Лекция 3.
Сетевое программирование

Агенда

I. Как устроена сеть
II. Сетевое программирование в Java
    - InetAddress, URL
    - Потоки ввода/вывода
    - Сокеты
    - Принципы построения серверов
III. Неблокирующий ввод/вывод. Пакет java.nio

Прикладные задачи

Часть I

КАК УСТРОЕНА СЕТЬ

Информация

=

`byte[]`

Передача информации (TCP/IP)

Уровень приложения

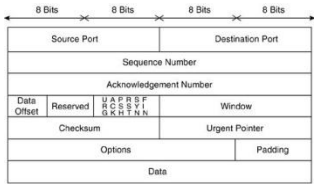
Транспортный уровень (TCP, UDP)

Сетевой уровень (IP)

«Компьютерные сети»

Передача информации (TCP/IP)

картинка



датаграмма

пакеты

010011
001110
100110

Адресация (почтовая)

От кого: Юрьев Юрий Юрьевич

Откуда: РФ, г. Москва, ул. Ежзикл,
д. 7, корп. 2, кв. 11

Кому: Иванов Иван Иванович

Куда: РБ, г. Минск, ул. Абвгд,
д. 12, кв. 58

220001

Адресация (TCP/IP)

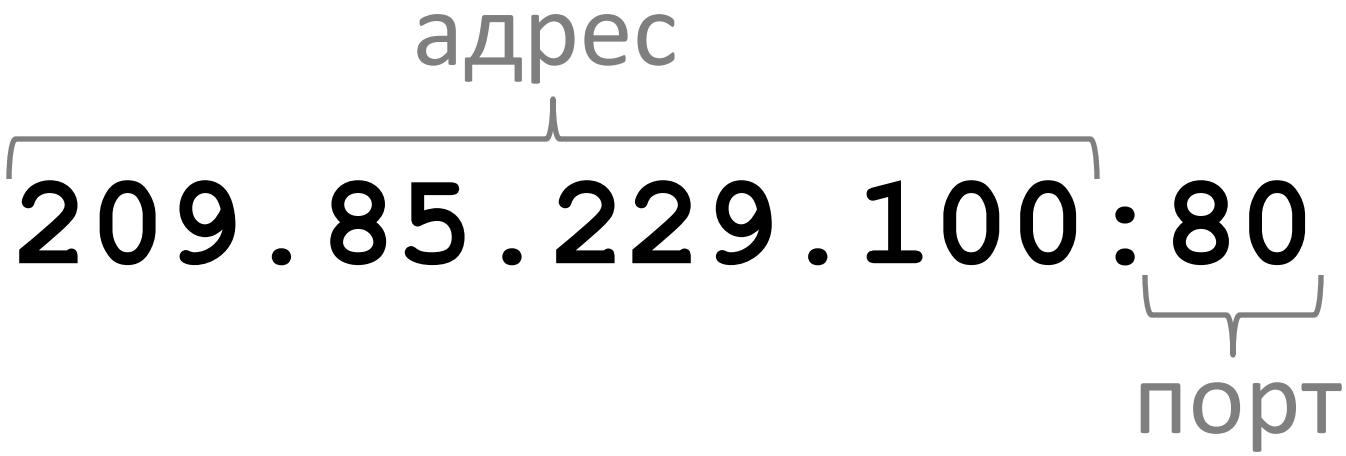
От кого:

179.11.5.230:45132

Кому:

209.85.229.100:80

Формат




127.0.0.1
loopback

Часть II

СЕТЕВОЕ ПРОГРАММИРОВАНИЕ В JAVA

Утилита

```
public static void out(String s) {
    System.out.println(s);
}
```

java.net.InetAddress

```
InetAddress[] all =
  InetAddress.getAllByName("google.com");
for (InetAddress address : all) {
  out(address.getHostAddress());
}
```

```
----------------------------------------
209.85.229.100
209.85.229.101
209.85.229.102
209.85.229.113
209.85.229.138
```

DNS – Domain Name System

java.net.InetAddress

```
InetAddress localhost =
  InetAddress.getLocalHost();

out(localhost.getHostAddress());
out(localhost.getHostName());
out(localhost.getCanonicalHostName());
```

```
----------------------------------------
192.168.1.56
YATSKEVICH
YATSKEVICH.americas.hpqcorp.net
```

URL — Uniform Resource Locator

или:

Единый указатель ресурсов

URL: Примеры

http://facebook.com

http://afisha.tut.by

http://ru.wikipedia.org/wiki/URL

file:///C:/Users/

file:///home/ivan/lecture_3.pdf

URL: Преимущество

Унифицированный доступ к ресурсам – нет разницы, находится ли он в сети либо на нашем компьютере

java.net.URL: Как создать

```
new URL("http://google.com");

new URL("http", //протокол
        "apple.com“, //домен
        "/favicon.ico");//файл
```

java.net.URL: Чтение

```
URL google = new URL("google.com");
InputStream in = google.openStream();
//читаем из потока in
```

```
----------------------------------------
<!doctype html><html itemscope
itemtype="http://schema.org/WebPage"><head
><meta http-equiv="content-type"
content="text/html; charset=windows-
1251"><meta itemprop="image"
co...........
```

java.net.URL: Чтение/запись

```
URL google = new URL("google.com");
URLConnection connection =
  url.openConnection();
InputStream in =
  connection.getInputStream();
OutputStream out =
  connection.getOutputStream();
```

Правильные URL

- Большие буквы `A-Z`
- Маленькие буквы `a-z`
- Цифры `0-9`
- `-_.!~*'(,)`

Имеют особое значение

`/ & ? @ # ; $ + = %`

http://www.google.com/#hl=en&

q=2+%2B+2+%3D+5+%3F

java.net.URLDecoder

```
String decodedQuery =
URLDecoder.decode(
        “q=2+%2B+2+%3D+5+%3F”,
        “UTF-8”);
out(decodedQuery);
```

---

q=2 + 2 = 5 ?

java.net.URLEncoder

```
String encodedQuery =
URLEncoder.encode(
        "q=2+2=5?", "UTF-8");
out(encodedQuery);
```

---

q%3D2%2B2%3D5%3F

Потоки ввода/вывода

OutputStream	**InputStream**
void write(int b)	int read()
void write(byte[] data)	int read(byte[] data)
void write(byte[] data, …)	int read(byte[] data, …)
void flush()	int available()
void close()	void close()

Имплементации

- FileInputStream/FileOutputStream
- ByteArrayInputStream/ByteArrayOutputStream
- url.openStream() //InputStream

Фильтрация потоков

- BufferedInputStream/BufferedOutputStream
- PushbackInputStream
- DataInputStream/DataOutputStream
- Фильтры для сжатия
- Digest фильтры
- Фильтры для шифрования

PrintStream

- Зависимость от платформы
  - Кодировка
  - Окончания строк (`\n` или `\r\n`)
- Обработка ошибок

Связывание потоков

```
OutputStream out =
new BufferedOutputStream(
    new FileOutputStream(
        “file.txt”));
```

Связывание потоков

```
FileOutputStream out =
new FileOutputStream("file.txt");
BufferedOutputStream bufferedOut =
new BufferedOutputStream(out);
//пишем в out
//пишем в bufferedOut
```

Readers/Writers

Если мы работаем с символами либо строками – используем обертки над потоками

Readers/Writers

```
OutputStream out = ...;
OutputStreamWriter writer = new
OutputStreamWriter(out, "UTF-8");
writer.write(
"Кириллица в нормальной кодировке"
);
```

Многопоточность

1. Как создать поток?

Как создать поток?

```
public class MyThread extends Thread {
  public void run() { ... }
}
```

```
public class MyHandler implements Runnable {
  public void run() { ... }
}
new Thread(new MyHandler());
```

Многопоточность

2. Как запустить задачу в отдельном потоке?

Как запустить задачу в отдельном потоке?

```
new MyThread().start();
```

```
Runnable job = new Runnable() {...};
executorService.execute(job);

Callable<Integer> task = new Callable<~>() {
  Integer call() { ... }
}
executorService.submit(job);
```

Многопоточность

3. Базовые средства синхронизации?

Базовые средства синхронизации?

```
public synchronized void doAction() { ... }
```

```
private final Object lock = new Object();
public void doAction() {
 synchronized (lock) { ... }
}
```

```
private volatile int count = 0;
```

Сокеты

- Сокеты ≈ Потоки ввода/вывода
- Сокет = соединение между двумя хостами
- 2 типа:
  - клиентский (`java.net.Socket`)
  - серверный (`java.net.ServerSocket`)

Сокет со стороны клиента

Главная задача – подключится к удаленному хосту

Сокет со стороны клиента: как создать

```
new Socket("berkeley.edu", 80);

InetAddress berkeley =
InetAddress.getByName("berkeley.edu");
new Socket(berkeley, 21);
```

Сокет со стороны клиента: запись/чтение

```
Socket client = new Socket(...);
in = client.getInputStream();//*
out = client.getOutputStream();//*
```

* – работаем
как с привычными потоками I/O, в том числе используем *буферизацию*

Сокет со стороны клиента: закрытие

```
Socket client = null;
try {
   client = new Socket(...);
   //общаемся с сервером
} finally {
   if (client != null) client.close();
}
```

Сокет со стороны клиента: полузакрытые сокеты

```
Socket client = ...;
client.shutdownInput();

// и/или

client.shutdownOutput();
```

Сокет со стороны сервера

Главная задача – слушать входящие соединения и обслуживать (serve) их

Сокет со стороны сервера: еще раз о портах

Порты от **0** до **1024** являются системными*

* – для запуска приложения могут понадобится права root

Сокет со стороны сервера: создание

```
new ServerSocket(7777);
```

Сокет со стороны сервера: создание

`BindException` – порт занят либо недостаточно прав на прослушивание порта

Сокет со стороны сервера: принимаем соединения

```
ServerSocket server = new ServerSocket(...);
while (true) {
   Socket client = server.accept();
   //общаемся к client сокетом
}
```

Взаимодействие клиента с сервером

```
ServerSocket server =
new ServerSocket(7777);
```

```
Socket client =
new Socket(host, 7777);
```

```
while (true) {
  Socket client =
server.accept();
```

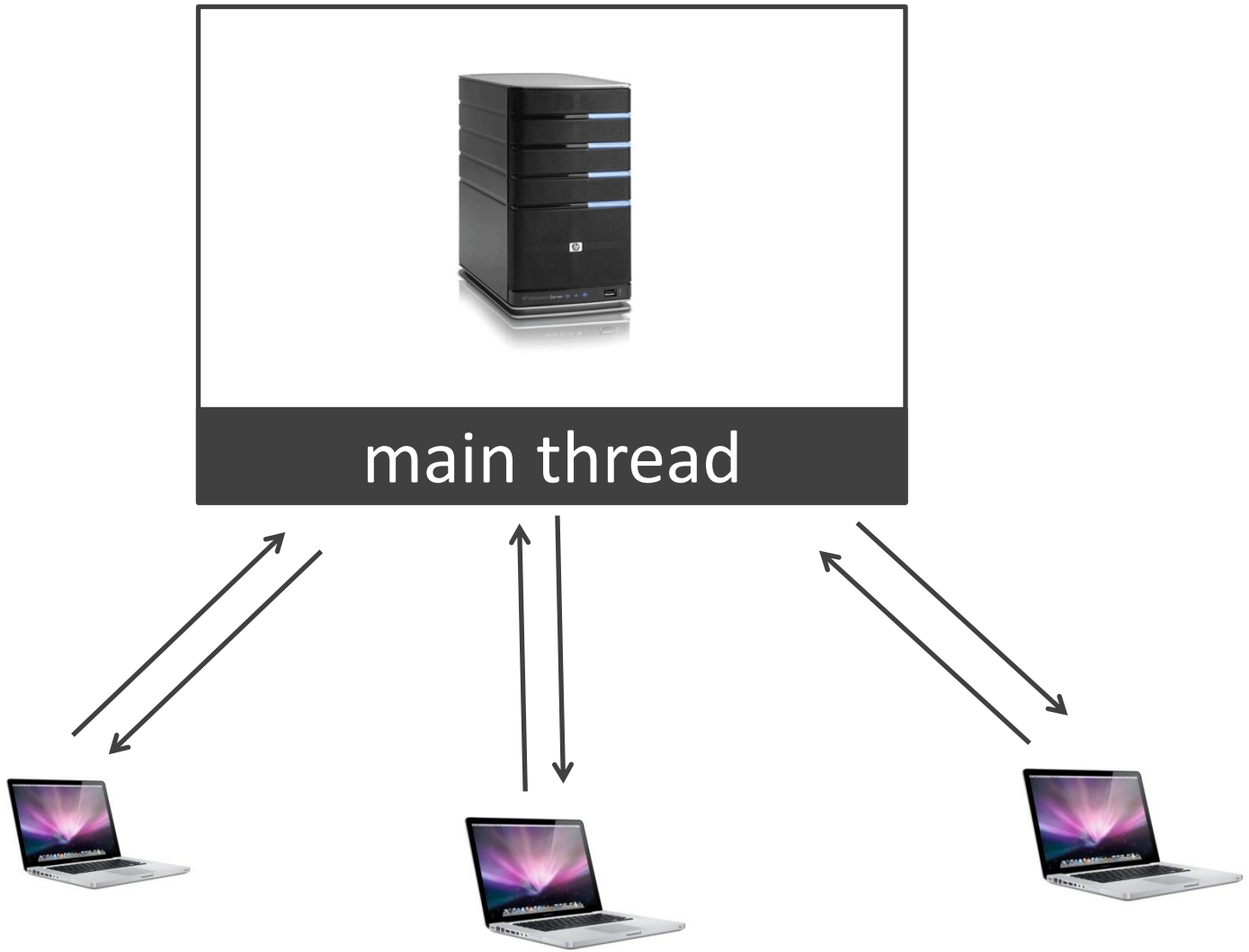
```
  //общаемся к client
сокетом
```

```
client.getInputStream();
client.getOutputStream();
```

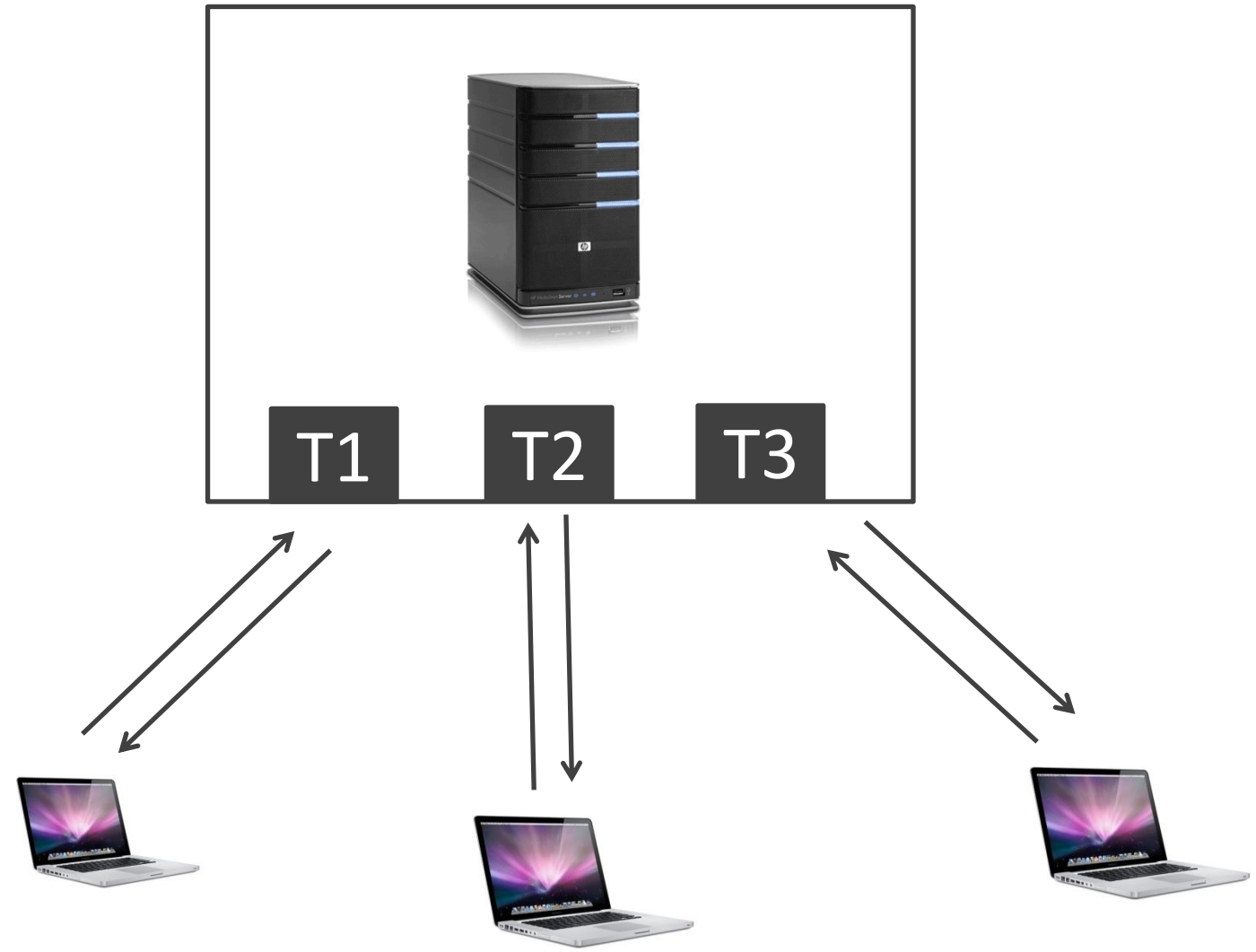
```
}
```

Взаимодействие клиента с сервером

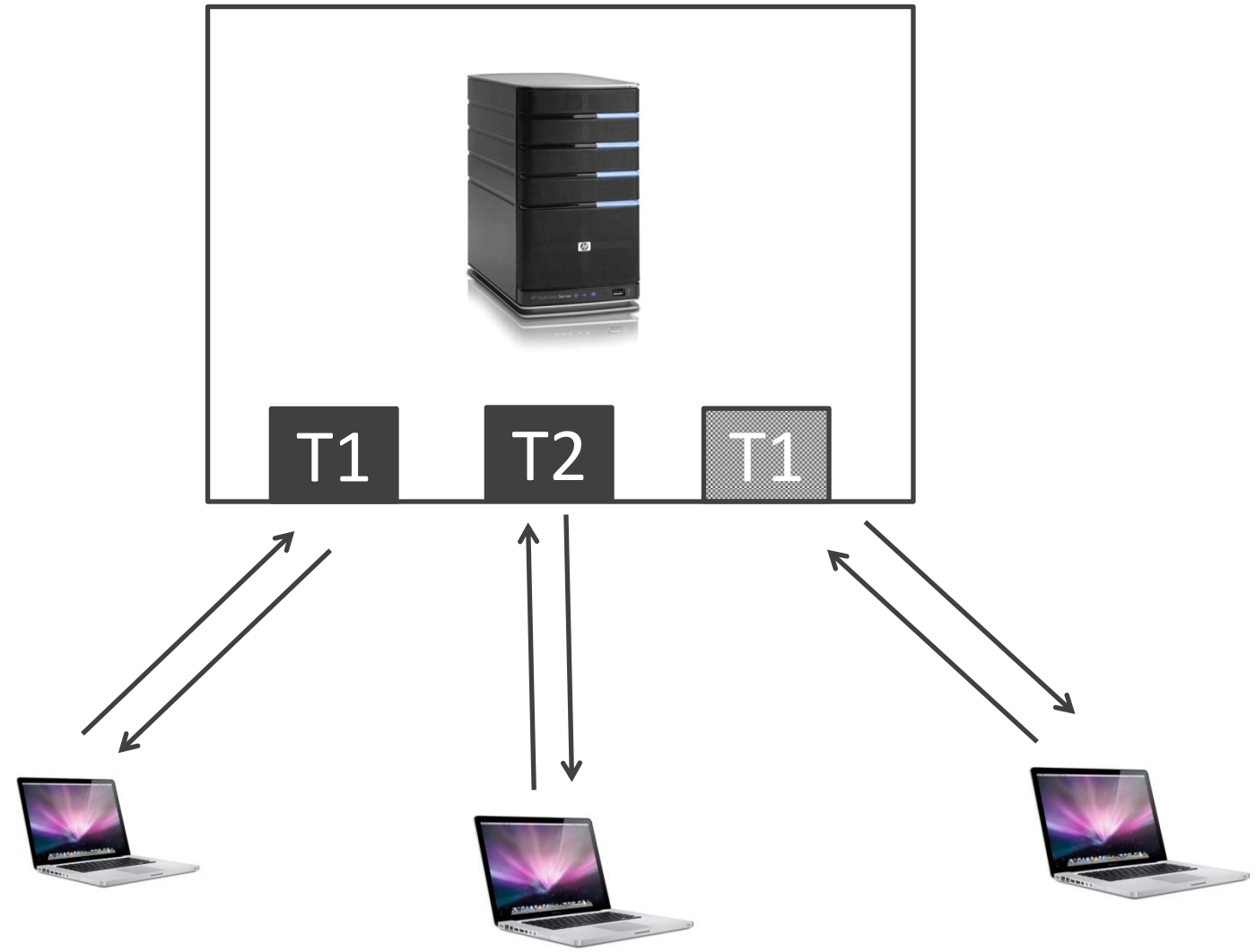
Модели построения серверов: один поток для всех клиентов



Модели построения серверов: поток для каждого клиента



Модели построения серверов: pool потоков



Часть III

НЕБЛОКИРУЮЩИЙ ВВОД/ВЫВОД. ПАКЕТ JAVA.NIO

Проблемы классических моделей

- Проблема C10K
- CPU, носители информации, сеть становятся быстрее со временем
- Затраты на порождение потоков (нужен ~1Mb RAM)
- Переключение между потоками (context switching)

Java решение – New I/O

- Нацелен на сервера с высокой нагрузкой
- Решает проблемы производительности

- *НО!* Немного сложнее в разработке

На стороне клиента

java.nio:

```
SocketAddress berkeley =
new InetSocketAddress("berkeley.edu",80);
SocketChannel client =
SocketChannel.open(berkeley);
```

Классический сокет:

```
InetAddress berkeley =
InetAddress.getByName("berkeley.edu");
Socket client =
new Socket(berkeley, 80);
```

На стороне клиента: чтение

java.nio:

```
SocketChannel client = ...;
ByteBuffer buffer =
ByteBuffer.allocate(1024);
client.read(buffer);//читаем в буфер
```

Классический сокет:

```
Socket client = ...;
in = client.getInputStream();
byte[] buffer = new byte[1024];
in.read(buffer);
```

На стороне клиента: запись

java.nio:

```
SocketChannel client = ...;
buffer.put(new byte[] {'a'});
buffer.flip();
client.write(buffer);//пишем из буфера
```

Классический сокет:

```
Socket client = ...;
out = client.getOutputStream();
buffer = new byte[] {'a'};
out.write(buffer);
```

Неблокирующие операции

java.nio:

```
SocketChannel client = ...;
client.configureBlocking(false);
while (buffer.hasRemaining() &&
     client.write(buffer) != -1) ;
```

Классический сокет:

```
Socket client = ...;
???
```

Channels vs. Streams

- Потоки I/O работают с байтами **последовательно**
- Каналы работают с **блоками** (наборами байт)

Buffers

- position
- capacity
- limit
- mark

Buffers

- ByteBuffer.*allocate*(512)
- buffer.flip()
- buffer.put(byte[] data)
- buffer.clear()

На стороне сервера

java.nio:

```
SocketAddress ftp = new InetSocketAddress(21);
ServerSocketChannel server =
ServerSocketChannel.open();
ServerSocket serverSocket = server.socket();
serverSocket.bind(ftp);
```

Классический сокет:

```
ServerSocket client = new ServerSocket(21);
```

Схема работы

- Создаем Selector

```
Selector sel = Selector.open()
```

- Регистрируем серверный канал

```
server.register(sel, SelectionKey.OP_ACCEPT)
```

- Выбираем готовые каналы

```
while (true) {
  sel.select();
  // --->
}
```

Схема работы (продолжение)

- Проходим по всем готовым каналам

```
Set<SelectionKey> readyKeys = s.selectedKeys();
Iterator iter = readyKeys.iterator();
while (iter.hasNext()) {
  SelectionKey key = iter.next();
  iter.remove();
  //непосредственно обработка
}
```

- Обрабатываем каналы

```
key.isAcceptable()
key.isWriteable()
key.isReadable()
```

Схема работы (продолжение)

- Получаем канал
  ```
  key.channel();//SocketChannel либо
  ServerSocketChannel
  ```
- Ассоциируем данные
  ```
  key.attach(data);
  ```
- Получаем прикрепленные данные
  ```
  key.attachment();
  ```
- Пишем/читаем

Спасибо!

Ваши вопросы

Материалы

1. Java Network Programming, 3rd Edition, Elliotte Rusty Harold
2. Custom networking tutorials by Oracle. http://docs.oracle.com/javase/tutorial/networking/
3. C10K Problem. http://www.kegel.com/c10k.html