

Програмування-1

Лекція 17
Networking

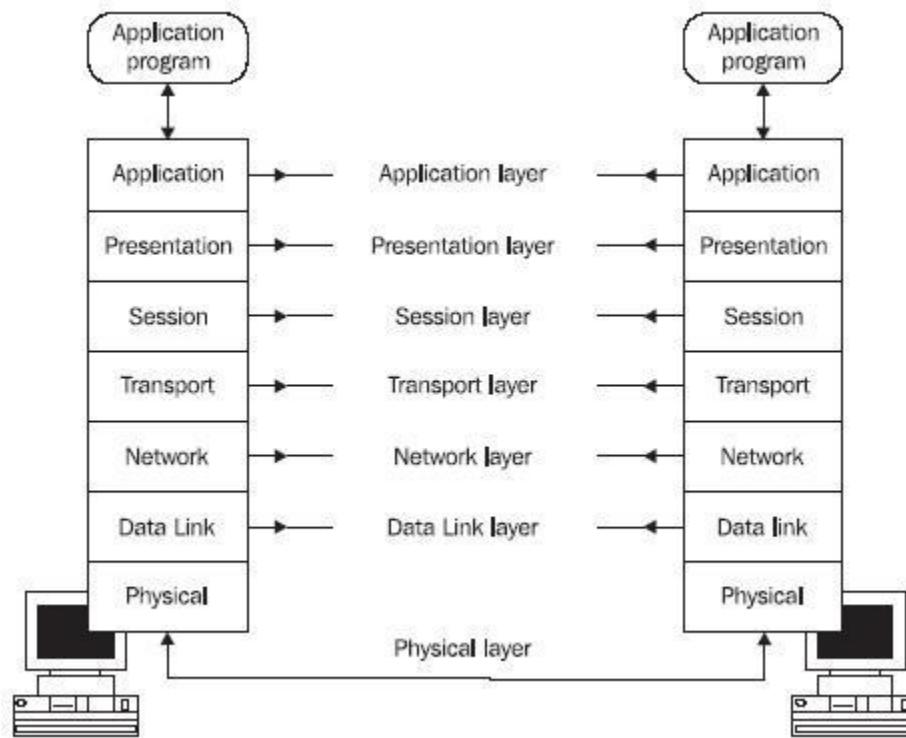
Мережева модель OSI

- OSI model - Open Systems Interconnection model
- Базова еталонна модель взаємодії відкритих систем
- 7-рівнева модель

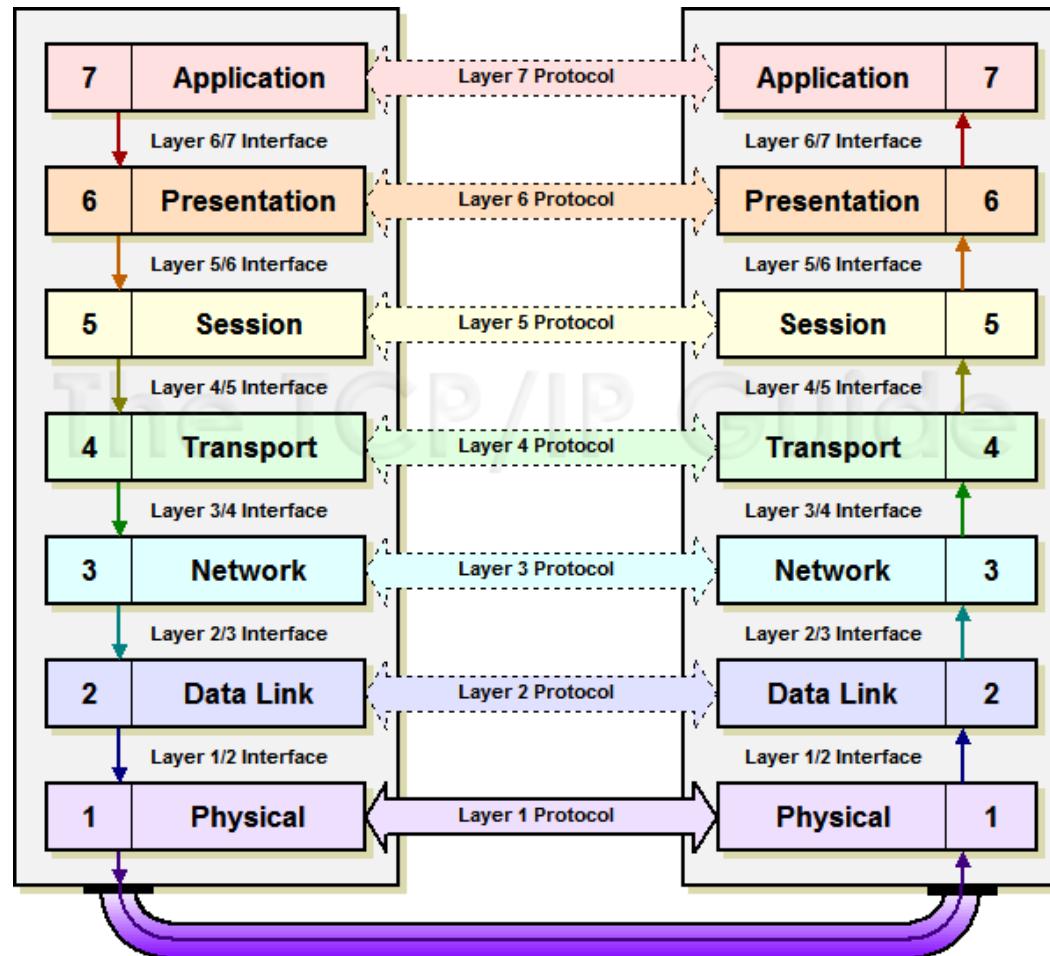
Мережева модель OSI

Layer			Protocol data unit (PDU)	Function ^[3]
Host layers	7	Application	Data	High-level APIs , including resource sharing, remote file access
	6	Presentation		Translation of data between a networking service and an application; including character encoding , data compression and encryption/decryption
	5	Session		Managing communication sessions , i.e. continuous exchange of information in the form of multiple back-and-forth transmissions between two nodes
	4	Transport	Segment , Datagram	Reliable transmission of data segments between points on a network, including segmentation , acknowledgement and multiplexing
Media layers	3	Network	Packet	Structuring and managing a multi-node network, including addressing , routing and traffic control
	2	Data link	Frame	Reliable transmission of data frames between two nodes connected by a physical layer
	1	Physical	Symbol	Transmission and reception of raw bit streams over a physical medium

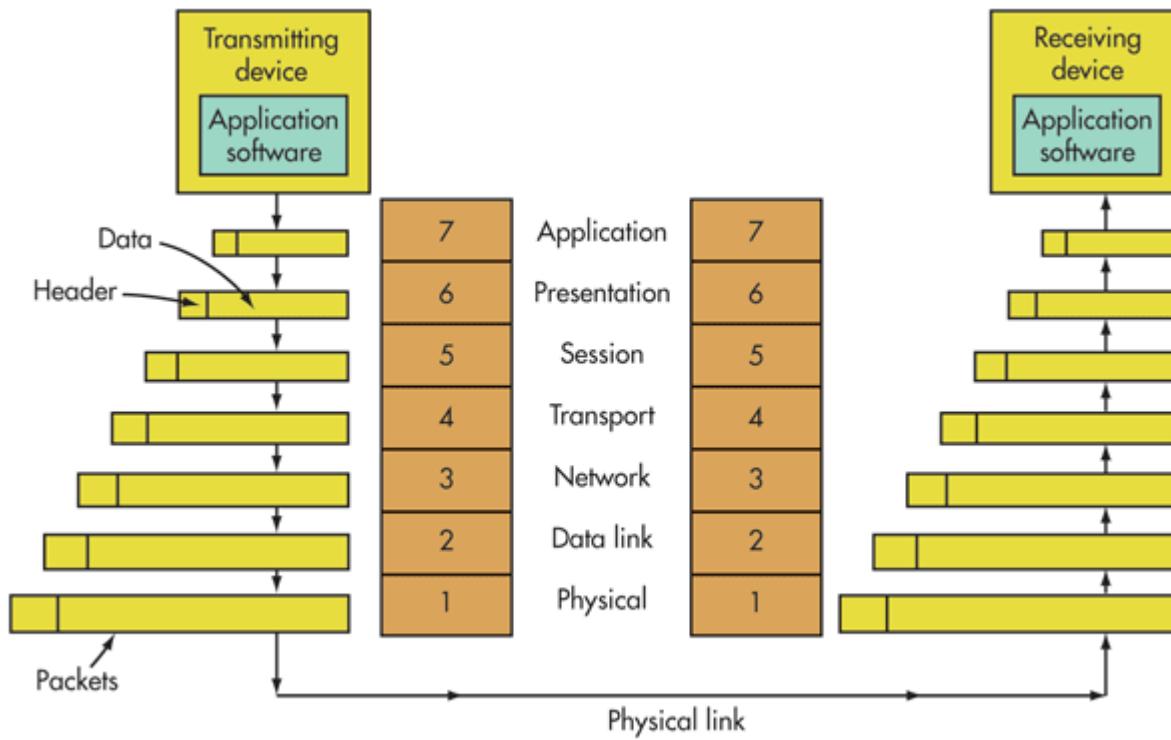
Мережева модель OSI



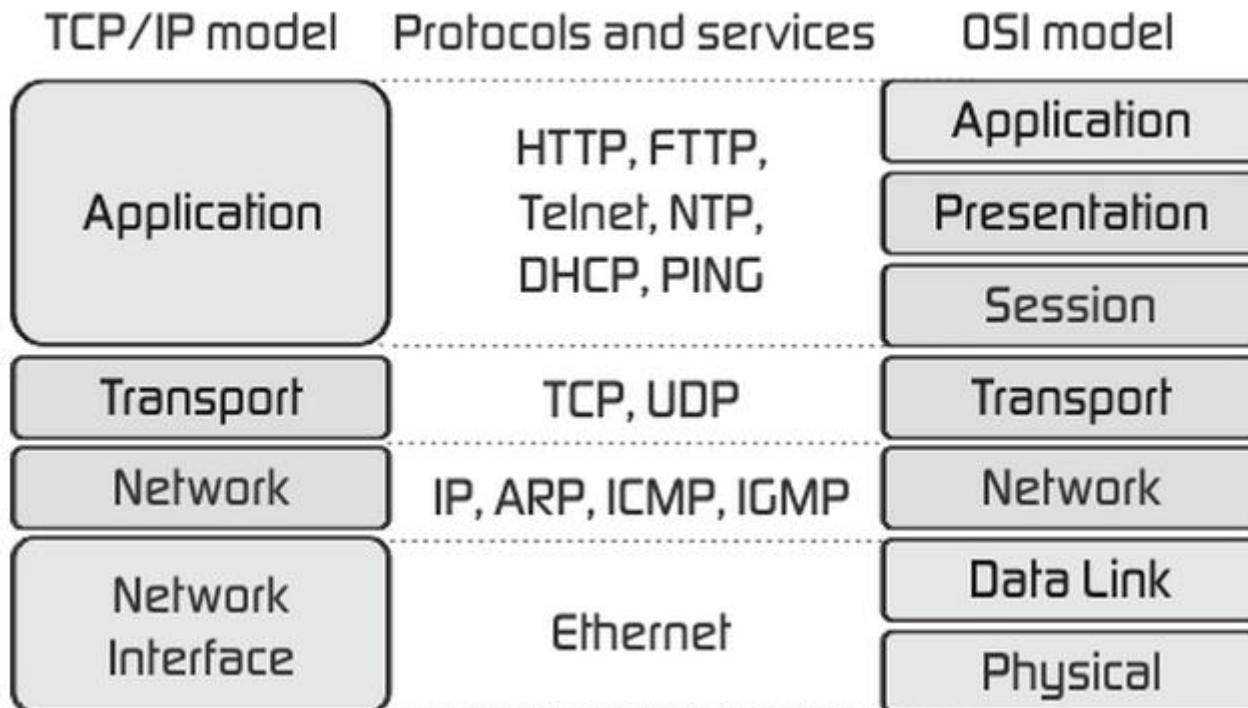
Protocol vs Interface



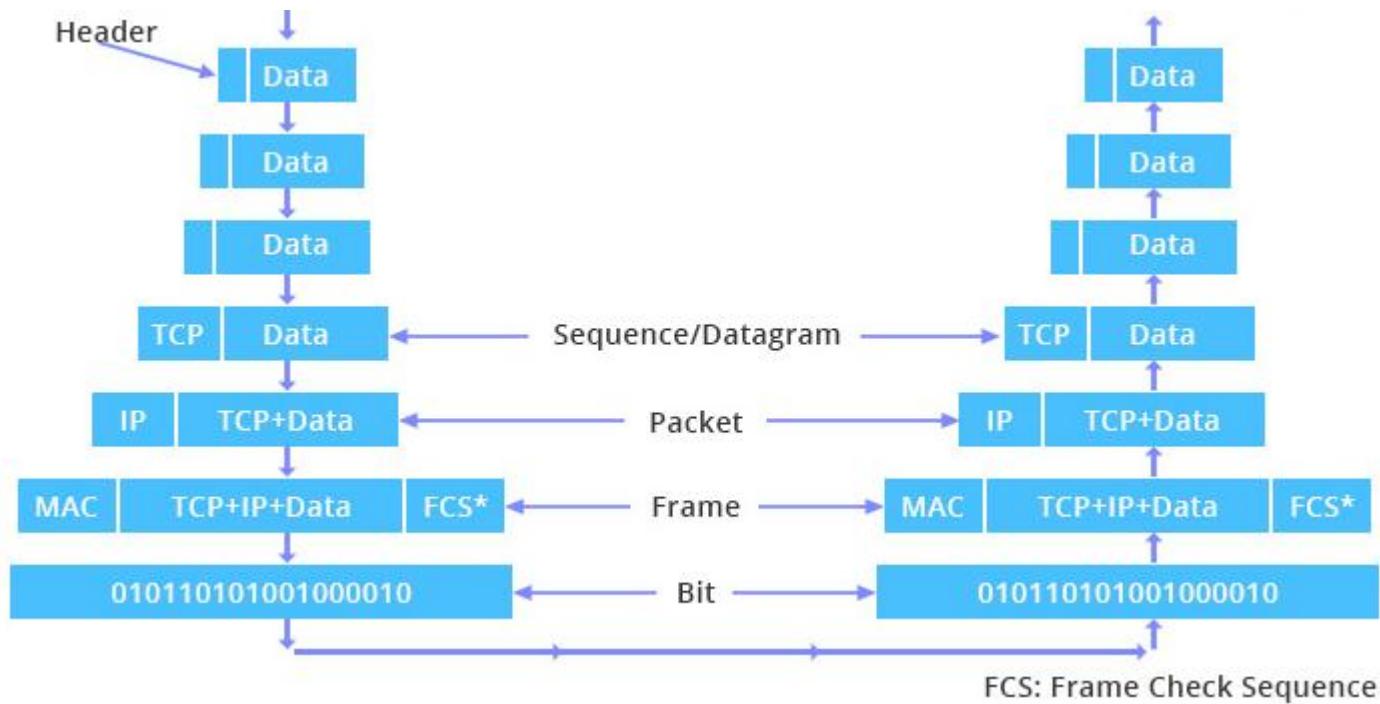
Packet = Header + Data *[+Tail]*



TCP/IP vs OSI



Packet = Header + Data [+Tail]



Що потрібно знати про TCP/IP?

- IP
- DNS
- TCP / UPD
- Port
- NAT/PAT
- MAC
- Socket
- MTU
- ping, netstat, ipconfig

IP

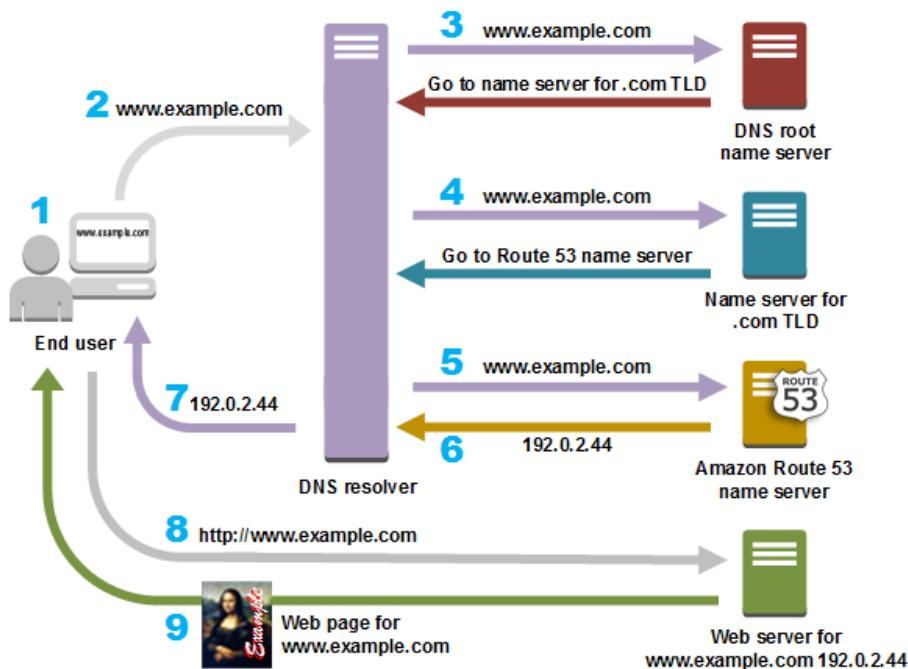
- IP – Internet Protocol
- Версії IP
 - IPv4 – 32-роздрядна адреса
 - IPv6 – 128-роздрядна адреса
- localhost, loopback, 127.0.0.1
 - Насправді: 127.0.0.1 – 127.255.255.254
- MTU – maximum transmission unit
 - Provider specific
 - Minimum (гарантований) MTU:
 - IPv4: 576 байт
 - IPv6: 1280 байт

IP

- IP-адреса умовно складається з **адреси мережі** та **адреси хоста** у мережі
- **Маска мережі**
 - потрібна для того, щоб вказати, де **адреси мережі**, а де **адреса хоста**
- hostIP & netMask == netIP
- у двійковому вигляді (32 розряди):
 - **1111....11110000...000**
- **2 способи запису**
 - XXX.XXX.XXX.XXX
 - /XX
- **Приклад:** **192.168.0.0 – 192.168.0.255**
 - IP: **192.168.0.0** Mask: **255.255.255.0**
 - IP: **192.168.0.0 / 24**

DNS

- DNS - Domain Name System
- “google.com” = “172.217.16.14”



TCP vs UDP

Item	TCP	UDP
Stands For	Transmission Control Protocol	User Datagram Protocol
Protocol	Connection Oriented	Connectionless
Security	Makes Checks For Errors And Reporting	Makes Error Checking But No Reporting
Data Sending	Slower	Faster
Header Size	20 Bytes	8 Bytes
Segments	Acknowledgement	No Acknowledgement
Typical Applications	- Email	- VoIP

TCP vs UDP



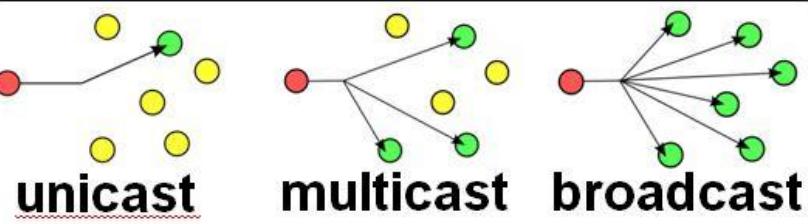
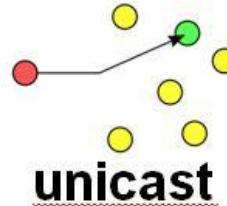
TCP



UDP

- Slower but reliable transfers
- Typical applications:
 - Email
 - Web browsing

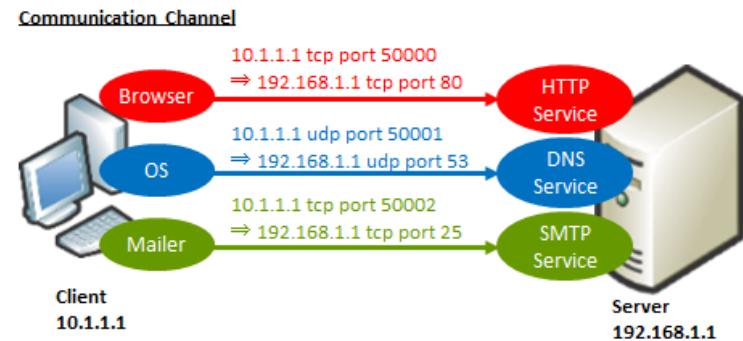
- Fast but non-guaranteed transfers (“best effort”)
- Typical applications:
 - VoIP
 - Music streaming



Порт

- На одному фізичному хості може бути кілька сервісів

- Потрібна унікальна адреса
- Порт – адреса сервісу на хості



- Порт: 0..65535
 - 0..1023 – system
 - 1024..49151 – registered
 - 49152.. 65535 – dynamic

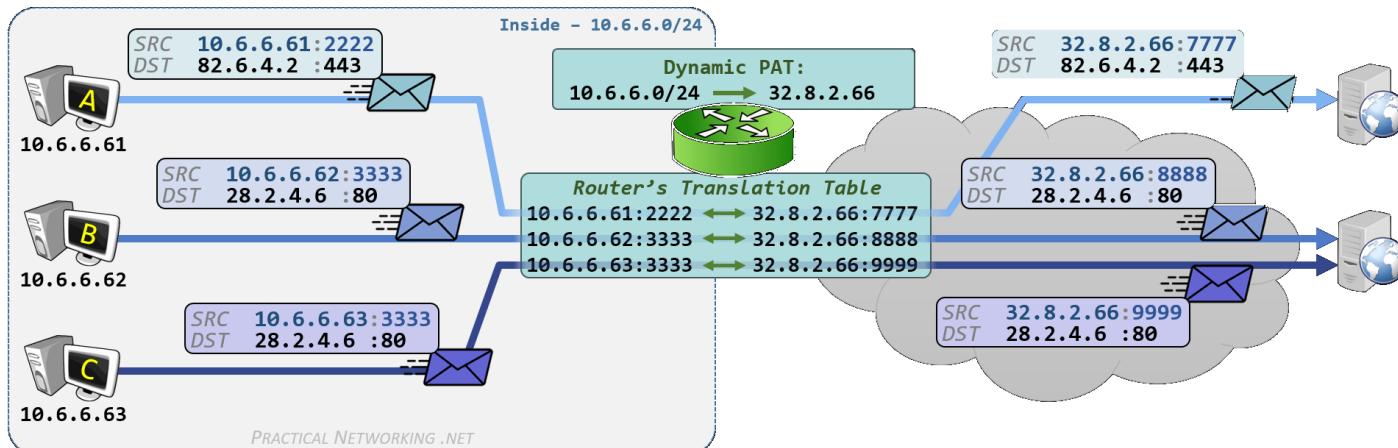
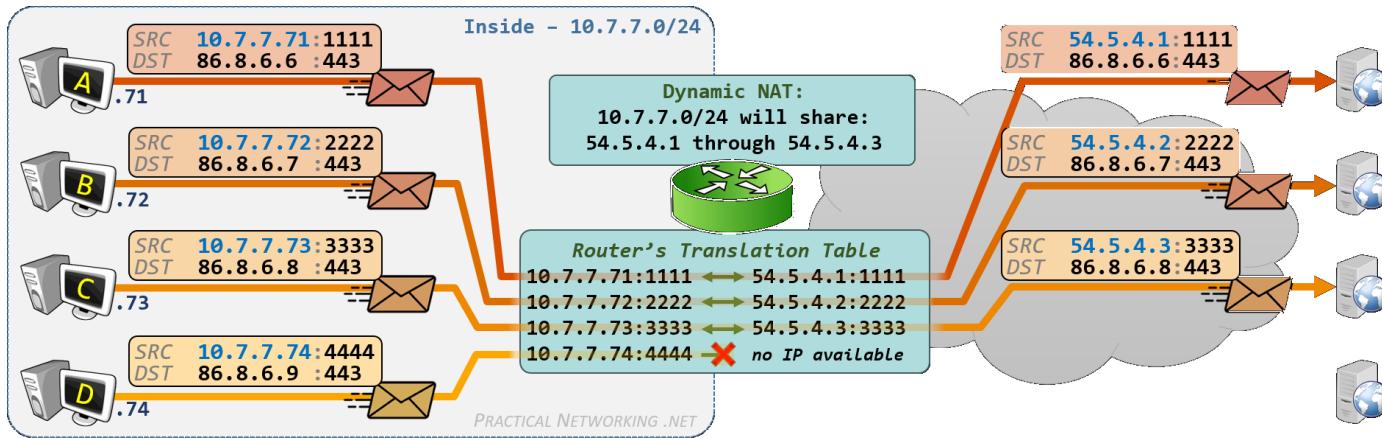
Local address, NAT/PAT

- Local IPv4 addresses:

24-bit block	10.0.0.0 – 10.255.255.255	single class A network
20-bit block	172.16.0.0 – 172.31.255.255	16 contiguous class B networks
16-bit block	192.168.0.0 – 192.168.255.255	256 contiguous class C networks

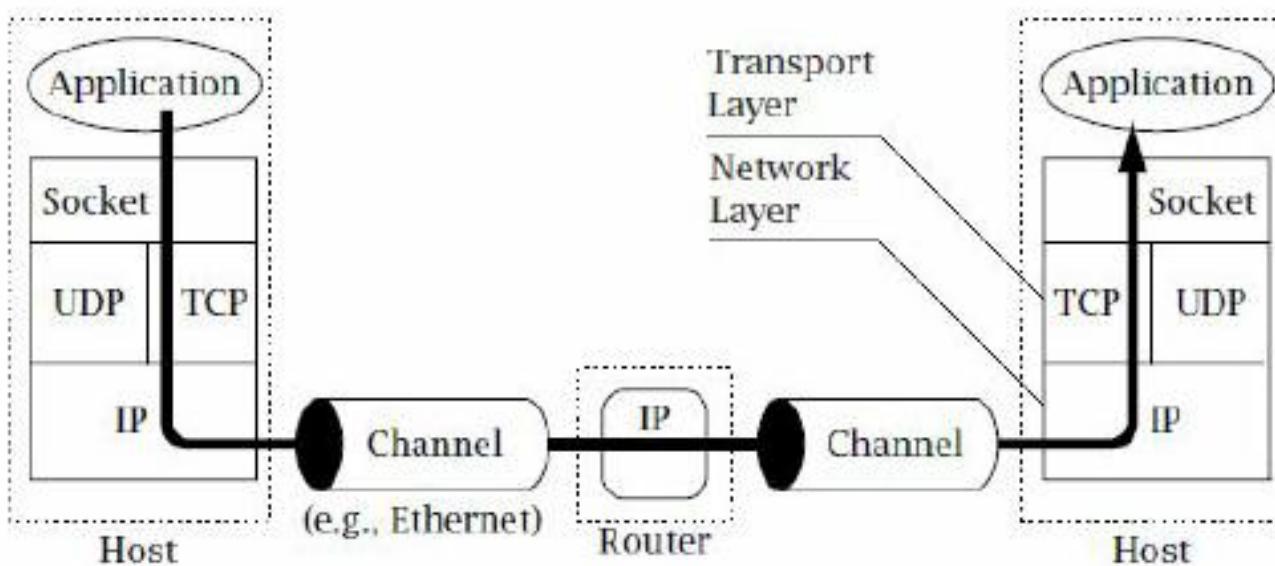
- NAT - Network Address Translation
- PAT - Port Address Translation

NAT / PAT

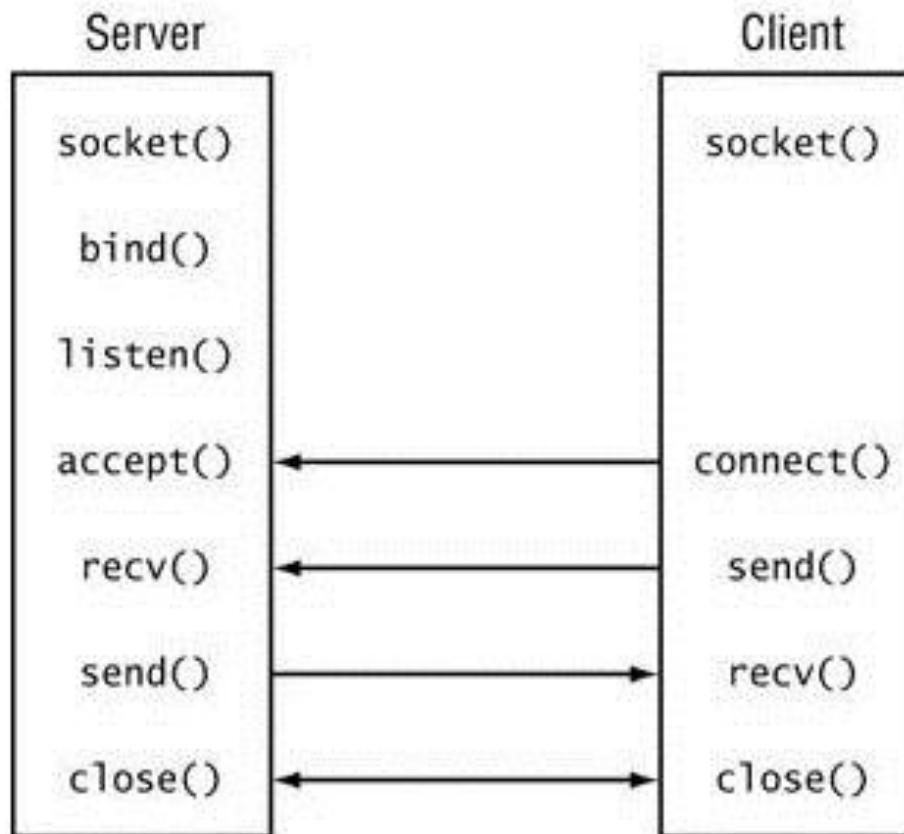


Socket

- Socket – інтерфейс між додатком та TCP/IP



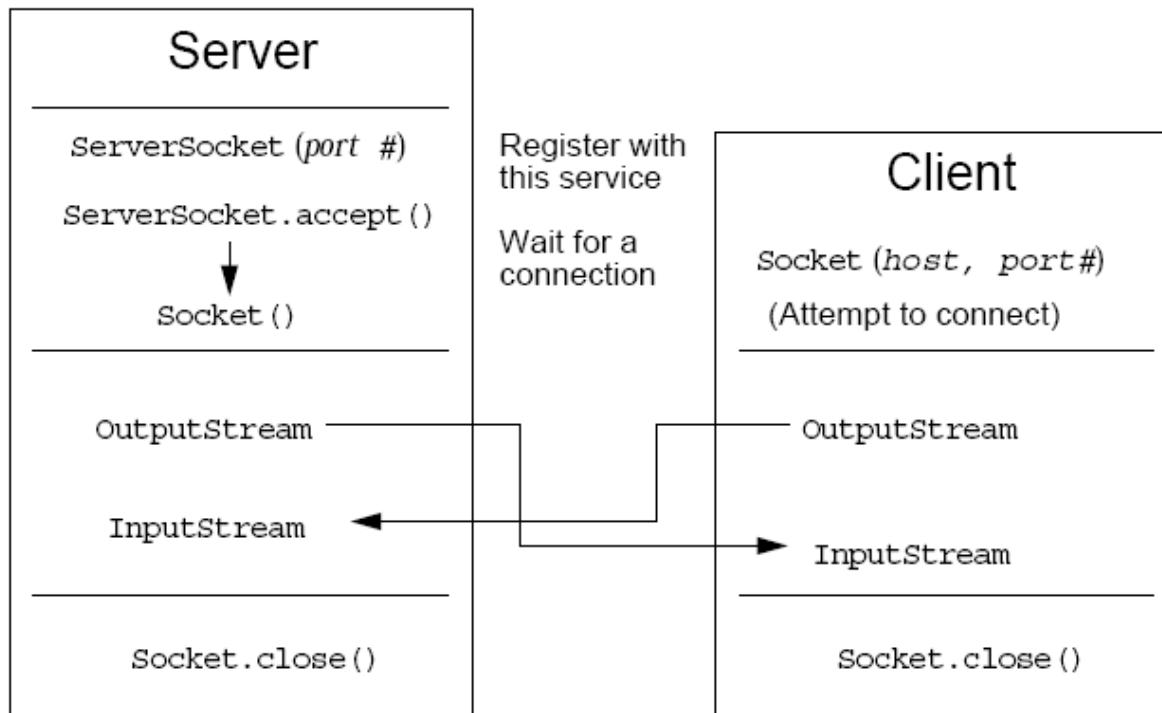
Socket



java.net

- `Socket, ServerSocket` - TCP
- `DatagramSocket, DatagramPacket` - UDP
- `InetAddress, Inet4Address, Inet6Address`
- `InetSocketAddress`

Java TCP client/server



Java TCP Server

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

public class SimpleServer {

    public static void main(String args[]) {
        ServerSocket serverSocket= null;

        try {
            serverSocket = new ServerSocket(5432);
        } catch (IOException e) {
            e.printStackTrace();
            return;
        }
    }
}
```

Java TCP Server

```
while (true) {  
    try {  
        Socket clientSocket = serverSocket.accept();  
  
        OutputStream os = clientSocket.getOutputStream();  
        BufferedWriter bw = new BufferedWriter(  
            new OutputStreamWriter(os));  
  
        bw.write("Hello Net World!\n");  
  
        bw.close();  
        clientSocket.close();  
    } catch (IOException e) {  
        e.printStackTrace();  
    }  
}  
}
```

Java TCP Client

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

public class SimpleClient {
    public static void main(String args[]) {
        try {
            Socket socket = new Socket("127.0.0.1", 5432);
            InputStream is = socket.getInputStream();
            BufferedReader br = new BufferedReader(
                new InputStreamReader(is));
            System.out.println(br.readLine());
            br.close();
            socket.close();
        } catch (ConnectException connExc) {
            System.err.println("Could not connect.");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```