Розробка мережевих програм

Лекція 10.1

доц. кафедри Інформатики
Сінельнікова Т.Ф.

- Основи роботи в мережі
- Огляд сокетів
- Клієнт-сервер
- Зарезервовані сокети
- Приклад взаємозвязаних процедур клієнта
- Proxy-сервери
- Адресація Internet
- Служба доменних імен (DNS)
- Мережеві класи та інтерфейси
- Клас InetAddress
- Виробничі методи
- Сокети TCP / IP клієнтів
- Конструктори для створення soket-Об'єктів
- Використання URL
- Клас URLConnection
- Сокети TCP / IP серверів
- Конструктори класу ServerSocket
- Дейтаграми
- Клас DatagramPacket
- Дейтаграмний сервер і клієнт

Основи роботи в мережі

Кен Томпсон (Ken Thompson) І Денніс Рітчі (Dennis Ritchie) Розробили операційну систему UNIX спільно з мовою C в Bell Telephone Laboratories,Murray Hill,New Jersey, В 1969 р.
Протягом багатьох років розвиток UNIX виконувалося в Лабораторіях Белла, кількох університетах і науково-дослідних установах, що мали PDP-Машини фірми DEC, Для роботи на яких UNIX і була розроблена.
У 1978 р. Біл Джой (Bill Joy) Вів проект в каліфорнійському Університеті з метою додавання низки нових властивостей до UNIX, Таких як віртуальна пам'ять і повноекранні можливості дисплея.
На початок 1984 р., як раз, коли Білл покинув університет і заснував фірму Sun Microsystems, Він випустив систему 4.2BSD, Відому як Berkeley UNIX.
Система 4.2BSD була випущена з швидкою файлової системою, надійними сигналами, межпроцессорной зв'язком і, що найбільш важливо, що працює в мережі.
Berkeley-Реалізація протоколів TCP / IP залишається вихідним еталоном для зв'язку в Internet.
Парадигма сокета для межпроцессорной і мережевого зв'язку також була широко прийнята поза Berkeley.
Навіть Windows iMacintosh в кінці 80-х почали говорити " Berkeley-Сокети ".

Огляд сокетів

Мережевий сокет (network socket) Дуже схожий на електричний з'єднувач (роз'єм).

Різні мережеві з'єднувачі забезпечують стандартні шляхи поставки корисного навантаження.

Все, що розуміє стандартний протокол, може "підключатися" до сокету і вступати у зв'язок.

Для електричних з'єднувачів не має значення, підключаєте ви лампочку або тостер. До тих пір поки вони забезпечують подачу електрики 50 Гц, 220 В, пристрої будуть працювати.

Як створюється ваш рахунок за електрику.

Існує вимірювач десь між вашим будинком та іншою частиною мережі. За кожен кіловат потужності, яка проходить через цей вимірювач, вам приходить рахунок. Рахунок надходить в ваш "адресу". Хоча електрику "тече" вільно по силової мережі, всі роз'єми у вашому будинку мають специфічний адресу.

Огляд сокетів

Та ж сама ідея застосовується і до мережевих роз'ємів - сокетів, за винятком того, що ми говоримо про протоколи TCP / IP і IP-Адреси, а не про електронах і вуличних адресах.

IP (Internet Protocol, Протокол Інтернету) - це протокол маршрутизації нижнього рівня, який розділяє дані на невеликі пакети і посилає їх за різними адресами через мережу, не гарантує доставку відправлених пакетів пункту призначення.

TCP (Transmission Control Protocol, Протокол управління передачею) - це протокол більш високого рівня, який вміє міцно з'єднувати разом пакети, сортуючи і ретранслюючи їх у міру необхідності для надійної передачі даних.

Третій протокол - *UDP* (User Datagram Protocol, Дейтаграмний протокол користувача) - слід за TCP і може застосовуватися безпосередньо для підтримки швидкої, без встановлення з'єднання, але, правда, ненадійною транспортування пакетів.

Клієнт-сервер

Сервер - Це все, що має деякий розділяється (колективно використовується) ресурс.

Існують *обчислювальні сервери,* які забезпечують обчислювальну потужність; *сервери друку,* які керують сукупністю принтерів; *дискові сервери,* які надають працює в мережі дисковий простір, і *Web-Сервери,* які зберігають Web-Сторінки.

Клієнт - просто будь-який інший об'єкт, який хоче отримати доступ до специфічного серверу.

Сервер - це постійно доступний ресурс, у той час як клієнт може "відключитися" після того, як він був обслужений.

Клієнт-сервер

В Berkeley-Сокетах поняття сокета дозволяє окремим комп'ютера обслуговувати багато різних клієнтів відразу, так само, як і обслуговувати безліч різних типів інформації.

Для управління таким обслуговуванням вводиться поняття *порту,* який є пронумерованим сокетом на конкретній машині.

Кажуть, що процес сервера "слухає" порт, поки клієнт не з'єднається з ним.

Серверу дозволяють прийняти багато клієнтів, приєднаних до одного й того ж номеру порту, хоча кожен сеанс унікальний.

Щоб керувати множинними підключеннями клієнта, процес сервера повинен бути багатопотоковий або мати деякі інші засоби мультиплексування одночасного введення / виведення.

Зарезервовані сокети

Після з'єднання протокол високого рівня гарантує жорстку залежність з'єднання від того порту, який ви використовуєте. TCP / IP резервує 1024 нижніх порту для певних протоколів.

Порт з номером

- 21 - для протоколу FTP,
- 23 - для протоколу Telnet,
- 25 - для протоколу електронної пошти (e-mail),
- 79 - для протоколу Finger,
- 80 - для протоколу HTTP,
- 119 - для протоколу телеконференцій і т. д.

Наприклад, HTTP (HyperText Transfer Protocol, Протокол передачі гіпертексту) - це протокол, який Web-Браузери і сервери використовують для пересилання сторінок гіпертексту і зображень.

Коли клієнт запитує файл з HTTP-Сервера (дія, відоме як *"Стук"* (Hit)), він просто друкує ім'я файлу в спеціальному форматі до визначених порту і зчитує вміст файлу.

Сервер відповідає також кодовою числом стану, щоб повідомити клієнта, чи може запит бути виконаний і чому.

Приклад взаємозвязаних процедур клієнта

Сервер	Клиент
Слушает порт 80	Соединяется с портом 80
Принимает соединение	Записывает "GET /index.html HTTP/l.0\n\n"
Читает вплоть до второго символа конца строки (\n)	
Видит, что GET — известная команда и что HTTP/1.0 является правильной версией протокола	
Читает локальный файл с именем /index.html	
Записывает "HTTP/1.0 200 OK\n\n"	"200" означает "Прибывает файл"
Копирует содержание файла в сокет	Читает и отображает содержание файла
Отбой	Отбой

Proxy-сервери

Proxy-Сервер - Це спеціальна службова мережева програма, зазвичай працює на комп'ютері провайдера, призначена для кількох цілей: прискорення роботи клієнта (наприклад, зчитування HTML-Сторінок), зашиті клієнта від вірусів, обмеження доступу клієнта до певних протоколів, серверів і т. п.

Proxy-Сервер пояснює клієнтську сторону протоколу іншого сервера.

Це потрібно, коли клієнти мають деякі обмеження на те, з якими серверами вони можуть з'єднуватися.

Таким чином, клієнт з'єднується з proxy-Сервером, який не має таких обмежень, а той, у свою чергу, здійснює зв'язок для клієнта.

Proxy сервер має додаткові можливості фільтрувати деякі запити або кешувати результати цих запитів для майбутнього використання. Кешуючий proxy HTTP-Сервер може допомогти зменшити вимоги до смуги пропускання на з'єднанні локальної мережі з Internet.

Коли до популярного Web-Сайту звертаються сотні користувачів, proxy-Сервер може один раз отримати вміст популярних сторінок Web-Сервера, заощаджуючи дорогі міжмережеві передачі і забезпечуючи клієнтам більш швидкий доступ до цих сторінок.

Адресація Internet

Будь-який комп'ютер в Internet має *адресу.* Адреса Internet - Це число, яке унікально ідентифікує кожен комп'ютер в мережі.

BIP-Адресах 32 біта, і ми використовуємо їх як послідовність з чотирьох чисел між 0 і 255, розділених крапками.

Перші кілька біт адреси представляють клас мережі (класи позначаються буквами A, B, C, D або E).

Остання - ідентифікує індивідуальний комп'ютер. (В одиночній мережі класу C допустимо до 256 комп'ютерів.)

Ця схема дозволяє існувати на мережах класу C приблизно півмільярда пристроїв.

Служба доменних імен (DNS)

Мережа Internet не була б дуже зручним місцем для навігації, якщо б кожен повинен був використовувати *числову* адресацію.

Важко, наприклад, уявити рекламний адресу у вигляді "http:/ / 192.9.9.1 / ".

Для паралельної ієрархії імен існує інший - *символьний* спосіб адресації.

У мережній адресації широко використовується служба *доменних імен* (DNS,Domain Naming Service).

Подібно до того, як чотири числа IP-адреси описують мережеву ієрархію зліва направо, символьний Internet-Адресу, званий *доменним адресою,* описує розміщення машини в просторі імен справа наліво.

Наприклад, адреса www.starwave.com знаходиться в домені com (Зарезервованому для американських комерційних сайтів), сайт називається starwave (По імені компанії), а www - Ім'я специфічного комп'ютера, який є Web-Сервером starwave.

www відповідає самому правому номеру в еквівалентному IP-Адресі.

Мережеві класи та інтерфейси

Java підтримує протоколи **TCP / IP** як шляхом розширення вже встановленого інтерфейсу поточного введення / виводу, так і додаючи властивості, необхідні для побудови мережевих об'єктів вводу / виводу. Java підтримує два сімейства протоколів - **TCP** і**UDP**.

TCP-протоколи використовуються для надійного *потокового* введення / виводу через мережу.

UDP-Протоколи підтримують більш просту і, отже, більш швидку двоточкову модель, орієнтовану на дейтаграми.

Клас InetAddress

Клас InetAddress використовується, щоб інкапсулювати як числовий IP-Адресу, так і доменну адресу.

Ви взаємодієте з цим класом, використовуючи доменну адресу хост-комп'ютера, який більш зручний і зрозумілий, ніж його числовий еквівалент.

Цей числовий адреса схована всередині класу InetAddress.

Виробничі методи

Клас InetAddress не має видимих конструкторів.

Для створення InetAddress об'єкта потрібно використовувати один з доступних *виробничих* методів.

Виробничі методи (factory methods) - Просто угода, за допомогою якого статичні методи в класі повертають примірник даного класу.

Це зроблено замість перевантаження конструктора різними списками параметрів, коли наявність унікальних імен методів призводить до більш ясним результатами.

Для створення екземплярів типу InetAddress можна використовувати три методи

Виробничі методи

- **static InetAddress getLocalHost () throws UnknownHostException**
- **static InetAddress getByName (String *hostName)* throws UnknownHostException**
- **static InetAddress [] getAllByName (String hostName) throws UnknownHostException**

Метод getLocalHost про просто повертає об'єкт InetAddress, Який представляє локальний хост-комп'ютер.

Метод getByName () Повертає об'єкт типу (класу) InetAddress для комп'ютера, ім'я якого передається йому в параметрі hostName.

Якщо ці методи не здатні розпізнати ім'я комп'ютера, вони викидають виключення типу UnknownHostException.

Сокети TCP / IP клієнтів

TCP / IP сокети використовуються для того, щоб здійснити надійні, двонаправлені, постійні, двоточкові і потокові з'єднання між хост-комп'ютерами в Internet.

Такі сокети можна використовувати для підключення системи введення / виводу Java до інших програм, які можуть постійно перебувати на локальній машині або на будь-який інший машині в Internet.

Аплети можуть встановлювати сокет-з'єднання тільки з хост-машиною, з якою аплет був завантажений.

В Java є два види TCP-сокетів: один - для серверів, а інший - для клієнтів.

Сокети TCP / IP клієнтів

Клас serverSocket розроблений так, щоб бути "слухачем", який перед виконанням будь-якої операції очікує з'єднання з клієнтами.

Клас Socket розроблений так, щоб з'єднуватися з серверними сокетами і ініціалізувати протокольні обміни.

Створення socket-Об'єкта неявно встановлює з'єднання між клієнтом і сервером.

Немає жодних методів або конструкторів, які явно демонструють подробиці установки цього з'єднання.

Конструктори для створення soket-Об'єктів

Синтаксис конструктора	Опис
Socket (string hostWame, int port)	Створює сокет, що з'єднує локальну хост-машину з іменованої хост-машиною і портом; може викидати виняток UnknowhHostException або IOException
Socket (InetAddress ipAddress, , int port)	Створює сокет, аналогічний предидущемуно використовується вже існуючий об'єкт класу InetAddress і порт, може викидати виняток lOException

Використання URL

Web-Протоколи - Це вільна (не пов'язана) сукупність протоколів високого рівня і файлових форматів, об'єднана в Web-Браузері (програмі навігації по мережі).

Один з найбільш важливих аспектів цієї сукупності полягає в тому, що Тім Бернерс-Лі (Tim Bemers-Lee) Винайшов масштабований спосіб розміщення всіх ресурсів мережі.

Можливість надійно іменувати все що завгодно і де завгодно, стає дуже потужною парадигмою (загальним принципом). Даний принцип втілений через універсальний локатор ресурсів (URL,Uniform Resource Locator).

URL забезпечує розумну, зрозумілу форму унікальної ідентифікації адресної інформації в Internet-Мережах.

Усередині мережевої бібліотеки класів java.net є клас url, Який забезпечує простий і короткий API-інтерфейс для доступу до Internet-Інформації з використанням URL-Адрес.

Клас *URLConnection*

URLConnection - Клас загального призначення для доступу до атрибутів віддаленого ресурсу.

Виконавши з'єднання з віддаленим сервером, можна використовувати URLConnection для перегляду властивості віддаленого об'єкта перед фактичної його транспортуванням в локальну програму.

Ці атрибути визначаються специфікацією HTTP-Протоколу і, як такі, мають сенс тільки для URL-Об'єктів, які використовують даний протокол.

```
/ / Демонструє URLConnection.
import java.net. *; import java.io. *; import
java.util.Date;
class UCDemo {public static void main (String
args []) throws Exception {.
int з ;
URL hp = new URL
("http://www.starwave.com/people/naughton/");
URLConnection hpCon = hp.openConnection ();
System.out.println ("Дата: "+ New Date
(hpCon.getDate ()));
System.out.println ("Тип вмісту: "+
HpCon.getContentType ());
System.out.println ("Термін зберігання: "+
HpCon.getExpirationO);
System.out.println ("Останнє зміна: "+
new Date (hpCon.getLastModifiedO));
int len = hpCon.getContentLength ();
System.out.println ("Content-Length:" + len); if
(len> 0)
{
System.out.println ("=== Content ===");
  InputStream input = hpCon.getInputStream ();
int i = len;
while (((c = input.read ())! = -1) && (- i> 0))
(System.out.print ((char) c);
}
input.close ();} else (
System.out.println ("Немає вмісту");}
}
}
```

```
Дата:   Fri Jan 29 16:32:41 CST 1999
Тип вмісту:   text/html
Термін зберігання: Про
Остання зміна:  Wed Jan 20 18:37:54 CST 1999
Content-Length: 275
== - Content ===
<body onload=doRedirect()>
<script language="JavaScript">
<! -
function doRedirectO {
location = "http://homepages.go.com/ ~ pjn" }
```

Сокети TCP / IP серверів

Java надає іншій сокет-клас для створення серверних додатків.

Клас ServerSocket використовується для створення серверів, які прослуховують або локальні, або віддалені програми клієнта, щоб з'єднуватися з ними на опублікованих портах.

Клас ServerSocket сильно відрізняється від нормального класу Socket.

Коли ви створюєте об'єкт класу ServerSocket, Він реєструє себе в системі, як має інтерес до сполук клієнта.

Конструктори serversocket вказують номер порту, на який ви хочете приймати з'єднання, і (за бажанням) якої довжини потрібна чергу для зазначеного порту.

Довжина черги повідомляє системі, скільки підключень клієнта вона може залишати затриманими раніше, ніж повинна просто відкинути з'єднання. Значення за замовчуванням - 50.

При несприятливих умовах конструктори можуть викидати виняток типу IOException.

Конструктори класу ServerSocket

Конструктор	Опис
ServerSocket (int port)	Створює сокет сервера на зазначеному порте з довжиною черги за замовчуванням (50)
ServerSocket (int *port,* int maxQueue)	Створює сокет сервера на зазначеному порте з максимальною довжиною черги maxQaeue
ServerSocket (int port, int maxQueue, InetAddress locaZAddress)	Створює сокет сервера на зазначеному порте з максимальною довжиною черги maxGueue. На груповому хост-комп'ютері (комп'ютер, який має декілька мережевих адаптерів або сконфигурирован з декількома IP-адресами для одиночного мережевого адаптера) localAddress визначає IP-адресу, з яким цей сокет пов'язаний

Дейтаграми

Для більшості міжмережевих потреб вас буде задовольняти TCP/IP-Стиль роботи в мережі.

Він забезпечує серіалізовать, передбачуваний і надійний потік пакетних даних.

Протокол TCP включає багато складних алгоритмів, що мають справу з управлінням перевантаженням на переповнених мережах і вельми песимістичними очікуваннями втрат пакетів. Він призводить до досить неефективного способу транспортування даних.

Альтернативу забезпечують дейтаграмним протоколи.

Дейтаграми

Дейтаграми - Це невеликі пакети інформації, незалежно транспортуються між машинами.

Після того як дейтаграмма направляється призначеному їй адресату, немає ніякої гарантії, що вона прибуде в пункт призначення, і хтось зможе там її отримати.

Аналогічно, коли дейтаграма приймається, немає гарантії, що вона не була пошкоджена при пересиланні або що той, хто її послав, все ще перебуває у пункті передачі і готовий отримати відповідь.

Java реалізує дейтаграми у верхній частині UDP-Протоколу, використовуючи два класи.

Клас *DatagramPacket*

Об'єкт DatagramPacket може бути створено одним з чотирьох конструкторів. Перший конструктор визначає буфер, який прийме дані, і розмір пакета. Він застосовується для отримання даних через DatagramSocket. Друга форма дозволяє визначати зміщення до буферу, в якому дані будуть зберігатися. Третя форма специфікує цільовий адресу і порт, що використовуються класом DatagramSocket, Щоб визначити, куди дані пакета будуть відправлені. Четверта форма передає пакети, що починаються з вказаним зміщенням в наборі даних. Існує чотири конструктора:

- **DatagramPacket (byte *data*[ ], Int *size*)**
- **DatagramPacket (byte *data*[], Int offset, int size)**
- **DatagramPacket (byte *data*[], Int *size,* InetAddress *ipAddress,* int *port*)**
- **DatagramPacket (byte *data*[ ], Int *offset,* int *size,* InetAddress *ipAddress*,int *port*)**

Дейтаграмний сервер і клієнт

```
/ / Демонструє дейтаграми.
import java.net. *
class WriteServer {
public static int serverPort '= 666;
public static int cllentPort. = 999;
public static int buffer_size = 1024;
public static DatagramSocket ds;
public static byte buffer!] = new byte
[buffer size];
public static void TheServerO throws
Exception {int pos = 0; while (true) {
int з = System.in.read (); switch (c)
{case -1:
System.out.printing "Server Quits.");
Return; case '\ r':
break; case '\ n':
ds.send (new DatagramPacket (buffer,
pos,
InetAddress.getLocalHost (),
clientPort)); pos = 0; break; default:
buffer [pos + +] = (byte) c;
 }
 }
 }
 }
```

```
public static void TheClientO throws
Exception (while (true) {
DatagramPacket p = new
DatagramPacket (buffer, buffer.length);
ds.receive (p);
System.out.println (new String
(p.getData (), 0, p.getLengthO));}}
public static void main (String args [])
throws Exception (if (args.length == 1)
{
ds = new DatagramSocket
(serverPort); TheServer ();} else {
ds = new DatagramSocket (clientPort);
TheClientO;
}
 )
 }
```

Дякую за увагу!