

Sistemas Operativos II

Web Services

Web Services

- camada de software para facilitar a interação entre cliente e servidor, tornando-a mais rica e mais estruturada.
- incluem uma API que permite aceder a serviços remotos através da rede
- independentemente da linguagem das aplicações cliente e servidor, os pedidos e respostas são *usualmente* codificados numa **R**epresentação **E**xterna de **D**ados (RED) em **XML** e transmitidos sobre HTTP
 - Outra possível RED: **JSON**
- são identificados por um URI (URL ou URN)

Interface e formato das mensagens

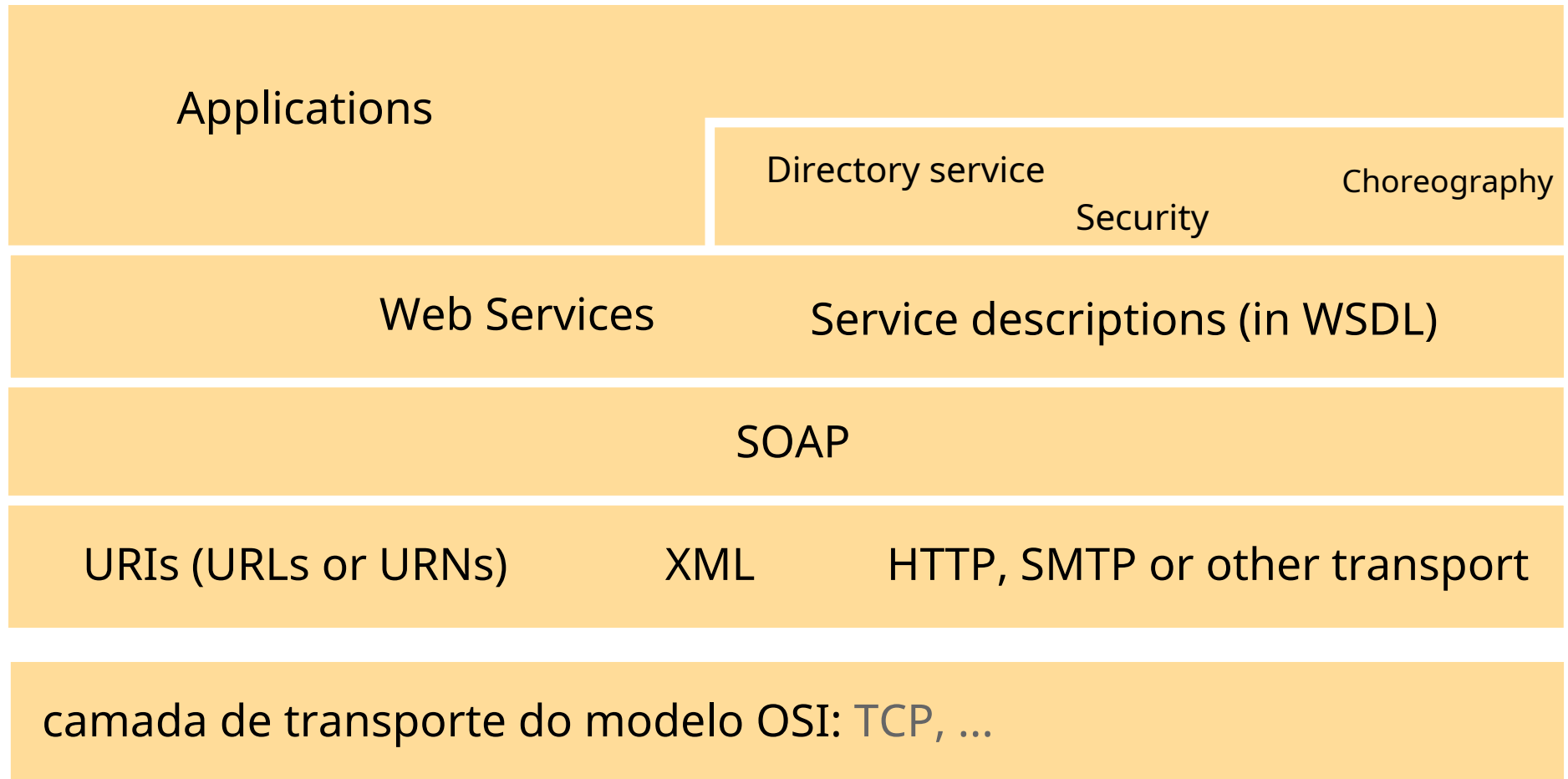
Service Description

- Tal como no CORBA e Java RMI, há uma descrição para a **interface** do web service. Há também a especificação do protocolo de codificação e comunicação das mensagens e a localização do Web Service (URL ou URN)
- A IDL usada é a *Web Services Description Language* (WSDL)

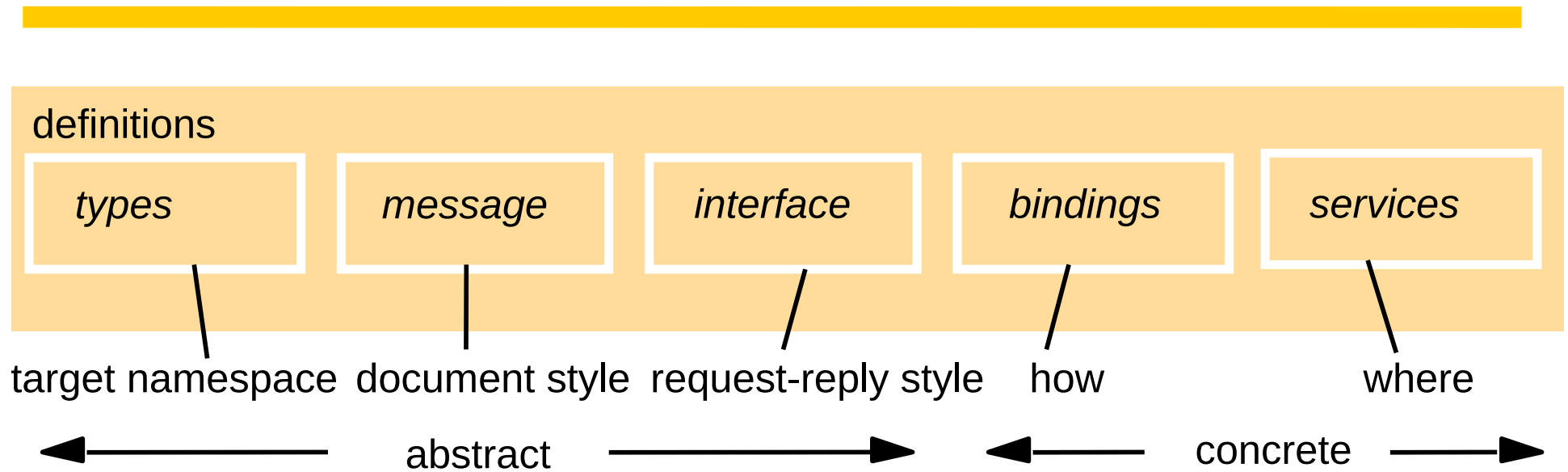
SOAP - Simple Object Access Protocol

- extensão de XML-RPC
- é um protocolo para troca de mensagens, usualmente sobre HTTP, que trata do correto encapsulamento dos dados em XML
- possíveis protocolos para envio das mensagens HTTP, SMTP, TCP, UDP
- não levanta problemas na presença de firewalls

Web Services: camada de software usada pelas aplicações



Principais componentes da *Service Description WSDL*



WSDL para Request e Reply para a operação newShape()

```
message name = "ShapeList_newShape"
```

```
part name = "GraphicalObject_1"  
  type = "ns:GraphicalObject "
```

```
message name = "ShapeList_newShapeResponse"
```

```
part name = "result"  
  type = "xsd:int"
```

tns – target namespace xsd – XML schema definitions

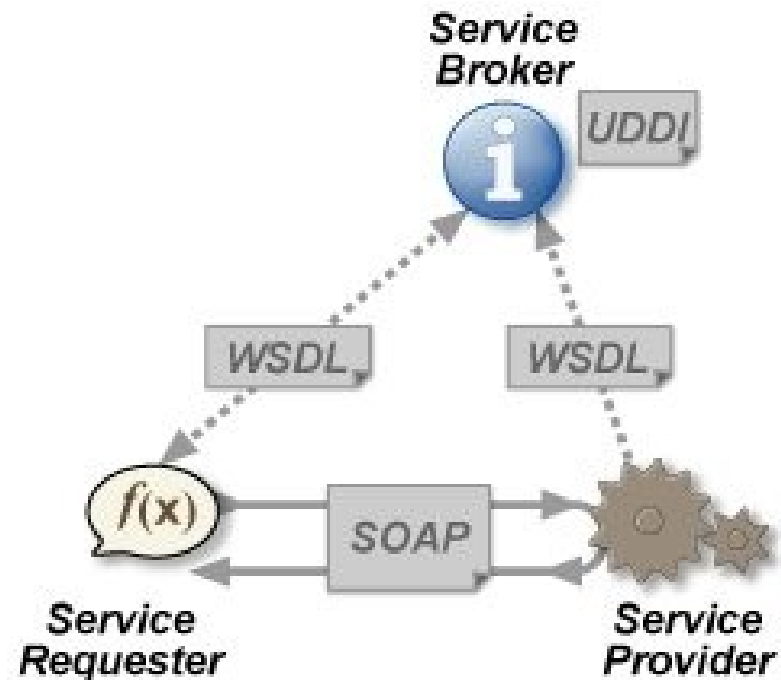
```
operation name = "newShape"  
  pattern = In-Out
```

```
input message = "tns:ShapeList_newShape"
```

```
output message = "tns:ShapeList_newShapeResponse"
```

Web Services: arquitetura

- tal como em Java RMI, o cliente do Web Service poderá consultar a descrição do serviço de nomes ou de diretoria
- **UDDI:** *Universal Description Discovery and Integration*
 - protocolo para publicar e pesquisar meta-informação sobre *web services*... permite que uma aplicação **descubra** e **use** um *web service* em *runtime*.



Web Services:

podem ser encadeados, formando um serviço mais complexo e abrangente

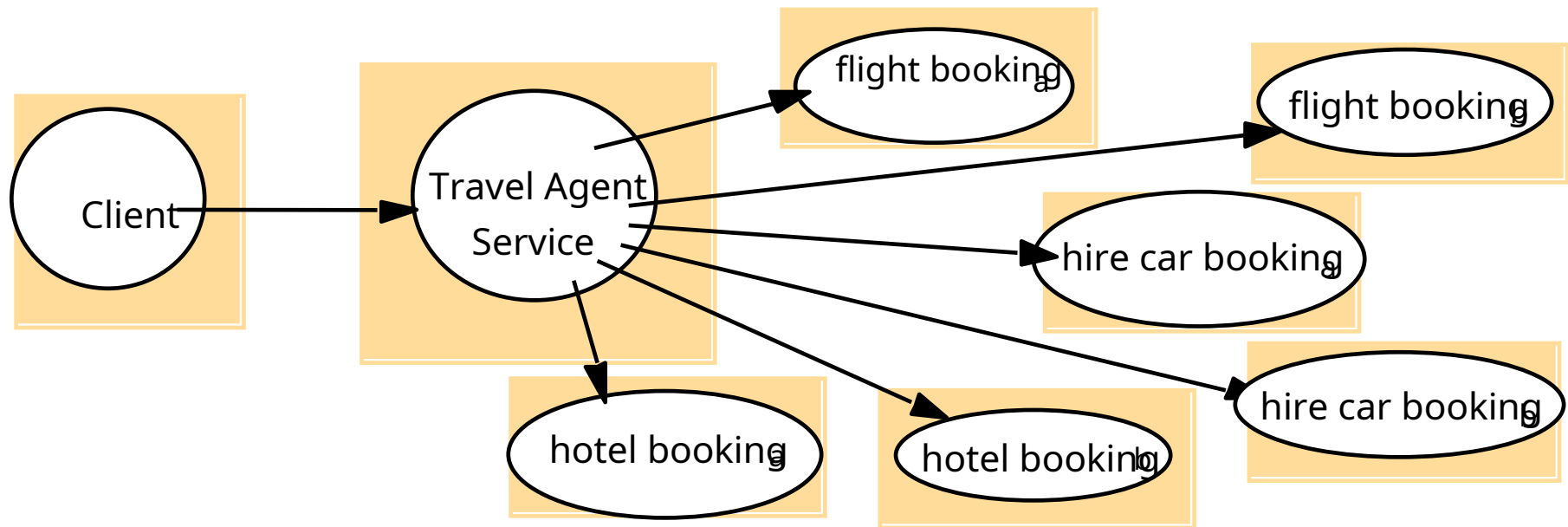


Figure 19.2 The 'travel agent service' combines other web services

Web Services

- servem de suporte à computação distribuída via internet, facilitando a cooperação de aplicações baseadas em diferentes linguagens
- os detalhes de SOAP e XML são usualmente escondidos por APIs (Java, Perl, Python, C++). A *service description* pode ser usada para gerar as rotinas de *marshalling* e *unmarshalling* de forma automática.
- diferença relativamente ao Object Model Distribuído: um *web service* é assegurado por um único objeto
 - o garbage collection neste caso não é tão crítico, porque não expõe obj a referências remotas
 - a referência remota para o objeto não é relevante (ele é o único associado ao serviço)
 - Para lá do Web Service, podem existir muitos objetos, para funcionalidade de apoio...
- Uso de XML em SOAP e nos dados:
 - vantagem: mais legível, por humanos
 - desvantagem: processamento mais lento que formatos binários

Figure 19.3 SOAP message in an envelope

- cada mensagem é colocada num *envelope* SOAP
- mensagem SOAP pode servir para:
 - | enviar uma notificação
 - comunicação cliente-servidor (protocolo Request-Reply)

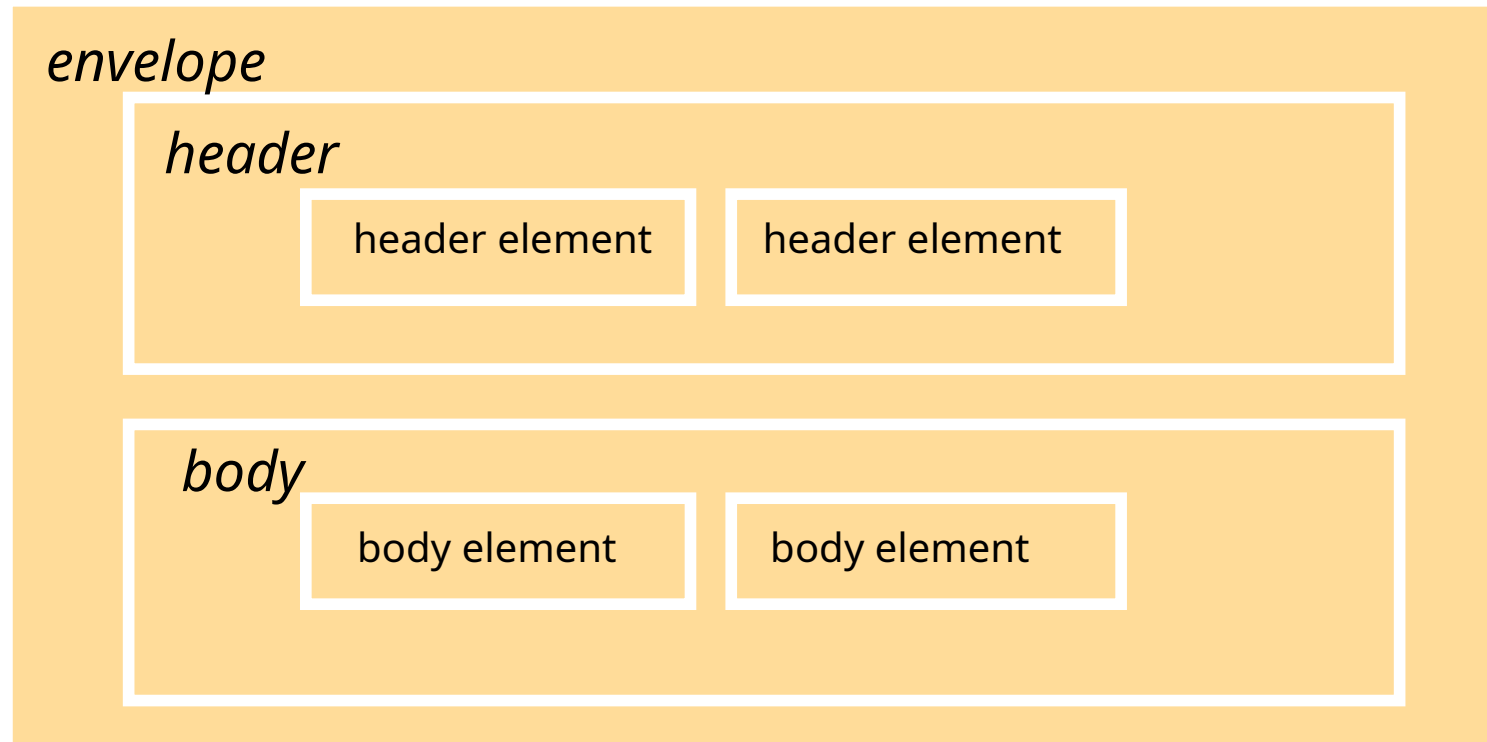
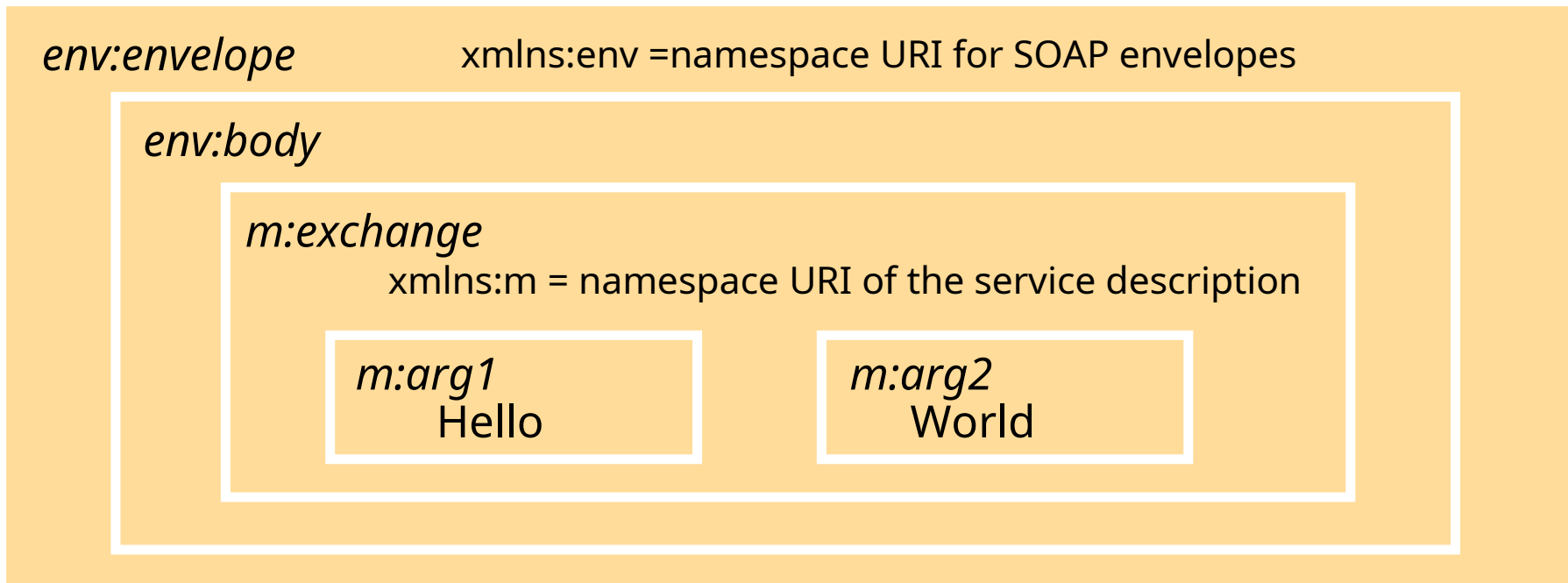


Figure 19.4 Example of a simple request (without headers)



each XML element is represented by a shaded box with its name in italic followed by any attributes and its content

Note-se a definição de XML Namespaces (env e m)
Os SOAP Headers destinam-se a essencialmente a componentes de software intermediários...

Figure 19.5 Example of a reply corresponding to the request in Figure 19.4

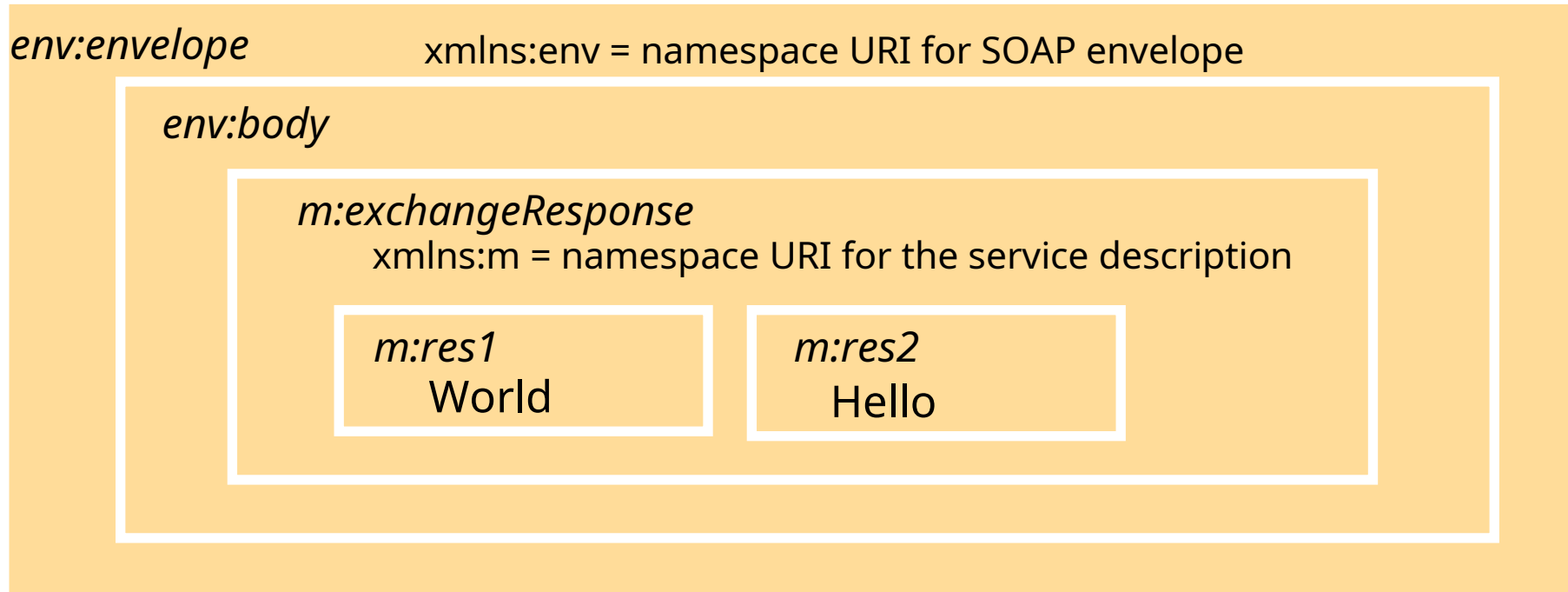
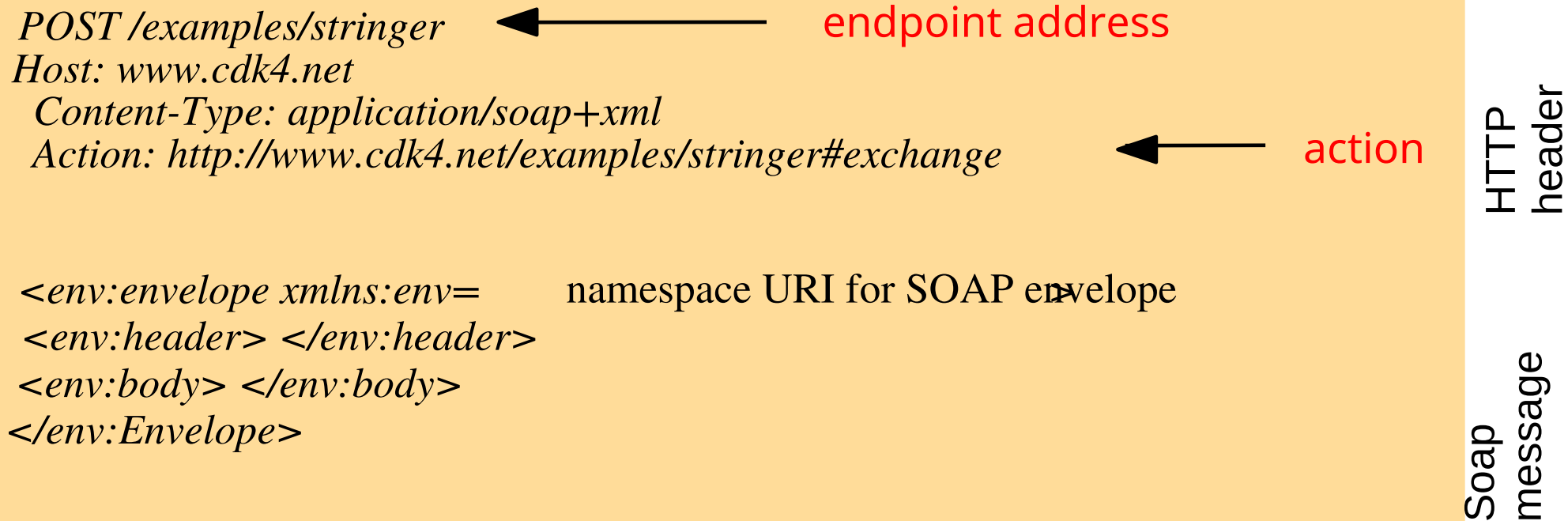


Figure 19.6 Use of HTTP POST Request in SOAP client-server communication



SOAP em Java

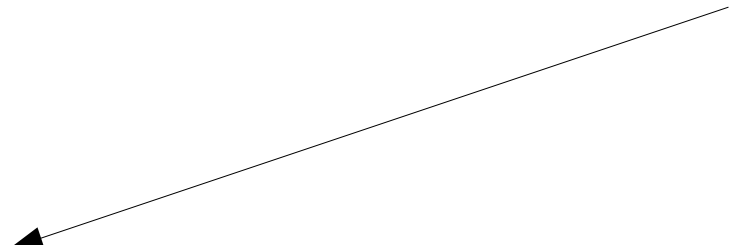
- **JAX-RPC** é uma API que permite construir Web Services e respectivos clientes
- **JAX-WS** significa Java API for XML Web Services e inclui:
 - **JAX-WS** 2.0 - é uma extensão do JAX-RPC 1.0.
 - Java Architecture for XML Binding (JAXB)
 - SOAP with Attachments API for Java (SAAJ)
 - ... faz o mapeamento entre tipos Java e definições XML a usar nas mensagens
- A API do JAX-WS esconde a complexidade do protocolo SOAP
 - O programador não precisa gerar ou fazer o parse explícito de mensagens SOAP
- Web Service **Endpoint** é a implementação do Web Service
 - Fica no servidor

SOAP em Java

- JAX-RPC e JAX-WS têm ferramentas de desenvolvimento de Web Services com algumas diferenças
- Por exemplo para o JAX-WS, a implementação do *endpoint* deve obedecer a:
 - It **must** carry a `javax.jws.WebService` annotation
 - It **may** extend `java.rmi.Remote` either directly or indirectly.
 - Any of its methods **may** carry a `javax.jws.WebMethod` annotation
 - All of its methods **may** throw `java.rmi.RemoteException` in addition to any service-specific exceptions.
 - All method parameters and return types **must be** compatible with the JAXB 2.0 Java to XML Schema mapping definition.
 - The implementing class **must not** be declared `final` and must not be abstract.
 - The business methods of the implementing class **must be** public, and **must not** be declared static or final.
 - Business methods that are exposed to web service clients **must be** annotated with `javax.jws.WebMethod`.
 - A method **parameter** or **return value** type **must not** implement the `java.rmi.Remote` interface either directly or indirectly **
- ** - idem para JAX-RPC, onde as classes desses argumentos ou resultado devem possuir um construtor por defeito, público.

Figure 19.7 Java web service interface ShapeList

Exemplo do Livro, para [JAX-RPC](#). [Aqui](#), isto é importante



```
import java.rmi.*;  
public interface ShapeList extends Remote {  
    int newShape(GraphicalObject g) throws RemoteException;  
    int numberOfShapes()throws RemoteException;  
    int getVersion() throws RemoteException;  
    int getGOVersion(int i)throws RemoteException;  
    GraphicalObject getAllState(int i) throws RemoteException;  
}
```


Figure 19.8 Java implementation of the ShapeList “server”

```
import java.util.Vector;

public class ShapeListImpl implements ShapeList {
    private Vector theList = new Vector();
    private int version = 0;
    private Vector theVersions = new Vector();

    public int newShape(GraphicalObject g) throws RemoteException{
        version++;
        theList.addElement(g);
        theVersions.addElement(new Integer(version));
        return theList.size();
    }

    public int numberOfShapes(){}
    public int getVersion() {}
    public int getGOVersion(int i){ }
    public GraphicalObject getAllState(int i) {}
}
```

Figure 19.9 Java implementation of the ShapeList client

```
package staticstub;
import javax.xml.rpc.Stub;

public class ShapeListClient {
    public static void main(String[] args) { /* pass URL of service */
        try {
            Stub proxy = createProxy();
            proxy._setProperty
                (javax.xml.rpc.Stub.ENDPOINT_ADDRESS_PROPERTY, args[0]);
            ShapeList aShapeList = (ShapeList)proxy;
            GraphicalObject g = aShapeList.getAllState(0);
        } catch (Exception ex) { ex.printStackTrace(); }
    }

    private static Stub createProxy() {
        return
            (Stub) (new MyShapeListService_Impl().getShapeListPort());
    }
}
```

Web Services em Java: Servidor e descrição

Assumindo que se começa por desenvolver a *Service Interface* e a sua implementação (*endpoint*)

- há ferramentas para gerar automaticamente o Skeleton e a descrição do serviço em WSDL
- o serviço vai correr num Servlet Container (exemplo: Apache Tomcat)
- Depois de preparado, o web service deve estar num ficheiro “.war”, que pode ser *deployed* (instalado) no *Servlet Container*
 - o dispatcher do *servlet container* identifica a operação no pedido
 - pelo header http Action
 - e invoca o método apropriado no respetivo Skeleton
- É possível gerar a classe do Proxy, para o cliente, em runtime, a partir da *service description*... mas não é sempre assim.

Segurança no XML

funcionalidade para assinar, cifrar e manipular chaves

- Para determinadas garantias de segurança, não basta proteger o canal... o próprio documento tem de incorporar alguma meta-informação (assinatura, informação de chaves...)
- para ser validada posteriormente à cessação do canal de comunicação
- o XML permite várias formas sintáticas para os mesmos dados
- a assinatura digital do XML é precedida de uma conversão para *Canonical XML*
 - *representação normalizada e serializada de XML*

Figure 19.16 Algorithms required for XML signature

<i>Type of algorithm</i>	<i>Name of algorithm</i>	<i>Required</i>	<i>reference</i>
Message digest	SHA-1	Required	Section 7.4.3
Encoding	base64	Required	[Freed and Borenstein 1996]
Signature	DSA with SHA-1	Required	[NIST 1994]
(asymmetric)	RSA with SHA-1	Recommended	Section 7.3.2
MAC signature (symmetric)	HMAC-SHA-1	Required	Section 7.4.2 and Krawczyk <i>et al.</i> [1997]
Canonicalization	Canonical XML	Required	Page 810

Figure 19.17 Algorithms required for encryption (the algorithms in Figure 19.16 are also required)

<i>Type of algorithm</i>	<i>Name of algorithm</i>	<i>Required</i>	<i>reference</i>
Block cipher	TRIPLEDES,	required	Section 7.3.1
	AES 128		
	AES-256		
	AES-192	optional	
Encoding	base64	required	[Freed and Borenstein 1996]
Key transport	RSA-v1.5,	required	Section 7.3.2 [Kaliski and Staddon 1998]
	RSA-OAEP		
Symmetric key wrap (signature by shared key)	TRIPLEDES	required	[Housley 2002]
	KeyWrap,		
	AES-128 KeyWrap,		
	AES 256KeyWrap		
	AES-192 KeyWrap	optional	
Key agreement	Diffie-Hellman	optional	[Rescorla, 1999]

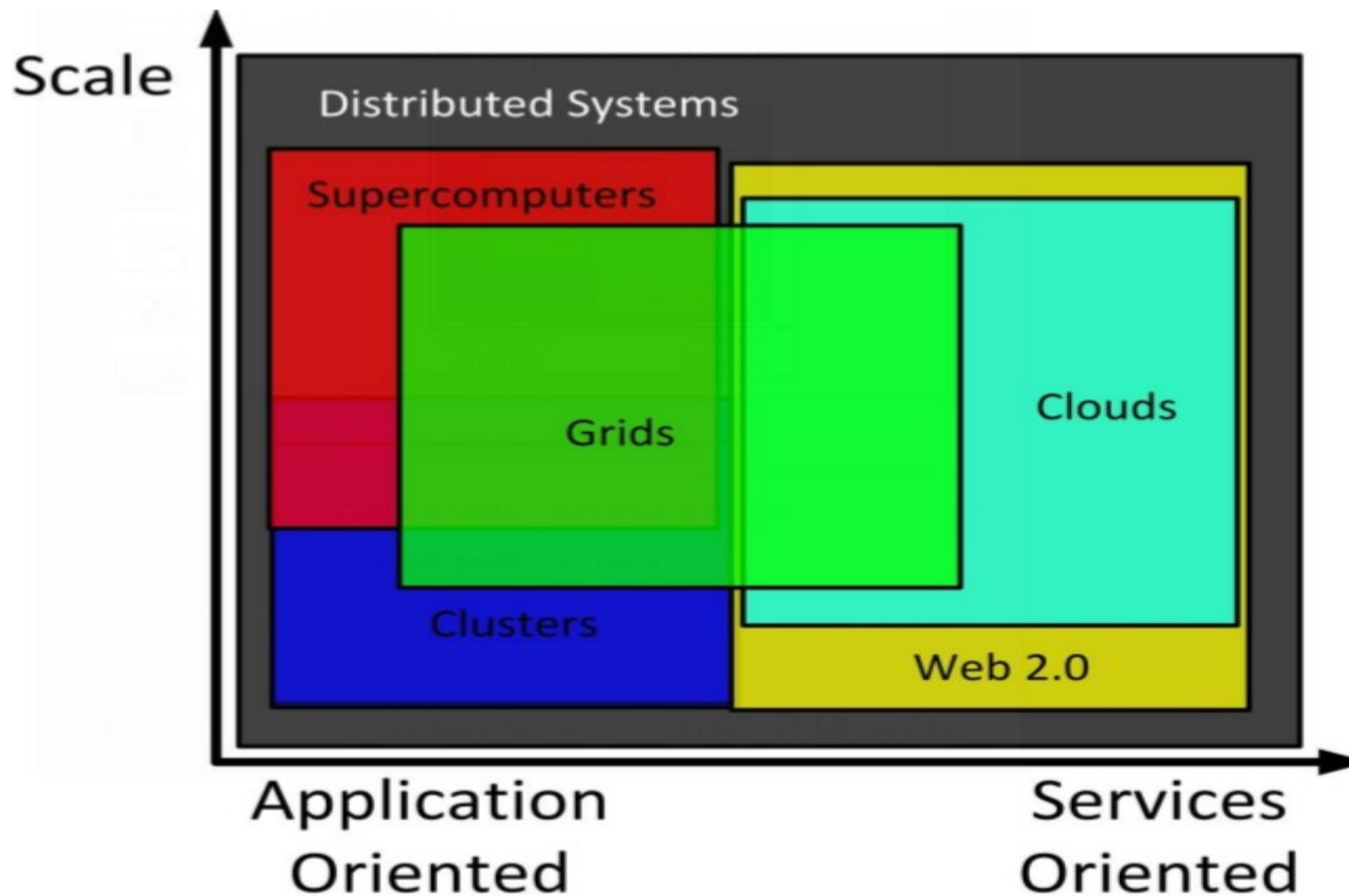
Alguns Web Services da Amazon

<i>Web service</i>	<i>Description</i>
Amazon Elastic Compute Cloud (EC2)	Web-based service offering access to virtual machines of a given performance and storage capacity
Amazon Simple Storage Service (S3)	Web-based storage service for unstructured data
Amazon Simple DB	Web-based storage service for querying structured data
Amazon Simple Queue Service (SQS)	Hosted service supporting message queuing (as discussed in Chapter 6)
Amazon Elastic MapReduce	Web-based service for distributed computation using the MapReduce model (introduced in Chapter 21)
Amazon Flexible Payments Service (FPS)	Web-based service supporting electronic payments

GRID Computing

- *middleware* desenhado para permitir e otimizar a partilha de recursos (computadores, dados, sensores, software) em larga escala
- usualmente os utilizadores destes sistemas (cientistas, engenheiros) colaboram para alcançar um objetivo comum, como um estudo que requer o processamento de grandes quantidades de dados, por exemplo.
- os recursos estão alojados em computadores de diferentes plataformas, com ambiente heterogéneo
- o *middleware* **GRID** pode assentar em **Web Services**

Grid & Cloud



Grid & Cloud

Grid

- Usualmente heterogêneas (mas não necessariamente)
- Lidar com enorme volume de dados ou processamentos complexos
- Geograficamente dispersas
- Exemplos
 - Open Science Grid (OSG) - <http://www.opensciencegrid.org/>
 - LHC Computing Grid (CERN)
 - Middleware: Globus Toolkit

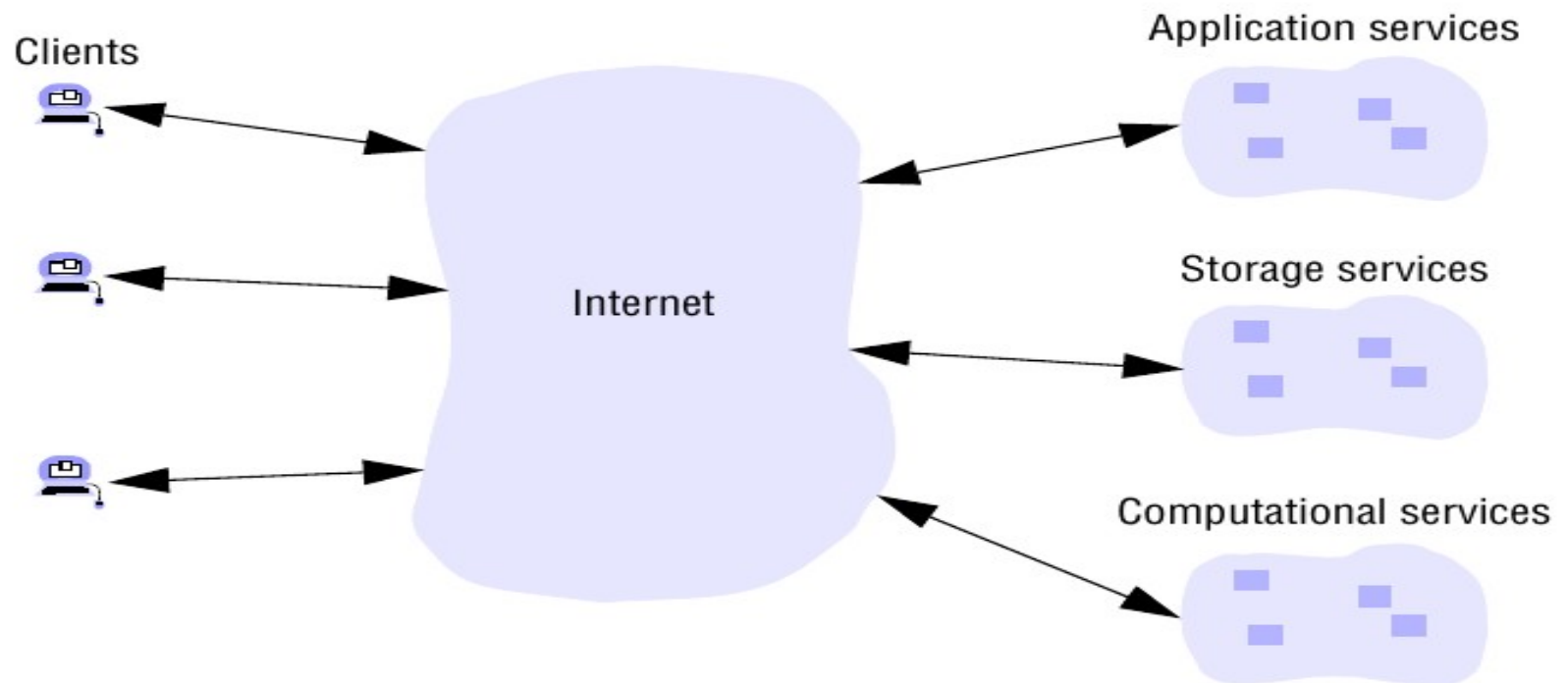
Cloud

- Paradigma “recente” de computação distribuída de larga escala
 - Motivações:
 - Uso mais geral (muitos ou poucos dados); maior leque de utilizadores
 - vocação para os serviços
 - **Virtualização**
 - Alocação de recursos dinâmica em função das necessidades
 - Exemplos:
 - Amazon, GoogleApps, Windows Azure

Cloud Computing

Cloud é um serviço de armazenamento e/ou computação baseado na Internet. Reduz a necessidade de armazenamento, software e capacidade de processamento do lado dos terminais junto do utilizador.

Cloud computing



Arquitetura REST

- **REST** - *Representational State Transfer*
 - É uma arquitetura para interação em sistemas distribuídos
 - Surgiu em 2000, da tese de doutoramento de Roy Thomas Fielding
- Objetos têm estado, cuja representação é transportada por pedidos HTTP
 - PUT; POST; GET; DELETE
 - Cada pedido tem um significado próprio (criar, alterar, consultar e apagar objetos)
- Em comparação com os SOAP based Web Services:
 - Mais leve
 - Ainda intelegível
 - Formato dos dados pode ser JSON ou XML (mantém flexibilidade)

Arquitetura REST

- Analogia com SOAP:
 - mensagem de consulta de dados sobre o utilizador nº 12345

- SOAP WS:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:body pb="http://www.acme.com/phonebook">
    <pb:GetUserDetails>
      <pb:UserID>12345</pb:UserID>
    </pb:GetUserDetails>
  </soap:Body>
</soap:Envelope>
```

- REST:

```
http://www.acme.com/phonebook/UserDetails/12345
```

- O url inclui os parâmetros da consulta