

Introduction aux Services Web

Département Sciences Informatiques
Jean-Yves Tigli – tigli@polytech.unice.fr
<http://www.tigli.fr>

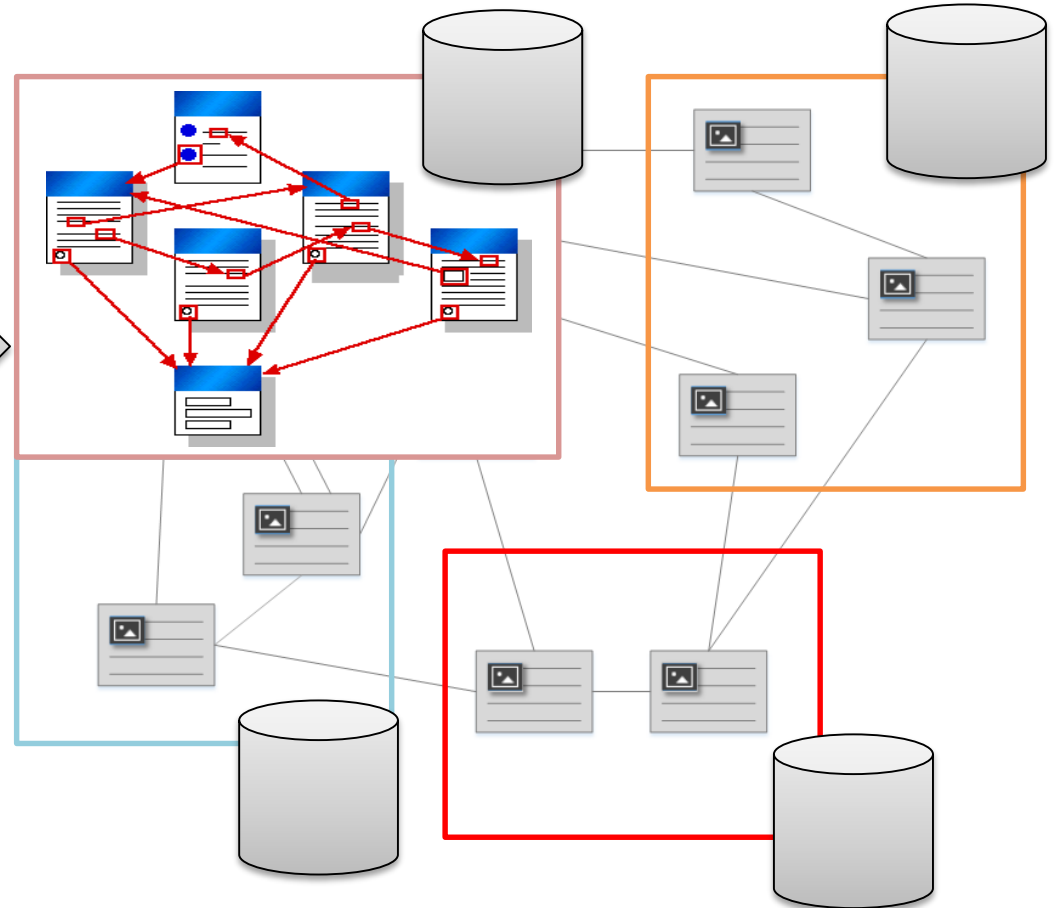
SI 3^{ème} année



Le web de H2M à M2M

Principes du Web Statique H2M

- ✓ Serveurs,
- ✓ Pages
- ✓ Liens hypertexte



Client / Serveur Particuliers

- ✓ **Page Web:**
 - Pointés par une URL
 - La plupart des pages WEB se composent de:
 - Une page HTML de base,
 - Différentes références à des « objets »

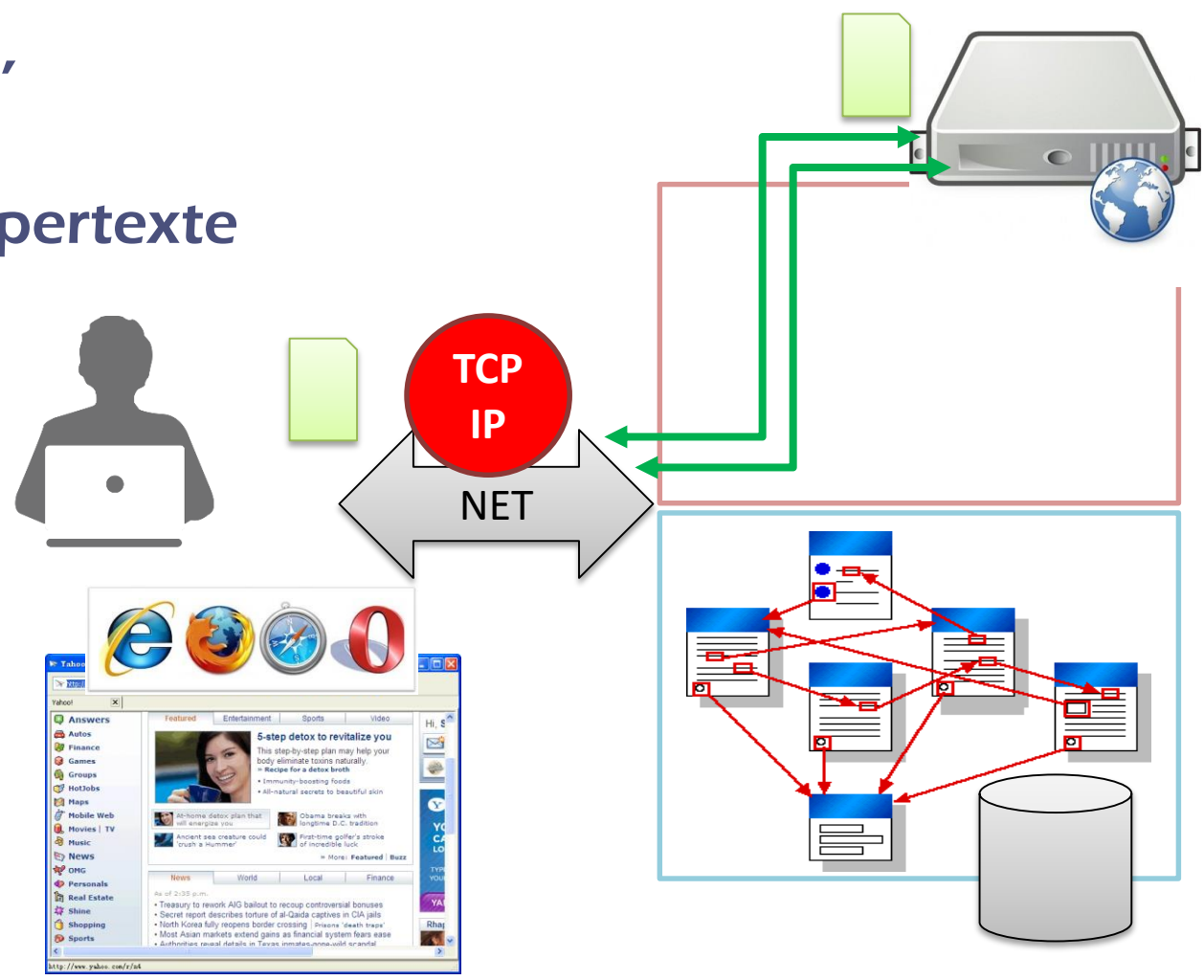
- ✓ **L'agent utilisateur (client) pour le Web s'appelle un "browser" (butineur en français)**
 - Microsoft Internet Explorer, Mozilla FireFox, Opera, Safari, Google Chrome, ...

- ✓ **Un serveur pour le Web s'appelle un serveur Web :**
 - Apache, Microsoft Internet Information Server (IIS), ...

Principes du Web Dynamique

H2M

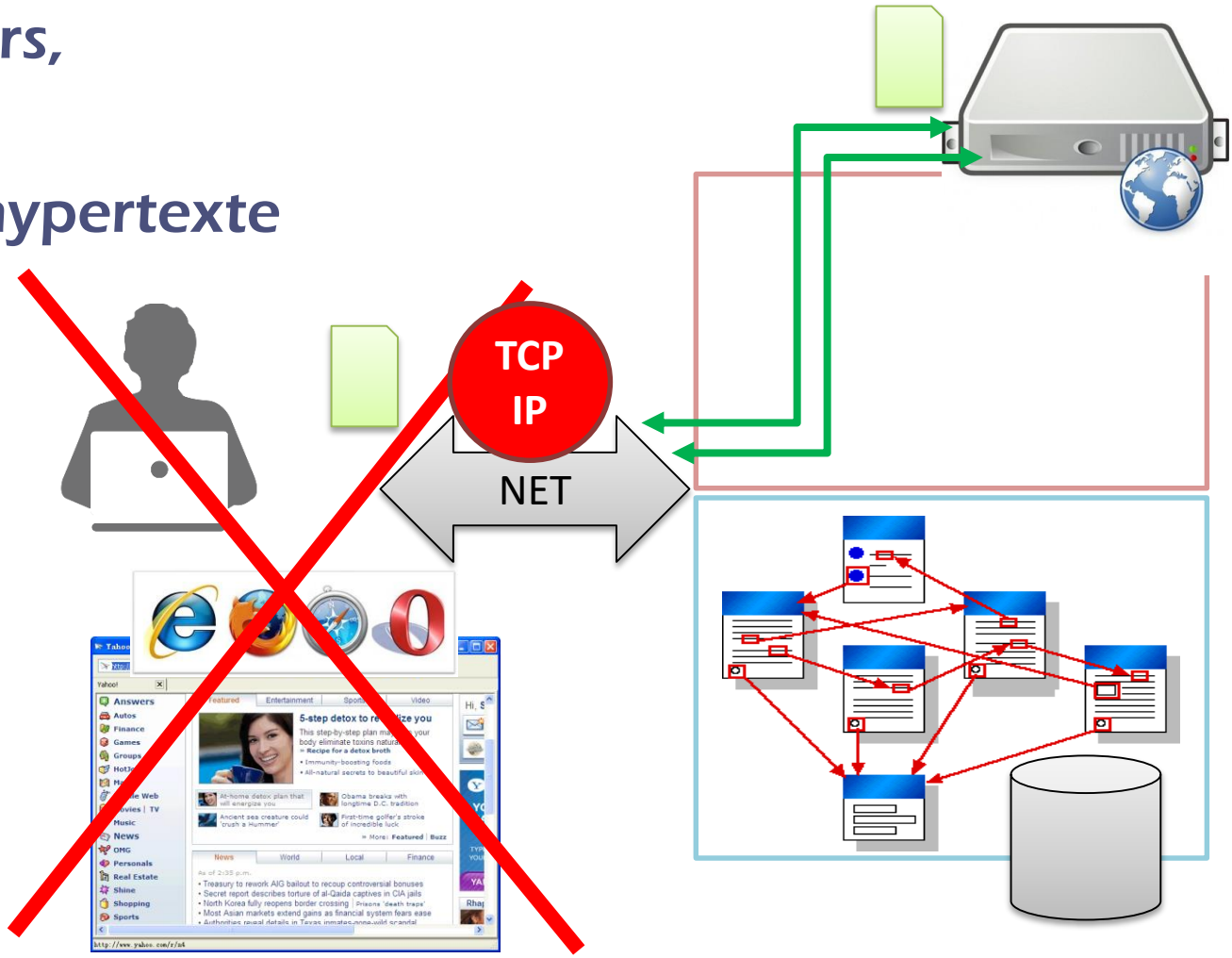
- ✓ Serveurs,
- ✓ Pages
- ✓ Liens hypertexte



LE WEB M2M

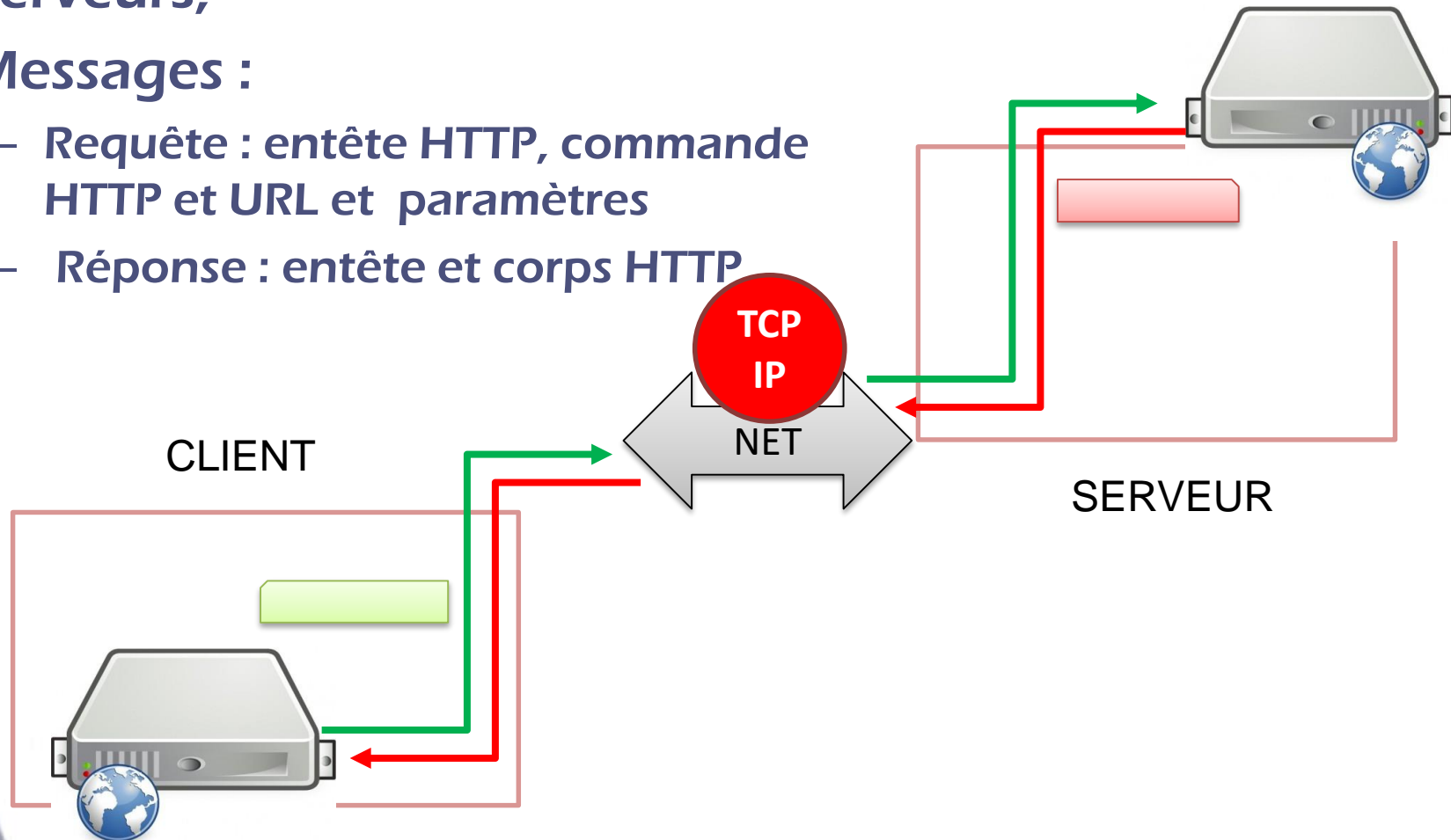
Principes du Web M2M

- ✓ Serveurs,
- ✓ Pages
- ✓ Liens hypertexte



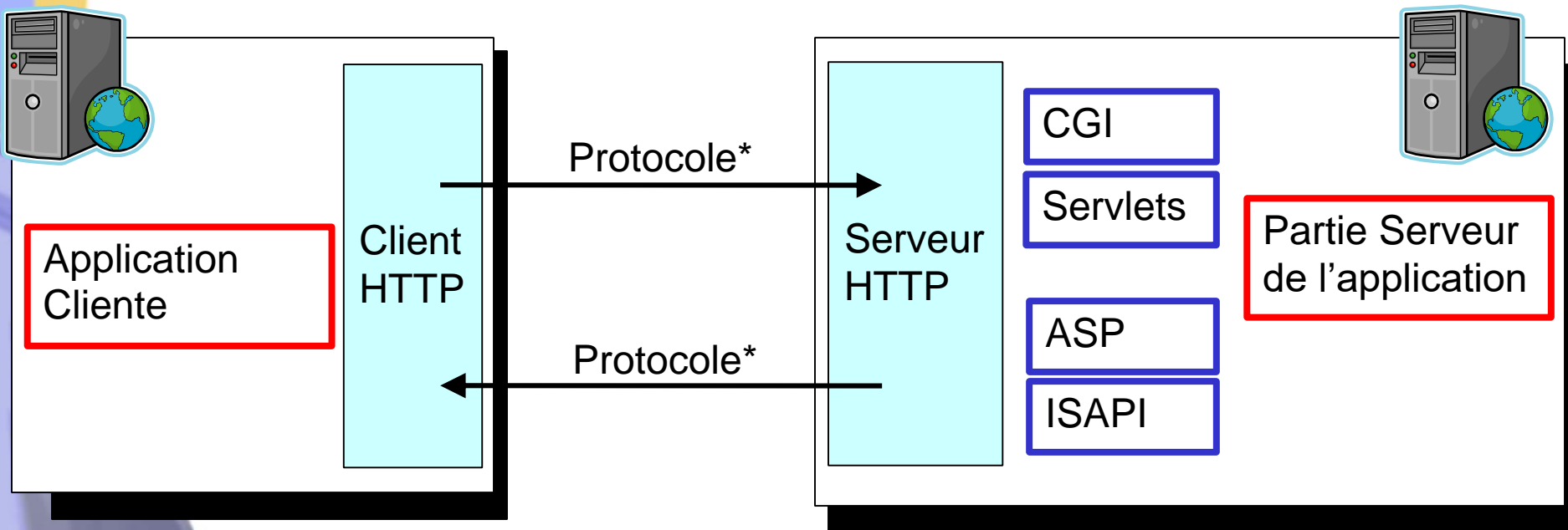
Principes du Web M2M

- ✓ Serveurs,
- ✓ Messages :
 - Requête : entête HTTP, commande HTTP et URL et paramètres
 - Réponse : entête et corps HTTP



Web M2M : Des CGI Bin aux Services Web

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

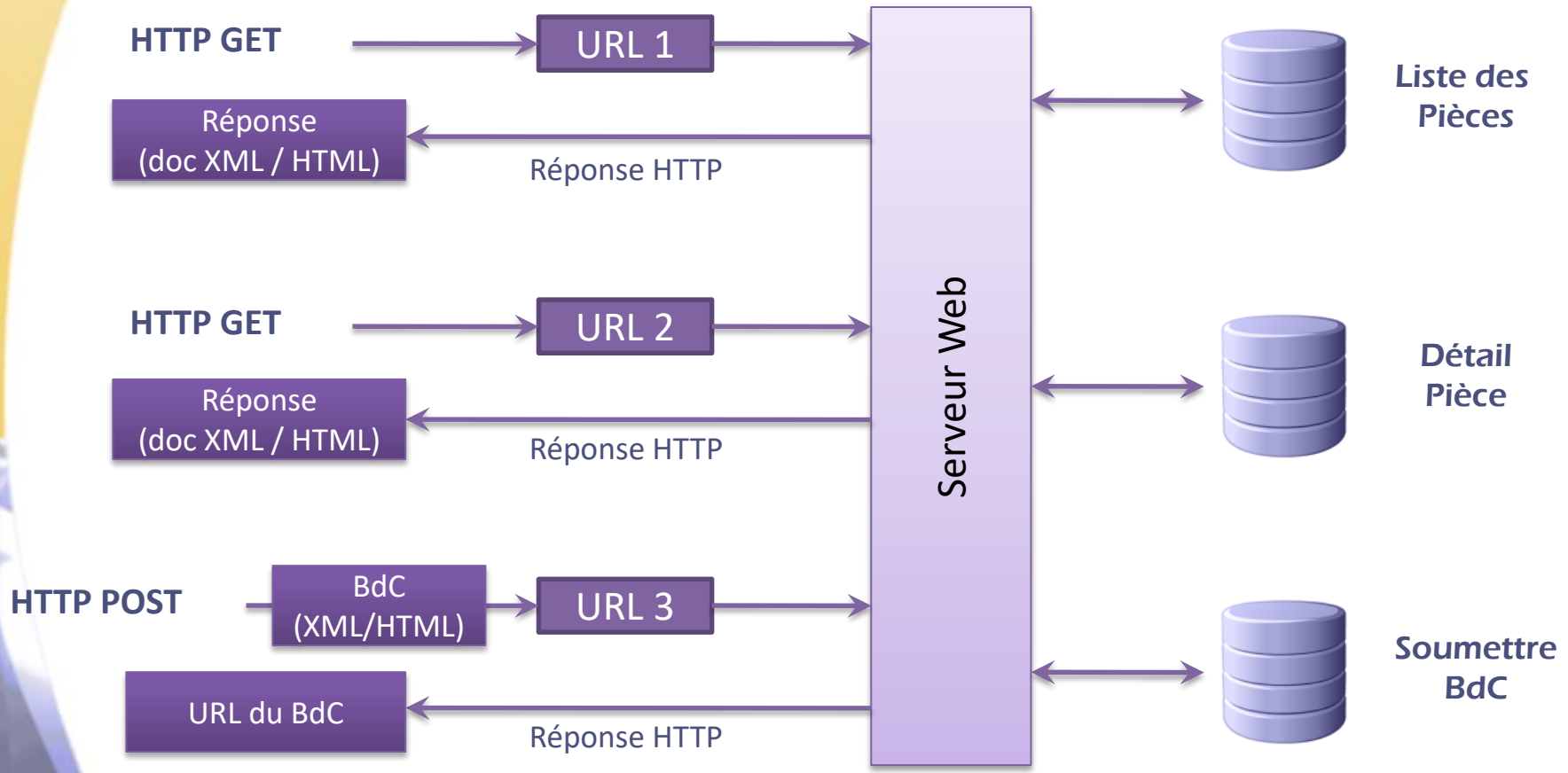


ROA versus SOA

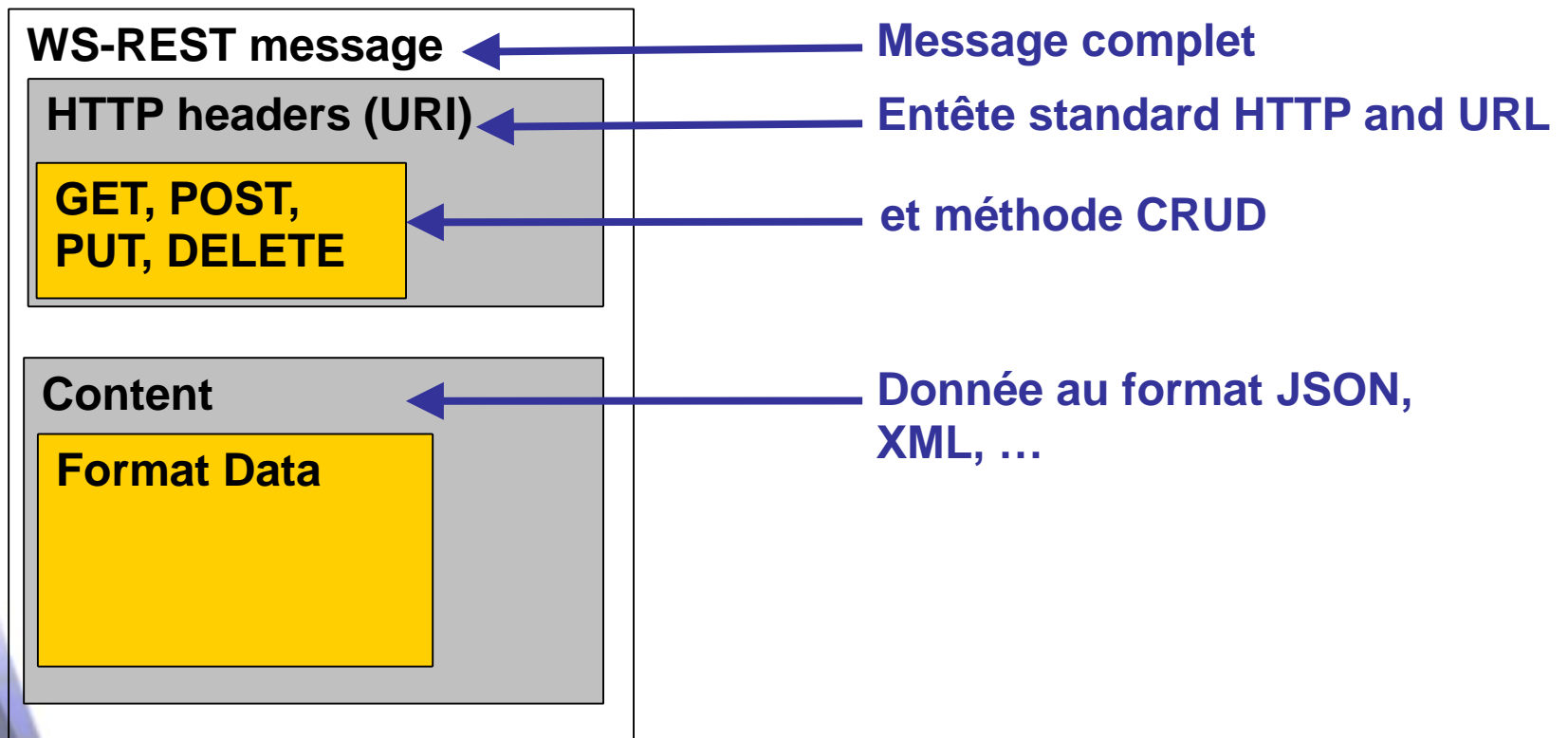
Ressource oriented Approach
Service oriented Approach

REST versus WS-SOAP

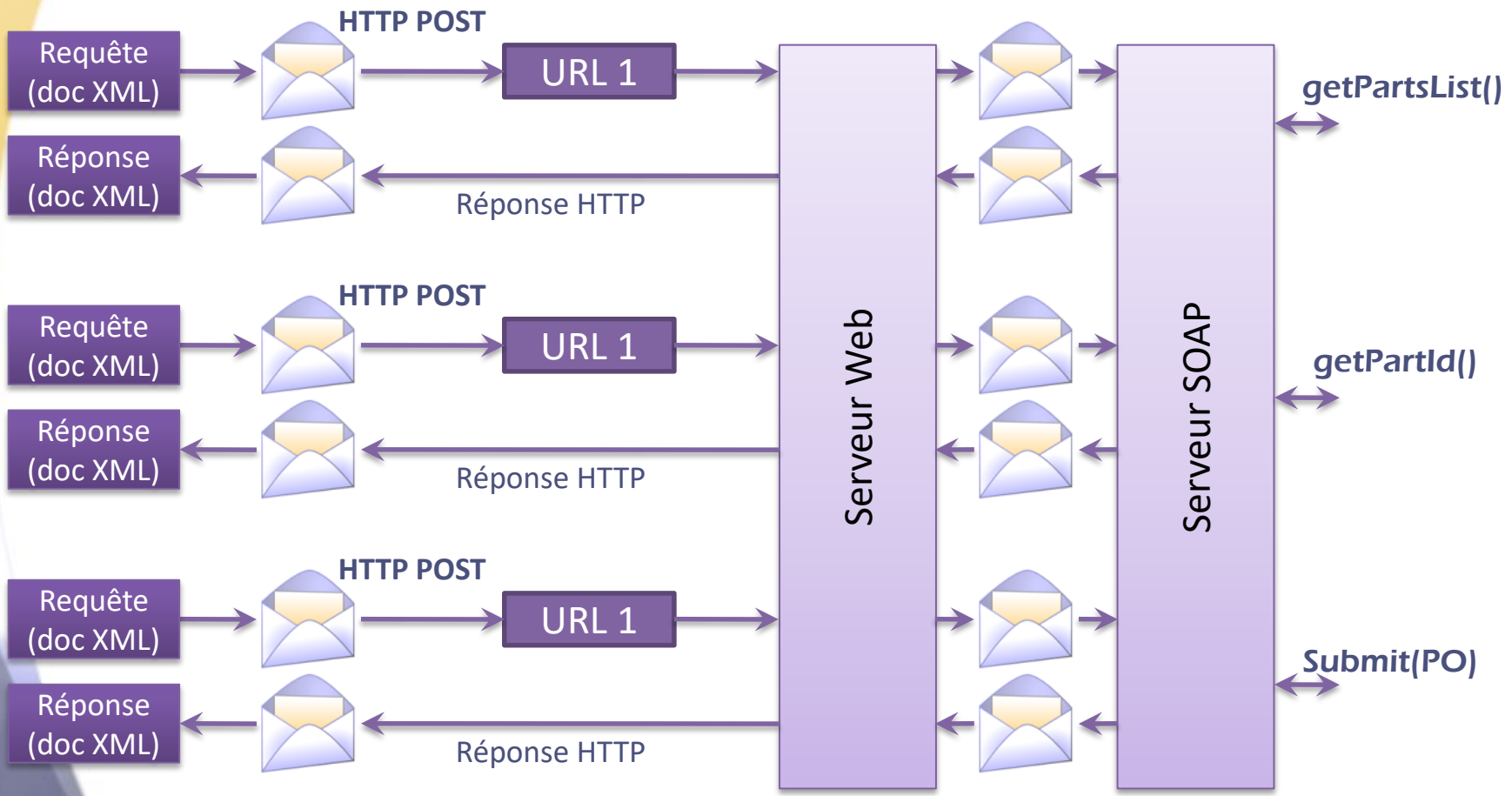
REST pour une approche ROA



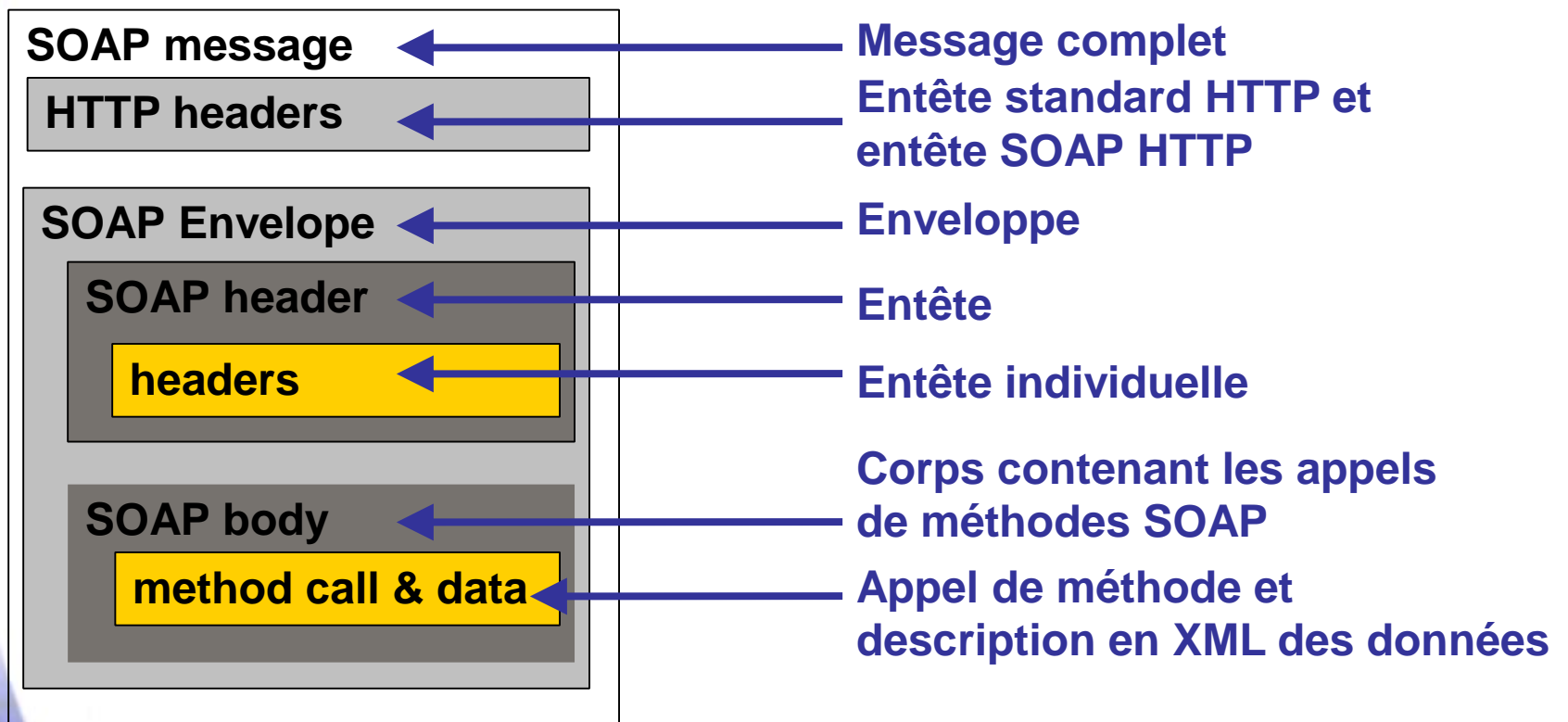
La structure des messages WS-REST



SOAP pour une approche SOA



La structure des messages WS-SOAP



Architecture orientées ressources

Web Service de type REST (WS-REST)
Les ressources sont identifiées par des URL

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 marge 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

Achitecture orientée service

WS-SOAP, WS-* du W3C

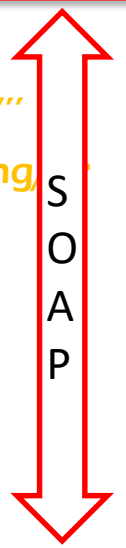
Exemple de requête SOAP utilisant HTTP

✓ Demande de cotation à un serveur :

```
POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAP-Action: "Some-URI"
```

Une seule commande
HTTP/POST
« envoi de message SOAP »

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/enveloppe/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding"
  <SOAP-ENV:Body>
    <m:GetLastTradePrice xmlns:m="Some-URI">
      <symbol>DIS</symbol>
    </m:GetLastTradePrice>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```



Exemple de réponse SOAP utilisant HTTP

✓ Réponse du serveur

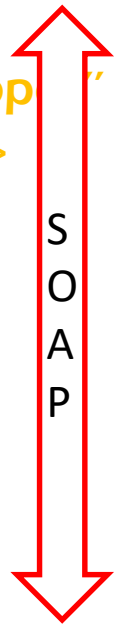
HTTP/1.1 200 OK

Content-Type: text/xml; charset="utf-8"

Content-Length: nnnn

Réponse au HTTP/POST

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
  <SOAP-ENV:Body>
    <m:GetLastTradePrice xmlns:m="Some-URI">
      <Price>34.5</Price>
    </m:GetLastTradePrice>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```



Modèle de message

- ✓ **SOAP permet une communication par message**
 - d'un expéditeur vers un récepteur
- ✓ **Structure d'un message**
 - **Envelop (enveloppe)**
 - **Élément racine**
 - **Namespace : SOAP-ENV**
<http://schemas.xmlsoap.org/soap/envelope/>
 - **Header (entête)**
 - **Élément optionnel**
 - **Contient des entrées non applicatives (transactions, session, ...)**
 - **Body (corps)**
 - **Contient les entrées du message**
 - **Nom d'une procédure, valeur des paramètres, valeur de retour**
 - **Peut contenir les éléments « fault » (erreurs)**

Entête d'un message: Header

- ✓ Contient des entrées non applicatives
 - Transactions, sessions, ...
- ✓ L'attribut `mustUnderstand` par exemple
 - Si absent ou =0 l'élément est optionnel pour l'application réceptrice
 - si =1, l'élément doit être compris par l'application réceptrice sinon le traitement du message par le récepteur doit échouer

✓ Exemple

```
<SOAP-ENV:Header>
```

```
  <t:Transaction xmlns:t="Some-URI" SOAP-  
  ENV:mustUnderstand="1">
```

```
  </t:Transaction>
```

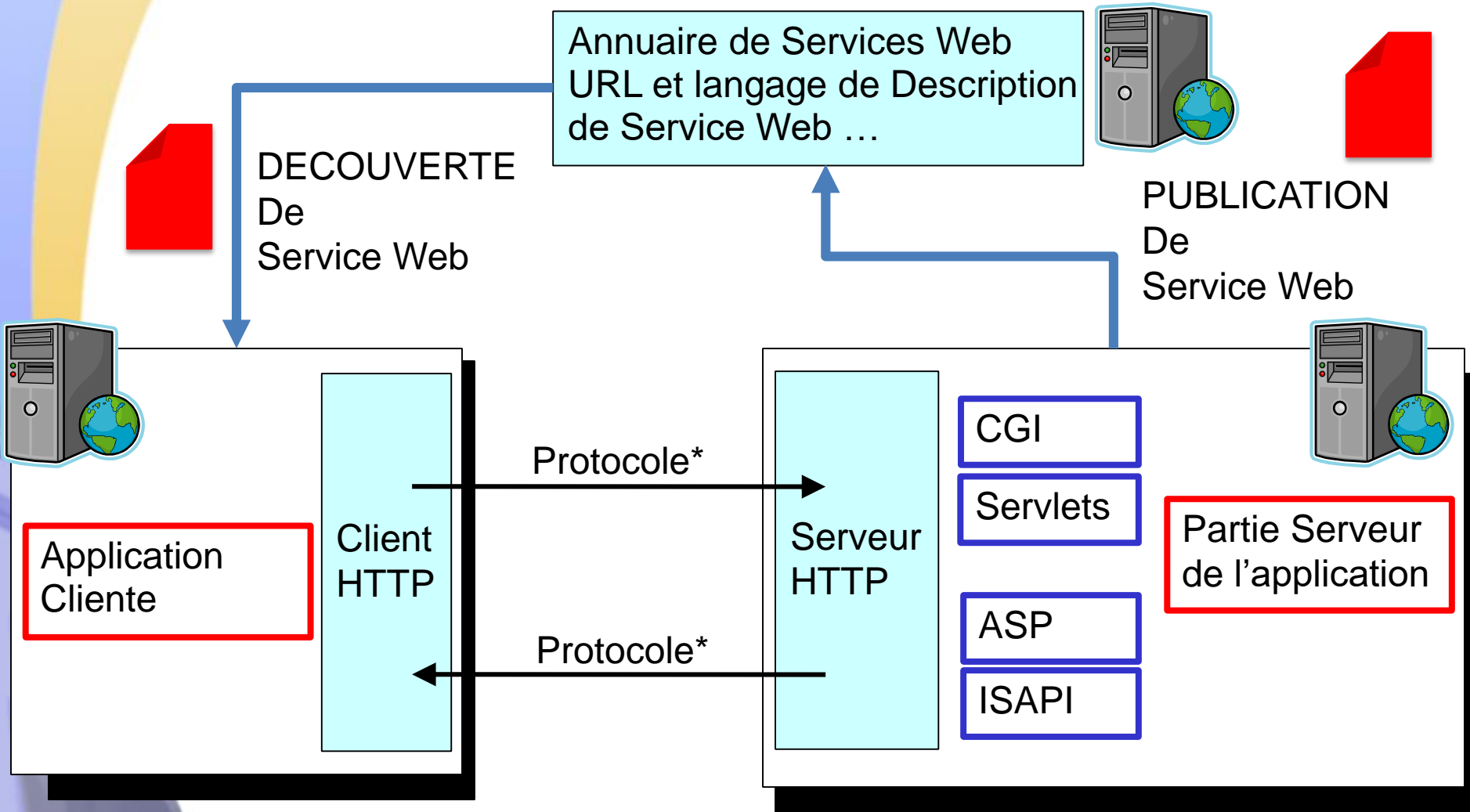
```
</SOAP-ENV:Header>
```

Corps d'un message

- ✓ Contient des entrées applicatives
- ✓ Encodage des entrées
- ✓ Namespace pour l'encodage
 - SOAP-ENC <http://schemas.xmlsoap.org/soap/encoding/>
 - XSD : Schéma XML
- ✓ Principe des règles d'encodage
 - Les règles d'encodage définissent un système de type
 - SOAP utilise les conventions XSD
 - Les tableaux et les références sont typés de manière spécifique en utilisant XSD

Service WEB – Publication et Découverte

Publication et Découverte



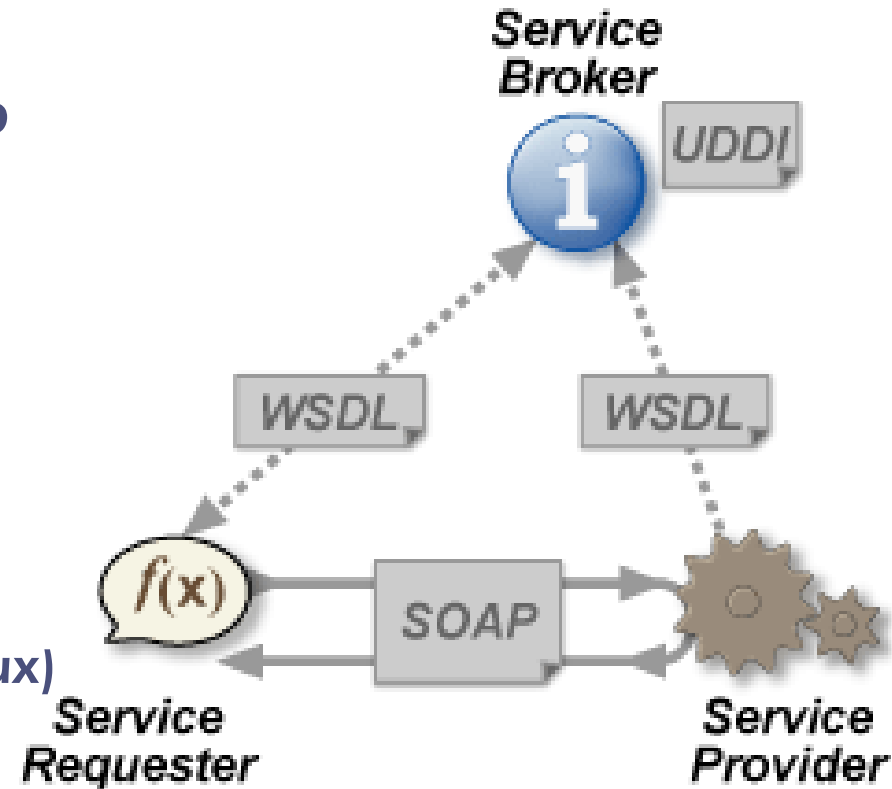
Langage de description d'un Web service REST

- ✓ **Essentiellement par**
 - **API documentée sur une page WEB**
 - **Par téléchargement d'une librairie pour le client**
- ✓ **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.



Langage de description d'un Web service SOAP: WSDL

- ✓ Un langage dérivé d'XML : Web Service Description Language
- ✓ Objectif
 - Interface publique d'accès à un Web Service
 - Comment communiquer pour utiliser le service (ensemble d'opérations et de messages abstraits reliés (bind) à des protocoles et des serveurs réseaux)
- ✓ Grammaire XML (schema XML)
 - Modulaire (import d'autres documents WSDL et XSD)
- ✓ Séparation entre la partie abstraite et concrète



Les concepts de WSDL 1.1

- ✓ **<types>**
 - Contient les définitions des types (utilise un système de typage comme XSD)
- ✓ **<message>**
 - Décrit les noms et types d'un ensemble de champs à transmettre
 - Paramètres d'une invocation, valeur du retour, ...
- ✓ **<portType>**
 - Décrit un ensemble d'opérations et les messages impliqués (0 ou 1 en entrée, 0 ou n en sortie). Partie la plus importante
- ✓ **<binding>**
 - Spécifie une liaison d'un <porttype> à un protocole concret (SOAP1.1, HTTP1.1, MIME, ...). Un portType peut avoir plusieurs liaisons !
- ✓ **<port>**
 - Spécifie un point d'entrée (endpoint) comme la combinaison d'un <binding> et d'une adresse réseau
- ✓ **<service>**
 - Pour agréger un ensemble de ports

<definitions>

<import>

<documentation>

<types>

<message>

<portType>

<binding>

<service>