NGÔN NGỮ LẬP TRÌNH JAVA

Nội dung

Cơ bản về ngôn ngữ lập trình Java

- Lập trình hướng đối tượng
- Biến, từ khoá, kiểu dữ liệu
- Biểu thức, các cấu trúc điều khiển
- Dữ liệu kiểu mảng

Các khía cạnh nâng cao của lập trình hướng đối tượng

- Thiết kế lớp
- Thiết kế lớp nâng cao
- Xử lý ngoại lệ
- Generics
- Java Collection Framework
- Multithread& Concurrency
- Java Database Programming
- **Java Network Programming**

JAVA NETWORK PROGRAMMING

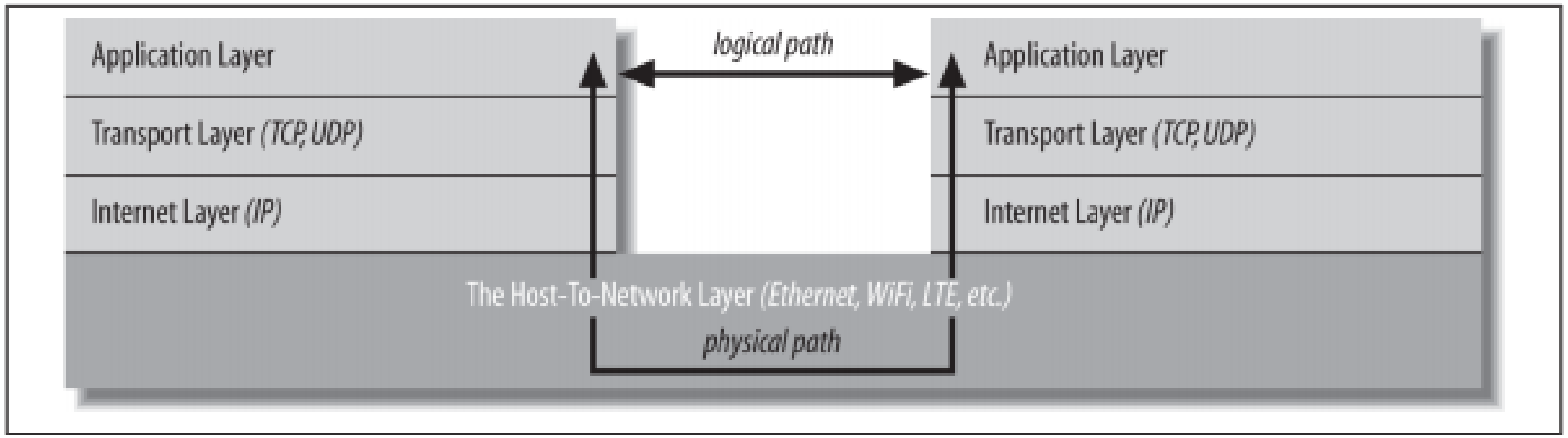
Nội dung

1. Network programming
   1. Network protocols
   2. Socket programming
2. ServerSocket class
3. Socket class
4. InetAddress class
5. Example
   1. Socket client
   2. Socket server

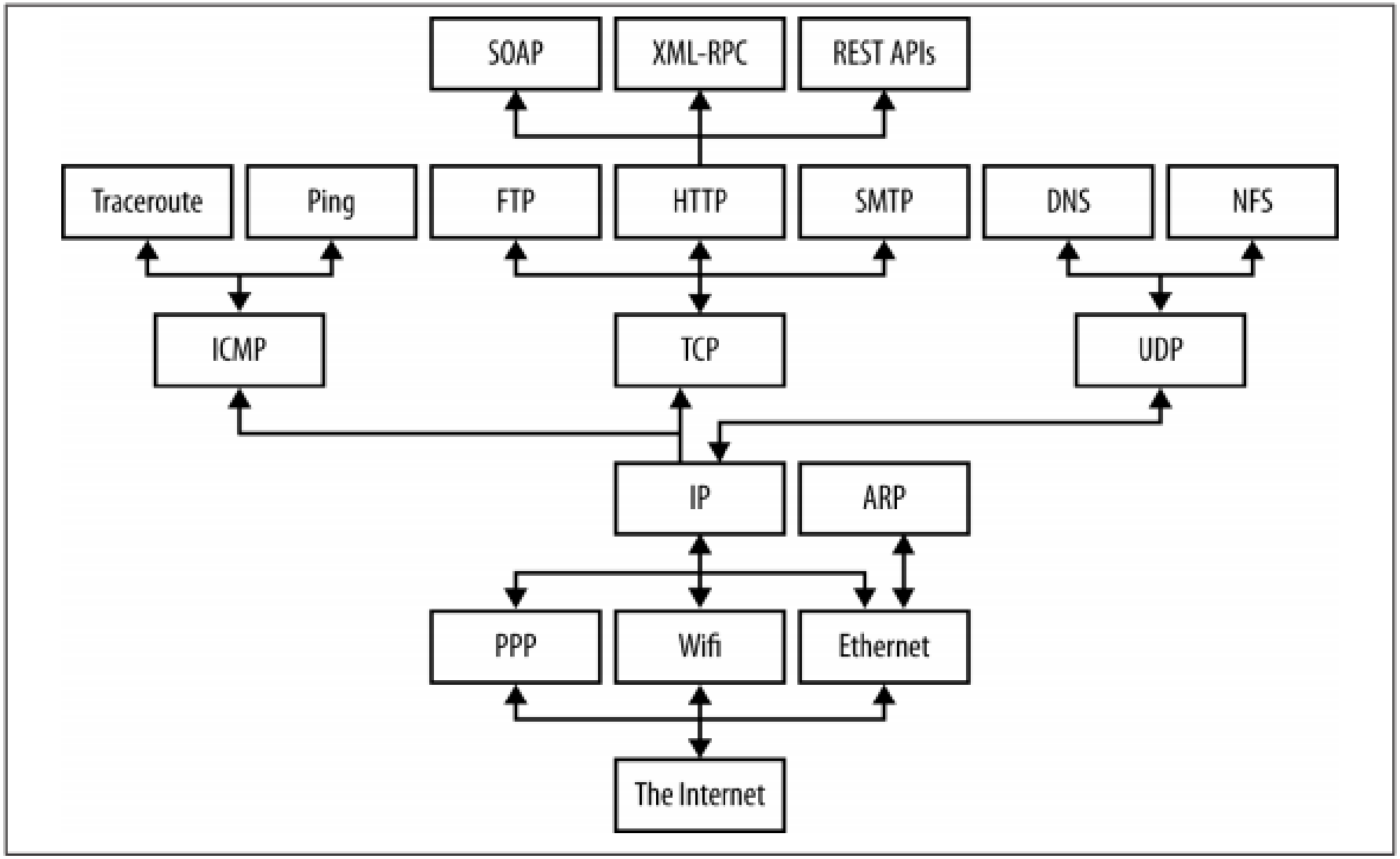
1. Network programming (1/3)

- Lập trình mạng:
  - Viết chương trình thực thi trên nhiều thiết bị
  - Các thiết bị được kết nối sử dụng một hệ thống mạng
- Gói java.net :
  - chứa các lớp và các interfaces cung cấp các hình thức giao tiếp mức thấp
  - Hỗ trợ hai giao thức cơ bản: TCP & UDP
- TCP (Transmission Control Protocol) :
  - hỗ trợ giao tiếp tin cậy giữa 2 ứng dụng
  - Vận hành dựa trên IP (Internet Protocol)
- UDP (User Datagram Protocol)
  - Giao thức không hướng kết nối, cho phép các gói dữ liệu được truyền đi giữa 2 ứng dụng.

TCP/IP four layer models



Protocols in different layers



Well-known ports assignments

Protocol	Port	Protocol	Purpose
FTP	21	TCP	This port is used to send FTP commands like `put` and `get`.
SSH	22	TCP	Used for encrypted, remote logins.
Telnet	23	TCP	Used for interactive, remote command-line sessions.
smtp	25	TCP	The Simple Mail Transfer Protocol is used to send email between machines.
time	37	TCP/UDP	A time server returns the number of seconds that have elapsed on the server since midnight, January 1, 1900, as a four-byte, unsigned, big-endian integer.
whois	43	TCP	A simple directory service for Internet network administrators.
finger	79	TCP	A service that returns information about a user or users on the local system.
HTTP	80	TCP	The underlying protocol of the World Wide Web.
POP3	110	TCP	Post Office Protocol version 3 is a protocol for the transfer of accumulated email from the host to sporadically connected clients.
NNTP	119	TCP	Usenet news transfer; more formally known as the "Network News Transfer Protocol."
IMAP	143	TCP	Internet Message Access Protocol is a protocol for accessing mailboxes stored on a server.
dict	2628	TCP	A UTF-8 encoded dictionary service that provides definitions of words.

1. Network programming (2/3)

- Socket programming:
  - Cung cấp cơ chế giao tiếp giữa 2 máy tính sử dụng TCP.
  - Client tạo một socket và kết nối tới server sử dụng socket đó
  - Khi kết nối đã được khởi tạo, server cũng tạo ra một đối tượng socket,
  - Client và server kết nối thông qua việc đọc và ghi trên đối tượng socket.
- Lớp java.net.ServerSocket cung cấp cơ chế cho phép một server lắng nghe và thiết lập kết tới các clients.
- Lớp java.net.Socket biểu diễn một socket

1. Network programming (3/3)

- Trình tự thiết lập kết nối giữa 2 thiết bị sử dụng socket:
  - 1 Server khởi tạo 1 đt ServerSocket, với số port giao tiếp
  - Server gọi phương thức accept(), cho phép đợi đến khi có 1 client kết nối tới Server
  - Để kết nối tới 1 server, Client khởi tạo 1 đt Socket, với tên server và số port giao tiếp
  - Hàm khởi tạo của lớp Socket cố gắng kết nối tới port xác định của server . Nếu giao tiếp được thiết lập, client có 1 đt Socket với khả năng kết nối tới Server
  - Ở phía server, phương thức accept() trả về tham chiếu đến một đt socket mới trên server đang được kết nối tới socket của client.
- Sau khi kết nối được thiết lập, giao tiếp có thể diễn ra qua I/O stream.
  - Mỗi socket có cả OutputStream và InputStream
  - OutputStream của client kết nối tới InputStream của Server
  - InputStream của server kết nối tới OutputStream của Client

2. ServerSocket Class

- ServerSocket class được sử dụng bởi ứng dụng trên server để đăng ký 1 port và lắng nghe/đợi yêu cầu của client trên port đó
- 4 hàm khởi tạo

SN	Methods with Description
1	**public ServerSocket(int port) throws IOException** Attempts to create a server socket bound to the specified port. An exception occurs if the port is already bound by another application.
2	**public ServerSocket(int port, int backlog) throws IOException** Similar to the previous constructor, the backlog parameter specifies how many incoming clients to store in a wait queue.
3	**public ServerSocket(int port, int backlog, InetAddress address) throws IOException** Similar to the previous constructor, the InetAddress parameter specifies the local IP address to bind to. The InetAddress is used for servers that may have multiple IP addresses, allowing the server to specify which of its IP addresses to accept client requests on
4	**public ServerSocket() throws IOException** Creates an unbound server socket. When using this constructor, use the bind() method when you are ready to bind the server socket

2. ServerSocket Class

- Một số phương thức hay sử dụng

SN	Methods with Description
1	**public int getLocalPort()** Returns the port that the server socket is listening on. This method is useful if you passed in 0 as the port number in a constructor and let the server find a port for you.
2	**public Socket accept() throws IOException** Waits for an incoming client. This method blocks until either a client connects to the server on the specified port or the socket times out, assuming that the time-out value has been set using the setSoTimeout() method. Otherwise, this method blocks indefinitely
3	**public void setSoTimeout(int timeout)** Sets the time-out value for how long the server socket waits for a client during the accept().
4	**public void bind(SocketAddress host, int backlog)** Binds the socket to the specified server and port in the SocketAddress object. Use this method if you instantiated the ServerSocket using the no-argument constructor.

3. Socket Class

- Java.net.Socket class: biểu diễn socket mà cả client và server sử dụng dùng để kết nối.
- Client khởi tạo 1 đt socket
- Server lấy 1 đt socket từ giá trị trả về của phương thức accept()

3. Socket Class

- 5 hàm khởi tạo

SN	Methods with Description
1	**public Socket(String host, int port) throws UnknownHostException, IOException.** This method attempts to connect to the specified server at the specified port. If this constructor does not throw an exception, the connection is successful and the client is connected to the server.
2	**public Socket(InetAddress host, int port) throws IOException** This method is identical to the previous constructor, except that the host is denoted by an InetAddress object.
3	**public Socket(String host, int port, InetAddress localAddress, int localPort) throws IOException.** Connects to the specified host and port, creating a socket on the local host at the specified address and port.
4	**public Socket(InetAddress host, int port, InetAddress localAddress, int localPort) throws IOException.** This method is identical to the previous constructor, except that the host is denoted by an InetAddress object instead of a String
5	**public Socket()** Creates an unconnected socket. Use the connect() method to connect this socket to a server.

3. Socket Class

- Một số phương thức hay sử dụng

SN	Methods with Description
1	**public void connect(SocketAddress host, int timeout) throws IOException** This method connects the socket to the specified host. This method is needed only when you instantiated the Socket using the no-argument constructor.
2	**public InetAddress getInetAddress()** This method returns the address of the other computer that this socket is connected to.
3	**public int getPort()** Returns the port the socket is bound to on the remote machine.
4	**public int getLocalPort()** Returns the port the socket is bound to on the local machine.
5	**public SocketAddress getRemoteSocketAddress()** Returns the address of the remote socket.
6	**public InputStream getInputStream() throws IOException** Returns the input stream of the socket. The input stream is connected to the output stream of the remote socket.
7	**public OutputStream getOutputStream() throws IOException** Returns the output stream of the socket. The output stream is connected to the input stream of the remote socket
8	**public void close() throws IOException** Closes the socket, which makes this Socket object no longer capable of connecting again to any server

4. InetAddress Class

- InetAddress class biểu diễn địa chỉ IP
- Một số phương thức hay sử dụng

SN	Methods with Description
1	**static InetAddress getByAddress(byte[] addr)** Returns an InetAddress object given the raw IP address .
2	**static InetAddress getByAddress(String host, byte[] addr)** Create an InetAddress based on the provided host name and IP address.
3	**static InetAddress getByName(String host)** Determines the IP address of a host, given the host's name.
4	**String getHostAddress()** Returns the IP address string in textual presentation.
5	**String getHostName()** Gets the host name for this IP address.
6	**static InetAddress InetAddress getLocalHost()** Returns the local host.
7	**String toString()** Converts this IP address to a String.

5. Example - Socket client

```
// File Name GreetingClient.java

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

public class GreetingClient
{
   public static void main(String [] args)
   {
      String serverName = args[0];
      int port = Integer.parseInt(args[1]);
      try
      {
         System.out.println("Connecting to " + serverName
                            + " on port " + port);
         Socket client = new Socket(serverName, port);
         System.out.println("Just connected to "
                     + client.getRemoteSocketAddress());
         OutputStream outToServer = client.getOutputStream();
         DataOutputStream out =
                      new DataOutputStream(outToServer);

         out.writeUTF("Hello from "
                     + client.getLocalSocketAddress());
         InputStream inFromServer = client.getInputStream();
         DataInputStream in =
                       new DataInputStream(inFromServer);
         System.out.println("Server says " + in.readUTF());
         client.close();
      }catch(IOException e)
      {
         e.printStackTrace();
      }
   }
}
```

5. Example - Socket server (1/2)

```
// File Name GreetingServer.java

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

public class GreetingServer extends Thread
{
   private ServerSocket serverSocket;

   public GreetingServer(int port) throws IOException
   {
      serverSocket = new ServerSocket(port);
      serverSocket.setSoTimeout(10000);
   }

   public void run()
   {
      while(true)
      {
         try
         {
            System.out.println("Waiting for client on port " +
            serverSocket.getLocalPort() + "...");
            Socket server = serverSocket.accept();
            System.out.println("Just connected to "
                  + server.getRemoteSocketAddress());
            DataInputStream in =
                  new DataInputStream(server.getInputStream());
            System.out.println(in.readUTF());
```

5. Example - Socket server (2/2)

```
            DataOutputStream out =
                 new DataOutputStream(server.getOutputStream());
            out.writeUTF("Thank you for connecting to "
              + server.getLocalSocketAddress() + "\nGoodbye!");
            server.close();
         }catch(SocketTimeoutException s)
         {
            System.out.println("Socket timed out!");
            break;
         }catch(IOException e)
         {
            e.printStackTrace();
            break;
         }
      }
   }
   public static void main(String [] args)
   {
      int port = Integer.parseInt(args[0]);
      try
      {
         Thread t = new GreetingServer(port);
         t.start();
      }catch(IOException e)
      {
         e.printStackTrace();
      }
   }
}
```

5. Example - Result

Compile client and server and then start server as follows:

```
$ java GreetingServer 6066
Waiting for client on port 6066...
```

Check client program as follows:

```
$ java GreetingClient localhost 6066
Connecting to localhost on port 6066
Just connected to localhost/127.0.0.1:6066
Server says Thank you for connecting to /127.0.0.1:6066
Goodbye!
```