DEV-J130. Java SE. Разработка многоуровневых приложений.
Основы сетевого программирования

© Ю.Д.Заковряшин, 2007-2022. E-mail: ydz@mail.ru

Рассматриваемые вопросы

- Общие принципы построения сетевых приложений.
- Обзор пакета java.net.
- Разработка сетевых приложений с использованием стеков протоколов UDP/IP и TCP/IP.
- Пример реализации сетевого приложения.

Общие термины и понятия

Сетевые протоколы

- Сетевой протокол — набор правил, соглашений и действий, определяющий соединение и обмен данными между включёнными в компьютерную сеть устройствами.
- Протокол передачи данных — набор определённых правил или соглашений интерфейса логического уровня, который определяет обмен данными между различными программами.
- Стек протоколов — это иерархически организованный набор сетевых протоколов, который используется для взаимодействия узлов в сети.

Эталонная модель OSI

- Эталонная OSI модель (Open Systems Interconnection model) – это логическая модель, состоящая из семи уровней, для каждого из которых определён отдельный набор сетевых функций. Модель разработана Международной Организацией по Стандартизации (International Standardization Organization – ISO) в 1984.

Сетевая адресация

- Сетевой адрес – идентификатор устройства, работающего в компьютерной сети.
- Наиболее часто используются следующие сетевые адреса:
  - MAC-адрес состоит из идентификатора производителя и номера устройства, присваиваемого устройству производителем;
  - IP-адреса, который состоит из адреса подсети и адреса устройства в подсети.

Сетевая адресация

- Доменные сетевые адреса – идентификаторы в виде символических (строковых) имён. Используются сетевыми протоколами DNS, WINS и т. п.
- URL (Uniform Resource Locator) – идентификатор установленной формы, определяющий положение какого-либо ресурса (файла, документа) в сети www (World Wide Web). Обобщенная спецификация URL имеет следующий вид:
  протокол:субпротокол//доменное_имя : порт/локальный_путь/имя_ресурса#раздел_документа

Сетевая адресация

- Порт программы – идентификатор, который может быть назначен процессу на компьютере для обеспечения сетевого взаимодействия. В большинстве ОС порт идентифицируется целым беззнаковым числом в диапазоне 0 – 65535.
- Сокет - специальная структура данных, содержащая сетевой адрес компьютера и номер порта на этом компьютере. Сокет позволяет адресовать в сети конкретный процесс на конкретном компьютере или ином сетевом устройстве.

IP протокол

- IP (Internet Protocol) – предназначен для определения адресата и доставки ему информации.
- IP предоставляет услугу для вышестоящих уровней, но не гарантирует целостность доставляемой информации.
- IP способен инкапсулировать другие протоколы, предоставлять место, куда они могут быть встроены.

Особые адреса в IP-протоколе

- Адрес сети – например, адрес локальной сети 192.168.0.0.
- Общий сетевой адрес (broadcasting address) – например, для локальной сети 192.168.255.255.
- Специальный адрес (loopback) – 127.0.0.1 (localhost).

Представление сетевых адресов

- В пакете java.net для представления сетевых адресов определены классы:
  - **InetAddress**
  - **Inet4Address**
  - **Inet6Address**
  - **SocketAddress**
  - **InetSocketAddress**
  - **URL**
  - **URI**

Класс InetAddress

- **Основные методы класса `InetAddress`**
- `static InetAddress getByName (String host);`
- `static InetAddress[] getAllByName (String host);`
- `static InetAddress getByAddress (byte[] addr);`
- `static InetAddress getLocalHost ();`
- `String getCanonicalHostName ();`
- `String getHostAddress();`
- `boolean isLoopbackAddress ();`
- `boolean isMulticastAddress ();`
- `boolean isReachable (int timeout );`

Класс URL

- **Основные методы класса URL:**
- `URL (String spec);`
- `URL (String protocol, String host, int port, String file);`
- `String getProtocol ();`
- `String getHost ();`
- `String getPath ();`
- `String getFile ();`
- `int getPort ();`
- `int getDefaultPort ();`
- `InputStream openStream ();`
- `Object getContent ();`

Стек протоколов UDP/IP

Протокол UDP

- Обмен данными по протоколу UDP осуществляется с помощью специальных пакетов – датаграмм (datagram).
- В рамках протокола UDP постоянное соединение между компьютерами в сети не устанавливается.
- Протокол UDP не гарантирует доставки пакета адресату.
- Протокол UDP не гарантирует сохранения порядка получения пакетов адресатом, поскольку каждый пакет может отправляться по индивидуальному маршруту.

Протокол UDP: Алгоритм обмена данными

- Создается экземпляр класса DatagramSocket с его привязкой к определенному локальному порту, а в случае такой необходимости и к локальному адресу.
- Для приема данных создается «пустой» экземпляра класса DatagramPacket с буфером заданного размера.
- Вызывается метод receive() класса DatagramSocket. После его завершения из пакета извлекаются данные, а также адрес и порт отправителя.
- Для отсылки данных создается другой экземпляр класса DatagramPacket и заполняется нужными данными. Кроме того, для данного пакета указывается адрес и порт назначения.
- Вызывается метод send() класса DatagramSocket.
- Сокет закрывается.

Обмен данными по протоколу UDP

- Основные классы:
  - `DatagramPacket`
  - `DatagramSocket`

Класс DatagramPacket

- Основные методы класса `DatagramPacket`:
  - `DatagramPacket(byte[] buf, int length);`
  - `DatagramPacket(byte[] buf, int length, InetAddress address, int port);`
  - `InetAddress getAddress();`
  - `int getPort();`
  - `SocketAddress getSocketAddress();`
  - `int getLength();`
  - `byte[] getData();`
  - `void setAddress(InetAddress iaddr);`
  - `void setData(byte[] buf);`
  - `void setPort(int port);`
  - `void setSocketAddress(SocketAddress address).`

Класс DatagramSocket

- Основные методы класса `DatagramSocket`:
  - `DatagramSocket()throws SocketException;`
  - `DatagramSocket (int port)throws SocketException;`
  - `void close();`
  - `boolean isClosed();`
  - `InetAddress getLocalAddress();`
  - `int getLocalPort();`
  - `void receive(DatagramPacket p) throws IOException;`
  - `void send(DatagramPacket p) throws IOException;`

Пример

- Разработка клиент-серверного приложения по загрузке файла на сторону клиента с использованием стека протоколов UDP/IP.
- Описание варианта использования:
  - клиент передаёт на сервер имя файла;
  - если файл на сервере присутствует, то сервер сообщает клиенту его размер и ждёт ответа о готовности принять файл;
  - в случае получения от клиента сообщения о готовности принять файл, сервер передаёт клиенту запрошенный файл;
  - если файл на сервере отсутствует или он имеет слишком большой размер (зависит от реализации), то сервер передаёт клиенту код ошибки.

Стек протоколов TCP/IP

Стек протоколов TCP/IP

- Стек протоколов TCP/IP (Transmission Control Protocol/Internet Protocol) — модель, описывающая процесс передачи цифровых данных по сетям.
- Модель выделяет 4 уровня протоколов взаимодействия:
  - прикладной уровень (соответствует прикладному уровню, уровню представления и сеансовому уровню модели OSI);
  - транспортный уровень;
  - сетевой уровень;
  - канальный уровень (соответствует канальному и физическому уровням модели OSI).

Сравнение моделей OSI и TCP/IP

Модель OSI	Модель TCP/IP
Прикладной уровень (application layer)	Прикладной уровень
Уровень представления (presentation layer)	
Сеансовый уровень (session layer)	
Транспортный уровень (transport layer)	Транспортный уровень
Сетевой уровень (network layer)	Сетевой уровень
Канальный уровень (data link layer)	Канальный уровень
Физический уровень (physical layer)	

Общие свойства

- Стек протоколов TCP/IP:
  - использует постоянное соединение узлов сети в процессе обмена данными;
  - гарантирует доставку пакета;
  - гарантирует порядок доставки пакетов.

Алгоритм работы на стороне клиента

- Создать экземпляр класса Socket;
- Получить ссылки на входной и выходной потоки класса Socket;
- Произвести операции чтения из входного потока сокет;
- Произвести запись в выходной поток сокета;
- Закрыть входной и выходной потоки сокета;
- Закрыть сокет.

Алгоритм работы на стороне сервера

- Создать экземпляр класса ServerSocket;
- Получить ссылку на экземпляр класса Socket с помощью метода accept();
- Получить ссылки на входной и выходной потоки класса Socket;
- Произвести операции чтения из входного потока сокета;
- Произвести запись в выходной поток сокета;
- Закрыть входные и выходные потоки сокета;
- Закрыть сокет, связанный с клиентом;
- Закрыть серверный сокет.

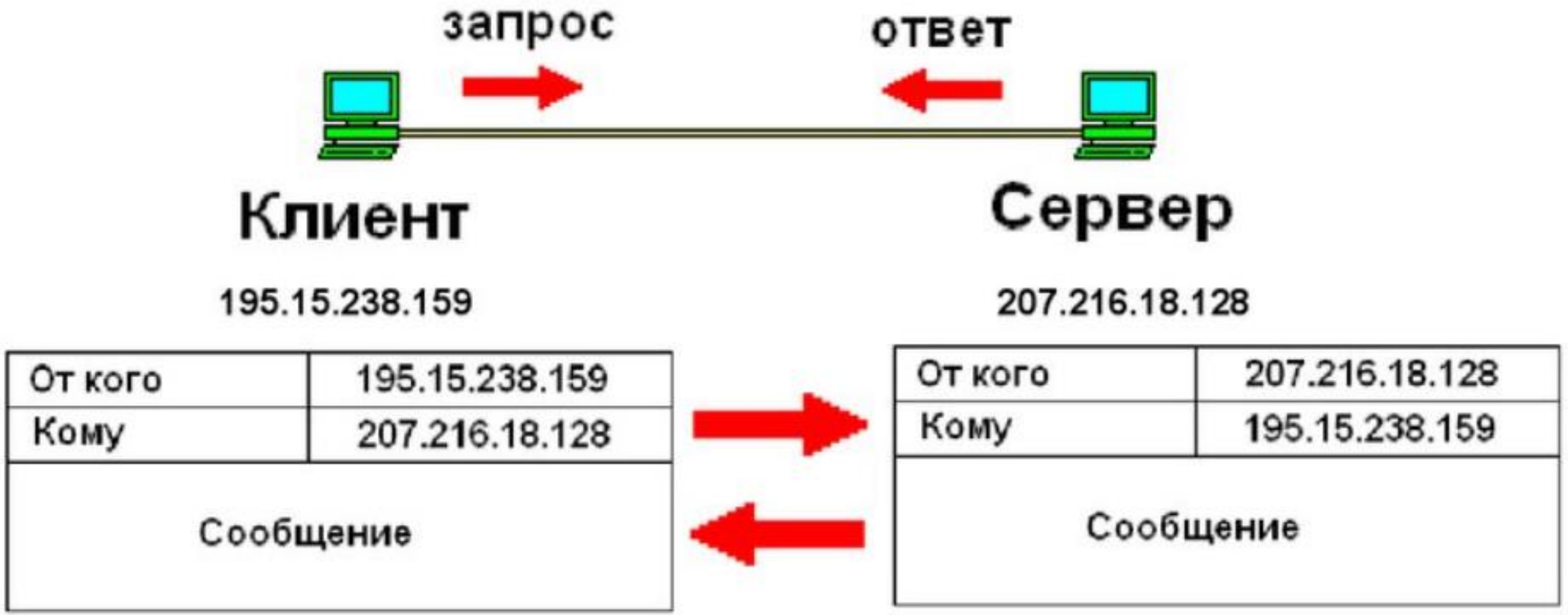
Процедура установки соединения

- Соединение узлов (A – узел-клиент, B – узел-сервер):
  1. узел A посылает узлу B специальный пакет SYN – запрос на соединение;
  2. узел B посылает в ответ пакет SYN-ACK, означающий готовность к соединению;
  3. узел A в ответ отсылает пакет ACK с подтверждением установки соединения.
- В процессе обмена синхронизируются начальные номера последовательности пакетов (Initial Sequence Number - ISN).

Процедура разрыва соединения

- Разрыв соединения узлов (A – узел, который инициирует разрыв, B – второй узел соединения):
  1. узел A посылает узлу B специальный пакет FIN – запрос на разрыв соединения;
  2. узел B в ответ отсылает пакет ACK с подтверждением разрыва соединения.
- После разрыва соединения на каждом из узлов освобождаются связанные с ним системные ресурсы.

Обмен данными



Протокол TCP

- Основные классы:
  - Socket;
  - ServerSocket.

Класс Socket

- Основные конструкторы класса:
  - `Socket ();`
  - `Socket (InetAddress host, int port);`
  - `Socket (String host, int port);`
  - `Socket(InetAddress address, int port, InetAddress localAddr, int localPort);`
  - `Socket(String address, int port, InetAddress localAddr, int localPort);`
  - `protected Socket(SocketImpl impl).`

Класс Socket

- Методы для управления соединением:
  - `void bind(SocketAddress bindpoint);`
  - `void connect(SocketAddress endpoint);`
  - `void connect(SocketAddress endpoint, int timeout);`
  - `void close() throws IOException.`

Класс Socket

- Диагностические методы класса:
  - `InetAddress getInetAddress();`
  - `int getPort();`
  - `InetAddress getLocalAddress();`
  - `int getLocalPort();`
  - `boolean isBound();`
  - `boolean isClosed();`
  - `boolean isConnected()`
  - `boolean isInputShutdown();`
  - `boolean isOutputShutdown();`

Класс Socket

- Методы для работы с входными и выходными потоками:
  - `InputStream getInputStream() throws IOException;`
  - `OutputStream getOutputStream() throws IOException;`
  - `void shutdownInput() throws IOException;`
  - `void shutdownOutput() throws IOException.`

Класс ServerSocket

- Основные конструкторы класса:
  - `ServerSocket();`
  - `ServerSocket(int port);`
  - `ServerSocket(int port, int backlog, InetAddress bindAddr).`

Класс ServerSocket

- Управление соединением:
  - `Socket accept();`
  - `void bind(SocketAddress endpoint);`
  - `void close();`

Класс ServerSocket

- Диагностические методы:
  - `InetAddress getInetAddress();`
  - `int getLocalPort();`
  - `SocketAddress getLocalSocketAddress();`
  - `boolean isBound();`
  - `boolean isClosed().`

Пример

- Разработка клиент-серверного приложения по загрузке файла на сторону клиента с использованием стека протоколов TCP/IP.
- Описание варианта использования:
  - клиент устанавливает соединение с сервером и передаёт на сервер имя файла;
  - если файл на сервере присутствует, то сервер сообщает клиенту его размер и ждёт ответа о готовности принять файл;
  - в случае получения от клиента сообщения о готовности принять файл, сервер передаёт клиенту запрошенный файл;
  - если файл на сервере отсутствует или он имеет слишком большой размер (зависит от реализации), то сервер передаёт клиенту код ошибки;
  - клиент разрывает соединение с сервером.

Заключение

- Обзор рассмотренных тем
- Вопросы