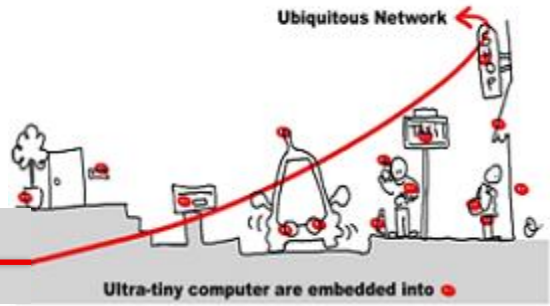
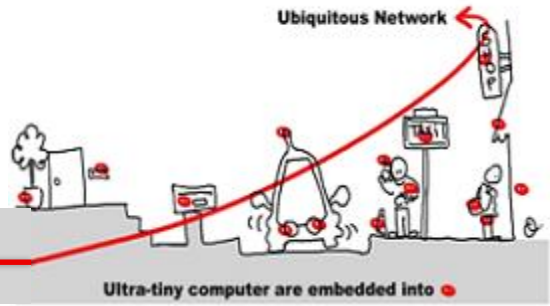


Services et Web Services, modèles et implémentations

Jean-Yves Tigli
http://www.tigli.fr
Polytech of Nice - Sophia Antipolis University

Email : tigli@polytech.unice.fr

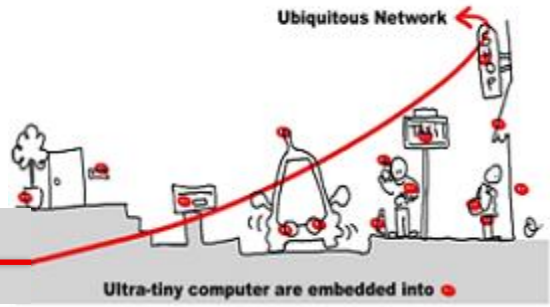


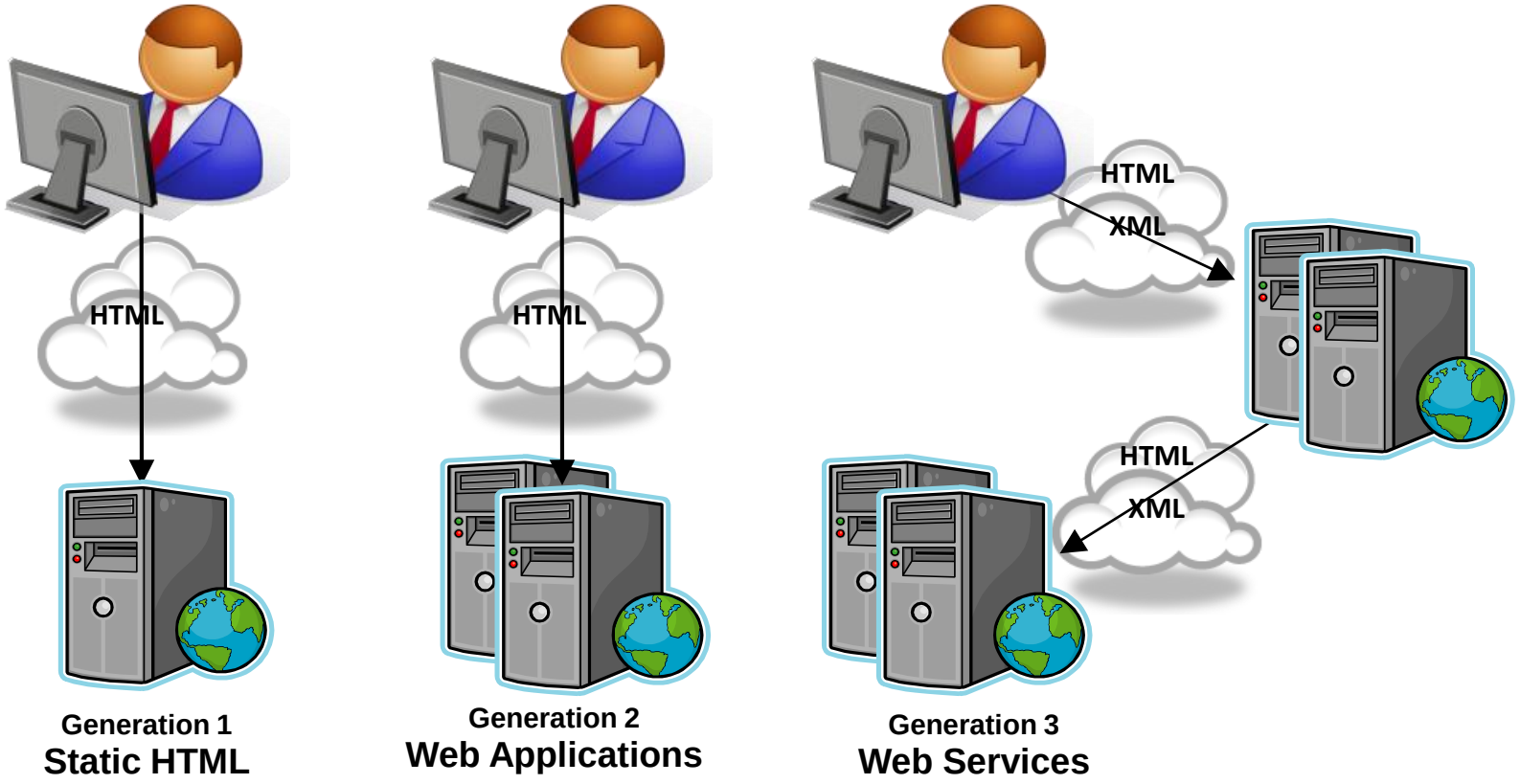
Introduction aux Services Web RESTFul

Chapitre 5 de «Fielding, Roy Thomas. Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation, University of California, Irvine, 2000.»

http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

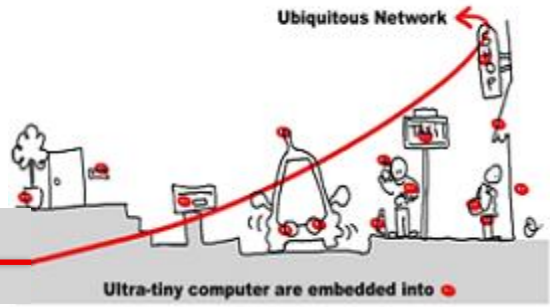
Rappel : du Web H2M au M2M



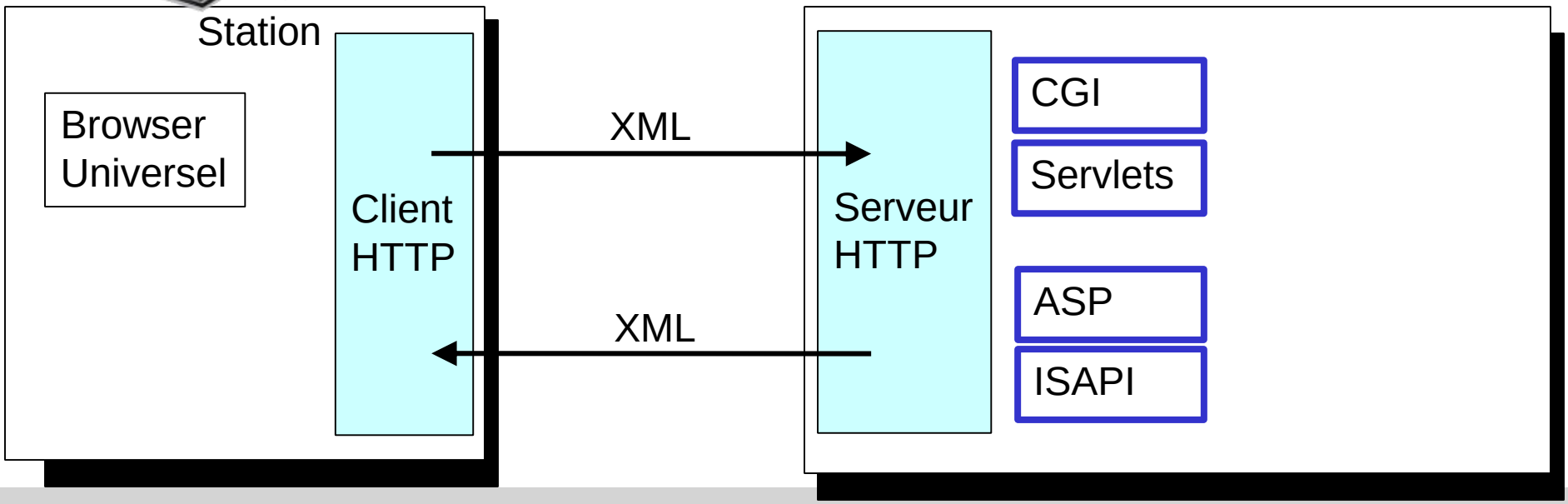


Machine to Machine (M2M)

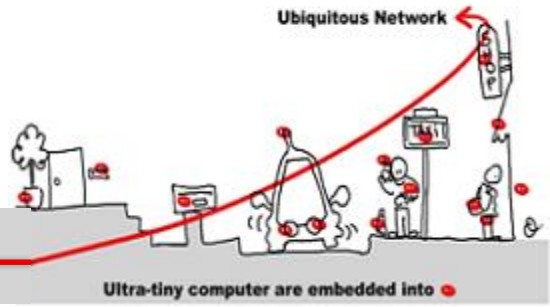
Rappel : Evolution du Web H2M au M2M



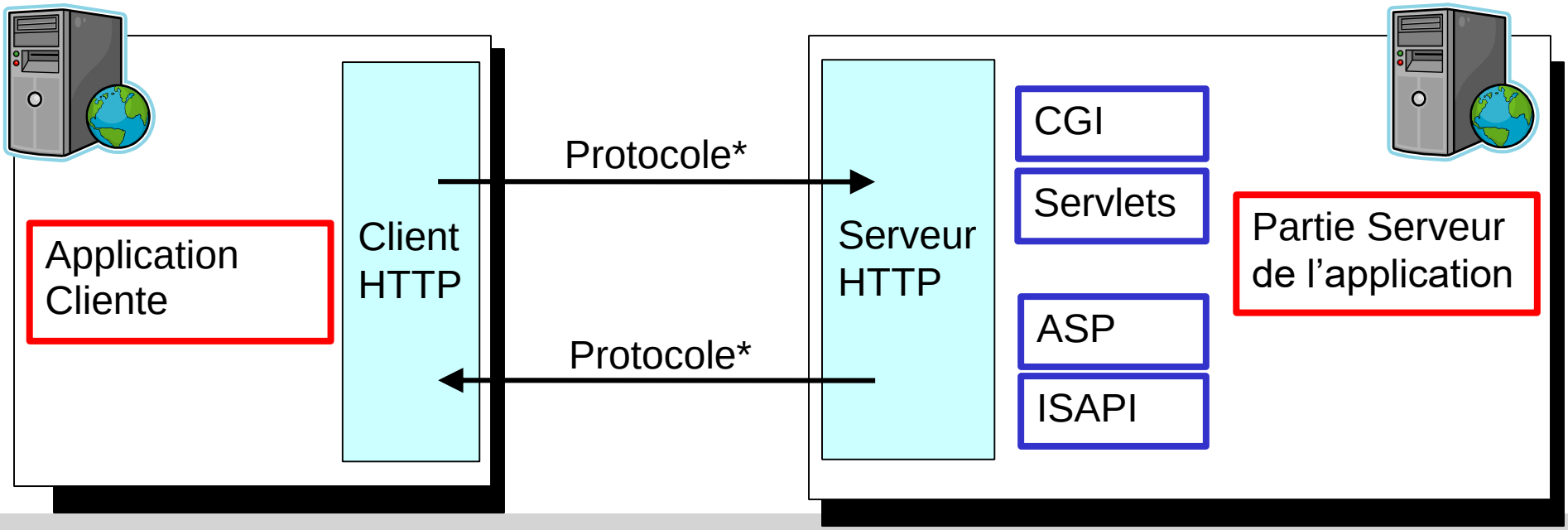
- Exemple HTTP + XML



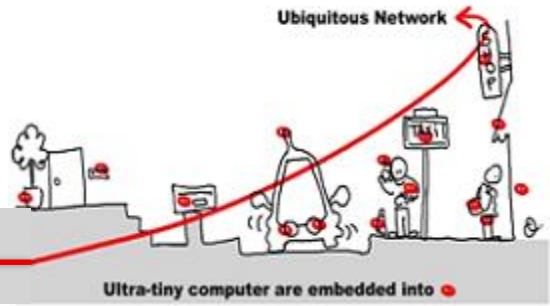
Web M2M : Middleware orienté Service Web



- Gérer l'interopérabilité avec HTTP (le WEB)
- Choisir un protocole de communication client/serveur over HTTP

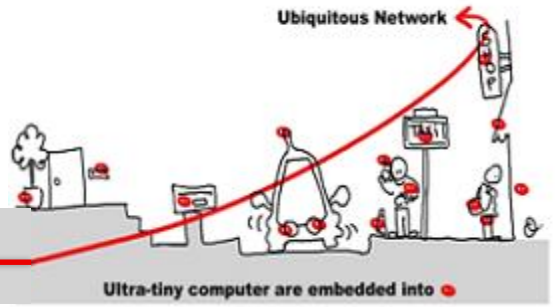


Caractéristiques des intergiciels orientés Service



- Caractéristiques
  - Architecture de type Client / Serveur
    - Le serveur rend des services à un client
  - Réutilisable
    - Par plusieurs clients (simultanément ou pas)
  - Indépendamment de
    - la plate-forme (UNIX, Windows, …)
    - du langage pour l'implémentation (VB, C#, Java, …)
    - la plate-forme de développement sous-jacente (.NET, J2EE, Axis…)

Quels objectifs pour les Services Web ?

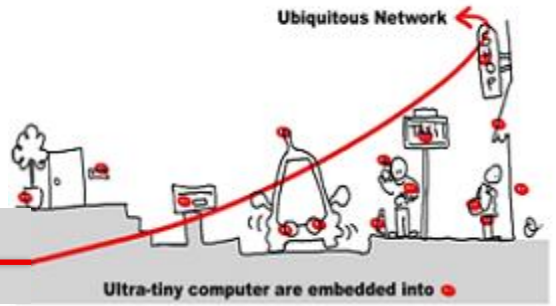


- Remplacer les outils (RPC, DCOM, CORBA, RMI) par une approche entièrement ouverte et interopérable, basée sur la généralisation des serveurs Web avec scripts CGI.

- Faire interagir des composants hétérogènes, distants, et indépendants avec un protocole standard (ex. SOAP).

- Passer les politiques de sécurité grâce en grande partie à une couche session basée sur HTTP (port 80).

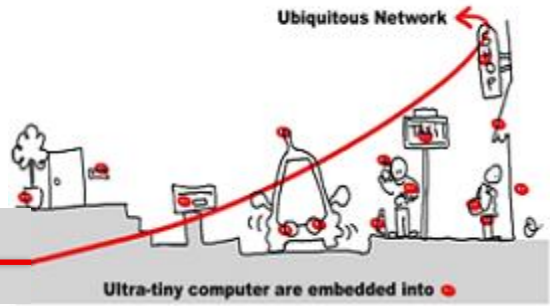
Services Web et Interopérabilité



- Plateform independent : A l'instar de Corba (précurseur historique), les Services Web gèrent l'interopérabilité au niveau du protocole d'échange.

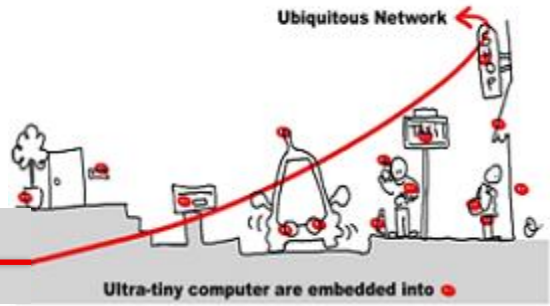
- Plateform dependent: d'autres approches gèrent l'interopérabilité par le portage de la plate-forme d'exécution (ex. OSGi, RMI sur Java et historiquement COM/DCOM)

Pour quoi faire ?



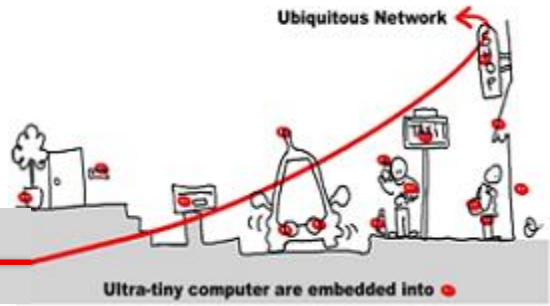
- Les Services Web permettent d'interconnecter :
  - Différentes entreprises
  - Différentes applications
  - Différents clients
  - Différents matériels

- Utilisé dans différents cadres:
  - B2B (Business To Business)
  - EAI (Enterprise Application Integration)
  - …



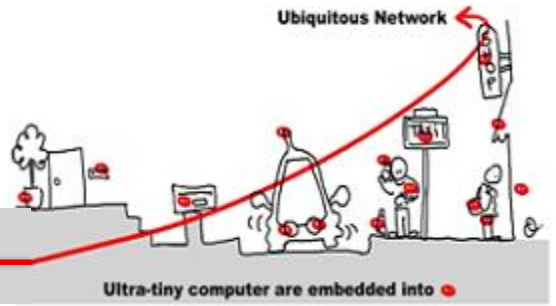
Architectures ROA - SOA

Architecture orientées ressources ou données



- Le Web aujourd'hui : extraction des ressources / ressources
- Les ressources sont identifiées par des URL
- Dénominations
  - Architectures Orientées Données (DOA)
  - Architectures Orientées Ressources (ROA)

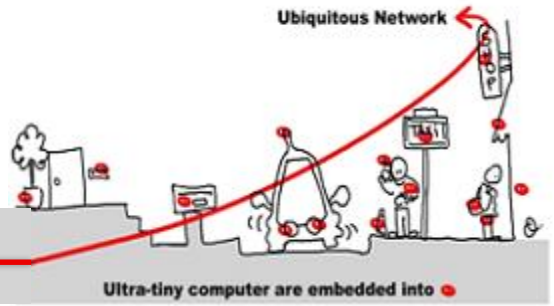
Les principes REST



- **REpresentational State Transfer**
- Style architectural pas seulement dédié aux architectures orientées services et aux communication entre machines.
- Aucune hypothèse sur les protocoles impliqués, seulement des contraintes
- Les systèmes qui suivent les principes de l'architecture REST sont souvent appelés *RESTful* et s'appuient sur le Web

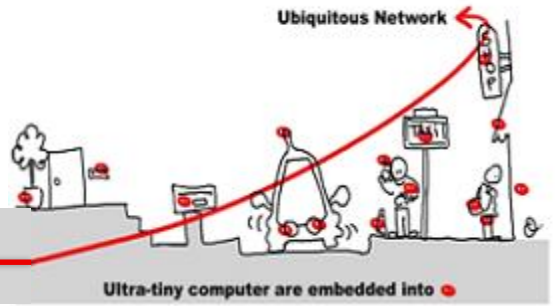
Chapitre 5 de la thèse de doctorat "Representational State Transfer (REST)". de Roy Fielding , 2000

Les principes REST ou ROA



- Ressources (Identifiant)
  - Entité identifiable dans le système (livre, agenda …)
  - URI et donc possiblement URL
  - Une URI identifie une seule Ressource
  - Une Ressource peut avoir plusieurs URI
  - Exemple :
    - Emploi du temps de tigli : /edt/prof/tigli/lundi
- Méthodes (Verbes)
  - Quatre opérations de base « CRUD » : Create (créer), Retrieve (lire), Update (mettre à jour), Delete (Supprimer)
  - Exemple méthodes HTTP : GET, POST, PUT, DELETE
  - Déjà adaptées à la manipulation de Ressources
- Représentation (Vue de l'état)
  - Informations transférées entre client et serveur
  - Exemple : XML, JSON, XHTML, CSV ….

Exemple RESTFul/XML



- Exemple de message HTTP RESTFul

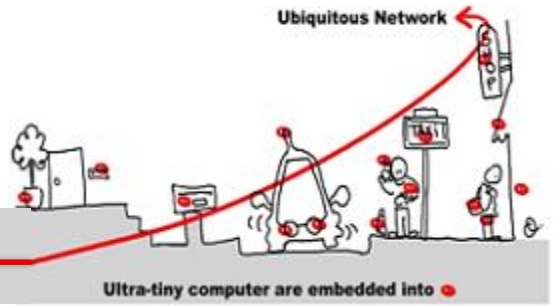
```
POST http://MyService/Person/
Host: MyService
Content-Type: text/xml; charset=utf-8
Content-Length: 123
<?xml version="1.0" encoding="utf-8"?>
```

HTTP Header
Commande POST

```
<Person>
 <ID>1</ID>
 <Name>M Vaqqas</Name>
 <Email>m.vaqqas@gmail.com</Email>
 <Country>India</Country>
</Person>
```

HTTP Body
XML representation
of a resource « Person »

Exemple RESTFul/JSON



- Exemple de message HTTP RESTFul

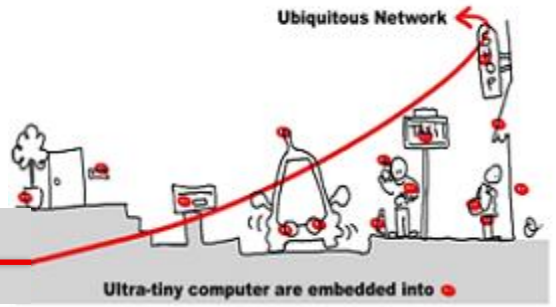
```
POST http://MyService/Person/
Host: MyService
Content-Type: text/xml; charset=utf-8
Content-Length: 123
<?xml version="1.0" encoding="utf-8"?>
```

HTTP Header
Commande POST

```
{
    "ID": "1",
    "Name": "M Vaqqas",
    "Email": "m.vaqqas@gmail.com",
    "Country": "India"
}
```

HTTP Body
JSON representation
of a resource

REST et invocation de méthode

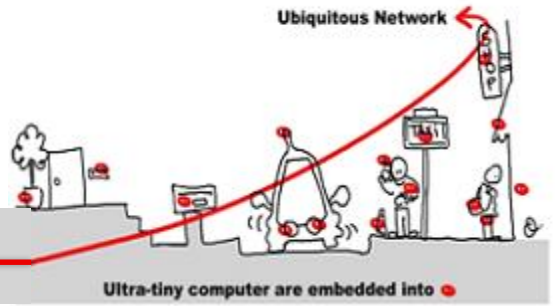


- Chaque demande REST contient une URL, de sorte que le serveur sait quelle ressource vous souhaitez accéder, mais il peut aussi contenir une méthode.
- Une méthode décrit alors quoi faire avec cette ressource.

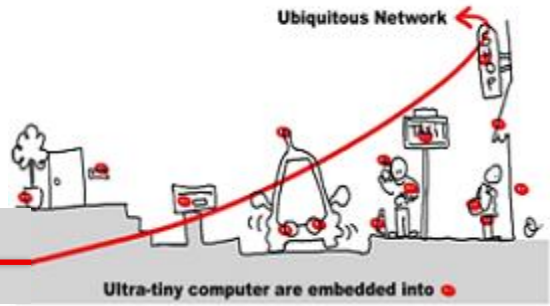
- Mais ce concept «méthode» n'est pas utilisé très souvent car en marger d'une approche ROA

- Habituellement, on utilise une URL comme un lien vers des données récupérées via la méthode GET, et modifiées (délétions, insertions, mises à jour) via la méthode POST

Cycle de Vie REST



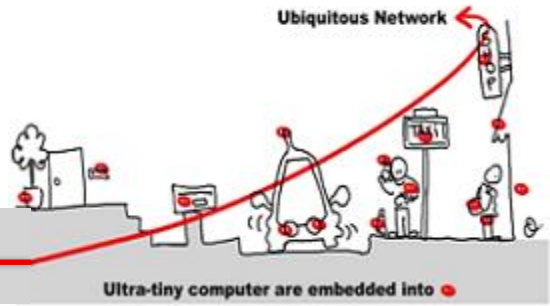
- Plus récemment REST s'est doté d'un langage de description de service : WADL (Web Application Description Language)
- Soumis en 2009 au W3C, il n'est toujours pas standardisé
- Le WADL est un format de fichier basé sur XML qui permet de décrire des applications REST.
- Cette spécification se heurte néanmoins à **la spécification WSDL 2.0, qui elle aussi permet la description de web services REST**.
- De plus, WADL est encore très mal supporté par l'ensemble des frameworks existants ce qui limite son utilisation.



Windows Communication Fundation

Quelques éléments sur la mise en œuvre de REST sous WCF

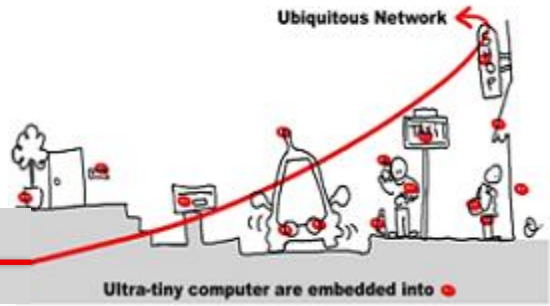
Application à WCF .Net



- Projet WCF / Bibliothèque de Services WCF
- Trois fichiers pour un Web Service REST
  - Configuration : App.Config
  - Interface : Iservice.cs & Contrat : Annotations dans le fichier
  - Implémentation : Service.cs

- TD Semaine 2

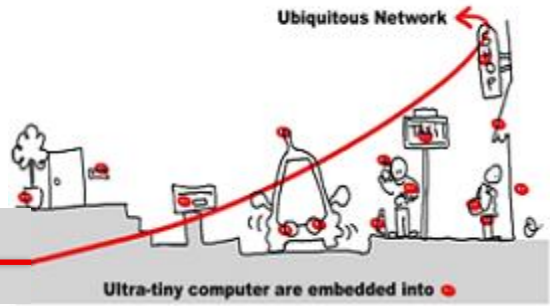
Interface et Contrat de Données



- Contrat de Données
- Utilise directement les commandes REST-CRUD (GET et POST)

```
namespace RESTFulWCFService
{
    [DataContract]
    public class OrderContract
    {
        [DataMember]
        public string OrderID { get; set; }

        [DataMember]
        public string OrderDate { get; set; }

    }
}
```

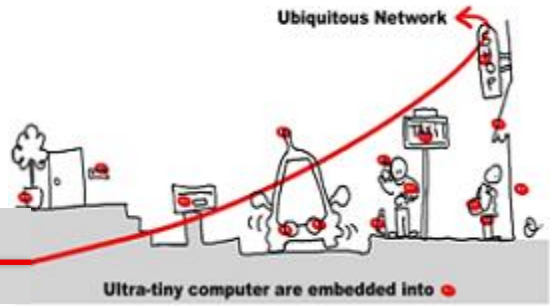
Interface et Contrat d'Opération



- WebGet (GET) et WebInvoke (POST PUT DELETE) permettent de spécifier l'utilisation des commandes GET et POST
- UriTemplate: Uri de l'API
- RequestFormat & ResponseFormat : WebMessageFormat XML ou JSON.

```
[ServiceContract]

public interface IService {

    [OperationContract] [WebGet]

    string EchoWithGet(string s);

    [OperationContract] [WebInvoke]

    string EchoWithPost(string s);
}
```

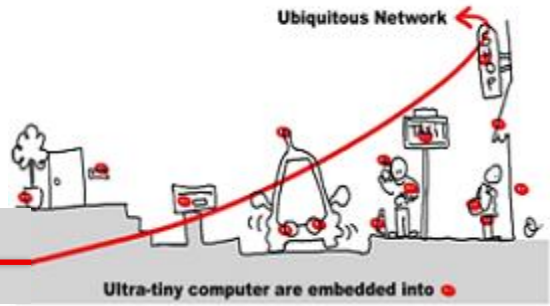
Implémentation du service



- Les annotations sur WebGet et WebInvoke peuvent être rajoutées ici aussi

```
public class Service : IService {

        public string EchoWithGet(string s)
        {
                return "You said " + s;
        }

        public string EchoWithPost(string s)
        {
                return "You said " + s;
        }

}
```

App.config



- Un grand nombre de paramètres de configuration sont aussi présents dans le fichier app.config
- Pour un WS de type REST, il faudra en fixer quelques uns tels que binding=« webHttpBinding »

- Voir TD 2.b semaine 2