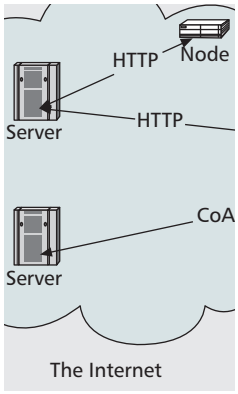
EMBEDDED WEB SERVICES

ZACH SHELBY, SENSINODE LTD.



The authors present an overview of the Web architecture, its core REST concepts and the current state-of-the-art in Web Services. Two key activities required in order to achieve efficient Embedded Web Services are introduced.

ABSTRACT

The Internet of Things is the next big possibility and challenge for the Internet. Extending the web architecture to this new domain of constrained wireless networks and devices will be key to achieving the flexibility and scalability needed to make it a success. Web services have proven to be indispensable in creating interoperable communications between machines on today's Internet, but at the same time the overhead and complexity of web service technology such as SOAP, XML, and HTTP are too high for use in the constrained environments often found in machine-to-machine applications (e.g., energy monitoring, building automation, and asset management). This article first gives an overview of the web architecture, its core REST concepts, and the current state of the art in web services. Two key activities required in order to achieve efficient embedded web services are introduced: a fresh approach to a web application transfer protocol and efficient payload encoding. The article analyzes the most promising payload encoding techniques and introduces the new IETF Constrained RESTful Environments (CoRE) standardization activity.

INTRODUCTION

The spread of ubiquitous Internet protocol technology over the next decade to automation systems, mobile personal devices, building automation, the smart grid, and other smart objects will involve a revolution on the same scale as the spread of the Internet in the '90s. Let's call this revolution the Internet of Things. It will potentially involve trillions of embedded devices, and will greatly increase the size and scope of the Internet.

Figure 1 shows one onion ring model for the Internet of Things. We can think of the Internet of Things as a third ring of today's Internet, which already consists of a stable core (routers, servers) and the quickly growing fringe (personal computers, smart phones). As new embedded applications such as smart metering, logistics, building automation, and personal sensing become IP enabled, they add an entirely new dimension to the Internet with exciting possibilities and challenges. The integration of low-power wireless and wired technology into the Internet architecture is required to enable the Internet of Things on a universal scale.

The IPv6 over low-power wireless area network (6LoWPAN) standards [1, 2] are playing a key role in driving the Internet of Things. This new standard now allows the use of IPv6 with even the most constrained networks; for example, the popular IEEE 802.15.4 wireless standard, industrial, scientific, and medical (ISM) band telemetry radios, over low-rate power-line communications (PLC), using very simple embedded microcontrollers. Together with Internet Engineering Task Force (IETF) routing over low-power and lossy network (ROLL) standardization, the recent ZigBee adoption of IPv6, and maturity in IPv6 technology, we have a complete set of networking technologies for integrating embedded devices and systems into the Internet. Networking alone, however, does not make the Internet useful. Applications today depend on the Web architecture using the Hypertext Transfer Protocol (HTTP) and Extensible Markup Language (XML) to create web services. This powerful and extensible paradigm is quickly making its way into Internet of Things applications, allowing today's web developers to continue using their existing skills.

Web service technology today is a poor match for the vast majority of constrained networks, machine-to-machine (M2M) applications, and devices that could potentially benefit from Internet protocols. At one extreme, embedded PCs running full IPv6 protocol stacks on Ethernet-like networks have no problem with today's web services. At the other extreme, constrained low-power and lossy networks involve high packet loss (5–10 percent is common), frequent topology change (due to radio frequency [RF] fading and mobility), low throughput (10 kb/s is typical), and useful payload sizes often less than 100 bytes. At the same time, constrained embedded devices are optimized for low cost and power consumption, dedicating most of their resources to embedded tasks. How can we make the web a reality for this class of networks and devices?

The interaction model of M2M applications differs greatly from how web services are used today. Exchanges are short-lived, battery-powered devices mostly sleep, multicast is commonly required in automation, and flow control is needed across the entire M2M network. The basic nature of M2M applications is often asynchronous rather than synchronous. What would a web service transfer protocol aimed specifically at M2M applications look like?

This article gives an overview of web services today and recent advancements in embedded

1536-1284/10/$25.00 © 2010 IEEE

web service technology. This includes optimizations in payload encoding, application protocols, security, and web discovery while evolving the web architecture to support new interaction models and the special requirements of M2M applications. The new IETF Constrained RESTful Environment (CoRE, datatracker.ietf.org/wg/core) standardization effort is introduced, which aims at standardizing an application protocol for embedded web services.

Many projects have been actively moving toward embedded web services. Major European Framework Programme 7 (FP7) projects include SENSEI (www.sensei-project.eu), which aims at fundamentally integrating the embedded and Internet worlds, Sensors Anywhere (SANY, sanyip.eu), which built a sensing framework using Open Geospatial Consortium tools, and Global Sensor Network (GSN, sourceforge.net/projects/ gsn), which is a sensor network middleware.

In the end, standardization will be the most important factor in the success of embedded web services. Important activities include IETF 6LoWPAN, ROLL, and CoRE, the World Wide Web Consortium (W3C) Efficient XML Interchange (EXI) effort [3], the European Telecommunications Standards Institute (ETSI) M2M technical committee, the Open Geospatial Consortium, OASIS open information exchange efforts such as Open Building Information Exchange (oBIX), and IPv6 automation profile activities around ZigBee IP.

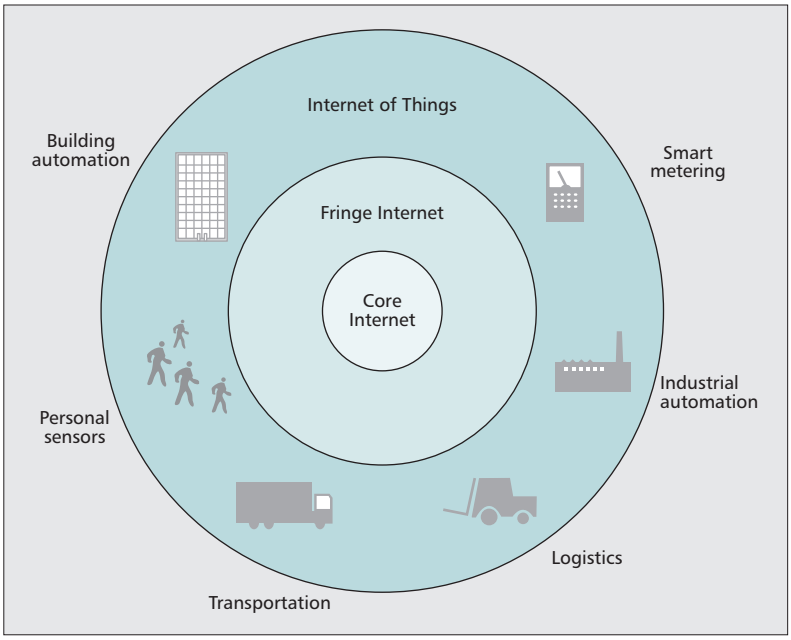


Figure 1. *The Internet of Things is quickly increasing the size of the Internet.*

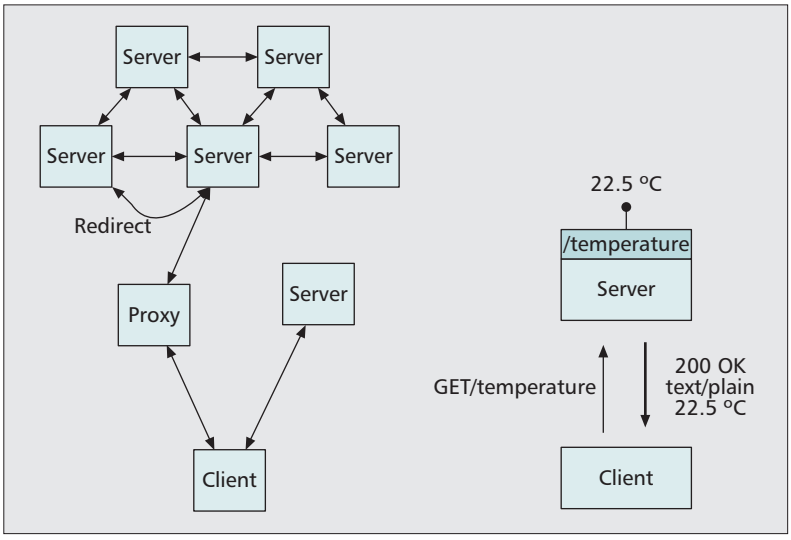


Figure 2. *The web architecture is flexible and scalable, and resources are server controlled. In the example a resource related to indoor temperature is requested by a client. The server returns the current value of 22.5 °C in text format.*

Web Services

To understand how to extend the web to constrained networks and devices, it is important to understand the core architecture behind the web. While the Internet protocol ties together heterogeneous networks into the Internet, the Web is a loosely coupled application layer architecture. Resources are key to the web architecture, which are server-controlled abstractions made available by an application process and identified by Universal Resource Identifiers (URIs). These server controlled resources are accessed by clients in a synchronous request/response fashion using methods such as GET, PUT, POST, and DELETE of HTTP, as shown in Fig. 2. Resource state is kept only by the server, which allows for the caching, proxying, and redirection of requests and responses. Web resources may contain links to other resources, which creates a distributed web between Internet endpoints, resulting in a highly scalable and flexible architecture as shown in Fig. 2. These core web concepts are commonly described as Representational State Transfer (REST) [4, 5]. Today, HTTP is almost exclusively used on today's Internet for manipulating web resources, although REST can also be realized with other application protocols.

What Is a Web Service?

The web is usually used by humans to access hypertext (HTML) and other media via web browsers; however, more and more often the web is used for communications between machines. Interoperable communications between machines on the Internet is often achieved using web services. There are two general ways of realizing web services: applying REST for the manipulation of resources using HTTP, or via remote procedure call (RPC) style interactions using, say, the Simple Object Access Protocol (SOAP).

The structure of a typical web service stack is shown in Fig. 3, showing both SOAP and REST. RESTful web services make use of HTTP to access resources, and may be formally described using, for example, the Web Service Description Language (WSDL) or Web Application Description Language (WADL). An HTTP request message is first sent from the client to the server. The client specifies the resource on the server to request in the form of a URI and method (GET, POST, PUT or DELETE). The server responds with a possible response body and a code. HTTP is able to carry an arbitrary payload using any

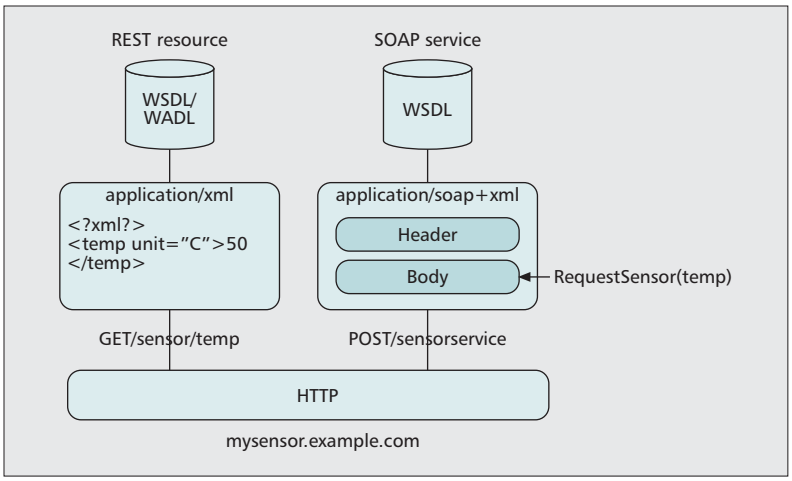


Figure 3. *SOAP and REST style web services.*

MIME type and encoding. The body of the HTTP web service message is in a format known by the client and server, often XML following a mutually understood Schema.

In RPC style web services, the HTTP URL and POST method are only used as a pipe for incoming and return SOAP calls, which are formally described by WSDL. In order to perform sequences of RPCs the SOAP protocol is included in the XML body, adding a layer of complexity and dependence on XML.

When designing interoperable communications for constrained embedded devices, the RESTful paradigm has many advantages over RPC style interactions. These include less overhead, less parsing complexity, statelessness, and tighter integration with HTTP.

Although HTTP is widely used for realizing web services, it is by no means the only protocol used between machines. The Extensible Messaging and Presence Protocol (XMPP, RFC 3920) is typically used for instant messaging and presence. XMPP is also being applied as a more general message-oriented middleware. The Session Initiation Protocol (SIP, RFC 3261), normally used to control multimedia sessions, is sometimes used also for communications in embedded applications. SIP is also an integral part of the IP Multimedia Subsystem (IMS) architecture. In specialized cases SOAP can also be bound to other transports, such as in the SOAP-over-UDP binding specification used in Device Profile Web Services (DPWS).

The Embedded Web

The technologies deployed for web services today are usually not suitable for constrained networks and devices, especially wireless battery power ones. Although RESTful web services are simple in concept, the protocols and payload formats used to realize them today were not designed with these types of applications, networks, or devices in mind. Problems with these protocols include:

Overhead: The ideal UDP payload over 6LoWPAN and IEEE 802.15.4 is just 60–80 bytes for reasonable performance (avoiding fragmentation). HTTP headers alone are easily larger than this. The use of XML payloads adds further overhead as it represents content inefficiently. The embedded web will require both highly efficient application transfer protocols and payload formats to meet the expected overhead.

TCP binding: Web services today depend on TCP, which has performance problems over lossy links, sensitivity to mobility, no multicast support and high overhead for short-lived transactions. Furthermore the congestion and flow control model does not match M2M interaction models well [6].

Pull model: In sensor networks wireless nodes are typically sleeping over 90 percent of the time, making the HTTP request/response pull model inappropriate. Other interaction models are needed such as eventing and publish/subscribe.

Complexity: Although HTTP may in theory seem simple, when used by modern HTTP servers, clients, and proxies it is not. HTTP has evolved into a highly complex protocol as used between modern servers and browsers. A large number of features and optional headers may be employed, increasing embedded device complexity. The use of XML as a payload adds further parsing complexity. In RPC web services this is further compounded by the use of SOAP.

In order to fully realize the Internet of Things, the RESTful web service paradigm needs to be extended into the constrained domain. To do this we need a fresh approach to both the transfer protocol used to convey REST semantics, and the payload formats exchanged between applications. At the same time, this needs to be done as a natural extension of today's HTTP web.

Payload Compression and Encoding

Reducing the overhead of XML is a key requirement for realizing embedded web services, as XML is both sparse and difficult to parse. This is challenging, as the complexity of creating and parsing content must be minimized for very constrained devices at the same time. Although compression algorithms such as gzip are commonly used between HTTP clients and servers, these algorithms are rarely appropriate for the embedded web because of high complexity and low performance for small payloads. An alternative to compressing XML is to encode it with an alternative binary representation. In this section promising technologies for encoding XML are introduced and compared.

Extensible XML Interchange (EXI) is a very compact representation for the XML Information Set that is intended to simultaneously optimize performance and the utilization of computational resources. The EXI format uses a hybrid approach drawn from information and formal language theories for encoding XML information. Using a relatively simple algorithm, it reliably produces efficient encodings of XML event streams. The EXI specification allows for a simplified mode of operation called schema-informed mode. Here a schema is used to form a grammar and state-machine limited to that schema, enabling extremely efficient encoding. The result of the simple state machine is that even the most minimal embedded devices can work directly with the encoding without the need

to work with a full XML parser. For a primer and more information about EXI, see [3].

Fast Infoset is an International Telecommunication Union — Telecommunication Standardization Sector (ITU-T) defined standard (X.891) that specifies a binary encoding for the XML Information Set based on ASN.1 encoding. Unlike XML compression schemes, the Fast Infoset standard has the dual benefits of both compression and performance, making it a good choice for moving large XML data between low bandwidth systems or for high performance systems such as servers utilizing web services.

Binary Extensible Markup Language (BXML) is a straightforward encoding for XML that is designed for efficient processing in general with an emphasis on dense numeric data, which is one of the weakest areas for textual XML. This encoding specification, designed by the Open Geospatial Consortium (OGC), is specified for GML compression.

Table 1 shows the results of a simple comparison between the EXI, BXML, and Fast Infoset encoding techniques. Typical sensor markup examples using the Resource Description Format (RDF), ZigBee Smart Energy, and OGC SensorML formats were used. Schema-informed EXI encoding is up to 97 percent smaller than the equivalent XML, providing overheads reasonable for the small usable payload of 6LoWPAN and similar constrained networks. It should be noted that special care must be taken in designing appropriate schemas and payloads for use with this type of encoding to reap the full benefits of EXI.

XML is not the only format used for payload representation in RESTful web services. One popular alternative is the JavaScript Object Notation (JSON) [RFC4627]. An example JSON encoding of the representation in Fig. 2 would be

```
{"temp_C": 22.5}
```

Industry segments often define their own payload representations, such as the binary representation of oBIX v1.1 in building automation, or American National Standards Institute (ANSI) C12 tables used in electrical metering. In simple cases, it sometimes even makes sense to represent a specialized payload as a binary octet stream (Content-Type: application/octet-stream).

IETF Constrained RESTful Environments

Recently the IETF started a new working group on Constrained RESTful Environments (CoRE). The aim of this new group is to extend the web architecture described earlier specifically for M2M applications using constrained networks and devices. The CoRE working group has the goal of producing two items:

- **Constrained Application Protocol (CoAP):** The group has defined an application transfer protocol that realizes a minimal subset of REST along with resource discovery, subscription/notification, and the use of appropriate security measures [7].

Encoding	Complexity	RDF	Smart Energy	SensorML
XML	Medium	206	409	300
*EXI**	Low	6 (3%)	13 (3%)	57 (19%)
BXML	Medium	177 (86%)	210 (51%)	177 (59%)
Fast Infoset	Medium	143 (69%)	200 (49%)	185 (62%)

* Strict schema-informed mode with bit alignment

Table 1. *Comparison of encoding results (size of encoded objects in Bytes).*

- **Security bootstrapping:** The group will also define a set of security bootstrapping methods for use in constrained environments in order to associate devices and set up keying material for secure operation.

The working group focuses on requirements coming from smart energy, building automation, and other machine-to-machine (M2M) applications. These applications have been analyzed in several publications and their requirements summarized in [8].

One view of the CoRE architecture is illustrated in Fig. 4. The constrained environment envisioned is an extension of the current web architecture. CoAP may be used between constrained devices, between servers and devices in the constrained environment, or even across the Internet. One important requirement of the working group is to ensure a simple mapping between HTTP and CoAP so that the protocols can be proxied transparently, or the same interface design can be realized over either transport. Finally, CoAP will support IP multicast, allowing it to be used for point-to-multipoint interactions commonly required in automation.

The Constrained Application Protocol

Where HTTP has become the ubiquitous protocol for the web we know today, CoAP aims at being the ubiquitous protocol for M2M applications over constrained environments on the Internet of Things. Just as HTTP was designed as a transfer protocol for web media content, CoAP is a transfer protocol that M2M applications use to realize web services in an interoperable way. CoAP could be thought of as a compression for HTTP, as the same application transfer paradigm and basic features are supported; however, CoAP is at the same time designed to be an ideal match for M2M applications.

In order to realize the web architecture while meeting the demanding requirements of M2M applications, CoAP has the following key features.

Compact header: CoAP includes a compact binary header with extensible options. The protocol has a base header size of just 4 bytes, and a total header of 10–20 bytes for a typical request.

Methods and URIs: In order for a client to access resources of a server, CoAP supports the request methods GET, PUT, POST, and DELETE. These methods are answered using a subset of HTTP compatible response codes. CoAP supports URIs, which is a key feature of the web architecture.

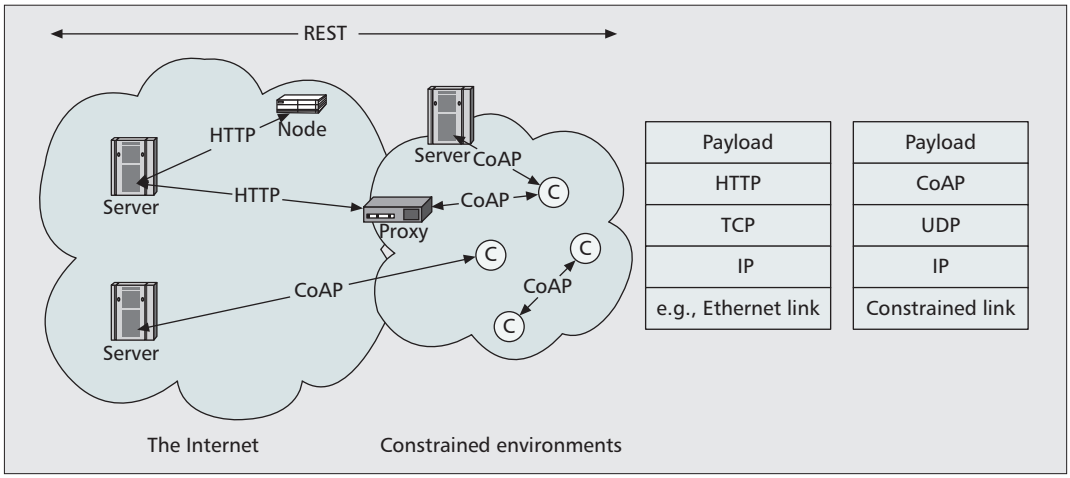


Figure 4. *The Constrained RESTful Environment architecture.*

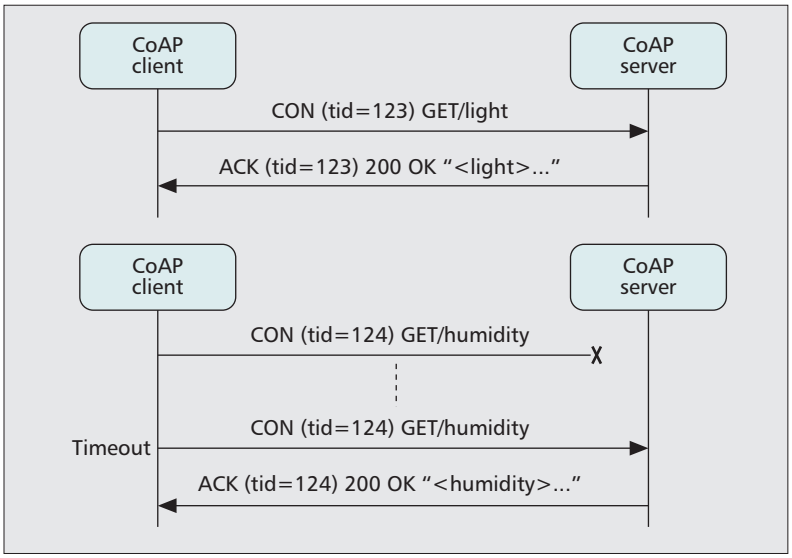


Figure 5. *Typical CoAP message exchanges. Above, a successful confirmable (CON) message is acknowledged (ACK). Below, a retransmission is performed before a successful ACK is received.*

Content types: As CoAP is a transport protocol for any number of different uses, it is possible to explicitly indicate the content type of the payload in the header. This is achieved by assigning codes to a subset of Internet media types.

Simple caching: Caching for resource representations is supported to optimize for the performance of constrained environments.

Transport binding: CoAP uses UDP for its transport, and provides a simple stop-and-wait reliability mechanism when needed by a request. The use of transport layer security is also supported through an optional binding to Datagram Transport Layer Security (DTLS).

Resource discovery: In order to find and use CoAP resources autonomously, a built-in resource discovery format is supported (think of this as index.html for machines). This may be used to discover the list of resources offered by a device, or for a device to advertise or post its resources to a directory service. Resource discovery is discussed further.

Subscription: One major shortcoming of HTTP is its reliance on a pull model. CoAP uses an asynchronous approach to support the push of information from servers to clients using subscriptions. Subscription is discussed further.

The protocol stacks of HTTP and CoAP are compared in Fig. 4. CoAP provides a very simple and low-overhead way to achieve web service interactions between constrained devices over challenging wireless networks like IEEE 802.15.4. Figure 5 shows a typical CoAP message exchange, first a successful request followed by a request requiring retransmission. The full specification of CoAP is available at [7].

Resource Discovery

The discovery of resources is common on the web, and is called web discovery in the HTTP community. One form of web discovery is performed by humans accessing the default resource of a server (index.html), which often includes links to other web resources available on that or related servers. Web discovery can also be performed by machines if standardized interfaces and resource descriptions are available. New approaches being standardized at the IETF include the well-known resource path /.well-known/scheme (RFC 5785), formats such as the HTTP link header format, the Atom Publishing Protocol (RFC 4287), HTML links, or Link-based Resource Descriptor Discovery (LRDD) using the XML Resource Descriptor (XRD) format. Several related techniques are in common use today. Web-based Distributed Authoring and Versioning (WebDAV) is a protocol used to manage, change, and move documents on web servers (RFC 4918). The discovery and authentication of user identity on web sites is often achieved using OpenID, while resource sharing to third parties using OAuth is a complimentary approach. OpenID and other frameworks make use of Extensible Resource Identifiers (XRIs) and XRI Data Interchange (XDI) to provide digital identifiers along with the Extensible Resource Descriptor Sequence (XRDS) format for discovering associated services.

In CoRE we are dealing with autonomous devices and embedded systems; thus, the importance of uniform, interoperable resource discov-

ery is much greater than on the web today. To ensure interoperability between CoAP endpoints, the protocol includes a technique for discovering and advertising resource descriptions. As these descriptions are machine interpreted, the description format itself is also being standardized.

To achieve resource discovery, CoAP servers are encouraged to provide a resource description available via a well-known URI such as /.well-known/core for resource discovery. This description is then accessed with a GET request on that URI. The same description could be advertised, or even posted to a directory of descriptions. The description format is based on the HTTP link header format [9] as an Internet media type carried in the payload, which is simple and easy to parse.

Subscription

On today's web the request/response nature of HTTP assumes that transactions are always client initiated, often called a pull model. This makes it inefficient for a client to know when a resource has changed, requiring it to periodically poll the resource. In embedded systems the problem is compounded by low-power wireless devices, which are in sleep mode most of the time and unable to answer poll requests. Recent approaches on the web for moving beyond the pull model include various HTTP bidirectional and long poll techniques, or publishing link feeds using Really Simple Syndication (RSS) or the Atom Publishing Protocol.

In CoRE the support for a native push model for servers to initiate transactions with clients was identified as an important requirement. Realizing a push model for CoAP just requires a subscription interface for requesting a response about a particular resource each time it changes, as specified in [10]. Due to the asynchronous design of the transfer protocol over UDP, no special notification messages are needed.

Conclusion

This article introduces recent activities in the extension of the web architecture into constrained environments using embedded web services. By applying the fundamental REST architecture to this domain, efficient web service interfaces between embedded machines can be achieved. But this requires a fresh approach to the application transfer protocol, transports, and payload formats used. Recent advancements in XML encoding with W3C EXI along with industry-specific formats have shown promising results in minimizing payload overhead and parsing complexity. The new IETF Constrained RESTful Environments working group is developing a new application transfer protocol designed specifically for realizing embedded web services. This is being achieved by realizing only the core functionality of REST, along with features not included in HTTP such as built-in resource discovery and subscription.

The Internet of Things will still bring Internet and communications engineers plenty of challenges in the future as the scale of the Internet grows from billions to trillions of nodes and the interaction models change as autonomous interactions start to outnumber human initiated transactions. The most important challenges to come include routing scalability, naming and identification, security, privacy, and the manageability of such a network.

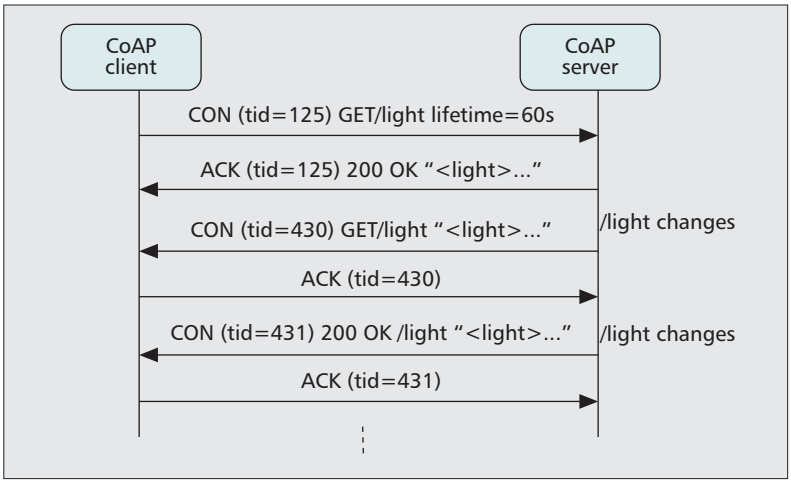


Figure 6. *A typical CoAP subscription message exchange.*

Acknowledgments

The author thanks Carsten Bormann, Lisa Dusseault, and Cullen Jennings for the inspiring discussions leading up to this article and for useful comments that improved the manuscript. He also thanks his co-authors on CoAP related work, colleagues in the SENSEI project, and all the dedicated folks who made the CoRE working group a reality. This work was partially funded by SENSEI, a Large Scale Collaborative Project supported by the European 7th Framework Programme, contract number 215923.

References

[1] Z. Shelby and C. Bormann, *6LoWPAN: The Wireless Embedded Internet*, Wiley, 2009, p. 254.
[2] G. Montenegro *et al.*, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks," IETF RFC 4944, Sept. 2007.
[3] Efficient XML Interchange (EXI) Primer http://www.w3.org/TR/exi-primer/.
[4] R. T. Fielding, "Architectural Styles and the Design of Network-Based Software Architectures," Ph.D. Thesis, University of California, Irvine, 2000.
[5] RESTful Web Services http://java.sun.com/developer/technicalArticles/ WebServices/restful/
[6] L. Eggert, "Congestion Control for the Constrained Application Protocol (CoAP) ," draft-eggert-core-congestion-control-00, work in progress, June 2010.
[7] Z. Shelby, B. Frank, and D. Sturek, " Constrained Application Protocol (CoAP)," draft-ietf-core-coap-02, work in progress, Sept. 2010.
[8] Z. Shelby *et al.*, "CoAP Requirements and Features," draft-shelby-core-coap-req-00, work in progress, Feb. 2010.
[9] M. Nottingham, "Web Linking," IETF RFC 5988, Aug. 2010.
[10] K. Hartke and C. Bormann, "Observing Resources in CoAP," draft-hartke-coap-observe-01, Aug. 2010; draft-hartke-coap-observe-02, work in progress, Oct. 2010.

Biography

Zach Shelby (zach@sensinode.com) is co-founder and head of research at Sensinode Ltd., where he leads Internet of Things research, standardization and IPR activities for the company. Before starting Sensinode, he led wireless networking research at the Centre for Wireless Communications, Finland. He is actively involved with IETF standardization and EU research projects, and is co-author of the recent book *6LoWPAN: The Wireless Embedded Internet*.