An Energy Aware Resource Design Model for Constrained Networks

N. Correia, G. Schütz, A. Mazayev, J. Martins and A. Barradas

***Abstract*—The Internet of Things (IoT) is expected to incorporate objects and sensor networks of all kinds, and in particular constrained sensor networks where energy consumption is a critical issue. In order to increase the lifetime of such networks, intelligent and standard-based solutions should be used. Here we address this challenge through the use of CoRE interfaces for resource design. These interfaces allow the server side to compose/organize resources and the client side to discover and determine how to consume such resources, besides allowing decisions to be easily integrated into the operation of the network. An energy aware resource design model is proposed, based on CoRE interfaces, for the design of resources matching client needs. Based on this model we develop an algorithm that proved to be energy efficient.**

***Index Terms*—Internet of Things, constrained networks, energy saving, CoRE interfaces.**

I. Introduction

Sensors can be organized in networks and, as sensor networks become more and more part of our lives, better ways to collect data and distribute it over the network are needed. For this purpose, the Constrained RESTful Environments (CoRE) working group, within Internet Engineering Task Force (IETF), developed a Web application transfer protocol called Constraint Application Protocol (CoAP) [1]. Later, an extension to CoAP, called Observe, has been proposed to give clients the ability to observe resource changes [2], [3].

CoRE realizes the Representational State Transfer (REST) architecture in a suitable form for constrained nodes. In such environments, CoRE based applications will most likely need to discover resources hosted by constrained servers, referred to as CoRE Resource Discovery [4], [5]. Besides the specification of Web discovery and linking for constrained environments, a set of Interface Types for resource design is now on an ongoing standardization process [6]. These Interface Types allow the server side to compose/organize resources and the client side to discover and determine how to consume such resources.

In this article we propose the use of CoRE Interfaces for energy saving purposes, allowing the lifetime of sensor networks to increase, and an energy aware resource design model is developed. Based on this model we create an algorithm, for resource design and resource observation planning, that proved to be energy efficient. When compared with other approaches this has the advantage of requiring less control messages for resource design/reconfiguration and set up of observations, reducing even more energy consumption, besides being a standard-based solution.

Authors are with CEOT research center. N. Correia and A. Barradas ({ncorreia,abarra}@ualg.pt) are with the Faculty of Science and Technology and G. Schütz (gschutz@ualg.pt) is with Institute of Engineering, both at University of Algarve, Portugal.

The remainder of this article is organized as follows. In Section II the ongoing work on CoRE Interfaces for resource design is discussed. Section III presents the motivation, the resource assignment model and algorithm. In Section IV results are discussed, while in Section V some conclusions are drawn.

II. Interfaces for Resource Design

A set of Interface Types for resource design patterns has been proposed in [6]. These Interface Types allow the server side to compose/organize resources, and the client side to discover and determine how to consume such resources, which can be of Collection type.

A. Collections

A Collection is a resource representing one or more related resources. Collections allow resources to be organized for discovery, and for various forms of bulk interaction according to the Interface Type used. A Collection Interface Type defines the associated content format, data types, URI templates, REST methods, parameters and responses. The Interface Types discussed in [6] include: Link List, Batch, Linked Batch, Hypermedia Collection and Binding.

B. The Linked Batch Interface

One of the Interface Types defined in [6] is the Linked Batch. This interface, specified by link parameter *if=”core.lb”*, allows the content of a Collection to change dynamically according to the control of a Web client. A request with a POST method and a CoRE Link Format content appends links to the Collection, a request with a PUT method allows the Collection to be updated, and DELETE removes the entire Collection. Assuming that the following Collection needs to be created/filled to be used as a resource,

```
</list/>;rt="1stfloor";if="core.lb"
```

the following POST would add two resources to Collection "$</list/>$", while the GET obtains a single SenML data object including both resource values.

```
REQ: POST /list/
 ct=40,
     </s/light>,</s/temp>

RES: 2.04 Changed

REQ: GET /list/

RES: 2.05 Content
 ct=40,
     {"e":[
      { "n": "/s/light", "v": 123, "u": "lx" },
      { "n": "/s/temp", "v": 27.2, "u": "degC" }]
     }
```

The possibility of building/updating Collections dynamically, through interfaces, allows intelligent decisions to be done regarding the design of Collection resources and allows easy integration of such Collection resources into the network. Here in this article this is done with the objective of saving energy.

III. Applying Collections and Link Batch Interface to the Energy Saving Problem

When using CoAP/Observe, a client may observe a resource through multiple client-to-server registration steps because proxies can be introduced for scalability purposes. This means that the registration steps from multiple observations, can be carefully planned so that notifications are aggregated at intermediate proxies, as proposed in [7], [8]. This leads to energy saving and better bandwidth utilization, increasing the network lifetime and reducing delay. Here we optimize registration steps, as in [7], [8]. However, we apply CoRE Interfaces and Collections to build the aggregates (in our case Collections). The advantages are:

- Less control messages are used to set up/change registrations since Collections become available as resources, which can be observed. In [7], [8], registration is done individually for each subject inside an aggregate. Thus, the set up/change of registrations requires multiple individual messages (per subject) to be sent, increasing energy consumption. The use of less control messages allows re-optimization procedures to be more frequent.
- The Linked Batch Interface is used for Collection creation/update, enabling the dynamic control by an entity where the optimization of registration steps, using Collections, would be done. This allows optimization decisions to be more easily integrated into the network.

As far as known no other work on CoAP/Observe registration planning exists, besides [7], [8].

A. Resource Design Model

1) Definitions: Let us assume a constrained network with a set of nodes and wireless links denoted by $\mathcal{N}$ and $\mathcal{L}$. A resource hosted by node $n \in \mathcal{N}$ can be:

- a resource element per se, locally produced or in cache as a consequence of n being an observer;
- a locally created/produced Collection of resource elements built for the purpose of reducing energy consumption.

Furthermore, such resource elements can be atomic or aggregate resource elements (e.g. a resource element in cache, or included in a Collection, can be the outcome of an observation made to another Collection).

Definition 1 (Resource Producer): A node $n \in \mathcal{N}$ is a resource producer if it has locally created/produced resources. The set $\mathcal{R}(n)$ includes all the resources produced at $n \in \mathcal{N}$, and r is an element of such set.

Definition 2 (Resource Element Type): A resource $r \in \mathcal{R}(n)$ at node $n \in \mathcal{N}$ can be of Atomic or Aggregate type, $T(r) = \{t : t \in \{Atomic, Aggregate\}\}$, and aggregate resources include more than one Atomic or Aggregate element. That is, $T(e_i) = \{t : t \in \{Atomic, Aggregate\}\}$, $\forall e_i \in r$. Resource elements have a weight factor, $W(r)$, that is related with the size of their notifications. More specifically, this will be $W(r) = \delta + data(r)$, where δ is a constant notification envelop factor (same in all notifications) and $data(r)$ is a varying data content factor (grows with number of data values in notification).

Definition 3 (Resource Consumer): A resource consumer is any node $n \in \mathcal{N}$ that has a set of sensor resource interests, denoted by $\mathcal{I}(n)$. A resource interest $i \in \mathcal{I}(n)$ is assumed to be of Atomic type.

Definition 4 (Observation): A node $n \in \mathcal{N}$ is responsible for a set of observations on resources, denoted by $\mathcal{O}(n)$. An observation $o \in \mathcal{O}(n)$ regards to a resource included in $\cup_{n \in \mathcal{N}} \mathcal{R}(n)$ and has a source node $src(o) \in \mathcal{N}$ associated with it. Node n may be the resource consumer, wishing to meet interests in $\mathcal{I}(n)$, or an intermediate/proxy node. The node $src(o)$ may be the producer of the resource or a proxy node.

2) The Model: Let us assumed that $\mathcal{R}^U$ is a universe set including all feasible resource assignment solutions. That is, $\mathcal{R}^U = \{\mathcal{R}_1, \mathcal{R}_2, ..., \mathcal{R}_{|\mathcal{R}^U|}\}$ and $\mathcal{R}_i = (\mathcal{R}_i(n_1), \mathcal{R}_i(n_2), ..., \mathcal{R}_i(|\mathcal{N}|)),\ \forall i \in \{1, ..., |\mathcal{R}^U|\}$. Thus, each element in $\mathcal{R}^U$ is a feasible resource assignment containing the resources produced by each node. This means that feasible solutions may include locally produced Atomic resources and Collections created for energy saving purposes.

Let us assume also that $\mathcal{O}^U$ is a universe set including all feasible observation assignments in the network. That is, $\mathcal{O}^U = \{\mathcal{O}_1, \mathcal{O}_2, ..., \mathcal{O}_{|\mathcal{O}^U|}\}$ and $\mathcal{O}_j = (\mathcal{O}_j(n_1), \mathcal{O}_j(n_2), ..., \mathcal{O}_j(|\mathcal{N}|)),\ \forall j \in \{1, ..., |\mathcal{O}^U|\}$.

Based on the previous information, the following cost function $f : \mathcal{R}^U \times \mathcal{O}^U \to \Re^+$ is introduced:

$$f(\mathcal{R}_i, \mathcal{O}_j) = \sum_{n \in \mathcal{N}} \sum_{r \in \mathcal{R}_i(n)} CF(r, \mathcal{O}_j) \tag{1}$$

where

$$CF(r, \mathcal{O}_j) = \sum_{n_k \in \mathcal{N}} \sum_{o \in \mathcal{O}_j(n_k) : r = o} OWD(src(o), n_k) \times \times \sum_{e_i \in r} W(e_i) \tag{2}$$

The $OWD(n_i, n_j)$ value is the one-way delay from n_i to n_j and $W(e_i)$ is the weight factor of resource element e_i. That is, the $CF(r, \mathcal{O}_j)$ accounts for the cost associated with all observations on resource r in the network. Then the best resource design (allocation of resources to producers and set of observations operating on the network) is given by:

$$(\mathcal{R}_i, \mathcal{O}_j)^* = argmin_{\mathcal{R}_i \in \mathcal{R}^U, \mathcal{O}_j \in \mathcal{O}^U}\{f(\mathcal{R}_i, \mathcal{O}_j)\} \quad (3)$$

Each $\mathcal{O}_j \in \mathcal{O}^U$ must verify the following condition: $\forall n \in \mathcal{N}, \mathcal{I}(n) \subset \cup_{o \in \mathcal{O}_j(n)} \mathcal{X}(o)$, where $\mathcal{X}(o) = \{e_i | e_i \in o\}$. This means that observations being performed on the network must match the interests of client nodes. Note that a node can be client for a set of resources and server for another set of resources, meaning that multiple scenarios (e.g. M2M) are being considered.

3) Properties: The previous collection assignment model has the following properties.

Property 1 (Local Optimal Observation): Given a feasible resource assignment $\mathcal{R}_i \in \mathcal{R}^U$, if $\exists \mathcal{O}_j^* \in \mathcal{O}^U : f(\mathcal{R}_i, \mathcal{O}_j^*) \leq f(\mathcal{R}_i, \mathcal{O}_k), \forall \mathcal{O}_k \in \mathcal{O}^U, \mathcal{O}_k \neq \mathcal{O}_j^*$ then $\mathcal{O}_j^*$ is a local optimal observation for the resource assignment $\mathcal{R}_i \in \mathcal{R}^U$.

Property 2 (Optimal Observation): Given a local optimal observation $\mathcal{O}_j^*$ for a feasible resource assignment $\mathcal{R}_i \in \mathcal{R}^U$, if $f(\mathcal{R}_i, \mathcal{O}_j^*) \leq f(\mathcal{R}_k, \mathcal{O}_l), \forall \mathcal{R}_k \in \mathcal{R}^U, \mathcal{R}_k \neq \mathcal{R}_i, \forall \mathcal{O}_l \in \mathcal{O}^U, \mathcal{O}_l \neq \mathcal{O}_j^*$, then $\mathcal{O}_j^*$ is an optimal observation.

B. Algorithm

The following algorithm is proposed to implement and validate the resource design model previously discussed. This includes the following steps.

1) Step 1: Define an initial resource assignment, $\mathcal{R} = (\mathcal{R}(n_1), \mathcal{R}(n_2), ..., \mathcal{R}(|\mathcal{N}|))$, including Atomic resources (already available at producer nodes) only.

2) Step 2: Create list of possible Collections. A Collection may include Atomic resources or other Collection resources. This is done as follows.

```
$\mathcal{C} = \emptyset$ /* list of collections */;
$max\_resource\_size = 1$;
repeat
    for each $(n_i, n_j), \forall n_i, n_j \in \mathcal{N}$ do
        /* available resources include atomic resources and possible collections at list */;
        $\mathcal{R}'(n_i) = \mathcal{R}(n_i) \cup \{r \in \mathcal{C} : host(r) = n_i\}$;
        $\mathcal{R}'(n_j) = \mathcal{R}(n_j) \cup \{r \in \mathcal{C} : host(r) = n_j\}$;
        for each $(r_i \in \mathcal{R}'(n_i), r_j \in \mathcal{R}'(n_j)))$ such that $\exists n_k \in \mathcal{N} : \{r_i \cup r_j\} \subset \mathcal{I}(n_k) \wedge (size(r_i) \geq max\_resource\_size || (size(r_j) \geq max\_resource\_size)$ do
            if $OWD(n_i, n_j) \times [\delta + data(r_i)] + OWD(n_j, n_k) \times [\delta + data(r_i) + data(r_j)] < OWD(n_i, n_k) \times [\delta + data(r_i)] + OWD(n_j, n_k) \times [\delta + data(r_j)]$ then
                /* define new collection resource and node hosting it */;
                $resource = \{r_i, r_j\}$;
                $host = n_j$;
                /* add to list if collections */;
                $\mathcal{C} \leftarrow c = \{resource, host\}$;
            end
        end
    end
    $max\_resource\_size = max\_resource\_size + 1$;
until $\mathcal{C}$ does not change ;
```

3) Step 3: Gradually insert Collections from list into resource assignment and see if best feasible observation assignment reduces energy consumption. If so, Collections are kept as resources, otherwise are removed. This is done as follows:

```
$cost = FeasibleObsAssignment(\mathcal{R})$;
for $c \in \mathcal{C}$ do
    /* collection inserted as virtual node; it is a consumer (interests same as collection content) and a producer (of aggregate resource). */;
    $\mathcal{N} = \mathcal{N} \cup \{c\}$;
    $\mathcal{L} = \mathcal{L} \cup \{l = (c, n) : n = dst(l') \wedge host(c) = src(l'), l' \in \mathcal{L}\}$;
    $\mathcal{L} = \mathcal{L} \cup \{l = (n, c) : n = src(l') \wedge host(c) = dst(l'), l' \in \mathcal{L}\}$;
    $\mathcal{I}(c) = \{e \in r : r = resource(c)\}$;
    $\mathcal{R}(c) = resource(c)$;
    /* find best feasible observation assignment */;
    $cost' = FeasibleObsAssignment(\mathcal{R})$;
    if $cost' < cost$ then
        $\mathcal{R}(host(c)) = \mathcal{R}(host(c)) \cup resource(c)$ ;
        $cost = cost'$;
    end
    /* remove virtual node and related links */;
    $\mathcal{N} = \mathcal{N} \backslash \{c\}$;
    $\mathcal{L} = \mathcal{L} \backslash \{l = (c, n) : n = dst(l') \wedge host(c) = src(l'), l' \in \mathcal{L}\}$;
    $\mathcal{L} = \mathcal{L} \backslash \{l = (n, c) : n = src(l') \wedge host(c) = dst(l'), l' \in \mathcal{L}\}$;
end
```

The $FeasibleObsAssignment(\mathcal{R})$ function finds the best feasible observation assignment that matches the interests of nodes (non virtual and virtual), having as goal the minimization of energy consumption (Property 1). That is, find $\mathcal{O}_j^*$ so that $f(\mathcal{R}_i, \mathcal{O}_j^*)$ is minimized considering a specific $\mathcal{R}_i$. This is solved by the following integer linear problem:

$$Min \sum_{l \in \mathcal{L}} \sum_{r \in \cup_{n \in \mathcal{N}} \mathcal{R}(n)} \alpha_{l,r} \times [\delta + data(r)] \quad (4)$$

$$\sum_{n_j \in \mathcal{N}} \sum_{\{r \in \mathcal{R}(n_j) : i \in r\}} \sum_{\{l \in \mathcal{L} : src(l) = n_j\}} \lambda_{l,r}^{n_i,i} = 1, \forall n_i \in \mathcal{N}, \\ , \forall i \in \mathcal{I}(n_i) \quad (5)$$

$$\sum_{n_j \in \mathcal{N}} \sum_{\{r \in \mathcal{R}(n_j) : i \in r\}} \sum_{\{l \in \mathcal{L} : src(l) = n_k\}} \lambda_{l,r}^{n_i,i} - \\ - \sum_{n_j \in \mathcal{N}} \sum_{\{r \in \mathcal{R}(n_j) : i \in r\}} \sum_{\{l \in \mathcal{L} : dst(l) = n_k\}} \lambda_{l,r}^{n_i,i} = 0, \\ , \forall n_i \in \mathcal{N}, \forall i \in \mathcal{I}(n_i), \\ , \forall n_k \notin \{n \in \mathcal{N} : i \in r, \exists r \in \mathcal{R}(n)\} \wedge n_k \neq n_i \quad (6)$$

$$\sum_{n_j \in \mathcal{N}} \sum_{\{r \in \mathcal{R}(n_j) : i \in r\}} \sum_{\{l \in \mathcal{L} : dst(l) = n_i\}} \lambda_{l,r}^{n_i,i} = 1, \forall n_i \in \mathcal{N}, \\ , \forall i \in \mathcal{I}(n_i) \quad (7)$$

$$\alpha_{l,r} \geq \frac{1}{|\mathcal{N}| \times | \cup_{n \in \mathcal{N}} \mathcal{I}(n)|} \sum_{n_i \in \mathcal{N}} \sum_{i \in \mathcal{I}(n_i)} \lambda_{l,r}^{n_i,i}, \forall l \in \mathcal{L}, \\ , \forall r \in \cup_{n \in \mathcal{N}} \mathcal{R}(n) \quad (8)$$

where $\lambda_{l,r}^{n_i,i}$ and $\alpha_{l,r}$ are binary variables. The first indicates if the interest $i \in \mathcal{I}(n_i)$ of consumer $n_i \in \mathcal{N}$ is met through resource $r \in \cup_{n \in \mathcal{N}}$, and such resource flows through link $l \in \mathcal{L}$, while the second indicates

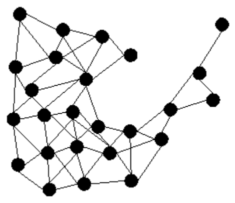

Fig. 1. Network topology under test.

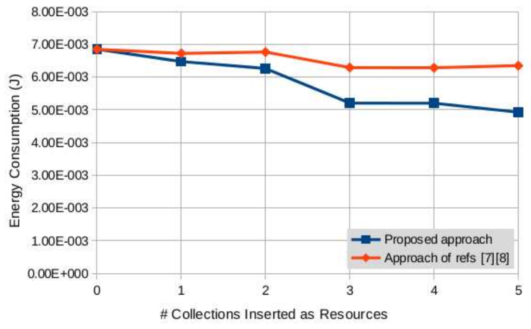

Fig. 2. Energy consumption.

if resource r flows through l due to at least an observation request. Note that this ensures flowing of resources into Collections and Consumers according to their interests. CPLEX[1] is used to solve this problem.

IV. Analysis of Results

The tested network is shown in Figure 1, has 25 nodes and was randomly generated using the algorithm in [9]. Each node $n \in \mathcal{N}$ produces a single unique Atomic resource, a quarter of the nodes are randomly selected to become consumers, the average number of subject interests per node is four and are randomly selected. It is assumed that nodes use 52.2mW when transmitting/receiving, values for Z1 motes taken from [10]. Thus, the per byte time energy consumption is $E^{TR} = 167.04 \times 10^{-8}$J. Regarding the δ and $data(r)$ these are 50% of the average notification/registration message size, which is assumed to be of 50 bytes.

Figure 2 shows the energy consumed with the registration phase and transmission of a single notification to all nodes with interests (consumers). Figure 3 shows the number of registration messages required for observation set up using. Plots include the [7], [8] approach for comparison[2]

It is possible to observe that the proposed approach is able to reduce energy consumption since less control/registration messages, when compared with [7], [8], are used. Thus, observation set up and any future changes, regarding Collection content and placement, can be done with a lower impact on energy consumption and, therefore, the network can be kept optimized if frequent updates are required.

[1]IBM ILOG CPLEX Optimizer.

[2]Both approaches perform aggregation so notification msgs are assumed to be similar, while registration msgs is what differentiates them. Thus, the registration steps returned by our algorithm are used to calculate the energy consumption of both. For [7], [8] there will be a msg per subject, while for the approach proposed here there will be a msg per Atomic/Collection resource.

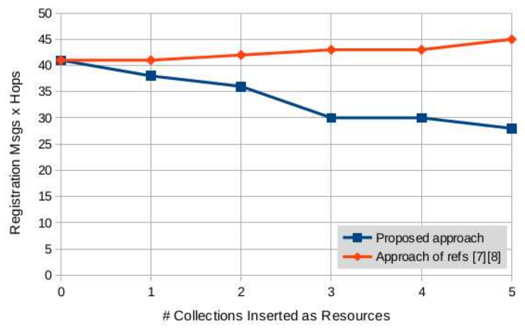

Fig. 3. Control/Registration messages for observation set up.

V. Conclusions

In this article CoRE Interfaces are applied to the energy saving problem in constrained networks. A resource design model is built and an algorithm is developed for resource design. This approach is effective in reducing energy consumption, as less control and notification messages will exist to meet client resource observation needs. The use of CoRE interfaces allows resources to be dynamically changed/created.

Acknowledgment

This work was supported by FCT (Foundation for Science and Technology) from Portugal within CEOT research center and UID/MULTI/00631/2013 project.

References

[1] Sye Loong Keoh, Sandeep S. Kumar, and Hannes Tschofenig: Securing the Internet of Things: A Standardization Perspective, IEEE Internet of Things Journal, May 2014.
[2] K. Hartke: ”Observing Resources in CoAP”, draft-ietf-core-observe-16, IETF, 2014.
[3] A. Ludovici, E. Garcia X. Gimeno and A. Calveras Auge: ”Adding QoS support for timeliness to the observe extension of CoAP”, IEEE Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), October 2012.
[4] Z. Shelby: ”Constrained RESTful Environments (CoRE) Link Format”, RFC 6690, 2012.
[5] Z. Shelby: ”CoRE Resource Directory”, draft-ietf-core-resource-directory-05, 2015.
[6] Z. Shelby, M. Vial and M. Koster: ”Reusable Interface Definitions for Constrained RESTful Environments”, draft-ietf-core-interfaces-04, 2015.
[7] N. Correia, D. Sacramento and G. Schütz: ”Dynamic Aggregation and Scheduling in CoAP/Observe Based Wireless Sensor Networks”, IEEE Internet of Things, Vol. PP, No. 99, 2016.
[8] D. Sacramento, G. Schütz and N. Correia: ”Aggregation and Scheduling in CoAP/Observe Based Wireless Sensor Networks”, IEEE International Conference on Communications (ICC), June 2015.
[9] Furuzan Atay Onat and Ivan Stojmenovic: ”Generating Random Graphs for Wireless Actuator Networks”, IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, June 2007.
[10] Girum Ketema Teklemariam, Jeroen Hoebeke, Ingrid Moerman and Piet Demeester: ”Facilitating the Creation of IoT Applications Through Conditional Observations in CoAP”, EURASIP Journal on Wireless Communications and Networking, 2013.