

Services

Modèle ABC et WCF

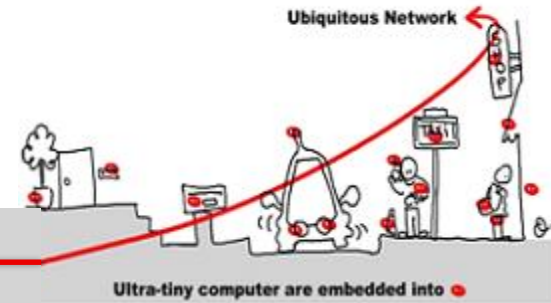
Jean-Yves Tigli

<http://www.tigli.fr>

Polytech of Nice - Sophia Antipolis University

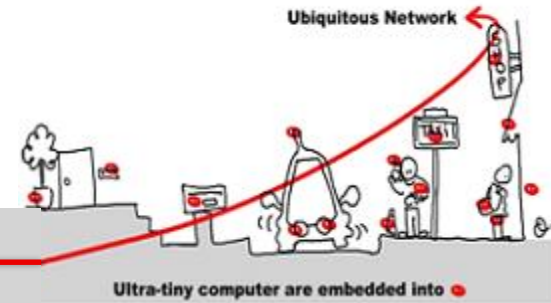
[Email : tigli@polytech.unice.fr](mailto:tigli@polytech.unice.fr)

Introduction à Windows Communication Foundation



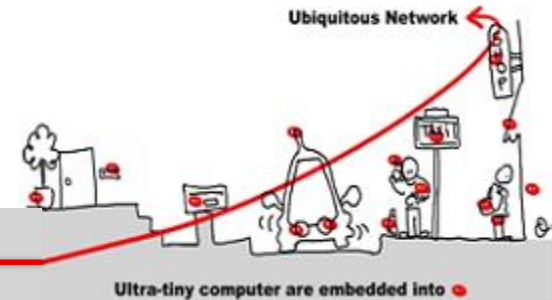
- WCF est un framework unifié pour les applications distribuées, avec son modèle de programmation et son API
- C'est le standard pour les applications distribuées Microsoft .Net
- Les caractéristiques de WCF sont :
 - Un modèle de programmation orienté service
 - Une interopérabilité avec des services non WCF grâce à l'inclusion des standards du domaine
 - Une extensibilité qui permet à un client / serveur d'être configuré pour interopérer avec REST, SOAP, JSON, XML, et autres standards binaires.

Introduction à Windows Communication Foundation

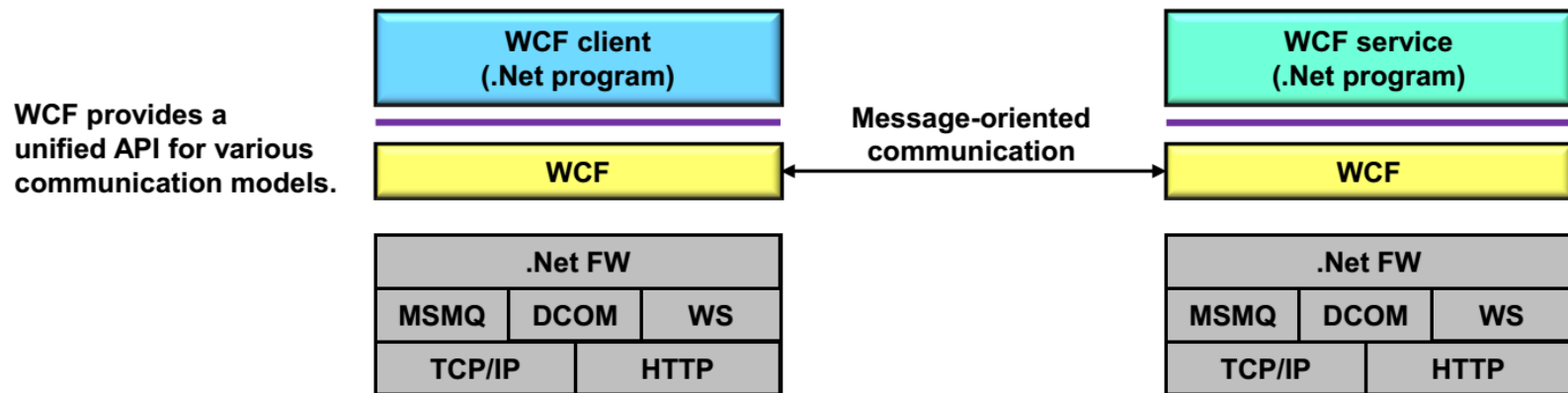


- Modèle de programmation unifiée, orienté service
- Remplace et étend :
 - .NET Remoting
 - DCOM
 - ASP.NET Web services
 - MSMQ (Queued Messaging)
 - .NET Enterprise Services
- Neutralité dans les protocoles, flexibilité, extensibilité

Introduction à Windows Communication Foundation

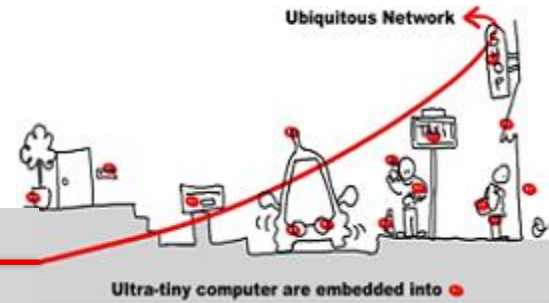


- WCF utilise de nombreux types de communication
- WCF est au sommet du framework .Net

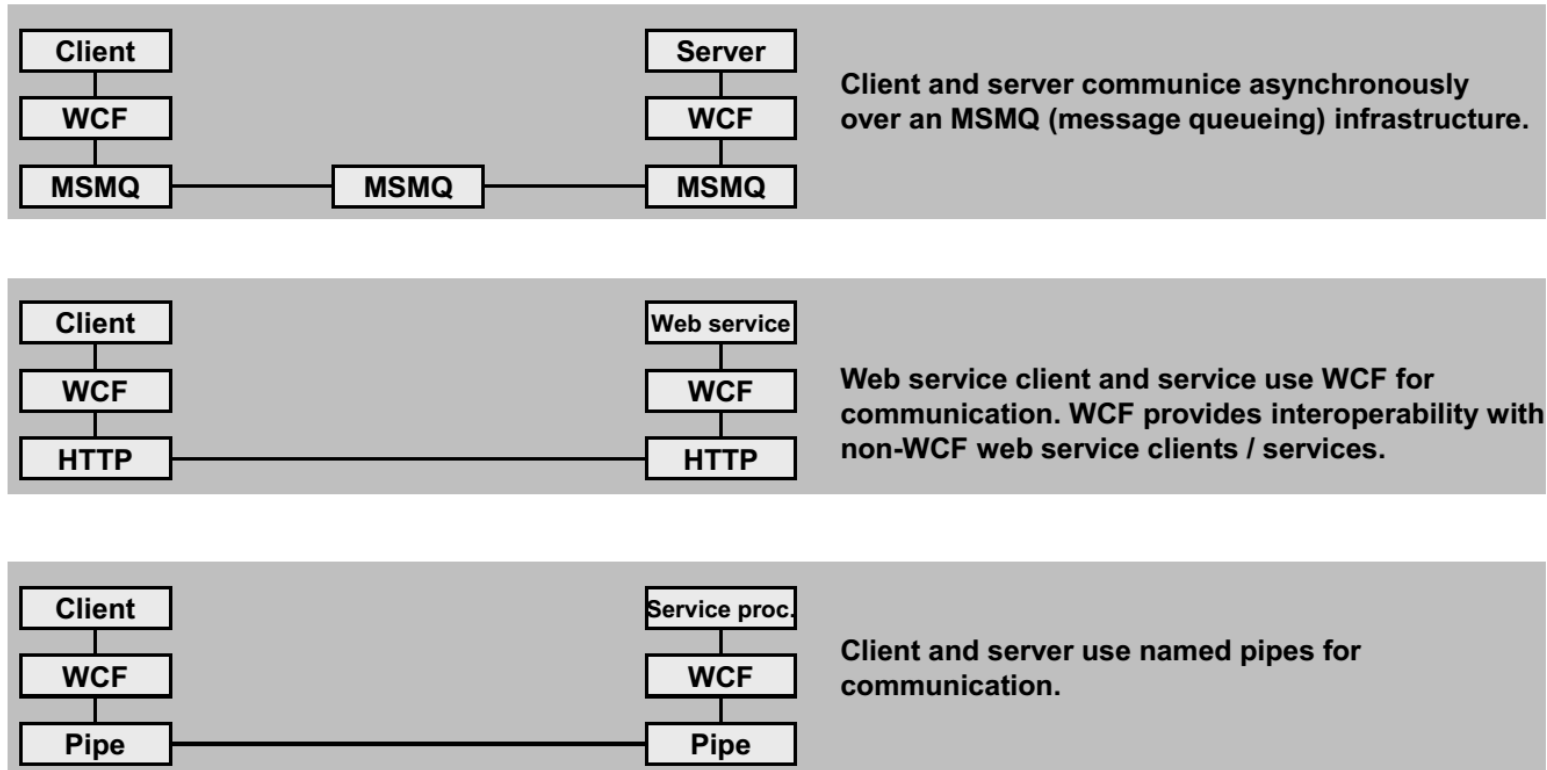


- En fait WCF est une approche orientée service qui va au-delà des Web Service
- En séparant les notions de contrat, de protocole et de format de message
- En séparant une description abstraite d'un service et son implémentation

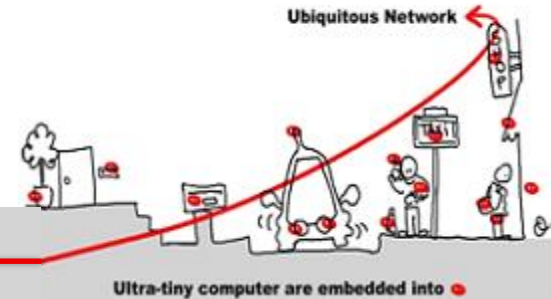
Introduction à Windows Communication Foundation



- WCF peut donc s'appuyer sur de multiples protocoles de communication



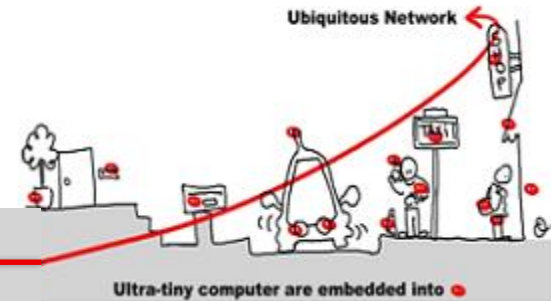
Modèle ABC de Service



- Le concept central de WCF est la notion de service
- Qui est fourni par un « endpoint »
- Qui est accessible par une protocole de transport
- Ainsi un service WCF est défini à travers les informations ABC :
 - L'adresse A où le service est disponible (l'URI du endpoint dans le cas d'un WS)
 - Le binding B ou comment accéder au service (quel protocole de transport ? Avec quel format ?)
 - Le contrat C ou que fournit l'interface du service (opérations, types de données ...)
- Exemple dans le cas de WSDL

WCF term	Question	WSDL element
A (Address)	<i>Where</i>	<code><service></code> including element <code><endpoint></code>
B (Binding)	<i>How</i>	<code><binding></code>
C (Contract)	<i>What</i>	<code><types></code> <code><interface></code>

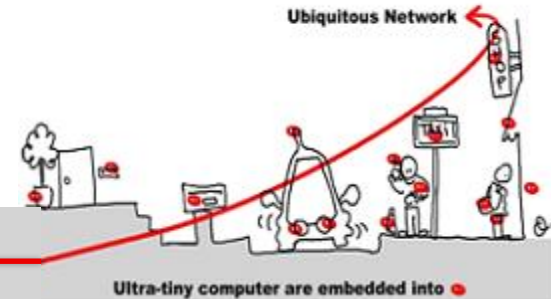
Le modèle de programmation WCF (Classes et Interfaces)



- Les éléments ABC de WCF sont disponibles comme des classes spécifiques

WCF term	Corresponding class library (namespace)
A (Address)	<code>System.Uri</code>
B (Binding)	<code>System.ServiceModel</code> E.g. <code>BasicHttpBinding</code> (SOAP, non-secure, interoperable, non-duplex) <code>WebHttpBinding</code> (REST-style binding, i.e. non-SOAP)
C (Contract)	Interfaces / classes annotated with <code>System.ServiceModel</code> attributes: <code>[OperationContract]</code> <code>[ServiceContract]</code> <code>[MessageContract]</code> Data contract (definitions of types used in operation contracts): <code>[DataContract]</code> (<code>System.Runtime.Serialization</code>)
E (Endpoint)	<code>System.ServiceModel.ServiceEndpoint</code>

Le « A » de ABC



- L'adresse WCF définit où le service est disponible (endpoint reference, à l'instar du standard WS-Addressing)

WS-Addressing EPR schema:

<pre><wsa:EndpointReference> <wsa:Address>xs:anyURI</wsa:Address> <wsa:ReferenceProperties>... </wsa:ReferenceProperties> ? <wsa:ReferenceParameters>... </wsa:ReferenceParameters> ? <wsa:PortType>xs:QName</wsa:PortType> ? <wsa:ServiceName PortName="xs:NCName"?>xs:QName</wsa:ServiceName> ? <wsp:Policy> ... </wsp:Policy>* </wsa:EndpointReference></pre>	<p>URI Properties to identify the endpoint Parameters associated with an endpoint Endpoint type (WSDL 1.0 portType, WSDL 2.0 interface) Link to WSDL service element containing the endpoint descr. Security settings of endpoint</p>
--	---

WS-addressing EPR see http://www.w3.org/Submission/ws-addressing/#_Toc77464317

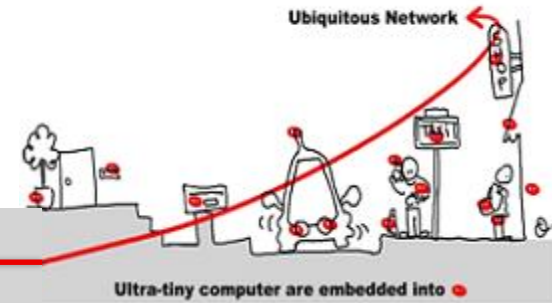
Example endpoint address URI:



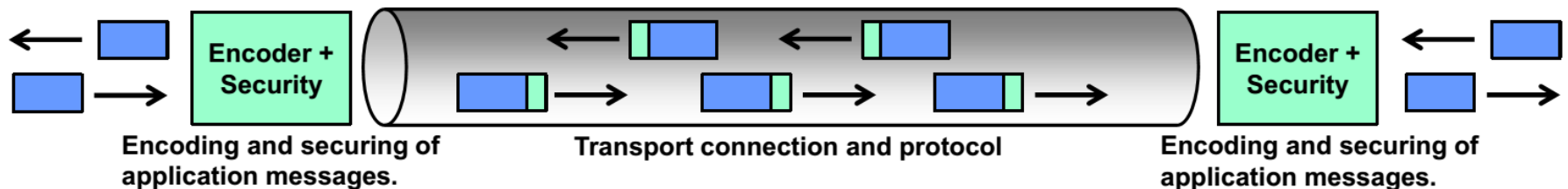
Endpoint address class `System.ServiceModel.EndpointAddress:`

`EndpointAddress.Uri` → URI
`EndpointAddress.Headers` → Reference properties and parameters
`EndpointAddress.Identity` → Security settings

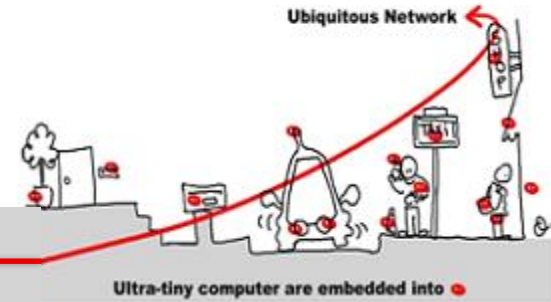
Le « B » de ABC



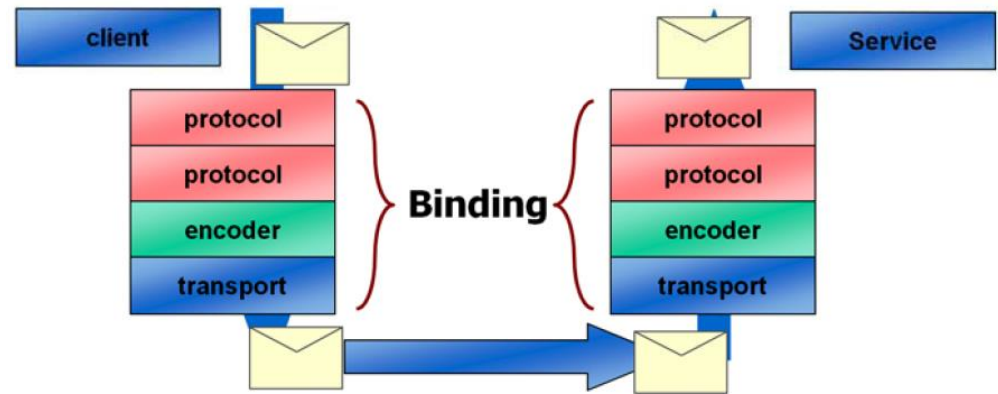
- Le binding est le moyen par lequel le endpoint est accessible
- Il contient les éléments suivants :
 - Le protocole de transport
 - Exemples: TCP, HTTP, MSMQ
 - Le format des messages
 - Exemples: Text/XML (SOAP), binary, MTOM (Message Transfer Optimized Mechanism).
 - Les paramètres de configuration de la sécurité :



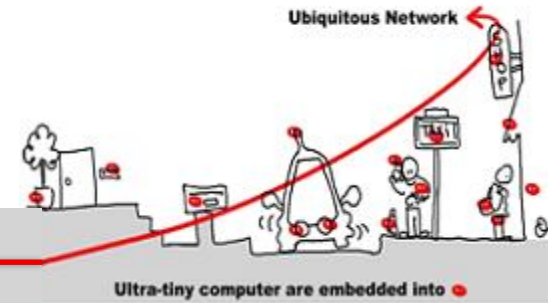
Bindings



- Plusieurs Binding:
 - **Intéropérabilité**
 - basicHttpBinding
 - webHttpBinding
 - wsHttpBinding, wsFederationHttpBinding
 - **Spécifique à WCF**
 - netTcpBinding
 - netPeerTcpBinding
 - netNamedPipeBinding
 - netMsmqBinding
 - **Interopérabilité MSMQ**
 - MsmqIntegrationBinding
 - **Extensible** (écrivez le votre)



Le « B » de ABC



- WCF fournit les binding suivants :

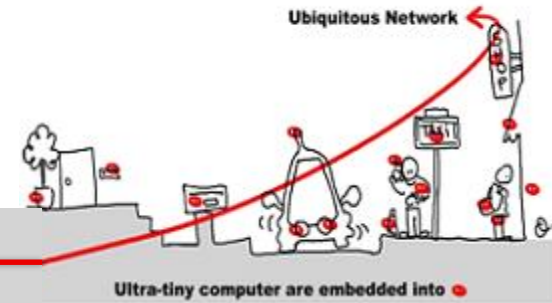
Binding	Interoperability	Security	Session	Transactions	Duplex	Encoding
BasicHttpBinding	WS-I Basic Profile	N, T, M, m	N	N	No	Text, MTOM
WSHttpBinding	WS-* standards	T, M, m	N, RS, SS	N, Yes	No	Text, MTOM
WSDualHttpBinding	WS-* standards	M, m	RS, SS	N, Yes	Yes	Text, MTOM
WSFederationHttpBinding	WS-Federation	N, M, m	RS, SS	N, Yes	No	Text, MTOM
NetTcpBinding	.NET	T, M, m, N	TS, RS, SS	N, Yes	Yes	Binary
NetNamedPipeBinding	.NET	T, N	N, TS	N, Yes	Yes	Binary
NetMsmqBinding	.NET (WCF)	M, T, N	N, TS	N, Yes	No	Binary
NetPeerTcpBinding	.NET	T	N	N	Yes	N/A
MsmqIntegrationBinding	MSMQ	T	N	N, Yes	No	MSMQ
BasicHttpContextBinding	WS-I Basic Profile	N, T, M, m	N	N	No	Text, MTOM
NetTcpContextBinding	.NET	N, T, M, m	T, RS, SS	N, Yes	Yes	Binary
WSHttpContextBinding	WS-* standards	T, M, m	N, RS, SS	N, Yes	No	Text, MTOM
WebHttpBinding	HTTP (REST)	N	N	N	No	POX

More details on WCF bindings see <http://msdn.microsoft.com/en-us/library/ms730879.aspx>.

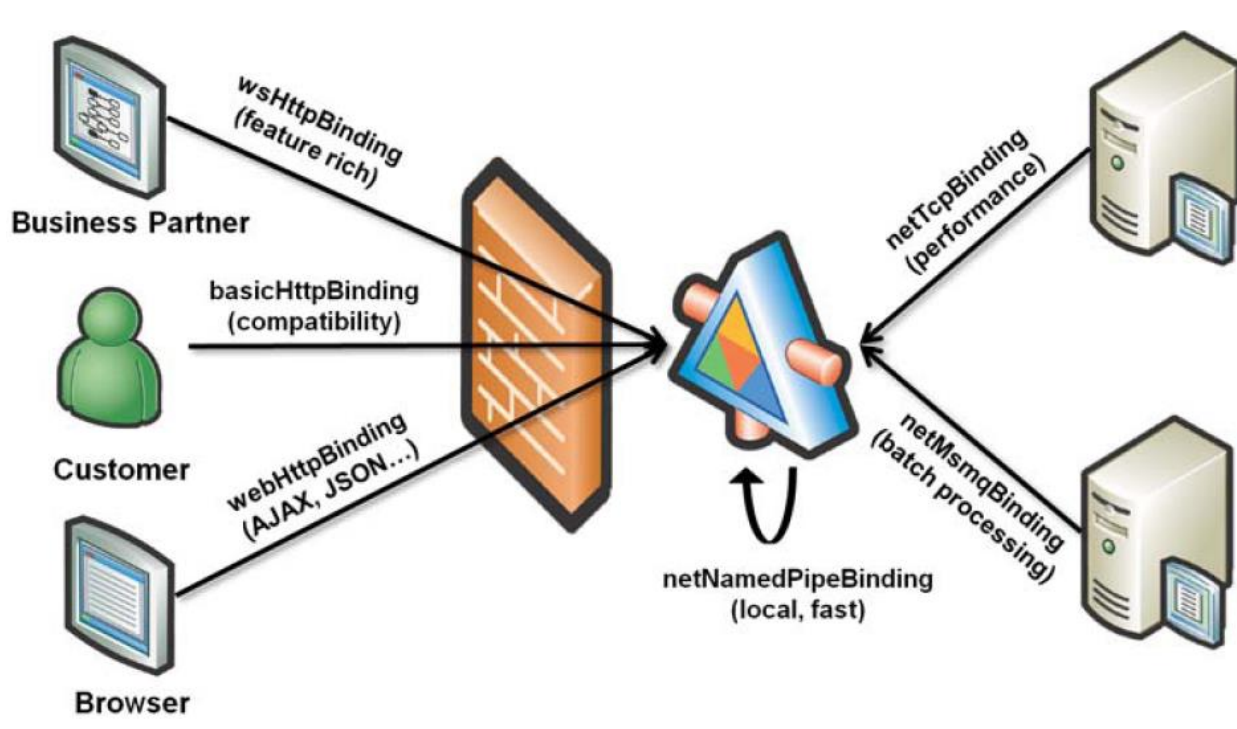
Key:

N: None RS: Reliable Session (WS-ReliableMessaging) POX: Plain Old XML
T: Transport SS: Security Session
M: Message TS: Transport Session
m: mixed

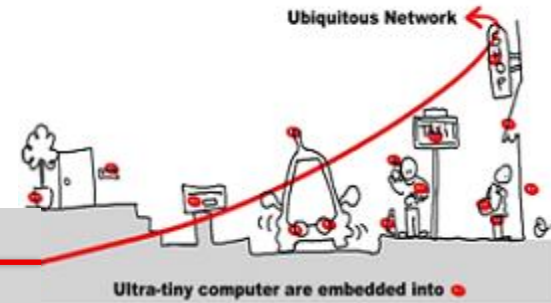
Choisir son Binding



- D'abord – Il faut choisir différents middlewares
- Ensuite – il faut choisir différents bindings



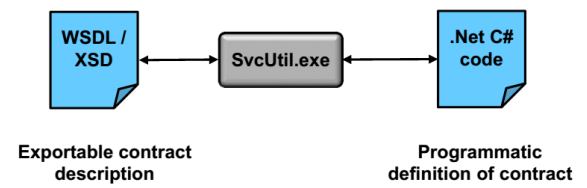
Le « C » de ABC



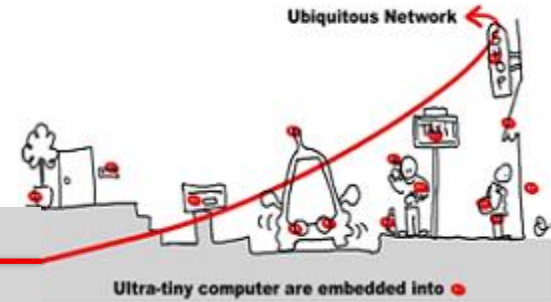
- Les interfaces WCF sont appelées des contrats
- Les contrats définissent des opérations, des structures de données et des messages
- Un contrat de service définit :
 - a. un groupement d'opérations dans un service (le .Net attribute [ServiceContract])
 - b. les signatures des operations (le .Net attribute [OperationContract])
 - c. Les types de données des opérations (le .Net attribute [DataContract])
- A partir d'un contrat vers du code ou vice-versa :

L' utilitaire SvcUtil.exe peut être utilisé pour les correspondances entre services abstraits, services interopérables (documents WSDL) et la définition des interfaces (code C #).

Soit SvcUtil.exe récupère la définition du service à partir d'un fichier WSDL sur le disque à l'aide du protocole DISCO (protocole propriétaire de Microsoft pour la découverte de service) ou directement à partir d'un service en cours d'exécution via par exemple un protocole de norme WS-MetadataExchange)



Contrats

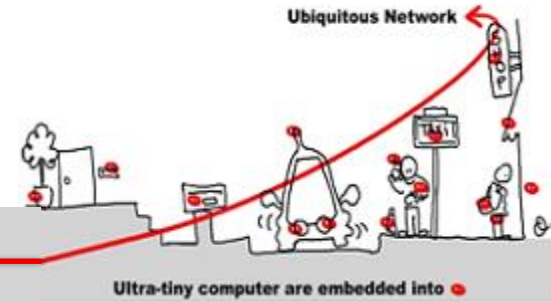


- Utilisation d'interfaces .Net annotées

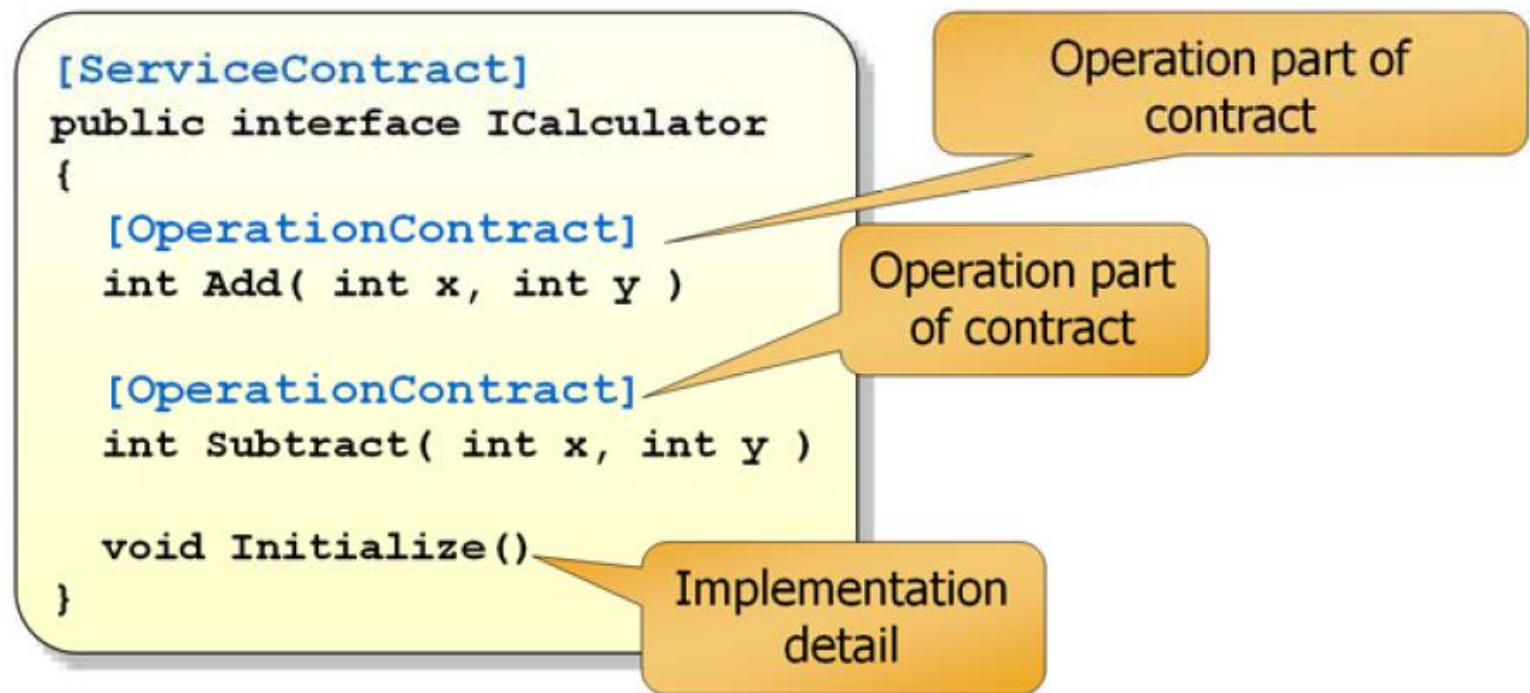
```
[ServiceContract]
interface ICalculator
{
    ...
}
```

Operations
defined here

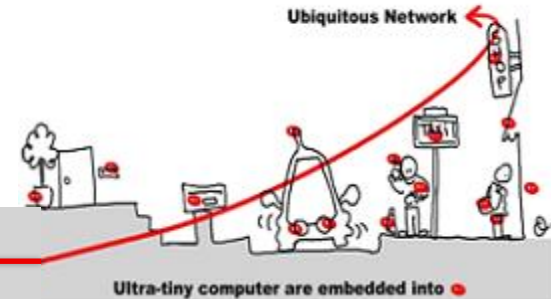
Contrats



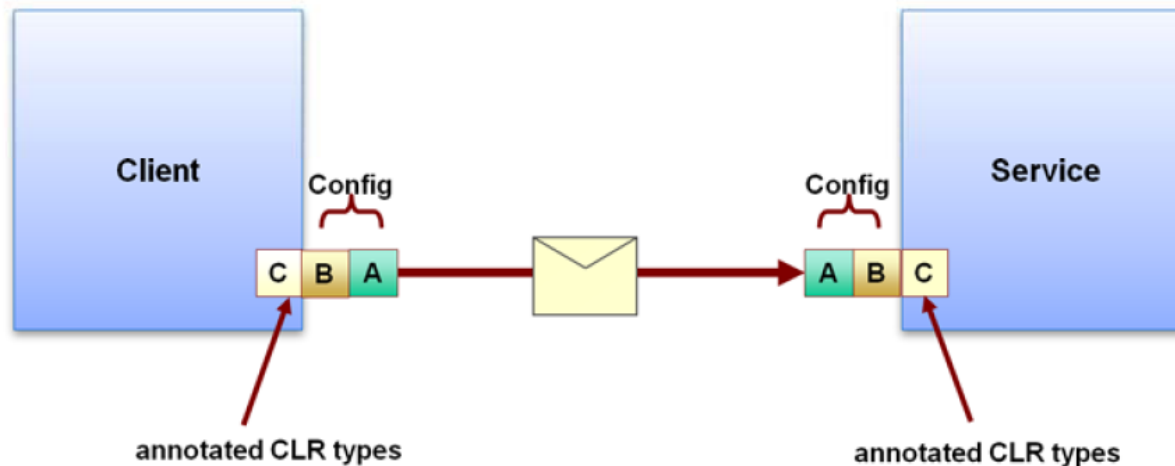
- Comme dans ASP.NET – exposition seulement des opérations explicitement annotées



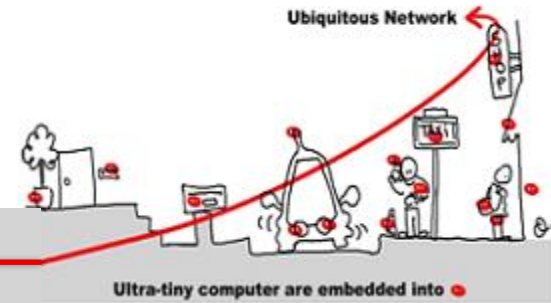
Endpoints



- Les Applications communiquent à travers des **endpoints**
- Les Endpoints sont définis par le modèle WCF **ABC**
 - Address (where is the service)
 - Binding (which transport and encoding to use)
 - Contract (what operations are available)

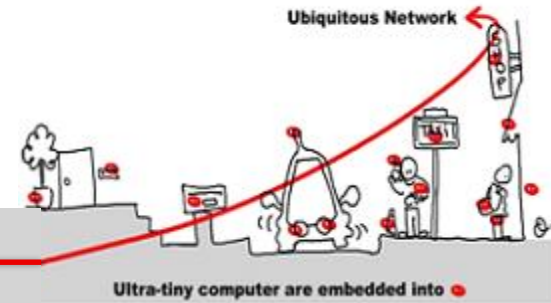


Endpoint et fichier de configuration



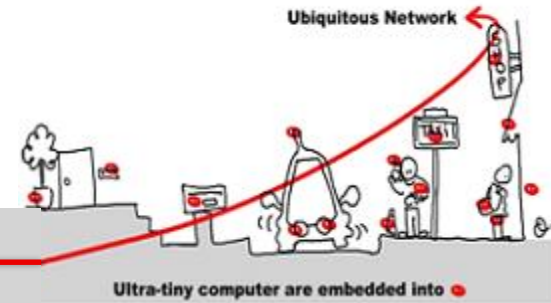
```
<services>
  <service name="Calc">
    <endpoint address="net.tcp://localhost/Service"
              binding="netTcpBinding"
              contract="IService" />
  </service>
</services>
```

Hébergement de Services



- Deux modèles supportés
 - Self-hosting: le service a son propre processus (console, winform)
 - WAS hosting, le service est hébergé sur un serveur WEB :
 - Windows Activation Service
 - Supported by IIS7
 - Emulated for IIS6 for HTTP/S

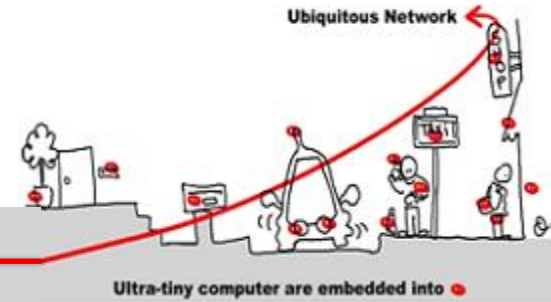
Self-hosting



- Make a **ServiceHost**, Add **Endpoints**, Call **Open**

```
serviceHost host = new serviceHost(typeof(Service));  
host.AddServiceEndpoint(typeof(IService),  
    new netTcpBinding(),  
    "net.tcp://localhost/service");  
  
host.open();  
  
console.WriteLine("service ready ...");  
console.ReadLine();  
  
host.close();
```

Création de Proxy Statique

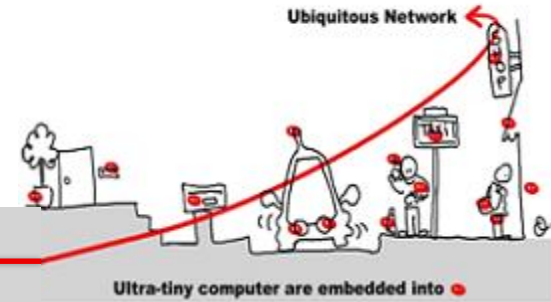


- Contrairement à .NET Remoting, qui ne supporte que la génération dynamique de proxy il y a aussi des générateurs statiques dans WCF

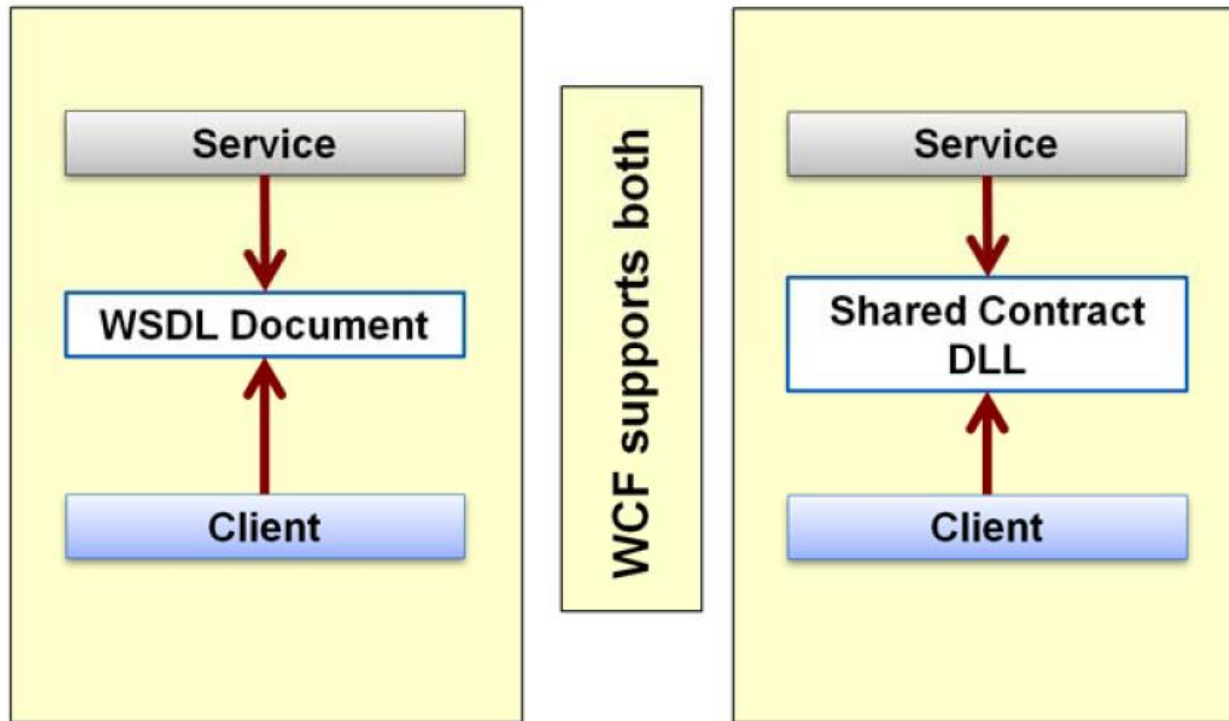
```
C:\>svcutil http://localhost/Basics/service  
/config:app.config /out:generatedProxy.cs
```

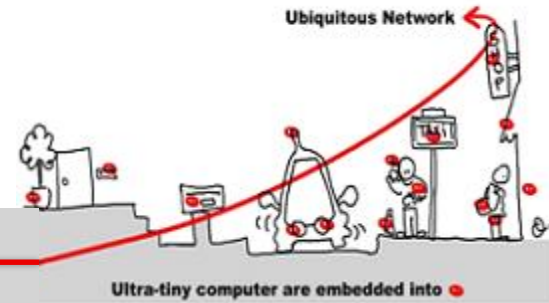
- Ou "Add Service Reference" dans VS

Meta Data



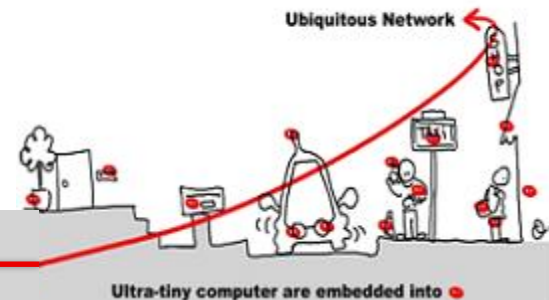
- Deux moyens de partager des contrats
- WSDL ou des DLL "Contract" partagées





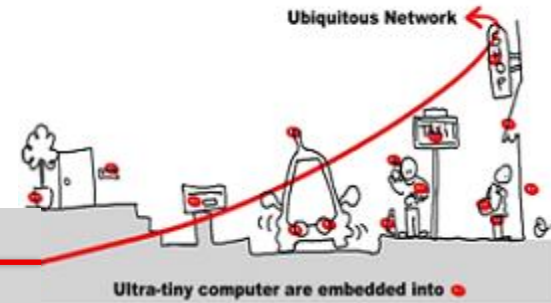
LES ETAPES POUR CRÉER UN SERVICE WCF

Les étapes pour créer un service WCF



- Selon le mode de déploiement, il faut définir des projets différents :
- Créer un projet Visual C# / WCF / WCF Service Library :
- Le fichier binaire produit sera une DLL (Dynamic Link Library), pour s'exécuter dans un WAS.
- Les fichiers de code source sont :
 - IService1.cs avec les attributs [ServiceContract] (service interface = contrat).
 - Service1.cs (implémentation du service).
 - App.config (fichier de configuration du transport binding et autres paramètres).
- Créer un projet Visual C# / WCF / WCF Service Application :
- Le fichier binaire produit sera une DLL (Dynamic Link Library), pour s'exécuter dans IIS.
- Les fichiers de code source sont :
 - IService1.cs avec les attributs [ServiceContract] (service interface = contrat).
 - Service1.cs (implémentation du service).
 - Web.config (fichier de configuration sur du HTTP binding et autres paramètres).

Les étapes pour créer un service WCF



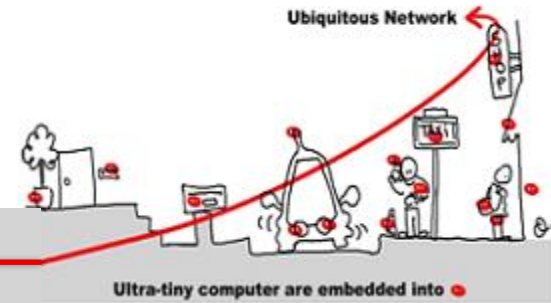
- Définition de l'interface (contrat de service comme une interface C# ou une classe C#):
 - Définit une interface d'un web service avec une interface .Net
 - Exemple un contrat défini en C# (contenant des contract sur des operations):

```
[ServiceContract(Namespace = "http://indigoo.WSMW")]  
public interface IMyInterface  
{  
    [OperationContract]  
    string MyFunction();  
    [OperationContract]  
    int MyOtherFunction();  
}
```

- Implémentation d'un contrat de service (classe C# implémentant l'interface):
 - Créer une classe C#

```
public class MyService : IMyInterface  
{  
    public string MyFunction() {...}  
    ...  
}
```


Les étapes pour créer un service WCF



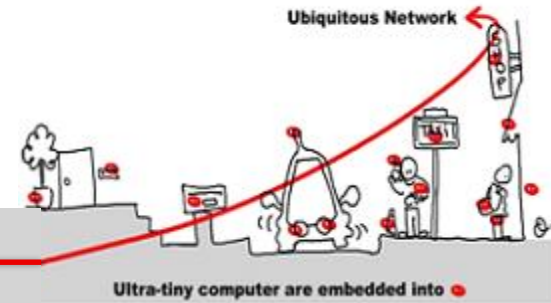
- Déploiement et exécution du service :
- Le déploiement se fait dans une DLL ou directement en lançant un nouveau processus (self-hosting service)
- Dans ce dernier cas, le code suivant doit être ajouté au service

```
selfHost.AddServiceEndpoint(typeof(MyService), new BasicHttpBinding(), "MyService");
```

- Enfin, le service peut-être lancé avec la méthode :

```
selfHost.Open();
```

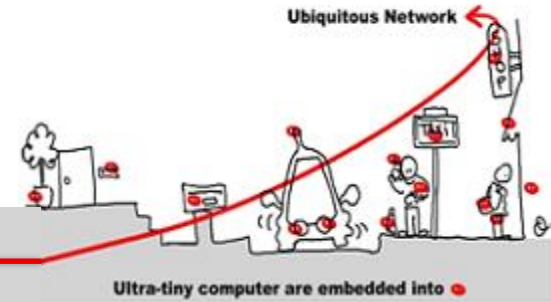
WCF, configurer plutôt que programmer



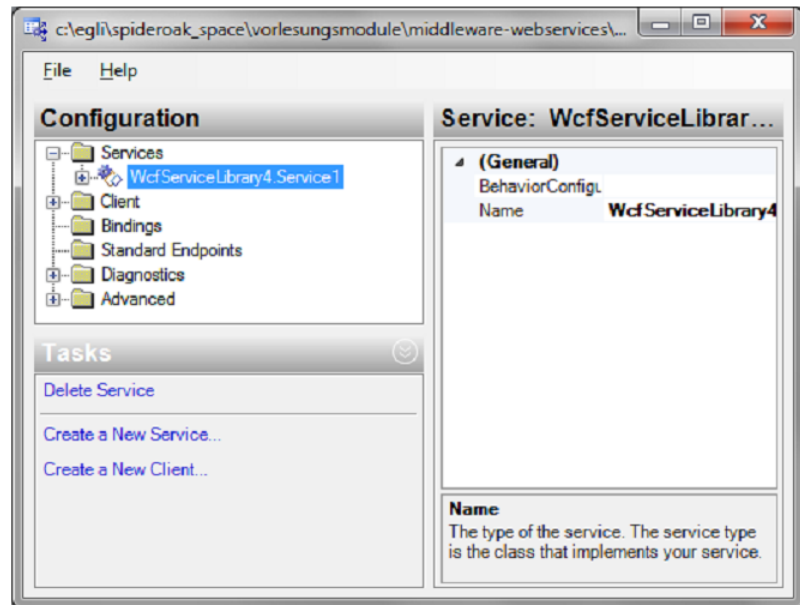
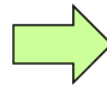
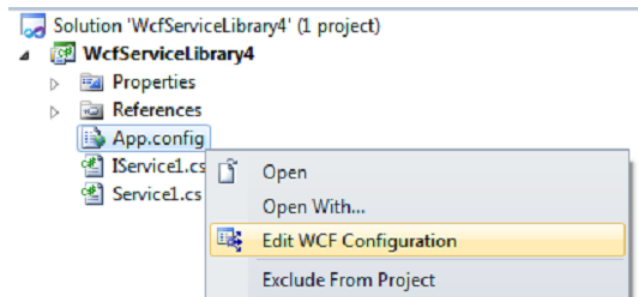
- Un Client et un service WCF peuvent être définis dans une configuration (déclarative en XML or programmatique avec des classes .Net).
- Bonnes pratiques :
 - Pour améliorer la réutilisabilité des services web dans des environnements différents, les contrats (C), les adresses (A) et le binding ('B') doivent être séparés.
 - Définir l'interface et son implémentation par programmation sans mentionner de protocole de transport et de format de message
 - La définition du protocole de transport, de l'adresse, et du format de message sera faite dans un fichier de configuration
 - Les fichiers de configuration sont App.config ou Web.config comme :

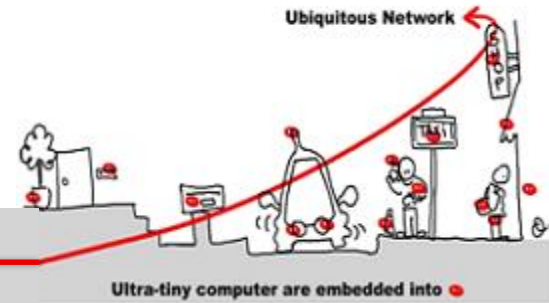
```
<system.ServiceModel>
  <services>
    <service>
      <endpoint/>
    </service>
  </services>
  <bindings>
    <!-- Specify one or more of the system-provided binding elements, for example, <basicHttpBinding> -->
    <!-- Alternatively, <customBinding> elements. -->
    <binding> <!-- For example, a <BasicHttpBinding> element. --></binding>
  </bindings>
  <behaviors> <!-- One or more of the system-provided or custom behavior elements. -->
    <behavior> <!-- For example, a <throttling> element. --> </behavior>
  </behaviors>
</system.ServiceModel>
```

WCF, configurer plutôt que programmer



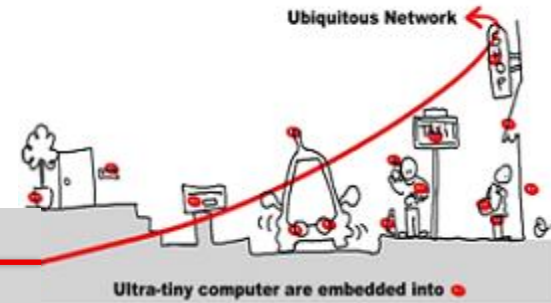
- La configuration du service peut se faire graphiquement avec SvcConfigEditor.exe:
Microsoft\SDKs\Windows\v7.0A\Bin\SvcConfigEditor.exe.





LE CODE

Définition de Contrat



```
using System.ServiceModel;
```

```
//a WCF contract defined using an interface
```

```
[ServiceContract]  
public interface IMath  
{  
    [OperationContract]  
    int Add(int x, int y);  
}
```

```
//the service class implements the interface
```

```
public class MathService : IMath  
{  
    public int Add(int x, int y)  
    { return x + y; }  
}
```

Implémentation de Service

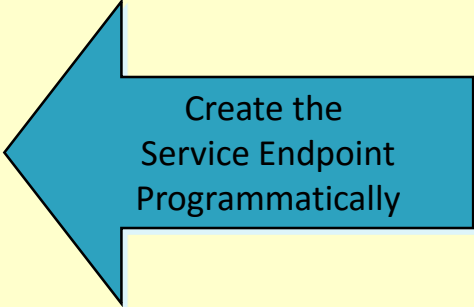
```
public class WCFServiceApp
{
    public void DefineEndpointProgrammable()
    {
        //create a service host for MathService
        ServiceHost sh = new ServiceHost(typeof(MathService));

        //use the AddEndpoint helper method to
        //create the ServiceEndpoint and add it
        //to the ServiceDescription
        sh.AddServiceEndpoint(
            typeof(IMath), //contract type
            new WSHttpBinding(), //one of the built-in bindings
            "http://localhost/MathService/Ep1"); //the endpoint's address

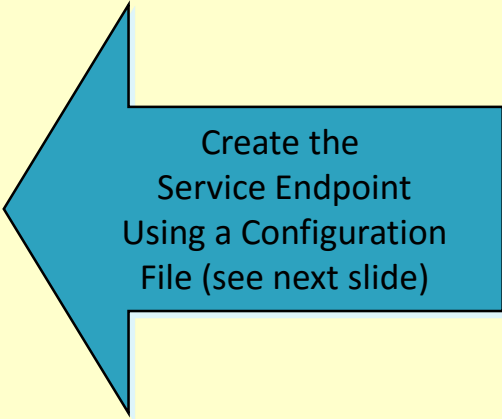
        //create and open the service runtime
        sh.Open();
    }

    public void DefineEndpointInConfig()
    {
        //create a service host for MathService
        ServiceHost sh = new ServiceHost (typeof(MathService));

        //create and open the service runtime
        sh.Open();
    }
}
```

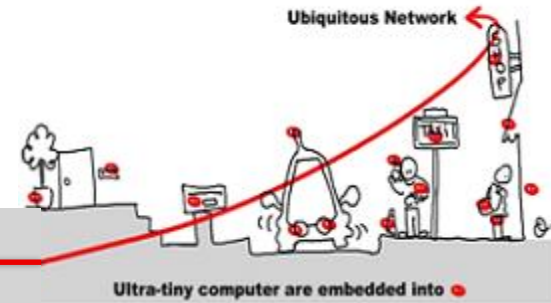


Create the
Service Endpoint
Programmatically



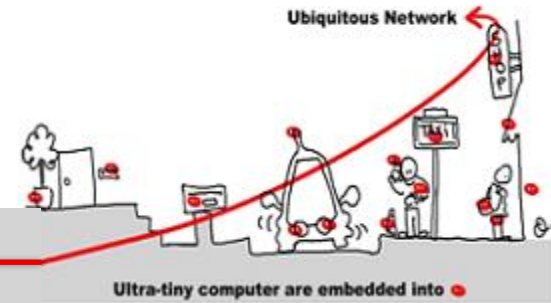
Create the
Service Endpoint
Using a Configuration
File (see next slide)

Fichier de Configuration



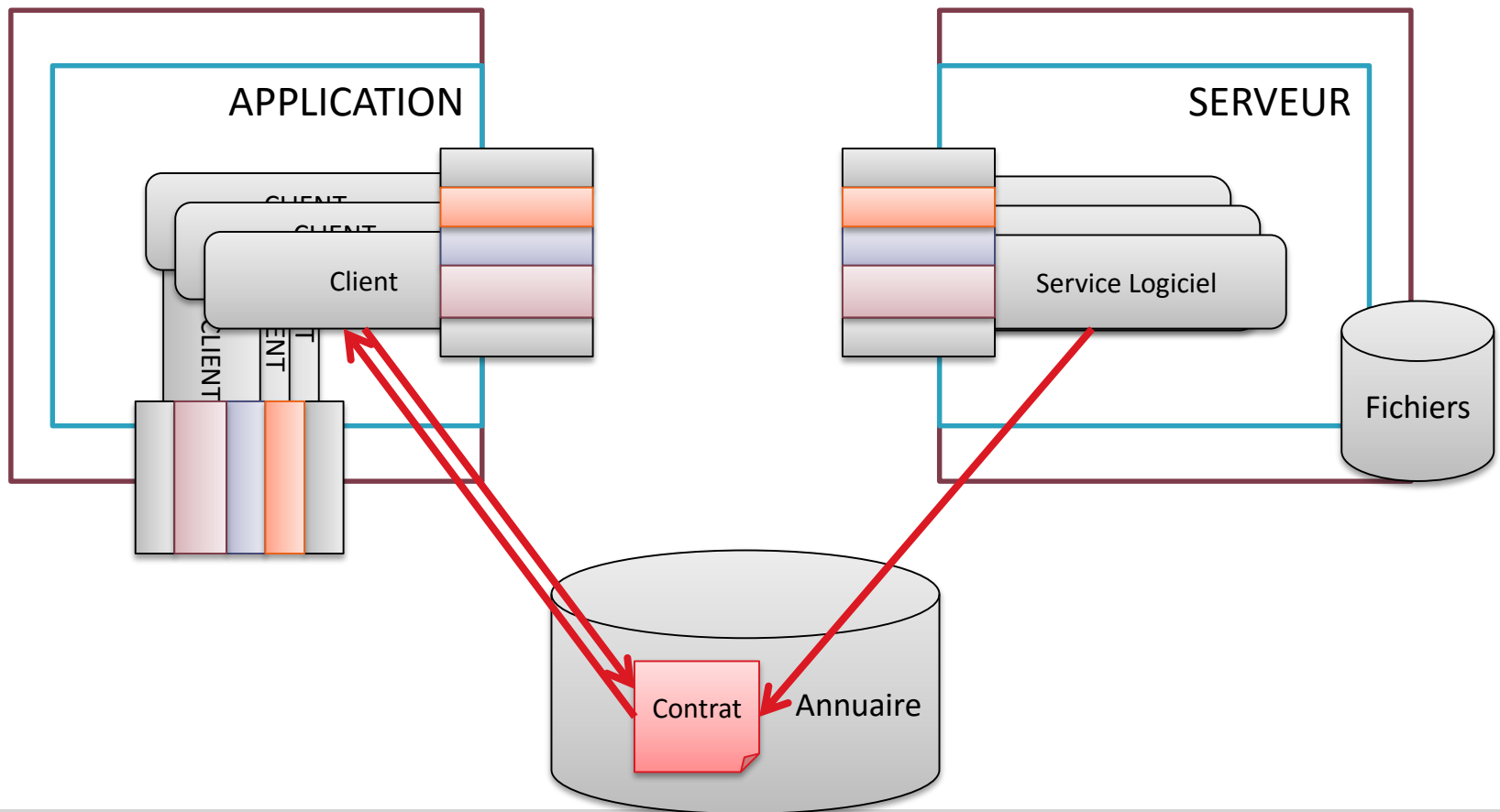
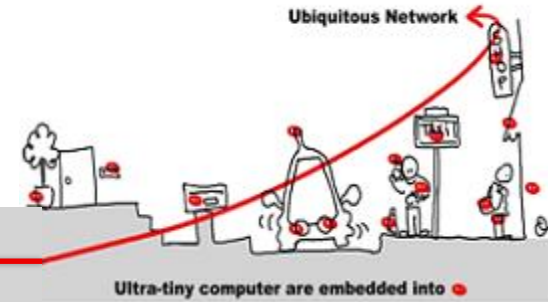
```
<!-- configuration file used by above code -->
<configuration
  xmlns="http://schemas.microsoft.com/.NetConfiguration/v2.0">
  <system.serviceModel>
    <services>
      <!-- service element references the service type -->
      <service type="MathService">
        <!-- endpoint element defines the ABC's of the endpoint -->
        <endpoint
          address="http://localhost/MathService/Ep1"
          binding="wsHttpBinding"
          contract="IMath"/>
        </service>
      </services>
    </system.serviceModel>
  </configuration>
```

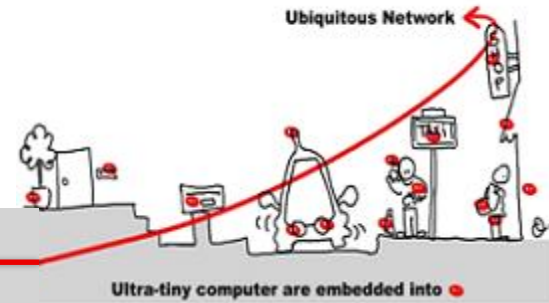

Implementation de Client Utilisant un Proxy Dynamique



```
public class WCFClientApp
{
    public void SendMessageToEndpointUsingChannel()
    {
        //this uses ChannelFactory to create the channel
        //you must specify the address, the binding and
        //the contract type (IMath)
        ChannelFactory<IMath> factory=new ChannelFactory<IMath>(
            new WSHttpBinding(),
            new EndpointAddress("http://localhost/MathService/Ep1"));
        IMath channel=factory.CreateChannel();
        int result=channel.Add(35,7);
        factory.Close();
    }
}
```

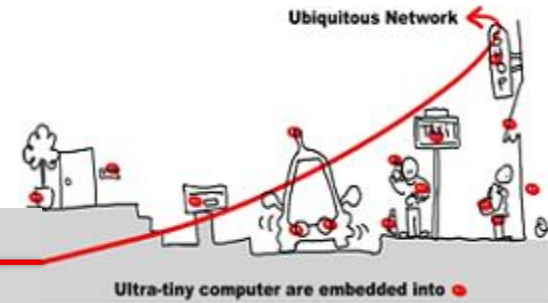
Architecture SOA / ROA



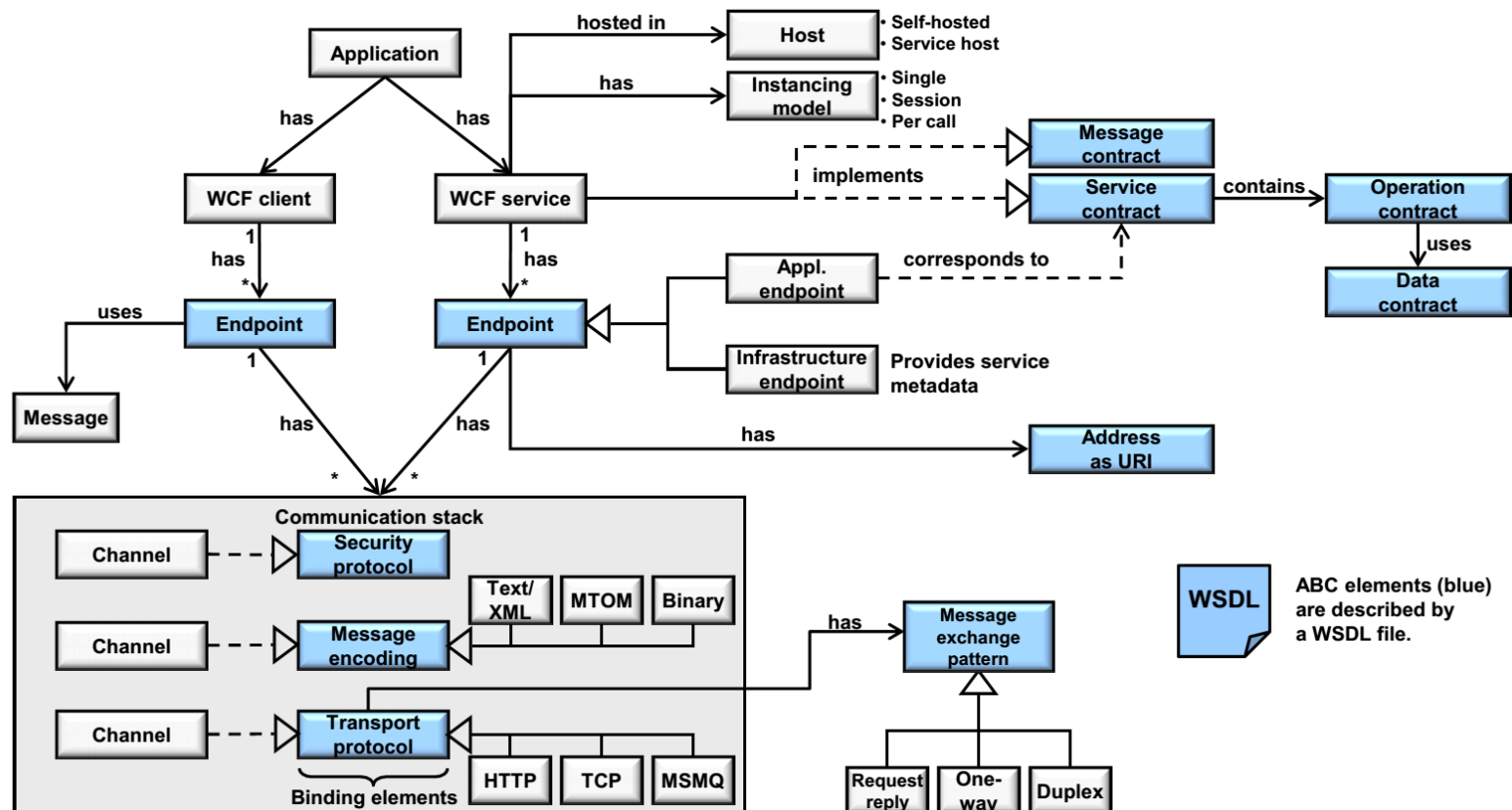


EN DETAIL

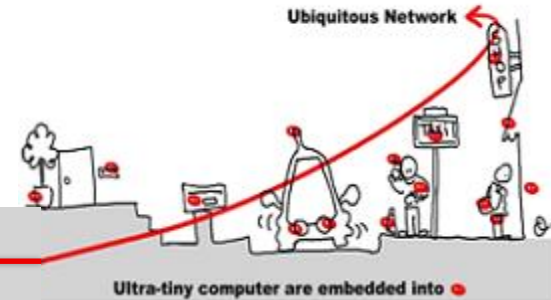
Modèle ABC de Service



- WCF fournit un modèle complet

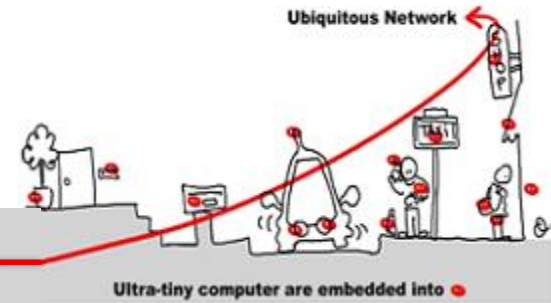


Modèle ABC de Service



- **Application**: une application à un client WCF et/ou un service WCF
- **Client WCF** : Un client est un composant avec un endpoint pour communiquer avec un service WCF.
- **Service WCF** : S'exécute dans son propre process (self-hosted) ou sur un serveur spécifique (specific service hosting process, comme IIS, Windows Activation Service, Windows Service).
- **Endpoint**: Client et Service WCF client utilise un endpoint pour communiquer.
- **Application endpoint**: Le endpoint sur lequel un service applicatif est exposé/fourni.
- **Infrastructure endpoint**: Part of WCF system to offer metadata of an application service.
- **Message**: unité d'échange de données entre client et service WCF. WCF est exclusivement basé sur des messages.
- **Adresse** : Une adresse physique comprend un nom d'hôte, un numéro de port, et un nom de service (URI). Le service applicatif est accessible avec cette adresse
- Exemple.: `http://localhost:8000/HSZ-TWSMW/DateTimeService`

Modèle ABC de Service



- **Le protocole de transport** (transport binding): définit le protocole utilisé pour transférer les messages (ex. HTTP over TCP, ou TCP, ...)
- **Le format des message** (message encoding binding element): définit le formatage des messages avant qu'ils soient envoyé sur le réseau grâce au protocole de transport (ex. Texte/XML, SOAP ou MTOM (Message Transmission Optimized Mechanism) pour le transfert efficace de données binaires).
- **Le protocole de sécurité** (security binding): définit les fonctions de sécurité qui sont appliquées aux messages (authentication, cryptage).
- **Le pattern d'échange de message** (vu sous le nom de PortType) : définit comment les messages sont échangés (request-reply, one-way, duplex).
- **La pile de communication** : comprend différents éléments de binding, au minimum le transport binding et le message encoding binding element.
- **L'hôte**: l'environnement d'exécution pour le service sur un process indépendant (self-hosted) ou sur un serveur spécifique (comme IIS, Windows Activation Service, Windows Service). Either self-hosted (specific process for service)