WWW технология

Судаков А.А.

“Параллельные и распределенные вычисления” Лекция 6

План

- Исторические сведения
- Клиенты и серверы
- Протоколы
- Серверы приложений
- WWW службы
- SOAP, XML RPC

Литература

- http://www.w3c.org
- http://hercules.xml.org/xml/resources_cover.shtml
- http://www.iks.inf.ethz.ch/education/ws04

Исторические сведения

- Сегодня WWW – «синоним» Интернет и распределенных приложений в пределах Интернет
- Начало 80-х годов создана в CERN для обмена документами в пределах Интернет
- 1990 г. язык HTML, позже XML
- В 1993 г SUN анонсировал Java – «платформенно-независимая» технология для создания программ, которая стала основой для разработки WWW приложений
- Конец 90-х стандарты для удаленных вызовов процедур через Интернет (SOAP, XML RPC)

Недостатки классических RPC

- Рассчитывались на локальные сети с большой скоростью доступа и малыми задержками
- Не рассчитывались на одновременное обслуживание большого количества клиентов
- Существует несколько несовместимых реализаций
- Разработка сложна, требует высокой квалификации специалистов

WWW

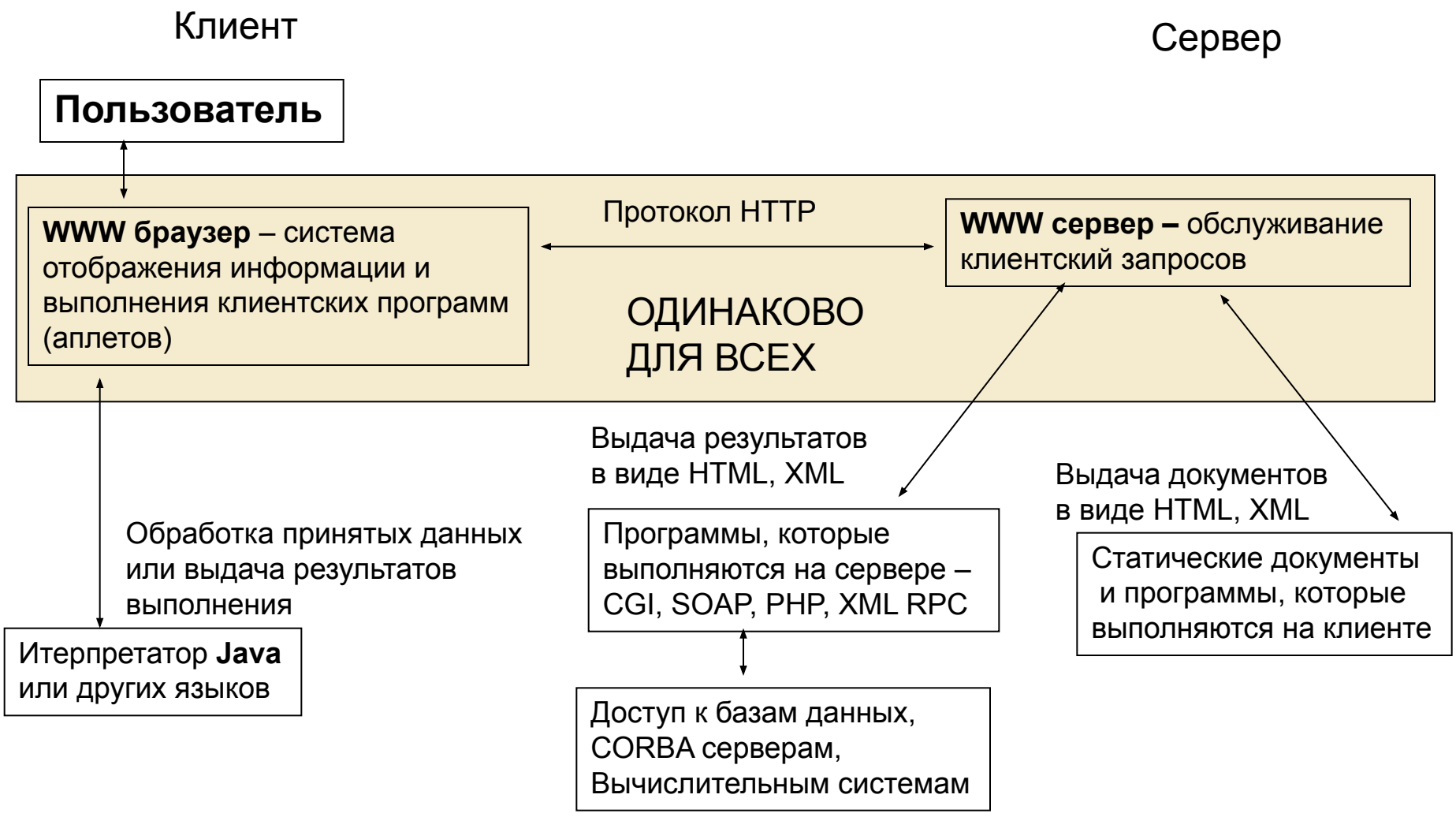
- WWW – всемирная паутина
- Большое количество серверов и клиентов, которые работаю с использованием общих протоколов передачи данных (http)
- Каждый ресурс (документ, программа) идентифицируется уникальным идентификатором URI – uniform resource identification

URI

- URL – locator
- URN - name

ftp://ftp.is.co.za/rfc/rfc1808.txt
http://www.ietf.org/rfc/rfc2396.txt
ldap://[2001:db8::7]/c=GB?objectClass?one
mailto:John.Doe@example.com
news:comp.infosystems.www.servers.unix
tel:+1-816-555-1212
telnet://192.0.2.16:80/

Взаимодействие клиентов и серверов



Протоколы и сериализация

- Транспортные протоколы – TCP/IP
- Представление данных – HTTP hyper text transfer protocol
  - Данные передаются (как правило) в текстовом виде (HTML, XML)
  - Сериализация и десериализация выполняются с помощью HTML, XML
  - Могут использоваться и другие протоколы
- Уровень прикладных программ
  - универсальный интерфейс для представления информации – WWW браузер
  - Логика работы разделена между сервером (www приложение, сервлет) и клиентом (аплеты)

Совместимость

- Для совместимости предпочтительно использовать текстовые форматы данных
  - Данные в таком формате всем людям понятны
    - 100000
    - “This is a pen”
  - То же в шестнадцатиричном формате
    - 186A0
    - 54 68 69 73 20 69 73 20 61 20 70 65 6E
- Во втором случае не зная формата трудно определить, что написано
- В текстовом виде это занимает больше места, но его легко может понять человек и просто обработать программно

HTML

- Таговая (tag, дескриптор) структура
  - Разбивка потока байтов на блоки, каждый со своим типом интерпретации
- Блок выделяется «таговыми скобками»
  - Открывающий дескриптор
  - Содержимое блоки
  - Закрывающий дескриптор
- Существует стандартный набор дескрипторов для описания форматирования документов

Пример HTML

```
<!doctype html public "-//W3C//DTD HTML 4.0 Transitional//EN">
<html >
 <head> <!—Інформація заголовка -->
 <base …> <!—базовий URL даного документа (не обов'язково) -->
 <title> Назва документа </title> <!—обов'язково-->
 <link …> <!--.зв'язок даного документа з іншими (не обов'язково) -->
 <meta …> <!—Інформація для броузера, пошукова і т. д. (не обов'язково)-->
 </head>
 <body>
 <!--…Основний вміст документа…-->
<TABLE параметри >
      <TR параметри >   <!— новий рядок таблиці-->
      <TD параметри >…</TD>        <!— елемент рядку-->
      <TD параметри >…</TD>        <!— елемент рядку-->
      . . .
      </TR><!— закінчення рядку -->
      <TR параметри >   <!— новий рядок -->
       . . .
      </TR>
</TABLE>
 </body>
</html>
```

Языки гипертекстовой разметки

- SGML (Standard Generalized Markup Language) – 1960-е года (IBM)
  - широко используется в промышленности
  - Общее описание языков гипертекстовой разметки
- HTML – 1980-1990 года (CERN)
  - Подмножество SGML
  - Используется для форматирования текста
- XML – 1990-е года (w3c)
  - Язык описания структуры информации с возможностью создания своих типов дескрипторов
  - Универсальное средство описания данных любой структуры

XML

- Extensible Markup Language
  - В отличие от HTML рассчитан не на то, как форматировать данные при выводе, а на то, чтобы описать назначение блока данных
  - Это обеспечивается возможностью введения описания новых тагов
- HTML

```
<table > // это начинается таблица
    <tr> // это начинается новая строка таблицы
        <td>… </td> //это ячейка таблицы в текущей строке
    </tr>
</table>
```

- XML

```
<user_info> //это начинается информация о пользователе
    <name> …</name> //это имя пользователя
    <nick>…</nick > //это псевдоним пользователя
</user_info>
```

XML

- Широко используется в WWW технологии
  - Форматирование документов
  - Описания структур документов
  - Описания интерфейсов, типов данных
  - Описания входных и выходных параметров при вызовах программ
  - Описания методов сериализации/десериализации объектов при передачи по сети
- Все это в текстовом виде и независимо от аппаратной и программной платформы

Синтаксис XML документа

- Каждый открывающий дескриптор должен иметь закрывающий, если в блоке есть вложенные данные
  - <name> имя </name>
  - <name value="имя"/>
- Блоки не должны перекрываться
  - <name> имя <date> </name> </date> - нельзя
- корневой блок должен быть один

```
<?xml version="1.0"?>
<root>
    <name> ..</name>
    <name> …</name>
    …
<\root>
```

Синтаксис дескриптора блока

<**имя атрибут1**=“значение” **атрибут2**=“значение” … **атрибутN**=“значение”>
Данные блока
<**имя**>

- Пример

<**user name**=“saa” **age=“**16”>
<office >
<address=“”> … </address>
</office>
</**user**>

Пространства имен (namespace)

- Разные пользователи могут использовать одинаковые имена дескрипторов, атрибутов, имен, но смысл у них может быть разным
- Пример

<addr> 0xc0048000 </addr> //адрес памяти

<addr> Lomonosov str. 80</addr> //домашний адрес

- Для решения проблемы были введены пространства имен

Определение пространства имен

- Введены для того, чтобы отличать одинаковые имена в разных контекстах
- Пространство имен – определяется следующим образом

```
xmlns:префикс="уникальное имя"
```

- Уникальное имя – URI, который никуда может не указывать
- Пример

```
xmlns:my_ns="http://my_ns.xml/my_ns"
```

Использование пространства имен

```
<city:addr xmlns:city="http://www.city.ns.org" >
   Lomonosov Str. 80
</city:addr>
<mem:addr xmlns:mem="http://www.mem.ns.org" >
    0xc0048000
</mem:addr>
```

То же самое

```
<addr xmlns="http://www.city.ns.org" >
   Lomonosov Str. 80
</addr>
<addr xmlns="http://www.mem.ns.org" >
    0xc0048000
</addr>
```

Описания типов документов (DTD)

- Для того, чтобы четко указать какие элементы могут иметь какие атрибуты и какие типы значений могут эти атрибуты принимать

```
<!DOCTYPE GANGLIA_XML [
    <!ELEMENT CLUSTER (HOST | HOSTS | METRICS)*>
    <!ATTLIST CLUSTER NAME CDATA #REQUIRED>
    <!ATTLIST CLUSTER OWNER CDATA #IMPLIED>
    <!ATTLIST CLUSTER LATLONG CDATA #IMPLIED>
    <!ATTLIST CLUSTER URL CDATA #IMPLIED>
    <!ATTLIST CLUSTER LOCALTIME CDATA #REQUIRED>
]>
```

Если что-то не совпадает, то XML документ не верный

XML документ в соответствии с описанием

```
<CLUSTER
   NAME="cluster.univ.kiev.ua”
   LOCALTIME="1120060429"
   OWNER="National Taras Shevchenko
   University of Kyiv"
   LATLONG="50° 28' North; 30° 29' East;"
   URL="http://www.cluster.kiev.ua/">
….
```

Схемы

- Типизация помогает избежать ошибок
- Схемы – описание сложных структур данных (по аналогии со структурами и объектами C, C++)
- Фактически – описание типа
  - Какие элементы каких типов могут входить в сложную структуру данных
  - Сколько и каких элементов может в этой структуре быть
- Используется XML вместо DTD
- Хорошо подходит для запросов и ответов от баз данных, передачи RPC параметров

Пример схемы

C++

```
class person_t {
    string name;
    string surname;
    time_t birth_date;
};
```

XML

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <xsd:element name="person" type="personType"/>
   <xsd:complexType name=" personType ">
    <xsd:sequence>
    <xsd:element name="name" type="xsd:string"/>
    <xsd:element name="surname" type="xsd:string"/>
    <xsd:element name="birth_date" type="xsd:date"/>
    </xsd:sequence>
   </xsd:complexType>
</xsd:schema>
```

Пример объектов (структур)

C++

```
person_t person;
person.name=“Olexandr ”;
person.surname=“Sudakov ”;
person.birth_date=“23-12-197
   3”;
```

XML

```
<?xml version="1.0"?>
<person>
   <name> Olexandr </name>
    <surname>
    Sudakov
   </surname>
   <birth_date>
    23-12-1973
   </birth_date>
</person>
```

Document Object Model (DOM)

- Представление сложных объектов в виде XML документов
- Представление XML документов в виде структур данных на каком-либо языке программирования
  - Сериализация/десериализация
  - Передача аргументов и возврат значений
  - Простое создание XML документов

Пример

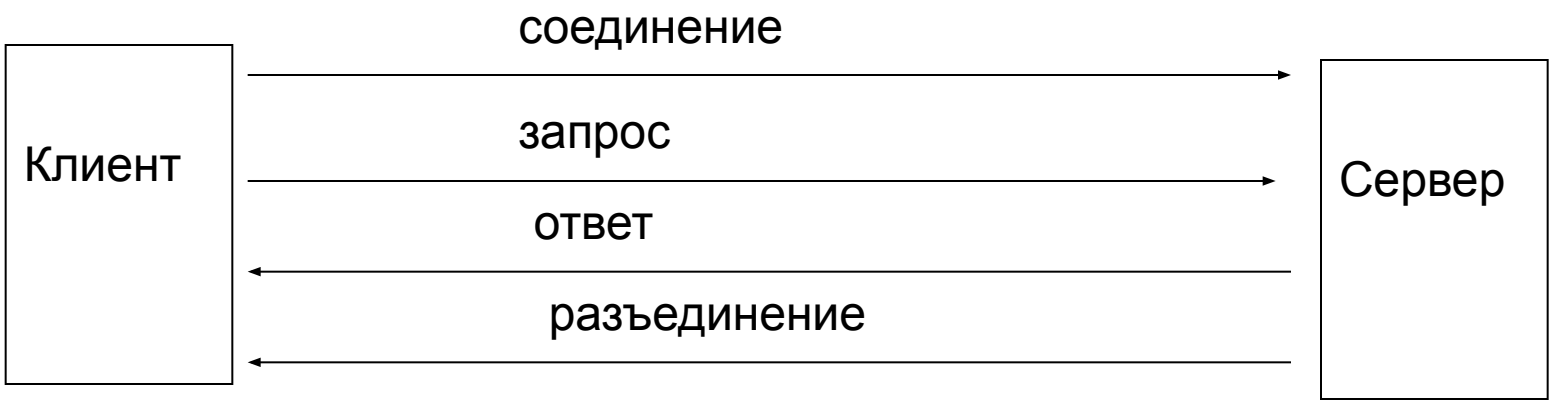
```
<script type="text/javascript">
function loadXML()
{
var xmlDoc = new ActiveXObject("Microsoft.XMLDOM")
xmlDoc.async="false"
xmlDoc.load("note.xml")

to.innerText=
xmlDoc.getElementsByTagName("to").item(0).text
from.innerText=
xmlDoc.getElementsByTagName("from").item(0).text
header.innerText=
xmlDoc.getElementsByTagName("heading").item(0).text
body.innerText=
xmlDoc.getElementsByTagName("body").item(0).text
}
</script>
</head>
```

Сетевой протокол прикладного уровня

- HTTP – hypertext transfer protocol
- RFC 2616 HTTP/1.1
- RFC 1945   HTTP/1.0
- Работает по верх протокола TCP, рассчитанного на соединение
- HTTP рассчитан на передачу запроса (в виде сообщений) серверу и получение ответа (в виде сообщений)

HTTP сеанс



- Клиент устанавливает соединение
- Клиент отправляет на сервер запрос
- Сервер обрабатывает запрос и отправляет ответ
- Сервер обычно разрывает соединение после отправки всего ответа

Формат запроса

- Метод запроса, URI и протокол запроса
- Заголовок
  - Опциональные параметры запроса
  - Опциональные параметры команды
- Пустая строка

```
GET http://www.univ.kiev.ua/ HTTP/1.0
```

- Методом GET
- Запросили список корневого каталога сервера www.univ.kiev.ua
- Протокол HTTP/1.0

Формат ответа

- Протокол ответа, Код статуса
- Параметры заголовка ответа
- Пустая строка
- Данные ответа
- Разрыв соединения

```
HTTP/1.1 200 OK
Date: Tue, 28 Jun 2005 13:52:43 GMT
Server: Apache/1.3.29 (Unix) mod_fastcgi/2.4.2 PHP/4.3.9
      mod_perl/1.26 mod_ssl/2.8.16 OpenSSL/0.9.7d
Connection: close
Content-Type: text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2
      Final//EN">
<HTML>
 <HEAD>
  <TITLE>Index of /</TITLE>
 </HEAD>
 <BODY>
…
```

Методы запроса

- `GET` – получить заданный URL
- `PUT` – поместить нижеследующие данные в заданный URL
- `POST` – передать нижеследующие данные ресурсу, связанному с URL
- `DELETE` – удалить заданный URL

Работа за брандмауэром и кэширование

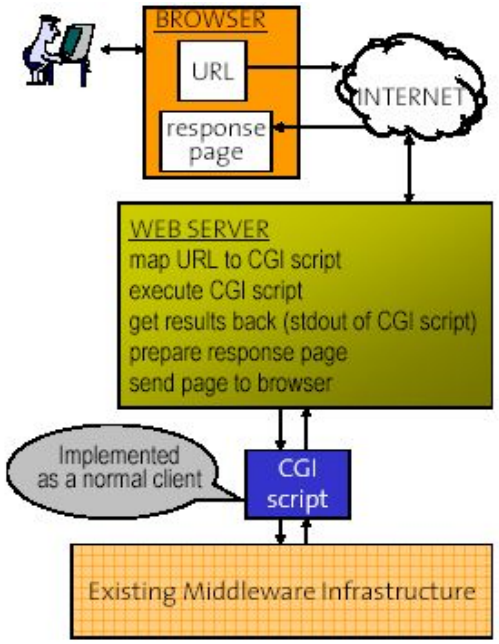
- HTTP хорошо рассчитан на работу за брандмауэром
  - Клиент является активным и к клиенту присоединяться не нужно невозможно
- HTTP удобен для кэширования информации
  - Клиент обращается к прокси-серверу с просьбой о доставке URI
  - Прокси-сервер либо доставляет URI, и кэширует, либо возвращает закэшированные данные, если они есть
- Достаточно легко сжимать и шифровать информацию в потоке

Основное использование WWW

- Серверы данных
  - Возможность получить данные (документы, программы) с другой машины на свою машину и запустить или открыть их
- Серверы приложений
  - Возможность запустить программу (сервлет) на другой машине и увидеть результат ее выполнения на своем браузере
- WWW службы
  - Возможность разработать распределенную программу, которая состоит из множества компонент, находящихся на разных машинах (аналогично CORBA)

Серверы приложений

- Клиент обращается к серверу (пользователь «кликает»)
- Сервер запускает приложение, связанное с URI, указанным клиентом
- Приложение выполняет сложную работу и выдает результат в виде HTML



Основные подходы к созданию серверов приложений

- CGI – common gateway interface
  - Клиент передает серверу URI и параметры программы
  - Сервер запускает программу с заданными параметрами (cgi обработчик)
  - Программа-обработчик возвращает результаты выполнения в виде HTML на стандартный вывод, который непосредственно передается клиенту
- Контейнер-серверы
  - Сервер сам является интерпретатором с некоторого языка программирования (Java)
  - С URI связан некоторый исполняемый файл (сервлет)
  - Сервер запускает сервлет с параметрами, переданными клиентом
  - Сервлет возвращает результат

Передача параметров на сервер (CGI)

- Метод GET
  - Как часть URI

GET http://cluster/ganglia/index?c=cluster.univ.kiev.ua&h=ss20-15.univ.kiev.ua HTTP/1.0
Пустая строка

- Метод POST
  - В заголовке запроса

```
POST http://cluster/ganglia/index HTTP/1.0
Content-Length: 48

&c=cluster.univ.kiev.ua
&h=ss20-15.univ.kiev.ua
```

Ответ сервера

```
HTTP/1.1 200 OK
Date: Wed, 29 Jun 2005 16:17:40 GMT
Server: Apache/2.0.51 (Fedora)
X-Powered-By: PHP/4.3.8
Set-Cookie: gs=grid.org.ua%40; expires=Thu, 30-Jun-2005 16:17:41 GMT
Expires: Mon, 26 Jul 1997 05:00:00 GMT
Last-Modified: Wed, 29 Jun 2005 16:17:41 GMT
Cache-Control: no-cache, must-revalidate
Pragma: no-cache
Connection: close
Content-Type: text/html; charset=UTF-8

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<HTML>
<HEAD>
<TITLE>Ganglia:: Host Report</TITLE>
```

Обработка параметров при запуске CGI программы

- Метод GET – программе передается URL
  - Через системную переменную
  - Через динамически создаваемые переменные с уже разобранным синтаксисом
- Метод POST – программа считывает данные со стандартного ввода
  - Данные обрабатываются самой программой
  - Данные обрабатываются специальной библиотечной функцией

Недостатки CGI и их преодоление

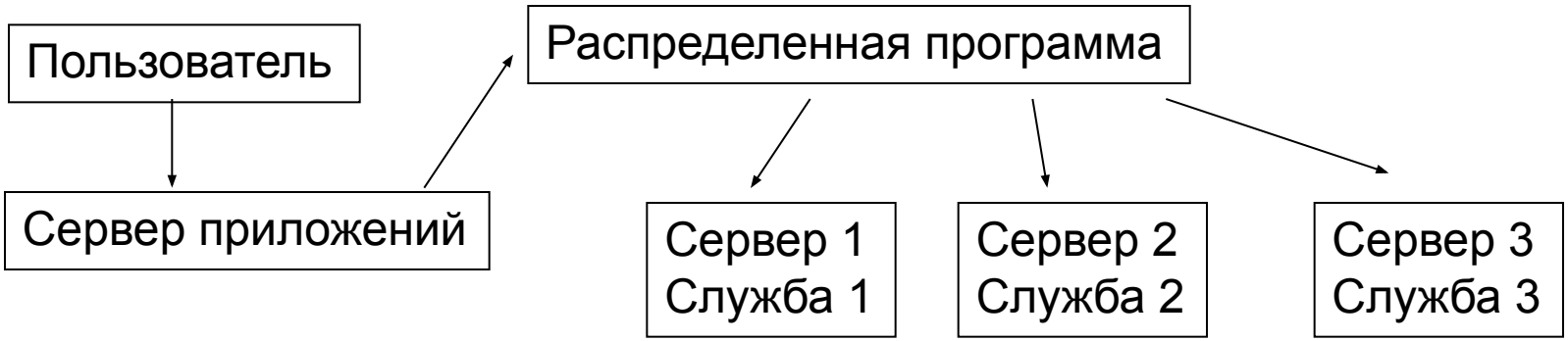
- Необходимость запуска программы на каждый запрос
  - При большом количестве запросов много процессорного времени уходит только на запуск
- Сервлеты – отдельный поток
  - Значительно облегчается переключение контекста

Сервлеты

- Играют ту же роль, что и CGI скрипт, но
  - Написаны на Java и хорошо портируются
  - Выполняются как функция в отдельном потоке WWW сервера
  - Для сервлетов необходим контейнер-сервер, который их может выполнять
  - Сервлеты очень хорошо подходят для создания WWW служб с использованием распределенной компонентной модели
- Java Beans – технолгия создания распределенных программ

WWW службы

- Серверы приложений – результат получает пользователь
- WWW службы – части большой распределенной программы, которые работают на разных WWW серверах (RPC)

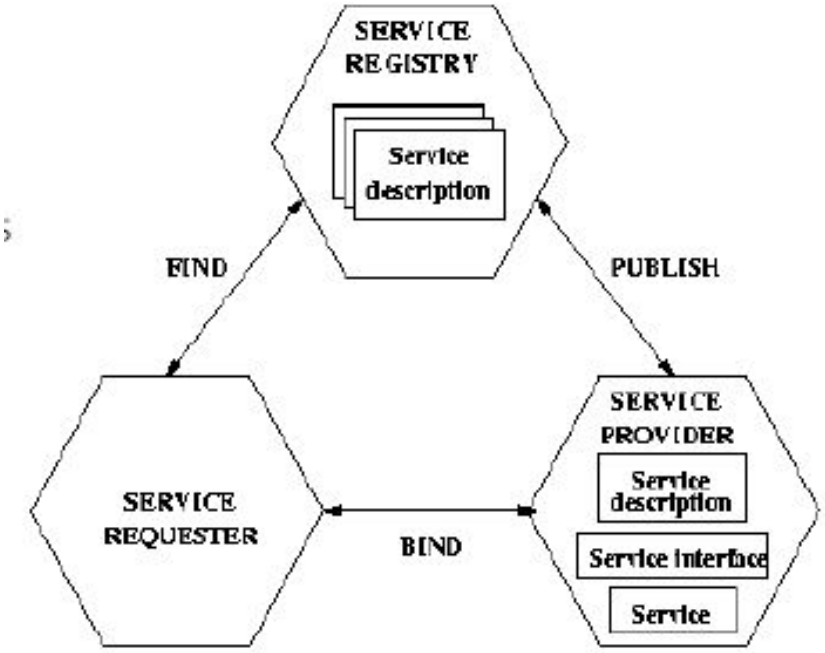


Определение WWW службы

- WWW служба – это программа, которая идентифицируется с помощью URI и все интерфейсы (структура входных и выходных данных) который могут быть описаны с помощью XML
- WWW служба может быть интегрирована (использована) в любой программе, которая умеет работать с XML посредством обращения через Интернет

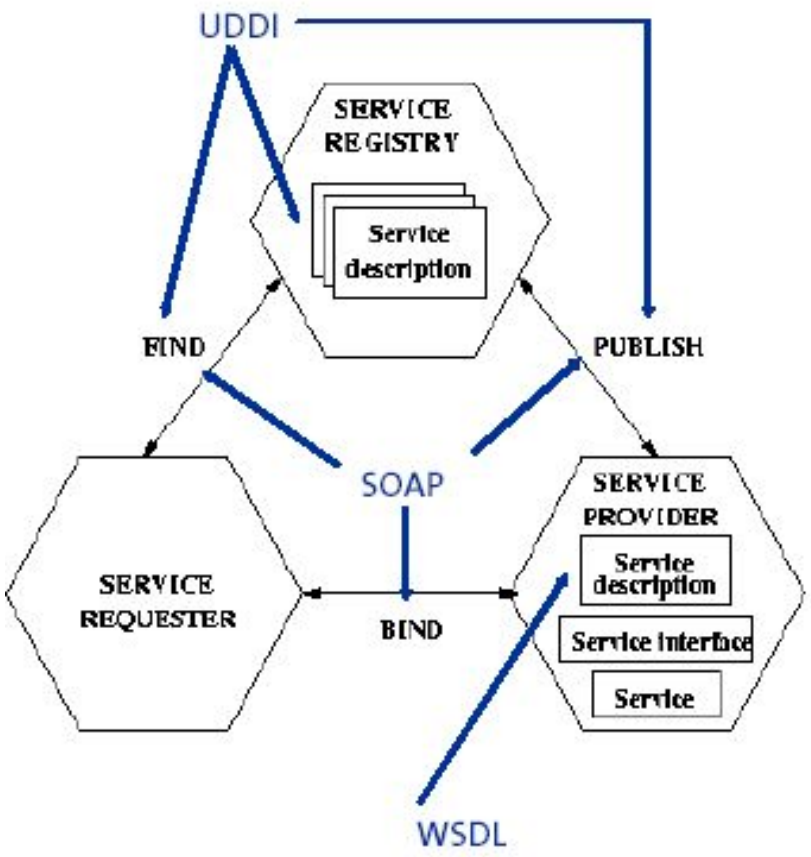
Структура системы WWW служб

- Инициатор запроса - клиент
- Провайдер службы – сервер, где выполняется программа от имени инициатора запроса
- Реестр служб – где прописаны все службы, где провайдер может себя зарегистрировать, а инициатор запроса получить информацию о необходимых службах



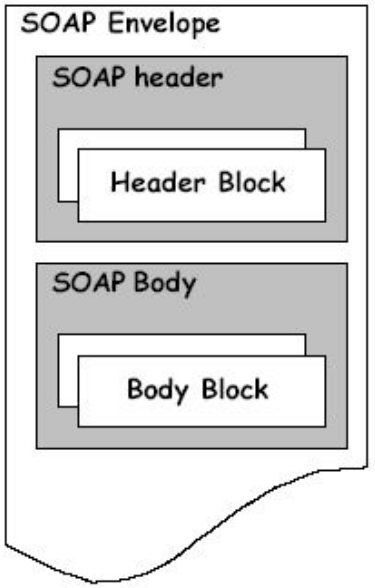
Стандарты для WWW служб

- SOAP – simple object access protocol
- UDDI – universal description and discovery protocol
- WSDL – Web Service Description Language

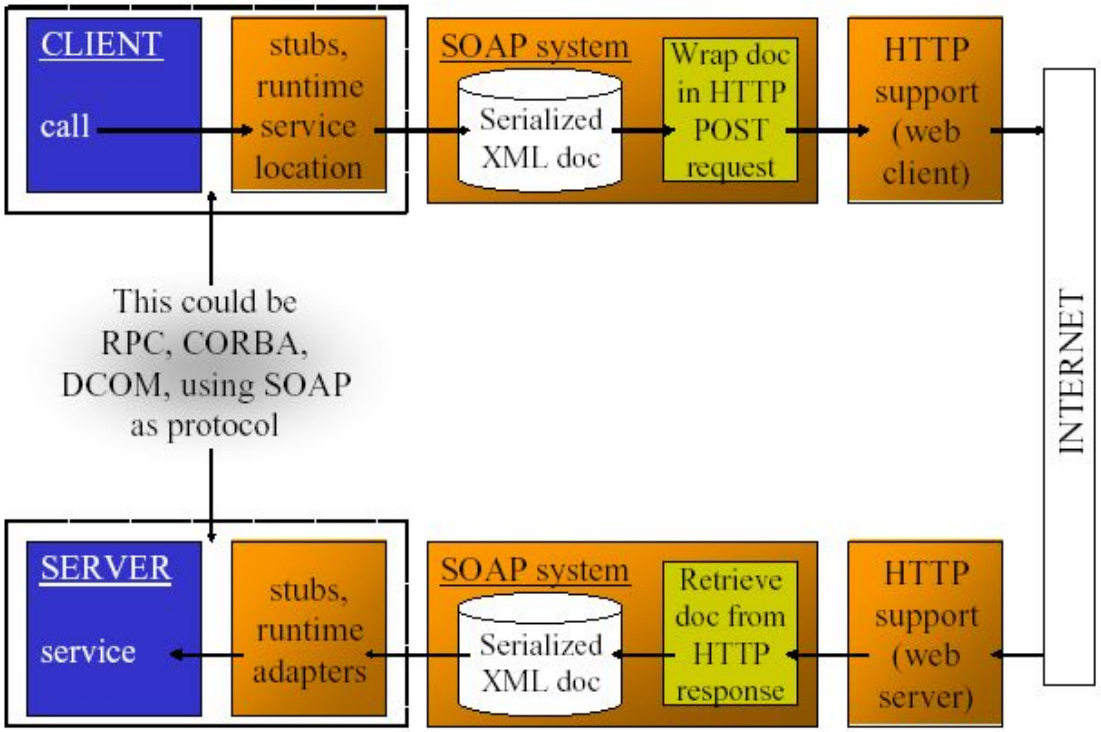


Протокол SOAP

- Инкапсуляция различных протоколов в XML и передача с помощью HTTP
- Состоит из оболочки, заголовка и тела сообщения, элементы которых могут быть структурированы
- Так же как и HTTP состоит из запросов и ответов на них



SOAP – RPC в пределах Интернет



- Оболочка – кодировка
- Заголовок – информация, которая не касается программы обработчика
- Тело – параметры запроса

```
<SOAP-ENV:Envelope
     xmlns:SOAP-ENV=
"http://schemas.xmlsoap.org/soap/envelope/"
     SOAP-ENV:encodingStyle=
"http://schemas.xmlsoap.org/soap/encoding/"/>
    <SOAP-ENV:Header>
        <t:Transaction
            xmlns:t="some-URI"
            SOAP-ENV:mustUnderstand="1">
                5
        </t:Transaction>
    </SOAP-ENV:Header>

    <SOAP-ENV:Body>
        <m:GetLastTradePrice xmlns:m="Some-URI">
            <symbol>DEF</symbol>
        </m:GetLastTradePrice>
    </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

Связь SOAP с транспортным протоколом

- Обычно HTTP
- Метод GET
  - входные данные – URI
  - Выходные данные – SOAP
- Метод POST
  - Входные данные SOAP
  - Выходные данные - SOAP

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

<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">
      <symbol>DIS</symbol>
   </m:GetLastTradePrice>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

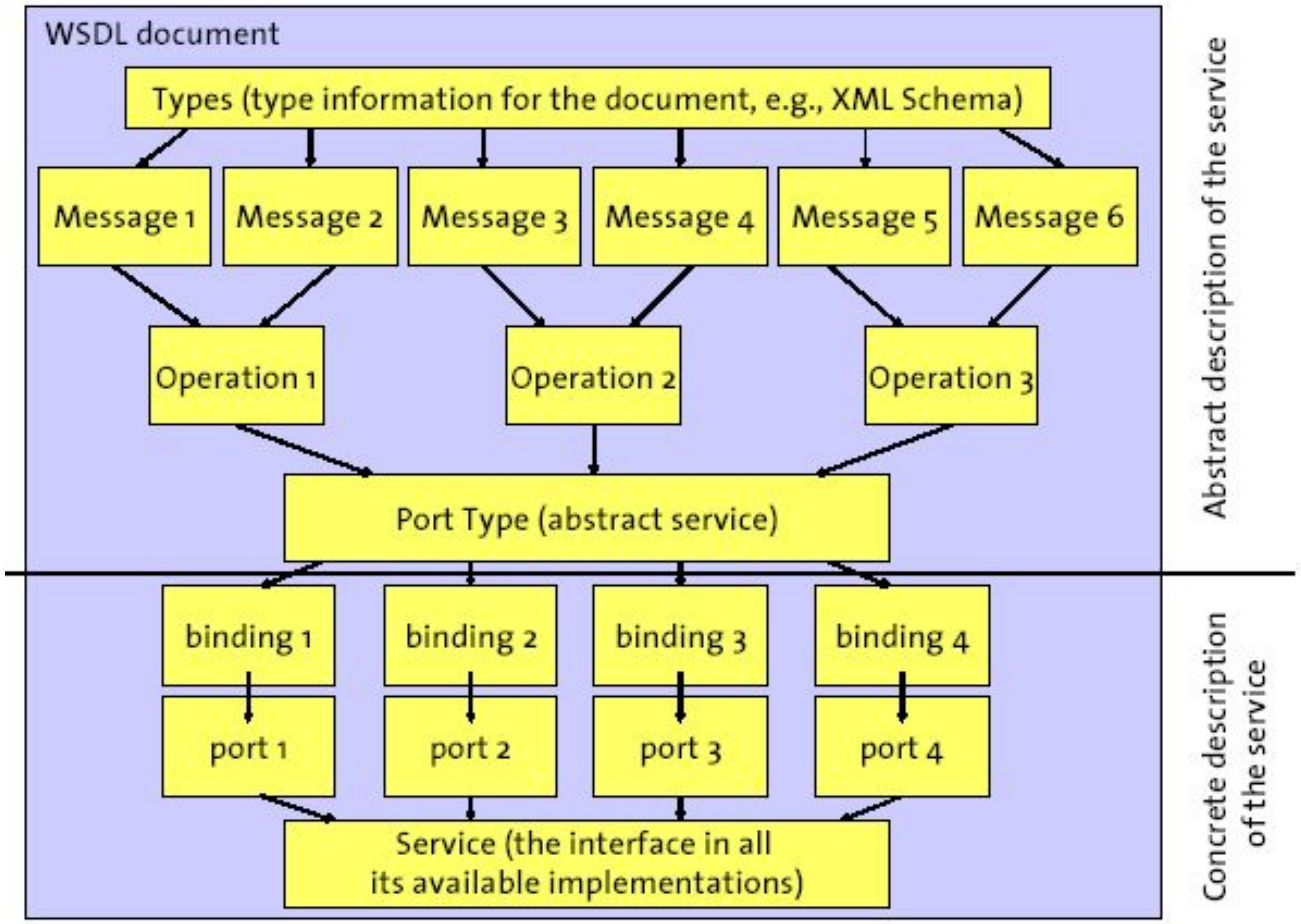
XML RPC

- http://www.xmlrpc.com/
- Простой протокол удаленных вызовов процедур через Интернет

WSDL – аналог IDL

- Описание WWW служб на языке XML
- Для работы достаточно знать только интерфейсы
  - XML схема, которая используется
  - Какие типы сообщений могут использоваться для передачи данных
  - Какие операции (методы) может выполнять служба
  - Какой транспортный протокол может использоваться для каких сообщений
  - Где работает служба (на каком сервере)
  - Откуда служба может быть доступна (контроль доступа)

WSDL – XML документ



Пример описания типов

PURCHASE ORDER TYPE

```
<element name="PO" type="tns:POType"/>
<complexType name="POType">
  <all>
    <element name="id" type="string"/>
    <element name="name" type="string"/>
    <element name="items">
      <complexType>
        <all>
          <element name="item" type="tns:Item" minOccurs="0" maxOccurs="unbounded"/>
        </all>
      </complexType>
    </element>
  </all>
</complexType>
```

ITEM TYPE

```
<complexType name="Item">
  <all>
    <element name="quantity" type="int"/>
    <element name="product" type="string"/>
  </all>
</complexType>
```

INVOICE TYPE

```
<element name="Invoice" type="tns:InvoiceType"/>
<complexType name="InvoiceType">
  <all>
    <element name="id" type="string"/>
  </all>
</complexType>
```

From Web Services Description Language (WSDL) 1.1 W3C Note 15 March 2001

Пример описания операций

- Операция – функция
- Сообщение – типы передаваемых аргументов и возвращаемых значений для операций
- Типы взаимодействий

ONE-WAY:

```
<wsdl:operation name="Purchase">
  <wsdl:input name="Order" message="PO"/>
</wsdl:operation>
```

REQUEST-RESPONSE:

```
<wsdl:operation name="Purchase">
  <wsdl:input name="Order" message="PO"/>
  <wsdl:output name="Confirm" message="Conf"/>
  <wsdl:fault name="Error" message="POError"/>
</wsdl:operation>
```

Порты

- Порт – набор всех операций службы, доступных по одному протоколу
- Порту соответствуют операции и типы сообщений
- Операции определяются для каждого порта

```
<message name="m1">
 <part name="body" element="tns:GetCompanyInfo"/>
</message>

<message name="m2">
 <part name="body" element="tns:GetCompanyInfoResult"/>
 <part name="docs" type="xsd:string"/>
 <part name="logo" type="tns:ArrayOfBinary"/>
</message>

<portType name="pt1">
 <operation name="GetCompanyInfo">
     <input message="m1"/>
     <output message="m2"/>
 </operation>
</portType>
```

Привязки (binding) и службы

- Привязка
  - связь портов с протоколом (soap)
- Служба
  - Набор портов доступных по разным протоколам
  - Разные порты службы имеют одни и те же операции, но доступны по разным протоколам

```
<binding name="b1" type="tns:pt1">
    <operation name="GetCompanyInfo">
        <soap:operation soapAction="http://example.com/GetCompanyInfo"/>
            <input>
              <soap:body use="literal"/>
            </input>
            <output>
              <mime:multipartRelated>
                <mime:part>
                  <soap:body parts="body" use="literal"/>
                </mime:part>
                <mime:part>
                  <mime:content part="docs" type="text/html"/>
                </mime:part>
                <mime:part>
                  <mime:content part="logo" type="image/gif"/>
                  <mime:content part="logo" type="image/jpeg"/>
                </mime:part>
              </mime:multipartRelated>
            </output>
    </operation>
</binding>
<service name="CompanyInfoService">
        <port name="CompanyInfoPort"binding="tns:b1">
                <soap:address location="http://example.com/companyinfo"/>
        </port>
</service>
```

UDDI

- Высокоуровневая инфраструктура для WWW служб
  - Поиск WWW служб
  - Регистрация WWW служб
  - Обеспечение надежности
  - Координация (балансировка нагрузки)
  - …

Примеры использования WWW

- GRID системы – распределенные системы для обеспечения совместного использования и координации вычислительных ресурсов, которые географически распределены
- http://alien.cern.ch/
  - ГРИД система, в которую входят вычислительные кластеры различных стран для расчетов по ядерной физике
  - Обеспечивается балансировка нагрузки, совместное использование дисковых ресурсов

Вопросы ?