilities. The micro-ontology must be still recognizable and the support for RDF included.

5. CONCLUSION AND FUTURE WORK

This paper shows the feasibility of the concept. The constrained devices like sensors are context-aware because they have a certain purpose or task i.e. measure the temperature in degree Celsius and share it with other devices. This is their context. They only need to know their own micro-ontology and the namespace of the top-level ontology.

The micro-ontology is a subset of the top-level ontology. As there are more capable devices like smartphones participating in data exchange, they have knowledge about the top-level ontology and can process the data. In our scenario the top-level ontology is about environment, and the micro-ontology is about the temperature.

The semantic data should be described in a standardized way i.e. by RDF or JSON-LD. The Turtle representation of data is more compact then other RDF formats. The length of a message is limited. The challenge is to reduce the overhead caused by semantic description. This can be done by splitting the Turtle in prefixes definition and the triples themselves. Assuming the more capable device knows the prefixes we transport only the triples.

We took a closer look at three M2M protocols: MQTT, MQTT-SN and CoAP. We applied the principles above. Every protocol follows another architecture paradigm or has a different intension in sense of nodes or networks. Anyway we could apply our principles to them all.

Summarizing we call this approach MOCAP – micro-ontology context-aware protocol. The next step will be a case study with some use cases and different device classes. We need to evaluate the micro-ontology, its size etc. We follow the work of RDF Stream Processing Community Group (RSP)[5] and Web of Things (WoT)[6] community groups at W3C.

6. REFERENCES

[1] *MQTT Version 3.1.1. Edited by Andrew Banks and Rahul Gupta. 29 October 2014. OASIS Standard. http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html. Latest version: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html.*

[2] *Semantic Sensor Network XG Final Report. Edited by Laurent Lefort, Cory Henson and Kerry Taylor. W3C Incubator Group Report 28 June 2011. http://www.w3.org/2005/Incubator/ssn/XGR-ssn-20110628/. Latest version: http://www.w3.org/2005/Incubator/ssn/XGR-ssn/.*

[3] Bormann, C., Ersue, M., and Keranen, A. 2014. Terminology for Constrained-Node Networks. RFC 7228. *RFC Editor*, May 2014. DOI=10.17487/RFC7228.

[4] Gyrard A., Datta S. K., Bonnet C., Boudaoud K. 2015. *A Semantic Engine for Internet of Things: Cloud, Mobile Devices and Gateways.* http://www.eurecom.fr/en/publication/4555/download/cm-publi-4555_1.pdf.

[5] Jara, A. J., Olivieri, A. C., Bocchi, Y., Jung, M., Kastner, W., and Skarmeta, A. F. 2014. Semantic Web of Things: an analysis of the application semantics for the IoT moving towards the IoT convergence. *IJWGS* 10, 2/3, 244–260. DOI=10.1504/IJWGS.2014.060260.

[6] Jennings C., Shelby Z., Arkko J. 2015. Media Types for Sensor Markup Language (SENML).

[7] Rahman, A. and Dijk, E. 2014. Group Communication for the Constrained Application Protocol (CoAP). RFC 7390, October 2014. DOI=10.17487/RFC7390.

[8] Shelby, Z. 2012. Constrained RESTful Environments (CoRE) Link Format. RFC 6690, August 2012. DOI=10.17487/RFC6690.

[9] Shelby, Z., Hartke, K., and Bormann, C. 2014. The Constrained Application Protocol (CoAP). RFC 7252, June 2014. DOI=10.17487/RFC7252.

[10] Stanford-Clark A., Truong H. L. 2013. MQTT For Sensor Networks (MQTT-SN). Protocol Specification, Version 1.2.

[11] Wang, W., De, S., Cassar, G., and Moessner, K. 2013. Knowledge Representation in the Internet of Things: Semantic Modelling and its Applications. *Automatika Journal* 54, 4, 388–400. DOI=10.7305/automatika.54-4.414.

[12] Zhang X., Zhao Y., Liu W. 2015. Transforming Sensor Data to RDF based on SSN Ontology. *Advanced Science and Technology Letters* 2015, 81, 95–98. DOI=10.14257/astl.2015.81.20.

[5] https://www.w3.org/community/rsp/

[6] http://www.w3.org/WoT/