Resource Redesign in RELOAD/CoAP Overlays for the Federation of Sensor Networks

L. Rodrigues(✉), J. Guerreiro, and N. Correia

CEOT, FCT, University of Algarve, 8005-139 Faro, Portugal
{lrodrig,jdguerreiro,ncorreia}@ualg.pt

Abstract. With 5G technologies new platforms for global connectivity of sensing devices are expected, many of them relying on federated sensor networks. Such federation can be done through P2P overlays where proxy nodes, of constrained environments, announce device resources or data. However, P2P resources may end up including similar device resource entries, if these can be announced under different P2P resource umbrellas, posing consistency and efficiency problems. Changes in a device resource (or data) will require the update of multiple P2P resources. To avoid this, a two-layer overlay architecture is used in this article so that P2P resources can include references to P2P anonymous resources, specifically created to avoid duplicate entries. In this article, procedures to keep P2P resources (anonymous or not) updated over time are proposed. Results show that these are effective in avoiding duplicate entries.

Keywords: Federated networks · Wireless sensor networks RELOAD · CoAP · CoAP usage

1 Introduction

The Internet of Things (IoT) brings the opportunity for Things to connect globally and exchange data within the existing Internet infrastructure [1]. This means that sensing devices may interconnect for wide-area coverage, eventually cooperating to accomplish certain tasks, allowing for wide-area complex applications to be developed (e.g., environmental monitoring, earthquake/tsunami early-warning systems, correlation of health data, ...). Since 5G technologies meet the requirements of mobile communications and needs for Thing data transmission, these are expected to have a key role in such scenarios, enabling new platforms for global connectivity to emerge [2]. Such platforms may rely on the federation of multiple sensor networks.

In federation scenarios two standards end up having a significant role: (i) REsource LOcation And Discovery (RELOAD) base protocol, providing a generic self-organizing Peer-to-Peer (P2P) overlay network service supporting

V. Sucasas et al. (Eds.): BROADNETS 2018, LNICST 263, pp. 31–40, 2019.
https://doi.org/10.1007/978-3-030-05195-2_4

different applications through the use of *Usages* [3]; (*ii*) Constrained Application Protocol (CoAP), a Web application transfer protocol for RESTful services to be provided in constrained nodes and networks, similarly to Hypertext Transfer Protocol (HTTP) [4]. A Usage in RELOAD defines how data is mapped into something (data object) that can be stored in the P2P overlay, how resources stored at the overlay are identified, how to retrieve data, and how to secure data. Since CoAP is expected to be a common application layer transfer protocol at constrained devices, a CoAP Usage for RELOAD has been proposed in [5]. This allows proxy nodes, of constrained environments, to form a P2P overlay to announce available resources and sensor data, so that any client can discover them. This is illustrated in Fig. 1. P2P models have the advantage of being decentralized, scalable and resilient.

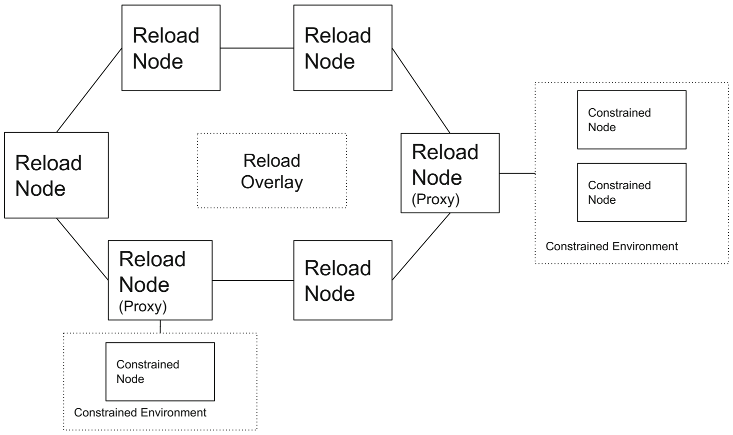

Fig. 1. RELOAD/CoAP overlay network.

A RELOAD/CoAP overlay may announce device resources or sensor data. When announcing device resources (e.g., temperature sensor), the P2P data object (called P2P resource) includes entries (linking information) for physical device resources to be reached using the CoAP protocol. When announcing sensor data, P2P resources include sensor information directly (value, timestamp, lifetime). Furthermore, device resources managed by different proxies/RELOAD nodes can be announced under a unique P2P resource umbrella, allowing for the aggregation of a set of device resources of the same type or with similar characteristics. For example, a P2P temperature resource may include temperature sensors from multiple places. This means, however, that P2P resources may end up including similar device resource entries (or similar sensor data in the caching case) because multiple combinations of device resources may give rise to independent P2P resources. This poses consistency/efficiency problems since multiple P2P resources must be updated when device resource entries change (for each device resource, suppliers also would have to keep track of which P2P resources to update). This becomes critical as more and more objects integrate in the IoT (possible combinations of resources will increase exponentially).

To overcome the previously mentioned consistency problem, a two-layer overlay architecture, relying on P2P anonymous resources, is used in this article. P2P anonymous resources can then serve as entries in other P2P resources, similarly to device resources. Procedures are proposed to keep the population of P2P anonymous resources, and any reference to them, updated over time as new P2P resources are announced or existing ones are updated/removed.

The remainder of this article is organized as follows. In Sect. 2, work related with RELOAD/CoAP architectures is discussed. Section 3 analyses the two-layer approach and required resource redesign procedures, and results are discussed in Sect. 5. Section 6 concludes the article.

2 Related Work

The RELOAD/CoAP architecture for wide area sensor and actuator networking, using the previously mentioned CoAP application usage for RELOAD, was initially proposed in [1]. The advantages of such architecture are discussed, which include integration with the Web, self-organization, scalability, robustness, and simulations are performed to compare its performance against a traditional client/server architecture. Federation of autonomous sensor networks, that have to collaborate for achieving a common task, is also discussed in [6]. However, such discussion does not focus on a distributed system, like the P2P-based architecture, and just 2-vertex distinct paths between every pair of WSNs is ensured, while minimizing the number of deployed relay nodes and having average node degree concerns.

Having the RELOAD/CoAP architecture proposed in [1] as a basis, P2P overlays for network coding based constrained environments are discussed in [7]. The goal is for encoding vectors and encoded data to be stored, and a decoding service to be used in case of packet loss. A work more closer to the one discussed in this article, also based on RELOAD/CoAP architecture proposed in [1], is the one in [8]. The focus is also on an effective announcement of P2P resources, for consistency purposes, but no P2P anonymous resources are created. That is, device resource entries can only be replaced by references to existing P2P resources but no extra P2P anonymous resources can be created for consistency purposes.

3 RELOAD/CoAP Overlay Networks

CoAP is a Web application transfer protocol designed for the special requirements of constrained environments [4]. CoAP provides a request/response interaction model between application endpoints and `coap(s)` URI scheme is used to identify CoAP resources. CoAP URI supports the path prefix `/.well-known` so that clients can discover resources available at the host, or discover any policy/information about the host, before making a request [9]. Since CoAP is expected to be a common application layer transfer protocol, a CoAP Usage for RELOAD was proposed in [5].

RELOAD/CoAP nodes, usually proxies, can announce/store NodeID to resource link mappings in the overlay. As an example, if a node participating in overlay `overlay1.com`, with NodeID `9996172`, wants to register a structure with its sensors, a mapping similar to the following would be used:

```
Resource-ID=h("coap://overlay-1.com/proxy-1/.well-known/")
KEY=9996172
VALUE=[
</sensors/temp-1>; rt="temperature-c"; if="sensor",
...
]
```

The hash over the URI, `h(...)`, is used for indexing. The `KEY` is the Node-ID of the RELOAD/CoAP node responsible for device resources at the `VALUE` section, which follow the CoRE Link Format specified in [10]. After P2P resource fetching, an AppAttach request can be sent to a specific Node-ID (direct connection for CoAP message exchange).

A P2P resource may also announce device resources managed by different nodes. The following example shows a temperature related P2P resource involving multiple RELOAD nodes:

```
Resource-ID = h("coap://overlay-1.com/temperature/.well-known/")

KEY =  9996172,
VALUE = [
   </sensors/temp-1>;rt="temperature-c";if="sensor",
   </sensors/temp-2>;rt="temperature-c";if="sensor"
]

KEY = 9996173,
VALUE = [
   </sensors/temp-a>;rt="temperature-c";if="sensor",
   </sensors/temp-b>;rt="temperature-c";if="sensor"
]
```

Therefore, different combinations of device resources can be made available under different P2P resource umbrellas, allowing for the aggregation of a set of device resources of the same type or with similar characteristics.

3.1 Motivation

The proliferation of similar entries at multiple P2P resources has disadvantages: (*i*) keeping all P2P resources up-to-date becomes difficult; (*ii*) tracking which P2P resources have which device resource entries becomes difficult (new P2P resources can emerge at any time, if devices are available to the public). To overcome this problem, a two-layer overlay approach is used. More specifically, extra P2P anonymous resources are created, and content of announced P2P resources is changed in order to point to such anonymous resources leading to device resource entries of interest. A P2P resource will, therefore, have one of the following content formats:

- *Anonymous* - Includes references to P2P anonymous resources, each following the CoRE Link Format.
- *KeyValue* - Includes `(KEY,VALUE)` entries following the CoAP Usage format.

As an example, assuming that `coap://overlay-1.com/ heat-related-illness` includes temperature device resource entries similar to `coap://overlay-1.com/temperature`, plus extra CO2 entries, its content would be:

```
Resource-ID = h(coap://overlay-1.com/heat-related-illness/.well-
                                                           known)
[
    <coap://overlay-1.com/10001/>;rt="temperature
                                 heat-related-illness";if="leaf",
    <coap://overlay-1.com/10002/>;rt="heat-related-illness";
                                 if="leaf"
}
```

and the content of `coap://overlay-1.com/temperature` would be:

```
Resource-ID = h(coap://overlay-1.com/temperature/.well-known)
[
    <coap://overlay-1.com/10001/>;rt="temperature
                                 heat-related-illness";if="leaf"
]
```

While these have an Anonymous content format, the P2P anonymous resources `coap://overlay-1.com/10001/` and `coap://overlay-1.com/10002/` would follow the CoAP Usage format, storing `(KEY,VALUE)` entries with temperature and CO2 device resource links, respectively. Note that the `if` specifies how to deal with the endpoint, and "leaf" means that its content follows the KeyValue format (otherwise the Anonymous format is assumed). Basically, a graph exists where P2P resources are nodes and references are links, and multiple hops may exist until `(KEY,VALUE)` entries are reached. The `rt` indicates resource types for such entry, which end up being the parent P2P resources at the graph. An overlay service would ensure that the client, performing a fetch to a P2P resource, receives just the `(KEY,VALUE)` entries and would not be aware of P2P anonymous resources.

The previously discussed two-layer overlay approach is able to avoid duplicate entries. However, existing P2P resources may have to be redesigned whenever new ones are created or existing ones are removed/modified. Procedures for this purpose are discusses next.

4 Two-Layer Overlay Approach

4.1 Assumptions and Operation

Let us assume a set of original RELOAD/CoAP P2P resources, $\mathcal{R}$, where each $r \in \mathcal{R}$ includes a set of `(KEY,VALUE)` entries denoted by $\mathcal{P}_r$, where each $p_r \in \mathcal{P}_r$ includes a set of device resource entries denoted by $\mathcal{E}_{p_r}$. Let us assume also that

P2P resources in $\mathcal{R}$ were redesigned according to the previously mentioned two-layer overlay approach, giving rise to $\mathcal{R}'$. That is, its content has changed so that references to P2P anonymous resources are used to avoid duplicates. The set of created P2P anonymous resources is denoted by $\mathcal{A}$. Such redesign should be done having the following assumptions as a basis.

Assumption 1 (P2P Resource Content). *The content of a P2P resource $r \in \mathcal{R}' \cup \mathcal{A}$ will be of type $T(r)$, where $T(r) \in \{``Anonymous", ``KeyValue"\}$. More specifically, if $r \in \mathcal{R}'$ then $T(r) = \{Anonymous\}$, meaning that an entry, denoted by $p_r \in \mathcal{P}_r$, will be a reference to a P2P anonymous resource following the CoRE Link Format. If $r \in \mathcal{A}$, then $T(r) = \{Anonymous\}$ if r is not a leaf and $T(r) = \{KeyValue\}$ if r is a leaf.*

This means that a resource in $\mathcal{R}'$ will always have links to P2P anonymous resources, while P2P anonymous resources in $\mathcal{A}$ will have either links to leaf P2P resources with (KEY, VALUE) entries or links to other P2P anonymous resources.

Assumption 2 (Number of Entries). *A device resource entry can not be included in more than one P2P resource. That is, for any $e \in \mathcal{E}_{p_r}$, $p_r \in \mathcal{P}_r$ and $r \in \mathcal{R}$, the following must hold: $|\{a \in \mathcal{A} : e_{p_a} = e, e_{p_a} \in \mathcal{E}_{p_a}, p_a \in \mathcal{P}_a\}| = 1$. A P2P anonymous resource entry can be included in multiple P2P resources.*

Assumption 3 (Strict Coverage). *P2P resources must be redesigned while not changing the content to be returned. That is, $\bigcup\limits_{p_r \in \mathcal{P}_r} \mathcal{E}_{p_r} = \bigcup\limits_{p_{r'} \in \mathcal{P}_{r'}} \mathcal{E}_{p_{r'}}, \forall r \in \mathcal{R}$, $\forall r' \in \mathcal{R}'$.*

Upon a P2P resource fetch, an overlay service[1] would have to fetch the referenced P2P anonymous resources recursively, so that device resource entries are reached and returned to the client following the CoAP Usage format. This information is the one used by clients to perform AppAttachs.

To keep assumptions valid, P2P resources (anonymous included) will have to be updated when P2P resources are created/modified/removed. A set of merge/split procedures, for this purpose, are discussed next. These procedures assume the existence of a P2P resource per proxy (e.g., `coap://overlay-1.com/KEY=9996172`) containing references to the P2P anonymous resources including proxy's `(KEY,VALUE)` entries. These are referred to as proxy P2P resources.

4.2 Resource Redesign Procedures

P2P Resource Creation: When a new P2P resource r is to be created, initially with (KEY, VALUE) entries in its content, the procedure Create shown below must be performed. At line 3, the P2P anonymous resources, whose entries are

[1] A service discovery mechanism like ReDiR can be used to distribute load among RELOAD/CoAP nodes able to provide such service, ensuring scalability.

fully included in r, are obtained. A call to these should replace corresponding entries in r. Line 7 fetches P2P anonymous resources sharing content with r. These should be analyzed by Split procedure. At this procedure, the splitting is performed at lines 2–4. This is recursive so that parent nodes are analyzed for splitting too (call at line 10).

```
CREATE($r$)
1: for $p_r \in \mathcal{P}_r$ do
2:     Extract $p_r$'s KEY and fetch corresponding proxy P2P resources, $r'$
3:     $\mathcal{I} = \{p_{r'} \in \mathcal{P}_{r'} : p_{r'} \subseteq p_r\}$
4:     for $p_i \in \mathcal{I}$ do
5:         Replace $p_i \cap p_r$ content in $r$ by reference $p_i$
6:     end for
7:     $\mathcal{I}' = \{p_{r'} \in \mathcal{P}_{r'} \backslash \mathcal{I} : p_{r'} \cap p_r \neq \emptyset\}$
8:     SPLIT($\mathcal{I}', p_r$)
9:     for $p_i \in \mathcal{I}'$ do
10:        Replace $p_i \cap p_r$ content in $r$ by reference $p_i$
11:    end for
12:    Create P2P anonymous resource if intact $p_r$ content exists, and replace it by
       reference.
13: end for
```

```
SPLIT($\mathcal{I}', p_r$)
1: for $p_i \in \mathcal{I}'$ do
2:     Create resource with content $p_i \cap p_r$
3:     Create resource with content $p_r \backslash \{p_i \cap p_r\}$
4:     Update references at resources in $rt(p_i)$
5: end for
6: if no changes exist then
7:     Return
8: end if
9: for $p_i \in \mathcal{I}'$ do
10:    SPLIT($rt(p_i), p_r$)
11:    Delete $p_i$
12: end for
```

P2P Resource Removal: When a P2P resource $r \in \mathcal{R}'$ is to be removed, the procedure Remove shown below must be performed. Lines 2–5 will remove references to the resource being deleted. The Merge procedure analyses common content, for possible joins. At this procedure, the $\mathbb{P}()$ is the powerset. Merges will be performed at children nodes recursively. It is assumed that the corresponding rt information is updated accordingly.

```
REMOVE($r$)
 1: for $p_r \in \mathcal{P}_r$ do
 2:     for $i \in rt(p_r)$ do
 3:         Fetch $i$
 4:         Remove $r$ from $rt(\{p_i \in \mathcal{P}_i : p_i = p_r\})$
 5:     end for
 6:     MERGE($rt(p_r)$)
 7: end for
 8: Delete $r$
```

```
MERGE($resources$)
 1: $\mathcal{U} = \bigcup_{i \in resources} \mathcal{P}_i$
 2: $\mathcal{I} = \arg\max_{\mathcal{S} \in \mathbb{P}(\mathcal{U})}(|\{i \in \mathcal{S} : rt(i) = \bigcap_{j \in \mathcal{S}} rt(j)\}|)$
 3: if $|\mathcal{I}| = 1$ then
 4:     Replace content $\mathcal{I}$, at every $r' \in resources$, by $i \in \mathcal{I}$
 5:     Delete $i \in \mathcal{I}$
 6:     MERGE($resources$)
 7: else
 8:     if $|\mathcal{I}| > 1$ then
 9:         Create new resource $r$ with content of every $i \in \mathcal{I}$
10:         Replace content $\mathcal{I}$, at every $r' \in resources$, by $r$
11:         Delete every $i \in \mathcal{I}$
12:         if $r$ is not of leaf then
13:             MERGE($r$)
14:         end if
15:     end if
16: end if
```

P2P Resource Update: It is assumed that a P2P resource removal is performed, followed by a P2P resource creation.

5 Scenario Analysis

To evaluate the advantages of having P2P anonymous resources, different scenarios were tested. In such scenarios there are public P2P resources (URIs known to the public), each having a set of P2P anonymous resource entries (i.e., of Anonymous content type). The P2P anonymous resources being referenced can then be of Anonymous or KeyValue content type. In the first case another calling level is being built, which may reference any existing P2P anonymous resources. A maximum of two levels exists. Table 1 shows the ranges used to define the number of P2P resources or resource/sensor entries (a random number is picked to run a test). The following plots show the average of 20 tests performed for each scenario.

Table 1. Scenarios under test.

Scenario	Public P2P resources	Anonymous resources per level	Refs for anonymous content	Refs for KeyValue content
A	[1–10]	[5–10]	[1–5]	[1–5]
B	[1–20]	[10–20]	[1–10]	[1–10]
C	[1–30]	[15–30]	[1–15]	[1–15]
D	[1–40]	[20–40]	[1–20]	[1–20]
E	[1–50]	[25–50]	[1–25]	[1–25]
F	[1–60]	[30–60]	[1–30]	[1–30]

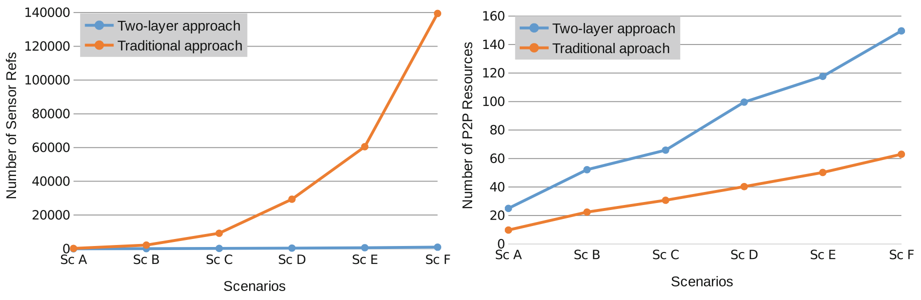

Fig. 2. Number of sensor references and P2P resources.

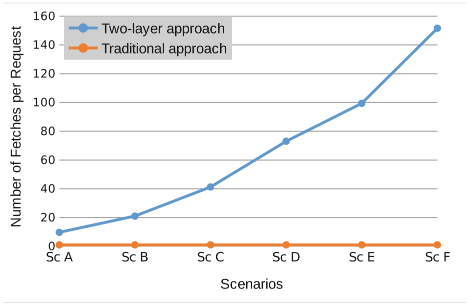

Fig. 3. Number of fetches per public P2P resource.

As shown in Fig. 2, if P2P anonymous resources are not created then the total number of sensor entries will grow exponentially due to duplicate entries. The proposed solution is able to keep a single reference to sensor entries, resulting into very low values for the total number of entries. Duplicate entries are avoided, however, at the expense of additional fetches. Figure 3 shows the number of required fetches per public P2P resource. Note, however, that fetching in parallel will reduce latency (i.e., for two levels the overall delay converges to the time of two fetches in series).

6 Conclusions and Further Work

Resource redesign procedures were proposed to keep P2P resources in RELOAD/CoAP architectures updated over time, while ensuring that sensor resource entries are unique. This ensures data consistency, better coordination of cooperating systems, and timeliness of notifications. Future work includes comparing this approach with other optimization approaches.

Acknowledgment. Work supported by FCT (Foundation for Science and Technology) from Portugal within CEOT (Center for Electronic, Optoelectronic and Telecommunications) and UID/MULTI/00631/2013 project.

References

1. Mäenpää, J., Bolonio, J.J., Loreto, S.: Using RELOAD and CoAP for wide area sensor and actuator networking. EURASIP J. Wireless Commun. Networking **2012**, 121 (2012)
2. Zhang, D., et al.: One integrated energy efficiency proposal for 5G IoT communications. IEEE Internet Things J. **3**(6), 1346–1354 (2016)
3. Jennings, C., et al.: REsource LOcation And Discovery (RELOAD) Base Protocol. RFC 6940 (2014)
4. Shelby, Z., et al.: The Constrained Application Protocol (CoAP). RFC 7252 (2014)
5. Jimenez, J., et al.: A Constrained Application Protocol (CoAP) Usage for REsource LOcation And Discovery (RELOAD). RFC 7650 (2015)
6. Lee, S., Younis, M., Lee, M.: Optimized bi-connected federation of multiple sensor network segments. Ad Hoc Networks **38**, 1–18 (2016)
7. Al-Hawri, E., Correia, N., Barradas, A.: RELOAD/CoAP P2P overlays for network coding based constrained environments. In: Camarinha-Matos, L.M., Parreira-Rocha, M., Ramezani, J. (eds.) DoCEIS 2017. IAICT, vol. 499, pp. 307–315. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56077-9_30
8. Rodrigues, L., Guerreiro, J., Correia, N.: RELOAD/CoAP architecture with resource aggregation/disaggregation service. IEEE PIMRC, IoT Workshop, Valencia 4–7 September 2016 Spain (2016)
9. Nottingham, M., Hammer-Lahav, E.: Defining Well-Known Uniform Resource Identifiers (URIs). RFC 5785 (2010)
10. Shelby, Z.: Constrained RESTful Environments (CoRE) Link Format. RFC 6690 (2012)