Resource Discovery in the Internet of Things

By Akbar Rahman and Chonggang Wang, InterDigital Communications, Inc.

The *World Wide Web* (WWW or Web) is a global collection of connected documents and other resources that reside on the Internet. The introduction of the *Internet of Things* (IoT) is expected to dramatically increase the size of the Web in the near future and thus necessitates a fundamental change to the existing mechanisms of discovering resources. In IoT, the vision is that a significant number of new types of devices (or "things") such as fridges, car sensors, traffic lights, and so on, will be dynamically connected to the Web for communication and control. These IoT devices will have radically different characteristics from existing Web servers and users. This article looks at a key protocol development occurring at the *Internet Engineering Task Force* (IETF) for allowing IoT devices to discover resources via a new logical node called a *Resource Directory* (RD).

Resource Discovery in the Traditional Web

The basic unit of addressing on the Web is the *Uniform Resource Identifier* (URI) which identifies a resource [1]. The resource may, for example, be a restaurant review website page for a human user to read. Or in a more abstract form, the resource may be a software process to be triggered by a *Business-to-Business* (B2B) Web application as part of an automated stock market trading system. The key challenge in all cases is how the user can quickly find the correct URI for the resource that they are interested in out of all possible URIs in the entire Web space. This process is referred to as resource discovery.

The most well-known and powerful resource discovery mechanism in the current Web is the one employed by Web search engines such as *Baidu, Bing, Google, Yahoo,* etc. Specifically, search engines utilize the mechanism of Web crawlers (also called spiders, ants or robots) to periodically browse the Web to create a dynamic index of the resources of most publicly available websites. A website being defined as a server that hosts resources which can be accessed using *Hypertext Transfer Protocol* (HTTP) [2]. Human users can then send a search request, via a Web browser client, to lookup the specific resources that they are interested in.

Figure 1 shows the overall resource discovery process based on Web crawlers. Figure 1 is given in the context of a search engine, but very similar processes are followed by academic researchers, market research companies and others. However, unlike a search engine, these other entities typically do not send crawlers to cover the entire Web to discover all possible resources. Instead, they send crawlers to cover parts of the Web to discover the specific type of resource that they are interested in. For example, a market research company may send their Web crawlers to discover all the resources related to a specific type of product in a given geographic area as part of a pricing comparison study.

In Figure 1, Web crawlers start crawling out from the search engine server to an initially provisioned seed list of URIs. This seed list typically consists of very popular websites with a lot of URIs to other sites (that is, *hyperlinks*). From these initial websites, the Web crawlers start crawling outward to all connected hyperlinks. At each new website that it discovers, the Web crawler creates a copy of the website which it sends back to the search engine [3]. The search engine records all the received information in a bulk database and later processes it to create an optimized index for fast lookups. Then when a given search request comes from a Web browser client looking for some specific resource, the search engine will go quickly through its index using its own proprietary algorithm to find one or more matches. Finally, the search engine will return to the client a list of URIs and selected application content pertaining to the

resources that match the client's search parameters. This information will be displayed on the user's Web browser interface. The human user will then select ("click on") the URI(s) that she wants to visit.

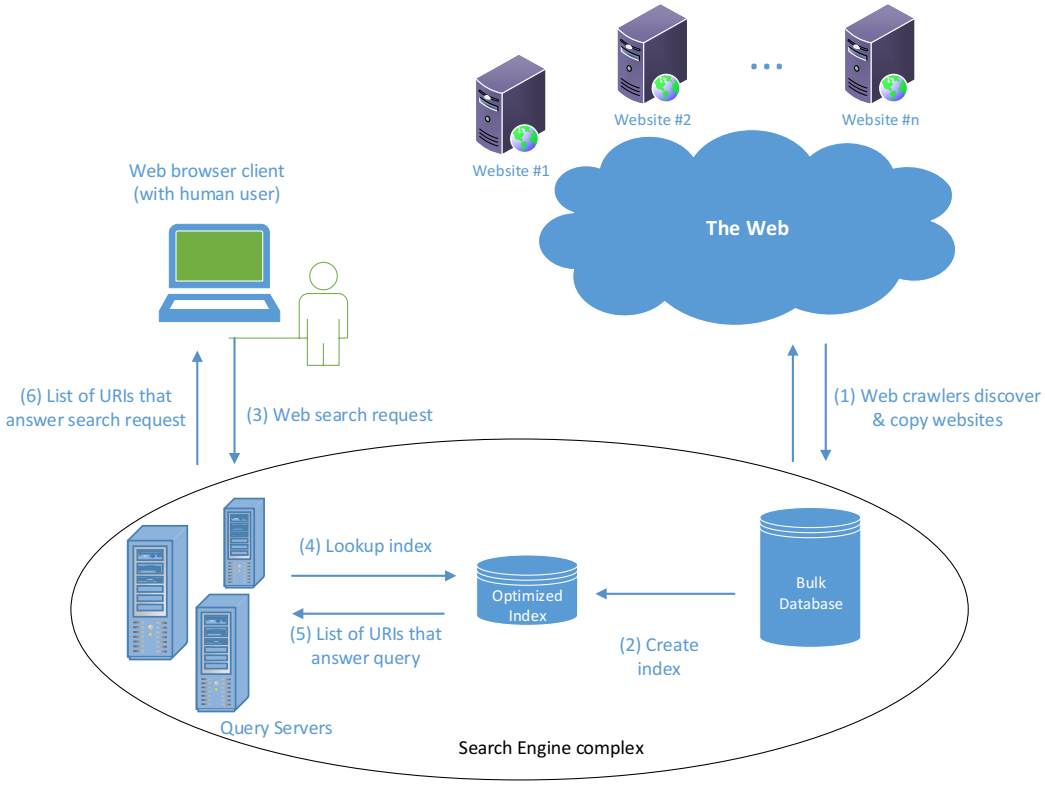

Figure 1: Overview of Traditional Resource Discovery Process by Web Search Engines

Following are some key observations about resource discovery in the traditional Web:

1) In terms of network configuration, the search engine functionality, or whichever entity is dispatching the Web crawler, is typically located on a set of centralized servers and related databases with high speed and large bandwidth Internet connectivity. The resources that are discovered by the Web crawlers may be widely distributed across the entire Internet. The Web browser based clients that interface to the human users and sends the search (lookup) requests are typically located at the edge of the network.

2) Search engines primarily use a *pull model* to get resource information. In this approach, the receiving node (that is, search engine) goes out and explicitly requests information (that is, via Web crawlers) from the sending node (that is, content websites). However, a small number of URIs such as the initial seed list of URIs (for example, very popular websites) may be obtained without using the pull model, but these are always a small fraction of the URIs in a search engine index.

3) The list of resources returned by the search engine for a given Web search (lookup) request may vary from a few URIs to potentially hundreds or even thousands of URIs. The order that these URIs are presented to the human user via the search engine Web interface is called the ranking of the resources. This is critical because when there is a large number of URIs returned to the user for a given search request, the user will typically only select ("click on") the top few ranked URIs.

4) The ranking of resources is ultimately an algorithmic decision internal to the search engine. However, it can be affected by external input such as *Search Engine Optimization* (SEO) techniques

which are used by website developers to try to get their specific URIs ranked higher by search engines than other URIs with similar application content. For example, a simple SEO technique is to have website content clearly tagged (for example, titles, section headings, etc.) and correlated to the website metadata. This metadata is an important input for the search index engine. A more sophisticated SEO technique is to have hyperlinks to a given website from as many other websites as possible as this is taken as a measure of content popularity by search engines. There are many other SEO techniques [4].

The Resource Discovery Problem in IoT

As mentioned previously, a key characteristic of current Web discovery technology is the use of Web crawlers to fan out and discover resources across the Internet. However, the implicit assumption in this approach is that Web servers are always active and available to be easily discovered by Web crawlers which arrive in an unscheduled manner. However, this conflicts with the expected nature of many IoT devices which may have only intermittent connectivity to the Web.

The primary reason for this intermittent connectivity is because many IoT devices will have a limited power supply (for example, battery or solar power), and so to conserve their power they may only "wake up" (that is, become active) when required to perform a specific function. For example, a fire detection sensor, acting as a mini Web server, may only wake up and connect to the Web to send a warning message to a remote controller when it senses a certain amount of smoke in its vicinity. At most other times, the fire detection sensor will be "asleep" (that is, in a low power state and not active) and unreachable via the Web. A secondary reason for intermittent connectivity is because many IoT devices will be connected to the Web by low power and lossy wireless networks. These wireless networks are more susceptible to interference and temporary loss of connectivity than traditional wired or cellular networks [5].

Another key difference between IoT devices and other Web infrastructure is that a majority of IoT devices may be deployed in semi-closed networks. For example, the IoT devices in a home such as a lighting or heating control system may have Internet connectivity only through a fire-walled home gateway. So the IoT devices and their associated resources may be accessible by the home owner through a smart phone control application with the proper security credentials from anywhere in the Internet. However the home IoT devices will not, for example, be discoverable by Web crawlers dispatched by a search engine as the Web crawlers will not be able to traverse the fire-walled home gateway.

Therefore, the current pull model of Web discovery cannot be directly applied to the expected deployments of IoT networks. In other words, current Web crawler technology will not be able to reliably discover a significant percentage of IoT devices which may be asleep or unconnected for significant periods of time, or may be located in semi-closed networks. This will result in traditional Web discovery techniques not producing accurate discovery results for IoT scenarios.

Resource Directories to Solve the IoT Discovery Problem

The solution currently being standardized in IETF to address the IoT resource discovery problem is based on a new logical network node called the *Resource Directory* [6] [7]. The RD idea was originally conceived and validated in the *European Union* (EU) funded SENSEI research program before coming to the IETF for standardization [8]. The RD is defined in [6] to be applicable to a given domain and not the entire Web. The domain is a logical grouping of IoT devices that are related to an RD. An RD may support multiple domains. The details of defining the extent of a given domain boundary however is left to implementation and not specified. Typically, the RD domains specified in IETF use cases are building-wide, campus-wide or city-

wide. The domain concept maps well into the expected deployment model of IoT devices in semi-closed networks. For example, in the simplest case there would be a one-to-one mapping between each semi-closed network and a domain. The RD approach thus provides a distributed resource discovery mechanism for IoT scenarios. Figure 2 shows an example of some typical RD domains.

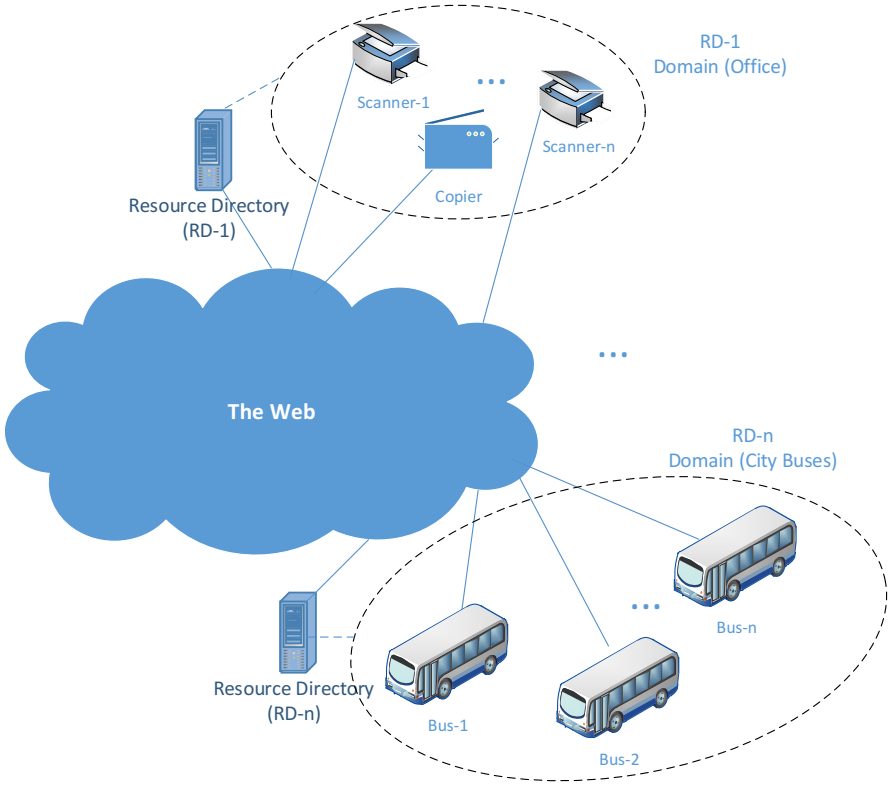

Figure 2: Typical Resource Directory Domains

The resource registration step is done in a push fashion by IoT devices acting as mini Web servers pushing their resource information into the RD resource database. Figure 3 shows the architecture of a given RD. All the IoT devices acting as Web servers will first register their resources (URIs) via a registration interface. Discovery can then be performed on the registered resources by an IoT client using the lookup interface. Mutual authentication, encryption and access control are required for both the registration and lookup interfaces to ensure security and privacy of the entire resource discovery process.

A given device may use both the registration interface (as a Web server), and the lookup interface (as a client). The client may be located anywhere in the Web but must have some knowledge regarding which specific RD to direct the resource discovery request to. For example, a newly installed home light controller may perform a lookup on its own home RD to find all the lights installed in the house. Or, a national smart grid controller may perform a lookup on a known RD in a remote city to find all the electric transformers located in that city.

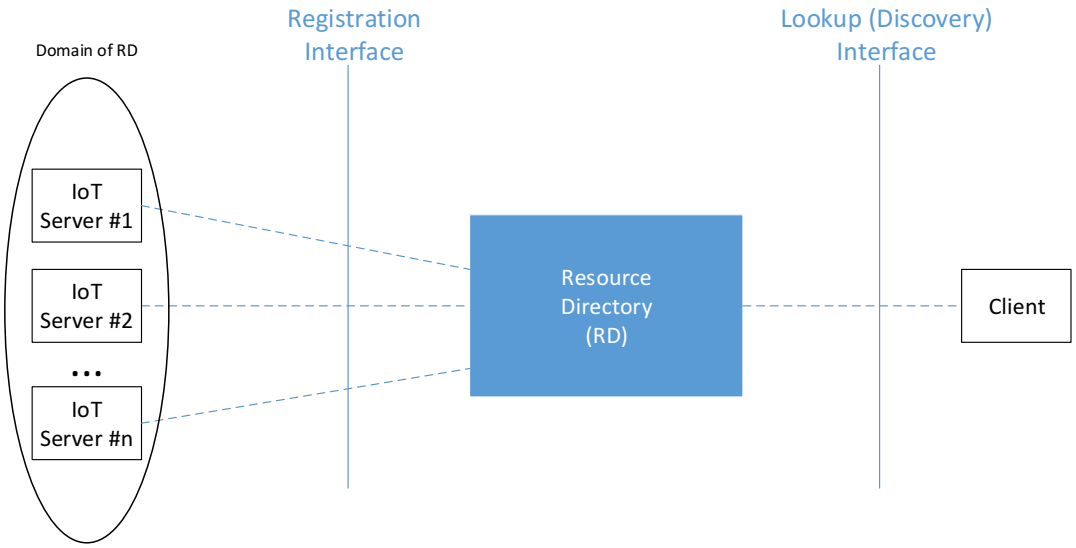

Figure 3: IoT Resource Directory Architecture (adapted from [6])

IoT devices communicate with the RD using a *Representational State Transfer* (REST) based protocol similar to HTTP but optimized for IoT. This protocol is referred to as the *Constrained Application Protocol* (CoAP) [9]. The resource information pushed by the IoT servers into the RD uses CoAP messages with a specific payload format termed the *Link Format* [7]. Only the URI, hyperlinks and some metadata is sent from the IoT device to the RD. Application content is not sent to the RD. Table 1 shows a comparison of the main resource discovery features of a traditional Web search engine and an IoT RD.

Table 1: Comparison of Resource Discovery Features of Web Search Engine versus IoT Resource Directory

Node / **Characteristic**	**Traditional Web Search Engine (for example Google)**	**IoT Resource Directory**
(1) How is resource information initially received by node?	Mainly pulled from target website by Web crawlers after initial visit	Mainly pushed by target IoT devices directly to RD (usually after power up)
(2) How is updated resource information transferred to node?	Pulled from target website by Web crawlers that re-visit according to their search engine policy	Pushed by target IoT device directly to RD according to their own update policy
(3) What resource information is transferred to node?	The entire website (that is, URIs, hyperlinks, metadata, and most application content)	URIs, hyperlinks, and metadata (but no application content is transferred)
(4) What transfer protocols are supported?	HTTP	CoAP (also some limited HTTP support exists. Further possible enhancements are discussed in [12])
(5) What is the scope of a client discovery request for resources?	Global (that is, covers entire Internet)	Local within given RD domain (for example, city-wide)
(6) Typical end user that generates query for resources	Human user (via a Web browser client) sends a search request	- IoT device (that is, acting as both the client and end user) sends lookup request - May also be used occasionally by human user (for example, via a CoAP enabled Web browser client as part of management activities)
(7) Are resource discovery results ranked?	Yes	No (but being discussed as a future enhancement in [12])
(8) Are the resource discovery results machine readable?	No (but may support it in the future with further adoption of Semantic Web concept)	Yes (that is, results strictly follow Link Format [7])

Resource Directory Protocol Considerations

As mentioned previously, CoAP is a Web transfer protocol, similar to HTTP, but optimized for IoT scenarios. CoAP provides a request/response interaction model between clients and servers. It supports key Web concepts such as URIs and Internet media types. CoAP messages are sent over *User Datagram Protocol* (UDP), and the CoAP header is encoded in a simple binary format. A CoAP request consists of a method (that is, GET, PUT, POST, and DELETE) that is applied to a resource identified by its URI, and a payload described by an Internet media type as well as other meta-data. CoAP messages may easily be inter-worked with HTTP in the forward or reverse directions via special cross-protocol proxies [9]. In addition, CoAP leverages *Datagram Transport Layer Security* (DTLS) [10] to provide a secure session between the communicating parties.

In CoAP, every physical IoT device is assumed to have one or more resources each identified by a URI. A resource may contain application information gathered by the IoT device (for example, temperature), or may be a method to control the device (for example, turn it ON/OFF). An example CoAP request and response pair is shown in Table 2.

Table 2: Example CoAP GET Request and Response

Request	GET coap://heater.net/temperature Note: Where Method = GET URI = coap://heater.net/temperature URI-Scheme component = coap:// URI-Host component = heater.net (or alternatively may be an IP address and Port Number) URI-Path component = /temperature
Response	2.05 Content "22.3 C" Note: Where Response code = 2.05 (indicating successful processing) Payload = 22.3 Celsius (C) temperature reading

The following sections describe the key protocol steps and security characteristics related to RDs.

Finding the Resource Directory

The first step is for the IoT devices, or *End Points* (EPs) as they are called in [6], to find the appropriate RD. The most dynamic method for finding the RD is using IP multicast. Specifically, the device will send a CoAP multicast message to the CoAP IPv4 or IPv6 addresses reserved for this purpose [11]. An alternative method would be, for example, factory pre-provisioning of the RD information in the device.

Assuming the IP multicast method of finding the RD, each device (EP) sends a CoAP GET request to a specific URI-Path as shown in Figure 4. Specifically, the CoAP GET request is sent by multicast to the reserved "**`/.well-known/core`**" URI-Path. (Note that the URI-Scheme and URI-host components are not shown for simplicity in this and subsequent figures). All the devices in the domain will then get this request because it is sent by IP multicast [11]. However, only the RD will reply as the request URI has a query string for resource type (**`rt`**) added to the end (that is, **`?rt=core.rd*`**). This indicates that the message is meant for the RD. The RD then responds indicating its URI-Path (that is, **`/rd`**) for subsequent registration or lookup requests [6].

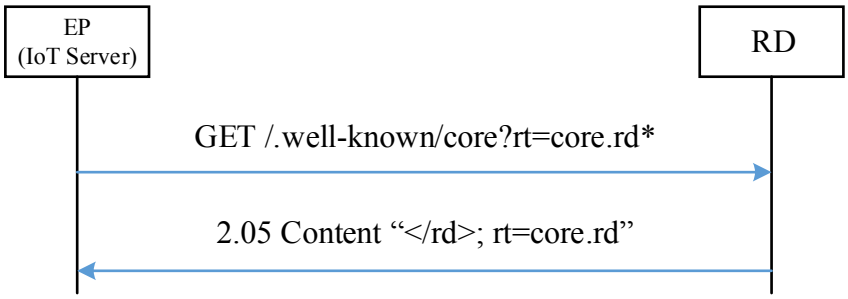

Figure 4: Finding a Resource Directory (adapted from [6])

Registering Resources

After finding the RD, each IoT device (EP) will register its own resources to the RD using the RD's registration interface as shown in Figure 5. This is accomplished by each device sending a CoAP POST request directly to the RD with its list of URIs (that is, **`/sensor...`**) in the message payload, along with a query string identifying the registering device (that is, **`?ep=node1`**). The message payload containing the list of URIs being registered is formatted in the Link Format [7]. The RD then responds with the resulting URI-Path (that is, **`/rd/4521`**) that it created to store the device's resources [6].

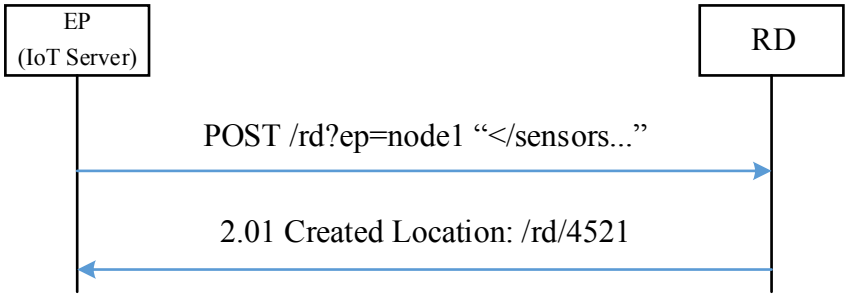

Figure 5: Registration of URIs to a Resource Directory (adapted from [6])

Resource Lookup (Discovery) by Client

The RD also supports a lookup interface for clients to make a discovery request on the RD database. The client may be located in the RD domain or may be outside of it. The client is aware of a given RD because it used the locating mechanism described previously, or it may have learned of the RD through other methods (for example, pre-provisioning). Figure 6 shows a typical resource lookup request where a client is interested in finding all URIs related to "temperature." Specifically, the client will send a GET request to the RD Lookup interface indicating that it is interested in the resource type of temperature in the query string (that is, **`?rt=temperature`**). The RD will then respond with a message containing the list of URIs of all the devices that it has in its registration database which match this criteria [6]. The response message is formatted using the Link Format [7].

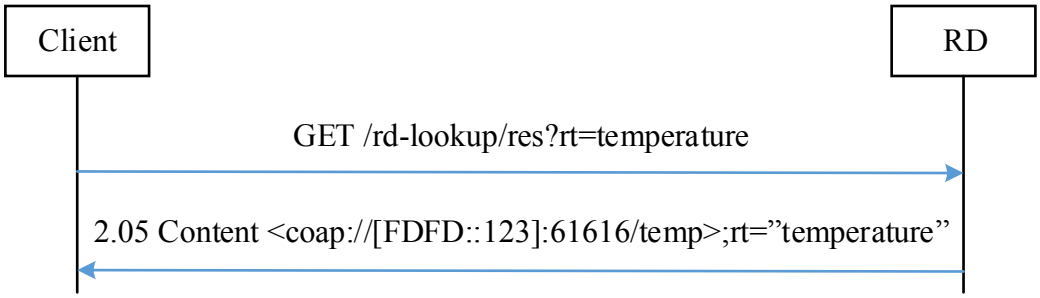

Figure 6: Resource Lookup (Discovery) Request sent to a Resource Directory (adapted from [6])

Other types of lookup requests may also be sent. For example, the RD may be queried to find out all the URIs supported by a given IoT device. Or, the RD may be queried to find out the identities of all the IoT devices in a given domain. The great majority of lookup requests to the RD will be sent by other IoT devices, without any human in the loop, for automated command and control. However, resource lookup requests may also be sent occasionally by humans via a CoAP enabled Web browser interface for management activities.

Resource Directory Security Characteristics

Both the RD registration and lookup interfaces are protected by multiple layers of security to ensure that only authorized parties are able to access the RD. Specifically, mutual authentication is first required between the RD and any device attempting to access it. This is accomplished using either pre-shared encryption keys, raw public keys, or X.509 security certificates as the security credentials [9]. The appropriate credentials are used in the initial handshake of the DTLS session establishment to perform mutual authentication between the RD and the device or client accessing it. Once the mutual authentication is completed, the cipher suite to be used for the DTLS session is negotiated. Then all subsequent messages exchanged between the RD and the device or client are securely encrypted via DTLS so that no unauthorized third party can decipher the communications [6].

In addition to the DTLS security, the RD will also perform a fine grained access control of any device attempting to communicate with it. Access control will be performed separately on the RD registration and lookup interfaces. Access control may be performed either at the domain, device or resource level [6]. This is especially important on the lookup interface for privacy and security reasons. For example, in a hospital setting many medical devices such as blood pressure monitoring devices may be registered to a RD, but only authorized medical staff should be able to discover a given device for privacy reasons. Or in a home setting, a visitor may be allowed to freely discover the television, but will blocked from discovering the front door lock for security reasons.

Examples of Resource Directory Implementations

In parallel with the ongoing standardization efforts in IETF for the RD protocol [6] [12], there are several open source and commercial instances of RDs that have successfully interoperated with various IoT devices. Some examples are briefly described below.

The Californium open source software project is a popular CoAP framework for IoT deployments. It is written in the *Java* programming language and specifically includes support for back-end infrastructure as part of its project scope. As such, it has released software loads that implement RD functionality which can be run on general purpose servers [13].

On the commercial front, the semiconductor and software company, ARM, has released several products for the IoT market. One of their products is a middleware offering called the "mbed Device Server." This middleware includes support of RD functionality. This middleware software can run on various server hardware platforms [14].

Another company which has done a lot of RD development work is the telecommunications equipment and service provider, Ericsson. Ericsson has done a lot of early prototyping and research [15] in the RD concept starting from the initial EU SENSEI project days [8]. They have also participated in an open source software project for a cloud based IoT gateway that includes RD functionality [16].

Alternative Approaches to Discovery

The RD is not the only approach to the discovery problem for IoT networks. Alternatively, there are other methods such as *Domain Name Service – Service Discovery* (DNS-SD) which allows lookup of a given service via DNS [17]. There is also *Universal Plug and Play* (UPnP) which allows discovery of devices in home networks [18].

The key difference between these other discovery methods and the RD approach is that the RD is geared towards resource discovery in the context of a REST based Web model which means discovery of URIs and related metadata. The other existing discovery approaches are mainly oriented to discovering IP addresses, ports and related parameters. So they are complementary to the URI discovery methods but cannot replace them. The only other widely used URI discovery scheme is the Web crawler approach described previously which has the shortcomings in IoT deployments as described in Table 1.

Conclusion

The existing REST based Web architecture and protocols have been extremely successful and a driving force behind the explosive growth of the Internet during the last twenty years. A key part of the Web's success are search engines like *Google* and *Bing* which use Web crawlers to discover resources (that is, URIs) efficiently. However, the existing model of resource discovery is expected to undergo radical changes with the addition in the future of an increasing number of IoT devices acting as both mini Web servers and clients. IETF is currently standardizing protocol support for the Resource Directory which will be optimized for distributed IoT resource discovery.

It is expected that an increasing number of discovery requests in the future will be handled by RDs for scenarios involving IoT devices. In parallel, human users will continue to heavily use traditional Web search engines like Google. There is also expected to be some cross-usage as traditional Web browsers may start to support CoAP software modules (plug-ins) and hence allow human users to directly make queries to RDs. However, a limiting feature of this interaction will be the security and privacy requirements of IoT deployments. Specifically, many IoT resources such as personal health monitoring devices will have sensitive information which is not meant for public distribution, and they may also be located in semi-closed networks. Strong security and privacy is supported by the current RD model which requires strict mutual authentication, encryption and access control for both registration and discovery of IoT resources.

References

[1] T. Berners-Lee, et al., "Uniform Resource Identifier (URI): Generic Syntax,"
IETF RFC 3986, January 2005.

[2] R. Fielding, et al., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing,"

IETF RFC 7230, June 2014.

[3] The Web Robots Pages: **http://www.robotstxt.org/**

[4] "What Is SEO, Search Engine Optimization?,"
http://searchengineland.com/guide/what-is-seo

[5] M. Dohler, et al., "Routing Requirements for Urban Low-Power and Lossy Networks," IETF RFC 5548, May 2009.

[6] Z. Shelby, et al., "CoRE Resource Directory," Internet Draft, work in progress,
draft-ietf-core-resource-directory-07.txt, March 2016.

[7] Z. Shelby, et al., "Constrained RESTful Environments (CoRE) Link Format," IETF RFC 6690, August 2012.

[8] S. Jokic, et al., "Evaluation of an XML Database Based Resource Directory Performance,"
http://www.smartsantander.eu/downloads/Presentations/XML_RD_Telfor_2011_v1.0Srdjan.pdf

[9] Z. Shelby, et al., "The Constrained Application Protocol (CoAP)," IETF RFC 7252, June 2014.

[10] E. Rescorla, et al, "Datagram Transport Layer Security Version 1.2", IETF RFC 6347, January 2012.

[11] A. Rahman, et al., "Group Communication for the Constrained Application Protocol (CoAP)," IETF RFC 7390, October 2014.

[12] A. Rahman, "Advanced Resource Directory Features," Internet Draft, work in progress,
draft-rahman-core-advanced-rd-features-02.txt, March 2016.

[13] Californioum (Cf) CoAP Framework:
http://www.eclipse.org/proposals/technology.californium/

[14] ARM mbed Device Server:
https://www.mbed.com/en/development/cloud/mbed-device-server/

[15] Ericsson Research Blog, Having a headache using legacy IoT devices?
http://www.ericsson.com/research-blog/internet-of-things/headache-using-legacy-iot-devices/

[16] Ericsson Research Blog, A Computational Engine for the Internet of Things:
https://www.ericsson.com/research-blog/internet-of-things/computational-engine-internet-things/

[17] S. Cheshire, et al., "DNS-Based Service Discovery," IETF RFC 6763, February 2013.

[18] Wikipedia, "Universal Plug and Play":
https://en.wikipedia.org/wiki/Universal_Plug_and_Play

AKBAR RAHMAN is a Principal Engineer at InterDigital Communications and is based in their office in Montreal, QC, Canada. He has been closely involved in IoT protocol development at IETF for several years. He has multiple IETF RFCs published in the areas

of IoT and Internet architecture. He has a BASc from the University of Waterloo, Canada. He can be reached at **Akbar.Rahman@InterDigital.com**

CHONGGANG WANG is a Member of Technical Staff at InterDigital Communications and is based in their office in King Of Prussia, PA, USA. He is Editor-in-Chief of the IEEE IoT Journal, and a Distinguished Lecturer for the IEEE Communications Society. He has a PhD from the Beijing University of Posts and Telecommunications. He can be reached at **Chonggang.Wang@InterDigital.com**