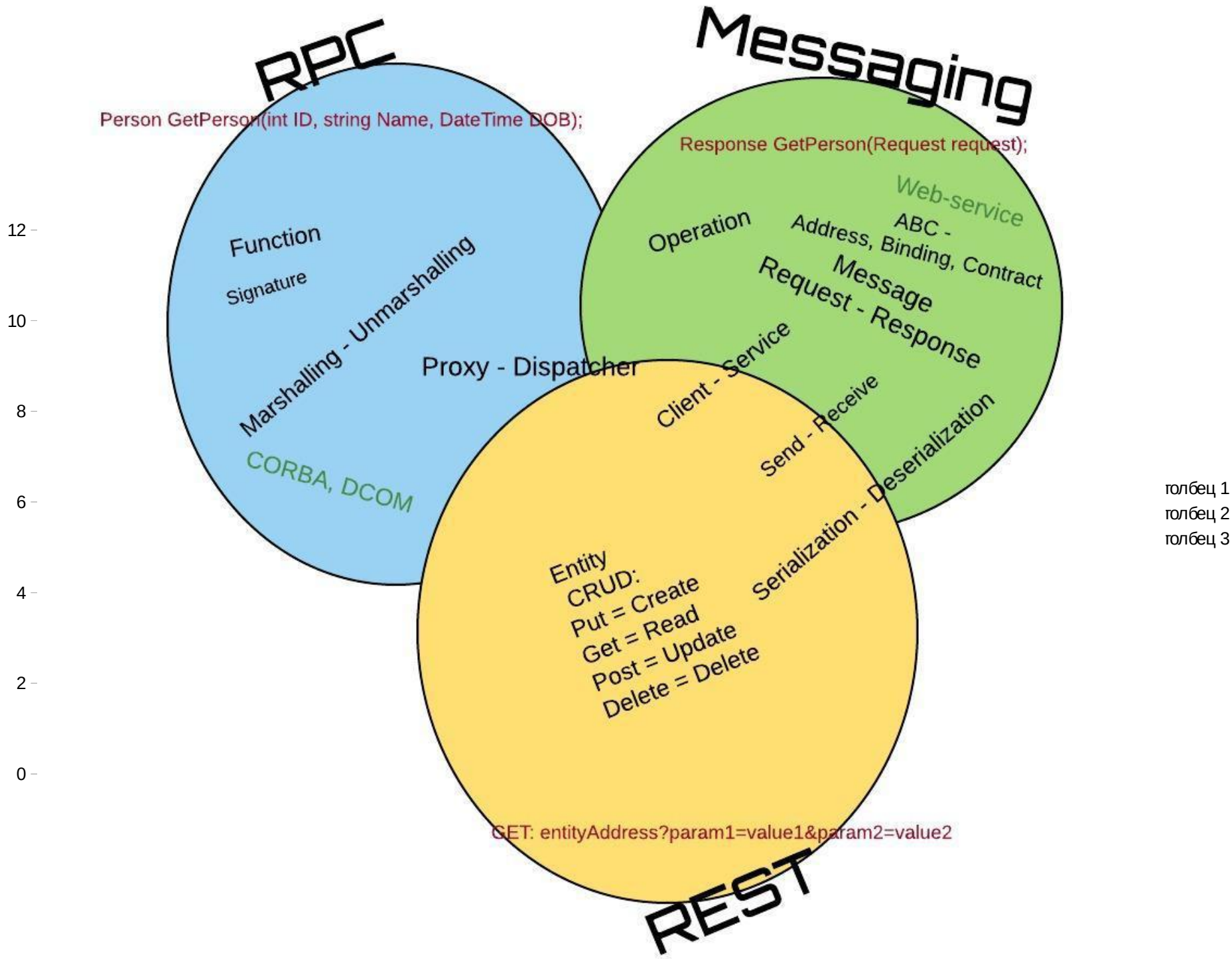
Лекция №5

RPC



голбец 1
голбец 2
голбец 3

Удаленный вызов процедур

Удалённый вызов процедур, реже Вызов удалённых процедур (от англ. **Remote Procedure Call, RPC**) — класс технологий, позволяющих компьютерным программам вызывать функции или процедуры в другом адресном пространстве (как правило, на удалённых компьютерах). Обычно реализация RPC технологии включает в себя два компонента: сетевой протокол для обмена в режиме клиент-сервер и язык сериализации объектов (или структур, для необъектных RPC). Различные реализации RPC имеют очень отличающуюся друг от друга архитектуру и разнятся в своих возможностях: одни реализуют архитектуру SOA, другие CORBA или DCOM. На транспортном уровне RPC используют в основном протоколы TCP и UDP

Удаленный вызов процедур

Существует множество технологий, обеспечивающих RPC:

- DCE/RPC — Distributed Computing Environment / Remote Procedure Calls (бинарный протокол на базе различных транспортных протоколов, в том числе TCP/IP и Named Pipes из протокола SMB/CIFS)
- **DCOM — Distributed Component Object Model** известный как MSRPC Microsoft Remote Procedure Call или «Network OLE» (объектно-ориентированное расширение DCE RPC, позволяющее передавать ссылки на объекты и вызывать методы объектов через таковые ссылки)
- ZeroC ICE
- **JSON-RPC**— JavaScript Object Notation Remote Procedure Calls (текстовый протокол на базе HTTP) см. спецификацию: RFC-4627
- .NET Remoting (бинарный протокол на базе TCP, UDP, HTTP)
- **Java RMI** — Java Remote Method Invocation — см. спецификацию: http://java.sun.com/j2se/1.5.0/docs/guide/rmi/index.html
- **SOAP** — Simple Object Access Protocol (текстовый протокол на базе HTTP) см. спецификацию: RFC-4227
- Sun RPC (бинарный протокол на базе TCP и UDP и XDR) RFC-1831 второе название ONC RPC RFC-1833
- **XML RPC** (текстовый протокол на базе HTTP) см. спецификацию: RFC-3529
- Routix.RPC

RPC

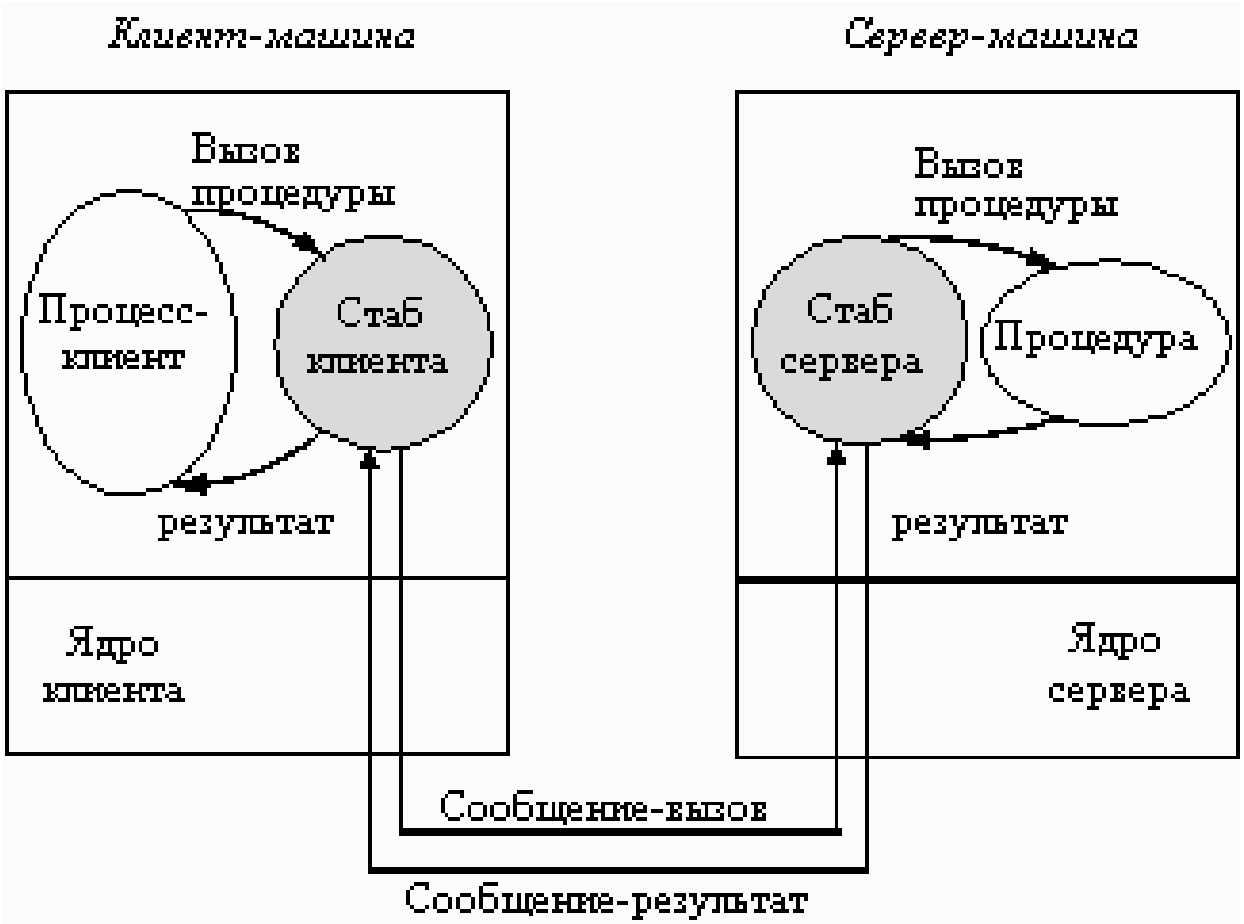
Средства удаленного вызова процедур предназначены для облегчения организации распределенных вычислений. Наибольшая эффективность использования RPC достигается в тех приложениях, в которых существует интерактивная связь между удаленными компонентами с небольшим временем ответов и относительно малым количеством передаваемых данных. Такие приложения называются RPC-ориентированными.

Реализация удаленных вызовов существенно сложнее реализации вызовов локальных процедур.

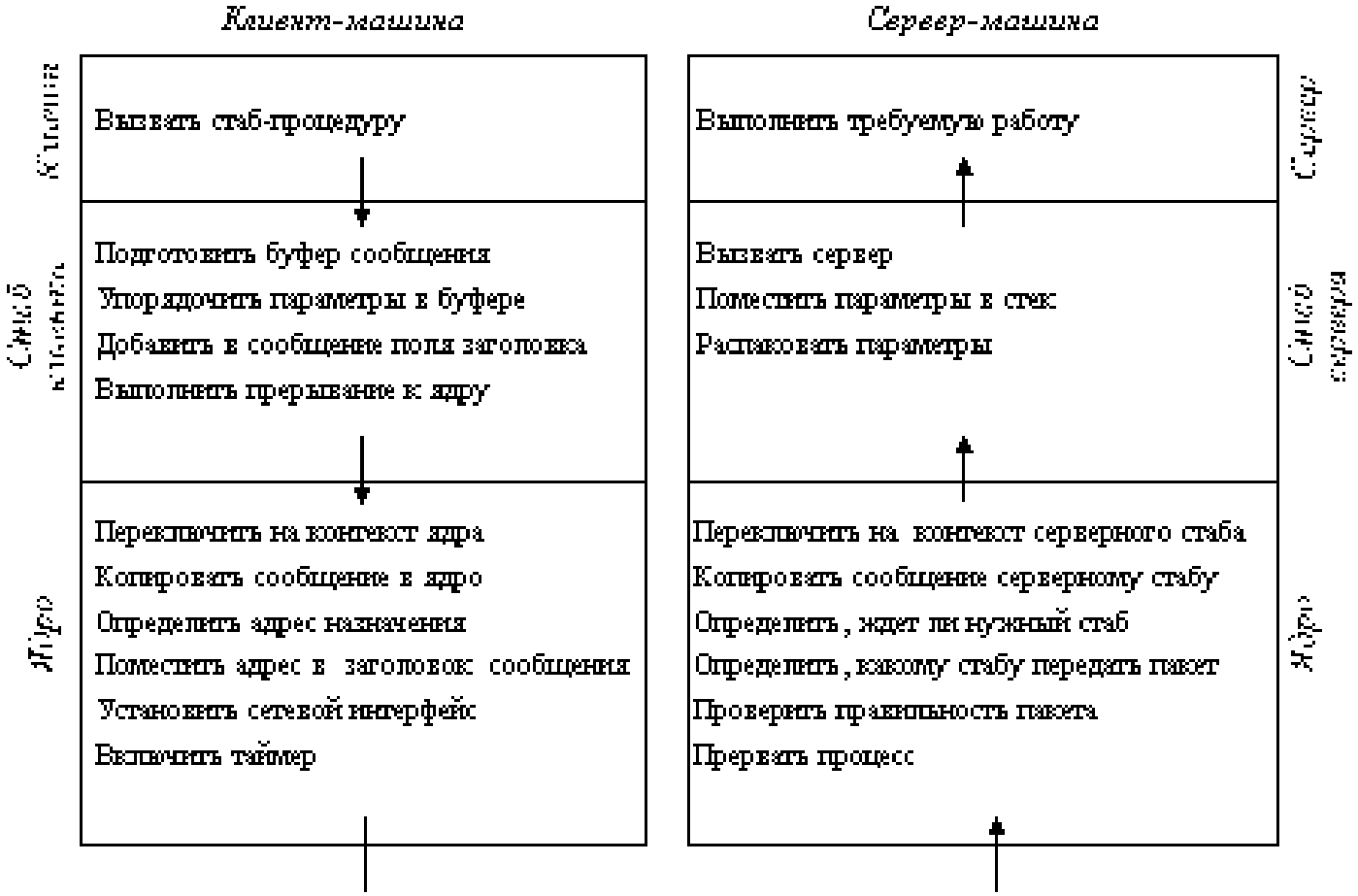
RPC

RPC достигает прозрачности следующим путем. Когда вызываемая процедура действительно является удаленной, в библиотеку помещается вместо локальной процедуры другая версия процедуры, называемая клиентским стабом (stub - заглушка). Подобно оригинальной процедуре, стаб вызывается с использованием вызывающей последовательности (как на рисунке 3.1), так же происходит прерывание при обращении к ядру. Только в отличие от оригинальной процедуры он не помещает параметры в регистры и не запрашивает у ядра данные, вместо этого он формирует сообщение для отправки ядру удаленной машины.

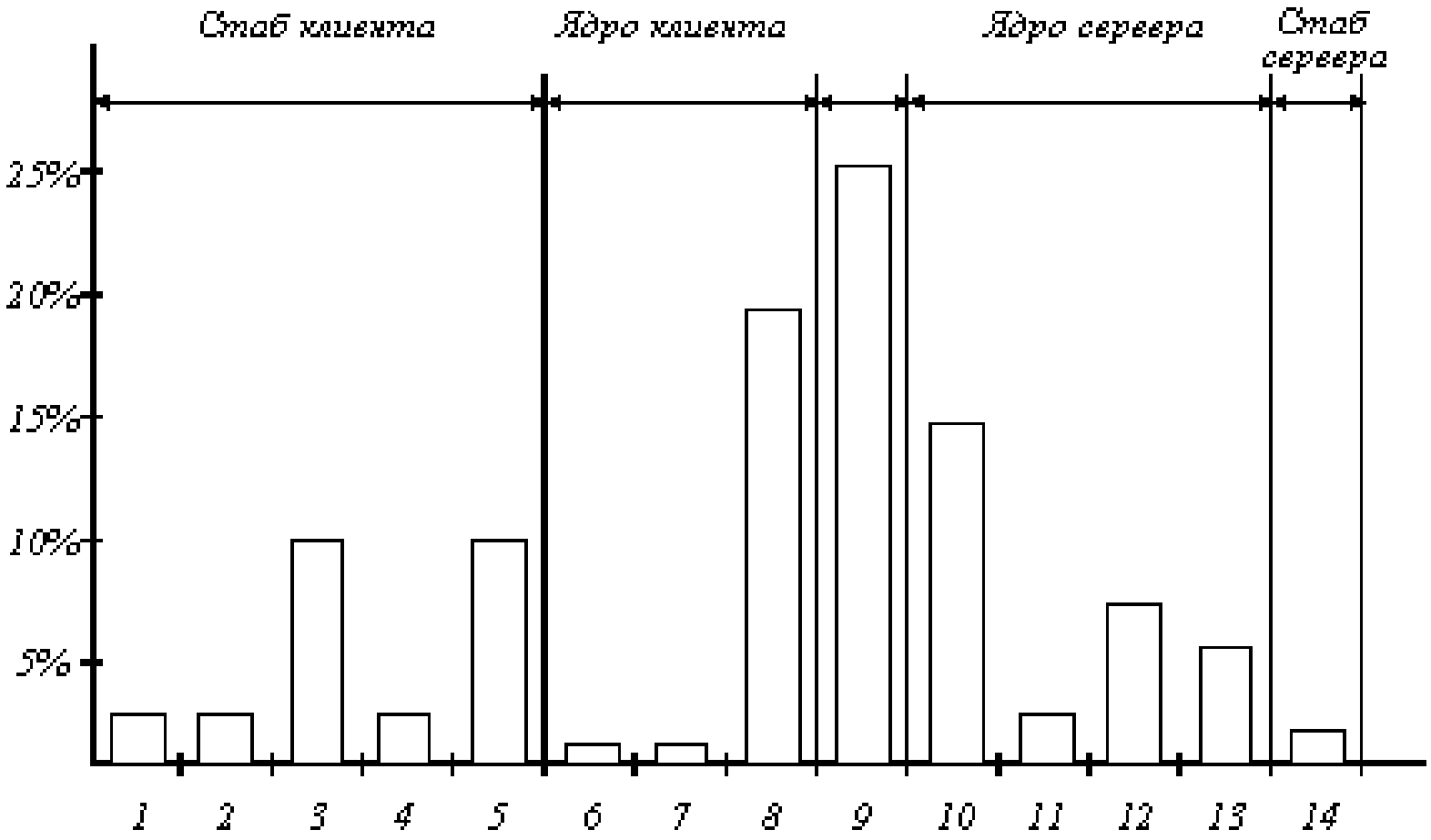
RPC



Этапы выполнения RPC



Распределение времени RPC



XML-RPC

XML-RPC (сокр. от англ. Extensible Markup Language Remote Procedure Call — XML-вызов удалённых процедур) — стандарт/протокол вызова удалённых процедур, использующий XML для кодирования своих сообщений и HTTP в качестве транспортного механизма. Является прародителем SOAP, отличается исключительной простотой в применении. XML-RPC, как и любой другой интерфейс Remote Procedure Call (RPC), определяет набор стандартных типов данных и команд, которые программист может использовать для доступа к функциональности другой программы, находящейся на другом компьютере в сети.

XML-RPC / Пример запроса

```
POST /RPC2 HTTP/1.0
User-Agent: Frontier/5.1.2 (WinNT)
Host: betty.userland.com
Content-Type: text/xml
Content-length: 181

<?xml version="1.0"?>
<methodCall>
   <methodName>examples.getStateName</methodName>
   <params>
      <param>
         <value><i4>41</i4></value>
      </param>
    </params>
</methodCall>
```

XML-RPC

Типичный пример ответа на запрос XML-RPC:

```
<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
        <value><string>South Dakota</string></value>
    </param>
  </params>
</methodResponse>
```

XML-RPC

- C++ - http://xmlrpc-c.sourceforge.net/
- PHP - http://gggeek.github.io/phpxmlrpc/
- Java - http://ws.apache.org/xmlrpc/
- и много других языков

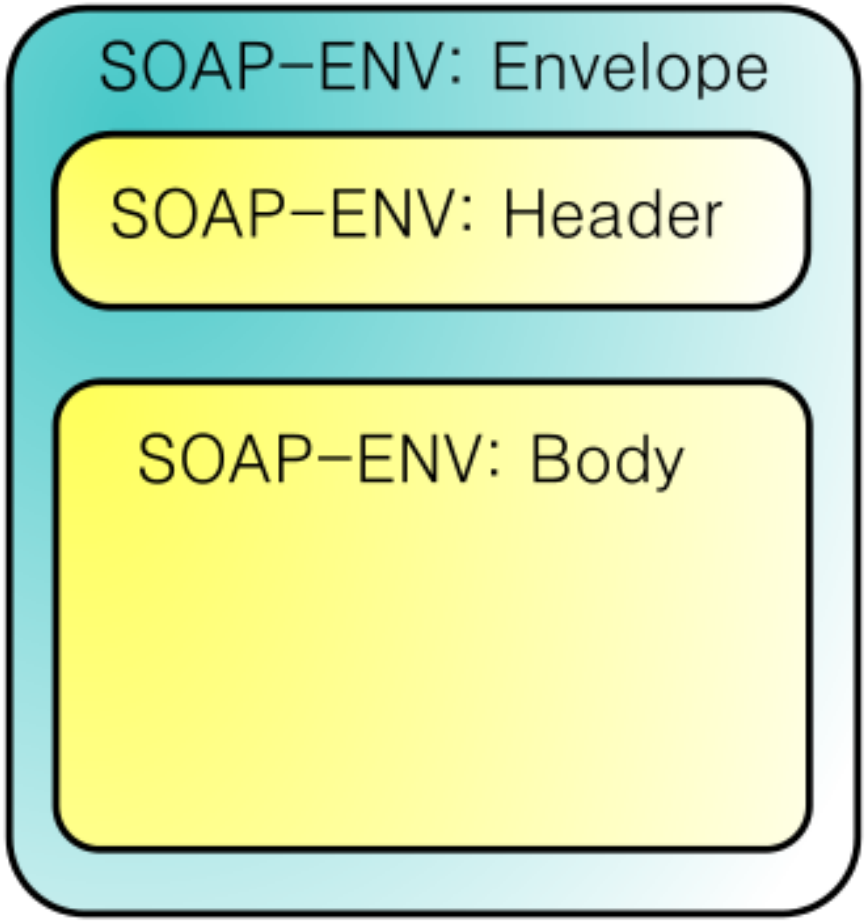
XML-RPC Server

```
import org.apache.xmlrpc.*;

public class JavaServer {

   public Integer sum(int x, int y){
      return new Integer(x+y);
   }

   public static void main (String [] args){

      try {

         System.out.println("Attempting to start XML-RPC Server...");

         WebServer server = new WebServer(80);
         server.addHandler("sample", new JavaServer());
         server.start();

         System.out.println("Started successfully.");
         System.out.println("Accepting requests. (Halt program to stop.)");

      } catch (Exception exception){
         System.err.println("JavaServer: " + exception);
      }
   }
}
```

XML-RPC Client

```
import java.util.*;
import org.apache.xmlrpc.*;

public class JavaClient {
   public static void main (String [] args) {

      try {
         XmlRpcClient server = new XmlRpcClient("http://localhost/RPC2");
         Vector params = new Vector();

         params.addElement(new Integer(17));
         params.addElement(new Integer(13));

         Object result = server.execute("sample.sum", params);

         int sum = ((Integer) result).intValue();
         System.out.println("The sum is: "+ sum);

      } catch (Exception exception) {
         System.err.println("JavaClient: " + exception);
      }
   }
}
```

SOAP



SOAP (Simple Object Access Protocol — простой протокол доступа к объектам) — протокол обмена структурированными сообщениями в распределённой вычислительной среде. Первоначально SOAP предназначался в основном для реализации удалённого вызова процедур (RPC). Сейчас протокол используется для обмена произвольными сообщениями в формате XML, а не только для вызова процедур. SOAP является расширением протокола XML-RPC

SOAP

SOAP has three major characteristics:

1) extensibility (security and WS-routing are among the extensions under development)
2) neutrality (SOAP can operate over any protocol such as HTTP, SMTP, TCP, UDP, or JMS)
3) independence (SOAP allows for any programming model)

SOAP / Message

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: 299
SOAPAction: "http://www.w3.org/2003/05/soap-envelope"

<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:m="http://www.example.org/stock/Surya">
  <soap:Header>
  </soap:Header>
  <soap:Body>
    <m:GetStockPrice>
      <m:StockName>GOOGL</m:StockName>
    </m:GetStockPrice>
  </soap:Body>
</soap:Envelope>
```

SOAP Server

SOAP Client

REST

REST (сокр. от англ. Representational State Transfer — «передача состояния представления») — архитектурный стиль взаимодействия компонентов распределённого приложения в сети. REST представляет собой согласованный набор ограничений, учитываемых при проектировании распределённой гипермедиа-системы. В определённых случаях (интернет-магазины, поисковые системы, прочие системы, основанные на данных) это приводит к повышению производительности и упрощению архитектуры. В широком смысле[уточнить] компоненты в REST взаимодействуют наподобие взаимодействия клиентов и серверов во Всемирной паутине. REST является альтернативой RPC.

RESTful

Для веб-служб, построенных с учётом REST (то есть не нарушающих накладываемых им ограничений), применяют термин «RESTful».

В отличие от веб-сервисов (веб-служб) на основе SOAP, не существует "официального" стандарта для RESTful веб-API. Дело в том, что REST является архитектурным стилем, в то время как SOAP является протоколом. Несмотря на то, что REST не является стандартом сам по себе, большинство RESTful-реализаций используют стандарты, такие как HTTP, URL, JSON и XML

REST / История

Данная технология стала популярной, когда она была подробно описана и представлена Роем Филдингом в его докторской диссертации под названием Architectural Styles and the Design of Network-based Software Architectures в 2000 году.

REST — это стиль архитектуры программного обеспечения для построения распределенных масштабируемых веб-сервисов. Рой выступал за использование стандартных HTTP методов так, чтобы придавать запросам определённый смысл, например:

- GET /object/list
- POST /object/list
- PUT /object/list

Весь их список: CONNECT, DELETE, GET, HEAD, OPTIONS, PATCH, POST, PUT, TRACE.

REST

Method	Method
200 OK	201 Created
202 Accepted	203 Not authorized
204 No content	205 Reset content
206 Partial content	
300 Multiple choice	301 Moved permanently
302 Found	303 See other
304 Not modified	306 (unused)
307 Temporary redirect	
400 Bad request	401 Unauthorized
402 Payment required	403 Forbidden
404 Not found	405 Method not allowed
406 Not acceptable	407 Proxy auth required
408 Timeout	409 Conflict
410 Gone	411 Length required
412 Preconditions failed	413 Request entity too large
414 Requested URI too long	415 Unsupported media
416 Bad request range	417 Expectation failed
500 Server error	501 Not implemented
502 Bad gateway	503 Service unavailable
504 Gateway timeout	505 Bad HTTP version

REST

Одна транзакция по такому API будет состоять, как минимум, из следующего:

- Метод запроса, например, GET
- Путь запроса, например, /object/list
- Тело запроса, например, форма
- Код ответа, например, 200 OK
- Тело ответа, например, данные в формате JSON

REST

Request

```
GET /
Accept: application/json+userdb
```

Response

```
200 OK
Content-Type: application/json+userdb

{
    "version": "1.0",
    "links": [
        {
            "href": "/user",
            "rel": "list",
            "method": "GET"
        },
        {
            "href": "/user",
            "rel": "create",
            "method": "POST"
        }
    ]
}
```

REST / Свойства

Свойства архитектуры, которые зависят от ограничений, наложенных на REST-системы:

- Производительность — взаимодействие компонентов системы может являться доминирующим фактором производительности и эффективности сети с точки зрения пользователя
- Масштабируемость для обеспечения большого числа компонентов и взаимодействий компонентов.

Влияние архитектуры REST на масштабируемость:

- Простота унифицированного интерфейса
- Открытость компонентов к возможным изменениям для удовлетворения изменяющихся потребностей (даже при работающем приложении)
- Прозрачность связей между компонентами системы для сервисных служб
- Переносимость компонентов системы путем перемещения программного кода вместе с данными
- Надежность является устойчивостью к отказам на уровне системы при наличии отказов отдельных компонентов, соединений, или данных.

REST / Требования

1) Модель клиент-сервер
2) Отсутствие состояний
3) Кэширование
4) Единообразие интерфейса (идентификация ресурсов, манипуляция через представление, «самоописание», гиперссылки)
5) Слои / инкапсуляция
6) *Код по требованию*

REST / Преимущества

- Надёжность;
- Производительность;
- Масштабируемость;
- Прозрачность системы взаимодействия;
- Простота интерфейсов;
- Портативность компонентов;
- Лёгкость внесения изменений;
- Способность эволюционировать, приспосабливаясь к новым требованиям.

JSON-pure API

Материалы

- http://citforum.ru/operating_systems/sos/glava_12.shtml
- https://en.wikipedia.org/wiki/Representational_state_transfer
- https://habrahabr.ru/post/265845/
-