RELOAD/CoAP Architecture with Resource Aggregation/Disaggregation Service

L. Rodrigues, J. Guerreiro and N. Correia
CEOT, FCT, University of Algarve
8005-139 Faro, Portugal
Emails: {lrodrig,jdguerreiro,ncorreia}@ualg.pt

***Abstract*—M2M communication is expected to occur at a global level and for this reason federations of device networks are also expected. In such large scale environments, a critical issue is how to discover the available resources in a scalable manner. For this purpose CoAP Usage for RELOAD, a generic self-organizing P2P overlay network service, has been proposed to be used as a lookup service, to store available resources and as a cache for sensor data. However, such approach alone does not allow building an aggregate resource hierarchy, a very relevant issue for an efficient organization of data in future IoT applications. Here we address this issue and propose an architecture incorporating a resource aggregation/disaggregation service.**

***Keywords*—RELOAD, CoAP, CoAP Usage, M2M, Aggregation/Disaggregation.**

I. Introduction

The idea behind the Internet of Things (IoT) is for everyday objects to be connected, IP-addressable and integrated into the Internet [1]. The IoT has been boosted by the availability of more affordable wireless modules, making this technology very attractive for many applications, and in future scenarios such small sensing devices are expected to interconnect over large geographical areas for Machine-to-Machine (M2M) communication. For such devices to interact effectively new solutions are needed, which may involve the federation of device/sensor networks.

In the context of wide-area sensor networks, and federated networks, some open Internet standards become important. For IP-based communications, the *Internet Engineering Task Force* (IETF) defined the *IPv6 over Low-Power Wireless Area Networks* (6LoWPAN) standard that enabled IPv6 over very constrained networks [2]. The IP protocol emerges, therefore, as the glue to interconnect heterogeneous devices. Also withing IETF, the *Constrained RESTful Environments* (CoRE) working group has focused on the development of *Constraint Application Protocol* (CoAP), a Web application transfer protocol intended to provide RESTful services in constrained nodes and networks [3]. More recently, a CoAP Usage for *REsource LOcation And Discovery* (RELOAD) base protocol has been proposed [4], [5]. RELOAD provides a generic self-organizing *Peer-to-Peer* (P2P) overlay network service, and CoAP Usage for RELOAD allows CoAP nodes to store resources in a RELOAD P2P overlay network. More specifically, the proxy nodes of federated constrained sensor networks, with enough capacity to run RELOAD, could form a P2P overlay/virtual network themselves to announce resources and for clients to be able to discover the available resources. More specifically, the overlay network would be used:

- as a lookup service
- to store resources (e.g. sensor, controller)
- as a cache for sensor data

Although constrained devices will be heterogeneous regarding their radio layer (e.g. long range modules: 2G, 3G, 4G; short range modules: xbee, zigbee), CoAP is expected to be a common application layer protocol. For this reason a CoAP Usage for RELOAD has been proposed, allowing P2P overlay networks to be built and sensor networks to be federated. Although being a standard-based scalable architecture, a desirable feature when millions of objects of all kinds are expected to be integrated into the IoT, we believe that such approach alone can not completely address another quite important issue. An efficient lookup of resources. More specifically, it does not allow resource aggregates to be hierarchically organized, a relevant issue for IoT resources to be well organized, as explained in more detail in Section IV. Here we propose a RELOAD/CoAP architecture with a resource aggregation/disaggregation service to overcome this limitation.

The remainder of this article is organized as follows. In Section II the CoRE and CoAP related standards are introduced, while the RELOAD framework and the CoAP Usage are discussed in Section III. The architecture being proposed is presented in Section IV, and its advantages are discussed, and Section V makes a scenario analysis for a better understanding of the proposal. Finally, in Section VI some conclusions and future work are drawn.

II. CoAP and CoRE

The CoRE realizes the REST architecture for more constrained nodes. In such environments, the possibility of discovering resources, hosted by constrained nodes, is important for applications to run without human intervention and for flexible interfaces to be

 DOI: 10.1109/PIMRC.2016.7794607

provided. Web discovery and linking was initially specified for *Hypertext Transfer Protocol* (HTTP) in [6], [7]. In the context of constrained nodes, the discovery of resources, their attributes and relations is referred to as CoRE Resource Discovery [8].

A. CoAP

While CoRE aims at realizing the REST architecture in a suitable form for constrained nodes and networks, CoAP emerges as the Web application transfer protocol that has been designed for the special requirements of these constrained environments, especially considering energy, building automation and M2M applications [3]. CoAP provides a request/response interaction model between application endpoints. The "*coap*" and "*coaps*" URI schemes are used to identify CoAP resources and the path prefix "*/.well-known*" is supported for clients to be able to discover resources available at the host, or discover any policy/information about the host, before requesting for resource values/notifications [7].

For clients to continuously observe resources, and keep representations updated over time, the Observe extension to CoAP has been proposed in [9]. After discovering a resource the observer/client can obtain the resource values by sending an extended GET request either to the server, having such resource in its namespace, or to a proxy to be used as an intermediate [10], [11]. The server, or proxy, will register the client as an observer, so that the client starts to receive notifications, and responds with an extended response. Extended requests/responses are CoAP requests/responses with an Observe option. Notifications can be kept in cache until they do not expire, which is controlled by the max-age CoAP option.

B. The CoRE Link Format

When a discovery is done, the URIs for the resources hosted by the server, attributes of resources and link relations are provided. In CoRE such a link collection is also a resource[1]. The format of the links is specified in [8] and is called CoRE Link Format. An Internet media type is assigned for CoRE Link Format payloads ("*application/link-format*").

The CoRE Link Format extends the HTTP Link Header field specified in [6], but it is a serialization of typed links and it is fairly compact (no special XML parsing is required, for example). Therefore, conversion between HTTP Link Header field and this link format is easily done. A resource will have:

- Multiple link descriptions separated by comma;
- Each link description includes "<" + URI-Reference + ">" plus a set of parameters ("*rt*", "*if*", ...), all separated by semicolons.

An example of a resource hosted by a CoAP server is:

```
</sensors/temp-1>;rt="temperature-c";if="sensor",
</sensors/temp-2>;rt="temperature-c";if="sensor"
```

An application-specific semantic type can be assigned to a resource using the "*rt*" parameter, either by indicating values/names (e.g. "*temperature*") directly, separated by a space, or by indicating an URI referencing a specific concept in an ontology (e.g. "*http://sweet.jpl.nasa.gov/2.0/phys.owl#temperature*").

The "*if*" parameter is used to specify an interface definition used to interact with the resource. This may include values/names (e.g. "*sensor*") directly, separated by a space, or an URI defining the interface (e.g. "*http://www.example.org/myapp.wadl#sensor*"). This way every resource, method, request and response is formally and precisely described.

C. CoRE Resource Directory

As previously said, CoAP provides a request/response interaction model between endpoints, supporting discovery of resources, and the use of Web linking for description and discovery of resources, hosted by constrained servers, is specified by the CoRE Link Format in [8]. However, direct discovery of resources may not be practical because nodes may go to sleep mode, and for this reason the use of a *Resource Directory* (RD) entity can be used, which has been specified in [12]. Such Web entity would host descriptions of resources held on other servers, allowing lookups from others. For more details on RD see [12].

More recently, an alternative to this centralized resource directory approach has been proposed that is based on RELOAD. Such approach, discussed in the following section, can be seen as a distributed RD.

III. RELOAD

RELOAD is a generic P2P framework for the management of self-organizing P2P overlay networks and pluggable application layers (Usages) can be incorporated in it [4]. Nodes can route messages to other nodes and store/retrieve data in the overlay when using RELOAD. Important features of RELOAD include security, Usage model, NAT traversal, optimized routing and overlay algorithm extension capability.

RELOAD can support several applications by the use of Usages, which specify application related data types, and rules for how to use services provided by RELOAD. That is, Usages describe specific data Kinds and behaviour related to the Usage (e.g. access policies). Recently, a SIP Usage for RELOAD has been defined in [13]. In [5], a CoAP Usage has also been proposed that defines a pluggable application layer for constrained networks. This CoAP Usage allows a P2P overlay network to be built, where devices would store their available resources, allowing federation of sensor networks.

[1]Either a single or multiple links can be seen as a resource.

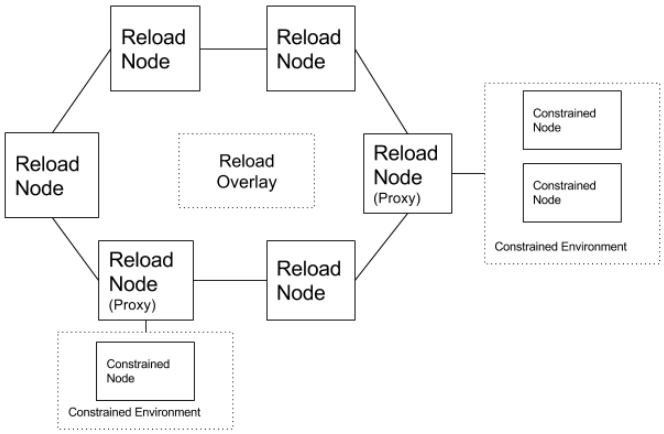

Fig. 1. Overlay topology when using CoAP Usage for RELOAD.

A. CoAP Usage

The CoAP Usage allows CoAP-based devices to store resources in a RELOAD P2P overlay, provides a lookup service and allows the use of RELOAD overlay as a cache for sensor data. This implementation, illustrated in Figure 1, avoids the use of centralized servers.

The CoAP nodes capable of using the RELOAD data storage functionality, usually proxies, will store/register mappings from their CoAP URI to their NodeID in the overlay. These CoAP nodes can be clients or peers in the overlay (see [4] for difference between client and peers) but both have NodeIDs. If a CoAP proxy node in overlay "*overlay1.com*", and using NodeID "*9996172*", wants to register sensors to URI "*coap://overlay-1.com/proxy-1/.well-known/*", the following mapping 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",
...
]
```

where "*h(...)*" is the hash over the URI, because the key spaces for storage in the overlay must be numeric. Any node performing a lookup for URI "*coap://overlay-1.com/proxy-1/.well-known/*" receives the information that the RELOAD node (proxy) responsible for the resource has NodeID "*9996172*", together with the sensors and paths. Then, a direct connection (AppAttach) to the NodeID obtained can be performed for CoAP instruction exchange. That is, after AppAttach negotiation the requesting node can access the values of the constrained nodes at the RELOAD node to which it is connected (e.g. GET /sensors/temp-1). At CoAP Usage, the CoAP-Registration data Kind definition states that the KindID is the CoAP URI and that the data model used is the dictionary (dictionary key is the NodeID).

B. ReDir-based Service Discovery

In P2P overlay networks, like RELOAD overlay instances, peers share their resources to provide the service to which they were designed for [14]. For this system to work some peers may also provide services to other peers, and peers may request such available services (e.g. TURN relay service to assist in the traversal of NATs or firewalls) [4], [15], [16]. Therefore, peers face the problem of finding the set of peers providing that service. Although RELOAD specifies particular discovery mechanisms, for TURN for example, a generic service discovery mechanism is not part of the base protocol. For this reason the ReDiR service discovery mechanism has been applied to RELOAD overlays in [14] in order to provide a generic service discovery mechanism.

A naive storing solution would be store Node-IDs of nodes providing a certain service, under some key k. This, however, will overload the node responsible for the service identified by key k. Such node not only might end up storing a large number of Node-IDs but also must answer all service lookup requests for that service. The ReDiR-based service discovery mechanism proposed for RELOAD avoids this ensuring that the load related with a certain service is distributed among the nodes providing the service [14].

IV. Proposed Architecture for Federation of Sensor Networks

A. Motivation

Millions of objects of all kinds are expected to be part of the IoT. In order to support such growth, open and standard-based network architecture solutions are required for interoperability between the various ecosystems [1]. Another quite important issue is the lookup of resources. In order to be efficient, such lookup should allow resources to be part of an aggregate resource hierarchy, a relevant issue for future IoT data to be well organized. The architecture proposed next addresses this issue by incorporating an extra resource aggregation/disaggregation service into a RELOAD/CoAP base architecture.

The RELOAD, together with the CoAP Usage, provides a rendezvous system for the lookup of resources and CoAP nodes storing these resources. This allows one to find which resources are served by a RELOAD node, by making a fetching using the hash of the corresponding URI. Besides this, resources from multiple RELOAD nodes can be fetched if these nodes perform a store to the same resource [5]. This is possible because the data model being used is dictionary. In the following example the resources from two RELOAD nodes, with KEY=9996172 and KEY=9996173, are available under the resource "*temperature*":

```
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"
]
```

Consider now a new resource wrapping up "*temperature*" together with another resource. For example, the new resource "*heat-related illness*" including "*temperature*" plus the following "*co2*" resource entry:

```
Resource-ID = h(coap://overlay-1.com/co2/.well-
                                        known/)

KEY =  9996172,
VALUE = [
   </sensors/co2-1>;rt="co2";if="sensor"
]
```

To make such new resource available, one possibility is to join sensor entries under the "*heat-related illness*" resource as follows:

```
Resource-ID = h(coap://overlay-1.com/heat-related-
                              illness/.well-known/)

KEY =  9996172,
VALUE = [
   </sensors/temp-1>;rt="temperature-c";if="sensor",
   </sensors/temp-2>;rt="temperature-c";if="sensor",
   </sensors/co2-1>;rt="co2";if="sensor"
]

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

This approach, however, is not efficient since information is being duplicated. More specifically, "*h(coap://overlay-1.com/temperature/.well-known/)*" and "*h(coap://overlay-1.com/heat-related-illness/.well-known/)*" have sensor entries in common, hindering future information management like updates and removals. Thus, updating temperature sensor entries, for example, requires changing multiple resources. This is particular relevant if many aggregates are built on top of "*temperature*". Besides this disadvantage, resources can become quite populated with sensor entries making it difficult to visualize the types of sensors included in a resource. Note that the data model is a dictionary and only a single entry per peer is allowed, meaning that we could not have the "*temperature*" and "*co2*" entries separated in the "heat-related-illness" resource, as they would require the same key.

B. Architecture and Aggregate Building Model

Due to the just mentioned limitations, we believe that resource announcement should include the possibility of referring to aggregates. That is, resources of the same type, or with similar characteristics, should be announced under a single resource and then new resources would be built using these aggregates. Note that this way of organizing data is expected to be very relevant in future IoT applications. Considering the previous example, the "*heat-related illness*" resource should be able to include "*temperature*" and "*co2*" resources. However, the "*temperature*" resource does not have a single peer NodeID associated with it to be used as key in the dictionary. To overcome this we propose a RELOAD/CoAP based architecture with an extra resource aggregation/disaggregation (A&D) service. In this architecture:

- Storage/search of CoAP resources is to be done through A&D servers, which provide such service to the RELOAD overlay network.
- A discovery mechanism, like ReDiR, can be used to find A&D servers, so that overload is distributed among servers.
- Aggregates are registered in the RELOAD overlay network having as key the NodeID of a resource A&D server. Non aggregate resources are registered in the RELOAD overlay network having as key the NodeID of the peer RELOAD node responsible for the resource.

For the previous example the "*heat-related illness*" resource would have the following format:

```
Resource-ID = h(coap://overlay-1.com/heat-related-
                                illness/.well-known/)

KEY =  1116140,
VALUE = [
   </overlay-1.com/temperature>;if="aggregate",
   </overlay-1.com/co2>;if="aggregate"
]
```

where RELOAD node 1116140 relates to an A&D server. In the following section this "*heat-related illness*" resource scenario is used to analyse this architecture and time diagram of its operation.

1) Aggregate Building Model: In order to determine the best way to build resources, an aggregate building model is defined. Let us assume a set of resources $\mathcal{R}$ that need to be available in the overlay network. A resource $r \in \mathcal{R}$ includes a set of (KEY,VALUE) entries, denoted by $\mathcal{V}_r$ and each $v_r \in \mathcal{V}_r$ includes a set of sensor entries denoted by $\mathcal{E}_{v_r}$. The job of A&D servers, regarding storage operations, should be to replace the (KEY,VALUE) entries $\mathcal{V}_r$, $\forall r \in \mathcal{R}$, by a new set of (KEY,VALUE) entries $\mathcal{V}'_r$ where sensor entries $\mathcal{E}_{v'_r}$ are resources that are available. That is, $\mathcal{E}_{v'_r} \subset \mathcal{R}$. The resources after replacements should be

$$\mathcal{R}'(\mathcal{R}) = \{(\mathcal{V}'_r, \mathcal{E}_{v'_r}) | \cup_{v'_r \in \mathcal{V}'_r} \mathcal{E}_{v'_r} = \cup_{v_r \in \mathcal{V}_r} \mathcal{E}_{v_r} \wedge \wedge \sum_{v'_r \in \mathcal{V}'_r} |\mathcal{E}_{v'_r}| = min_{v''_r \in \mathcal{V}''_r}\{|\mathcal{E}_{v''_r}|\} \quad (1)$$

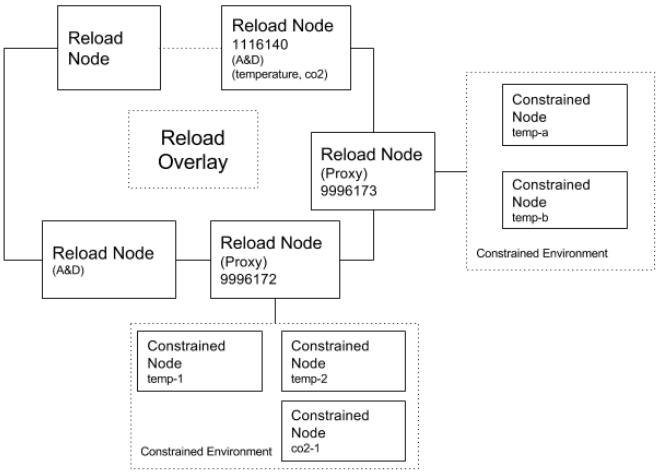

Fig. 2. RELOAD/CoAP overlay topology with A&D service.

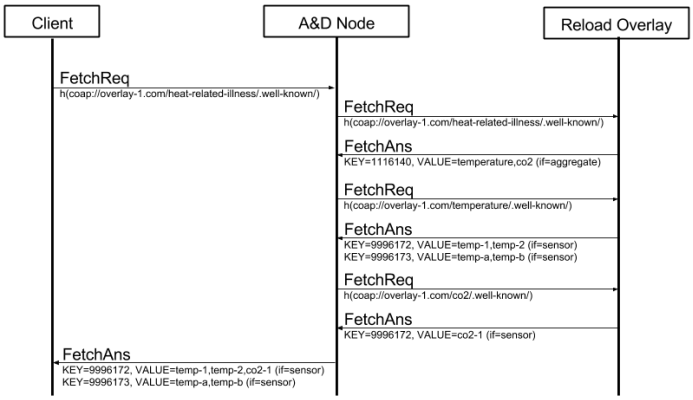

Fig. 3. Time diagram for scenario under analysis.

This means that the replacement should be made such that fetches to resources return the same sensor entries and the smallest number of entries exists.

V. Scenario Analysis

The architecture being proposed is shown in Figure 2 considering the "*heat-related illness*" resource scenario previously mentioned. In this Figure, besides the constrained environments we can see two A&D RELOAD nodes to which storage/fetches are sent. The RELOAD node 1116140 is responsible for the "*heat-related illness*" resource.

In this architecture the resource A&D servers become intermediates for the storage of resources and make the required disaggregation upon fetch operations by clients. More specifically, and as illustrated in the time diagram in Figure 3, a client instead of directly fetching the overlay for "*h(coap://overlay-1.com/heat-related-illness/.well-known/)*", it makes a fetching request directed to an A&D server (obtained using ReDir) asking for "*h(coap://overlay-1.com/heat-related-illness/.well-known/)*" to be solved. Such server fetches the overlay for that resource, obtaining an entry saying that RELOAD Node 1116140 is responsible for "*/overlay-1.com/temperature*" and "*/overlay-1.com/co2*" included in such resource. The server sends fetch requests to RELOAD Node 1116140, to get resources "*/overlay-1.com/temperature*" and "*/overlay-1.com/co2*", and after obtaining the answers the server sends the final disaggregated content (containing sensor entries) to the client. The interface "*aggregate*" indicates that the resource is an aggregate, meaning that the aggregate server with "*NodeID=1116140*" will be responsible for further steps towards disaggregation. Note that any updates to sensor entries would result into the update of final elements of aggregates (sensor node entries) avoiding having to search for all entries including such sensor node entries.

Note that fetch operations must be done through the A&D servers because special interfaces for correct disaggregation are needed. For example, all collected entries must be incorporated into an answer with sets of sensor entries associated with each node/proxy, using a dictionary as defined by CoAP Usage in order not to violate the standards.

As a final remark it is possible to state that this solution is fully scalable since the ReDir service discovery mechanism is assumed for the discovery of peers providing the A&D service. ReDir works on top of RELOAD overlays, and peers can become A&D servers at any time, meaning that there are no scaling problems.

VI. Conclusions and Future Work

In this article a RELOAD/CoAP based architecture including a resource aggregation/disaggregation service is proposed that allows efficient organization of M2M sensor data in P2P overlay networks since resource aggregates can be hierarchically organized. A scenario is analysed and aggregation/disaggregation operations are discussed together with the aggregate building model. From the analysis done we believe that this approach can successfully help on the efficient storage of available resource in federated device networks. We are currently working on effective heuristic algorithms for the aggregate building model that attempt to minimize the number of required fetches.

Acknowledgment

This work was 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] Jouni Mäenpää, Jaime Jiménez Bolonio and Salvatore Loreto: "Using RELOAD and CoAP for wide area sensor and actuator networking", EURASIP Journal on Wireless Communications and Networking, 2012:121, 2012.

[2] N. Kushalnagar, G. Montenegro, and C. Schumacher: IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals, RFC 4919, August 2007.

[3] Z. Shelby etal: "The Constrained Application Protocol (CoAP)", RFC 7252, 2014.

[4] C. Jennings et al: "REsource LOcation And Discovery (RELOAD) Base Protocol", RFC 6940, 2014.

[5] J. Jimenez et al: "A Constrained Application Protocol (CoAP) Usage for REsource LOcation And Discovery (RELOAD)", RFC 7650, 2015.

[6] M. Nottingham: "Web Linking", RFC 5988, 2010.

[7] M. Nottingham and E. Hammer-Lahav: "Defining Well-Known Uniform Resource Identifiers (URIs)", RFC 5785, 2010.

[8] Z. Shelby: "Constrained RESTful Environments (CoRE) Link Format", RFC 6690, 2012.

[9] K. Hartke: Observing Resources in CoAP, draft-ietf-core-observe-16, IETF, 2014.

[10] 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.

[11] 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.

[12] Z. Shelby: "CoRE Resource Directory", draft-ietf-core-resource-directory-05, 2015.

[13] C. Jennings et al: "A SIP Usage for RELOAD", draft-ietf-p2psip-sip-15, 2015.

[14] J. Maenpaa and G. Camarillo: "Service Discovery Usage for REsource LOcation And Discovery (RELOAD)", RFC 7374, 2014.

[15] J. Rosenberg: "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, 2010.

[16] R. Mahy, P. Matthews and J. Rosenberg: "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 5766, 2010.