ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

ΚΕΝΤΡΟ ΕΠΙΜΟΡΦΩΣΗΣ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗΣ

Sockets Network Programming

Αθ. Ανδρούτσος

Διαδικτυακές Εφαρμογές

- Προγράμματα που **επικοινωνούν μεταξύ τους μέσω δικτύου** χρησιμοποιώντας κλάσεις και μεθόδους του ***java.net*** package
- Το μοντέλο επικοινωνίας είναι γνωστό και ως προγραμματισμός δικτύου (network programming)
- Συνδυάζει την γλώσσα Java και τη σουίτα πρωτοκόλλων TCP/IP

Μοντέλο client-server

- Το μοντέλο πελάτη-εξυπηρετητή χρησιμοποιείται από τις περισσότερες δικτυακές εφαρμογές.
- Για παράδειγμα ένας Web εξυπηρετητής (ή αλλιώς Web Server ή HTTP Server) είναι ένα πρόγραμμα που τρέχει συνέχεια και στέλνει απαντήσεις όταν δέχεται αιτήσεις από web πελάτες (web clients, π.χ. Web Browsers). Η πλευρά του web client είναι αυτή που ξεκινά την επικοινωνία με τον εξυπηρετητή
- Τα ***προγράμματα που επικοινωνούν μέσω δικτύου*** θα πρέπει να χρησιμοποιούν ένα **πρωτόκολλο επικοινωνίας δηλαδή ένα κοινό λεξιλόγιο** για την επικοινωνία.
- Στο παράδειγμα της υπηρεσίας Web το πρωτόκολλο επικοινωνίας είναι το HTTP (Hypertext Transfer Protocol)

Πρωτόκολλα επικοινωνίας

- Η εφαρμογή του πελάτη και του εξυπηρετητή μπορεί να θεωρηθεί ότι επικοινωνούν μέσω ενός πρωτοκόλλου του επιπέδου εφαρμογής (π.χ. HTTP για το Web) αλλά **στην πραγματικότητα πολλά επίπεδα επικοινωνίας (και πολλά πρωτόκολλα) περιλαμβάνονται στο μοντέλο επικοινωνίας** (βλ. TCP/IP).
- Για παράδειγμα οι Web clients και servers επικοινωνούν στο επίπεδο μεταφοράς χρησιμοποιώντας το πρωτόκολλο Transmission Control Protocol (TCP)
- Το TCP με τη σειρά του χρησιμοποιεί το Internet Protocol (IP) και το IP επικοινωνεί με το DLC στα χαμηλότερα επίπεδα

Στρωματοποίηση (1)

- Η τεχνολογία επικοινωνίας υπολογιστών έχει να αντιμετωπίσει ένα πλήθος από προβλήματα που αφορούν διαδικασίες διασύνδεσης, μεθόδους δρομολόγησης, τεχνικές μεταφοράς δεδομένων, μεθοδολογίες διαχείρισης συνδέσεων, τρόπους παρουσίασης των δεδομένων και φυσικά υλοποίηση εφαρμογών και υπηρεσιών.

Στρωματοποίηση (2)

- Η διαδικασία επίλυσης όλων αυτών των προβλημάτων ακολουθεί την τεχνική του διαίρει και βασίλευε, όπου κάθε κλάση ομοειδών λειτουργιών αντιμετωπίζονται ξεχωριστά σε διαφορετικά επίπεδα (στρώματα) επίλυσης της κάθε κλάσης
- Η αρχιτεκτονική αυτή ονομάζεται στρωματοποιημένη αρχιτεκτονική και κάθε στρώμα ή επίπεδο (Layer) είναι υπεύθυνο για την αντιμετώπιση μιας συγκεκριμένης κατηγορίας λειτουργιών του δικτύου

Open Systems Interconnection

- Το πιο γνωστό υπόδειγμα στρωματοποιημένης αρχιτεκτονικής είναι το μοντέλο διασύνδεσης ανοικτών συστημάτων (Open Systems Interconnection -- OSI) του οργανισμού προτυποποίησης ISO (International Standards Organization). Το μοντέλο **ISO/OSI** καθιερώνει επτά (7) επίπεδα επικοινωνίας ανάμεσα σε δύο υπολογιστικά συστήματα

Φυσικό Επίπεδο

- Το φυσικό επίπεδο ορίζει την διεπαφή ανάμεσα στις συσκευές μετάδοσης δεδομένων και το επικοινωνιακό μέσο καθώς και τους ηλεκτρομηχανικούς κανόνες μεταγωγής των bits στη γραμμή επικοινωνίας. Μερικά γνωστά πρότυπα φυσικού επιπέδου είναι η διεπαφή RS-232, τα τμήματα που αφορούν το φυσικό επίπεδο των τεχνολογιών τοπικών δικτύων (Ethernet, IEEE 802, Token Ring, FDDI) καθώς και τα φυσικά τμήματα των PPP, SLIP, ATM, Frame Relay, SONET, ADSL, DWDM, Wi-Fi, κλπ.
- Το φυσικό επίπεδο όπως και το επόμενο επίπεδο σύνδεσης δεδομένων **υλοποιούνται στην κάρτα δικτύου** σε επίπεδο Hardware

Επίπεδο Σύνδεσης Δεδομένων

- Το Επίπεδο Σύνδεσης Δεδομένων προσπαθεί να μετατρέψει ένα εγγενώς μη-αξιόπιστο μηχανισμό μεταφοράς δεδομένων που παρέχεται από το φυσικό επίπεδο σε μία ροή από bits που μεταδίδεται δίχως λάθη και δίχως απώλειες πακέτων
- Προς τα υψηλότερα επίπεδα, το επίπεδο σύνδεσης δεδομένων παρέχει ένα αξιόπιστο τρόπο μεταφοράς των δεδομένων
- Τα πιο γνωστά πρότυπα του επιπέδου σύνδεσης δεδομένων είναι τα πρότυπα HDLC και LLC (Logical Link Control - IEEE 802.2, τροποποιημένη εκδοχή του HDLC για τοπικά δίκτυα). Σε περιπτώσεις λαθών ή απώλειας δεδομένων χρησιμοποιούνται τεχνικές επαναμετάδοσης, όπως τα πρωτόκολλα Stop and Wait, Go back n, Selective Repeat

Επίπεδο Δικτύου

- Το επίπεδο δικτύου αποτελεί κι αυτό μέρος του δικτύου επικοινωνίας
- Η βασική του λειτουργία είναι η δρομολόγηση (routing) των πακέτων προς τον τελικό τους προορισμό
- Επίσης, το επίπεδο δικτύου ασκεί και έλεγχο ροής (flow control) των πακέτων, όχι απ΄ άκρη σ' άκρη, αλλά ανάμεσα στους επικοινωνιακούς κόμβους του δικτύου (hop-by-hop)

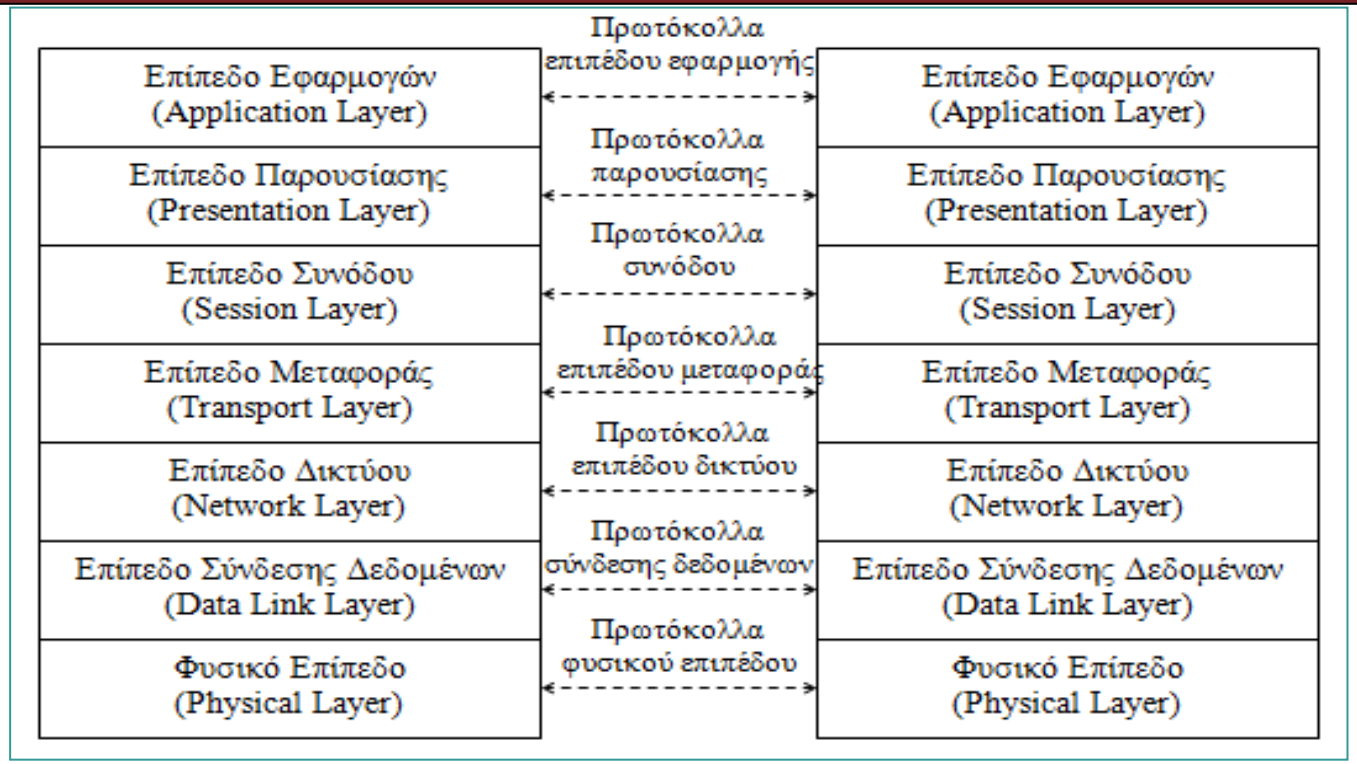
Επίπεδο Μεταφοράς

- Το επίπεδο μεταφοράς, δεν βρίσκεται στους κόμβους του επικοινωνιακού δικτύου, αλλά στους τελικούς υπολογιστές
- Το επίπεδο μεταφοράς δέχεται δεδομένα από το επίπεδο συνόδου , τα διασπά σε μικρότερα πακέτα και τα αποστέλλει στο επίπεδο δικτύου για μεταφορά
- Ανεξάρτητα από το επίπεδο δικτύου, το επίπεδο μεταφοράς εξασφαλίζει την αξιόπιστη μεταφορά των δεδομένων απ’ άκρη σ’ άκρη, ενώ ασκεί και απ’ άκρη σ’ άκρη έλεγχο ροής (end-to-end flow control)

Επίπεδα Συνόδου / Παρουσίασης

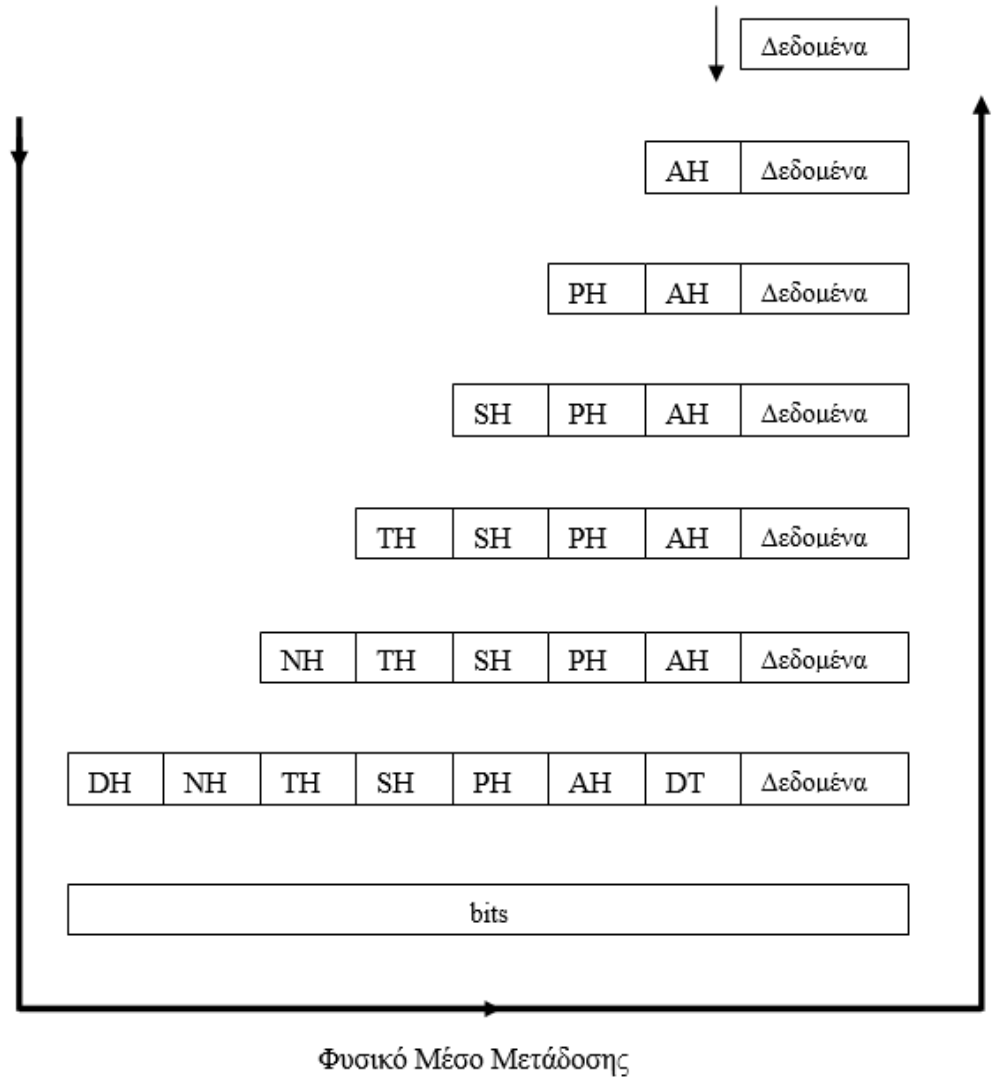
- Το επίπεδο συνόδου ελέγχει την επικοινωνία ανάμεσα στα δύο άκρα, είτε ομαδοποιώντας τα δεδομένα, είτε επανακάμπτοντας από διακοπές της σύνδεσης και γενικά παρέχοντας τον μηχανισμό εγκατάστασης των συνδέσεων. Σε πολλές εφαρμογές υπάρχει μικρή ή καθόλου ανάγκη ύπαρξης του επιπέδου συνόδου.
- Το επίπεδο παρουσίασης ορίζει την μορφή των πληροφοριών που ανταλλάσσονται ανάμεσα στα δύο άκρα και παρέχει υπηρεσίες μετατροπής των δεδομένων (συμπίεση, κρυπτογράφηση) από μια μορφή σε μια άλλη ανάλογα με τις ανάγκες των εφαρμογών

Μηχανισμός Επικοινωνίας (1)



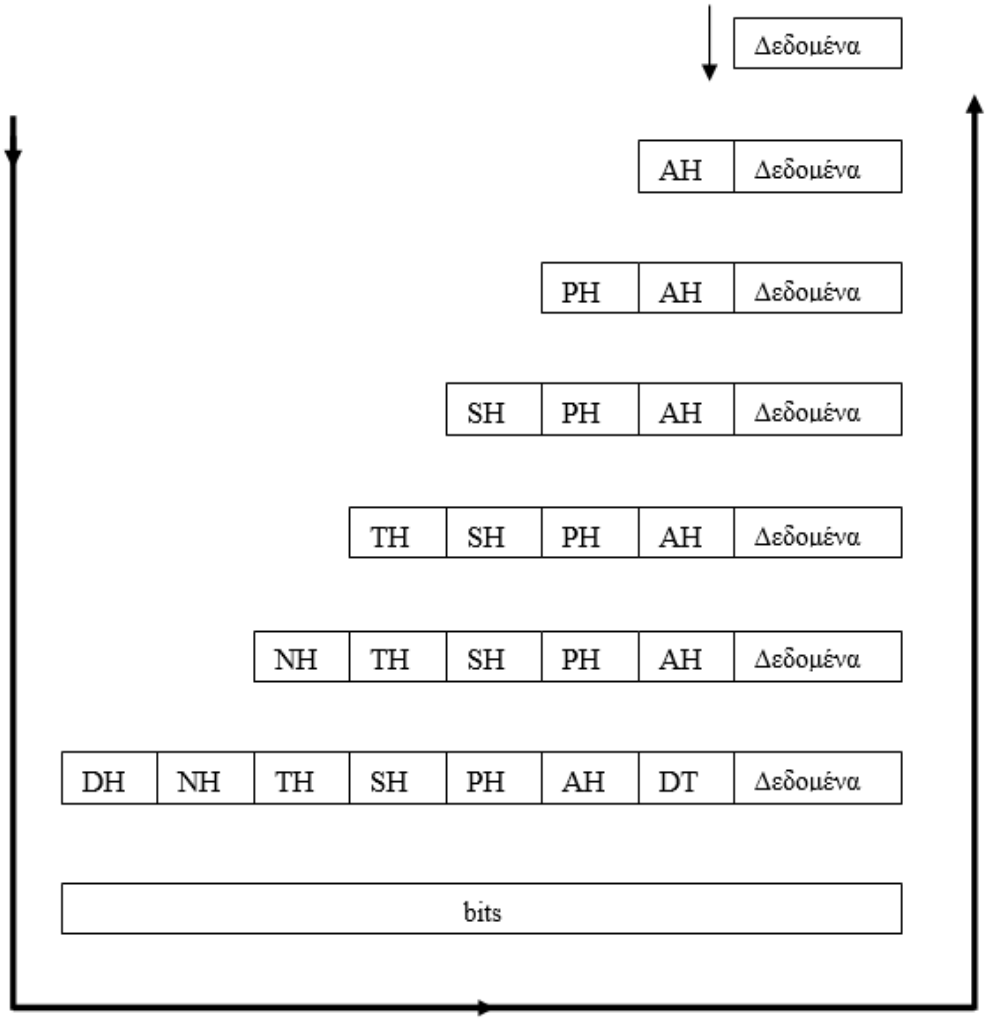
- Το επίπεδο εφαρμογών περιλαμβάνει όλες τις εφαρμογές ή υπηρεσίες στο επίπεδο των χρηστών, όπως υπηρεσίες ηλεκτρονικού ταχυδρομείου, μεταφοράς αρχείων, απομακρυσμένης πρόσβασης, κλπ. Δύο υπολογιστικά συστήματα που επικοινωνούν με βάση το μοντέλο OSI θα πρέπει να διαθέτουν τα επτά αυτά επίπεδα. Ο μηχανισμός της επικοινωνίας μεταξύ των συστημάτων γίνεται με την ανταλλαγή μηνυμάτων και την χρήση πρωτοκόλλων επικοινωνίας

Μηχανισμός Επικοινωνίας (2)



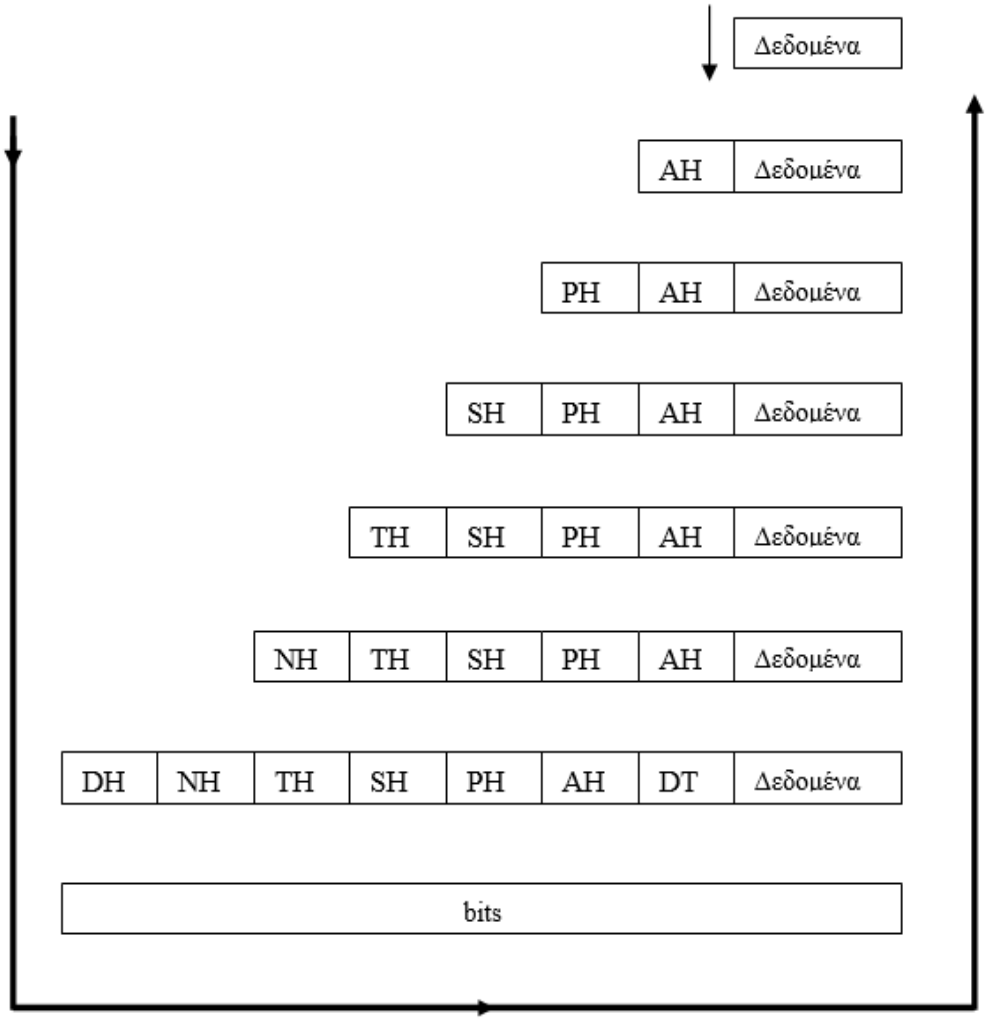
- Ο μηχανισμός της επικοινωνίας ανάμεσα σε δύο υπολογιστικά συστήματα γίνεται με την αποστολή και την λήψη **πακέτων δεδομένων, όπου συμπεριλαμβάνονται και οι επικεφαλίδες (headers)** που επισυνάπτονται από όλες τις διαδοχικές οντότητες
- Οι **επικεφαλίδες περιέχουν πληροφορίες διαχείρισης κάθε επιπέδου και πρωτοκόλλου**, που προσθέτει κάθε οντότητα ώστε να μπορεί να επικοινωνεί με την ομότιμη οντότητα του άλλου άκρου.

Μηχανισμός Επικοινωνίας (3)



- Πιο συγκεκριμένα, στα δεδομένα που εισέρχονται σε μια υπηρεσία του επιπέδου εφαρμογής προστίθεται μία επικεφαλίδα (header) του επιπέδου εφαρμογής, που περιέχει πληροφορίες σχετικές με την εφαρμογή που προσθέτει την επικεφαλίδα
- Στη συνέχεια, τα δεδομένα μαζί με την επικεφαλίδα εφαρμογής (Application Header – AH) μεταβιβάζονται στην επόμενη οντότητα του επιπέδου παρουσίασης, η οποία προσθέτει με τη σειρά της την επικεφαλίδα της (PH).

Μηχανισμός Επικοινωνίας (4)

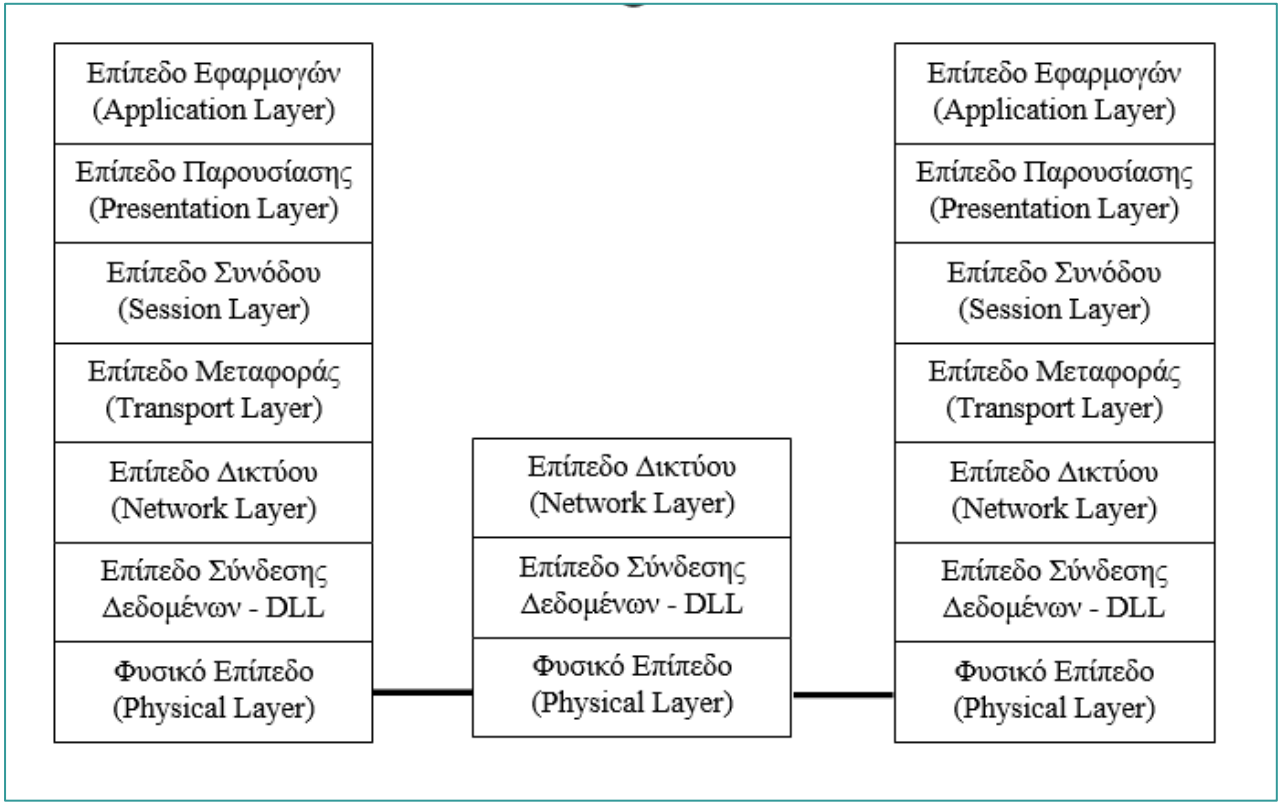


- Η ίδια διαδικασία επαναλαμβάνεται σε κάθε επίπεδο του υπολογιστή αποστολής, όπου με την σειρά τους οι διαδοχικές οντότητες επισυνάπτουν στο τέλος του μηνύματος που λαμβάνουν, την δική τους επικεφαλίδα
- Η οντότητα σύνδεσης δεδομένων εκτός από την επικεφαλίδα προσθέτει στο τέλος και μία ουρά (trailer) που συμβολίζει το τέλος του πακέτου το οποίο και αποστέλλει στο φυσικό επίπεδο.

Μηχανισμός Επικοινωνίας (5)

- Από την άλλη πλευρά ο υπολογιστής προορισμού αποσυναρμολογεί το πακέτο με τον ακριβώς αντίθετο μηχανισμό από αυτόν που συναρμολογήθηκε
- Η οντότητα του επιπέδου σύνδεσης δεδομένων αφαιρεί την επικεφαλίδα (και την ουρά) που είχε προσθέσει η ομότιμη οντότητα του συστήματος αποστολής, διαβάζει τις πληροφορίες που επισυνάπτονται στην επικεφαλίδα και αποστέλλει το πακέτο στην οντότητα του επιπέδου δικτύου
- Το επίπεδο δικτύου με τη σειρά του αφαιρεί την επικεφαλίδα δικτύου και μεταβιβάζει το μήνυμα στο παραπάνω επίπεδο μεταφοράς
- Με αυτό τον μηχανισμό αφαιρούνται διαδοχικά όλες οι επικεφαλίδες μέχρι και το επίπεδο εφαρμογής και τελικά τα δεδομένα φθάνουν στον προορισμό τους

Μηχανισμός Επικοινωνίας (6)



- Ενώ τα τελικά συστήματα για να επικοινωνήσουν πρέπει να έχουν και τα επτά επίπεδα του μοντέλου OSI, οι κόμβοι του δικτύου μεταγωγής πακέτων χρειάζονται μόνο τα τρία κατώτερα επίπεδα γιατί η βασική τους επιδίωξη και λειτουργία είναι η μεταγωγή των πακέτων, η οποία επιτελείται από τα επίπεδα δικτύου, σύνδεσης δεδομένων και το φυσικό επίπεδο.

Μειονεκτήματα ISO / OSI

- Παρότι το υπόδειγμα ISO/OSI είναι ένα πιστοποιημένο πρότυπο (de jure) από τον οργανισμό ISO, στην πράξη δεν χρησιμοποιήθηκε εκτεταμένα από την βιομηχανία υπολογιστών και επικοινωνιών. Ο βασικός λόγος είναι πως **η υλοποίηση των πρωτοκόλλων του μοντέλου OSI απαιτεί υψηλή υπολογιστική δύναμη του συστήματος και η εμπορική υλοποίηση του παραπάνω προτύπου για μικροϋπολογιστές υπήρξε πολύ κοστοβόρα σε όρους χώρων αποθήκευσης, χρόνου εκτέλεσης και γενικά υπολογιστικής ισχύος**
- Με την έλευση του Internet η παγκόσμια βιομηχανία τηλεπικοινωνιών υιοθέτησε αρχικά, εισήγαγε και χρησιμοποίησε στην πράξη (de facto) μια εναλλακτική αρχιτεκτονική προσέγγιση για τη διασύνδεση ανοικτών συστημάτων βασισμένη σε τέσσερα επίπεδα διασύνδεσης αντί για τα επτά επίπεδα του OSI.

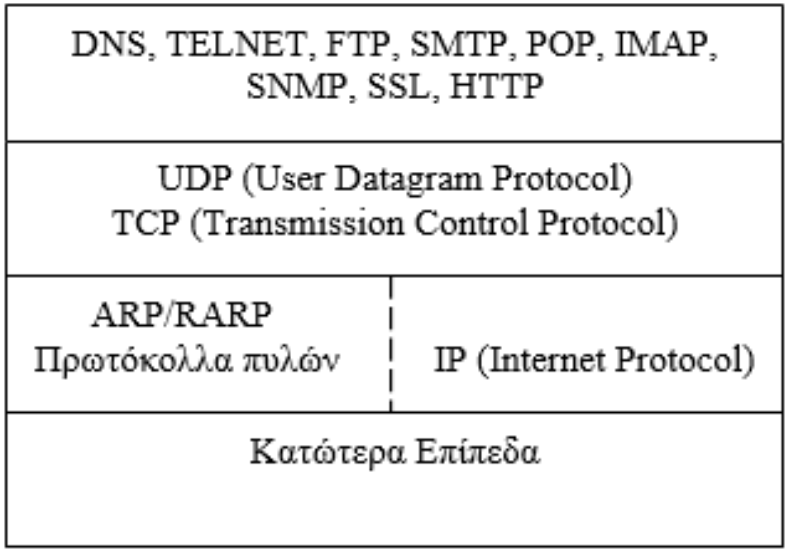
Επίπεδα αρχιτεκτονικής TCP/IP

- Η αρχιτεκτονική αυτή προσέγγιση των τεσσάρων επιπέδων περιλαμβάνει ένα σύνολο πρωτοκόλλων γνωστό ως οικογένεια πρωτοκόλλων TCP/IP (TCP/IP suite) και έχει λάβει την ονομασία της από τα δύο πιο σημαντικά πρωτόκολλα της οικογένειας
- Τα πρωτόκολλα **TCP (Transmission Control Protocol)** του επιπέδου 4 (μεταφοράς) και **IP (Internet Protocol)** του επιπέδου τρία (επίπεδο δικτύου)
- Η αναφορά αρίθμησης γίνεται με βάση το ISO / OSI

TCP/IP (1)

- Το μοντέλο TCP/IP καθιερώνει τέσσερα επίπεδα διασύνδεσης
- Η λειτουργία των πρωτοκόλλων των τεσσάρων επιπέδων είναι παρόμοια με την λειτουργία των πρωτοκόλλων του μοντέλου OSI, η υλοποίηση όμως είναι διαφορετική

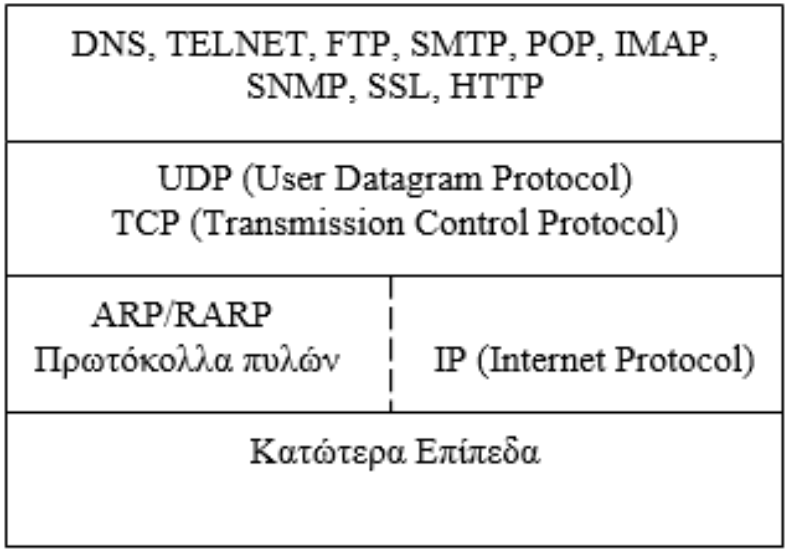
TCP/IP (2)



Τα κατώτερα επίπεδα περιλαμβάνουν τα επίπεδα DLC και φυσικό επίπεδο του μοντέλου OSI και οι προδιαγραφές τους δεν ορίζονται από το μοντέλο TCP/IP. Στο επίπεδο αυτό περιλαμβάνονται όλες οι τεχνολογίες διασύνδεσης των επιπέδων ένα και δύο

- Το **επίπεδο δικτύου** περιλαμβάνει το κύριο πρωτόκολλο δρομολόγησης **IP (IPv4, IPv6)** καθώς και συνοδευτικά πρωτόκολλα όπως τα ICMP (Internet Control Messaging Protocol), ARP (Address Resolution Protocol) , RARP (Reverse Address Resolution Protocol), και τα πρωτόκολλα πυλών (gateway protocols) BGP, IGMP, OSF

TCP/IP (3)

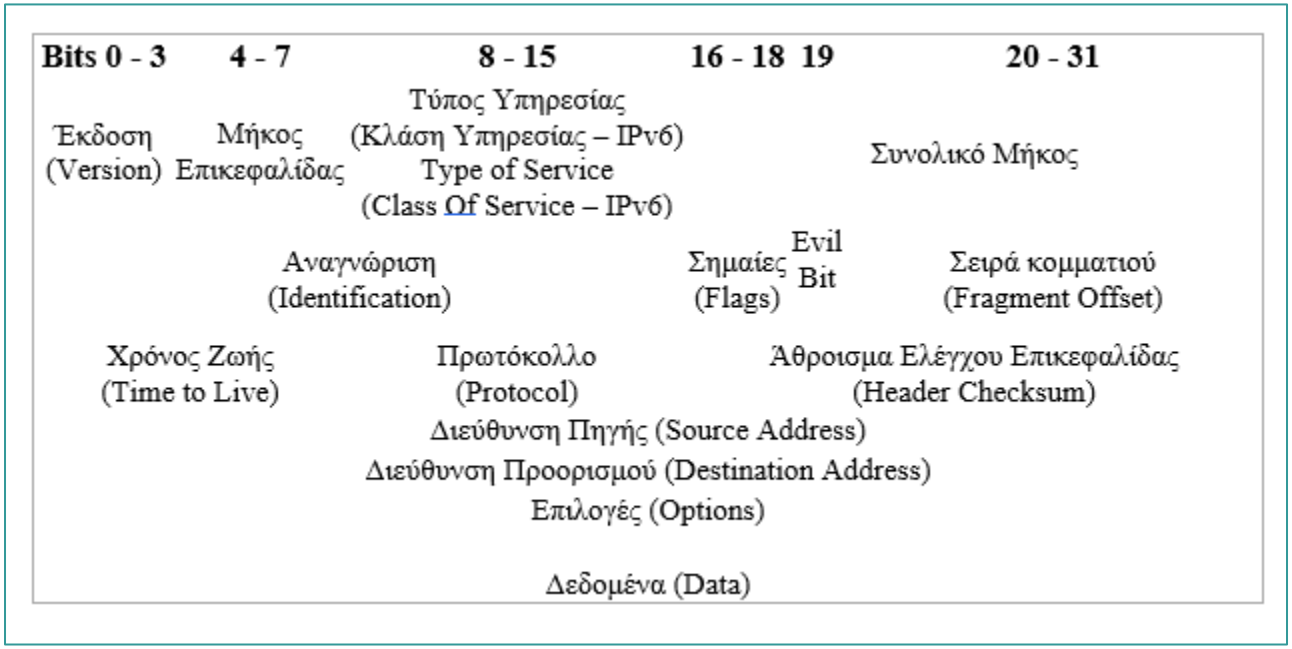


Το **επίπεδο μεταφοράς** περιλαμβάνει δύο κύρια πρωτόκολλα μεταφοράς, τα **UDP (User Datagram Protocol)** και **TCP (Transmission Control Protocol)**

Το **επίπεδο εφαρμογής** περιλαμβάνει όλες τις γνωστές εφαρμογές του Internet

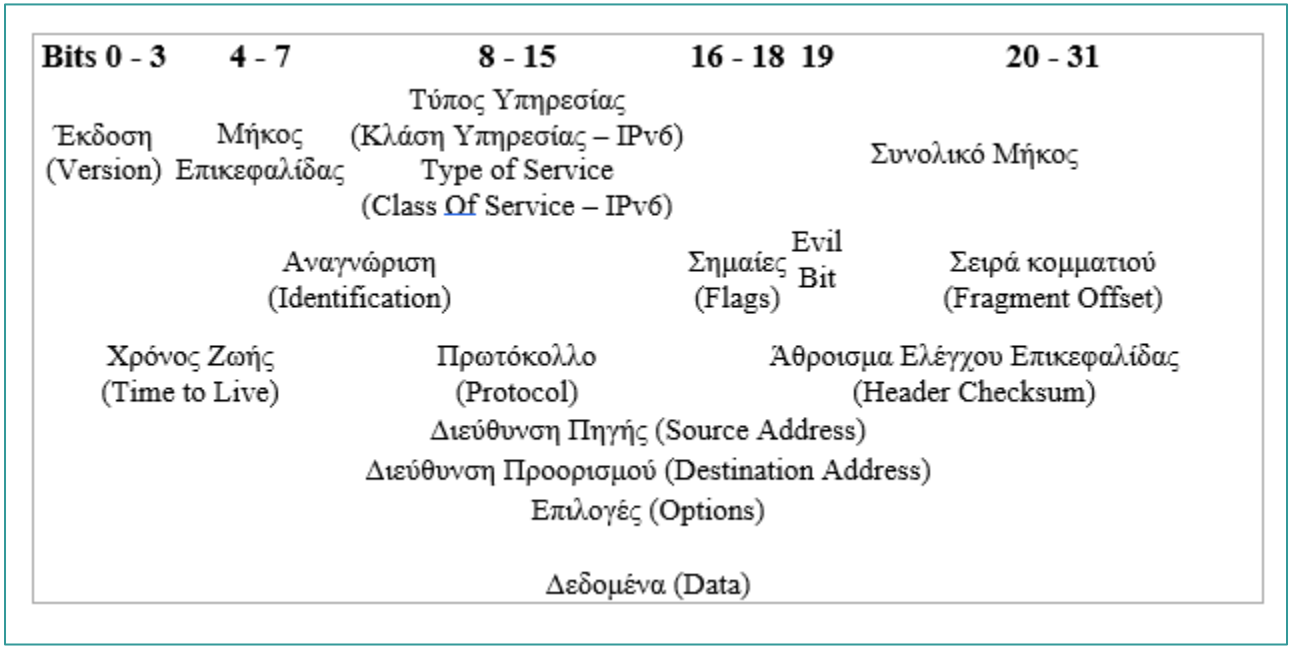
- Οι πιο γνωστές εφαρμογές του επιπέδου 7 είναι *DNS* (Domain Name Server), *FTP* (File Transfer Protocol), *SMTP* (Simple Mail Transfer Protocol), *POP* (Post Office Protocol), *IMAP*, TELNET, *SNMP* (Simple Network Management Protocol), *SSL* (Secure Socket Layer), και **HTTP (Hypertext Transfer Protocol)**

Δομή Πακέτου IP (1)



- Στο επίπεδο δικτύου η βασική δομή πληροφορίας που χρησιμοποιείται είναι το IP Datagram ή πακέτο IP (IP packet). Το κάθε πακέτο IP έχει μέγιστο μέγεθος 64KB και αποτελείται από μία επικεφαλίδα (IP header) που περιέχει πληροφορίες δρομολόγησης και διαχείρισης του επιπέδου δικτύου και μια περιοχή δεδομένων (payload)

Δομή Πακέτου IP (2)

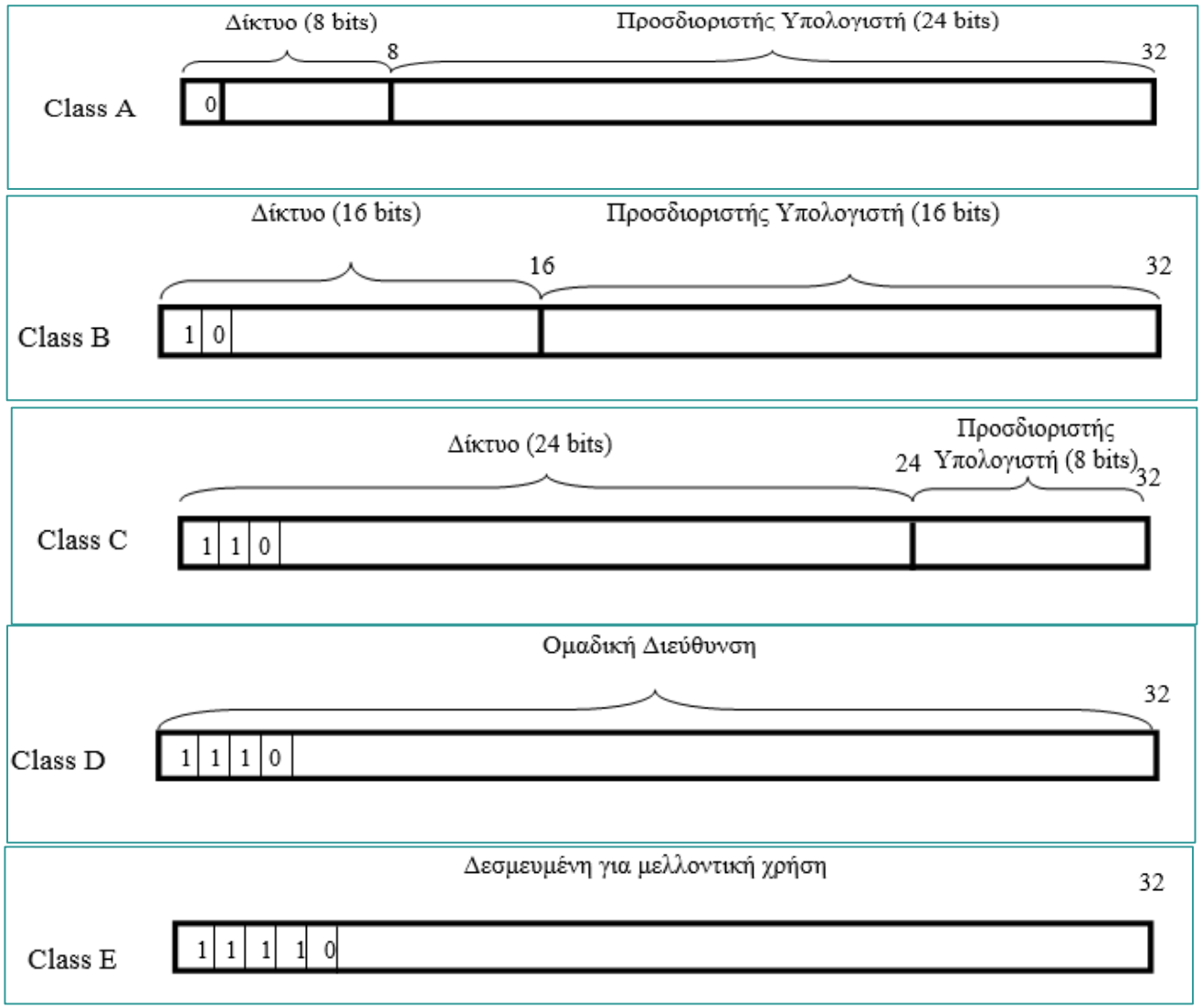


- Κατά τη διαδικασία της δρομολόγησης των πακέτων IP, η κύρια διαδικασία είναι η εύρεση διαδρομών από τον αποστολέα προς τον παραλήπτη κάθε IP πακέτου. Η κύρια διαχειριστική πληροφορία που πρέπει να περιέχει κάθε πακέτο κατά τη διαδικασία της δρομολόγησης είναι οι IP διευθύνσεις (IP address) αποστολής και προορισμού.

IP Διευθύνσεις (1)

- Κάθε IP διεύθυνση αποτελείται από τέσσερα bytes και συνεπώς παριστάνεται σε μορφή τεσσάρων αριθμών διαχωριζομένων με τελεία, που ο κάθε αριθμός μπορεί να λάβει τιμές από 0 έως 255, όσο δηλαδή και η αρίθμηση του κάθε byte
- Οι IP διευθύνσεις είναι διευθύνσεις τεσσάρων (4) bytes (32-bit) που αποτελούνται από δύο μέρη: τον προσδιοριστή του δικτύου και τον προσδιοριστή του υπολογιστή.
- Διακρίνουμε πέντε κατηγορίες (κλάσεις) IP διευθύνσεων, ανάλογα με το πόσα bits καταχωρούνται στο κομμάτι του προσδιοριστή δικτύου και πόσα στο κομμάτι προσδιοριστή του υπολογιστή

IP Διευθύνσεις (2)



- Για παράδειγμα, στην κλάση C το τρία πρώτα bits είναι πάντα ένα (1), ένα (1) και μηδέν (0) με αποτέλεσμα το εύρος των IP διευθύνσεων του πρώτου byte στην κλάση C να είναι από **192 έως 223**
- Δηλαδή μια έγκυρη διεύθυνση κλάσης C είναι η 195.251.248.120

Μάσκα Δικτύου (1)

- Κάθε υπολογιστής δικτύου όταν ξεκινάει, φορτώνει στη μνήμη του την IP διεύθυνσή του. Πως όμως μπορεί να γνωρίζει τις διευθύνσεις δικτύου του, σε ποια δίκτυα δηλαδή ανήκει;
- Ο διαχωρισμός της διεύθυνσης δικτύου από την διεύθυνση υπολογιστή γίνεται με τη χρήση μιας διεύθυνσης 32-bits, γνωστής ως μάσκα δικτύου (network mask) στην οποία όλα τα bits που αντιστοιχούν στη διεύθυνση δικτύου τίθεται ένα (1) και τα bits του υπολογιστή μηδέν (0).

Μάσκα Δικτύου (2)

- Ο κάθε υπολογιστής φορτώνει μαζί με την IP διεύθυνση και την μάσκα του και στη συνέχεια με μία απλή πράξη λογικού ΚΑΙ (AND) μεταξύ της διεύθυνσης και της μάσκας μπορεί να βρει την διεύθυνση δικτύου του.
- Για παράδειγμα μία class A διεύθυνση έχει μάσκα 255.0.0.0, μία class B διεύθυνση έχει μάσκα 255.255.0.0, και οι class C διευθύνσεις έχουν μάσκα 255.255.255.0.
- Κάθε υπολογιστής μπορεί να ανήκει σε περισσότερα από ένα δίκτυα και επομένως να έχει περισσότερες από μία IP διευθύνσεις

Ειδικές Διευθύνσεις IP

- Οι διευθύνσεις που έχουν **όλα τα bits στον προσδιοριστή του υπολογιστή μηδέν** συμβολίζουν διευθύνσεις δικτύου, ενώ αν έχουν όλα τα bits του υπολογιστή ένα συμβολίζουν ομαδικές διευθύνσεις (broadcast) του συγκεκριμένου δικτύου
- Οι διευθύνσεις δικτύου **127.0.0.1 (localhost)**, ονομάζονται loopback και δεν αποτελούν έγκυρες διευθύνσεις δικτύου αλλά χρησιμοποιούνται για τον έλεγχο της διεπαφής του δικτύου

Static vs dynamic

- Επίσης, οι διευθύνσεις IP μπορούν να είναι στατικές (static IP) δηλαδή να έχουν εκχωρηθεί μόνιμα σε ένα υπολογιστή από τον διαχειριστή του δικτύου ή δυναμικές (dynamic IP) δηλαδή να αλλάζουν κάθε φορά που ο υπολογιστής συνδέεται στο δίκτυο μέσω ενός παρόχου υπηρεσιών διασύνδεσης.
- Οι IP διευθύνσεις, όπως έχουμε αναφέρει, παίζουν πρωταρχικό ρόλο στη λειτουργία της δρομολόγησης

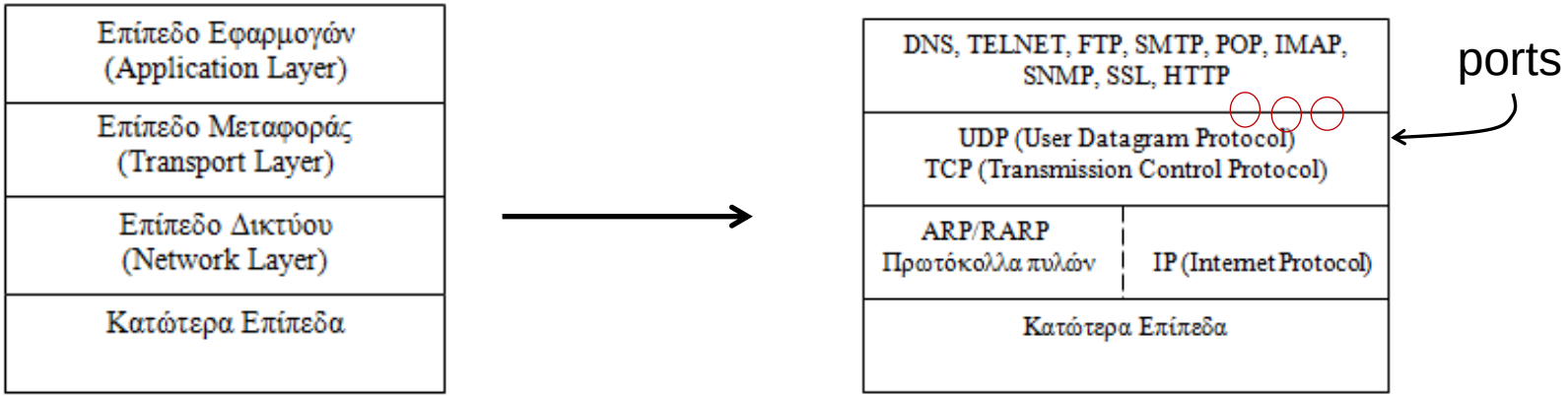
Λογικές vs Φυσικές Διευθύνσεις

- Οι διευθύνσεις IP είναι διευθύνσεις λογικές, ενώ οι διευθύνσεις του επιπέδου DLC είναι διευθύνσεις φυσικές
- Σε κάθε βήμα της δρομολόγησης οι IP διευθύνσεις του επόμενου βήματος πρέπει να αντιστοιχίζονται σε φυσικές διευθύνσεις.
- Η αντιστοίχιση IP διευθύνσεων σε φυσικές διευθύνσεις και το αντίστροφο είναι έργο των πρωτοκόλλων ARP / RARP

Πίνακες Δρομολόγησης

- Επίσης, η συμπλήρωση και διαχείριση του πίνακα δρομολόγησης στους δρομολογητές (routers) routers είναι έργο των πρωτοκόλλων πύλης, όπως τα BGP, OSPF.
- Το IPv6 (ή IPng) είναι μία νέα έκδοση του πρωτοκόλλου IP η οποία σχεδιάστηκε ως εξέλιξη του βασικού πρωτοκόλλου των διαδικτύων IPv4. Η παρουσίαση του έγινε από την IETF (Internet Engineering Task Force) το 1994 και περιγράφεται αναλυτικά σε μια σειρά από RFCs. Η πιο σημαντική αλλαγή του IPv6 ως προς το IPv4 είναι η αύξηση του μεγέθους της διεύθυνσης IP από 32 bit σε 128 bit

Sockets



- Ανάμεσα στα επίπεδα 4 και 7 (Μεταφοράς και Εφαρμογών) μπορούμε να θεωρήσουμε πως υπάρχουν κάποια σημεία επικοινωνίας (**θύρες / ports**) που επιτρέπουν την επικοινωνία ανάμεσα στα δύο επίπεδα
- **Socket** είναι μία τριάδα: ***IP διεύθυνση, port και πρωτόκολλο επιπέδου μεταφοράς (για παράδειγμα localhost, 8080, TCP)*** *που προσδιορίζει ένα σημείο επικοινωνίας (endpoint) ανάμεσα σε δύο απομακρυσμένες εφαρμογές*

TCP / UDP

- Τα δύο πρωτόκολλα του επιπέδου μεταφοράς που επικοινωνούν με το IP και παρέχουν τη διεπαφή ανάμεσα το ΛΣ και τον χρήστη είναι τα TCP και UDP
- Το TCP είναι αξιόπιστο προσανατολισμένο σε σύνδεση πρωτόκολλο που υλοποιεί έλεγχο ροής απ’ άκρη σ άκρη
- Το User Datagram Protocol (UDP) είναι μη αξιόπιστο πρωτόκολλο που υλοποιεί την τεχνική του best effort. Αν ένα πακέτο χαθεί στη διαδρομή δεν υπάρχει μηχανισμός επαναποστολής

Well-Known ports

Protocol	Port	Protocol
echo	7	TCP/UDP
discard	9	TCP/UDP
daytime	13	TCP/UDP
FTP data	20	TCP
FTP	21	TCP
SSH	22	TCP
telnet	23	TCP
smtp	25	TCP
time	37	TCP/UDP
whois	43	TCP
finger	79	TCP
HTTP	80	TCP

- Οι υπηρεσίες που παρέχονται μέσω του Internet χρησιμοποιούν well-known ports δηλαδή συγκεκριμένα πρωτόκολλα και θύρες επικοινωνίας, όπως φαίνεται στον αριστερά πίνακα
- Υπάρχουν και άλλα well-known ports καθώς επίσης και εφήμερα ports που εκχωρούνται σε εφαρμογές που τα χρησιμοποιούν περιστασιακά (όπως για παράδειγμα σε εφαρμογές πελάτες)

Sockets Programming

- Θα δούμε στη συνέχεια πως υλοποιούμε διαδικτυακές εφαρμογές με τη χρήση sockets που είναι ο βασικός χαμηλού επιπέδου τρόπος υλοποίησης διαδικτυακών εφαρμογών πάνω από τον οποίο έχουν δημιουργηθεί διάφορα APIs όπως το java.net

Παράδειγμα

- Υπηρεσία → TCP Day-Time
- Υλοποίηση πελάτη (client)
- Ο πελάτης εγκαθιδρύει μία σύνδεση TCP (TCP connection) με τον time-server και ο εξυπηρετητής απλά απαντά στέλνοντας πίσω την τρέχουσα ώρα και ημερομηνία σε human-readable μορφή δεδομένου ότι η ημερομηνία στα συστήματα UNIX ξεκινά από το UNIX epoch (1/1/1970) και αποστέλλεται ως διαφορά της τρέχουσας ημερομηνία από το 1/1/1970

Day-time TCP client


```
9  public class DayTimeCli {
10
11     public static void main(String[] args)  {
12         StringBuilder sb = new StringBuilder();
13         Socket socket = null;
14         try {
15             //InetAddress servAddress = InetAddress.getByName("time.nist.gov");
16             InetAddress servAddress = InetAddress.getByName("127.0.0.1");
17             int servPort = 13;
18             socket = new Socket(servAddress, servPort);
19             BufferedReader bf = new BufferedReader(new InputStreamReader(socket.getInputStream()));
20             String line = "";
21             while ((line = bf.readLine()) != null) sb.append(line).append("n");
22             System.out.println("Daytime server says it is: " + sb.toString());
23         } catch (IOException e) {
24             e.printStackTrace();
25         } finally {
26             try {
27                 if (socket != null) socket.close();
28             } catch (IOException e) {
29                 e.printStackTrace();
30             }
31         }
32     }
33 }
```

Δημιουργία dynamic (ephemeral, το port αποδίδεται αυτόματα από το ΛΣ) TCP socket στον client και ταυτόχρονα connect στον Server (γρ. 18)

Διάβασμα από το socket και προσθήκη στον StringBuilder

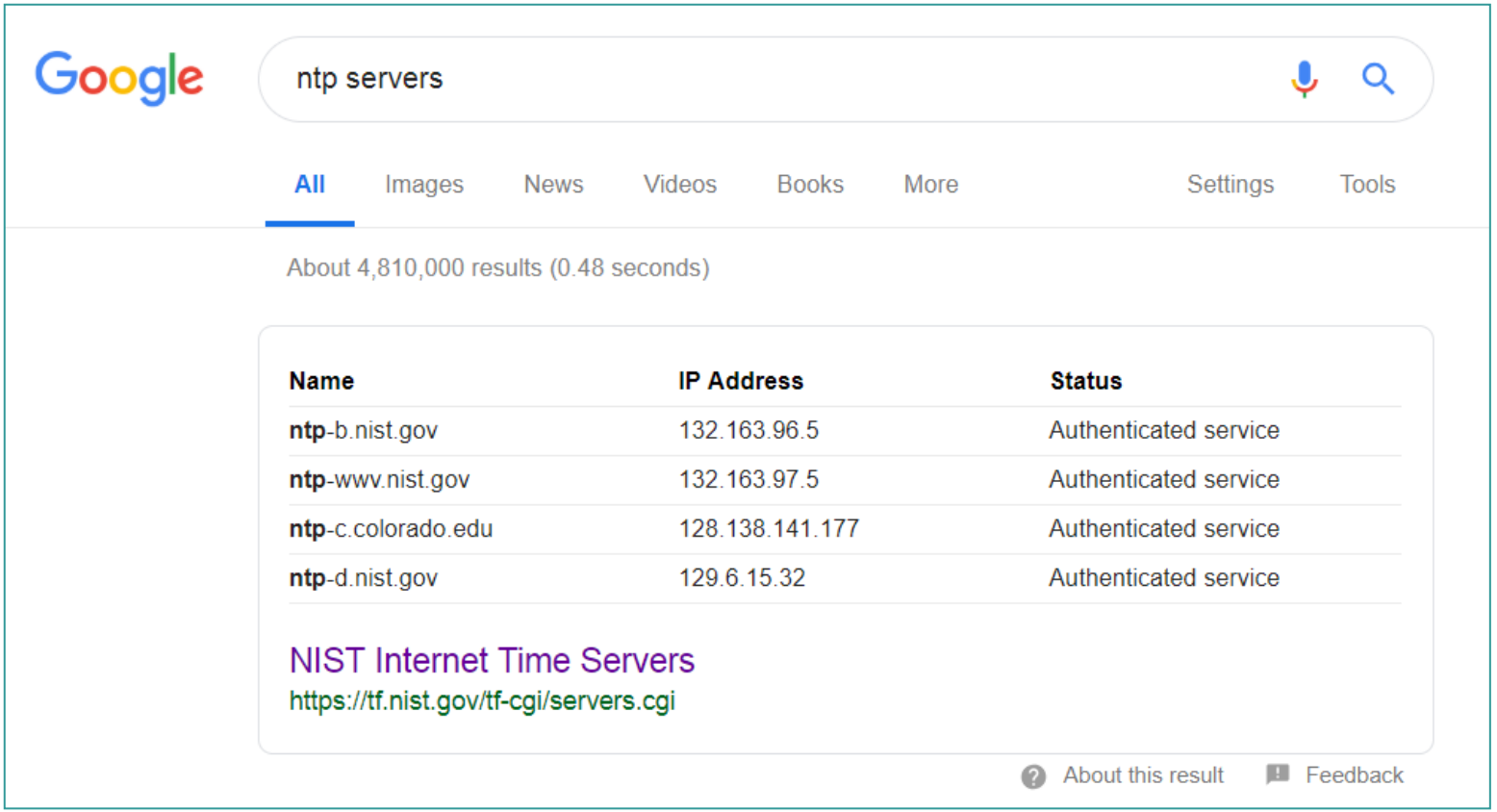
Συσχέτιση του InputStream του Socket με BufferReader

Day-time TCP client (2)

- Η κλάση Socket και ο constructor δημιουργούν ένα δυναμικό (εφήμερο, ορίζεται από το ΛΣ) TCP (stream) socket στην πλευρά του client.
- Όταν δημιουργείται το socket κάνουμε ταυτόχρονα connect στον Server (μπορεί να γίνει και σε δύο βήματα – 1. Socket() και στη συνέχεια connect())
- Σχετίζουμε το input stream του socket με ένα buffered reader, ώστε να μετατρέψουμε bytes σε χαρακτήρες
- Στη συνέχεια διαβάζουμε (read) μέσα σε μια while (όσο υπάρχουν δεδομένα) από το socket και εμφανίζουμε στο stdout (οθόνη)

NTP Servers

- **NTP**: Network Time Protocol - Δεν είναι το ίδιο με το **day-time protocol**
- **NIST**: National Institute of Standards and Technology
- Οι παρακάτω τρεις NIST servers (όχι του Colorado) υλοποιούν και το day-time protocol



While loop

- Επειδή δεν μπορούμε πάντα να υποθέτουμε ότι η απάντηση του server είναι τέτοια ώστε να μπορεί να διαβαστεί με ένα read γιαυτό όταν διαβάζουμε από ένα TCP socket, πρέπει πάντα να διαβάζουμε σε loops και να τερματίζει το loop είτε όταν το read επιστρέφει 0/null (i.e., ο server έκλεισε την σύνδεση) ή δημιουργηθεί IOException οπότε έχει συμβεί κάποιο λάθος
- Στην περίπτωσή μας το τέλος του while συμβαίνει όταν ο server κλείσει τη σύνδεση

Iterative daytime TCP server (1)

- Στη συνέχεια θα υλοποιήσουμε ένα δικό μας (τοπικό) εξυπηρετητή day-time
- Ο server μας θα είναι **TCP server** και θα είναι ***επαναληπτικός (iterative)***, δηλ. θα εξυπηρετεί τις αιτήσεις των clients **μία αίτηση τη φορά** (επομένως αν υπάρχουν πολλές αιτήσεις μία θα εξυπηρετείται και οι άλλες θα περιμένουν)

Iterative daytime TCP server (2)

- Οι Servers τρέχουν συνέχεια στο παρασκήνιο είτε ως processes ή ως threads
- Για να υλοποιήσουμε Threads μπορούμε να κάνουμε extend την κλάση Thread ή το Interface Runnable

Iterative daytime TCP server (3)


```
11  public class IterativeDayTimeServer extends Thread {
12
13      @Override
14      public void run() {
15          ServerSocket servFd;
16          final int serverPort = 13;
17
18          try {
19              servFd = new ServerSocket();
20              servFd.bind(new InetSocketAddress("127.0.0.1", serverPort), 10);
21
```

- Δημιουργούμε ένα ServerSocket. Το κάνουμε bind σε μία διεύθυνση (η 127.0.0.1 είναι διεύθυνση loopback δηλ. του τοπικού μας υπολογιστή) και σε ένα port (13 είναι το well-known port για το day-time service).

Iterative daytime TCP server (4)


```
18          try {
19              servFd = new ServerSocket();
20              servFd.bind(new InetSocketAddress("127.0.0.1", serverPort), 10);
21
22              while (true) {
23                  Socket connectedFd = servFd.accept();
24                  BufferedWriter bw = new BufferedWriter(
25                          new OutputStreamWriter(connectedFd.getOutputStream()));
26                  bw.write(new Date().toString());
27                  bw.flush();
28                  connectedFd.close();
29              }
30          } catch (IOException e) {
31              e.printStackTrace();
32          }
33      }
34  }
```

Η ***accept()*** δέχεται μία μόνο σύνδεση και δημιουργεί ένα **νέο Socket με τις ίδιες ιδιότητες** με το servfd που εξυπηρετεί τον client

- Όταν γίνει close() το **connectedSock** τότε η **accept()** μπορεί και δέχεται και εξυπηρετεί την επόμενη αίτηση στην ουρά. Μπορούμε να ορίσουμε 10 requests στην ουρά. Πάνω από 10 γίνονται drop.

Iterative daytime TCP server (5)


```
1   package gr.aueb.cf.testbed.ch26;
2
3   public class IterativeDayTimeServerApp {
4
5       public static void main(String[] args) {
6           IterativeDayTimeServer dayTimeServer = new IterativeDayTimeServer();
7           dayTimeServer.start();
8       }
9   }
10
```

Iterative daytime TCP server (5)

- Ο server περιμένει το client connection
- Όταν γίνει η σύνδεση **δημιουργείται ένας νέος περιγραφητής της σύνδεσης** (connectedSocket) που είναι ο connected descriptor
- Χρησιμοποιείται για την επικοινωνία με τον client
- Τα συνηθισμένα στυλ ατέρμονου loop που χρησιμοποιούνται είναι το while(true) { . . . } ή το for(;;) {. . .}

Iterative daytime TCP server (6)

- Ο server **κλείνει τη σύνδεση με την close()**
- Ο server **χειρίζεται ένα client τη φορά**
- Αν υπάρχουν περισσότερες αιτήσεις ταυτόχρονα ο πυρήνας τις βάζει στην ουρά του ServerSocket
- Το **default όριο είναι το 50**, μετά το οποίο όποιο αίτημα σύνδεσης έρχεται απορρίπτεται (ConnectionException στην πλευρά του client)

Iterative daytime TCP server (7)

- *Μπορούμε να θέσουμε εμείς όριο αιτήσεων στην ουρά* θέτοντας μία ακόμα παράμετρο στην bind (π.χ. 3, οπότε το max queue length είναι 3)

```
20          servfd.bind(new InetSocketAddress("127.0.0.1", serverPort), 3);
```

- Στην εφαρμογή μας θεωρούμε πως η συχνότητα requests είναι μικρή
- Αν ήταν πιο γρήγορη η συχνότητα αιτημάτων από του clients θα χρειαζόμασταν concurrent server

Iterative vs concurrent

- Ο προηγούμενος server μας ήταν iterative server και εξυπηρετεί έναν πελάτη τη φορά
- Για **ταυτόχρονη εξυπηρέτηση πολλών πελατών** χρειαζόμαστε concurrent servers (ταυτόχρονους εξυπηρετητές) που εξυπηρετούν ταυτόχρονα πολλαπλά αιτήματα

Concurrent Server (1)


```
9   public class ConcurrentDayTimeServer implements Runnable {
10      private final Socket sockFd;
11
12      public ConcurrentDayTimeServer(Socket sockFd) {
13          this.sockFd = sockFd;
14      }
15
16      @Override
17      public void run() {
18          try (BufferedWriter bf = new BufferedWriter(
19                  new OutputStreamWriter(sockFd.getOutputStream()));) {
20              bf.write(new Date().toString());
21              bf.flush();
22          } catch (IOException e) {
23              e.printStackTrace();
24          }
25      }
26  }
```

- Η βασική ιδέα είναι ότι ***ο Server δημιουργεί αντίγραφο του εαυτού του κάθε φορά που δέχεται και εξυπηρετεί μία νέα σύνδεση***

- Η run() υλοποιεί τη λογική του τη λογική του thread

Concurrent Server (2)


```java
public class ConcurrentServerApp {
    private static final int port = 13;

    public static void main(String[] args) {
        try (var servFd = new ServerSocket();) {
            servFd.bind(new InetSocketAddress("127.0.0.1", port));
            System.out.println("Server started ...");

            while (true) {
                Socket connectedFd = servFd.accept();
                Thread socketThread = new Thread(new ConcurrentDayTimeServer(connectedFd));
                socketThread.start();
            }

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

- Η main() δημιουργεί το Socket του Server και στη συνέχεια μέσα στη while δέχεται αιτήσεις, δημιουργεί νέο περιγραφητή του Socket και δημιουργεί νέο Thread του Server (άρα κάνει fork τον Server) και του περνάει ως παράμετρο τον νέο περιγραφητή για να εξυπηρετήσει τον client (δηλ. τη σύνδεση)

Echo Server

- Θα υλοποιήσουμε ένα TCP Echo client και ένα επαναληπτικό TCP echo Server στο port 7
- Ο client θα διαβάζει ένα κείμενο από ένα αρχείο και θα το αποστέλλει γραμμή-γραμμή στον Server
- Ο Server θα διαβάζει το αρχείο και θα το αποστέλλει πίσω στον client

Echo Client (1)


```
1  package aueb.elearn.echo;
2
3  import java.io.*;
4  import java.net.*;
5  import java.util.Scanner;
6
7  public class EchoCli {
8    public static void main(String a[]) {
9      try{
10           InetAddress servAddr = InetAddress.getByName("127.0.0.1");
11           final int servPort = 7;
12           Socket sockfd = new Socket(servAddr, servPort);
13
14           PrintWriter outfd = new PrintWriter(
15                   sockfd.getOutputStream(), true); // true for auto-flush
16           BufferedReader infd =  new BufferedReader(
17                   new InputStreamReader(sockfd.getInputStream()));
18
19           File file = new File("C:\\THANASSIS\\test.txt");
20           Scanner in = new Scanner(file);
```

Socket

File Descriptors

Scanner με Αρχείο

- Εδώ χρησιμοποιούμε αρχείο για να διαβάζουμε γιατί δεν είναι πάντα εύκολο να δίνουμε ειδικούς χαρακτήρες όπως αλλαγές γραμμής ή τέλος αρχείου από την κονσόλα

Echo Client (2)


```
22            String userInput = null;
23            do{
24                userInput = in.nextLine();
25                outfd.println(userInput);
26                outfd.flush();
27                System.out.println("Echo from Server:" + infd.readLine());
28            } while (!userInput.equals("BYE"));
29
30            sockfd.close();
31            in.close();
32        }catch (IOException e1){
33            e1.printStackTrace();
34        }
35    }
36 }
```

- Ο client διαβάζει γραμμή-γραμμή από το αρχείο, τα στέλνει στον Server (το flush χρειάζεται αν δεν έχουμε δώσει true στην 2η παράμετρο του PrintWriter) και τέλος εμφανίζει αυτό που διαβάζει από τον Server

Echo Server (1)


```
1  package aueb.elearn.echo;
2
3  import java.net.*;
4  import java.io.*;
5
6  class EchoServer2 extends Thread {
7     ServerSocket servfd;
8     final int serverPort = 7;
9
10    @Override
11    public void run() {
12       try {
13          servfd = new ServerSocket();
14          servfd.bind(new InetSocketAddress("127.0.0.1", serverPort), 3);
15
16          System.out.println("Echo Server is ready for accepting connections ..");
17
18          while (true) {
19             Socket connectedfd = servfd.accept();
20
21             PrintWriter outfd = new PrintWriter(connectedfd.getOutputStream(), true);
22             BufferedReader infd = new BufferedReader(new InputStreamReader(connectedfd.getInputStream()));
23             System.out.println("Client connected on port " + serverPort +". Servicing requests.");
```

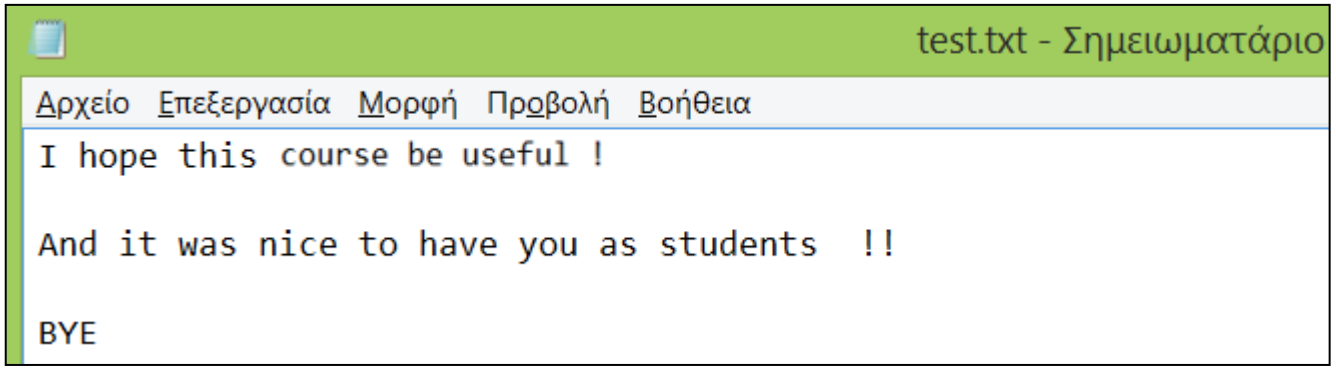
- Δημιουργούμε ένα Listening Socket καθώς και τους Περιγραφητές των αρχείων

Echo Server (2)

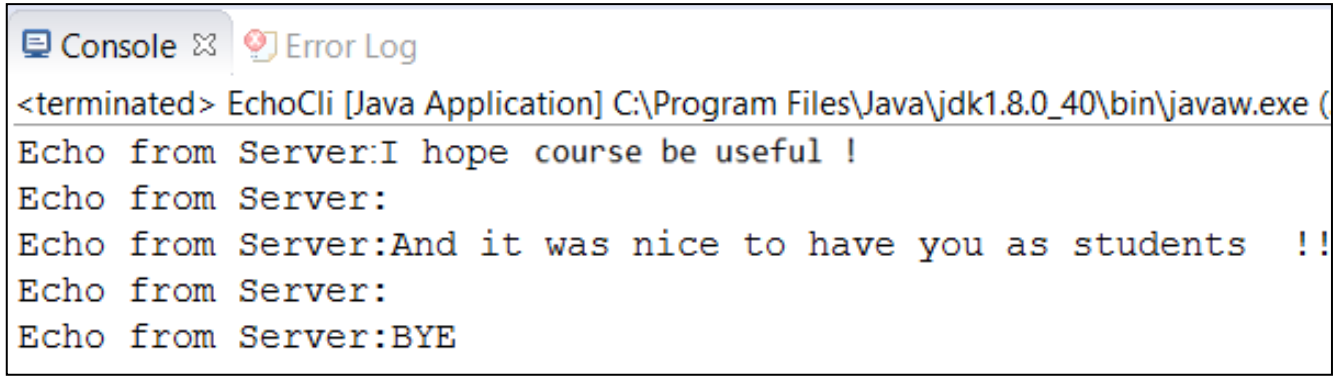

```
25                String inputLine;
26                do{
27                    inputLine = infd.readLine();
28                    outfd.println(inputLine);
29
30                } while (!inputLine.equals("BYE"));
31
32                connectedfd.close();
33            }
34        }
35        catch(IOException e) {
36           e.printStackTrace();
37        }
38    }
39 }
40
41 public class EchoServ {
42     public static void main(String args[]){
43         EchoServer2 es = new EchoServer2();
44         es.start();
45     }
46 }
```

- Μέχρι να βρεθεί “BYE” διαβάζουμε από τον client και γράφουμε πίσω στον client γραμμή-γραμμή
- Κλείνουμε τη σύνδεση

Demo - Αρχείο



- Αρχείο με το κείμενο που στέλνει ο client



- Η απόκριση του Server όπως την εμφανίζει ο client

Μικρή εργασία

- Υλοποιήστε ένα concurrent file server και τον αντίστοιχο client
- Ο client ζητάει ένα αρχείο στέλνοντας το file path και ο server αν υπάρχει το path στέλνει τα περιεχόμενα του αρχείου αλλιώς μήνυμα λάθους