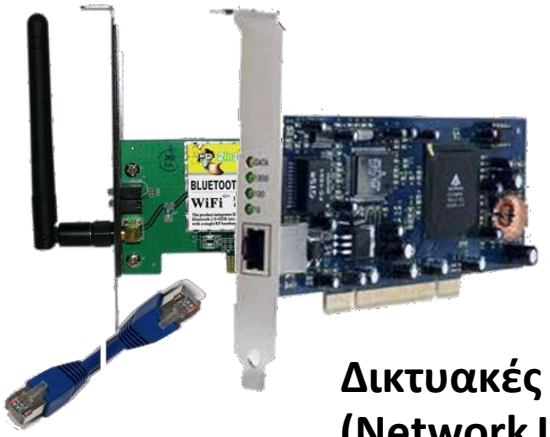
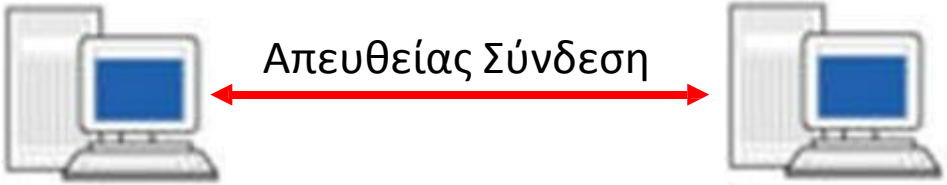
Δικτυακός Προγραμματισμός

Δικτυακή Ορολογία

ΗΥ

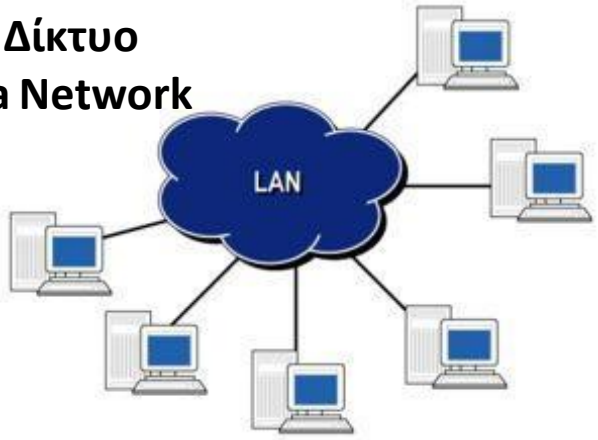


Δικτυακές Διεπαφές (Network Interfaces)

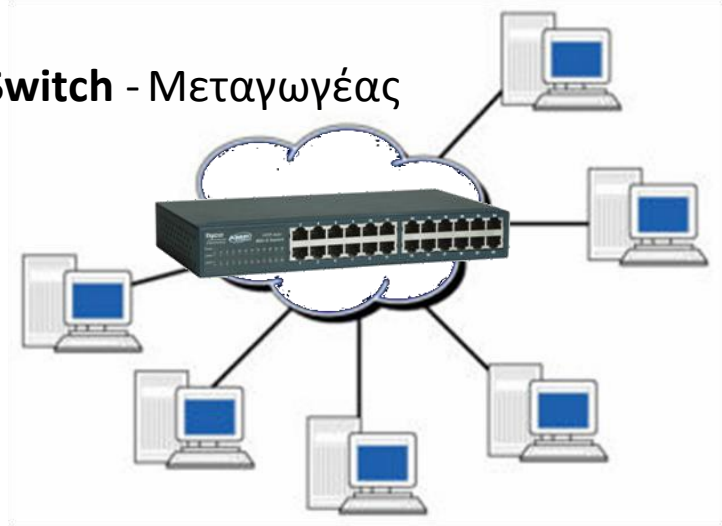


Απευθείας Σύνδεση

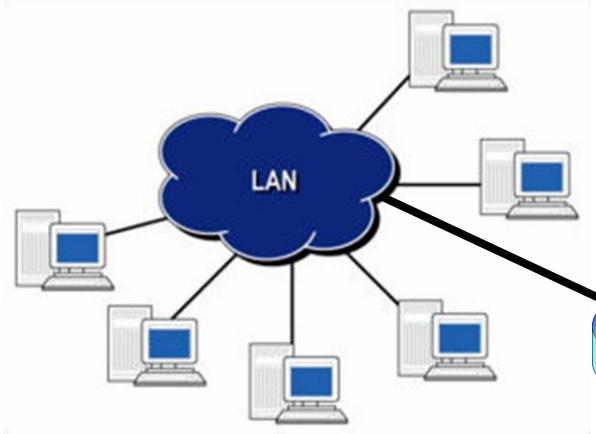
Τοπικό Δίκτυο
Local Area Network



Switch - Μεταγωγέας

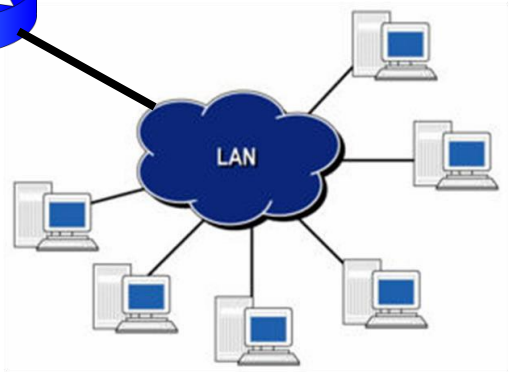


Διαμεταγωγή - Routing

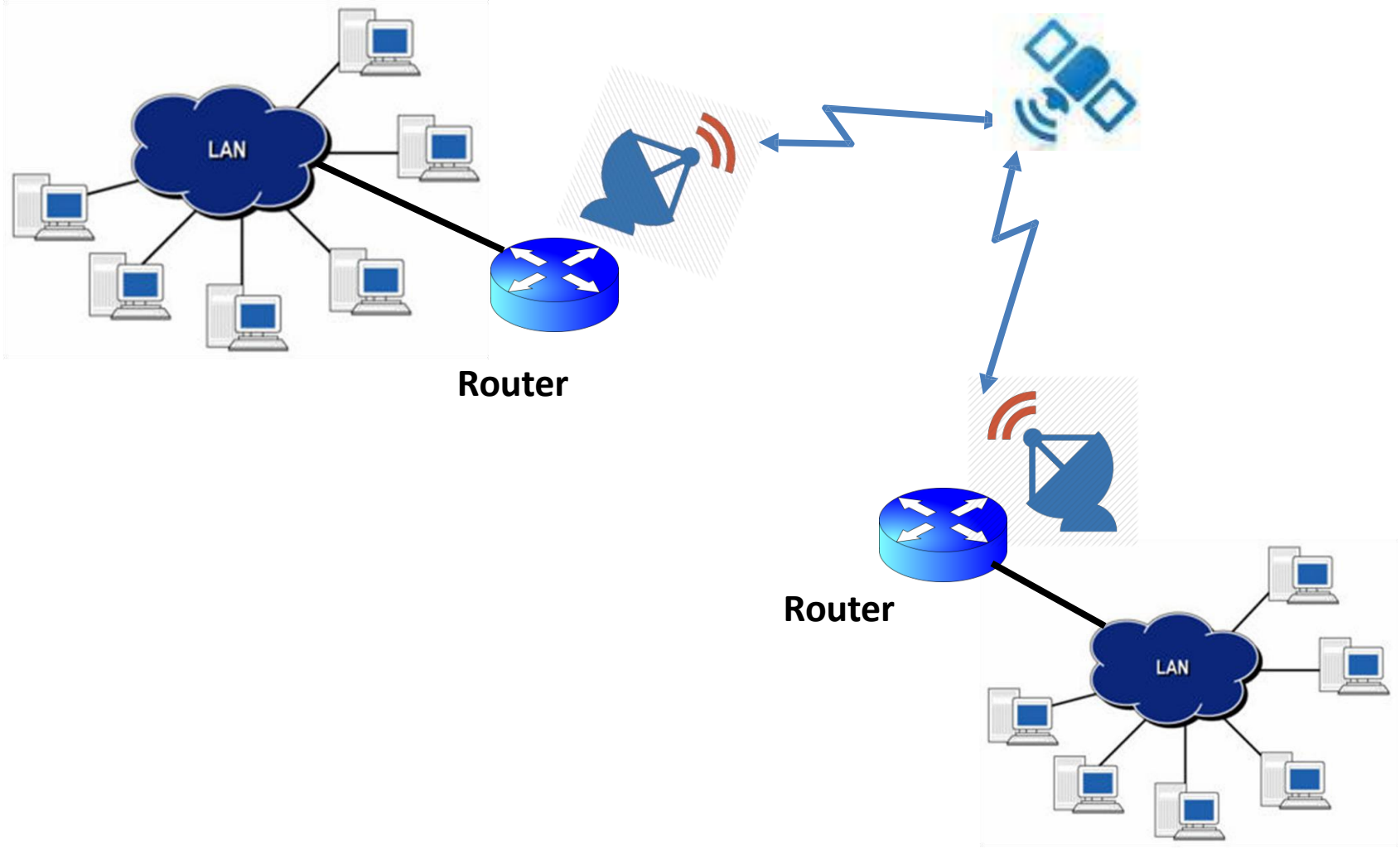


Router

Router



Διαμεταγωγή - Routing

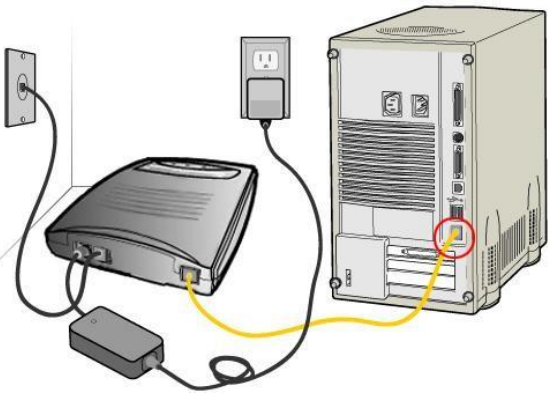


Διεμεταγωγή - Routing

Internet Service Provider (ISP)
Πάροχος Υπηρεσιών διαδικτύου

Modem/Router

Τηλεφωνικό Δίκτυο



Modem/Router

Πρωτόκολλα Επικοινωνιών

Για να μπορέσουν να επικοινωνήσουν δύο υπολογιστές μέσα σε ένα δίκτυο χρειάζεται να χρησιμοποιούν κοινούς κανόνες επικοινωνίας. **Πρωτόκολλο επικοινωνίας** είναι το σύνολο των κανόνων που ορίζει συγκεκριμένο τρόπο επικοινωνίας και ανταλλαγής των πληροφοριών/δεδομένων μεταξύ δυο οντοτήτων (π.χ. υπολογιστής, λογισμικό, άνθρωπος).

Στο χώρο των δικτύων υπάρχουν πολλά πρωτόκολλα επικοινωνίας τα οποία χρησιμοποιούνται είτε σε χαμηλό επίπεδο για την αποστολή και λήψη πακέτων πληροφοριών (**Transmission Control Protocol** - **User Datagram Protocol** / **Internet Protocol**) είτε σε υψηλότερο επίπεδο για τη παροχή υπηρεσιών, όπως για παράδειγμα η ηλεκτρονική αλληλογραφία (**Simple Mail Transfer Protocol**), ο παγκόσμιος ιστός (**Hyper Text Transfer Protocol**) κ.λπ.

Οι πληροφορίες στο δίκτυο μπορούν να ταξιδεύσουν με τη μορφή πακέτων δεδομένων, είτε μέσω του πρωτοκόλλου TCP είτε του πρωτοκόλλου UDP και ο καθένας από τους δύο έχει τα πλεονεκτήματα και τα μειονεκτήματα του.

Το πρωτόκολλο TCP

Το TCP λειτουργεί δημιουργώντας σύνδεση μεταξύ του αποστολέα και του παραλήπτη των δεδομένων. Μετά την επιτυχή σύνδεση, όλα τα δεδομένα αποστέλλονται από τον έναν υπολογιστή στον άλλο με την μορφή πακέτων μέσω αυτής της σύνδεση. Τα κύρια χαρακτηριστικά του TCP είναι τα εξής:

Αξιοπιστία - Το TCP χρησιμοποιεί διάφορους μηχανισμούς ώστε να διασφαλίσει ότι τα πακέτα που μεταδίδονται από τον αποστολέα θα φτάσουν σίγουρα στον παραλήπτη και στην σωστή σειρά. Οι μηχανισμοί αυτοί περιλαμβάνουν την επιβεβαίωση της λήψης πακέτου από τον παραλήπτη, την επαναποστολή των πακέτων που δεν έλαβε ο παραλήπτης.

Σειρά πακέτων - Εάν δύο πακέτα αποσταλούν σε μία σύνδεση το ένα μετά το άλλο, τότε το πρωτόκολλο TCP εγγυάται ότι θα φτάσουν στον παραλήπτη με την σωστή σειρά, όπως ακριβώς στάλθηκαν.

Αργό - Το πρωτόκολλο TCP θεωρείται ιδιαίτερα αργό, λόγω της δημιουργίας της σύνδεσης καθώς και των μηχανισμών αξιοπιστίας, στοιχεία που έχουν σημαντικό αντίκτυπο στην ταχύτητα μετάδοσης δεδομένων.

Το πρωτόκολλο UDP

Το UDP είναι ένα πιο απλό και «ελαφρύ» πρωτόκολλο σε σχέση με το TCP γιατί δεν υπάρχει η έννοια της σύνδεσης και κάθε πακέτο UDP, που ονομάζεται και «datagram», διανύει το δίκτυο ως μία ξεχωριστή αυτόνομη μονάδα και όχι ως μία σειρά πακέτων σε μία σύνδεση. Τα κύρια **χαρακτηριστικά του UDP** είναι τα εξής:

Αναξιόπιστο - Κατά την αποστολή ενός πακέτου, ο αποστολέας δεν είναι σε θέση να γνωρίζει εάν το πακέτο θα φτάσει σωστά στον προορισμό του ή εάν θα χαθεί μέσα στο δίκτυο. Δεν έχει προβλεφθεί η δυνατότητα επιβεβαίωσης λήψης πακέτου από τον παραλήπτη, ούτε η επαναμετάδοση ενός χαμένου πακέτου.

Δεν υπάρχει σειρά - Τα πακέτα UDP, σε αντίθεση με το TCP, δεν αριθμούνται και κατά συνέπεια δεν υπάρχει κάποια συγκεκριμένη σειρά με την οποία θα πρέπει να φτάσουν στον παραλήπτη.

Ελαφρύ – Τα ανωτέρω δύο στοιχεία καθιστούν το πρωτόκολλο ιδιαίτερα ελαφρύ κάτι έχει ως συνέπεια τη γρήγορη μετάδοση των δεδομένων.

Δικτυακή Διεύθυνση

Το κάθε πακέτο στο δίκτυο ενσωματώνει τις διευθύνσεις του αποστολέα καθώς και του παραλήπτη, ο τρόπος σύνταξης των οποίων καθορίζεται από το Internet Protocol . Υπάρχουν δύο εκδόσεις του πρωτοκόλλου, η 4 (IPv4) όπου ορίζει ότι το μήκος της διεύθυνσης είναι 32 bit ενώ στην νεότερη έκδοση την 6 (IPv4) το μήκος είναι 128 bit.

IPv4 60.23.20.30 Η διεύθυνση αποτελείται από 4 μέρη όπου το καθένα είναι ένα αριθμός από 0 έως 255 και μεταξύ τους χωρίζονται με τελεία. Με αυτό τον τρόπο σε ένα δίκτυο μπορούν να μετέχουν 2^{32} δηλαδή 4.294.967.296 υπολογιστές.

IPv6 FEDC:BA98:7654:3210:FEDC:BA98:7654:3210

Η διεύθυνση αποτελείται από 8 μέρη όπου το καθένα είναι ένας 16 bit αριθμός ρογ αναπαριστάται σε δεκαεξαδική μορφή και μεταξύ τους χωρίζονται με άνω κάτω τελεία. Με αυτό τον τρόπο σε ένα δίκτυο μπορούν να μετέχουν 2^{128} δηλαδή πάρα πολλοί υπολογιστές. Οι δύο εκδόσεις είναι συμβατές μεταξύ τους αφού η ανωτέρω v4 διεύθυνση μπορεί να αναπαρασταθεί σαν v6 ως

0000:0000:0000:0000:0000:0000:3C17:141E

Δημόσιες και ιδιωτικές δικτυακές διευθύνσεις

Η διαχείριση των δικτυακών διευθύνσεων βρίσκεται υπό τον έλεγχο του οργανισμού iana (http://www.iana.org), ο οποίος έχει χωρίσει τον κόσμο σε πέντε περιοχές /οργανισμούς και έχει μοιράσει το εύρος των διαθέσιμων διευθύνσεων σε αυτούς. Οι οποίοι με τη σειρά τους διαθέτουν τις IP διευθύνσεις σε κυβερνητικούς οργανισμούς και παρόχους , και αυτοί εν συνεχεία με τη σειρά τους, σε όλους εμάς τους τελικούς χρήστες.

Registry	Area Covered
AFRINIC	Africa Region
APNIC	Asia/Pacific Region
ARIN	North America Region
LACNIC	Latin America and some Caribbean Islands
RIPE NCC	Europe, the Middle East, and Central Asia

Δημόσιες και ιδιωτικές δικτυακές διευθύνσεις (συν.)

Για τη πρόσβαση ενός υπολογιστή στο διαδίκτυο χρειάζεται να υπάρχει μια IP διεύθυνση την οποία αποκτούμε μέσω ενός παρόχου και ονομάζεται δημόσια διεύθυνση. Για να στήσουμε ένα τοπικό δίκτυο μέσω του πρωτοκόλλου TCP/IP δεν χρειάζεται να καταφύγουμε σε ένα πάροχο αφού μπορούμε να χρησιμοποιήσουμε το εύρος των διευθύνσεων που ονομάζονται ιδιωτικές διευθύνσεις (private addresses)

IP ιδιωτικές διευθύνσεις	Πλήθος διευθύνσεων
10.0.0.0 - 10.255.255.255	16,777,216
172.16.0.0 - 172.31.255.255	1,048,576
192.168.0.0 - 192.168.255.255	65,536

Οι ιδιωτικές διευθύνσεις χρησιμοποιούνται όπως και οι δημόσιες, σε επίπεδο υπηρεσιών, αλλά σε τοπικό επίπεδο αφού δεν δρομολογούνται στο διαδίκτυο. Ο κάθε υπολογιστής που έχει εγκαταστήσει το πρωτόκολλο TCP/IP αποκτά και μια ειδικού τύπου διεύθυνση την 127.0.0.1 η οποία αναφέρεται στον ίδιο τον υπολογιστή.

Πόρτες Δικτυακών Συνδέσεων

Ο κάθε υπολογιστής που έχει εγκαταστήσει το πρωτόκολλο TCP/IP και έχει μια ή περισσότερες IP διευθύνσεις, ανεξαρτήτως αν είναι δημόσιες ή ιδιωτικές θωρούμε ότι έχει στη καθεμία από αυτές, 65535 **πόρτες συνδέσεων** και μπορεί οποιαδήποτε εφαρμογή λογισμικού να καταλάβει μια από αυτές και να παρέχει υπηρεσίες προς άλλους υπολογιστές. Αυτές οι εφαρμογές και κατ' επέκταση και οι υπολογιστές στους οποίους τρέχουν ονομάζονται **εξυπηρετητές** (servers).

Ο υπολογιστής που δημιουργεί σύνδεση (socket) σε μια IP διεύθυνση και σε μια πόρτα (socket port) για να επικοινωνήσει με ένα εξυπηρετητή, ονομάζεται **πελάτης** (client). Τη συγκεκριμένη σύνδεση λέμε ότι είναι τύπου **πελάτη-εξυπηρετητή** (client-server). Υπάρχουν και συνδέσεις όπου δεν υπάρχει ιεραρχία αλλά όλοι οι υπολογιστές είναι ίσοι μεταξύ τους, και ο καθένας ταυτόχρονα και σαν πελάτης και σαν εξυπηρετητής. Αυτά ονομάζονται **ομότιμα δίκτυα**.

Οι πόρτες επικοινωνίας που είθισται να προσφέρονται κάποιες δικτυακές υπηρεσίες ονομάζονται **γνωστές πόρτες** (well-known ports).

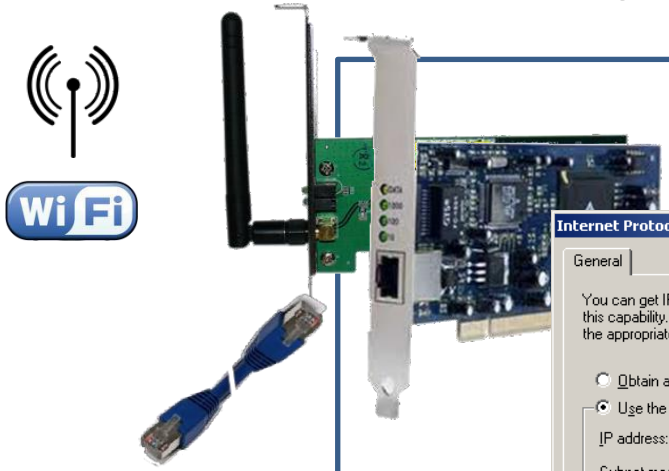
Domain Name Service

Ο οργανσιμός **Internet Corporation for Assigned Names and Numbers** (**ICANN)** είναι υπεύθυνος για το έλεγχο και συντονισμό των υπηρεσιών διευθυνσιοδότησης ονομάτων στο διαδίκτυο. Η χρήση των IP διευθύνσεων για τη σύνδεση και ανταλλαγή δεδομένων στο δίκτυο είναι προβληματική λόγω της δυσκολίας απομνημόνευσης τους, από τους ανθρώπους. Η υπηρεσία DNS επιτρέπει τη χρήση φιλικών ονομάτων, τα οποία αναφέρονται /μεταφράζονται σε IP διευθύνσεις.

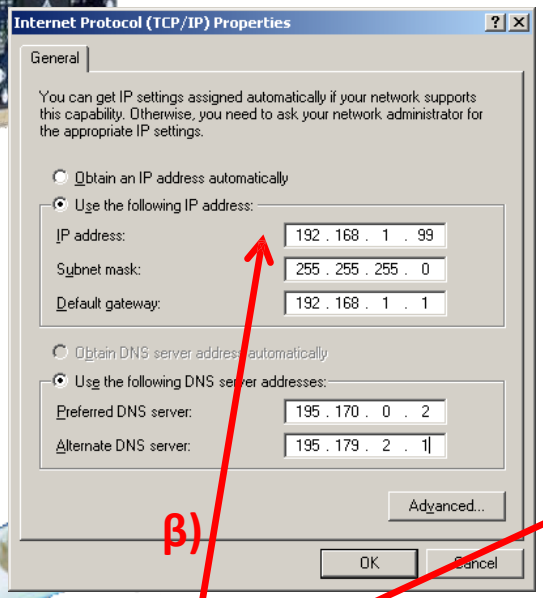
Η κάθε χώρα είναι υπεύθυνη για την υποστήριξη του πρώτου επιπέδου ονοματοδοσίας, που αναφέρεται σε αυτήν, όπου για την Ελλάδα το πρώτο επίπεδο είναι το .gr και υπεύθυνο ίδρυμα διαχείρισης είναι το Ινστιτούτο Πληροφορικής, στο Ίδρυμα Τεχνολογίας Έρευνας (http://www.gr)

Το ΙΤΕ είναι υπεύθυνο για το μητρώο Internet με κατάληξη .gr, και σε αυτό θα πρέπει να απευθυνθούμε για τη κατοχύρωση ονόματος, π.χ. mysite.gr και για τη προσθήκη στα μητρώα τους, της IP του μηχανήματος , το οποίο θα προσφέρει υπηρεσίες DNS για το mysite.gr. Στο συγκεκριμένο μηχάνημα θα τρέχει πάλι DNS υπηρεσία όπου θα διατηρεί τα ονόματα (π.χ. www ftp mail) και τις IPs που αντιστοιχούν σε αυτά.

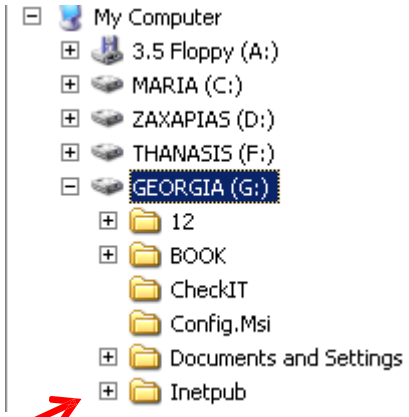
Ο υπολογιστής ενός HTTP εξυπηρετητή

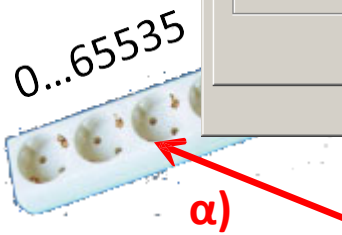


TCP/IP



Σύστημα Αρχείων





α)

β)

γ)

APACHE HTTP SERVER

IIS7 internet information services

Βασικές Ρυθμίσεις

α) Πόρτα επικοινωνίας
β) σε μια IP ή σε όλες τις IPs
γ) Φάκελος που εξυπηρετεί.

Το πακέτο java.net

Class Summary

Class	Description
Authenticator	The class Authenticator represents an object that knows how to obtain authentication for a network connection.
CacheRequest	Represents channels for storing resources in the ResponseCache.
CacheResponse	Represent channels for retrieving resources from the ResponseCache.
ContentHandler	The abstract class `ContentHandler` is the superclass of all classes that read an `Object` from a `URLConnection`.
CookieHandler	A CookieHandler object provides a callback mechanism to hook up a HTTP state management policy implementation into the HTTP protocol handler.
CookieManager	CookieManager provides a concrete implementation of `CookieHandler`, which separates the storage of cookies from the policy surrounding accepting and rejecting cookies.
DatagramPacket	This class represents a datagram packet.
DatagramSocket	This class represents a socket for sending and receiving datagram packets.
DatagramSocketImpl	Abstract datagram and multicast socket implementation base class.
HttpCookie	

Το πακέτο java.net (συν)

HttpCookie	An HttpCookie object represents an http cookie, which carries state information between server and user agent.
HttpURLConnection	A URLConnection with support for HTTP-specific features.
IDN	Provides methods to convert internationalized domain names (IDNs) between a normal Unicode representation and an ASCII Compatible Encoding (ACE) representation.
Inet4Address	This class represents an Internet Protocol version 4 (IPv4) address.
Inet6Address	This class represents an Internet Protocol version 6 (IPv6) address.
InetAddress	This class represents an Internet Protocol (IP) address.
InetSocketAddress	This class implements an IP Socket Address (IP address + port number) It can also be a pair (hostname + port number), in which case an attempt will be made to resolve the hostname.
InterfaceAddress	This class represents a Network Interface address.
JarURLConnection	A URL Connection to a Java ARchive (JAR) file or an entry in a JAR file.
MulticastSocket	The multicast datagram socket class is useful for sending and receiving IP multicast packets.
NetPermission	This class is for various network permissions.
NetworkInterface	This class represents a Network Interface made up of a name, and a list of IP addresses assigned to this interface.

Το πακέτο java.net (συν)

PasswordAuthentication	The class PasswordAuthentication is a data holder that is used by Authenticator.
Proxy	This class represents a proxy setting, typically a type (http, socks) and a socket address.
ProxySelector	Selects the proxy server to use, if any, when connecting to the network resource referenced by a URL.
ResponseCache	Represents implementations of URLConnection caches.
SecureCacheResponse	Represents a cache response originally retrieved through secure means, such as TLS.
ServerSocket	This class implements server sockets.
Socket	This class implements client sockets (also called just "sockets").
SocketAddress	This class represents a Socket Address with no protocol attachment.
SocketImpl	The abstract class `SocketImpl` is a common superclass of all classes that actually implement sockets.
SocketPermission	This class represents access to a network via sockets.
StandardSocketOptions	Defines the *standard* socket options.
URI	Represents a Uniform Resource Identifier (URI) reference.
URL	Class `URL` represents a Uniform Resource Locator, a pointer to a "resource" on the World Wide Web.
URLClassLoader	This class loader is used to load classes and resources from a search path of URLs

Το πακέτο java.net (συν)

URLConnection	The abstract class `URLConnection` is the superclass of all classes that represent a communications link between the application and a URL.
URLDecoder	Utility class for HTML form decoding.
URLEncoder	Utility class for HTML form encoding.
URLStreamHandler	The abstract class `URLStreamHandler` is the common superclass for all stream protocol handlers

Παράδειγμα δικτυακού εξυπηρετητή

Εκκινεί ο εξυπηρετητής και περιμένει για συνδέσεις σε συγκεκριμένη IP διεύθυνση και πόρτα σύνδεσης. ServerSocket

Ο εξυπηρετητής αποδέχεται τη σύνδεση. accept

Ο πελάτης κάνει σύνδεση στον εξυπηρετητή. Socket

Ο εξυπηρετητής αποθηκεύει τα ρεύματα εισόδου-εξόδου της σύνδεσης.

Ο πελάτης αποθηκεύει τα ρεύματα εισόδου-εξόδου της σύνδεσης.

Ο εξυπηρετητής και πελάτης διαβάζουν από τα αντίστοιχα ρεύματα εισόδου της σύνδεσης τους και γράφουν στα αντίστοιχα ρεύματα εξόδους της σύνδεσης τους.

Ο εξυπηρετητής κλείνει τη σύνδεση.

Ο πελάτης κλείνει τη σύνδεση.

Κατασκευαστές τηςκλάσης ServerSocket

Constructors

Constructor and Description
`ServerSocket()` Creates an unbound server socket.
`ServerSocket(int port)` Creates a server socket, bound to the specified port.
`ServerSocket(int port, int backlog)` Creates a server socket and binds it to the specified local port number, with the specified backlog.
`ServerSocket(int port, int backlog, InetAddress bindAddr)` Create a server with the specified port, listen backlog, and local IP address to bind to.

port - the port number, or 0 to use a port number that is automatically allocated.
backlog - requested maximum length of the queue of incoming connections.
InetAddress – Η κλάση αναπαριστά μια IP address

Μέθοδοι τηςκλάσης ServerSocket

Methods

Modifier and Type	Method and Description
`Socket`	`accept()` Listens for a connection to be made to this socket and accepts it.
`void`	`bind(SocketAddress endpoint)` Binds the `ServerSocket` to a specific address (IP address and port number).
`void`	`bind(SocketAddress endpoint, int backlog)` Binds the `ServerSocket` to a specific address (IP address and port number).
`void`	`close()` Closes this socket.
`ServerSocketChannel`	`getChannel()` Returns the unique `ServerSocketChannel` object associated with this socket, if any.
`InetAddress`	`getInetAddress()` Returns the local address of this server socket.
`int`	`getLocalPort()` Returns the port number on which this socket is listening.
`SocketAddress`	`getLocalSocketAddress()`

Κατασκευαστές τηςκλάσης Socket

Constructors

Modifier	Constructor and Description
	`Socket()` Creates an unconnected socket, with the system-default type of SocketImpl.
	`Socket(InetAddress address, int port)` Creates a stream socket and connects it to the specified port number at the specified IP address.
	`Socket(InetAddress host, int port, boolean stream)` **Deprecated.** *Use DatagramSocket instead for UDP transport.*
	`Socket(InetAddress address, int port, InetAddress localAddr, int localPort)` Creates a socket and connects it to the specified remote address on the specified remote port.
	`Socket(Proxy proxy)` Creates an unconnected socket, specifying the type of proxy, if any, that should be used regardless of any other settings.
`protected`	`Socket(SocketImpl impl)` Creates an unconnected Socket with a user-specified SocketImpl.

Μέθοδοι τηςκλάσης Socket

InetAddress	getInetAddress() Returns the address to which the socket is connected.
InputStream	getInputStream() Returns an input stream for this socket.
boolean	getKeepAlive() Tests if SO_KEEPALIVE is enabled.
InetAddress	getLocalAddress() Gets the local address to which the socket is bound.
int	getLocalPort() Returns the local port number to which this socket is bound.
SocketAddress	getLocalSocketAddress() Returns the address of the endpoint this socket is bound to, or null if it is not bound yet.
boolean	getOOBInline() Tests if OOBINLINE is enabled.
OutputStream	getOutputStream() Returns an output stream for this socket.
int	getPort() Returns the remote port number to which this socket is connected.
int	getReceiveBufferSize()

EchoServer

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

public class EchoServer {
    public static void main(String[] args) throws IOException {

        if (args.length != 1) {
            System.err.println("Usage: java EchoServer <port number>");
            System.exit(1);
        }

        int portNumber = Integer.parseInt(args[0]);
```

EchoServer (συνέχεια)

```
try (
        ServerSocket serverSocket =
            new ServerSocket(portNumber );
        Socket clientSocket = serverSocket.accept();
        PrintWriter out =
            new PrintWriter(clientSocket.getOutputStream(), true);
        BufferedReader in = new BufferedReader(
            new InputStreamReader(clientSocket.getInputStream()));
    ) {
        String inputLine;
        while ((inputLine = in.readLine()) != null) {
            out.println(inputLine);
        }
    } catch (IOException e) {
        System.out.println("An exception has occurred on port "
            + portNumber);
        System.out.println(e.getMessage());
    }
```

EchoClient

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

public class EchoClient {
    public static void main(String[] args) throws IOException {

        if (args.length != 2) {
            System.err.println(
                "Usage: java EchoClient <host name> <port number>");
            System.exit(1);
        }

        String hostName = args[0];
        int portNumber = Integer.parseInt(args[1]);
```

EchoClient (συνέχεια)

```
try (
        Socket echoSocket = new Socket(hostName, portNumber);
        PrintWriter out = new PrintWriter(echoSocket.getOutputStream(), true);
        BufferedReader in =
           new BufferedReader( new InputStreamReader(echoSocket.getInputStream()));
        BufferedReader stdIn = new BufferedReader( new InputStreamReader(System.in))
    ) {
        String userInput;
        while ((userInput = stdIn.readLine()) != null) {
           out.println(userInput);
           System.out.println("echo: " + in.readLine());
        }
    } catch (UnknownHostException e) {
        System.err.println("Don't know about host " + hostName);
        System.exit(1);
    } catch (IOException e) {
        System.err.println("Couldn't get I/O for the connection to " + hostName);
        System.exit(1);
    }
  }
}
```

UDP Server

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

class UDPServer
{
  public static void main(String args[]) throws Exception
    {
        DatagramSocket serverSocket = new DatagramSocket(7777);
        byte[] receiveData = new byte[1024];
        byte[] sendData = new byte[1024];
        while(true)
          {
            DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
            serverSocket.receive(receivePacket);
            String sentence = new String( receivePacket.getData());
            System.out.println("Your message is: " + sentence);
```

UDP Server (συν)

```
 InetAddress IPAddress = receivePacket.getAddress();
                int port = receivePacket.getPort();
                String strServerResponce = sentence.toUpperCase();
                sendData = strServerResponce.getBytes();
                DatagramPacket sendPacket =
                new DatagramPacket(sendData, sendData.length, IPAddress, port);
                serverSocket.send(sendPacket);
            }
        }
}
```

UDP Client

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

class UDPClient
{
   public static void main(String args[]) throws Exception
   {
      BufferedReader keybInput =
         new BufferedReader(new InputStreamReader(System.in));
      DatagramSocket clientSocket = new DatagramSocket();
      InetAddress IPAddress = InetAddress.getByName("127.0.0.1");
      byte[] sendData = new byte[1024];
      byte[] receiveData = new byte[1024];
      String sentence = keybInput.readLine();
      sendData = sentence.getBytes();
```

UDP Client

```
 DatagramPacket sendPacket = new DatagramPacket
                                                    (sendData, sendData.length, IPAddress, 7777);
    clientSocket.send(sendPacket);
    DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
    clientSocket.receive(receivePacket);
    String srvResponce = new String(receivePacket.getData());
    System.out.println("The server response :" + srvResponce);
    clientSocket.close();
  }
}
```