Efficiently observing Internet of Things Resources

Girum Ketema, Jeroen Hoebeke,
Ingrid Moerman, Piet Demeester
Information Technology Department
Ghent University - iMinds
Ghent, Belgium
{firstname.lastname}@intec.ugent.be

Li Shi Tao
Huawei Technologies
Huawei Base
Nanjing, Jiangsu, China
lishitao@huawei.com

Antonio J. Jara
Department of Information Technology
and Communications
University of Murcia
Murcia, Spain
jara@um.es

***Abstract*—The Constrained Application Protocol (CoAP) is a lightweight protocol that enables the implementation of RESTful embedded web services. Observe is one of the CoAP extensions, which allow servers to send every resource state change to interested clients. In this paper we present an interesting extension to the observe option, called conditional observation, where clients specify notification criteria along their observation request. We evaluate the feasibility of implementing this on a constrained device and evaluate the correct operation for a simple scenario. It is shown that the use of conditional observations can result in a reduced number of packets and power consumption compared to normal observe in combination with client-side filtering.**

Keywords - Conditional Observation, IoT, REST, CoAP

I. INTRODUCTION

Smart Objects have been in use for quite a while to interact with the real world and communicate the information to hosts connected to the Internet. They usually have limited bandwidth, processing and storage capacity [1]. The existing Internet protocols and applications are too heavy to be used directly on these objects. Recently, an IETF working group, called Constrained RESTful Environments (CoRE), has been founded specifically to work on the standardization of a framework for resource-oriented applications, allowing realization of RESTful embedded web services in a similar way as traditional web services.

Their work resulted in the Constrained Application Protocol (CoAP). CoAP provides a compact transfer protocol on top of UDP that realizes exactly the subset of HTTP methods (GET, PUT, POST and DELETE) that is necessary to offer RESTful web services in a Wireless Sensor Network (WSN)-compatible manner [2]. A simple mapping between HTTP and CoAP can be realized (and vice versa). CoAP can run on top of 6LoWPAN networks, but also on top of proprietary networks that are connected to IPv6 Internet. The details of the CoAP specification can be found in [2].

In addition to the main CoAP draft, a number of extensions have been proposed. One of those extensions is the observation of resource states through the introduction of the observe option, which allows clients to register with servers to be notified whenever the state of a resource changes. A client interested in observing a resource includes the option in its GET request. Whenever there is a change of the resource state, the server sends a notification to the client. As such, observe offers the possibility for a client to have an up-to-date representation of the resource without the client having to constantly poll for changes. If the client acts upon these states and is only interested in specific states, it is up to the client to filter out the values sent by the server, discarding resource states that are not significant enough for its purpose.

A better alternative to these observations in combination with client-side filtering could be to specify filtering criteria when sending the observe request. This way, the server sends a notification only when it meets the particular criteria. In this paper we will present such an extension of the observation functionality by allowing notification criteria to be specified along with observe requests. This approach will provide a built-in mechanism to the CoAP protocol to allow transfer of states of interest, rather than transferring all states.

As such, this paper contributes to further extend the CoAP protocol by providing a new, lightweight extension to publish values or events to interested subscribers, which is built into the protocol as an option. We also show the feasibility of implementing this new option on a constrained device with less than 64KB of memory. In addition, we also demonstrate the relevance of the new option compared to the use of the currently built-in observation mechanism in combination with client-side filtering. To the best of our knowledge, this is the first implementation to extend the CoAP protocol to allow conditional observation. It is clear that this paper does not try to compare performance of the new solution against normal observation.

Section 2 of the paper presents normal observation and its limitation. Section 3 introduces the new Condition option and our approach. The fourth section presents our implementation and evaluation results. Finally, the paper will conclude after related work is discussed.

II. MOTIVATION

One of the optional extensions of CoAP is observing the state changes of a resource as stated in [3]. The observation functionality has considerable importance in many applications such as home automation, building control and environmental monitoring. Figure 1, below shows a normal observation request and response communication between a client and a server. The client could be a smart object

responsible to switch on and off an air conditioning system while the server is a node with a temperature sensor.

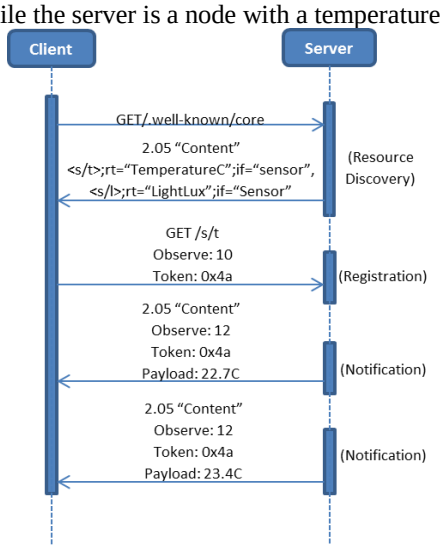

Figure 1: Normal Observe

In case of normal observation, if the client would like to switch on an air conditioning system depending on current temperature, it has to send an observe request to the server. Whenever the temperature changes, the node sends a notification to the client. However, the client will not do anything with that information unless the temperature is above a pre-determined threshold. Therefore, many packets received from the server are just wasted. Because filtering and processing happens by the client, this approach has a major impact on bandwidth and requires extra processing of packets that are not going to be used by the client.

To avoid this, authors of this paper have proposed a new CoAP option "Condition" as an extension to the Observe Option in order to support conditional observations [4]. This option can be used by a CoAP client to specify the conditions the client is interested in. Now, the CoAP server will send a notification response with the latest state change only when the criterion is met. In our earlier example, the client may say, "send me a notification only if the temperature is above 25^0C". Figure 2 shows the operation of conditional observation.

This approach is different from other similar works such as the CoRE interfaces draft detailed in [8]. Here, conditional observation requests are represented by URI queries. An important problem with this approach is its complexity. The queries that are generated have limited readability and could be difficult to represent. Furthermore, URI queries are very resource specific complicating automatic processing of conditional observations or code reuse over several resources. Using a CoAP Option for conditional observations makes this functionality independent of any specific resource implementation, whereas URI queries can be used for resource specific functionalities. Further, the link with the Observe option is lost by spreading this functionality over both URI queries and options and the multitude of URI queries that can occur makes it more complex for intermediaries to process this information.

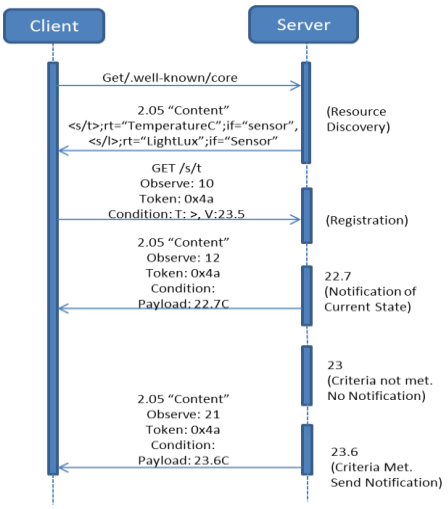

Figure 2: Conditional Observation

III. APPROACH

In this section, we will briefly present the new CoAP Option called Condition Option in order to support conditional observations. A detailed description can be found in [4]. This option has to be used in combination with the Observe option and can be used both in request and response messages. In a GET request message, the Condition option represents the condition the client wants to apply to the observation relationship. It is used to describe the resource states the client is interested in.

The Condition option is an elective option with length between 1 and 5 bytes. This option has a header and a value component. The header consists of Observation Type (5 bits), Value Type (2 bits) and Reliability Flag (1 bit). The value field could be between 0 and 4 bytes in size. The observation type field contains the type of condition the client is interested in. Using 5 bits, up to 32 different observation types can be specified. Currently some commonly occurring filtering options are identified based on realistic use cases including, time series, maximum response time, minimum response time, step, All Values Less Than, All Values Greater Than, Value Equal, Value Less Than Greater Than, and Periodic. The detailed description of these observation types can be found in [4].

IV. IMPLEMENTATION AND EVALUATION

Our implementation of conditional observations is based on ***Erbium*** – a low-power REST Engine for Contiki [5]. We extended this CoAP implementation to support the new Condition Option and provided some resources that allow conditional observations.

A. Experiment Setup

We used Sky sensor nodes in Cooja to run all our tests. The Sky nodes have an MSP430 16-bit CPU running at 3.9MHz with CC2420 radio chip. Sky nodes are highly constrained in memory having only 48kB of program memory and 10 kB RAM for data. We used one border router, one client, a few server nodes and variable numbers

of intermediate nodes with RPL as a routing protocol. The root for the RPL network is the border router. For different tests we used different hop counts and different number of servers. We used X-MAC as Radio Duty Cycling (RDC) protocol with 16 Hz wake-up frequency and CSMA as MAC layer protocol so that packets lost due to collision are retransmitted. The well-known Contiki power profiler, powertrace [6], was used to compute power consumption. In addition to power consumption, we also collected the number of packets transmitted in the network. As a proof of concept we used the ***"AllValuesGreaterThan"*** condition type for the experiment

To prove the significance and added value of conditional observations, we computed the power consumption and the number of packets generated for normal observations and conditional observations by using different scenarios. In a first set of experiments, we only used 1 CoAP server and 1 CoAP client with the number of intermediate nodes varying from 0 to 5. Next, we conducted several similar experiments by keeping the hop count to 4 and using two servers. In all cases, we used 100 pseudo-random integers between 20 and 29 to be sent to clients as temperature sensor readings. The average of the numbers is 24 and the values change every 4 or 8 seconds. To evaluate the impact of the condition value, we repeated each experiment 10 times with the condition value increasing from 20 through 29. We also repeated all experiments by sending the requests as confirmable and non-confirmable. In all scenarios, every experiment runs for 444 seconds.

B. Scenario 1 – Single Client and Single Server

To validate the correctness of the implementation, we set up one client and one server without intermediate nodes. We first run the experiment for normal observe and then for conditional observe with the condition values changing from 20 to 29. We measured the number of packets transmitted and the power consumption.

From the experiment we found out that the overall power consumption for normal observation was 2.25mW while that of conditional observation reduces from 2.24mW to 2.1mW as we go to the more extreme thresholds (Figure 3).

The results showed the correct operation of the implementation. The results also prove that the reduced power consumption of conditional observation, due to reduced number of packet transmissions, can be significant compared to transferring all state changes in combination with client-side processing. Of course, the real gain will depend of course on the resource states the client is interested in.

C. Scenario 2- One client and one server multiple hop connection

The next sets of experiments are done for multiple hops between the client and the server.

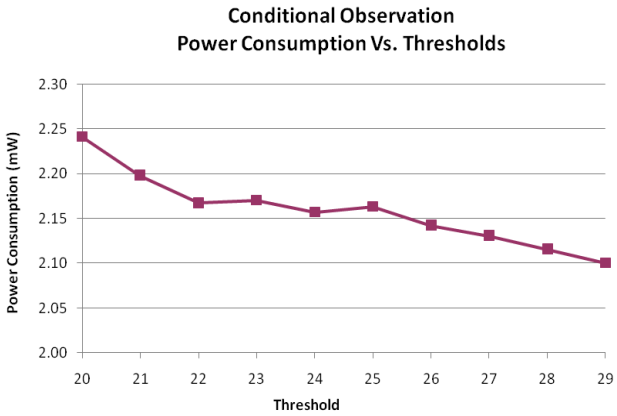

Figure 3: Power Consumption vs. Threshold of Conditional Observation

1) Power Consumption

To evaluate the power consumption, we ran the simulation with the number of hops between the client and the server increasing from 0 to 5. We did this both for normal observe (assuming filtering at the client) and for conditional observe. For the latter, we run the experiment three times with two extreme conditions and an average condition value. As it can be seen from the figure below, the average per-node power consumption for normal observe and client-side filtering is higher than that of conditional observation. The increase in power consumption is mainly due to the power consumed for transmission of mostly unimportant packets. It is also interesting to see that the power consumption gap between normal observe and conditional observe gets larger with larger network sizes.

In most cases, Internet of Things objects will only have a limited power supply, usually batteries. To compare the battery lifetime, we assumed our IoT nodes are using two Lithium AA batteries (in series, providing 3 Volts) with a capacity of 8820 Ampere-Second. Therefore, the batteries will have 26460 Joule (Watt/sec) total capacity. Figure 5 below shows the percentage increase of the resulting battery lifetime, expressed in days. Since power consumption is lower for conditional observations, there is a considerable increase in battery life.

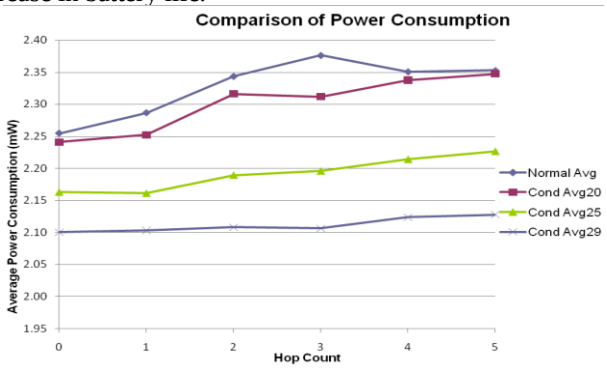

Figure 4: Per-node average power consumption for normal observation and conditional observation (3 Thresholds)

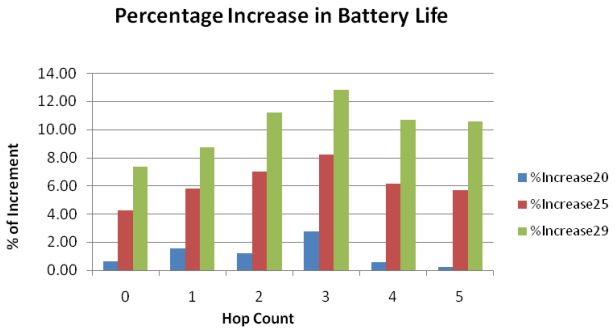

Figure 5: Percentage increase in battery life

In most cases, Internet of Things objects will only have a limited power supply, usually batteries. To compare the battery lifetime, we assumed our IoT nodes are using two Lithium AA batteries (in series, providing 3 Volts) with a capacity of 8820 Ampere-Second. Therefore, the batteries will have 26460 Joule (Watt/sec) total capacity. Figure 5 below shows the percentage increase of the resulting battery lifetime, expressed in days. Since power consumption is lower for conditional observations, there is a considerable increase in battery life.

2) Number of Packets Transmitted.

One of the scarce resources of constrained environments is bandwidth. Therefore, the number of packets transmitted is an important parameter to see the impact of conditional observations on the number of packets transmitted. For this purpose, we also measure the number of packets transmitted for every threshold (between 20 and 29) and every hop count (0 to 5).

It is clear that the average number of packets transmitted will be less for conditional observation. It is interesting to see that the difference between normal observe and conditional observe gets higher for higher hop counts.

D. Other Scenarios

We have also tested several other scenarios by sending packets as confirmable and non-confirmable; using multiple servers with multiple hops. In all cases, the gain of using conditional observation can be quite significant. Therefore, depending on the use case, it proves to be a good optimization for the CoAP protocol.

V. Related Work

There are a number of research activities under way on WSN and IoT. Different groups are using different approaches to come up with outstanding solutions and technologies. The Open Geospatial Consortium (OGC) Inc. developed the Sensor Observation Service (SOS) standard that deals with the specifications of data observation from different sensors in different, possibly geographically scattered, sensor networks [7]. The standard specifies that a GetObservation request may have several mandatory and optional parameters. One of the optional parameters is ***featureOfInterest***, which is similar to our observation type. However, this approach is more focused for geographical observations and is a subset of a bigger framework, which significantly differs from the IETF recommendation. The other related work is CoRE Interfaces proposed by Zack Shelby, as discussed in Section 2 of this paper.

VI. Conclusion

In this paper we presented and implemented the concept of conditional observations as an extension to the CoAP protocol in general and the Observe option in particular. Our implementation shows the feasibility of implementing this functionality on very constrained devices. We also presented comparative results of using normal observation in combination with client-side filtering versus conditional observations. From the results, it is evident that the conditional observations are a useful extension, both from an application point of view and from a network efficiency point of view. It enables clients to receive notifications that contain only state changes they are interested in. This has a twofold advantage: an application has the expressiveness to selectively collect data and the data of no interest does not have to travel over the network. The latter advantage will become even more important in larger constrained networks where notifications have to travel over multiple hops. As such, conditional observations can greatly contribute to the reduction of battery consumption and increase of network lifetime. Of course, the concrete gain will depend on the conditions of interest and thus the actual use cases: more extreme conditions will lead to larger gains.

In the future, we will continue the implementation to support condition types and do an evaluation based on real scenarios. We will also further evaluate run-time overhead and trade-off between reduced number of packets and processing requirement.

Acknowledgement

The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n°258885 (SPITFIRE project), from the IBBT ICON projects GreenWeCan and O'CareCloudS

References

[1] Vasseur, J. P. a. (2010). *Internetworking Smart Objects with IP*. Burlington: Elsevier Inc.

[2] Z. Shelby, K. H. (2012, June 25). *draft-ietf-core-coap-10*.

[3] Hartke, K. (2012, March 12). *draft-ietf-core-observe-05*.

[4] Shi Tao. Li, J. H. (2012, June). *draft-li-core-conditional-observe-02* .

[5] M. Kovatsch, S. D. (2011). A Low-Power CoAP for Contiki. Proceedings of the 8th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS 2011), (pp. 855-860). Valencia.

[6] A. Dunkels, J. E. Powertrace: Network-Level Power Profiling for Lowpower Wireless Networks. 2011.

[7] Open Geospatial Consortium Inc. (2007). *Sensor Observation Service*. Open Geospatial Consortium Inc.

[8] Shelby, Z. (2012, July 11). *CoRE Interfaces*. *draft-shelby-core-interfaces-03*