B/S体系软件设计

SOA and REST

胡晓军

SOA

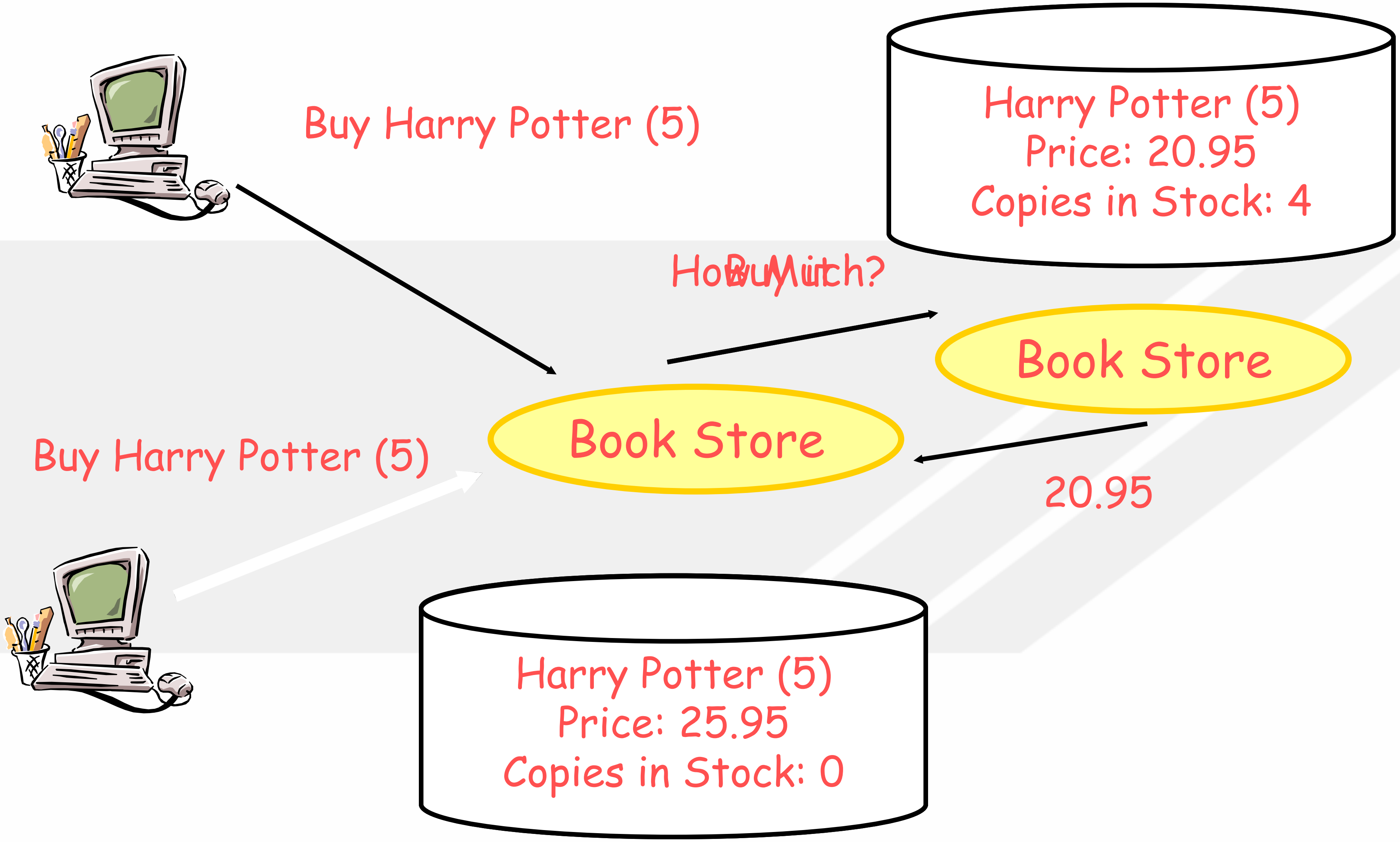
- Service Oriented Architecture，面向服务架构
- SOA是一种架构模型，与具体的技术无关
- SOA是基于组件技术的架构，将应用程序的不同功能单元（称为服务）通过这些服务之间定义良好的接口联系起来

- SOA是一种范式、概念、方法，主要是针对异构的大型分布式系统
- Web Service是实现SOA的重要技术

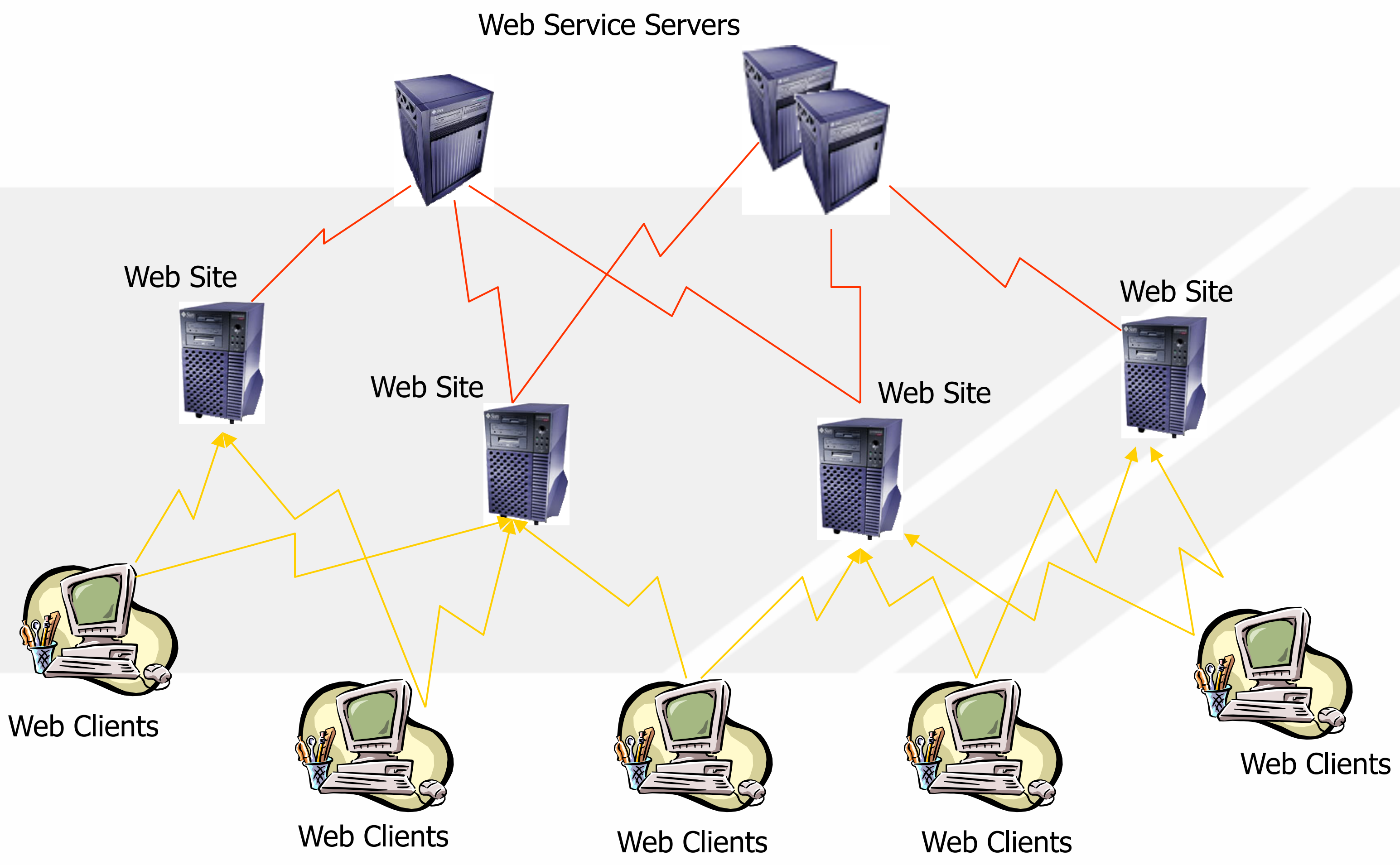
Web Service

- A unit of application logic that provide data and services to other applications
- Loosely coupled, stateless connectivity for distributed computing
- Generally transmitted over HTTP
- Uses entirely open standards driven by the key players (MS, IBM, …)
- Website – Application to User
- Web Service – Application to Application

Example Scenario



Web Service


Web Service Servers
Web Site
Web Site
Web Site
Web Site
Web Clients
Web Clients
Web Clients
Web Clients
Web Clients

Web Services Overview 1/2

- For the most part, similar to COM programming
- Based on simple, open standards
  - XML-based communication
- Communication = Messaging
- Client and Web Service are "loosely coupled"
- URL—the key to Web Services

```
http://<serverName>/<VirtualDir>/
  <fileName>/<methodName>?var=value
```

Web Services Overview 2/2

- Web Service Wire Formats
  - HTTP: GET and POST
  - SOAP
- Web Services Description Language (WSDL)
  - XML-based
  - Abstract description of the Web Service
- Transactions
  - ASP.NET transactions = COM+ transactions

The Standards

- XML
  - Data is sent around in XML because it is schema based and Founding basis of web services
  - Hierarchical structured data passed to and from the web services
- SOAP – Simple object access protocol
  - Communication Protocol for web services
  - is a W3C standard, present version is 1.0
  - Started as a light-weight way of accessing objects remotely. Remote procedure calls but at the object level, rather than procedure
  - It’s XML with a particular Schema that defines SOAP

The Standards

WSDL – Web Service Definition Language

- The way in which the Web Service's methods are defined. It defines what methods are available, what the parameters are (and their types) and the return type and its type
- It's XML with a particular schema that defines it as WSDL
- It also describes the ways in which the web service can be accessed
  - SOAP only, or SOAP and…
  - HTTP Get
  - HTTP Post
- Proxy classes are built by VS.NET for any WSDL you want

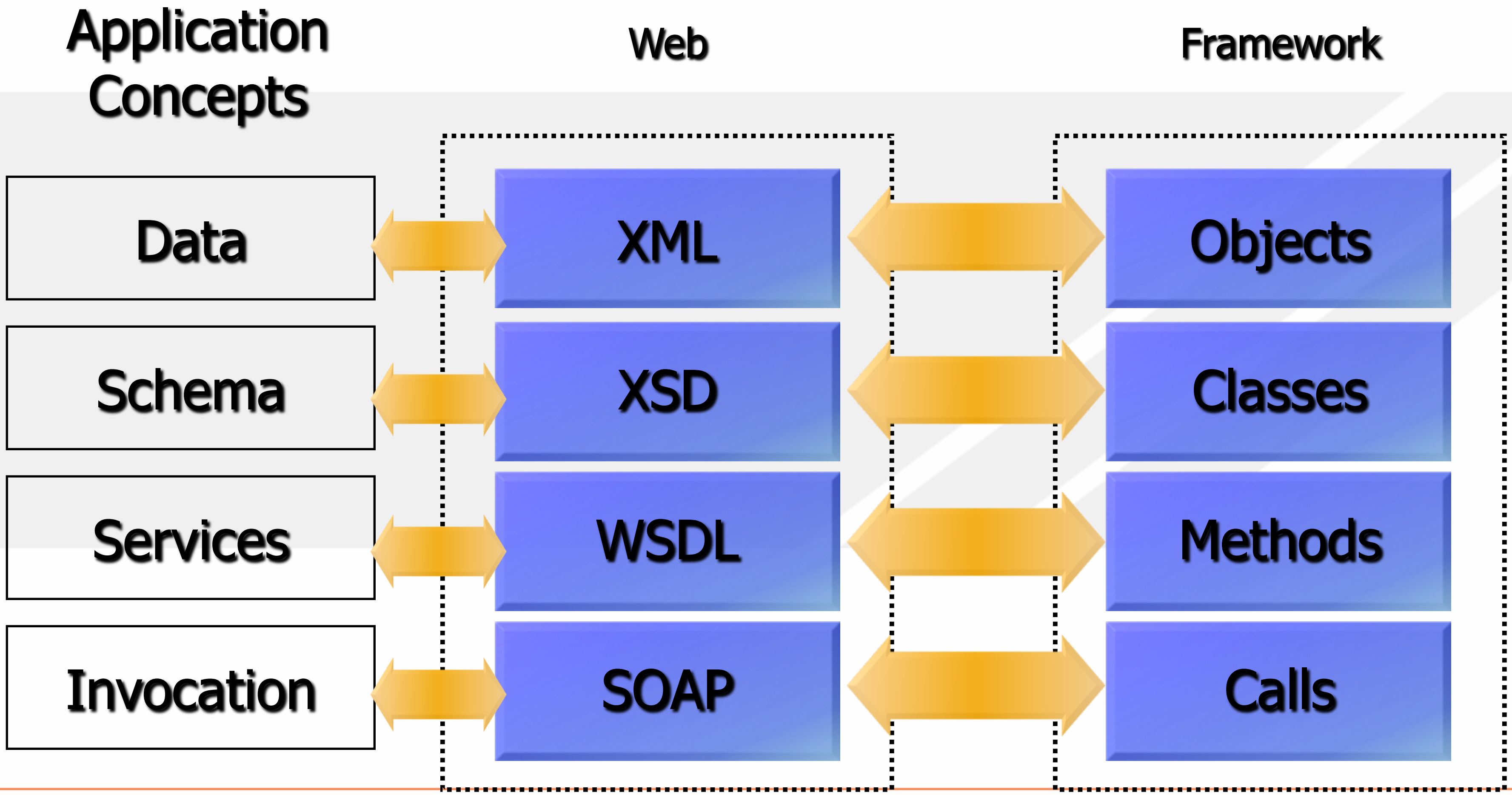
The Standards

- UDDI – Universal Discovery, Description and Integration
  - An open standard for finding web services, like a yellow-pages standard
  - We won't be using UDDI for anything as it is an Enterprise level thing

  - Microsoft has VSDisco in VS.NET shipped because UDDI wasn't agreed to when VS.NET was finished. It's turned off by default.

Mapping the protocols

The .NET Framework provides
a bi-directional mapping

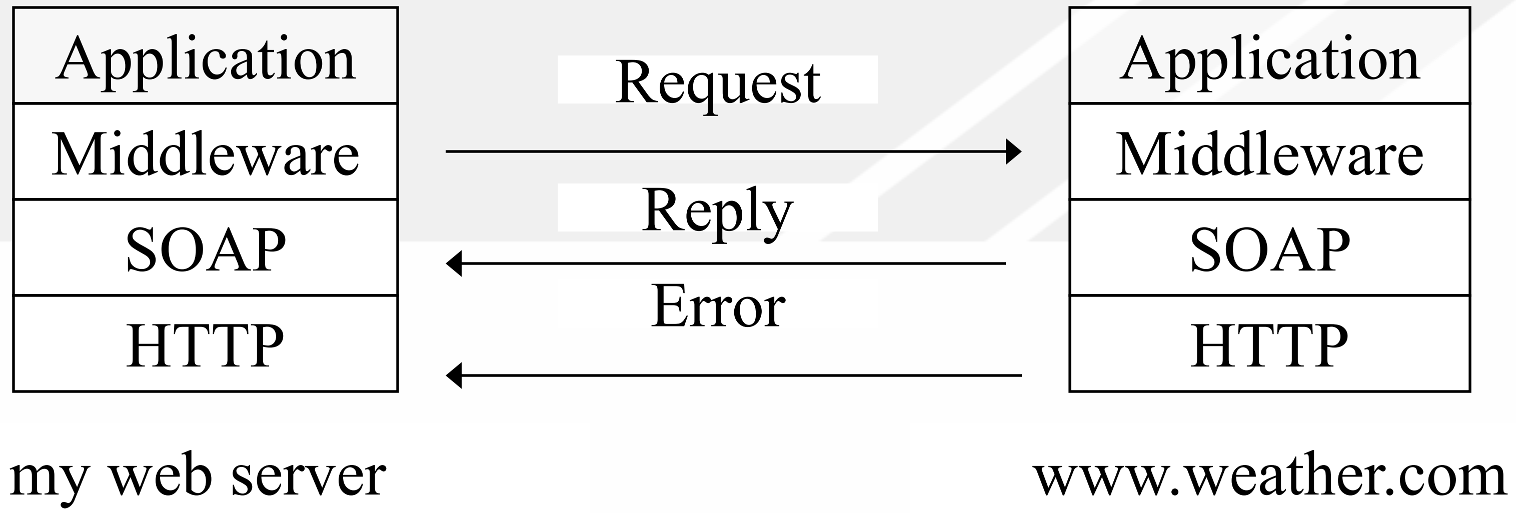


SOAP example

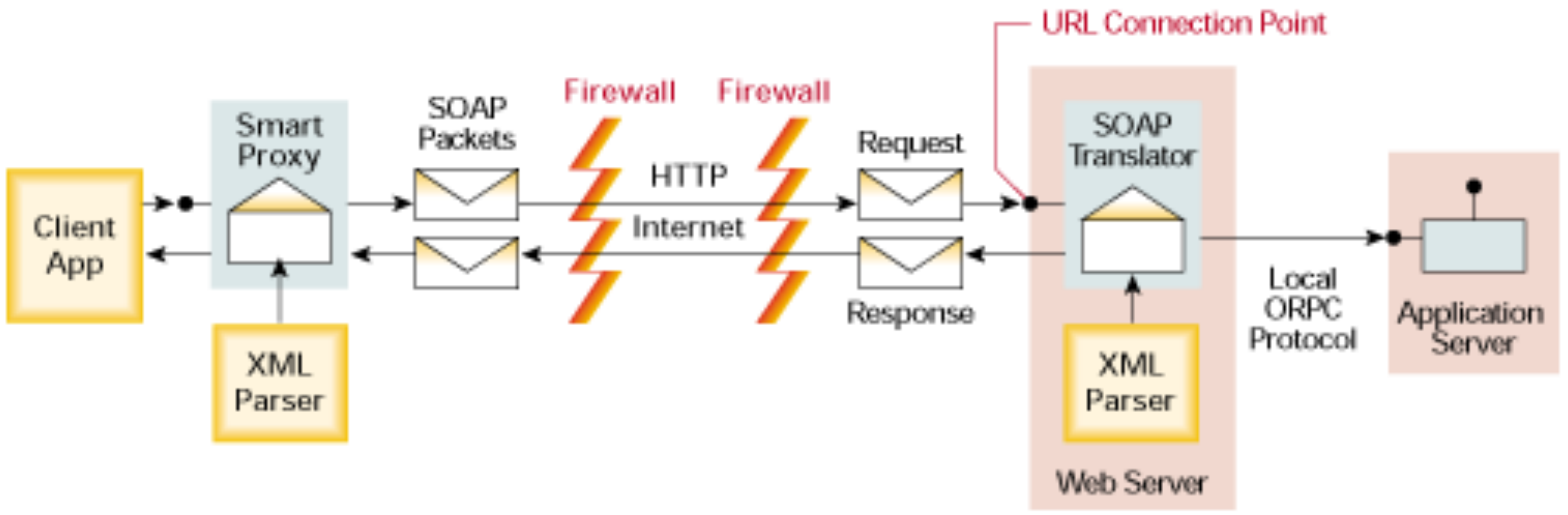
www.weather.com

float CurrentTemp(zip_code)

The process



SOAP Intuition



Request Example

```
POST /Temperature HTTP/1.1
Host: www.weather.com
Content-Type: text/xml
Content-Length: <whatever>
SOAPMethodName: <some-URI>#CurrentTemp

<SOAP:Envelope xmlns:SOAP="urn:schemas-xmlsoap-
   org:soap.v1">
   <SOAP:Body>
       <m:CurrentTemp xmlns:m="some-URI">
             <zip_code>37919</zip_code>
       <m:CurrentTemp>
   </SOAP:BODY>
<SOAP:Envelope>
```

Http Header

Soap Extensions

Xml Payload

URI- Uniform Resource Identifier

some-URI -> www.netsolve.com or www.globus.com

Response Example

```
HTTP/1.1 200 OK
Content-Type: text/xml
Content-Length: <whatever>

<SOAP:Envelope xmlns:SOAP="urn:schemas-xmlsoap-
   org:soap.v1">
   <SOAP:Header>
        <t:Transaction xmlns:t="some-URI">
              5
        </t:Transaction>
   </SOAP:Header>
   <SOAP:Body>
        <m:CurrentTempResponse xmlns:m="some-URI">
              <return>42</return>
        </m:CurrentTempResponse>
   </SOAP:Body>
</SOAP:Envelope>
```

Http Header

Xml Payload

SOAP: What it's NOT

- Object Activation
  - who invokes CurrentTemp function?
- Bi-directional Communications
- Distributed Garbage Collection
- Language Bindings unspecified
  - good for interoperability
  - Perl,C,java
  - Source of xml-rpc payload is immaterial

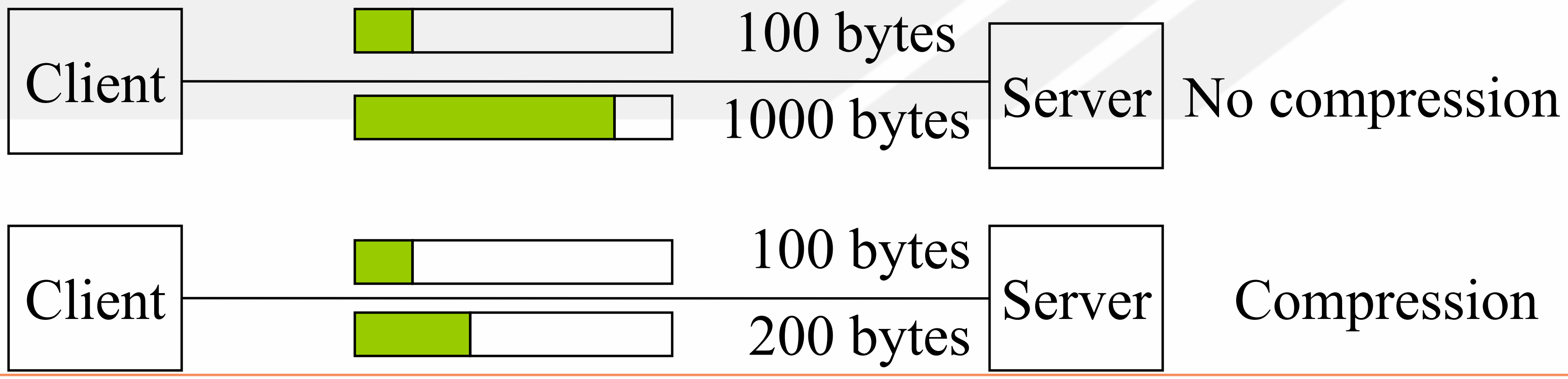
SOAP: Analysis

Size of Messages

Latency is key, not Bandwidth

High factors of Compression

Ascii + repetitive Tags



SOAP:Analysis

$$\begin{tikzcd}
\text{In-memory} \arrow[r] & \boxed{\text{encode}} \arrow[r] & \text{xml-rpc} \arrow[r] & \boxed{\text{decode}} \arrow[r] & \text{In-memory}
\end{tikzcd}$$

Strings - no conversion needed

Floating Point - sprintf,sscanf

“e-commerce” applications --- GOOD

text + integers

“grid” applications --- BAD

numeric intensive

There is always a tradeoff involved

SOAP: Analysis

Stateless Nature - Independent transactions

Most distributed applications are stateful

Programming model is different

State info with every transaction

Size of state info need

  “A cookie may not satisfy hunger”!!

Good for scalability

SOAP: Analysis

- Programming complexity
- Standards are in flux
- Maturity of tools
- Need open-source xml parsers
- xml.apache.org
  - xerces : parsers in xml,perl,c++
- A minor obstacle at best

SOAP: What can it teach us?

- Use XML for data exchange
  - can define our own xml-rpc if needed
  - the idea of encoding is what is important
  - can use TCP as transport
- HTTP tunneling
- Would be a short-sight on our part to ignore because of Microsoft tag

SOAP: Summary

Its not something path-breaking

“Right mix of technology at the right time”

Structure more important than content

XML - ASCII of the future

Holds lot of promise

Step in the right direction

Web Service的应用场合

面向应用集成

内部同构程序不建议使用**WebService**

考虑性能时不建议使用**WebService**

REST

- **REST(*Representational State Transfer*):**表述性状态转移，是一种架构风格。
- **Roy Thomas Fielding**，博士论文中提出
  - **Architectural Styles and the Design of Network-based Software Architectures**
- 一套简单的设计原则、一种架构风格**(**或模式**)**，不是一种具体的标准或架构。
- 基于**HTTP**、**URI**
- 无状态

REST设计原则

- 网络上的所有事物都被抽象为资源
  - 资源是数据与表现形式(***Representational***)的结合
- 每个资源对应一个唯一的资源标识**URI**
- 通过**HTTP**协议方法作连接器对资源进行操作
- 对资源的任何操作不改变资源标识**URI**
- 所有的服务器操作都是无状态的

无状态

- 使得连接器无需保存请求之间的应用状态，从而降低了物理资源的消耗并改善了可伸缩性
- 允许对交互进行并行处理，处理机制无需理解交互的语义
- 允许中间组件孤立地查看并理解一个请求，当需要对服务作出动态安排时，这是必需要满足的
- 强制每个请求都必须包含可能会影响到一个已缓存响应的可重用性的所有信息。

REST操作

- CRUD，针对Object
  - 获取资源的一个表示：HTTP GET
  - 创建一个新资源：HTTP POST
  - 修改已有资源：HTTP PUT
  - 修改部分资源：HTTP PATCH
  - 删除已有资源：HTTP DELETE
- 安全性与幂等性
  - GET请求是安全的
  - GET、PUT和DELETE请求是幂等的

如何设计好的Restful API

- CURD + 名词
  - /getAllUsersvs GET /users
- login/logout
  - POST /session
  - DELETE /session?id=xxx
- url层级
  - /users/1/posts vs /posts?userId=1
- Web Server只支持GET和POST