ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΔΙΚΤΥΩΝ

ΔΙΑΛΕΞΗ 4

ΔΙΔΑΣΚΩΝ: ΑΝΑΡΓΥΡΟΣ ΣΙΔΕΡΗΣ

<sideris@epp.teiher.gr>

<https://eclass2.teicrete.gr/courses/TP182/>

ΤΜΗΜΑ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΠΟΛΥΜΕΣΩΝ

ΤΕΙ ΚΡΗΤΗΣ

T.E.I. of Crete

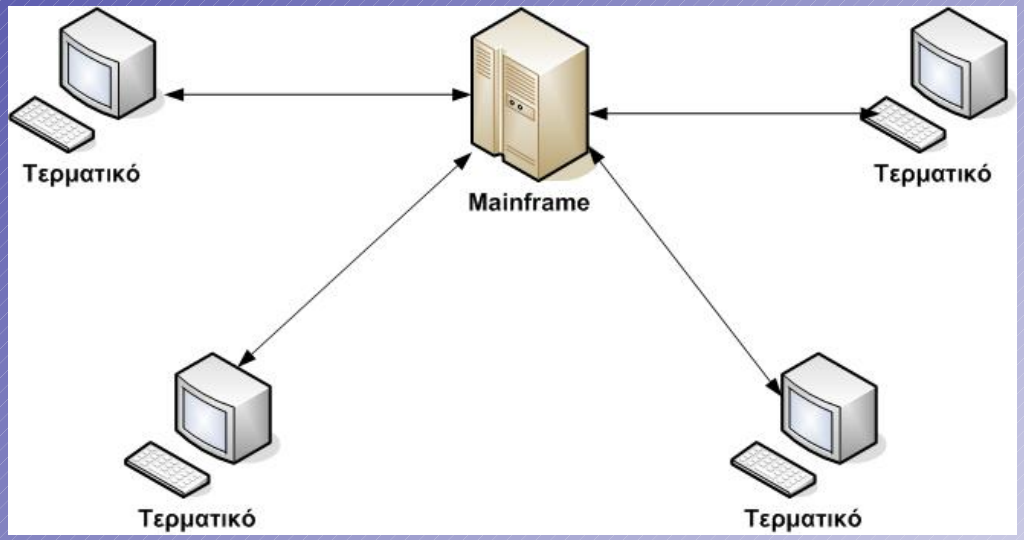
ΜΟΝΤΕΛΑ ΔΙΚΤΥΑΚΟΥ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Γενικά

- **ΜΟΝΤΕΛΑ ΔΙΚΤΥΑΚΟΥ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ**
  - **Mainframe Μοντέλο.**
  - **Μοντέλο Πελάτη-Εξυπηρετητή (Client-Sever).**
    - **2-tier**
    - **3-tier**
  - **Μοντέλο Peer to Peer.**
    - **Τοπολογία**
    - **Αλγόριθμοι.**

Μοντέλο Mainframe

- Όλες οι διεργασίες εκτελούνται μέσα σε ένα κεντρικό υπολογιστή.
- Η επικοινωνία μεταξύ χρηστών και κεντρικού υπολογιστή γίνεται μέσω απλών τερματικών.



Μοντέλο Client-Server

- Στο μοντέλο αυτό υπάρχει τουλάχιστον ένας πελάτης και ένας εξυπηρετητής.
- Οι πελάτες αιτούνται, οι εξυπηρετητές απαντούν.
  - Πελάτης ή εξυπηρετητής μπορεί να είναι μια διεργασία, εφαρμογή ή συσκευή.
- Συνήθως ο πελάτης αρχίζει την επικοινωνία ενώ ο εξυπηρετητής επεξεργάζεται τις αιτήσεις και στέλνει πίσω τις απαντήσεις.

Χαρακτηριστικά του C/S μοντέλου

- Συμμετρικές Λειτουργίες
- Ανεξάρτητο τοποθεσίας (Location Transparency).
- Ανεκτικό σε λάθη
- Ανεξάρτητο από το είδος πλατφόρμας.
- Διαμοιρασμό πόρων
- Επεκτασιμότητα

Λειτουργικές μονάδες του C/S μοντέλου (1)

- Τρείς λειτουργικές μονάδες:
  - Μονάδα Αναπαράστασης (Presentation Logic/User Interface)
  - Μονάδα Εφαρμογής (Application/Business Logic).
  - Μονάδα Υπηρεσίας Δεδομένων (Data Service).
- Μονάδα Αναπαράστασης (Presentation Logic/User Interface)
  - Συλλογή πληροφοριών από χρήστη.
  - Αποστολή των πληροφοριών στην μονάδα εφαρμογής για επεξεργασία.
  - Αποδοχή αποτελεσμάτων από την μονάδα εφαρμογής.
  - Παρουσίαση αποτελεσμάτων στον χρήστη.

Λειτουργικές μονάδες του C/S μοντέλου (2)

- Μονάδα Εφαρμογής (Application/Business Logic).
  - Δέχεται τα δεδομένα από την μονάδα αναπαράστασης.
  - Αλληλεπιδρά με την μονάδα υπηρεσίας δεδομένων ώστε να εκτελέσει τις απαιτούμενες εργασίες.
  - Στέλνει τα αποτέλεσματα των εργασιών στην μονάδα υπηρεσίας.

Λειτουργικές μονάδες του C/S μοντέλου (3)

- **Μονάδα Υπηρεσίας Δεδομένων (Data Service).**
  - **Αποθήκευση Δεδομένων.**
  - **Ανάκτηση Δεδομένων.**
  - **Διαχείριση Δεδομένων.**
  - **Ακεραιότητα Δεδομένων.**

Αρχιτεκτονική 2-tier (C/S) (1)

- Σε αυτή την αρχιτεκτονική η μονάδα εφαρμογής βρίσκεται:
  - Εξολοκλήρου μέσα στην μονάδα αναπαράστασης.
  - Εξολοκλήρου μέσα στην μονάδα υπηρεσίας δεδομένων με την μορφή αποθηκευμένων επεξεργασιών.
  - Μοιρασμένη ανάμεσα στις μονάδες υπηρεσίας δεδομένων και αναπαράστασης.
- Κλάσεις του 2-tier μοντέλου
  - Host based processing (dumb client).
  - Server based processing.
  - Cooperative processing.
  - Client based Processing.

Host based Processing

- Host based processing (dumb client).
  - Περιβάλλον όπου υπάρχει ένας κεντρικός υπολογιστής που τα κάνει όλα. Ο ρόλος του πελάτης είναι περιορισμένος.

Server based processing.

Server based processing.

- Ο πελάτης παρέχει την διεπαφή (interface).
- Thin client, Fat Server

Cooperative processing

- Cooperative processing.
  - Κατανεμημένη επεξεργασία

Client based Processing

- Client based Processing.
  - Όλη η επεξεργασία λαμβάνει χώρα στον πελάτη.
  - Fat client, Thin Server

Αρχιτεκτονική 3-tier (C/S)

- Σε αυτή την αρχιτεκτονική η μονάδα εφαρμογής είναι χωρισμένη από τις μονάδες παρουσίασης και υπηρεσίας δεδομένων.
- Με αυτό τον τρόπο οι επεξεργασίες μπορούν να διαχωριστούν πλήρως από την διεπαφή χρήστη και την διαχείριση των δεδομένων.
- Παράδειγμα:
  - Ο πελάτης (1ος tier) αιτείται υπηρεσίες μέσω της μονάδας εφαρμογής (2ος tier) η οποία επικοινωνεί με τον εξυπηρετητή (3ος tier) για την διεκπεραίωση τω αιτήσεων.

3-tier vs 2-tier

- 2-tier
  - Κατάλληλο για μικρά project.
  - Μικρό αριθμό χρηστών.
  - Χαμηλό φόρτο εργασίας.
- 3-tier
  - Κατάλληλο για μεγάλα project.
  - Μεγάλο αριθμό χρηστών.
  - Διαμοιρασμός φόρτου.
  - Ασφαλέστερο.
  - Επαναχρησιμοποίηση κώδικα.
  - Επεκτάσιμο.
  - Ιδανικό για εργασία σε ομάδες.

Προκλήσεις του C/S μοντέλου

- Συντήρηση-Διαχείριση.
  - Οι πελάτες και οι εξυπηρετητές τρέχουν σε διαφορετικά συστήματα λογισμικού και υλικού.
  - Είναι διασκορπισμένοι μέσα στο δίκτυο.
  - Είναι πρόκληση να κρατηθούν όλα τα μέρη ενημερωμένα σε τυχόν αλλάγες λειτουργικών παραμέτρων.
- Αναβάθμιση.
  - Λόγω της αλληλεξάρτησης του πελάτη με τον εξυπηρετητή σε περίπτωση αναβάθμισης του ενός επηρεάζεται και ο άλλος.

Peer to Peer μοντέλο

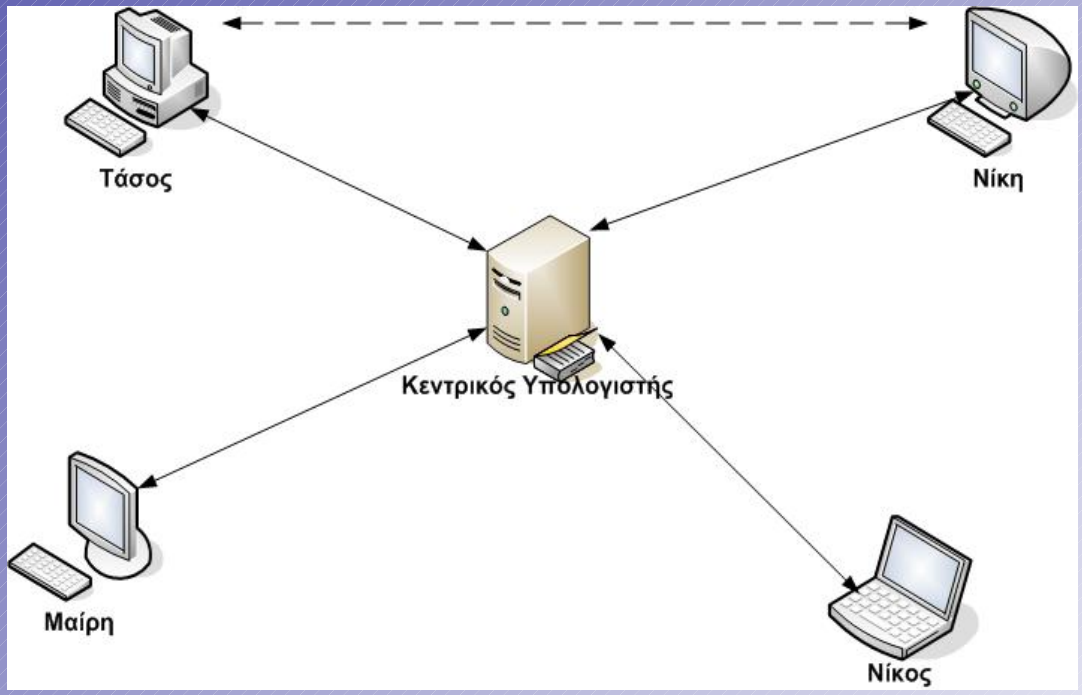
- Ομότιμες οντότητες.
  - Ο ρόλος του πελάτη και του εξυπηρετητή συνυπάρχουν σε κάθε οντότητα.
  - Κάθε οντότητα συνεισφέρει πόρους τόσο σε αποθηκευτικό χώρο, εύρος ζώνης και υπολογιστική ισχύ.

Κατηγοριοποίηση P2P συστημάτων

- Ανάλογα με την εφαρμογή τους.
  - Διαμοιρασμός αρχείων (Kazaa, Napster).
  - Τηλεφωνία (Skype).
  - Video/Audio Streaming (Peercast, Freecast).
  - Ομάδες συζητήσεων (IRC, Usenet).
  - Εκτέλεση χρονοβόρων διεργασιών (SETI).
- Ανάλογα την αρχιτεκτονική τους.
  - Κεντρικά p2p συστήματα.
  - Αποκεντρωμένα p2p συστήματα (“καθαρό” p2p).
  - Υβριδικά p2p συστήματα.
- Ανάλογα την δομή τους.
  - Δομημένα p2p συστήματα.
  - Μη δομημένα p2p συστήματα.

Κεντρικά P2P Συστήματα (1)

- Υπάρχει ένας κεντρικός υπολογιστής.
- Κάθε p2p οντότητα καταχωρεί πληροφορίες (διεύθυνση,περιεχόμενο) στο κεντρικό υπολογιστή.
- Κάθε p2p οντότητα χρησιμοποιεί τις αποθηκευμένες πληροφορίες στον κεντρικό υπολογιστή για να εντοπίσει το επιθυμητό περιεχόμενο.

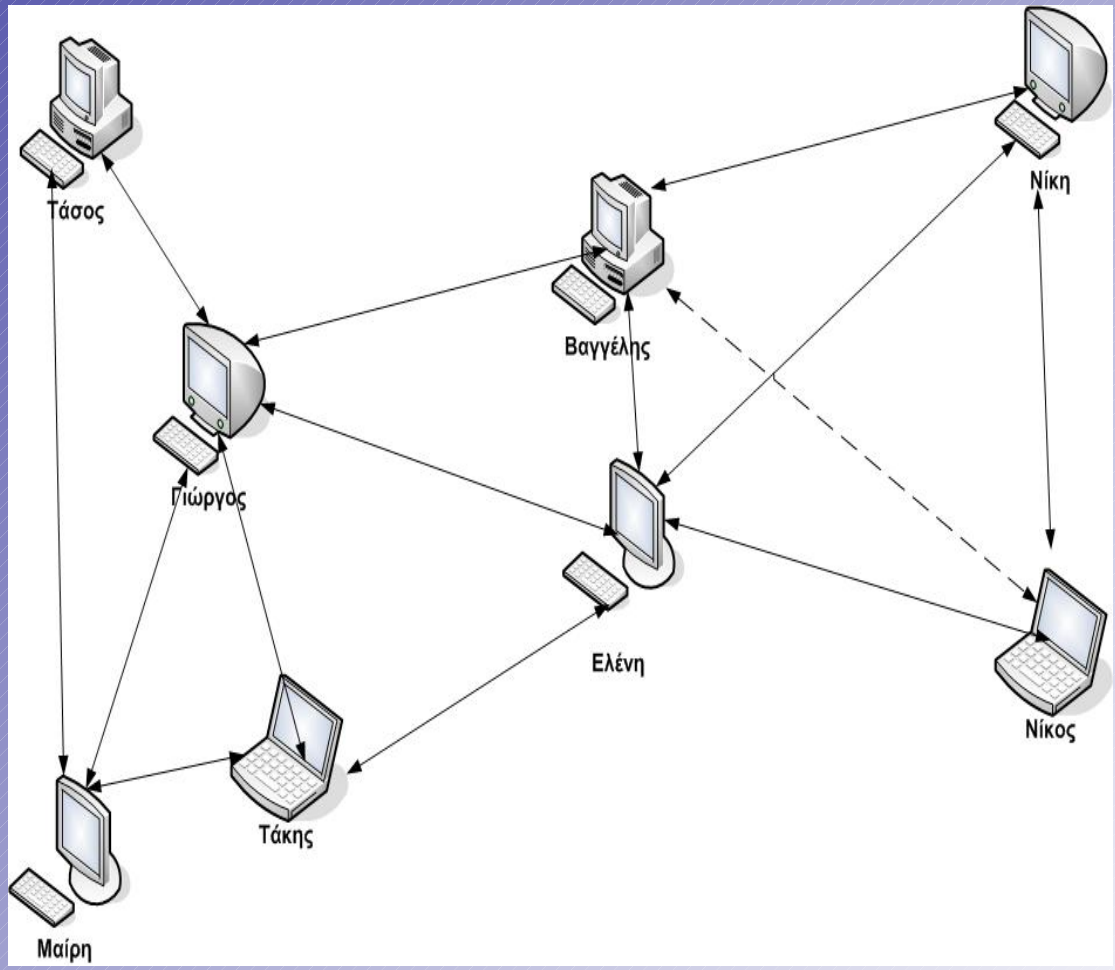


Κεντρικά P2P Συστήματα (2)

- Πλεονεκτήματα
  - Αποτελεσματική έρευνα/ανακάλυψη περιεχομένου.
    - Περιορισμένη χρήση του εύρους ζώνης.
    - Κάθε κόμβος δεν χρειάζεται να έχει πληροφορίες για την κατάσταση των γειτόνων του.
- Μειονεκτήματα
  - Κεντρικό σημείο αποτυχίας.
  - Περιορισμένη επεκτασιμότητα.
  - Κεντρικό σημείο ελέγχου.
- Παράδειγμα
  - Napster.

Αποκεντρωμένα p2p συστήματα (1)

- Σε αυτά τα συστήματα δεν υπάρχει κεντρικός υπολογιστής.
- Κάθε κόμβος χρησιμοποιεί το δίκτυο για να βρεί κάποιον άλλο
- Κάθε κόμβος γνωρίζει συνήθως μόνο τους άμεσους γείτονες του.
- Η συνεργασία μεταξύ των γειτόνων αποφέρει το επιθυμητό αποτέλεσμα (εύρεση αρχείων).

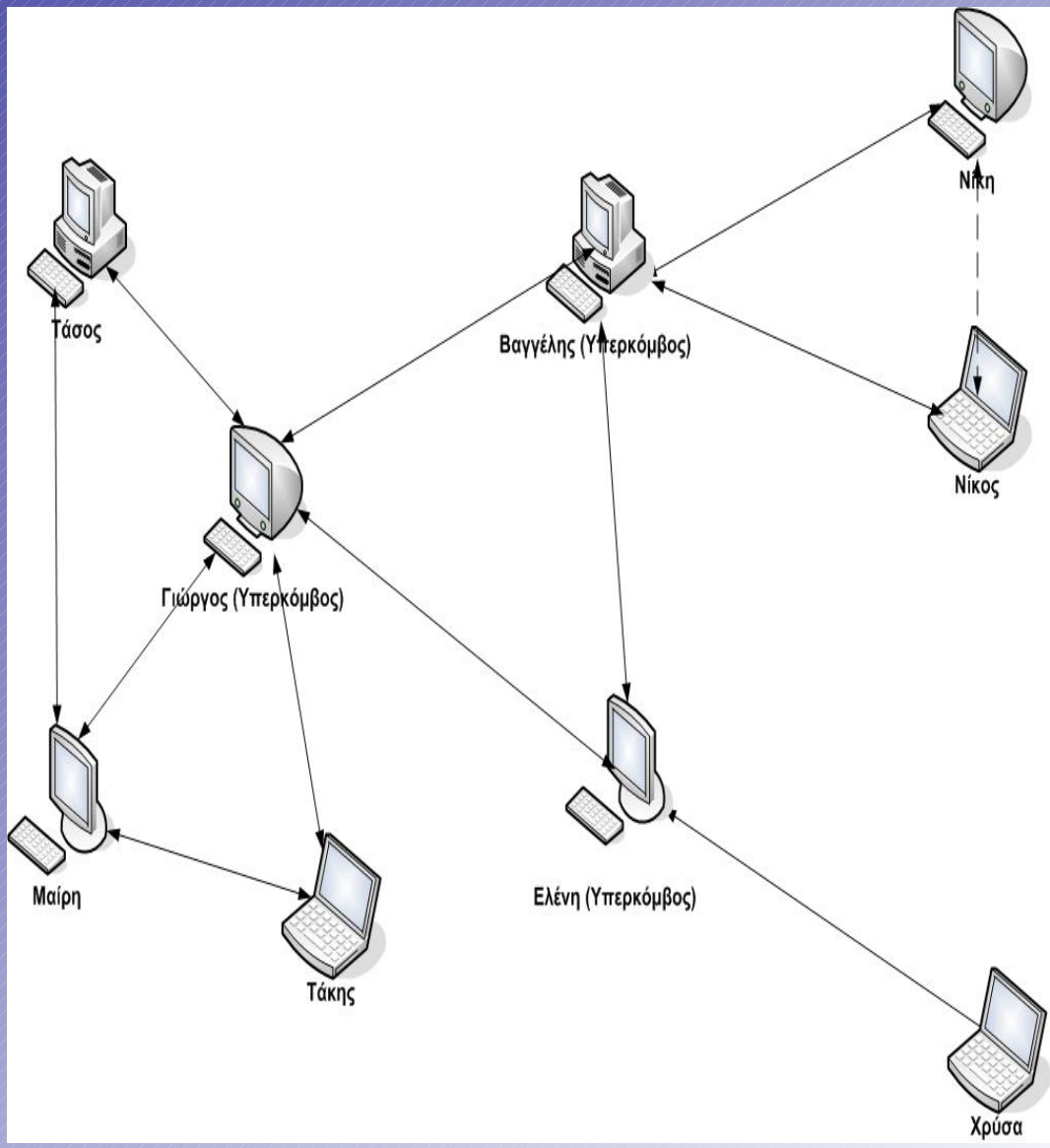


Αποκεντρωμένα p2p συστήματα (2)

- Πλεονεκτήματα:
  - Μη ύπαρξη κεντρικού σημείου αποτυχίας.
  - Κάθε κόμβος έχει πληροφορίες για την κατάσταση μόνο των γειτόνων του.
- Μειονεκτήματα:
  - Χρονοβόρες έρευνες ανακάλυψης αρχείων.
  - Εξάντληση των διαθέσιμων δικτυακών πόρων (bandwidth).
- Παράδειγμα:
  - Gnutella

Υβριδικά p2p συστήματα (1)

- Συνδυάζει χαρακτηριστικά των δύο προηγούμενων αρχιτεκτονικών.
- Ύπαρξη κόμβων (Υπέρ-κόμβοι) που είναι πιο «ισότιμοι» από τους άλλους.
- Οι Υπερκόμβοι εξυπηρετούν τους κόμβους της γειτονιάς τους.
- Η επικοινωνία με κόμβους άλλων γειτονιών γίνεται μέσω των υπερκόμβων.



Υβριδικά p2p συστήματα (2)

- Πλεονεκτήματα:
  - Αποτελεσματικό στην έρευνα και εντοπισμό περιεχομένου.
  - Δυναμική διαχείριση των υπερκόμβων.
  - Επεκτασιμότητα
- Μειονεκτήματα:
  - Κεντρικά σημεία διαχείρισης.
- Παράδειγμα:
  - Kazaa

Οφέλη των p2p συστημάτων

- Διαμοιρασμός φόρτου.
- Ανεκτικά σε περιπτώσεις σφάλματος σε ένα ή περισσότερους κόμβους.
- Για την περίπτωση του «καθαρού» p2p μοντέλου.
  - Απουσία κεντρικού ελέγχου!!!

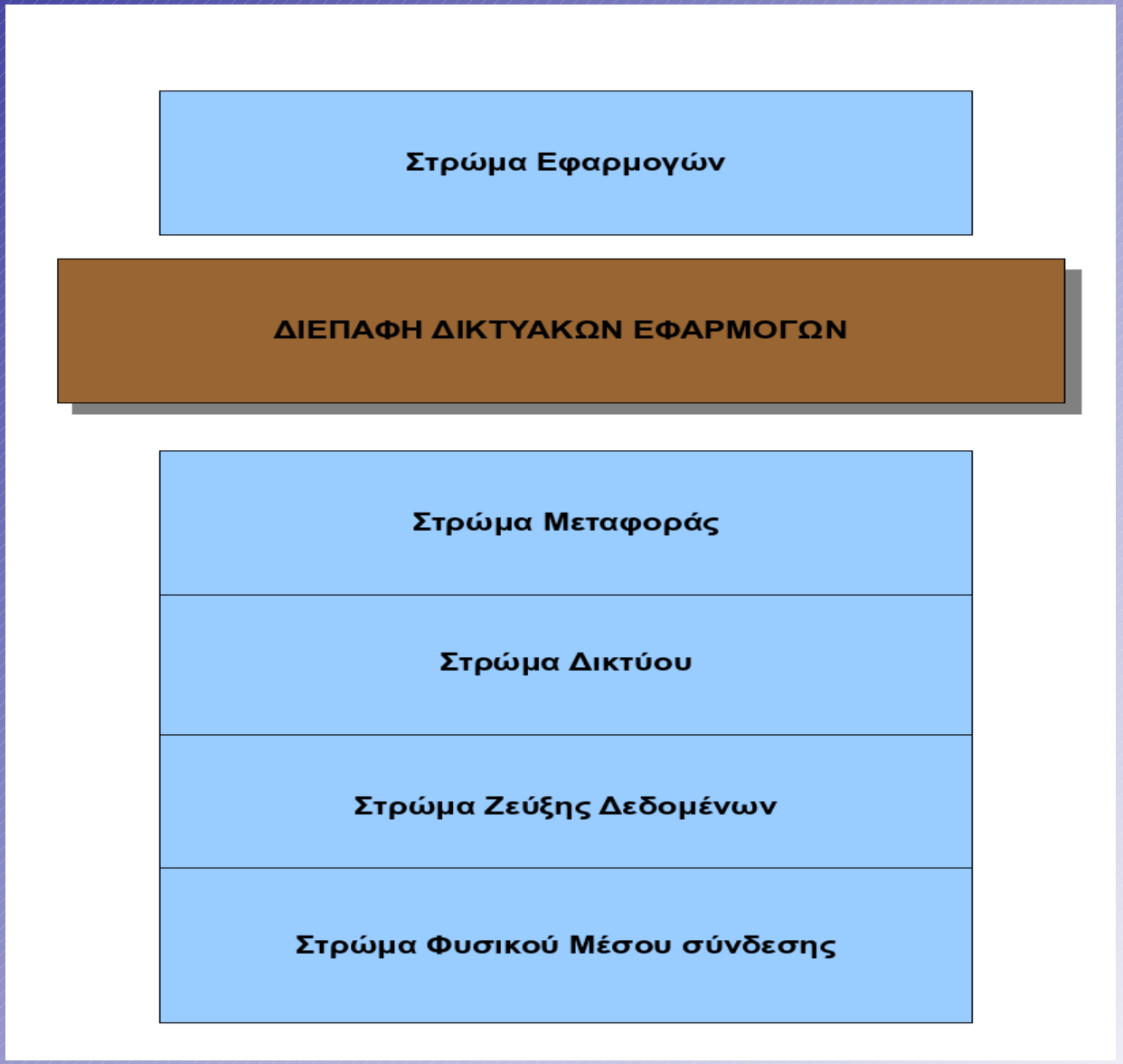
Προκλήσεις των p2p συστημάτων

- Απροβλεπτότητα
- Αξιοπιστία
- Διαθεσιμότητα
- Έλεγχος κόμβων και περιεχομένου.
- Αποτελεσματική έρευνα και ανακάλυψη κόμβων και περιεχομένου.
- Ασφάλεια.

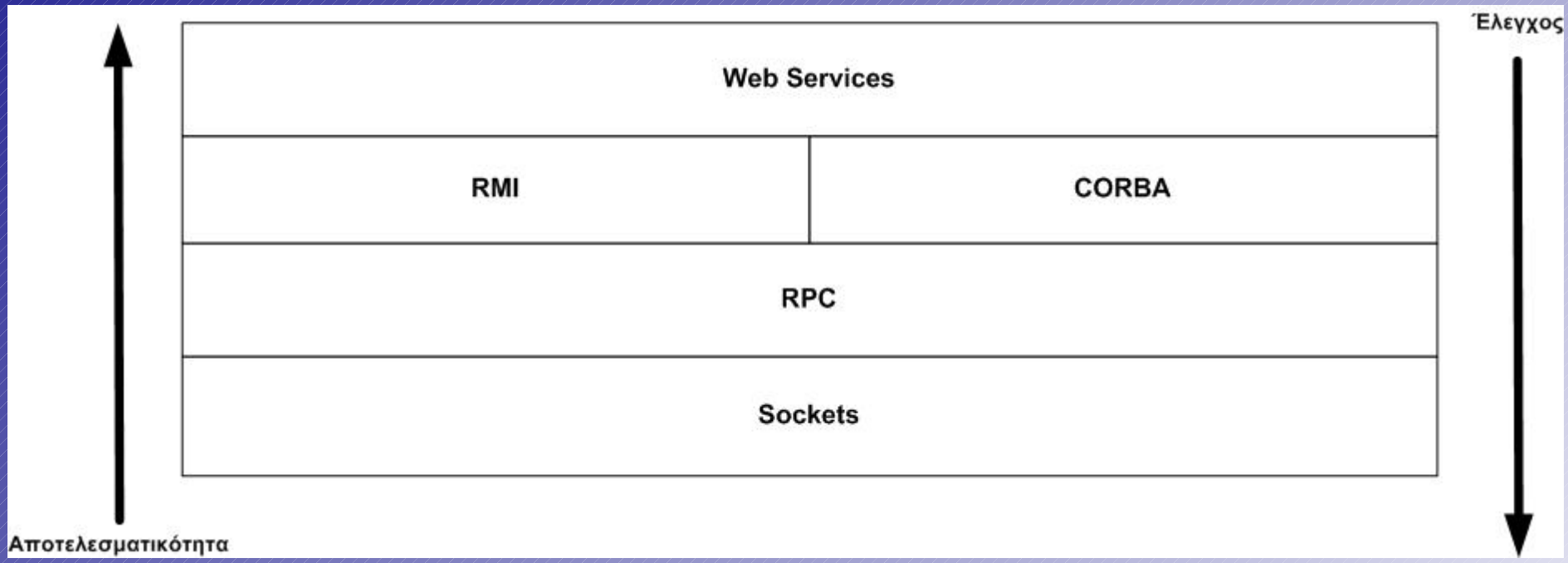
Υλοποίηση Δικτυακών Εφαρμογών

ΔΙΕΠΑΦΕΣ ΔΙΚΤΥΑΚΩΝ ΕΦΑΡΜΟΓΩΝ

- Sockets
- RPC (Remote Procedure Call)
- RMI (Remote Method Invocation)
- CORBA (Common Object Request Broker)
- Web Services



Ιεραρχία τεχνολογιών



Sockets (1)

Τι είναι socket;

- Τερματικό σημείο επικοινωνίας δικτυακών προγραμμάτων.
- Είναι μια διεπαφή που δίνει πρόσβαση στις εφαρμογές στις υπηρεσίες του δικτύου.
- Περιλαμβάνει την διεύθυνση, θύρα και είδος πρωτοκόλλου επικοινωνίας που χρησιμοποιεί το τερματικό σημείο σύνδεσης.

Sockets (2)

- Τα socket αναπτύχθηκαν πρώτη φορά από το πανεπιστήμιο Berkeley σε unix συστήματα (BSD Sockets).
- Λόγω της μεγάλης τους επιτυχίας υιοθετήθηκαν και από άλλους κατασκευαστές λογισμικού (WinSock, java sockets, MacTcp).
- Τα sockets στην πράξη είναι ένα σύνολο από:
  - Κλήσεις συστήματος (ρουτίνες).
  - Δομές.
- Οι κλήσεις συστήματος παρέχουν τις υπηρεσιές δικτύου όπως:
  - Καθορισμός τερματικών σημείων επικοινωνίας.
  - Άνοιγμα/κλείσιμο σύνδεσης.
  - Αποστολή/λήψη δεδομένων.
  - Διαχείριση σφαλμάτων.
- Οι δομές περιγράφουν τα δεδομένα που υπάρχουν στο δίκτυο:
  - Διευθύνσεις.
  - Θύρες.

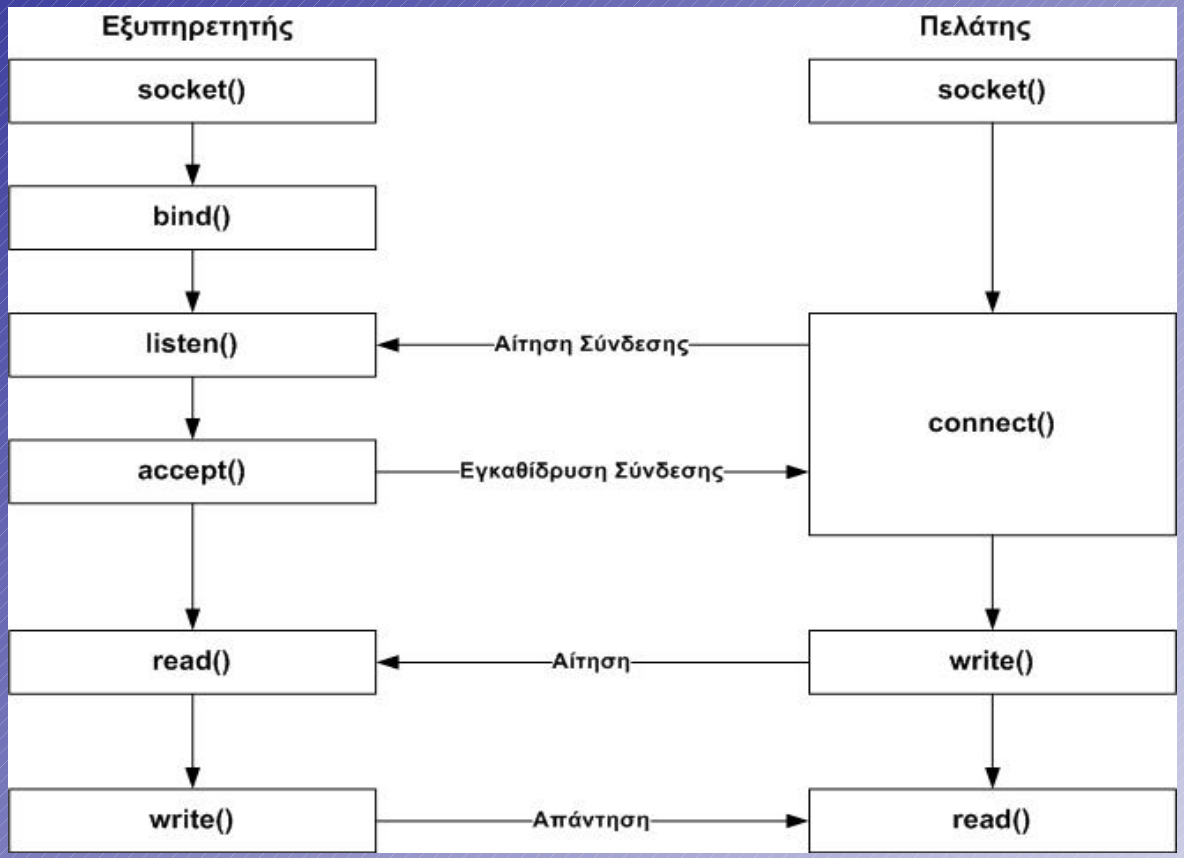
Είδη των Sockets

- Connection-oriented
  - Παρέχουν αξιόπιστες υπηρεσίες σύνδεσης.
  - TCP socket
- Connectionless
  - Μη αξίοπιστη μετάδοση δεδομένων.
  - UDP socket
- Raw
  - Παρέχει πρόσβαση σε κατώτερου επιπέδου πρωτόκολλλα δικτύου.
  - IP, ICMP

Λειτουργίες των sockets με σύνδεση (1)

- Κλήσεις συστήματος στην μεριά του εξυπηρετητή.
  - Socket(): Δημιουργεί το τερματικό σημείο σύνδεσης, προσδιορίζει το πρωτόκολλο επικοινωνίας.
  - Bind(): καταχωρεί και προσαρτά το socket στήν επιθυμητή θύρα επικοινωνίας.
  - Listen(): Παρακολουθεί για τυχόν αιτήσεις σύνδεσης.
  - Accept: Αποδοχή σύνδεσης.
  - Read(): Λήψη Δεδομένων.
  - Write(): Αποστολή Δεδομένων.
- Κλήσεις συστήματος στην μεριά του πελάτη.
  - Socket(): Δημιουργεί το τερματικό σημείο σύνδεσης, προσδιορίζει το πρωτόκολλο επικοινωνίας.
  - Connect(): Αίτηση σύνδεσης.
  - Read(): Λήψη Δεδομένων.

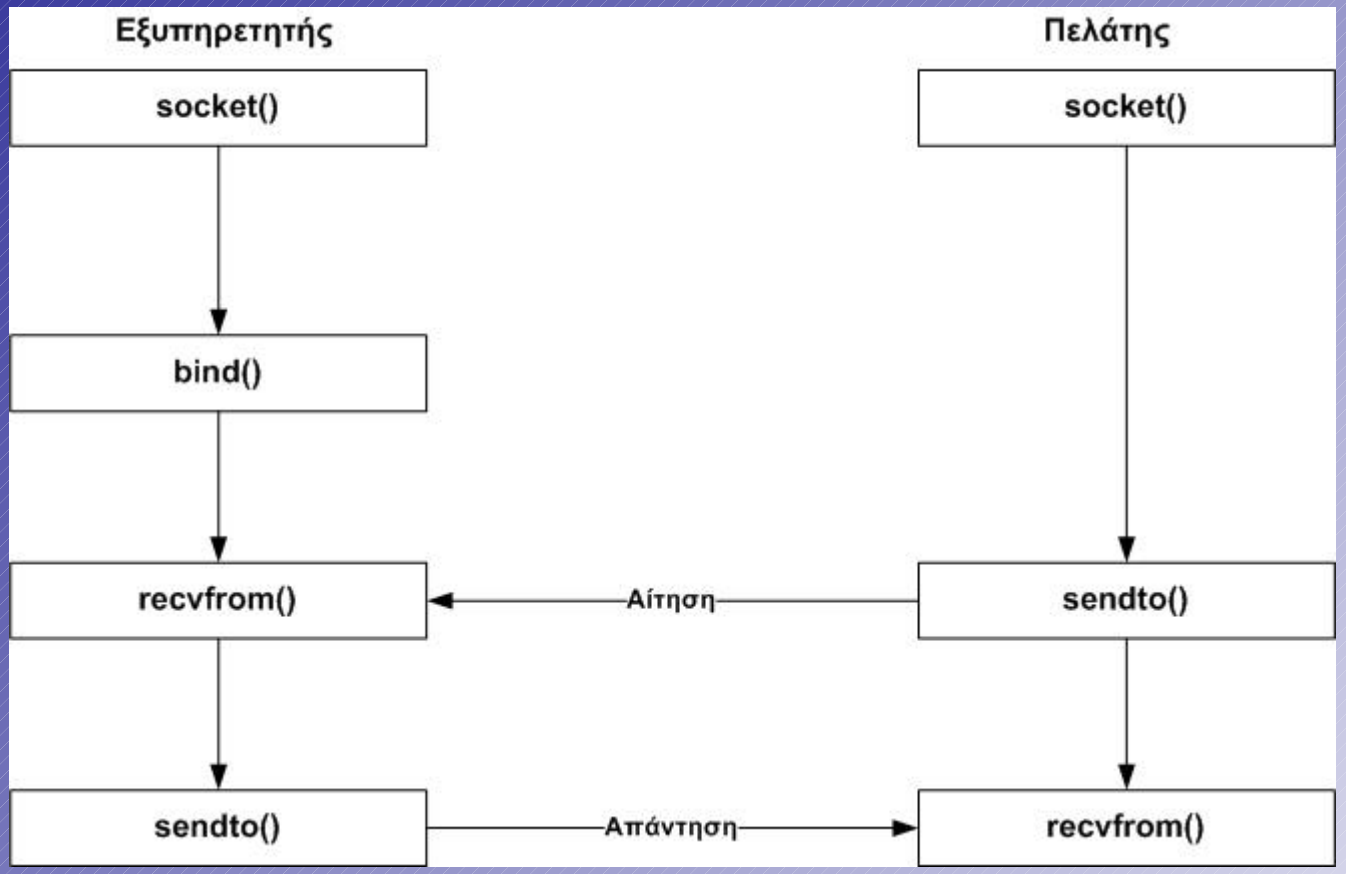
Λειτουργίες των sockets με σύνδεση (2)



Λειτουργίες των sockets χωρίς σύνδεση (1)

- Κλήσεις συστήματος στην μεριά του εξυπηρετητή.
  - Socket(): Δημιουργεί το τερματικό σημείο σύνδεσης, προσδιορίζει το πρωτόκολλο επικοινωνίας.
  - Bind(): καταχωρεί και προσαρτά το socket στήν επιθυμητή θύρα επικοινωνίας.
  - recvfrom(): Λήψη Δεδομένων.
  - sendto(): Αποστολή Δεδομένων.
- Κλήσεις συστήματος στην μεριά του πελάτη.
  - Socket(): Δημιουργεί το τερματικό σημείο σύνδεσης, προσδίορίζει το πρωτόκολλο επικοινωνίας.
  - recvfrom(): Λήψη Δεδομένων.
  - sendto(): Αποστολή Δεδομένων.

Λειτουργίες των sockets χωρίς σύνδεση (2)



Sockets με σύνδεση vs χωρίς σύνδεση

- Με σύνδεση
  - Εύκολα στον προγραμματισμό.
  - Το πρωτόκολλο μεταφοράς (TCP) κάνει την βαριά δουλειά (πχ έλεγχος για λάθη, επανεκπομπές).
  - Απαιτείται ξεχωριστό socket για κάθε σύνδεση.
- Χωρίς σύνδεση
  - Λιγότερα overhead
  - Κανένας περιορισμός σε αριθμό πελατών.

Παράδειγμα Server σε Java (1)

```
/**
 *
 * @author irons
 */

public class Main {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here

        new server();

    }

}
```

Παράδειγμα Server σε Java (2)

```
/**
 *
 * @author irons
 */
import java.io.*;
import java.net.*;

public class server {

    ServerSocket myserver;
    Socket myclient;
    PrintStream send_message;

    public server() {
        myserver = null;
        myclient = null;
        send_message = null;
```

Παράδειγμα Server σε Java (3)

```
try {
            myserver = new ServerSocket(56000);

        } catch (IOException e) {
            System.out.println(e);
        }

        try {
            myclient = myserver.accept();

       System.out.println("Connection from
  "+myclient.getInetAddress().getHostAddress());

            send_message = new PrintStream(myclient.getOutputStream());

            send_message.println("hello");

        } catch (IOException e) {
            System.out.println(e);
        }
```

Παράδειγμα Server σε Java (4)

```
try {
            send_message.close();

            myclient.close();

            myserver.close();

        } catch (IOException e) {

            System.out.println(e);

        }
    }
}
```

Παράδειγμα Client σε Java (1)

```
/**
 *
 * @author irons
 */

public class Main {

    /**
     * @param args the command line arguments
     */

    public static void main(String[] args) {
        // TODO code application logic here
        new Client();
    }

}
```

Παράδειγμα Client σε Java (2)

```
/**
 *
 * @author irons
 */
import java.io.*;
import java.net.*;

public class Client {

    Socket myclient;
    BufferedReader recv_message;
    String message;

    public Client() {
        myclient = null;
        recv_message = null;
        message = null;
```

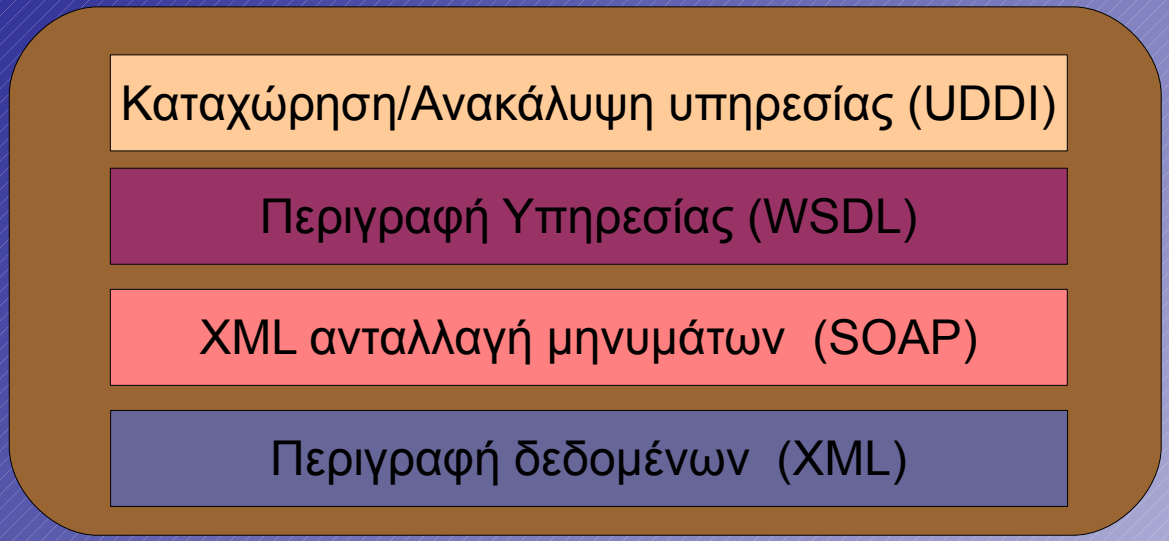
Παράδειγμα Client σε Java (3)

```
try {
            myclient = new Socket("127.0.0.1", 56000);

            recv_message = new BufferedReader(new
   InputStreamReader(myclient.getInputStream()));
            message = recv_message.readLine();

            System.out.println(message);
        } catch (IOException e) {
            System.out.println(e);
        }

        try {
            recv_message.close();
            myclient.close();
        } catch (IOException e) {
            System.out.println(e);
        }

    }
}
```

Web Services

- Παρέχουν:
  - Διαλειτουργικότητα μεταξύ ετερογενών εφαρμογών.
- Επιτρέποντας:
  - Την έκδοση, εντοπισμό και χρήση υπηρεσιών.
- Με τη χρήση:
  - UDDI (Universal Description, Discovery and Integration)
  - WSDL (Web Service Description Language)
  - SOAP (Simple Object Access Protocol)
  - XML (Extra Markup Language)

Web Service Stack



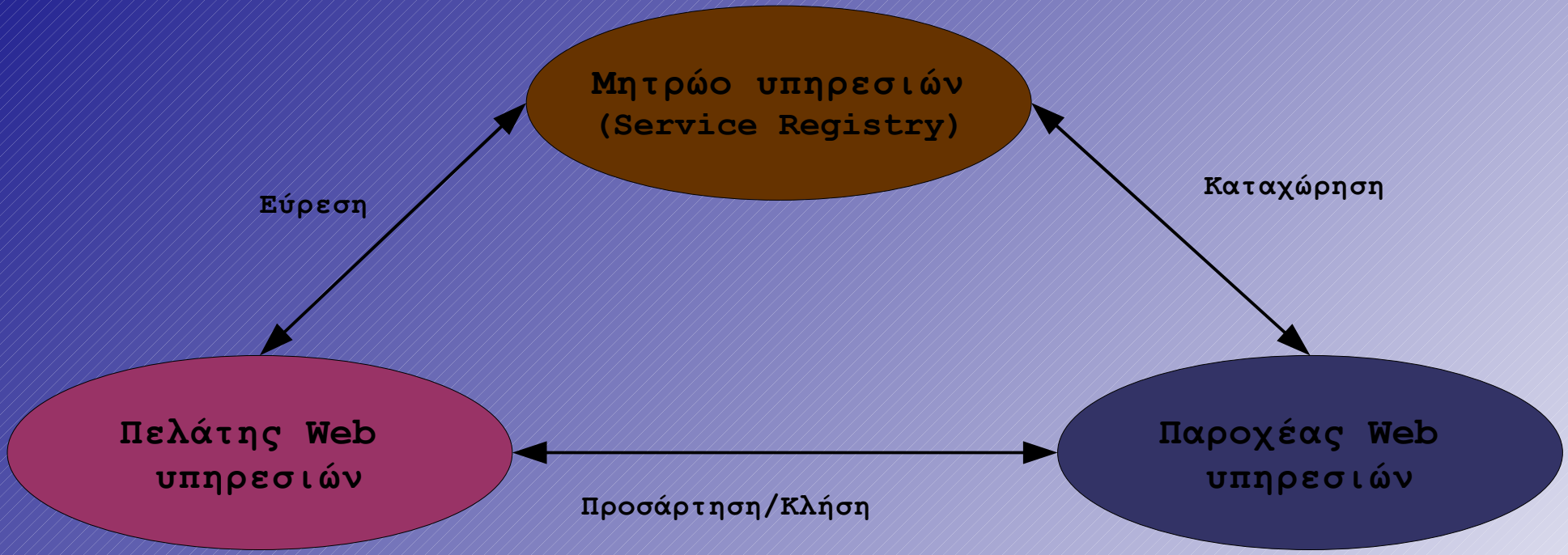
UDDI: Universal Description, Discovery and Integration

WSDL: Web Services Description Language

SOAP: Simple Object Access Protocol

XML: Extra Markup Language

Λειτουργία Web Services



UDDI

- Είναι ένα πρότυπο για μητρώα υπηρεσιών.
- Τριών ειδών ευρετήρια:
  - White pages: Πληροφορίες για τις εταιρίες που παρέχουν τις υπηρεσίες (όνομα, διεύθυνση.....).
  - Yellow pages: Ταξινόμηση υπηρεσιών ανάλογα με την λειτουργία τους.
  - Green pages: Παρέχει πληροφορίες για τη τοποθεσία και τις διεπαφές μιας υπηρεσίας. Συνήθως είναι το WSDL αρχείο που έχει δημιουργηθεί για την web υπηρεσία.
- Η επικοινωνία με το UDDI γίνεται μέσω του SOAP.
- Το UDDI παρέχει τα κατάλληλα εργαλεία για την καταχώρηση και αναζήτηση υπηρεσιών.

WSDL (1)

- Παρέχει:
  - Περιγραφή των Web υπηρεσιών.
    - Διεπαφές, μεθόδους.
  - Εντοπισμό των Web υπηρεσιών.
- Είναι γραμμένο σε XML.
- Βασικά στοιχεία του WSDL εγγράφου:

Στοιχείο	Λειτουργία
<types>	Προσδιορισμός του τύπου των δεδομένων (π.χ. ακέραιος)
<message>	Προσδιορισμός των μηνυμάτων (π.χ. Ορίσματα μιας μεθόδου)
<portType>	Προσδιορισμός των υποστηριζόμενων λειτουργιών (π.χ. διεπαφή μεθόδου)
<binding>	Προσδιορισμός των πρωτοκόλλων επικοινωνίας (π.χ SOAP)

WSDL (2)

- Προσδιορίζονται 4 τρόποι επικοινωνίας:
  - One-way: Λήψη μηνύματος χωρίς αποστολή απάντησης.
  - Request-Response: Λήψη μηνύματος και αποστολή απάντησης.
  - Solicit-Response: Αποστολή μηνύματος και αναμονή για λήψη απάντησης.
  - Notification: Αποστολή μηνύματος χωρίς αναμονή για λήψη απάντησης.

WSDL (3)

```
<definitions>

<message name="getName"> <!-- Ορίζουμε το όνομα του μηνύματος -->
<part name="name" type="xs:string"/> <!-- Ορίζουμε το όνομα της μεταβλητής για το μήνυμα
και τον τύπο της -->
</message>

<message name="getGreeting"> <!-- Ορίζουμε το όνομα του μηνύματος -->

  <part name="greeting" type="xs:string"/> <!-- Ορίζουμε το όνομα της μεταβλητής για το
μήνυμα και τον τύπο της -->

</message>



<portType name="Greeting_srv"> <!-- Ορίζουμε το όνομα της Web υπηρεσίας -->

  <operation name="setGreeting"> <!-- Ορίζουμε το όνομα της μεθόδου που θα επιτελέσει τη
υπηρεσία -->

    <input message="getName"/> <!-- Ορίζουμε την είσοδο στη μέθοδο -->

    <output message="getGreeting"/> <!-- Ορίζουμε την έξοδο από τη μέθοδο -->

  </operation>

</portType>



<binding type="Greeting_srv" name="test"> <!-- Ορίζουμε το ένα όνομα για την προσάρτηση και
αντιστοιχούμε με τη Web υπηρεσία-->
    <soap:binding style="document"
```

SOAP (1)

- Το SOAP επιτρέπει στις εφαρμογές να ανταλλάσσουν μηνύματα πάνω από το HTTP.
  - Το πρωτόκολλο επικοινωνίας για πρόσβαση σε Web υπηρεσίες.
- Χρησιμοποιεί το XML.
- Είναι ανεξάρτητο από γλώσσες προγραμματισμού και πλατφόρμες υλοποίησης.
- Η χρησιμοποίηση του HTTP σαν πρωτόκολλο μεταφοράς επιτρέπει την επικοινωνία ακόμα και μέσω τοίχων προστασίας
  - Γιατί;

SOAP (2)

- Ένα SOAP μήνυμα είναι ένα XML έγγραφο που αποτελείται από τα εξής στοιχεία:
  - Envelope: Προσδιορίζει το XML έγγραφο σαν SOAP μήνυμα.
  - Header: Περιέχει πληροφορίες επικεφαλίδας.
  - Body: Περιέχει πληροφορίες σχετικά με την κλήση και απάντηση της SOAP επικοινωνίας.
  - Fault: Περιέχει τυχόν λάθη και πληροφορίες για την κατάσταση της SOAP επικοινωνίας.

SOAP Envelope

- Το envelope ορίζει ότι το XML έγγραφο είναι SOAP μήνυμα
- Το xmlns:soap ορίζει το envelope σαν SOAP. (Υποχρεωτικό πεδίο)
- Το encodingStyle ορίζει το τύπο των δεδομένων στο SOAP μήνυμα.
- Παράδειγμα:

```
<?xml version="1.0"?>

    <soap:Envelope

    xmlns:soap="http://www.w3.org/2001/12/soap-envelope"

    soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

      ...

     Εδώ εισάγετε το υπόλοιπο μήνυμα

      ...
```

SOAP Header (1)

- Το header περιέχει ειδικές πληροφορίες για την εφαρμογή (π.χ. πιστοποίηση) και είναι προαιρετικό.
- Είναι το πρωτο στοιχείο μετά το envelope.
- Όλα τα στοιχεία μέσα στην επικεφαλίδα πρέπει να είναι με namespace
- Τρεις ιδιότητες για κάθε στοιχείο:
  - mustUnderstand (0|1): Ορίζει εάν ο παραλήπτης πρέπει υποχρεωτικά να επεξεργαστεί το εν λόγω στοιχείο ή όχι.
  - actor: Προσδιορίζει την οντότητα η οποία πρέπει να επεξεργαστεί το εν λόγω στοιχείο.
  - encodingStyle: καθορίζει το τύπο δεδομένων του στοιχείου καθώς και όλων των παιδιών του.

SOAP Header (2)

- Παράδειγμα:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<soap:Header>
<m:Nams xmlns:m="http://www.mysite.gr/employes/"
soap:mustUnderstand="1">2
</m:Nams>
</soap:Header>
</soap:Envelope>
```

SOAP Body

- Περιέχει το SOAP μήνυμα.
- Παράδειγμα αίτησης/απάντησης:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-
envelope"
soap:encodingStyle="http://www.w3.org/2001/12/so
ap-encoding">
<soap:Body>
  <m:GetName
xmlns:m="http://www.mysite.gr/students">
    <m:AM>22</m:AM>
  </m:GetName>
</soap:Body>
</soap:Envelope>
```

```
<?xml version="1.0"?>

<soap:Envelope

xmlns:soap="http://www.w3.org/2001/12/soap-
envelope"

soap:encodingStyle="http://www.w3.org/2001/12/so
ap-encoding">

<soap:Body>

  <m:GetNameResponse
xmlns:m="http://www.mysite.gr/students">

    <m:Name>”Argiris”</m:Name>

  </m:GetNameResponse>

</soap:Body>

</soap:Envelope>
```

SOAP Fault

- Είναι προαιρετικό και εμπεριέχεται σαν παιδί στο SOAP body.
- Υπάρχει ένα τέτοιο στοιχείο ανά μήνυμα.
- Περιέχει τα εξής υποστοιχεία:
  - faultcode: κωδικός σφάλματος.
  - faultstring: σύντομη περιγραφή του σφάλματος.
  - faultactor: ποιος το προκάλεσε.
  - detail: λεπτομερής περιγραφή του σφάλματος.

SOAP κωδικοί λάθους	Περιγραφή
VersionMismatch	Μη έγκυρο namespace στο SOAP Envelope.
MustUnderstand	Ένα στοιχείο του Header με τη ιδιότητα mustUnderstand=1 δεν έγινε κατανοητή.
Client	Μη έγκυρη μορφοποίηση μηνύματος η λανθασμένη περιέχουσα πληροφορία.
Server	Πρόβλημα στον εξυπηρετητή και το μήνυμα δεν επεξεργάστηκε.

SOAP/HTTP

- Παράδειγμα αίτησης απάντησης:

```
POST /InRecords HTTP/1.1

Host: www.mysite.gr

Content-Type: application/soap+xml; charset=utf-8

Content-Length: 221

<?xml version="1.0"?>

<soap:Envelope

xmlns:soap="http://www.w3.org/2001/12/soap-
envelope"

soap:encodingStyle="http://www.w3.org/2001/12/soa
p-encoding">

<soap:Body
xmlns:m="http://www.mysite.gr/students">

  <m:GetName>

    <m:AM>22</m:AM>

  </m:GetName>

</soap:Body>

</soap:Envelope>
```

```
HTTP/1.1 200 OK

Content-Type: application/soap+xml; charset=utf-8

Content-Length: 321

<?xml version="1.0"?>

<soap:Envelope

xmlns:soap="http://www.w3.org/2001/12/soap-
envelope"

soap:encodingStyle="http://www.w3.org/2001/12/soa
p-encoding">

<soap:Body
xmlns:m="http://www.mysite.gr/students">

  <m:GetNameResponse>

    <m:Name>"Argiris"</m:Name>

  </m:GetNameResponse>

</soap:Body>

</soap:Envelope>
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

- "Δίκτυα Υπολογιστών", A.S. Tanenbaum, Εκδόσεις Παπασωτηρίου, 3η έκδοση.
- *"Computer Networking: A Top - Down Approach Featuring the Internet", J.F.Kurose, K.W.Ross,Addison Wesley, 20043, 3rd edition*