Московский государственный университет им. М.В. Ломоносова

Распределенные объектные технологии

Лекция 2. Организация связи в РВС

Разработчик:
Г.И. Радченко, к.ф.-м.н.
E-mail: gleb.radchenko@gmail.com
Южно-Уральский государственный университет

Направление 010300.68
«Фундаментальная информатика и информационные технологии»

Проект комиссии Президента по модернизации и техническому развитию экономики России
«Создание системы подготовки высококвалифицированных кадров в области суперкомпьютерных технологий и специализированного программного обеспечения»

Роль связи в РВС

Протокол

Взаимодействие базируется на протоколах.

Протокол – это набор правил и соглашений, описывающий процедуру взаимодействия между компонентами системы.

Стек протоколов OSI

Организация обмена сообщениями

- **Прямая передача сообщений**
  - возможна только если принимающая сторона готова к приему сообщения в этот момент времени
- **Использование менеджера сообщений**
  - компонента высылает сообщение в очередь менеджера, из которой, в дальнейшем, принимающая сторона извлекает полученное сообщение

Прямая передача сообщений: сокеты

Прямая передача сообщений: сокеты

- Используется на основе интерфейса сокетов
- Т.е. используется непосредственно транспортный уровень в виде Middleware.

- **Сокет –** абстрактный объект, представляющий конечную точку соединения.
- **Сокет TCP/IP –** комбинация IP-адреса и номера порта, например 10.10.10.10:80.
- Интерфейс сокетов впервые появился в BSD Unix.

Пример реализации сокета

Язык C# поддерживает два типа сетевых соединений:

- серверные, реализуемые с помощью объектов класса TcpListener;
- клиентские, реализуемые с помощью объектов класса TcpClient.

Объекты TcpListener и TcpClient

- Объект класса TcpListener позволяет **только прослушивать** определенный порт компьютера.
- **Любые процессы передачи** данных через этот сокет **осуществляются с использованием объекта TcpClient**.
- TcpClient возвращается методом AcceptTcpClient() класса TcpListener, что обеспечивает сам процесс прослушивания порта.

Пример создания сервера

```
using System.Net;
using System.Net.Sockets;

Int32 port = 13000;

IPAddress localAddr = IPAddress.Parse("127.0.0.1");

TcpListener server = new TcpListener(localAddr,
  port);

server.Start();

//Начинаем прослушивание порта
TcpClient client = server.AcceptTcpClient();
//После подключения создаем поток сообщений
NetworkStream stream = client.GetStream();
```

Обмен сообщениями

Запись сообщений

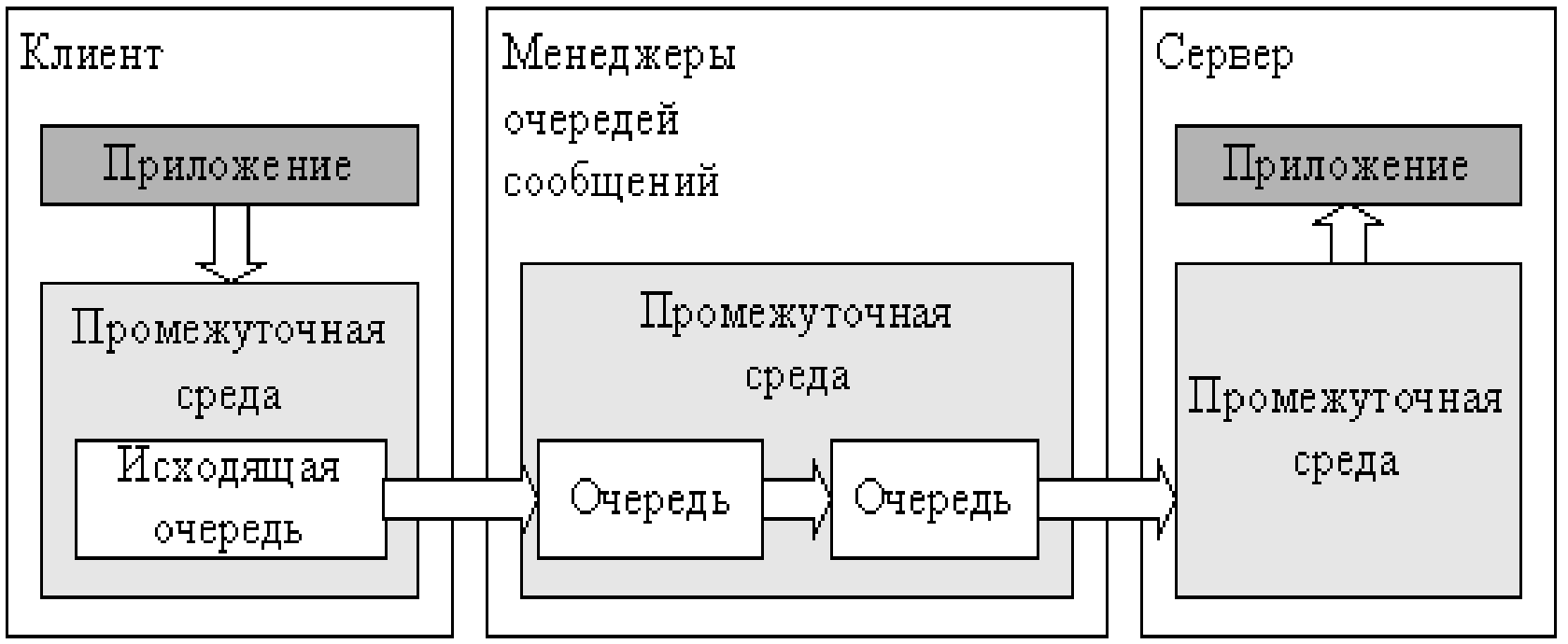
```
Byte[] bytes=new Byte[256];

String data = “text”;

bytes =
  System.Text.Encoding.UTF.Ge
  tBytes(data);

stream.Write(bytes, 0,
  bytes.Length);
```

Чтение сообщений

```
Byte[] bytes=new Byte[256];

String data = null;

int i = stream.Read(bytes, 0,
  bytes.Length);


data=System.Text.Encoding.UTF
  8.GetString(bytes,0, i);
```

Взаимодействие посредством менеджера сообщений

Использование менеджера сообщений



Использование менеджера сообщений

Использование очередей сообщений ориентировано на **асинхронный** обмен данными.

Достоинства

- время функционирования сервера не связано со временем работы клиентов;
- независимость промежуточной среды от средства разработки компонент и используемого языка программирования;
- считывать и обрабатывать заявки из очереди могут несколько независимых компонент, что дает возможность достаточно просто создавать устойчивые и масштабируемые системы.

Использование менеджера сообщений

Недостатки

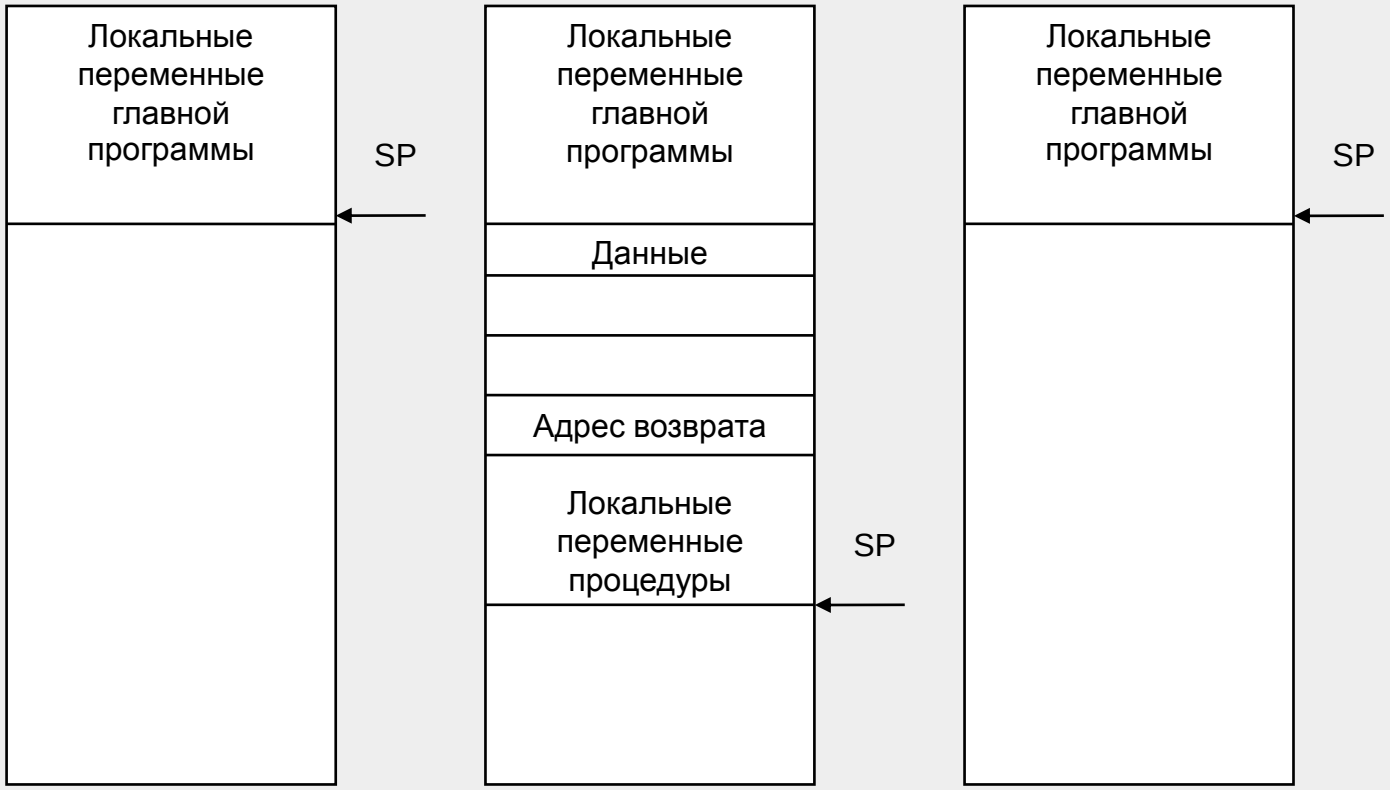
- необходимость явного использования очередей распределенным приложением;
- сложность реализации **синхронного** обмена;
- определенные накладные расходы на использование менеджеров очередей;
- сложность получения ответа: передача ответа может потребовать отдельной очереди на каждый компонент, посылающий заявки.

Удаленный вызов процедур
remote procedure call – RPC
remote method invocation - RMI

Технология RPC

- **Удаленный вызов процедур** (от англ. Remote Procedure Call, RPC) — технология, позволяющая компьютерным программам вызывать функции или процедуры в другом адресном пространстве.

Состояние стека при вызове локальной процедуры



До выполнения вызова

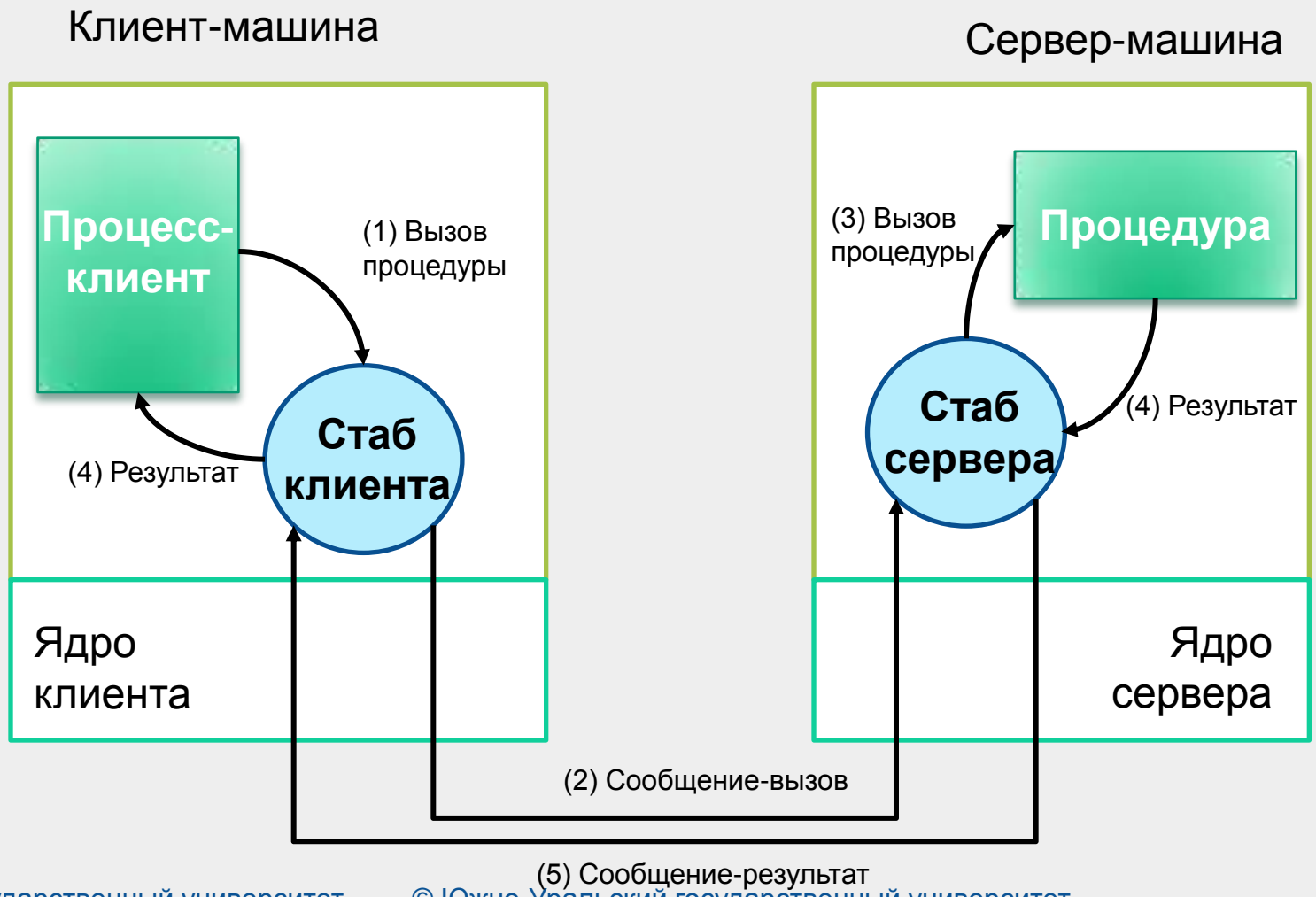
Во время выполнения

После возврата

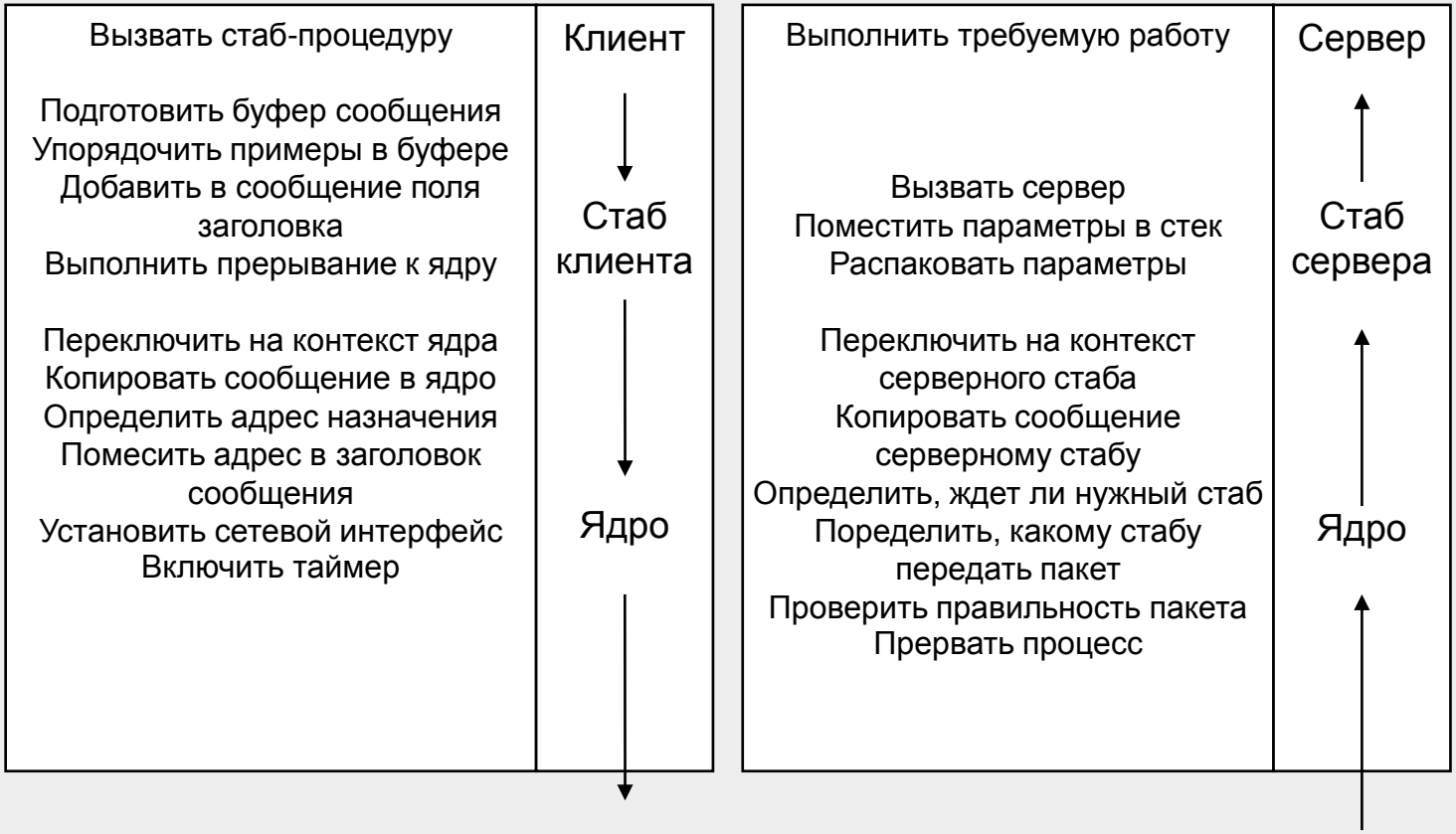
Реализация RPC

- Идея: вызов удаленной процедуры «прозрачен» для локального процесса
- Вместо локальной процедуры помещается «клиентский **стаб**» (**stub** – заглушка).
- Он вызывается также, как и локальная процедура, но вместо исполнения производит передача сообщения ядру удаленной машины.

Порядок вызова удаленной процедуры



Этапы выполнения RPC



Удаленный вызов методов

С точки зрения ООП была реализована концепция использования удаленных объектов Remote Method Invocation (RMI).

- **RMI** позволяет обеспечить прозрачный доступ к методам удаленных объектов, обеспечивая
  - доставку параметров вызываемого метода,
  - сообщение объекту о необходимости выполнения метода
  - и передачу возвращаемого значения клиенту обратно

Удаленный объект

- Удаленный объект – это совокупность некоторых данных, определяющих его состояние. Это состояние можно изменить путем вызова некоторых его методов.
- Методы и поля объекта, которые могут использоваться через удаленные вызовы доступны через **внешний интерфейс** класса объекта.

Посредник (proxy) и каркас

- Клиентская заглушка для вызова удаленного объекта называется **посредником** (**proxy**).
- **Посредник** реализует тот же интерфейс, что и удаленный объект.
- Заглушка на стороне **сервера** называется **каркасом** (skeleton в Java RMI)
- **Каркас** связывается с определенным экземпляром удаленного объекта и вызывает необходимый метод с требуемыми параметрами

Маршализация

- Для передачи параметров по сети используется **маршализация** (marshalling) - процесс преобразования параметров для передачи их между процессами при удаленном вызове
- Маршализация
  - **по ссылке** – экземпляр удаленного объекта находится на сервере и не покидает его, а для доступа используются посредники
  - **по значению** – удаленный объект сериализируется и его копия передается в другой процес

Использование удаленного объекта

Заключение

- Определено понятие протокола и дан обзор основных протоколов взаимодействия распределенных вычислительных систем.
- Дано описание основных методов взаимодействия распределенных вычислительных систем
  - Прямое взаимодействие (на примере сокетов)
  - Взаимодействие посредством менеджера сообщений
- Приведено описание технологий удаленного вызова процедур и удаленного вызова методов.