Основы создания сетевых приложений

Лекция 8

УНЦ «Инфоком»
Самара
2013

NetCracker®

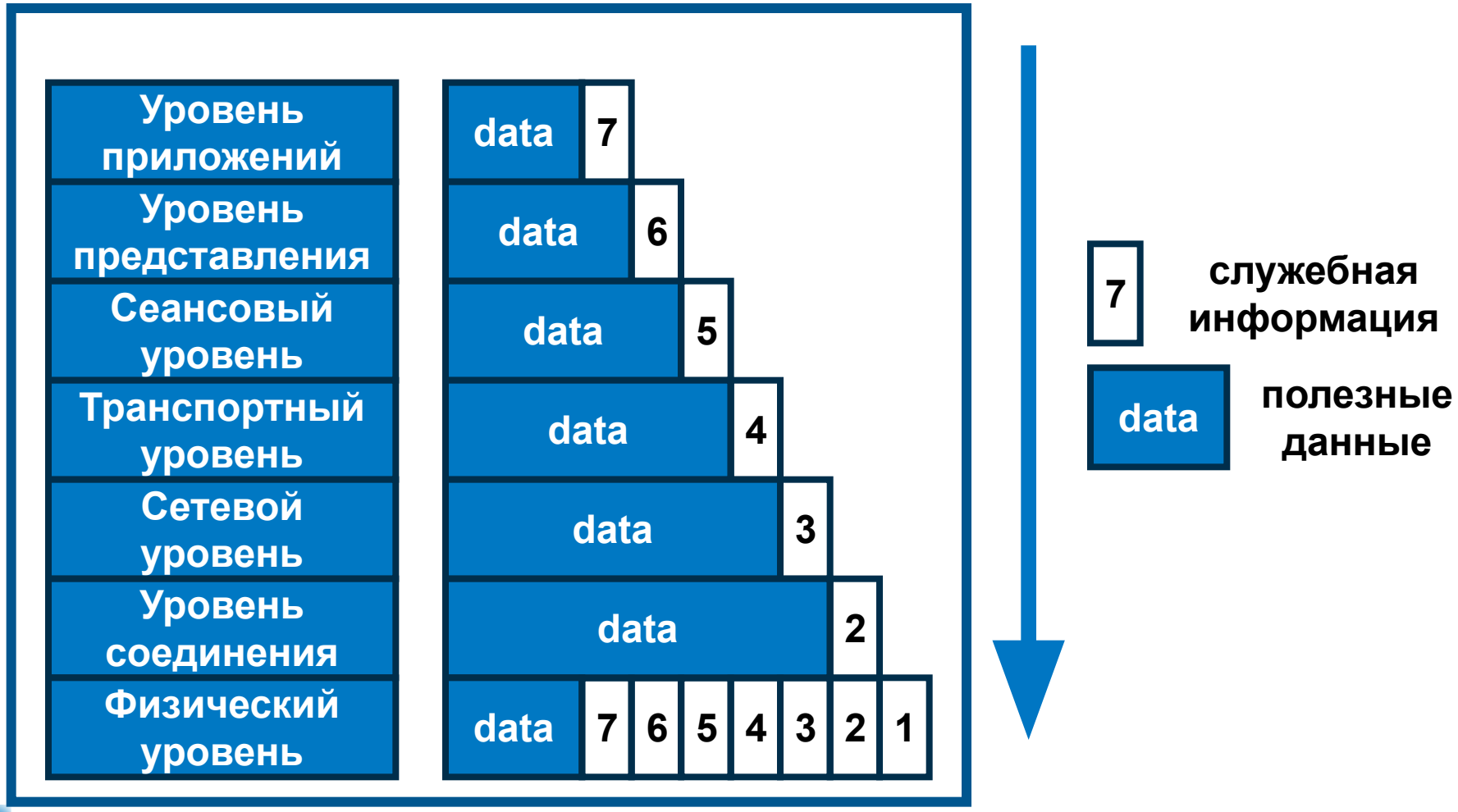
План лекции

- Протоколы транспортного уровня
- Сокеты
- Использование сокетов на Java
- Класс URL и его использование

Модель OSI

- Уровень приложений (layer 7, data)
- Уровень представления (layer 6, data)
- Сеансовый уровень (layer 5, data)
- Транспортный уровень (layer 4, segment)
- Сетевой уровень (layer 3, packet)
- Уровень соединения (layer 2, frame)
- Физический уровень (layer 1, bit)

Инкапсуляция пакета



Передача сообщения по сети

- Сообщение состоит из заголовка и данных
- Для каждого следующего уровня сообщение предыдущего уровня представляется как единое целое
- На физическом уровне сообщение содержит информацию всех семи уровней, кодируется и передается получателю

Transmission Control Protocol

- TCP – основанный на соединениях протокол, обеспечивающий надежную передачу данных между двумя компьютерами, с сохранением порядка следования данных

- Используется в: HTTP, FTP, Telnet и др.

User Datagram Protocol

- UDP – не основанный на соединениях протокол, реализующий пересылку независимых пакетов данных, называемых дейтаграммами, от одного компьютера к другому без гарантии их доставки

Основные характеристики TCP и UDP

TCP	UDP
Для работы устанавливает соединение	Работает без установления соединения
Гарантированная доставка	Гарантий доставки нет
Передача сообщения отдельными сегментами	Передает сообщения целиком в виде датаграмм
При получении сообщение собирается из сегментов	Принимаемые сообщения не объединяются
Пересылает заново потерянные сегменты	Подтверждений о доставке нет
Контроль потока сегментов	Без контроля потока датаграмм

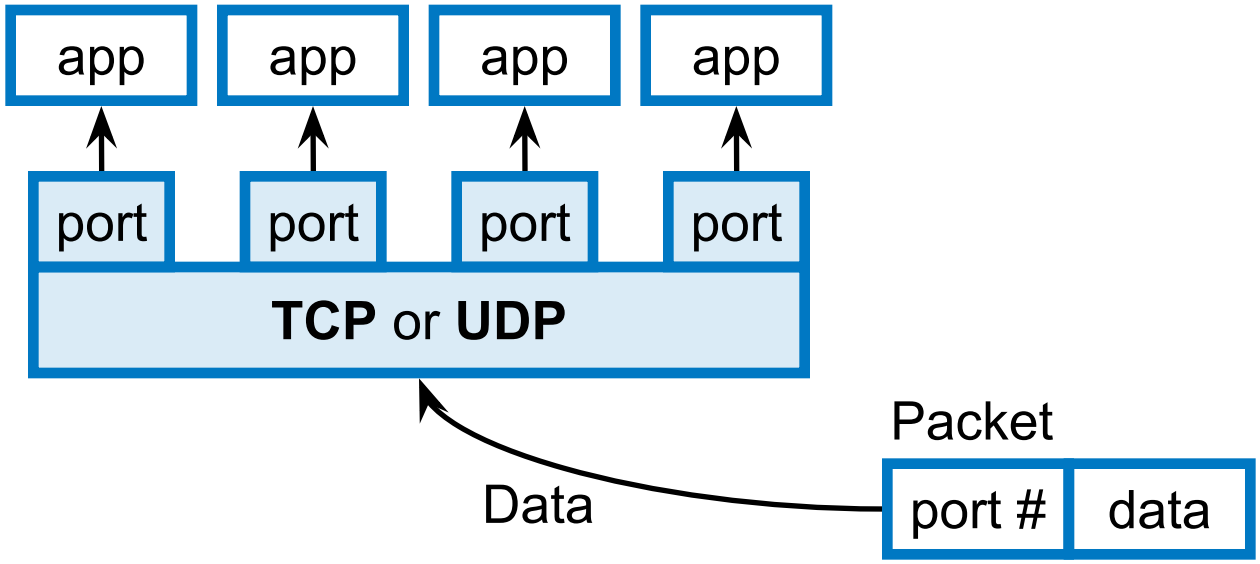
Модель «Клиент-сервер»

- Порядок работы
  - Каждая из сторон виртуального соединения называется «сокет» (socket)
  - Приложение-сервер инициализируется при запуске и далее бездействует, ожидая поступления запроса от клиента
  - Процесс-клиент посылает запрос на установление соединения с сервером, требуя выполнить для него определенную функцию
- Виды приложений-серверов
  - Сервер последовательной обработки запросов
  - Сервер параллельной обработки запросов

Понятие порта

- Компьютер (обычно) имеет только одно физическое соединение с сетью
- Соединение описывается, например, IP-адресом (32 бита на нынешний момент)
- Как различать информацию для различных приложений?

Понятие порта



- Сокет привязывается к порту
- Порт описывается 16-битным числом
- Порты 0-1023 зарезервированы

Интерфейс сокетов

- В 80-ых годах американское правительственное агентство по поддержке исследовательских проектов (ARPA), финансировало реализацию протоколов TCP/IP для UNIX в Калифорнийском университете в г. Беркли
- Разработан интерфейс прикладного программирования для сетевых приложений TCP/IP (TCP/IP API)
- TCP/IP sockets или **Berkeley sockets**

Связь с файловой системой

- Интерфейс сокетов – через системные вызовы UNIX
- Системные вызовы ввода-вывода UNIX выглядят как последовательный цикл:
  - открыть
  - считать/записать
  - закрыть
- Нет различий между файлами и внешними устройствами

Проблемы сетевого ввода/вывода

- Модель клиент-сервер не соответствует системе ввода-вывода UNIX
  - Не умеют устанавливать соединения
  - Используется фиксированный адрес файла
  - Соединение с файлом доступно на протяжении всего цикла запись-считывание
- Для не ориентированных на соединение протоколов фиксированный адрес – проблема: при передаче дейтаграммы адрес есть, а соединения нет

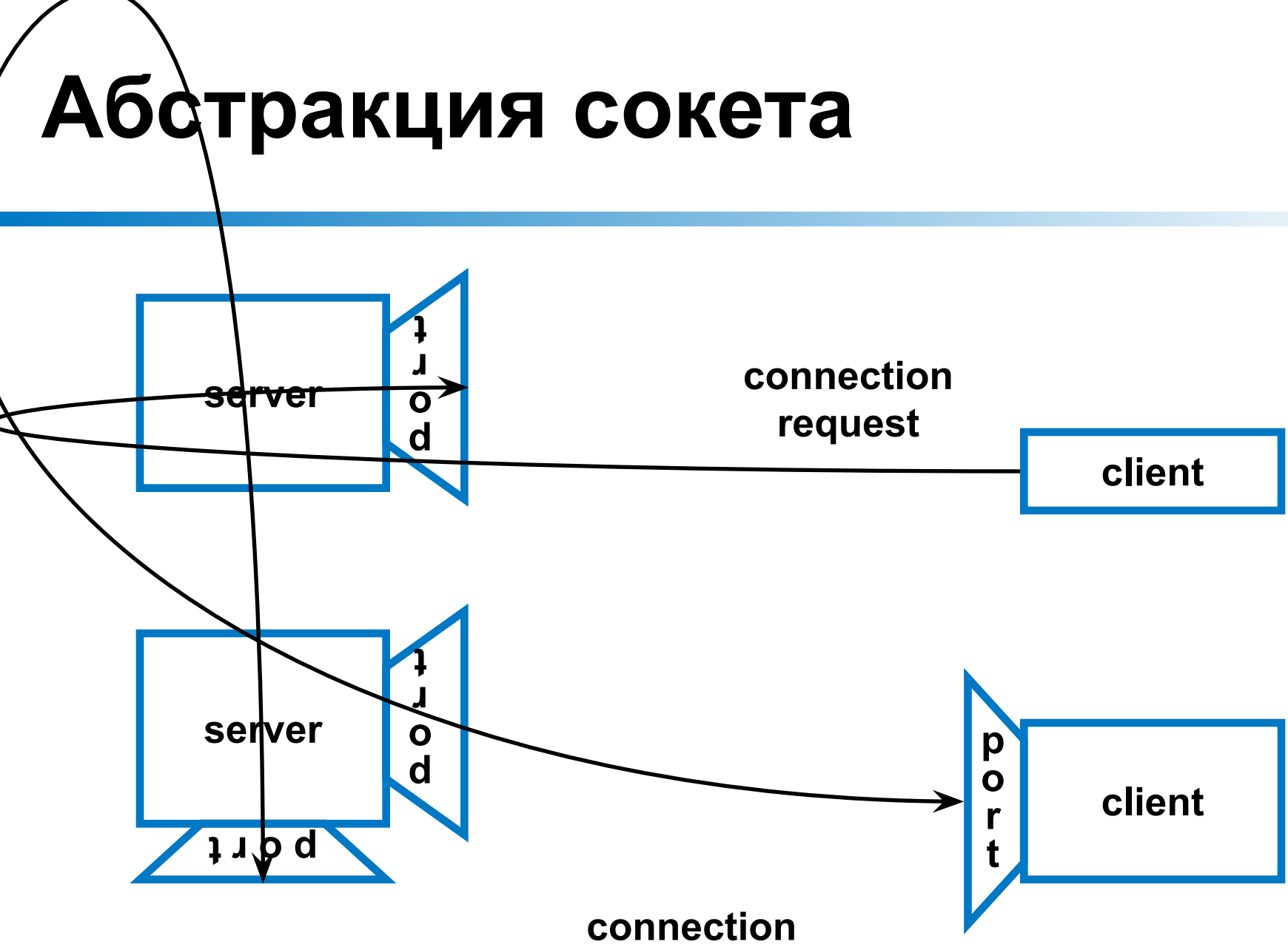
Абстракция сокета

- Сетевое соединение – это процесс передачи данных по сети между двумя компьютерами или процессами
- Сокет – конечный пункт передачи данных
- Для программ сокет – одно из окончаний сетевого соединения
- Для установления соединения каждая из сетевых программ должна иметь свой собственный сокет

Абстракция сокета

- Связь между двумя сокетами может быть ориентированной на соединение
- Связь между двумя сокетами может быть не ориентированной на соединение
- Сокет связан с номером порта

Абстракция сокета



А что же на Java?

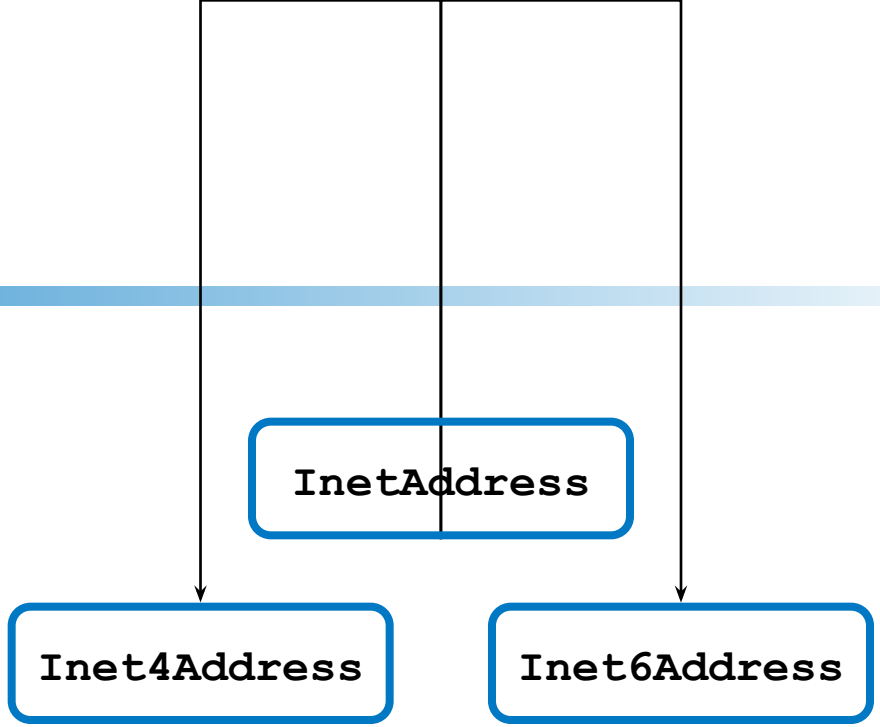
- Сокеты инкапсулированы в экземпляры специальных классов
- Все низкоуровневое взаимодействие скрыто от пользователя
- Существует семейство классов, обеспечивающих настройку сокетов и работу с ними

Пакет java.net

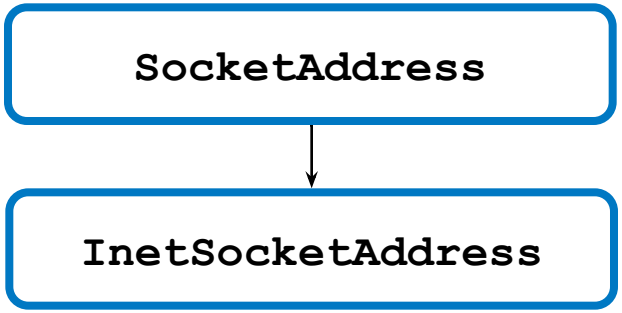
- Адресация
- Установление TCP-соединения
- Передача/прием дейтаграмм через UDP
- Обнаружение/идентификация сетевых ресурсов
- Безопасность: авторизация / права доступа

Адресация

- IP-адресация



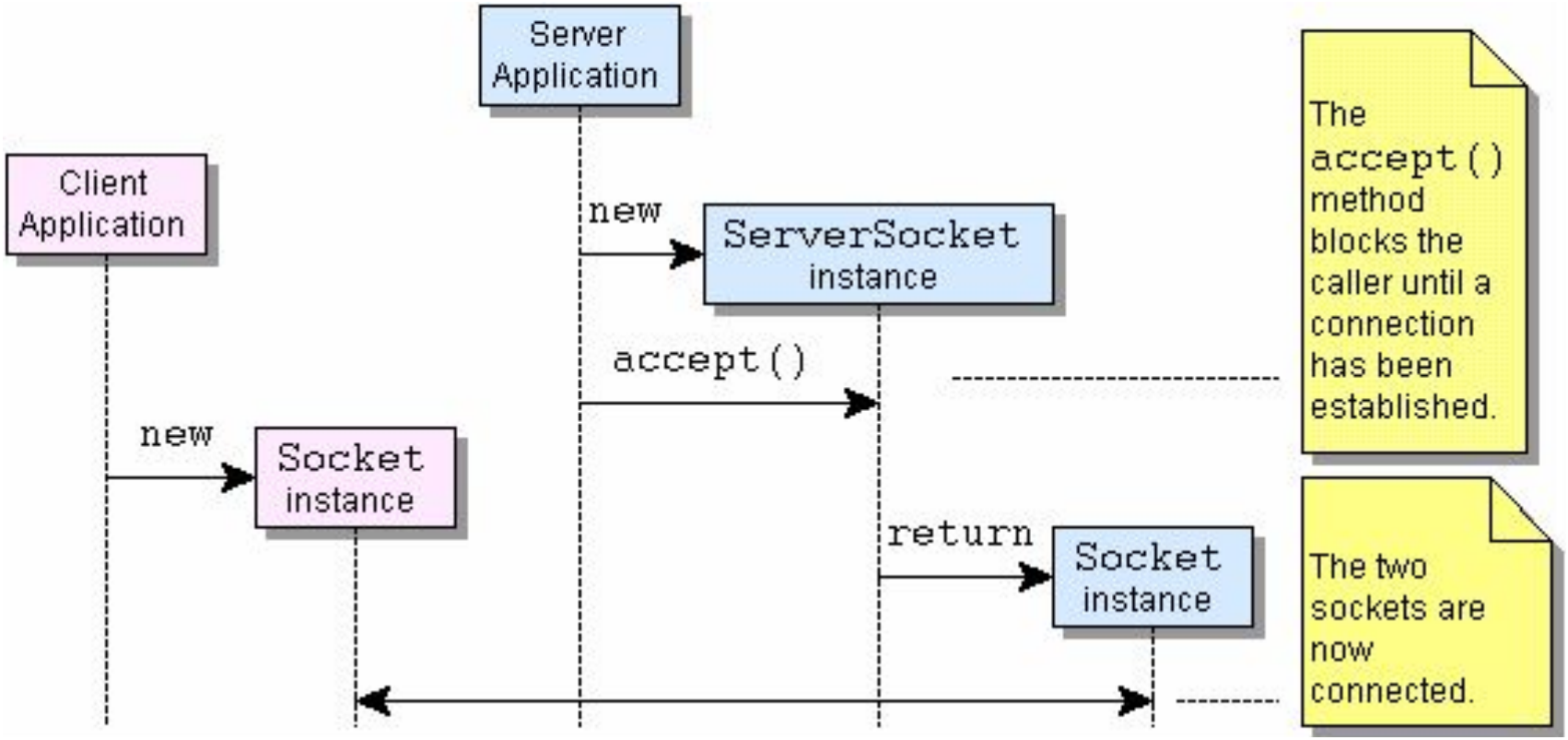
- Адрес сокета



Методы класса InetAddress

- `public static InetAddress getLocalHost();`
  создает объект класса для текущего локального узла
- `public static InetAddress getByName(String host);`
  создает объект адреса по имени удаленного узла сети
- `public static InetAddress[] getAllByName(String h);`
  возвращает массив адресов, связанных с узлом сети
- `public byte[] getAddress();`
  возвращает массив из четырех байт IP-адреса объекта
- `public String getHostName();`
  определение имени узла данного объекта адреса

Общая схема соединения



Класс Socket

- Реализует клиентский сокет и его функции
- Конструкторы
  - `Socket()`
  - `Socket(InetAddress address, int port)`
  - `Socket(InetAddress address, int port, InetAddress localAddr, int localPort)`
  - `Socket(String host, int port)`
  - `Socket(String host, int port, InetAddress localAddr, int localPort)`
- Методы
  - `void close()`
  - `InetAddress getLocalAddress()`
  - `InputStream getInputStream()`
  - `OutputStream getOutputStream()`
  - `static void setSocketImplFactory(SocketImplFactory fac)`
  - И прочие…

Порядок работы с клиентским сокетом

- Открытие сокета
- Открытие потока ввода и/или потока вывода для сокета
- Чтение и запись в потоки согласно установленному протоколу общения с сервером
- Закрытие потоков ввода-вывода
- Закрытие сокета

Пример клиента

```
public class EchoClient {
  public static void main(String[] args) throws IOException {
    Socket echoSocket = null;
    PrintWriter out = null;
    BufferedReader in = null;
    try {
      echoSocket = new Socket("taranis", 7);
      out = new PrintWriter(echoSocket.getOutputStream(), true);
       in = new BufferedReader(new
InputStreamReader(echoSocket.getInputStream()));
    } catch (UnknownHostException e) {
      System.err.println("Don't know about host: taranis.");
      System.exit(1);
    } catch (IOException e) {
      System.err.println("Couldn't get I/O for connection to
taranis.");
      System.exit(1);
    }
    ...
```

Пример клиента

```
    ...
    BufferedReader stdIn = new BufferedReader(new
          InputStreamReader(System.in));
    String userInput;

    while ((userInput = stdIn.readLine()) != null) {
      out.println(userInput);
      System.out.println("echo: " + in.readLine());
    }

    out.close();
    in.close();
    stdIn.close();
    echoSocket.close();
  }
}
```

Класс ServerSocket

- Реализует серверный сокет и его функции
- Конструкторы
  - `ServerSocket()`
  - `ServerSocket(int port)`
  - `ServerSocket(int port, int backlog)`
- Методы
  - `void close()`
  - `Socket accept()`
  - `void bind(SocketAddress endpoint)`
  - И прочие…

Создание серверного сокета

```
try {
    serverSocket = new ServerSocket(4444);
} catch (IOException e) {
    System.out.println(
        "Could not listen on port: 4444");
    System.exit(-1);
}
```

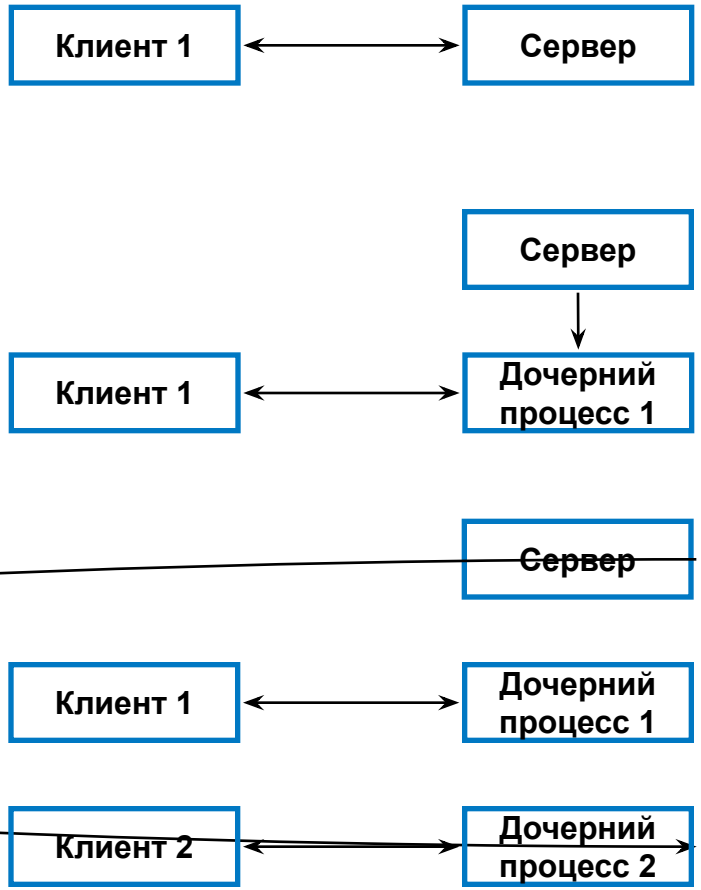
```
Socket clientSocket = null;
try {
    clientSocket = serverSocket.accept();
} catch (IOException e) {
    System.out.println("Accept failed: 4444");
    System.exit(-1);
}
```

Сервер параллельной обработки запросов

- **Стадия 1**
  Установление соединения клиент-сервер

- **Стадия 2**
  Сервер параллельной обработки передает управление дочернему процессу

- **Стадия 3**
  Если во время обработки запроса поступает запрос от другого клиента, сервер параллельной обработки передает управление новому дочернему процессу



Дейтаграммы

- **Дейтаграмма** – независимое, самодостаточное сообщение, посылаемое по сети, чья доставка, время (порядок) доставки и содержимое не гарантируются

- Могут использоваться как для адресной, так и для широковещательной рассылки

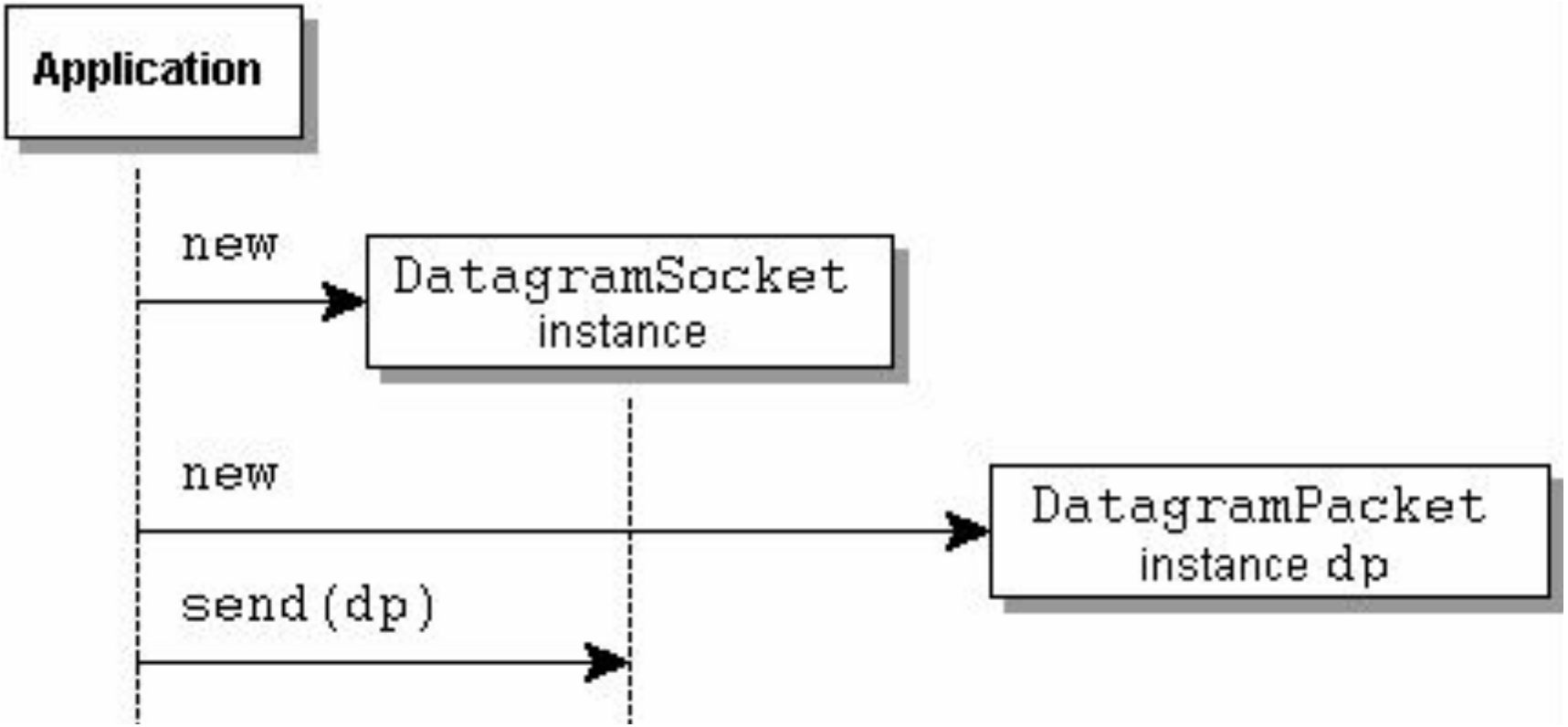
Класс DatagramPacket

- Экземпляры класса являются прототипами дейтаграмм-сообщений
- Конструкторы
  - `DatagramPacket(byte[] buf, int offset, int length, InetAddress address, int port)`
  - И прочие…
- Методы
  - `byte[] getData()`
  - `int getLength()`
  - `int getOffset()`
  - `SocketAddress getSocketAddress()`
  - `void setSocketAddress(SocketAddress address)`
  - `void setData(byte[] buf, int offset, int length)`
  - И прочие…

Класс DatagramSocket

- Экземпляры являются не ориентированными на соединение сокетами
- Конструкторы
  - `DatagramSocket()`
  - `DatagramSocket(int port, InetAddress laddr)`
  - И другие…
- Методы
  - `void bind(SocketAddress addr)`
  - `void close()`
  - `void connect(InetAddress address, int port)`
  - `void send(DatagramPacket p)`
  - `void receive(DatagramPacket p)`
  - И другие…

Передача дейтаграмм



Uniform Resource Locator

- URL – адрес ресурса в Интернет
- Имя протокола
  Протокол, используемый для связи
- Имя хоста
  Имя компьютера, на котором расположен ресурс
- Имя файла
  Путь к файлу на компьютере
- Номер порта
  Номер порта для соединения (необязателен)
- Ссылка
  Ссылка на обработчик работы с протоколом (необязательна)
- Может быть абсолютным и относительным

```
URL gamelan = new URL("http", "www.gamelan.com", 80,
                      "pages/Gamelan.network.html");
```

Прямое чтение из URL

```
import java.net.*;
import java.io.*;

public class URLReader {
  public static void main(String[] args) throws Exception
  {
    URL yahoo = new URL("http://www.yahoo.com/");
    BufferedReader in = new BufferedReader(
                            new InputStreamReader(
                              yahoo.openStream()));
    String inputLine;
    while ((inputLine = in.readLine()) != null)
      System.out.println(inputLine);
    in.close();
  }
}
```

Чтение из URL-соединения

```
import java.net.*;
import java.io.*;
public class URLConnectionReader {
  public static void main(String[] args) throws Exception
  {
    URL yahoo = new URL("http://www.yahoo.com/");
    URLConnection yc = yahoo.openConnection();
    BufferedReader in = new BufferedReader(
                            new InputStreamReader(
                              yc.getInputStream()));
    String inputLine;
    while ((inputLine = in.readLine()) != null)
      System.out.println(inputLine);
    in.close();
  }
}
```

Запись в URL-соединение

```
import java.io.*;
import java.net.*;
public class Reverse {
  public static void main(String[] args) throws Exception {
    if (args.length != 1) {
      System.err.println("Usage:  java Reverse" +
                         "string_to_reverse");
      System.exit(1);
    }
    String stringToReverse = URLEncoder.encode(args[0],
                                   "US-ASCII");
    URL url = new URL(
                "http://java.sun.com/cgi-bin/backwards");
```

Запись в URL-соединение

```
    URLConnection connection = url.openConnection();
    connection.setDoOutput(true);
    PrintWriter out = new PrintWriter(
                            connection.getOutputStream());
    out.println("string=" + stringToReverse);
    out.close();
    BufferedReader in = new BufferedReader(
                            new InputStreamReader(
                              connection.getInputStream()));
    String inputLine;
    while ((inputLine = in.readLine()) != null)
      System.out.println(inputLine);
    in.close();
  }
}
```

Спасибо за внимание!

NetCracker®

Дополнительные источники

- Арнолд, К. Язык программирования Java [Текст] / Кен Арнолд, Джеймс Гослинг, Дэвид Холмс. – М. : Издательский дом «Вильямс», 2001. – 624 с.
- Вязовик, Н.А. Программирование на Java. Курс лекций [Текст] / Н.А. Вязовик. – М. : Интернет-университет информационных технологий, 2003. – 592 с.
- Хорстманн, К. Java 2. Библиотека профессионала. Том 2. Тонкости программирования [Текст] / Кей Хорстманн, Гари Корнелл. – М. : Издательский дом «Вильямс», 2010 г. – 992 с.
- Эккель, Б. Философия Java [Текст] / Брюс Эккель. – СПб. : Питер, 2011. – 640 с.
- JavaSE at a Glance [Электронный ресурс]. – Режим доступа: http://www.oracle.com/technetwork/java/javase/overview/index.html, дата доступа: 21.10.2011.
- JavaSE APIs & Documentation [Электронный ресурс]. – Режим доступа: http://www.oracle.com/technetwork/java/javase/documentation/api-jsp-136079.html, дата доступа: 21.10.2011.