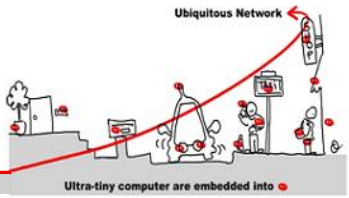
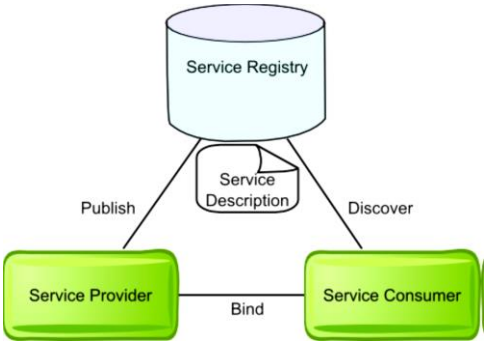
RESTful protocol and CoAP



1 Introduction to Service-oriented Middleware and CoAP

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.



(*) 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

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 and Web of Things

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.

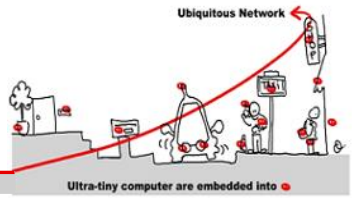
The idea behind RESTful HTTP is to use the existing features and capabilities of the WEB. REST does not invent new technologies, components or services. RESTful HTTP defines the principles and constrains to use the existing WEB-Standards in a better way.

1.1.1 Online commands to access to RESTful HTTP services

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



RESTful protocol and CoAP

To install Cygwin on Windows you can follow these steps : http://cygwin.com/install.html
In any case you must also install «curl» tool from http://curl.haxx.se/download.html, whatever your software environment.
In the case of linux you can use "sudo apt-get install curl" command.

«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://curl.haxx.se/docs/httpscripting.html
```

You can find more information about "curl" in these man pages and tutorials.

a. 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. More details and example are available on http://openweathermap.org/current. To use these services you must first sign up to get your own key on http://openweathermap.org/price. Your key is then the value of the "appid" parameter in the requests to the web services (ex. **http://api.openweathermap.org/data/2.5/weather?id=2172797&appid=<YOUR KEY HERE>**).

- The format of the data is it JSON or XML?
- Because I like to sky, how can I know if it's snowing at Courchevel?

Exercise 2 : "Jq" is a lightweight and flexible command-line JSON processor. It's available for all OS like linux and windows on https://stedolan.github.io/jq/download/. In this last case you can use it as a shell command using Cygwin. Don't forget to copy jq.exe in the bin directory of Cygwin. In the case of linux you can use "sudo apt-get install jq" command.

- Write some shell scripts using "curl" and "jq" that get temperature (Celsius), pressure (hPa), wind direction and speed, humidity and the weather for a given city. Basic filters that can be defined with jp are like these (https://stedolan.github.io/jq/manual/#Basicfilters).

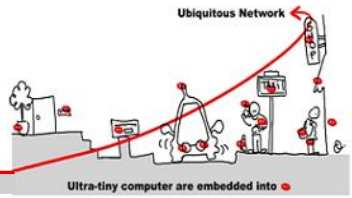
Be careful:

- If you need to add parameters in your URL, you must split url and the parameter in a «curl» call. Parameter must appear after –d option and before the url..

1.1.2 With real sensors

Thanks to "openhab" gateway (http://www.openhab.org/), a lot of under IP IoT protocols can be exported as MQTT publishers (sensors) and subscribers (actuators) but also RESTFul Web Services.

If you're interested in how to install it you can look at the website of the project. Here we are going to use it to manage some Zwave ,NOcean sensors, etc. They are popular standards for connected devices in smart homes. For this tutorial, one openhab server will be installed in the tutorial network with a set of Zwave devices (some details will be given by the teacher during the tutorial).



RESTful protocol and CoAP

The Web server OpenHab is reachable at the address 192.168.1.187:8080. Documentation is provided at http://192.168.1.187:8080/doc/index.html.

The sensors and actuators are reachable at the addresses http://192.168.1.187:8080/items/<name>, where <name> is the name of the device like for example "zwave_device_xxx".

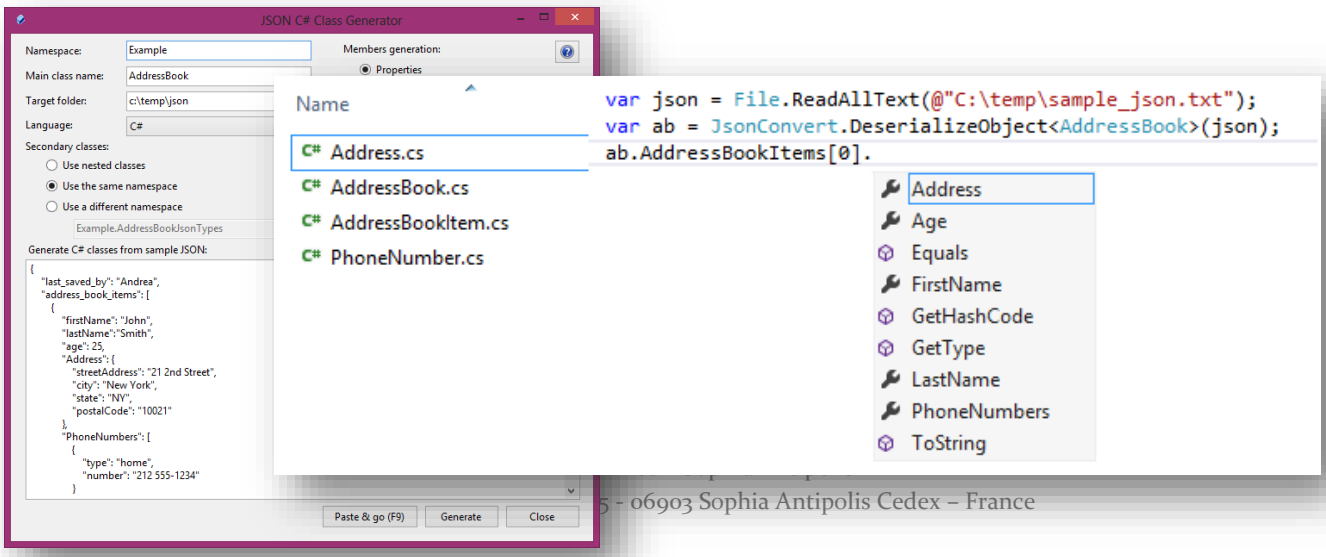
Exercise 3 : Test your access to the Zwave sensors and actuators through the provided web services, using online commands. What is the protocol to exchange data?

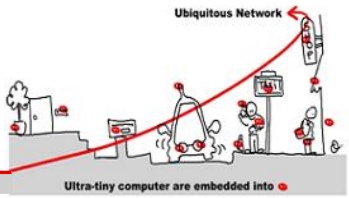
1.1.3 Access to RESTFul HTTP service using C#

First of all you must implement a simple HTTP/GET client based on the .Net class HttpWebRequest, which returns a JSON /text string in our cases.

```
// Returns JSON string
string GET(string url)
{
    HttpWebRequest request = (HttpWebRequest)WebRequest.Create(url);
    try {
        WebResponse response = request.GetResponse();
        using (Stream responseStream = response.GetResponseStream()) {
            StreamReader reader = new StreamReader(responseStream, Encoding.UTF8);
            return reader.ReadToEnd();
        }
    }
    catch (WebException ex) {
        WebResponse errorResponse = ex.Response;
        using (Stream responseStream = errorResponse.GetResponseStream())
        {
            StreamReader reader = new StreamReader(responseStream,
Encoding.GetEncoding("utf-8"));
            String errorText = reader.ReadToEnd();
            // log errorText
        }
        throw;
    }
}
```

Secondly, the output JSON/text must be deserialized into a C# object. To do that we can use "JSON C# Class Generator" from http://jsonclassgenerator.codeplex.com/ that generates C# classes from a sample JSON text, so you can use strongly typed programming with JSON.





RESTful protocol and CoAP

Exercise 4 : Write a program that give the weather and the temperature of a city.

1.1.4 With real sensors

Exercise 5 : Write a simple C# program to count the number of window openings thanks to a Zwave sensor to switch on the light connected to a Zwave remote power plug. If you want you can test other conditions with presence sensor detection, etc.

Be careful:

- In the case of the remote electrical plug you must send a POST / HTTP commands with data for the command on the body of the message. To do that, find an example on the web on how to use HttpWebRequest to send a POST.

1.2 Advanced Tutorial: CoAP (Constrained Application Protocol)

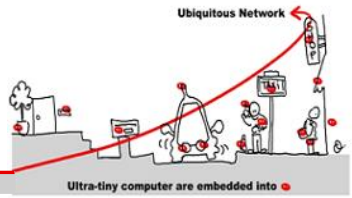
The Constrained Application Protocol (CoAP) is a RESTful web transfer protocol for resource-constrained networks and nodes. CoAP is designed to easily translate to HTTP for simplified integration with the web, while also meeting specialized requirements such as multicast support, very low overhead, and 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.

Exercise 6 : Find an example of a sequence of CoAP messages here :

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 |0 |  1  |    GET=1    |        MID=0x7d34         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  11   |  11   |    "temperature" (11 B) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 |2 |  0  |   2.05=69   |        MID=0x7d34         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     "22.3 C" (6 B) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Explain what is the purpose of each field and its value.



RESTful protocol and CoAP

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 7 : 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.

1.2.2 CoAP.NET library

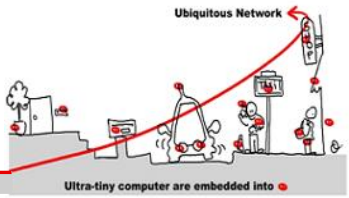
Numerous software libraries are available to implement CoAP Services (see http://coap.technology/impls.html). We choose CoAP.NET implementation in C# .NET to develop CoAP based services using Visual Studio on Windows.

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 8 :
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



RESTful protocol and CoAP

2 Advanced Tutorial

2.1 CoAP / HTTP gateway

Exercise 9 : 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 10 : 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.