UNIT-V

From the internet of things to the web of things-Designing RESTful smart things-Modeling Functionality as Linked Resources-Future of Web of things-Real time web of things, Finding and Describing Smart Things, Sharing Smart Things-Discussing the future Web of things-Conclusion- Semantic Web-Semantic web services, Semantic web services processes and Lifecycle-Ontology-Ontology Engineering Methodologies, Application of Ontology Engineering in the Internet of Things, Ontology and the Organizational Perspective, Ontology and the I-T system Perspective, Ontology and the Data Perspective.

5.1 From the Internet of Things to the Web of Things

As more and more devices are getting connected to the Internet, the next logical step is to use the World Wide Web and its associated technologies as a platfor m for smart things.

Cool Town project, Kindberg et al. (2002) proposed to link physical objects with Web pages containing information and associated services. Using infrared interfaces or bar codes on objects, users could retrieve the URI of the associated page simply by interacting with the object.

Another way to use the Web for real-world objects is to incorporate smart things into a standardized Web service architecture (using standards, such as SOAP, WSDL(Web Services Description language), UDDI (Universal Description, Discovery, and Integration) a way to publish and discover information.

Instead of these heavyweight Web services often referred to as WS-* technologies, recent “Web of Things” projects have explored simple embedded Hypertext Transfer Protocol (HTTP) servers and Web 2.0 technologies.

So far, projects and initiatives, subsumed here under the umbrella term “Internet of Things”, have focused mainly on establishing connectivity in a variety of challenging and constrained networking environments. A promising next step is to build scalable interaction models on top of this basic network connectivity and thus focus on the application layer.

The services that smart things expose on the Web usually take the form of a structured XML document or a JavaScript Object Notation (JSON) object, which are directly machine-readable. These formats can be understood not only by machines, but are also reasonably accessible to people. With this smart things can not only communicate on the Web, but also provide a user-friendly representation of themselves. This makes it possible to interact with them via Web browsers and thus explore the world of smart things with its many relationships

Web Of Things definition:

Web of Things is all about making devices accessible over the web using web protocols like HTTP, Web Socket, JSON etc. agnostic to any thing below the application layer, so that any device can be part of the universal WOT regardless of what protocols it uses to connect to internet.

IoT	WoT
Devices can be connected with any form of internet	WoT is made to handle and use the potential of IoT
It deals with actuators, sensors, computation, communication Interfaces.	It deals with web servers and Protocols.
Digitally Augmented objects make IoT	WoT is made up of the applications that are made for Io Devices.
Every IoT devices have a different Protocol	A single protocol is used for multiple/various IoT devices.
Programing is difficult because of multiple protocols	Programming is easy so it doesn't have multiple protocols.
All the protocols and standard are private and it cannot be accessed publicly	WoT can be accessed freely by anyone, anytime.

Dynamically generated real-world data on smart objects can be displayed on such "representative" Web pages, and then processed with Web 2.0 tools. For example, things can be indexed like Web pages via their representations, users can "google" for them, and their URI can be emailed to friends or it can be bookmarked. The physical objects themselves can become active and publish blogs or inform each other using services, such as Twitter.

5.2 Designing RESTful Smart Things

The "Web of Thing was unheard of before. Now, the use of REST as a universal interaction architecture, that interacts with smart things can be built around universally supported methods and a set of guidelines to Web-enable smart things and illustrate them with concrete examples of implemented prototypes. It was the architecture of the Web that allowed data and services to be shared in a way that

5.2.1 Modeling Functionality as Linked Resources

The central idea of REST revolves around the notion of a resource as any component of an application that is worth being uniquely identified and linked to. On the Web, the identification of resources relies on Uniform Resource Identifiers (URIs), and representations retrieved through resource interactions contain links to other resources, so that applications can follow links through an interconnected web of resources. Clients of RESTful services are supposed to follow these links, just like one browses Web pages, in order to find resources to interact with. In the case of the Sun SPOT, each node has a few sensors (light, temperature, accelerometer, etc.), actuators (digital outputs, LEDs, etc.), and a number of internal components (radio, battery). Each of these components is modeled as a resource and assigned a URI. For instance, typing a URI such as

http://.../sunspots/spot1/sensors/light

in a browser requests a representation of resource light of the resource sensors of spot1

Resources are primarily structured hierarchically and each resource also provides links back to its parent and forward to its children.

As an example, the resource

http://.../sunspots/spot1/sensors/

provides a list of links to all the sensors offered by spot1.

This interlinking of resources that is established through both, resource links and hierarchical URI, is not strictly necessary, but well-designed URIs make it easier for developers to "understand" resource relationship and even allow non-link based "ad-hoc interactions". In a nutshell, the first step when Web-enabling a smart thing is to design its resource network, Identification of resources and their relationships are the two important aspects

5.5 Advanced Concepts: The Future Web of Things

Even though there are many web standards and design principles that are leveraged for smart things, there are many open challenges remain. Three such challenges discussed are

1. Needs for real-time data of many smart things applications.
2. Finding and understanding services available in a global Web of Things.
3. Mechanisms for sharing smart things.

5.5.1 Real-Time Web of Things

HTTP is a stateless client/server protocol, This interaction model is well-suited for control-oriented applications where clients read/write data from/to embedded devices. However, this client initiated interaction models seem inappropriate for bi-directional event-based and streaming systems, where data must be sent asynchronously to the clients as soon as it is produced. Many

pervasive scenarios must deal with real-time information to combine stored or streaming data from various sources to detect spatial or temporal patterns, as is the case in many environmental monitoring applications. As such applications are often event-based and embedded devices usually have a low-duty cycle (i.e., sleep most of the time), smart things should also be able to push data to clients (rather than being continuously polled). To support the complex, data centric queries required for such scenarios, more flexible data models are required to expose sensor data streams over the Web.

Recent developments in the real-time Web to build such a data model that is more suited to the data-centric, stream-based nature of sensor-driven applications. As mentioned before, using syndication protocols, such as Atom, improves the model when monitoring, since devices can publish data asynchronously using AtomPub on an intermediate server or Smart Gateway. Nevertheless, clients still have to pull data from Atom servers. Web streaming media protocols (RTP/RTSP) have enabled transmission of potentially infinite data objects, such as Internet radio stations. Sensor streams are similar to streaming media in this respect. However, streaming media mainly support play and pause commands, which is insufficient for sensor streams where more elaborate control commands are needed.

The Extensible Messaging and Presence Protocol (XMPP) is an open standard for real-time communication based on exchanges of XML messages, and powers a wide range of applications including instant messaging. Although widely used and successful, XMPP is a fairly complex standard, which is often too heavy for the limited resources of embedded devices used in sensor networks. This model, called Comet, enables a Web server to push data back to the browser without the client requesting it explicitly. Since browsers are not designed with server-sent events in mind, Web application developers have tried to work around several specification loopholes to implement Comet-like behavior, each with different benefits and drawbacks. One general idea is that a Web server does not terminate the TCP connection after response data has been served to a client, but leaves the connection open to send further events.

The recent developments in Web techniques have allowed to build efficient and scalable publish/subscribe systems, It is suggested that a Web-based pub/sub model could be used to connect sensor networks with applications. PubSubHubbub (PuSH) is a simple, open pub/sub protocol as an extension to Atom and RSS. Parties (servers) speaking the PuSH protocol can get near-instant notifications (via callbacks) when a feed they are interested in is updated. The following model can be used to enable Web-based stream processing applications where users can post queries using an HTTP request to one or more sensors. The HTTP request collects the light and temperature sensor readings twice per second only if the light sensor value is not over “200“ and the temperature reading is less than “19“. All the data samples corresponding to these queries are then pushed into a feed on the message broker, where users can subscribe using the PuSH

protocol. They will then receive the data from the stream pushed from the broker via callbacks.

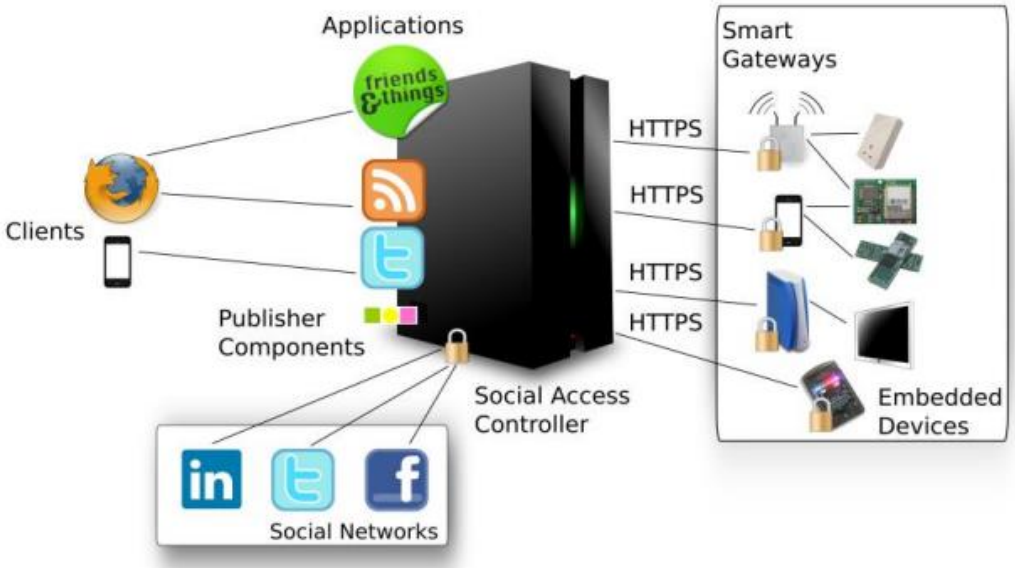
5.5.2 Finding and Describing Smart Things

- Another major challenge for a global Web of Things is searching and finding relevant devices among billions of smart things that will be connected to the Web.
- Finding them by browsing HTML pages with hyperlinks is literally impossible in this case, hence the idea of searching for smart things.
- Searching for things is significantly more complicated than searching for documents, as things are tightly bound to contextual information, such as location, are often moving from one context to another, and have no obvious easily indexable properties, such as human readable text in the case of documents.
- Beyond location, smart things need a mechanism to describe themselves and their services to be (automatically) discovered and used.
- But what is the best way to describe a thing on the Web so that both, humans and machines, can understand what services it provides?
- This problem is not inherent to smart things, but more generally a complex problem of describing services, which has always been an important challenge to be tackled in the Web research community, usually in the area of the Semantic Web.
- To overcome the limited descriptive power of resources on the Web, several languages have been proposed, such as RDF(Resource Description Format) or Microformats, designed for both, human and machines.
- Microformats provide a simple way to add semantics to Web resources.
- There is not one single Microformat, but rather a number of them, each one for a particular domain; a “geo” and “adr” microformat for describing places or an “hProduct” and “hReview” microformat for describing products.
- Each Microformat undergoes a “standardization” process that ensures its content to be widely understood and used, if accepted.
- Microformats are especially interesting in a Web of Things for two reasons; first they are directly embedded into Web pages and thus can be used to semantically annotate the HTML representation of a thing’s RESTful API.
- Secondly, Microformats (as well as RDFa) are increasingly supported by search engines, such as Google and Yahoo, where it is used to enhance the search results.
- For example, the “Geo” microformat could be used to localize search results close to you or, in our context, to to localize smart things in your direct vicinity.
- As an example, in Figure we use 5 microformats to describe a Sun SPOT and embed this semantic information directly in the HTML representation of the SPOT resources localize smart things in your direct vicinity.

Compound Microformats for Describing a Sun SPOT Using the Geo, hCard, hProduct and hReview Microformats

5.5.3 Sharing Smart Things

- The success of Web 2.0 mashups depends on the trend for Web 2.0 service providers (e.g., Google, Twitter, Wordpress, etc.) to provide access to some of their services through relatively simple, often RESTful, open APIs on the Web.
- Mashup developers often share their mashups on the Web and expose them through open APIs as well, making the service ecosystem grow with each application and mashup.
- The following Figure shows the simplified component architecture of a Social Access Controller (SAC), which serves as authentication proxy between clients and smart things.
- However, enabling such an open model for a Web of Things requires a sharing mechanism for physical things supporting access control to the RESTful services provided by devices.
- For example, one could share the energy consumption sensors in one's house with the community.
- However, this is a potentially risky process, given that these devices are part of our everyday life and their public sharing might result in serious privacy implications.



- HTTP already provides authentication mechanisms based on credentials and server-managed user groups. While this solution is already available for free on most Web servers, it still presents a number of drawbacks in the WoT context.
- First, for a large number of smart things it becomes quite unmanageable to share credentials for each of them.
- Then, as the shared resources are not advertised anywhere, sharing also requires the use of secondary channels, such as sending emails containing credentials to people.
- Several platforms, such as SenseWeb or Pachube propose to overcome these limitations by providing a central platform for people to share their sensor data.
- However, these approaches are based on a centralized data repository and are not designed to support decentralization and direct interaction with smart things.
- A promising solution is to leverage existing social structures of social networks (e.g., Facebook, Linkedin, Twitter, etc.) and their (open) APIs to share things.
- Using social networks enables users to share things with people they know and trust (e.g., relatives, friends, colleagues, fellow researchers, etc.), without the need to recreate yet another social network or user database from scratch on a new online service.
- Additionally, this enables advertising and sharing through a unique channel: you can use various well-known social networks to inform your friends about the sensors you shared with them by automatically posting messages to their profile or newsfeed.
- The SAC platform is an implementation of this idea. SAC is an authentication proxy between clients (e.g., Web browsers) and smart things.