

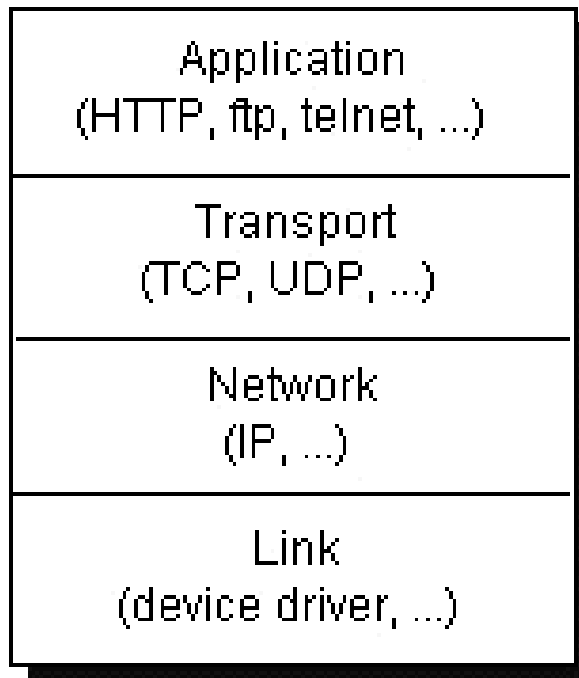


**СБЕРБАНК ТЕХНОЛОГИИ**

# **Java Networking with Sockets**

- Как приложения на разных компьютерах находят друг друга?
- Что такое хост, порт, сокет...?
- Как обмениваться данными по сети?

Компьютеры подключенные к Интернет общаются по протоколу TCP или UDP



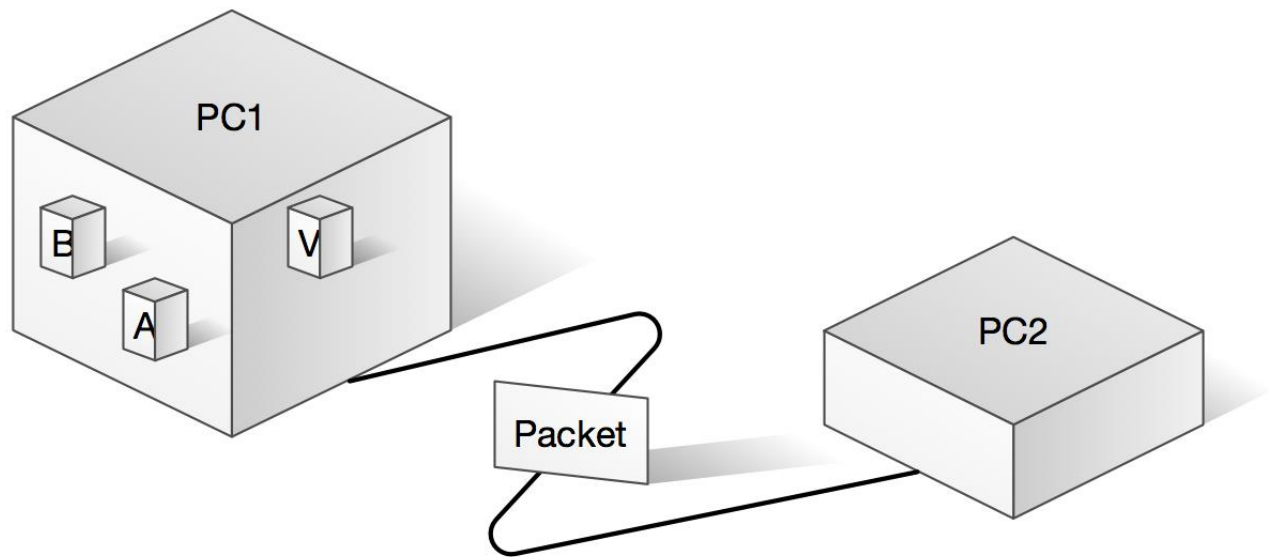
Сетевые приложения Java работают на прикладном уровне

- предоставляет поток данных с предварительной установкой соединения
- осуществляет повторный запрос данных в случае потери данных
- устраняет дублирование при получении двух копий одного пакета
- гарантирует целостность передаваемых данных и уведомление о результатах передачи.

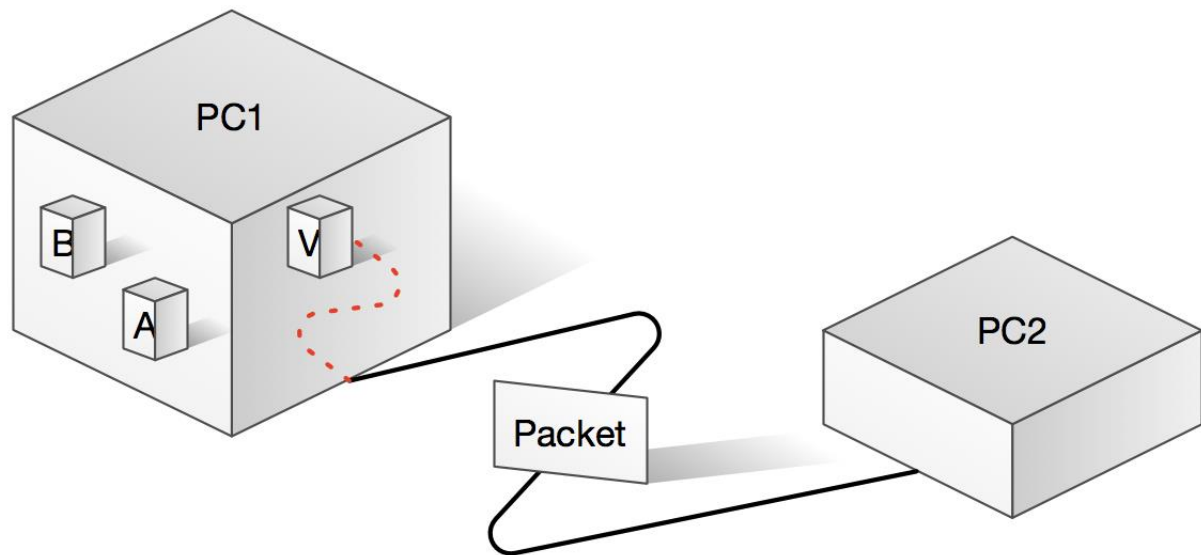
*Ex. HTTP/s, FTP, SSH, Telnet*

- предоставляет поток данных **без предварительной** установкой соединения
- без контроля над порядком передаваемых пакетов данных
- без контроля целостности данных (пакет может дублироваться или потеряться)

*Подходит для серверов, отвечающих на небольшие запросы от огромного числа клиентов или систем реального времени, где лучше сбросить пакет совсем чем получить его с задержкой*



Куда : 192.168.1.76 (32bit IP адрес)



Куда : 192.168.1.76 (32bit IP адрес)

Кому : 17764 (16bit порт)

Для работы с **TCP** и **UDP** протоколами в Java уже есть все необходимое

## TCP

`java.net.Socket`

`java.net.ServerSocket`

## UDP

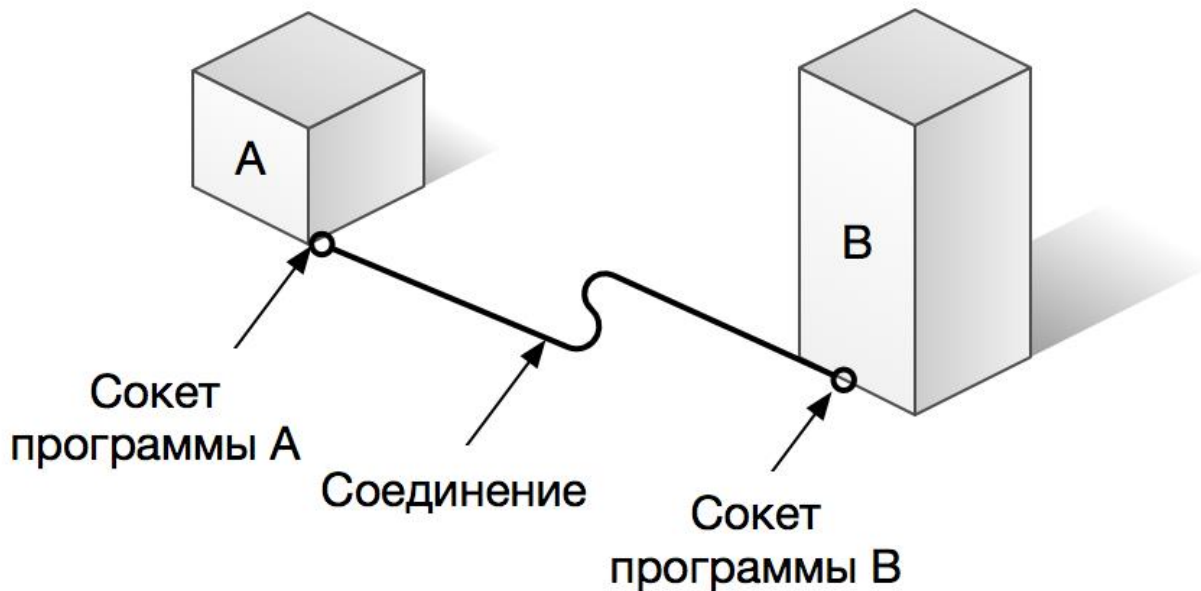
`java.net.DatagramPacket`

`java.net.DatagramSocket`

`java.net.MulticastSocket`



**Сокет** — абстрактный объект, представляющий конечную точку соединения



```
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;

public class ListenerForClientConnection {
    private static final int DEFAULT_PORT = 1234;

    public static void main(String[] args) throws IOException {
        int port = args.length == 0 ? DEFAULT_PORT : Integer.parseInt(args[0]);
        ServerSocket serverSocket = new ServerSocket(port); // регистрация сокета
        try (Socket clientSocket = serverSocket.accept()) { // ожидание соединения
            // работа с сокетом
        }
    }
}
```

```
mac:~ df$ telnet localhost 1234
Trying ::1...
Connected to localhost.
Escape character is '^]'.
Connection closed by foreign host.
mac:~ df$
```

```
import java.io.IOException;  
import java.net.Socket;
```

```
public class Client {  
    private static final String DEFAULT_HOST = "localhost";  
    private static final int DEFAULT_PORT = 1234;  
  
    public static void main(String[] args) throws IOException {  
        String serverHost = args.length > 0 ? args[0] : DEFAULT_HOST;  
        int serverPort = args.length > 1 ? Integer.parseInt(args[1]) : DEFAULT_PORT;  
        try (Socket server = new Socket(serverHost, serverPort)) { // подключиться к серверу  
            // работа с сокетом  
        }  
    }  
}
```

1. Открыть сокет
2. Открыть входящий и исходящие потоки
3. Считывать данные из входящего потока и писать во входящий поток в соответствии с правилами сервера (протокол)
4. Закрывать потоки
5. Закрывать сокет

Реализовать **сервер**, который будет возвращать текущее время на каждое подключение клиента

Реализовать **клиента**, который подключается к серверу, считывает строку и выводит ее на экран

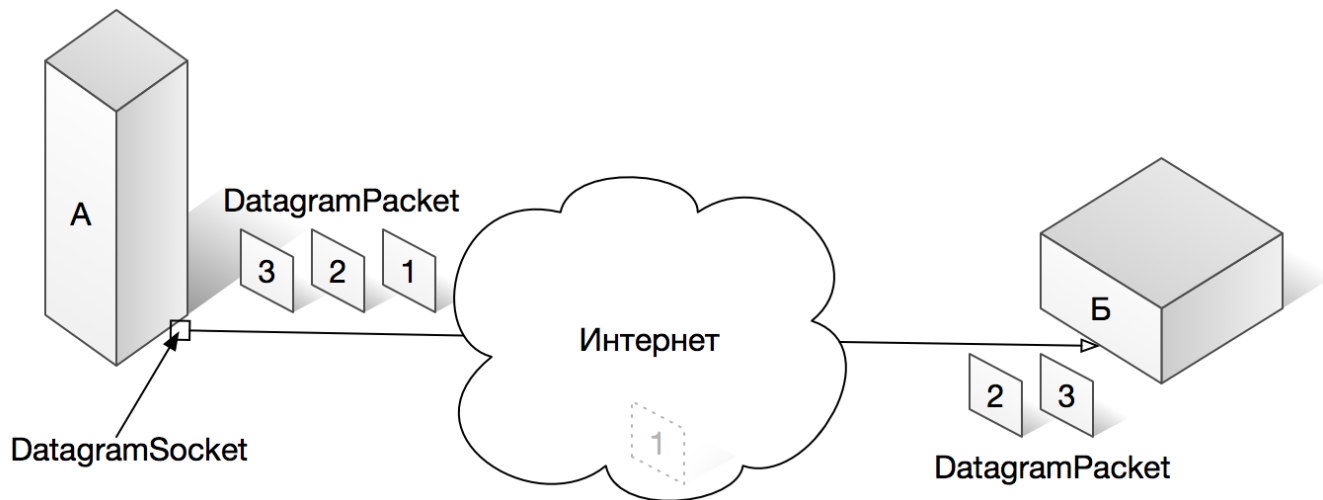
1. Создать серверный сокет
2. В цикле
  - а. Ожидать подключение клиента – получить клиентский сокет
  - б. Запустить общение с клиентом в отдельном потоке (*Thread*)

Разработать сервис «Гадалка» при подключении клиента сервис загадывает число и предлагает клиенту его угадать. Сервер считывает последовательно числа от клиента и отвечает клиенту угадал он или нет.

Сервис должен поддерживать одновременную работу с 10 клиентами.

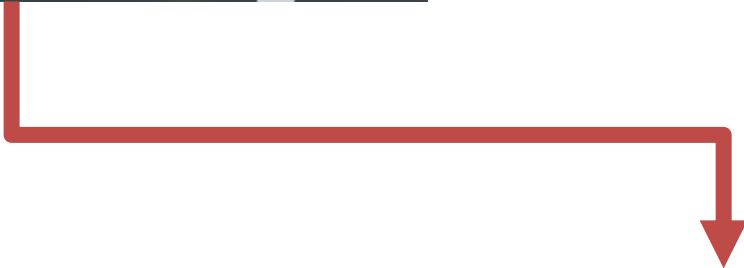


**Датаграмма** (*datagram*, **дейтаграмма**) — блок информации, передаваемый протоколом без предварительного установления соединения и создания виртуального канала.



```
InetAddress address = InetAddress.getByName(HOST);  
try (DatagramSocket socket = new DatagramSocket();  
    BufferedReader console = new BufferedReader(new InputStreamReader(System.in))) {  
  
    String line;  
    do { System.out.print("Enter message: ");  
        line = console.readLine();  
        byte[] bytes = line.getBytes();  
        DatagramPacket dp = new DatagramPacket(bytes, bytes.length, address, PORT);  
        socket.send(dp);  
    } while (!line.equals("exit"));  
}  
}  
}
```

```
Enter message: hello  
Enter message: world  
Enter message: exit
```



```
mac:~ df$ nc -ulc 0.0.0.0 -p 4321  
hello world  
exit  
read(net): Connection refused
```

```
try (DatagramSocket socket = new DatagramSocket(PORT)) {  
    byte[] buffer = new byte[BUFFER_SIZE];  
    String command = "";  
    do {  
        DatagramPacket datagram = new DatagramPacket(buffer, buffer.length);  
        socket.receive(datagram);  
        command = new String(datagram.getData(), 0, datagram.getLength());  
        System.out.print(command);  
    } while (!command.equals("exit"));  
}
```

Для отправки в группу теперь надо использовать специальный адрес доступный для широковещания. В сети IPv4 это любые адреса 224.0.0.0/8

```
InetAddress group = InetAddress.getByName("239.255.255.0");  
try (DatagramSocket socket = new DatagramSocket();  
    // ...  
    DatagramPacket dp = new DatagramPacket(bytes, bytes.length, group, PORT);  
    socket.send(datagramPacket);  
    // ...  
)
```

```
// -Djava.net.preferIPv4Stack=true
```

```
public static void main(String[] args) throws IOException {  
    try (MulticastSocket socket = new MulticastSocket(PORT)) {  
        InetAddress multicastAddress = InetAddress.getByName(MULTICAST_GROUP);  
        socket.joinGroup(multicastAddress);  
        byte[] buffer = new byte[BUFFER_SIZE];  
        String command = "";  
        do {  
            DatagramPacket datagramPacket = new DatagramPacket(buffer, buffer.length);  
            socket.receive(datagramPacket);  
            command = new String(datagramPacket.getData(), 0, datagramPacket.getLength());  
            System.out.print(command);  
        } while (!command.equals("exit"));  
        socket.leaveGroup(multicastAddress);  
    }  
}
```

- [http://www.tutorialspoint.com/java/java\\_networking.htm](http://www.tutorialspoint.com/java/java_networking.htm)
- <http://www.javaworld.com/article/2077322/core-java/core-java-sockets-programming-in-java-a-tutorial.html>
- <http://spec-zone.ru/Java/Docs/1.5.0/api/java/net/InetAddress.html>
- <https://ru.wikipedia.org/wiki/Мультивещание>
- <http://stackoverflow.com/questions/18747134/getting-cant-assign-requested-address-java-net-socketexception-using-ehcache>