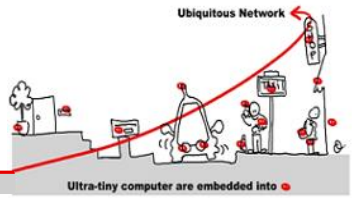
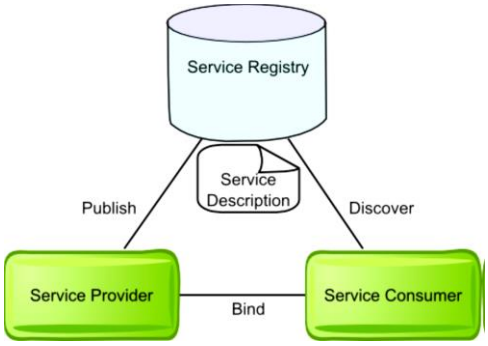



# RESTful protocol and CoAP

## 1 Introduction to Service-oriented Middleware and CoAP




(*) A Perspective on the Future of Middleware-based Software Engineering, Valérie Issarny, Mauro Caporuscio, Nikolaos Georgantas, Workshop on the Future of Software Engineering : FOSE 2007, 2007, Minneapolis, United States. pp.244-258, 2007, https://hal.inria.fr/inria-00415919

**Service-oriented Middleware*** is a kind of middleware based on the Service Oriented Architecture (SOA) paradigm that supports the development of distributed=software systems in terms of loosely coupled networked services. In SOA, networked resources are made available as autonomous software services that can be accessed without knowledge of their underlying technologies. Key feature of SOA is that services are independent entities, with well-defined interfaces, which can be invoked in a standard way, without requiring the client to have knowledge about how the service actually performs its tasks.

The SOA style (see Figure 2) is structured around three key architectural components: (i) service provider, (ii) service consumer, and (iii) service registry. In SOA-based environments, the Service-Oriented Middleware (SOM) is in charge of enabling the deployment of services and coordination among the three key conceptual elements that characterize the SOA style.

Popularity of service oriented computing is mainly due to its **Web Service** instantiation.

### 1.1 RESTful protocol

REST stands for REpresentational State Transfer. REST is web standards based architecture and uses HTTP Protocol for data communication. It revolves around resource where every component is a resource and a resource is accessed by a common interface using HTTP standard methods. REST was first introduced by Roy Fielding in 2000.

In REST architecture, a REST Server simply provides access to resources and REST client accesses and presents the resources. Here each resource is identified by URIs/ global IDs. REST uses various representations to represent a resource like text, JSON and XML. Now a days JSON is the most popular format being used in web services.

Even though REST is heavily influenced by the Web-Technology, in theory the REST architecture style is not bound to HTTP. However, HTTP is the only relevant instance of the REST. For this reason this article describes REST implemented by using HTTP. Often this is called RESTful HTTP.

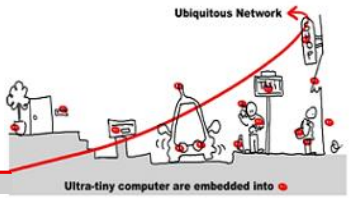
The idea behind RESTful HTTP is to use the existing features and capabilities of the WEB. REST does not invent new technologies, components or services. RESTful HTTP defines the principles and constrains to use the existing WEB-Standards in a better way.

**Exercise 1 :**

For the next exercise, you must choice the software environment you are using :

- Your virtual machine with linux
- Your native machine with Windows and Cygwin to enable shell script programming

To install Cygwin on Windows you can follow these steps : http://cygwin.com/install.html




# RESTful protocol and CoAP

In any case you must also install cURL tool from http://curl.haxx.se/download.html, whatever your software environment.

curl is a tool to transfer data from or to a server, using one of the supported protocols (DICT, FILE, FTP, FTPS, GOPHER, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, POP3, POP3S, RTMP, RTSP, SCP, SFTP, SMB, SMBS, SMTP, SMTPS, TELNET and TFTP). The command is designed to work without user interaction.

You can test it simply with:

```
curl http://yilmazhuseyin.com/blog/dev/
```

You can find more information about cURL in the man pages and in the tutorial (http://curl.haxx.se/docs/httpscripting.html)

**Exercise 2 :** In this exercise we're going to access to RESTFul services on http://openweathermap.org/.

All the documentation about the corresponding RESTFul API is on http://openweathermap.org/api.

- Write a shell script using curl to get temperature (Celsius), pressure (hPa), wind direction and speed, humidity and the weather for a given city.
- The format of the data is JSON or XML?
- How can you get data in XML format (see http://openweathermap.org/current)?
- Because I like to sky, how can I know if it's snowing at Courchevel?

## 1.2 CoAP (Constrained Application Protocol)

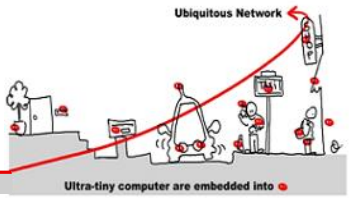
The Constrained Application Protocol (CoAP) is a RESTful web transfer protocol for resource-constrained networks and nodes. C oAP is designed to easily translate to HTTP for simplified integration with the web, while also meeting specialized requirements such as multicast support, very low overhead, and simplicity. The CoRE group has proposed the following features for CoAP: RESTful protocol design minimizing the complexity of mapping with HTTP, Low header overhead and parsing complexity, URI and content-type support, Support for the discovery of resources provided by known CoAP services.

### 1.2.1 CoAP Firefox user agent

In order to test the server from a web browser with CoAP URL, you must use firefox with an additional module called Copper (https://addons.mozilla.org/fr/firefox/addon/copper-270430/). Copper is CoAP user-agent for Firefox installs a handler for the 'coap' URI scheme and allows users to browse and interact with Internet of Things devices.

This user-agent will be very useful to test our CoAP server.

**Exercise 3 :** ETSI developed IoT CoAP Plugtests (http://www.etsi.org/plugtests/coap/coap.htm) with CoAP server at **COAP://COAP.ME** to test your CoAP clients. Use firefox to explore this CoAP service. On the top of the copper page you can visualize the header of the returned message after you sent a GET message. Verify the conformance of all the field in the header according to the CoAP RFC7252.




# RESTful protocol and CoAP

### 1.2.2 CoAP.NET library

Numerous software libraries are available to implement CoAP Services (see http://coap.technology/impls.html). We choose CoAP.NET implementation in C# .NET to develop CoAP based services using Visual Studio on Windows.

CoAP.NET 1.1.0 is a nugget package (https://www.nuget.org/packages/CoAP/), thus you can install the library for you project in Visual Studio using the package manager console (see http://docs.nuget.org/consume/package-manager-console to proceed).

The most detailed documentation on the library can be found on http://open.smeshlink.com/CoAP.NET/.

**Exercise 4 :**
The "CoAP Simple Service.zip" file available on the course web site, show you a simple development of a CoAP client and server that exchange "Hello World" message.
Modify this code to:

- Test your client to connect a remote CoAP server (with another URL than localhost)
- Create a second resource that manage a temperature value (integer) that we can change and read from a CoAP server
-

## 2 Advanced Tutorial

## 2.1 CoAP / HTTP gateway

**Exercise 5 :** Using Windows Communication Foundation (WCF) and CoAP.Net, write a Gateway that share a resource between a CoAP Service and a HTTP REST Service.

## 2.2 MQTT and CoAP interoperability

**Exercise 6 :** Propose a software architecture to connect a CoAP service on a MQTT broker to publish (PUT) and observe (GET) from a CoAP Client. Test it to read and write a simple temperature sensor value in the MQTT base.
C