ΔΙΚΤΥΑ ΥΠΟΛΟΓΙΣΤΩΝ

Στρώμα εφαρμογής

Περίληψη

- Εννοιολογικά θέματα και θέματα υλοποίησης για τα πρωτόκολλα εφαρμογής
  - Αρχιτεκτονικές εφαρμογών
  - Απαιτήσεις εφαρμογών για την υπηρεσία μεταφοράς
- Μερικές δημοφιλείς εφαρμογές και πρωτόκολλα του στρώματος εφαρμογής
  - Web και HTTP
  - P2P διανομή αρχείων

Περιεχόμενα

- Γενικά για τις εφαρμογές δικτύου
- Αρχιτεκτονικές εφαρμογών δικτύου
  - αρχιτεκτονική client-server
  - αρχιτεκτονική peer-to-peer
- Επικοινωνία διαδικασιών
  - διεπαφές στρώματος εφαρμογής
  - υπηρεσίες δικτύου που απαιτούνται για τις εφαρμογές
- Web and HTTP
- Εφαρμογές P2P
  - διανομή αρχείων
  - Internet telephony

Μερικές εφαρμογές δικτύου

- e-mail
- web
- instant messaging
- remote login
- P2P file sharing
- multi-user network games
- streaming stored video clips
- voice over IP
- real-time video conferencing
- grid computing
-
-
-

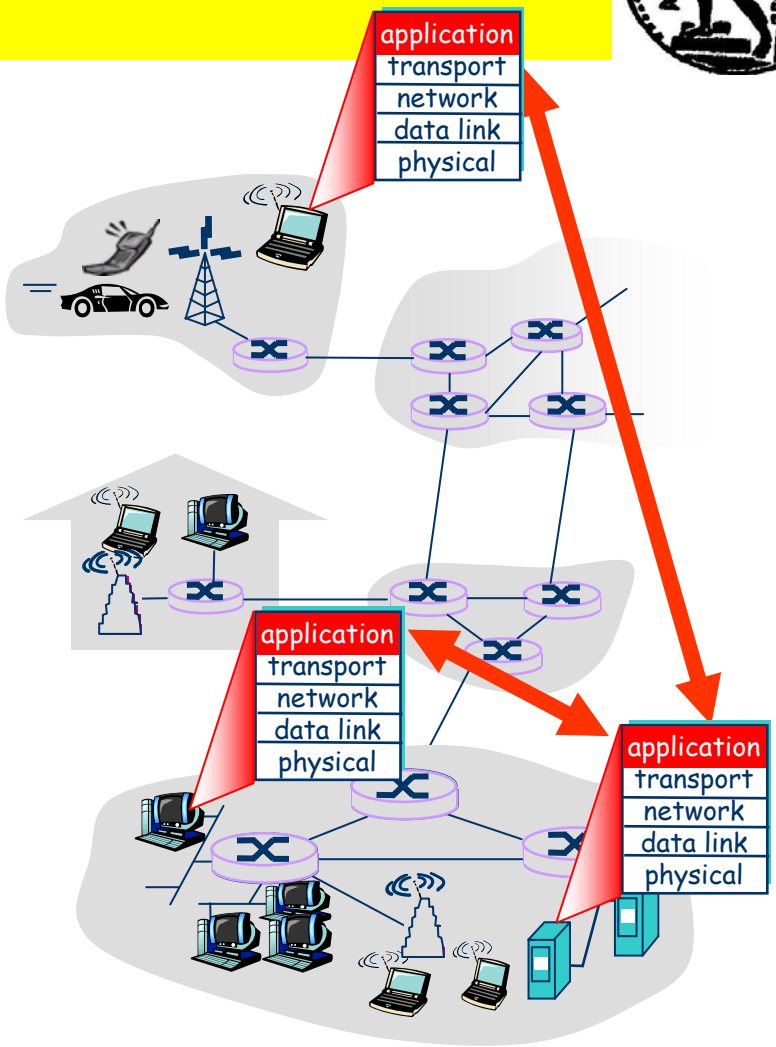
Δημιουργία εφαρμογής δικτύου

γράφονται προγράμματα που

- τρέχουν σε (διαφορετικά) τερματικά συστήματα
- επικοινωνούν πάνω από το δίκτυο
  - π.χ., το software του web server επικοινωνεί με το software του browser

λίγο software γράφεται για συσκευές του πυρήνα του δικτύου

- οι συσκευές του πυρήνα του δικτύου δεν τρέχουν εφαρμογές χρήστη
- οι εφαρμογές στα τερματικά συστήματα επιτρέπουν την ταχεία ανάπτυξη των εφαρμογών και τη διάδοσή τους



Αρχιτεκτονικές εφαρμογών δικτύου

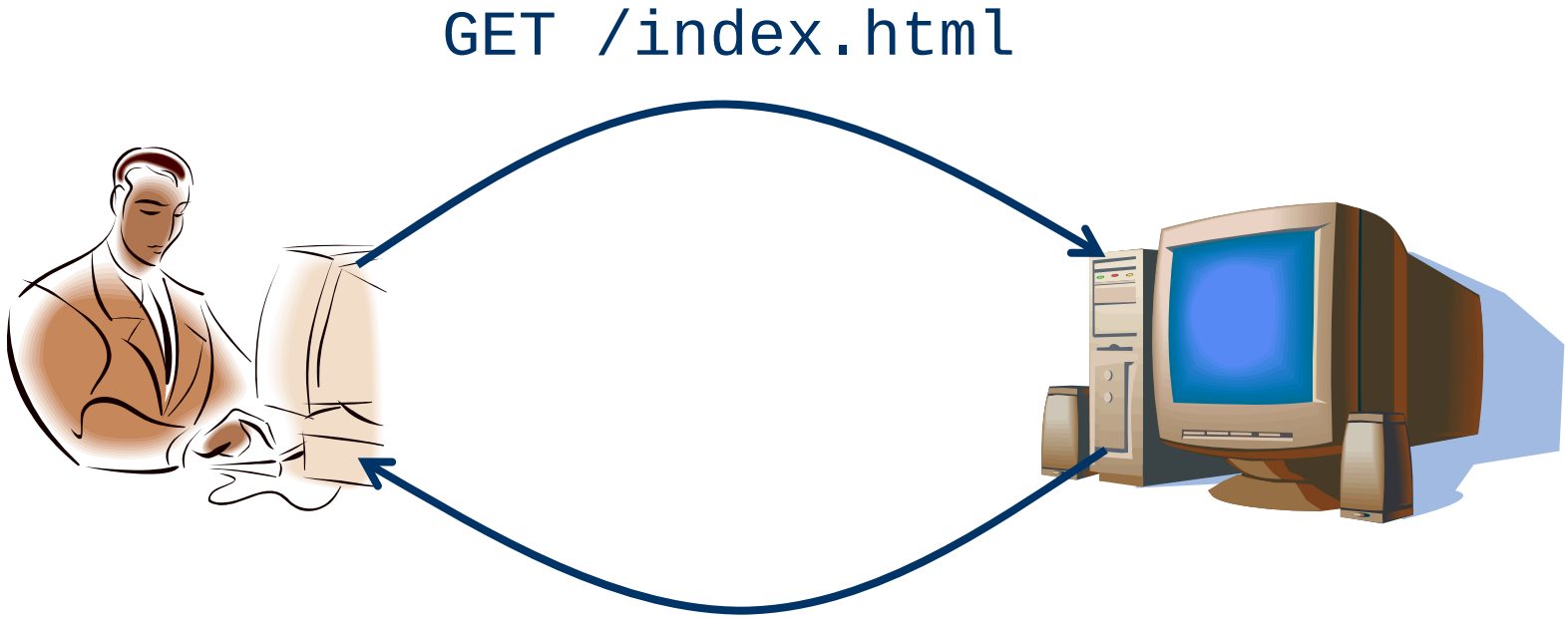
- Η αρχιτεκτονική μιας εφαρμογής είναι διαφορετική από την αρχιτεκτονική δικτύου
- Για τον δημιουργό μιας εφαρμογής, η αρχιτεκτονική δικτύου είναι σταθερή και παρέχει συγκεκριμένες υπηρεσίες στις εφαρμογές
- Η αρχιτεκτονική εφαρμογής σχεδιάζεται από τον δημιουργό της εφαρμογής και δείχνει πώς είναι δομημένη η εφαρμογή πάνω στα διάφορα τερματικά.
- Κατά την επιλογή αρχιτεκτονικής εφαρμογής, αυτός που δημιουργεί την εφαρμογή θα επιλέξει, πιθανότατα, μια από τις δύο επικρατούσες αρχιτεκτονικές: client-server και peer-to-peer (P2P)

Αρχιτεκτονικές εφαρμογών δικτύου

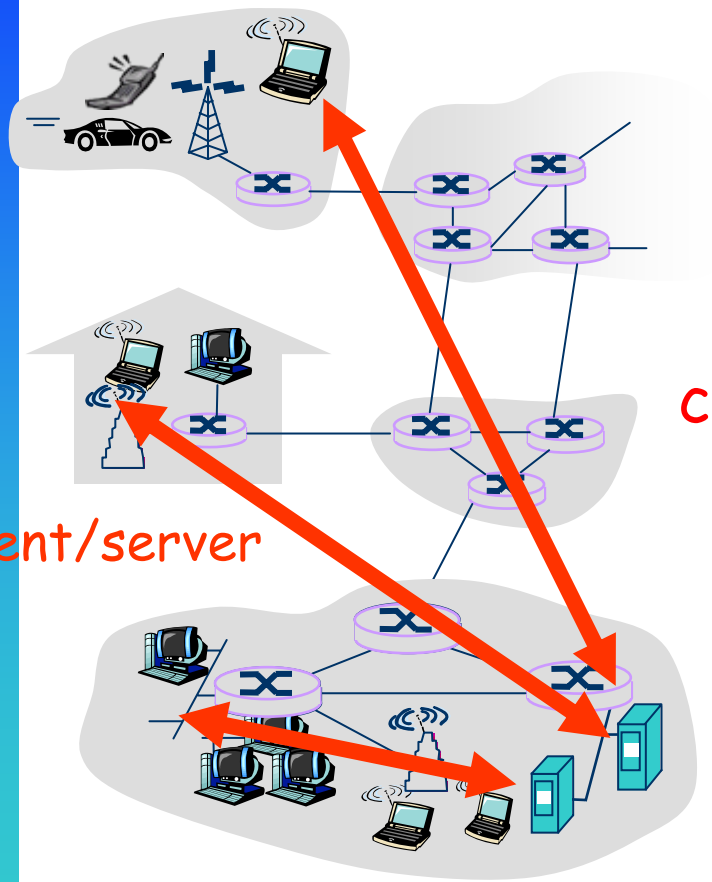
- Client-server: υπάρχει πάντα ένας ενεργοποιημένος host (server) που εξυπηρετεί αιτήσεις από πολλούς άλλους host (clients).
- Peer-to-peer (P2P): αξιοποιεί την άμεση επικοινωνία μεταξύ ζευγών περιστασιακά συνδεομένων host, που ονομάζονται ομότιμοι (peers)
- Υβριδική: συνδυάζει στοιχεία και client-server και P2P

Αρχιτεκτονική Client-server

- Πρόγραμμα Client
  - Τρέχει σε host
  - Ζητάει υπηρεσία
  - π.χ., Web browser
- Πρόγραμμα Server
  - Τρέχει σε host
  - Παρέχει υπηρεσία
  - π.χ., Web server



Αρχιτεκτονική Client-server



client/server

server:

- πάντα ενεργοποιημένος host
- μόνιμη διεύθυνση IP
- δέχεται αιτήσεις από πολλούς client hosts
- ομάδα από servers (server farm) για κλιμάκωση

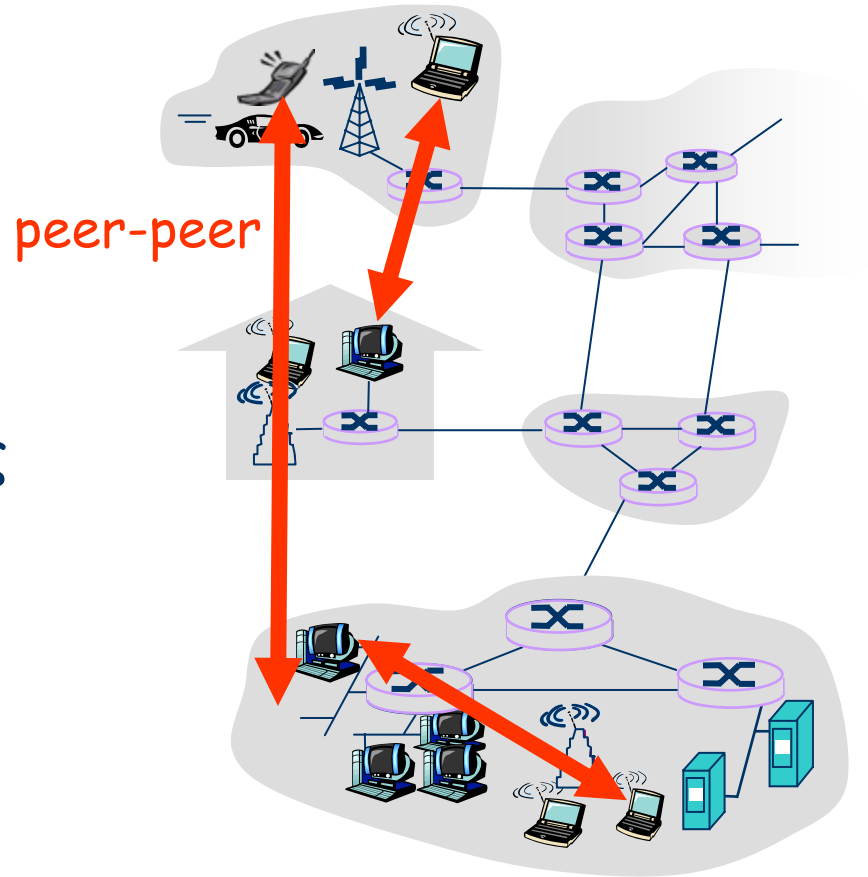
clients:

- επικοινωνούν με τον server
- μπορεί να συνδέονται περιστασιακά
- μπορεί να έχουν δυναμικές διευθύνσεις IP
- δεν επικοινωνούν απευθείας με άλλους clients
- πρέπει να ξέρουν τη διεύθυνση του server

Αμιγής αρχιτεκτονική P2P

- *όχι* πάντα ενεργοποιημένος server
- αυθαίρετα τερματικά συστήματα επικοινωνούν απευθείας
- οι ομότιμοι συνδέονται περιστασιακά και ανταλλάσσουν διευθύνσεις IP
- παράδειγμα: Gnutella

Πολύ κλιμακούμενη αλλά δύσκολα διαχειρίσιμη



Υβριδική client-server και P2P

Skype

- εφαρμογή voice-over-IP P2P
- κεντρικός server: βρίσκει τη διεύθυνση του απόμακρου μέρους
- σύνδεση client-client: άμεση (όχι μέσω server)

Instant messaging

- το chatting μεταξύ δύο χρηστών είναι P2P
- κεντρική υπηρεσία: ανίχνευση παρουσίας client/ εντοπισμός
  - ο χρήστης εγγράφει την IP διεύθυνσή του όταν είναι online
  - ο χρήστης επικοινωνεί με τον κεντρικό server για να βρει διευθύνσεις IP φίλων

Επικοινωνία διαδικασιών

Διαδικασία: πρόγραμμα που τρέχει σε κάποιον host.

- στον ίδιο host, δύο διαδικασίες επικοινωνούν χρησιμοποιώντας inter-process επικοινωνία (καθοριζόμενη από το OS).
- διαδικασίες σε διαφορετικούς host επικοινωνούν με ανταλλαγή μηνυμάτων

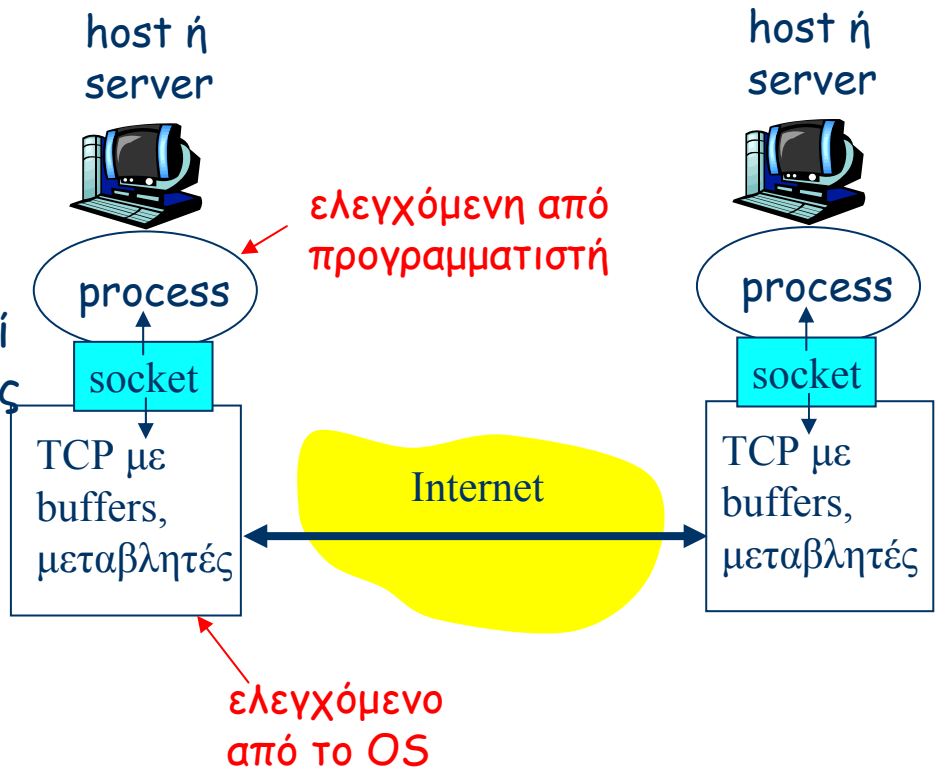
Διαδικασία client:
διαδικασία που αρχίζει την επικοινωνία

Διαδικασία server:
διαδικασία που αναμένει την επαφή

- Σημείωση: εφαρμογές με αρχιτεκτονικές P2P έχουν και διαδικασίες client και διαδικασίες server

Διεπαφή διαδικασίας-δικτύου υπολογιστών

- μια διαδικασία στέλνει /λαμβάνει μηνύματα στο/από το δίκτυο μέσω μιας software διεπαφής, που καλείται υποδοχή (socket)
- διαδικασία/υποδοχή ανάλογο προς το σπίτι/πόρτα
  - -η αποστέλλουσα διαδικασία ωθεί το μήνυμα προς την υποδοχή της
  - -η αποστέλλουσα διαδικασία βασίζεται στην υποδομή μεταφοράς που βρίσκεται στην άλλη πλευρά της υποδοχής, η οποία φέρνει το μήνυμα στην υποδοχή της διαδικασίας λήψης



- API: (1) επιλογή του πρωτοκόλλου μεταφοράς, (2) δυνατότητα ρύθμισης λίγων παραμέτρων (π.χ. max buffer size και MSS)

Διευθυνσιοδότηση διαδικασιών

- για να λάβει μηνύματα, η διαδικασία πρέπει να έχει *ταυτότητα*
- ο host έχει μοναδική διεύθυνση IP 32-bit
- δεν αρκεί η διεύθυνση IP του host στον οποίο τρέχει η διαδικασία για να προσδιορίσει τη διαδικασία.
- *η ταυτότητα* μιας διαδικασίας περιλαμβάνει και τη διεύθυνση IP και τους αριθμούς θυρών που σχετίζονται με τη διαδικασία στον host.
- Παραδείγματα αριθμών θυρών:
  - HTTP server: 80
  - Mail server: 25
- για να σταλεί μήνυμα HTTP στον web server edu-dy.cn.ntua.gr:
  - διεύθυνση IP address: 147.102.40.9
  - αριθμός θύρας: 80

Ποια υπηρεσία μεταφοράς χρειάζεται για μια εφαρμογή;

Απώλειες δεδομένων

- μερικές εφαρμογές (π.χ., audio) μπορεί να ανέχονται μερικές απώλειες
- άλλες εφαρμογές (π.χ., file transfer, telnet) απαιτούν 100% αξιόπιστη μεταφορά δεδομένων

Καθυστέρηση

- μερικές εφαρμογές (π.χ., Internet telephony, interactive games) απαιτούν μικρή καθυστέρηση για να είναι "αποτελεσματικές"

Εύρος ζώνης

- μερικές εφαρμογές (π.χ., multimedia) απαιτούν κάποιο ελάχιστο εύρος ζώνης για να είναι "αποτελεσματικές"
- άλλες εφαρμογές ("ελαστικές") χρησιμοποιούν οποιοδήποτε εύρος ζώνης είναι διαθέσιμο

Απαιτήσεις εφαρμογών για την υπηρεσία μεταφοράς

Εφαρμογή	Απώλειες	Εύρος ζώνης	Ευαισθησία στην καθυστέρηση
file transfer	όχι	ευέλικτο	όχι
e-mail	όχι	ευέλικτο	όχι
Web documents	όχι	ευέλικτο	όχι
real-time audio/ Video	ανεκτές	audio: 5kbps-1Mbps video:10kbps-5Mbps	ναι, εκατοντάδες ms
stored audio/video	ανεκτές	όπως ανωτέρω	ναι, λίγα sec
interactive games	ανεκτές	λίγα kbps – 10kbps	ναι, εκατοντ. ms
instant messaging	όχι	ευέλικτο	ναι και όχι

Υπηρεσίες μεταφοράς στο Internet

Υπηρεσία TCP:

- *με σύνδεση:* απαιτείται εγκατάσταση σύνδεσης μεταξύ των διαδικασιών client και server
- *αξιόπιστη μεταφορά* μεταξύ διαδικασίας εκπομπής και διαδικασίας λήψης
- *έλεγχος ροής:* ο πομπός δεν πλημμυρίζει τον δέκτη
- *έλεγχος συμφόρησης:* εμποδίζει τον πομπό όταν το δίκτυο είναι υπερφορτωμένο
- *δεν παρέχει:* χρονική εγγύηση και εξασφάλιση ελάχιστου εύρους ζώνης

Υπηρεσία UDP:

- αναξιόπιστη μεταφορά δεδομένων μεταξύ διαδικασίας εκπομπής και διαδικασίας λήψης
- δεν παρέχει: εγκατάσταση σύνδεσης, αξιοπιστία, έλεγχο ροής, έλεγχο συμφόρησης, χρονική εγγύηση ή εξασφάλιση εύρους ζώνης

Πρωτόκολλα στρώματος εφαρμογής

- Ένα πρωτόκολλο στρώματος εφαρμογής ορίζει:
  - Τύπους ανταλλασσόμενων μηνυμάτων,
    - π.χ., request, response
  - Συντακτικό μηνυμάτων:
    - ποια πεδία στο μήνυμα και πώς τα πεδία περιγράφονται
  - Σημασιολογία των μηνυμάτων
    - σημασία της πληροφορίας στα διάφορα πεδία
  - Κανόνες για το πότε και πώς οι διαδικασίες στέλνουν και απαντούν σε μηνύματα
- Public-domain protocols: ορίζονται στα RFC
  - επιτρέπουν διαλειτουργία
  - π.χ., HTTP, SMTP
- Proprietary protocols:
  - π.χ., Skype

Εφαρμογές στο Internet

Πρωτόκολλα εφαρμογής και μεταφοράς

Εφαρμογή	Πρωτόκολλο στρ. εφαρμογής	Πρωτόκολλο στρ. μεταφοράς
e-mail	SMTP [RFC 2821]	TCP
remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
file transfer	FTP [RFC 959]	TCP
streaming multimedia	HTTP (π.χ., YouTube), RTP	TCP ή UDP
Internet telephony	SIP, RTP ή proprie-tary (π.χ., Skype)	τυπικά UDP

Web και HTTP

Βασικοί ορισμοί

- Μια ιστοσελίδα αποτελείται από αντικείμενα
- Αντικείμενο μπορεί να είναι ένα αρχείο HTML, μια εικόνα JPEG, Java applet, αρχείο audio,…
- Μια ιστοσελίδα αποτελείται από βασικό αρχείο HTML που περιέχει αρκετά αναφερόμενα αντικείμενα
- Κάθε αντικείμενο διευθυνσιοδοτείται με ένα URL
- Παράδειγμα URL:

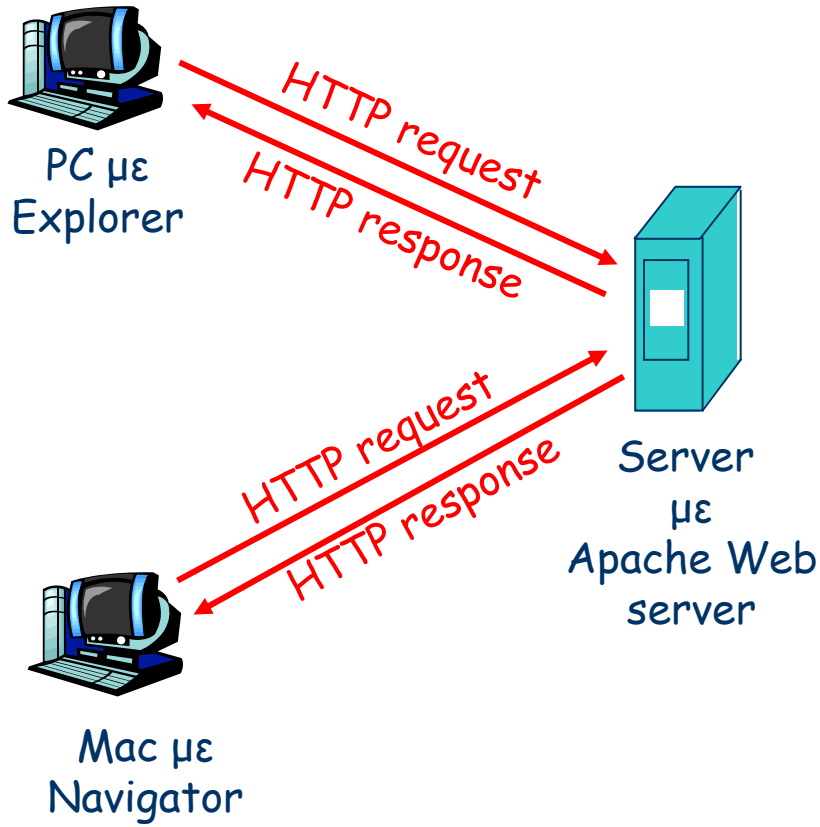
`www.someschool.edu/someDept/pic.gif`

όνομα host | όνομα διαδρομής

HTTP: Εισαγωγή

HTTP: HyperText Transfer Protocol

- πρωτόκολλο στρώματος εφαρμογής του Web
- μοντέλο client/server
  - *client:* browser που ζητάει, λαμβάνει, "απεικονίζει" Web αντικείμενα
  - *server:* ο Web server στέλνει αντικείμενα απαντώντας σε αιτήσεις
- HTTP 1.0: RFC 1945
- HTTP 1.1: RFC 2068



HTTP: Εισαγωγή (2)

Χρησιμοποιεί TCP:

- ο client ξεκινά σύνδεση TCP (δημιουργεί υποδοχή) προς τον server, θύρα 80
- ο server αποδέχεται τη σύνδεση TCP από τον client
- μηνύματα HTTP (μηνύματα πρωτοκόλλου στρώματος εφαρμογής) ανταλλάσσονται μεταξύ του browser (HTTP client) και του Web server (HTTP server)
- η σύνδεση TCP κλείνει

Το HTTP είναι "ακαταστατικό"

- ο server δεν κρατάει πληροφορίες για προηγούμενες αιτήσεις του client

Τα πρωτόκολλα που διατηρούν "κατάσταση" είναι πολύπλοκα!

- πρέπει να διατηρείται πληροφορία για το παρελθόν (κατάσταση)
- αν ο server/client χαλάσει, οι εικόνες τους για την κατάσταση μπορεί να είναι ασύμβατες και πρέπει να , ξαναγίνουν συμβατές

Συνδέσεις HTTP

Μη επίμονο HTTP

- Το πολύ ένα αντικείμενο στέλνεται πάνω από μια σύνδεση TCP.
- Το HTTP/1.0 χρησιμοποιεί μη επίμονο HTTP

Επίμονο HTTP

- Πολλά αντικείμενα μπορεί να σταλούν πάνω από την ίδια σύνδεση TCP μεταξύ client και server.
- Το HTTP/1.1 χρησιμοποιεί επίμονο HTTP στον default τρόπο λειτουργίας

Μη επίμονο HTTP

(περιέχει κείμενο, αναφορές για 10 εικόνες jpeg)

Υποθέστε ότι ο χρήστης εισάγει το URL
`www.someSchool.edu/someDepartment/home.index`

1a. Ο HTTP client ξεκινά μια σύνδεση TCP προς τον HTTP server στο www.someSchool.edu , port 80

1b. Ο HTTP server στον host www.someSchool.edu αναμένει για σύνδεση TCP στην θύρα 80, "αποδέχεται" τη σύνδεση ειδοποιώντας τον client

2. Ο HTTP client στέλνει HTTP *request message* (που περιέχει το URL) στην υποδοχή της σύνδεσης TCP. Το μήνυμα δείχνει ότι ο client θέλει το αντικείμενο someDepartment/home.index

3. Ο HTTP server λαμβάνει το μήνυμα αίτησης, σχηματίζει ένα *response message* που περιέχει το αντικείμενο που ζητήθηκε και στέλνει μήνυμα στην υποδοχή του

χρόνος

Μη επίνονο HTTP (2)

χρόνος

4. Ο HTTP server κλείνει τη σύνδεση TCP.

5. Ο HTTP client λαμβάνει το μήνυμα απάντησης που περιέχει το αρχείο html, απεικονίζει το html. Αναλύοντας το αρχείο html, βρίσκει αναφορές για 10 αντικείμενα jpeg

6. Τα βήματα 1-5 επαναλαμβάνονται για κάθε ένα από τα 10 αντικείμενα jpeg

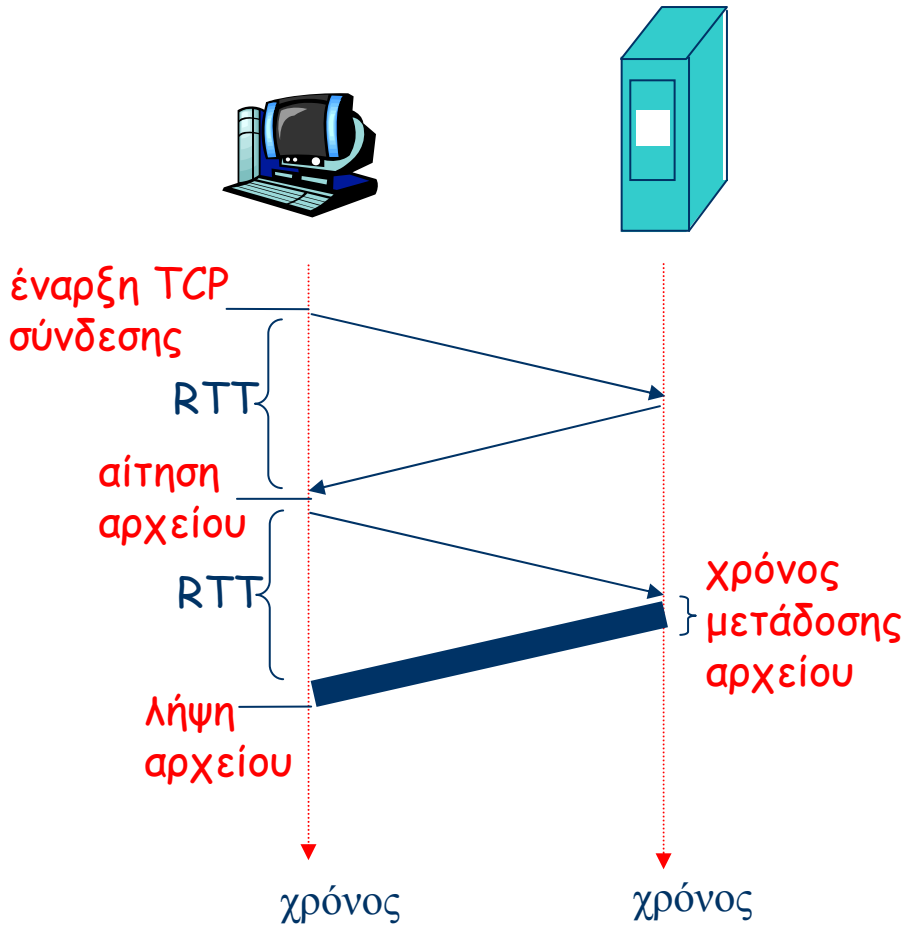
Μη επίμονο HTTP: χρόνος απόκρισης

Ορισμός του RTT: χρόνος για να πάει ένα μικρό πακέτο από τον client στον server και πίσω.

Το RTT περιλαμβάνει χρόνους διάδοσης και χρόνους αναμονής και επεξεργασίας στους ενδιάμεσους δρομολογητές

Χρόνος απόκρισης:

- ένα RTT για την έναρξη της σύνδεσης TCP
- ένα RTT για την αίτηση HTTP και την επιστροφή των πρώτων λίγων byte της HTTP απάντησης
- χρόνος μετάδοσης αρχείου



total = 2RTT+χρόνος μετάδοσης

Επίμονο HTTP

Μη επίμονο HTTP:

- απαιτεί 2 RTT ανά object
- overhead στο OS για *κάθε* σύνδεση TCP
- οι browser ανοίγουν συχνά παράλληλες συνδέσεις TCP για να φέρουν αναφερόμενα objects

Επίμονο HTTP

- ο server αφήνει ανοικτή τη σύνδεση μετά την αποστολή της απάντησης
- διαδοχικά μηνύματα HTTP μεταξύ των ίδιων client/server στέλνονται πάνω από την ανοικτή σύνδεση

Επίμονο ***χωρίς*** συνεχή παροχή:

- ο client κάνει νέα αίτηση μόνο όταν ληφθεί η προηγούμενη απάντηση
- ένα RTT για κάθε αναφερόμενο object

Επίμονο ***με*** συνεχή παροχή:

- ο client στέλνει αιτήσεις μόλις συναντήσει ένα αναφερόμενο object
- κατ' ελάχιστον ένα RTT για όλα τα αναφερόμενα objects
- default στο HTTP/1.1

Μήνυμα αίτησης HTTP

- δύο τύποι μηνυμάτων HTTP: *request, response*
- HTTP request:
  - ASCII (μορφή αναγνώσιμη από ανθρώπους)

γραμμή αίτησης (GET, POST, HEAD εντολές)

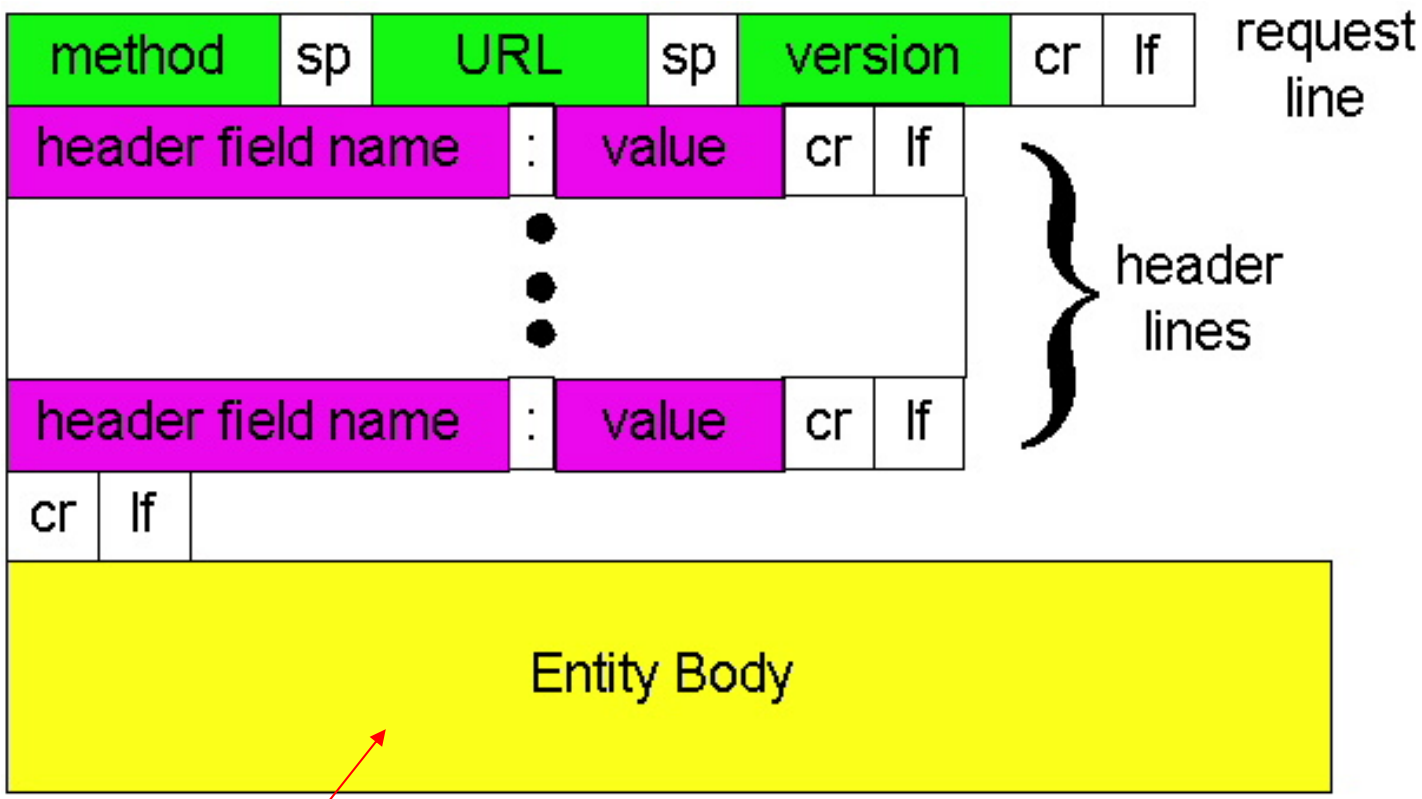
γραμμές επικεφαλίδας

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
Connection: close
User-agent: Mozilla/4.0
Accept-language:fr
```

Carriage return, line feed δηλώνει το τέλος του μηνύματος

(extra carriage return, line feed)

HTTP request: γενική μορφή



άδειο στο GET, χρησιμοποιείται στο POST

Αίτηση με φόρμα εισόδου

Μέθοδος Post:

- Η Web page περιέχει συχνά φόρμα εισόδου
- Τα δεδομένα εισόδου ανεβάζονται στον server μέσα στο entity body

Μέθοδος URL:

- Χρησιμοποιεί τη μέθοδο GET
- Τα δεδομένα εισόδου μπαίνουν στο πεδίο URL της γραμμής αίτησης:

```
www.somesite.com/animalsearch?monkeys&banana
```

Άλλες μέθοδοι αίτησης

Μέθοδος HEAD:

- Παρόμοια με τη GET
- Όταν ο server λαμβάνει αίτηση με τη μέθοδο HEAD απαντάει αφήνοντας εκτός του ζητούμενο αντικείμενο.
- Χρησιμοποιείται κυρίως για debugging.

Μέθοδος PUT:

- Επιτρέπει στον χρήστη να ανεβάσει ένα αντικείμενο σε συγκεκριμένη διαδρομή (directory) ενός server
- Χρησιμοποιείται επίσης από εφαρμογές που χρειάζεται να ανεβάσουν αντικείμενα σε Web servers

Μέθοδος DELETE:

- Επιτρέπει στον χρήστη ή σε μια εφαρμογή να απαλείψει ένα αντικείμενο από έναν Web server

Τύποι μεθόδων στο HTTP

HTTP/1.0

- GET
- POST
- HEAD
  - ζητά από τον server να μην συμπεριλάβει στην απάντηση το αιτούμενο object

HTTP/1.1

- GET, POST, HEAD
- PUT
  - τοποθετεί αρχείο στο entity body σε διαδρομή που καθορίζεται σε πεδίο URL
- DELETE
  - απαλείφει το αρχείο που ορίζεται στο πεδίο URL

HTTP response

γραμμή κατάστασης (πρωτόκολλο κωδ. κατάστασης κατάσταση)

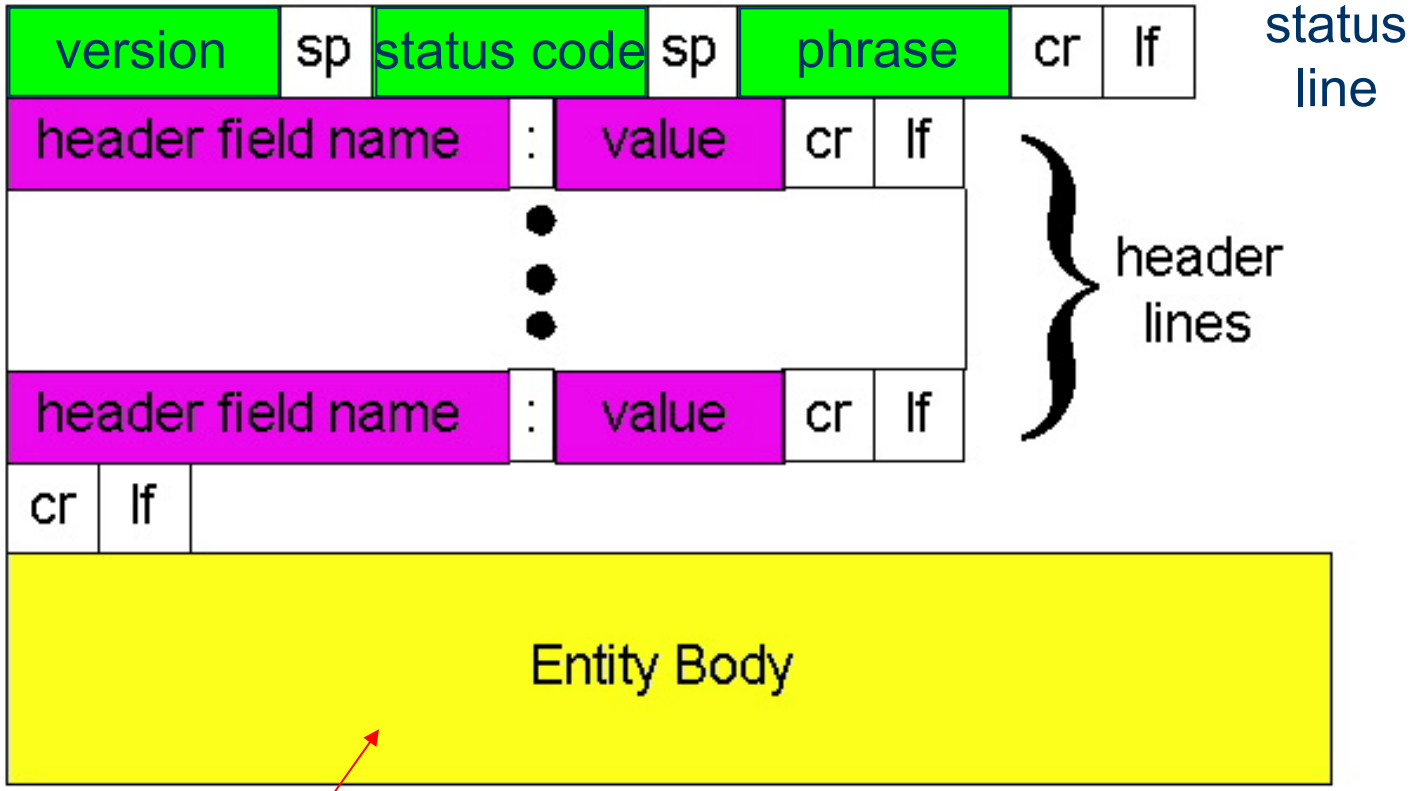
γραμμές επικεφαλίδας

δεδομένα, π.χ., ζητούμενο αρχείο HTML

```
HTTP/1.1 200 OK
Connection close
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998 …...
Content-Length: 6821
Content-Type: text/html

data data data data data ...
```

HTTP response: γενική μορφή



περιέχει το ζητηθέν αντικείμενο

HTTP response: κωδικοί κατάστασης

Στην πρώτη γραμμή του μηνύματος απάντησης server-> client.

Παραδείγματα κωδικών:

200 OK

- η αίτηση πέτυχε, το ζητηθέν αντικείμενο ακολουθεί μέσα σ' αυτό το μήνυμα

301 Moved Permanently

- το ζητηθέν αντικείμενο μετακινήθηκε, η νέα θέση καθορίζεται παρακάτω σ' αυτό το μήνυμα (Location:)

400 Bad Request

- το μήνυμα αίτησης δεν έγινε κατανοητό από τον server

404 Not Found

- το ζητηθέν αντικείμενο δεν βρέθηκε σ' αυτόν τον server

505 HTTP Version Not Supported

Κατάσταση user-server: cookies

Πολλές μεγάλες ιστοθέσεις χρησιμοποιούν cookies

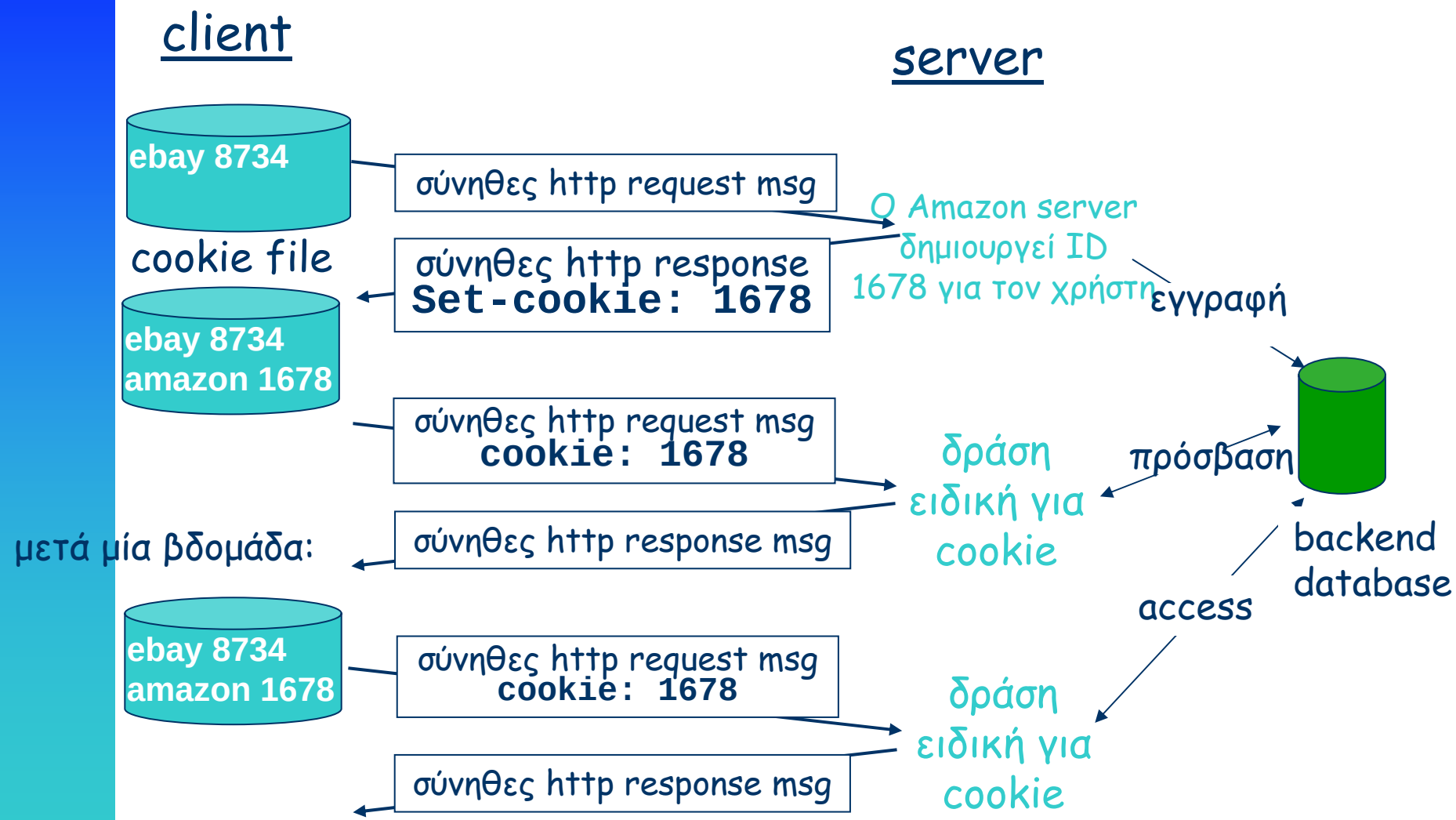
4 μέρη:

1) γραμμή επικεφαλίδας cookie στο μήνυμα HTTP *response*
2) γραμμή επικεφαλίδας cookie στο μήνυμα HTTP *request*
3) αρχείο cookie διατηρούμενο στον host του χρήστη και υφιστάμενο διαχείριση από τον browser του χρήστη
4) back-end database στο Web site

Παράδειγμα:

- Ο χρήστης Χ κάνει πάντα πρόσβαση στο Internet από το PC του
- επισκέπτεται το Amazon για πρώτη φορά
- όταν η αρχική αίτηση HTTP φθάσει στον Amazon server, ο server δημιουργεί:
  - μοναδική ID
  - εγγραφή στην backend database για την ID

Cookies: διατήρηση κατάστασης (2)



Cookies (3)

Τι μπορεί να μεταφέρουν τα cookies:

- εξουσιοδότηση
- κάρτες αγοράς
- συστάσεις
- Web e-mail

Cookies και ιδιωτικό απόρρητο:

- τα cookies επιτρέπουν στα sites να μάθουν πολλά για τους χρήστες
- οι χρήστες μπορεί να δίνουν όνομα και e-mail στα sites

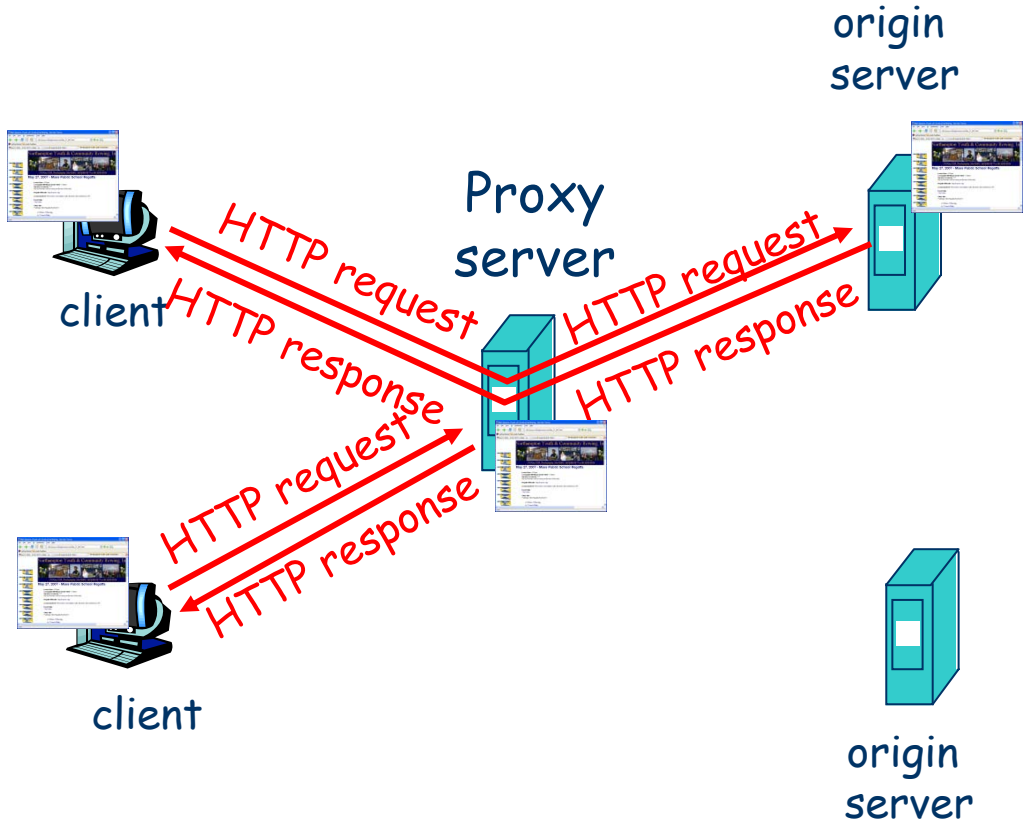
Πώς διατηρείται η "κατάσταση":

- τα άκρα του πρωτοκόλλου: διατηρούν την κατάσταση στον πομπό/δέκτη για πολλαπλές transactions
- cookies: μηνύματα http μεταφέρουν την κατάσταση

Web cache (proxy server)

Στόχος: ικανοποίηση της αίτησης του client χωρίς την ανάμιξη του αρχικού server

- ο χρήστης θέτει στον browser: Web accesses via cache
- ο browser στέλνει όλες τις αιτήσεις HTTP στην cache
  - υπάρχει το object στην cache: η cache επιστρέφει το object
  - αλλιώς η cache ζητά το object από τον αρχικό server και στη συνέχεια επιστρέφει το object στον client



Περισσότερα για το Web caching

- η cache λειτουργεί και ως client και ως server
- τυπικά η cache εγκαθίσταται από τον ISP (πανεπιστήμιο, εταιρία, οικιακό ISP)

Γιατί Web caching;

- περιορίζει τον χρόνο απόκρισης στην αίτηση του client
- περιορίζει την κίνηση στη ζεύξη πρόσβασης ενός ιδρύματος.
- Internet με μεγάλη πυκνότητα από cache: δίνει τη δυνατότητα σε "φτωχούς" παρόχους περιεχομένου να παραδίδουν περιεχόμενο με αποτελεσματικό τρόπο (αλλά το ίδιο κάνει και το P2P file sharing)

Παράδειγμα Caching

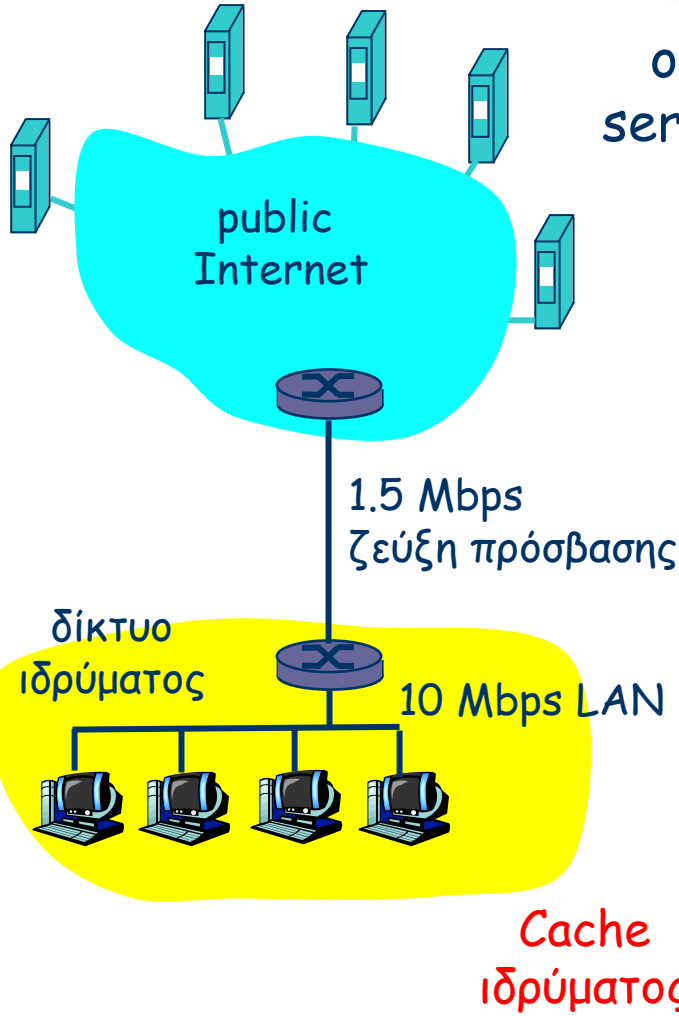
Υποθέσεις

- μέσο μέγεθος object = 100kbit
- μέσος ρυθμός αιτήσεων από τους browser του ιδρύματος προς τους αρχικούς server = 15 αιτήσεις/sec
- καθυστέρηση από τον δρομολογητή του ιδρύματος μέχρι οποιονδήποτε αρχικό server και πίσω μέχρι τον δρομολογητή = 2 sec

Συνέπειες

- χρησιμοποίηση στο LAN = 15%
- χρησιμοποίηση στη ζεύξη πρόσβασης = 100%
- συνολική καθυστέρηση = καθυστέρηση Internet + καθυστέρηση πρόσβασης + καθυστέρηση LAN

  = 2 sec + minutes + milliseconds



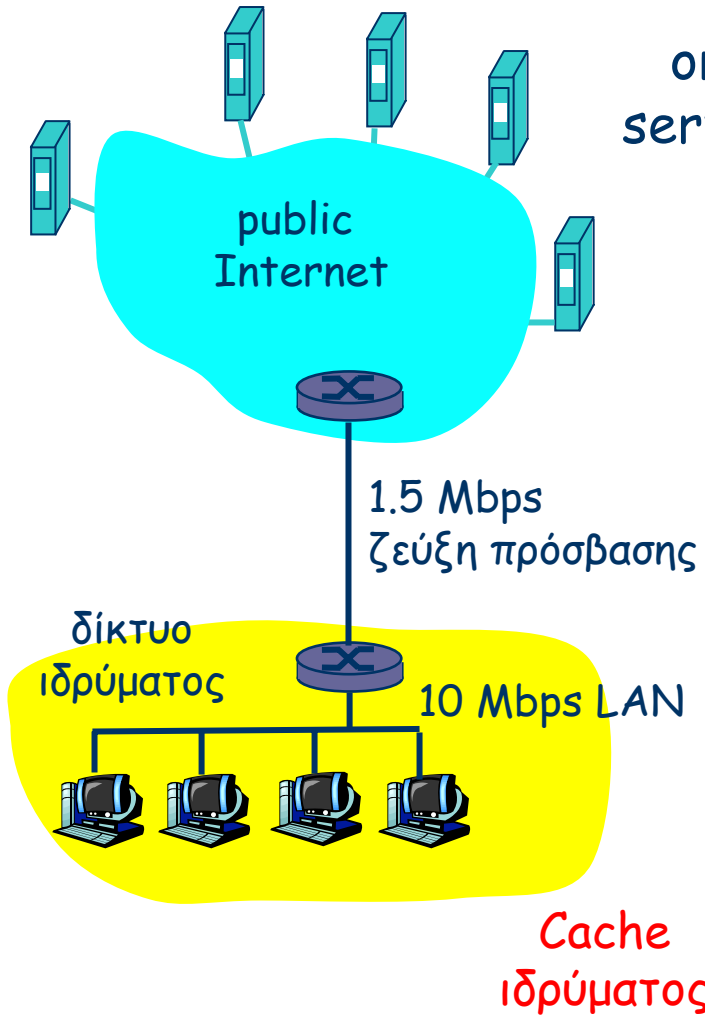
Παράδειγμα Caching (2)

εφικτή λύση

- αύξηση του εύρους ζώνης της ζεύξης πρόσβασης, έστω, στα 10 Mbps

συνέπεια

- χρησιμοποίηση στο LAN = 15%
- χρησιμοποίηση στη ζεύξη πρόσβασης = 15%
- συνολική καθυστέρηση = καθυστέρηση Internet + καθυστέρηση πρόσβασης + καθυστέρηση LAN
  = 2 sec + msecs + msecs
- συνήθως μια δαπανηρή αναβάθμιση



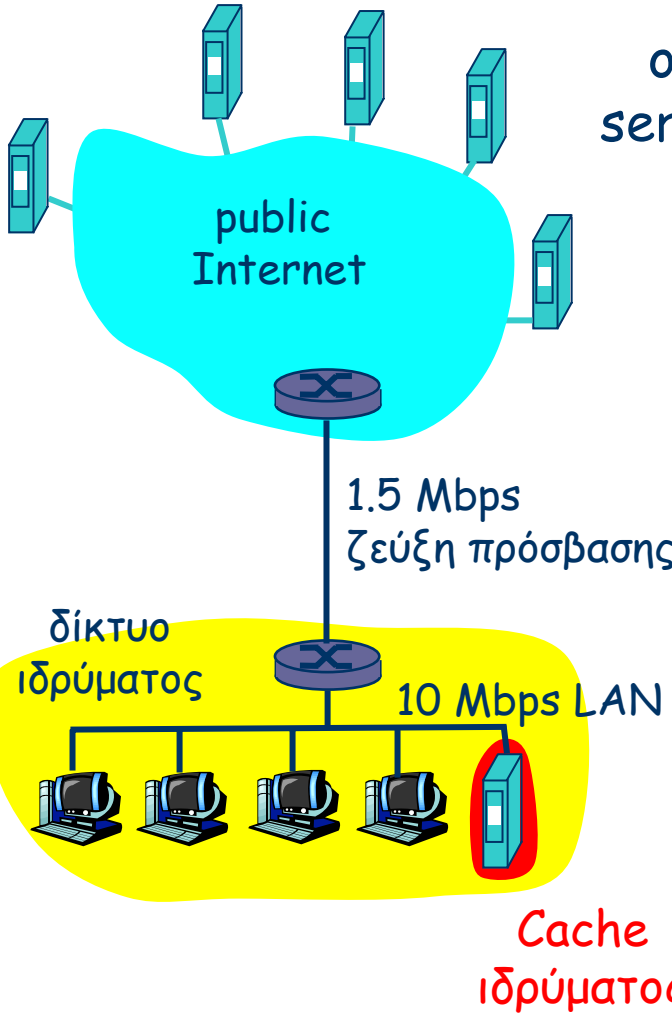
Παράδειγμα Caching (3)

εφικτή λύση: εγκατάσταση cache

- έστω ότι ο ρυθμός επιτυχίας είναι 0.4

συνέπεια

- 40% των αιτήσεων θα ικανοποιούνται σχεδόν αμέσως
- 60% των αιτήσεων θα ικανοποιούνται από τους αρχικούς server
- η χρησιμοποίηση της γραμμής πρόσβασης περιορίζεται στο 60%, με αποτέλεσμα αμελητέες καθυστερήσεις (έστω 10 msec)
- συνολική μέση καθυστέρηση = καθυστέρηση Internet + καθυστέρηση πρόσβασης + καθυστέρηση LAN = 0.6*(2.01) secs + 0.4*milliseconds < 1.4 secs

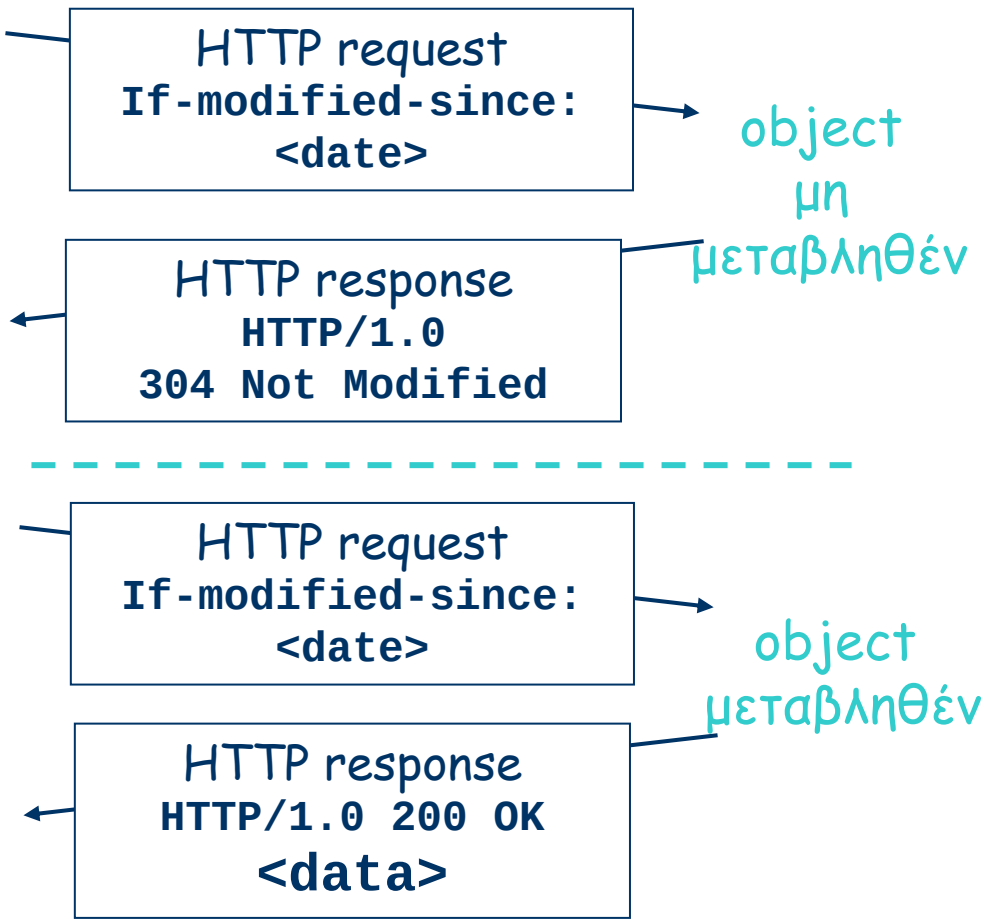


Δυνητικό GET

- Στόχος: να μην αποσταλλεί ένα object αν η cache έχει αποθηκευμένη ενημερωμένη έκδοσή του
- cache: προσδιορίζει την ημερομηνία του αποθηκευμένου αντιγράφου στην αίτηση HTTP

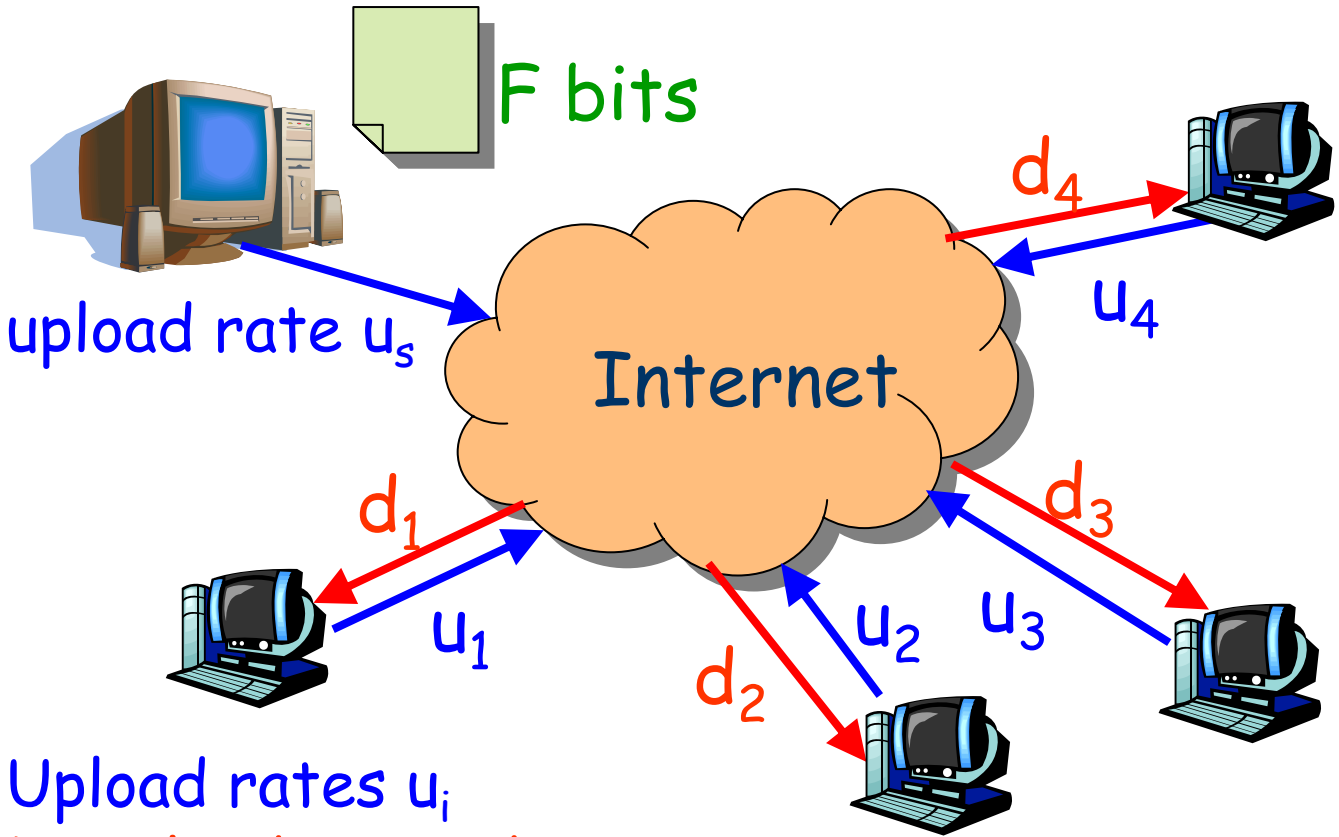
  If-modified-since: <date>
- server: η απάντησή του δεν περιέχει το object αν το αντίγραφο που βρίσκεται στην cache είναι ενημερωμένο:

  HTTP/1.0 304 Not Modified



Διανομή μεγάλου αρχείου από Server

Ερώτηση : Πόσος χρόνος απαιτείται για να διανεμηθεί ένα μεγάλο αρχείο εξ αρχής από έναν server σε *N άλλους υπολογιστές;*



Διανομή μεγάλου αρχείου από Server

- Μετάδοση από τον Server σε N δέκτες
  - Ο Server πρέπει να μεταδώσει NF bits
  - Απαιτείται, τουλάχιστον, χρόνος NF/u_s
- Λήψη δεδομένων
  - Ο πιο αργός δέκτης λαμβάνει με ρυθμό $d_{min} = min_i\{d_i\}$
  - Απαιτείται, τουλάχιστον, χρόνος F/d_{min}

Χρόνος για τη διανομή του F σε N clients με τη μέθοδο client/server = $d_{cs} = \max\{ NF/u_s, F/\min_i(d_i) \}$

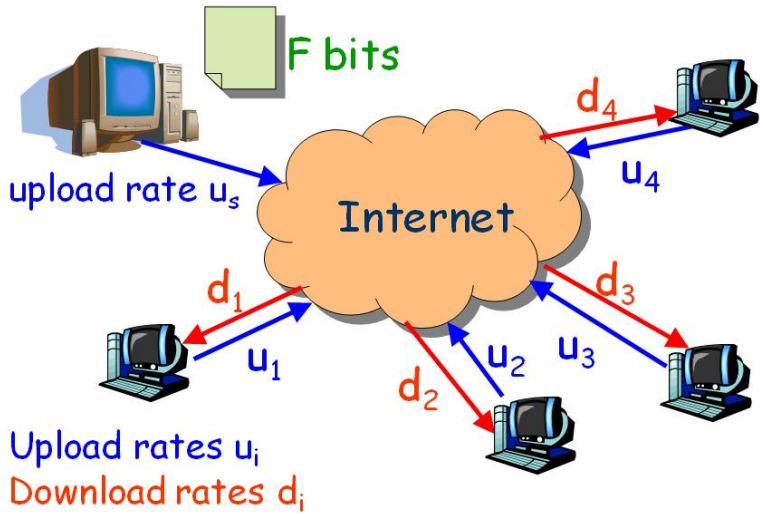
αυξάνει γραμμικά με το N (για μεγάλο N)

Επιτάχυνση της διανομής μεγάλου αρχείου

- Αύξηση του ρυθμού upload του server
  - Μεγαλύτερο εύρος ζώνης στη ζεύξη, για έναν server
  - Για πολλαπλούς servers, μεγαλύτερο εύρος ζώνης στη ζεύξη καθενός
  - Απαιτείται αύξηση της υποδομής
- Εναλλακτική λύση: να βοηθούν οι δέκτες στη διανομή του αρχείου
  - Peer-to-peer file sharing (P2P)
  - Οι δέκτες λαμβάνουν ένα αντίγραφο των δεδομένων
  - Στη συνέχεια το αναδιανέμουν σε άλλους δέκτες
  - Περιορίζεται έτσι ο φόρτος στον server

Διανομή μεγάλου αρχείου P2P

- ο server πρέπει να ανεβάσει ένα αντίγραφο σε χρόνο F/u_s
- ο client i χρειάζεται χρόνο F/d_i για να το κατεβάσει
- πρέπει να κατέβουν (συγκεντρωτικά) NF bits



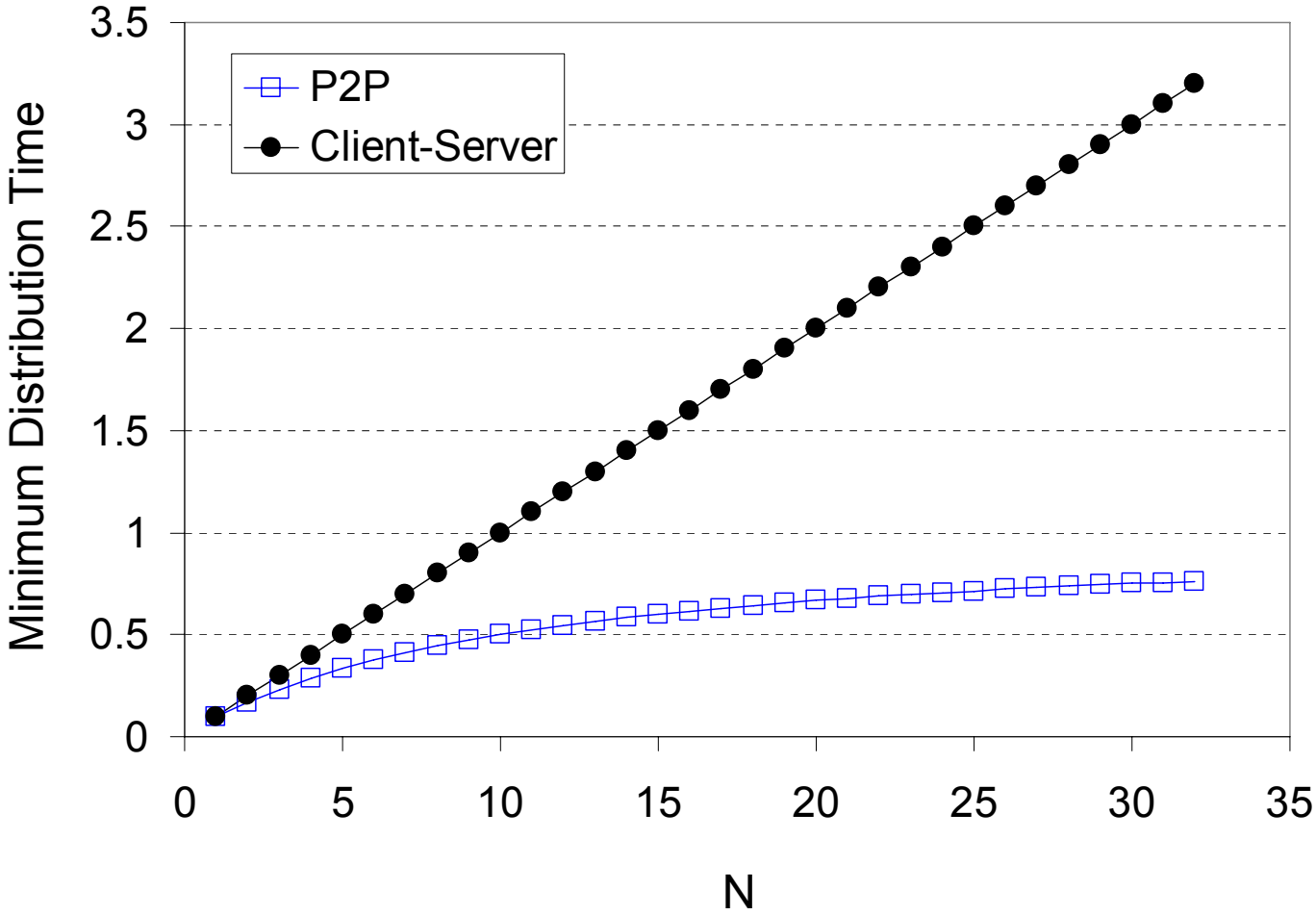
- ταχύτερος εφικτός ρυθμός upload (υποθέτοντας ότι όλοι οι κόμβοι στέλνουν αρχεία στον ίδιο ομότιμο): $u_s + \Sigma u_i$

$$d_{P2P} = \max \{ F/u_s, F/\min_i(d_i) , NF/(u_s + \Sigma u_i)\}$$

Σύγκριση μοντέλων Client-server και P2P

- Χρόνος Download
  - Client-server: $max\{NF/u_s, F/d_{min}\}$
  - Peer-to-peer: $max\{F/u_s, F/d_{min}, NF/(u_s+sum_i(u_i))\}$
- Η λύση peer-to-peer είναι αυτοκλιμακούμενη
  - Πολύ χαμηλότερη απαίτηση σε εύρος ζώνης server
  - Ο χρόνος διανομής αυξάνει αργά με το N
- Αλλά…
  - Οι peers μπορεί να εμφανίζονται και να φεύγουν
  - Οι peers πρέπει να βρουν ο ένας τον άλλον
  - Οι peers πρέπει να θέλουν να βοηθήσουν ο ένας τον άλλον

Σύγκριση μοντέλων Client-server και P2P



$F/u=1h$, $u_s=10u$, $d_{min} \geq u_s$

P2P file sharing

Παράδειγμα

- Ο Χ τρέχει μια εφαρμογή P2P client στο notebook του
- από καιρού εις καιρό συνδέεται στο Internet και λαμβάνει νέα διεύθυνση IP σε κάθε σύνδεση
- ζητά το αρχείο Α
- η εφαρμογή παρουσιάζει άλλους ομότιμους (peers) που έχουν αντίγραφο του Α.
- Ο Χ επιλέγει έναν από τους ομότιμους, τον Υ.
- το αρχείο αντιγράφεται από το PC του Υ στο notebook του Χ: HTTP
- ενώ ο Χ κατεβάζει το αρχείο, άλλοι χρήστες παίρνουν από τον Χ.
- η ομότιμη οντότητα του Χ είναι και Web client και ένας περιστασιακός Web server.

Όλοι οι ομότιμοι είναι servers
⇒ υψηλή κλιμάκωση!

Προκλήσεις για τη λύση P2P

- Οι peers έρχονται και φεύγουν
  - Οι peers συνδέονται περιστασιακά
  - Μπορεί να εμφανιστούν ή να φύγουν οποτεδήποτε
  - Ή να επανέλθουν με διαφορετική διεύθυνση IP
- Πώς εντοπίζονται οι σχετικοί peers;
  - Οι peers που είναι online τώρα
  - Οι peers που έχουν το περιεχόμενο που θέλουμε
- Πώς να δοθούν κίνητρα στους peers να παραμείνουν στο σύστημα;
  - Γιατί να μη φύγουν αμέσως μετά τη λήξη του download;
  - Γιατί να επιφορτίζονται να δίνουν περιεχόμενο σε οποιονδήποτε άλλον;

Αναζήτηση πληροφορίας σε P2P

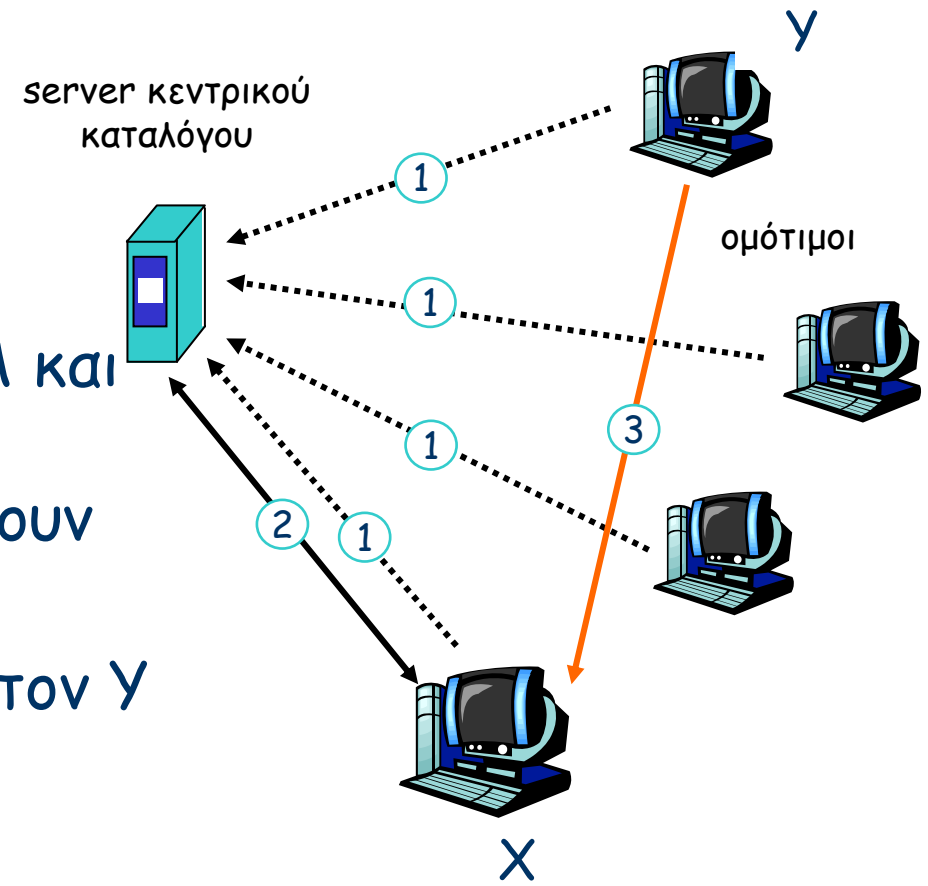
- Βασικό στοιχείο σε πολλές εφαρμογές P2P είναι ένας κατάλογος πληροφοριών – αντιστοίχηση πληροφορίας με θέση host
- Οι ομότιμοι δυναμικά ενημερώνουν τον κατάλογο και ψάχνουν στον κατάλογο
- Υπάρχουν διαφορετικές προσεγγίσεις όσο αφορά την οργάνωση του καταλόγου και τον εντοπισμό των σχετικών με την πληροφορία host από μια κοινότητα ομότιμων.

Εντοπισμός των σχετικών peers

- Τρεις κύριες προσεγγίσεις
  - Κεντρικός κατάλογος (Napster)
  - Query flooding (Gnutella)
  - Ιεραρχική διάρθρωση (KazaA, modern Gnutella)
- Σχεδιαστικοί στόχοι
  - Κλιμάκωση
  - Απλότητα
  - Αντοχή
  - Εύλογη δυνατότητα άρνησης

P2P: κεντρικός κατάλογος

1) όταν ένας ομότιμος συνδέεται, πληροφορεί τον server κεντρικού καταλόγου:
   - διεύθυνση IP
   - περιεχόμενο
2) Ο Χ αναζητά το αρχείο Α και η εφαρμογή παρουσιάζει άλλους ομότιμους που έχουν αντίγραφο του Α.
3) Ο Χ ζητά το αρχείο από τον Υ



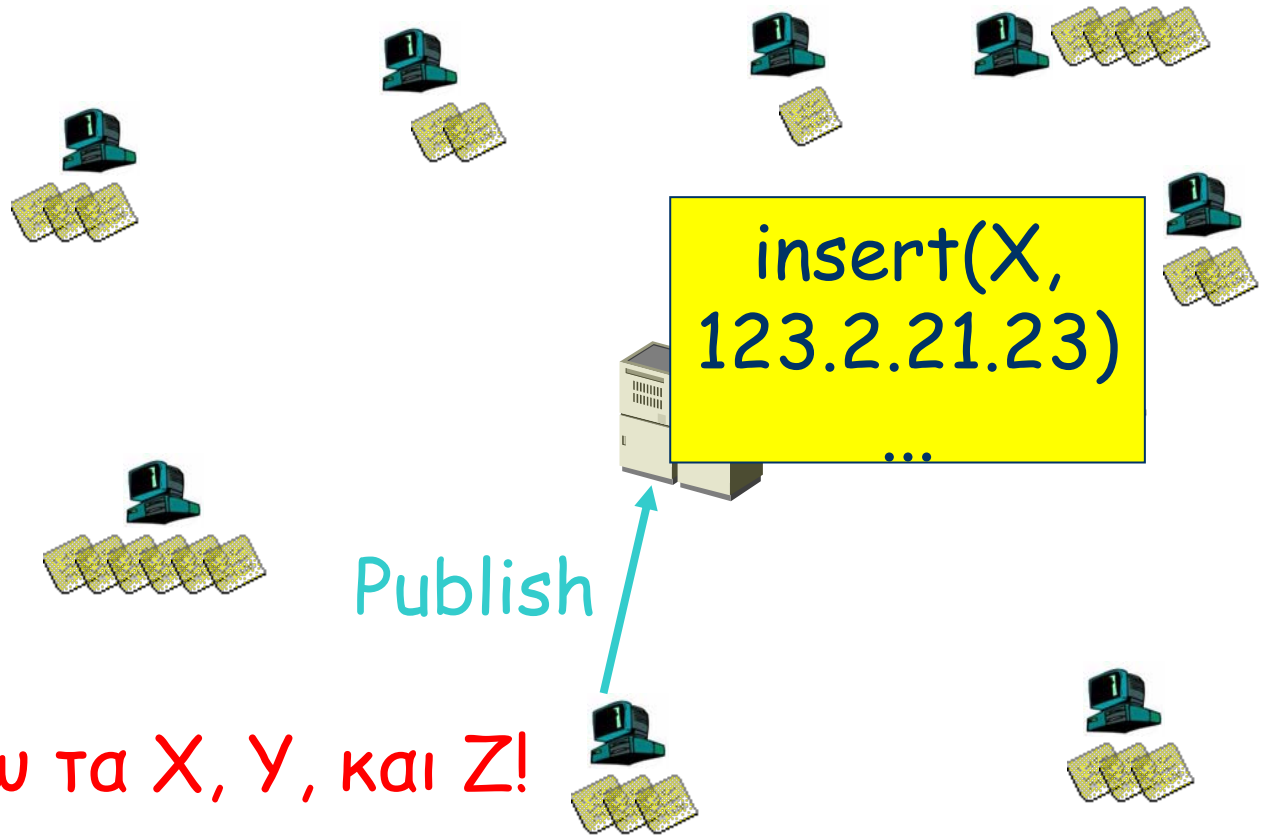
P2P κεντρικού καταλόγου: Napster

- Ιστορία του Napster: η άνοδος
  - Ιανουάριος 1999: Napster version 1.0
  - Μάϊος 1999: ίδρυση εταιρίας
  - Σεπτέμβριος 1999: πρώτες δικαστικές αγωγές
  - 2000: 80 εκατ. χρήστες
- Ιστορία του Napster: η πτώση
  - Μέσα 2001: εκτός λειτουργίας λόγω αγωγών
  - Μέσα 2001: δεκάδες από εναλλακτικές λύσεις P2P που ήταν δυσκολότερο να θιγούν, παρότι βαθμιαία έχουν περιοριστεί
  - 2003: ανάπτυξη υπηρεσιών πληρωμής, όπως iTunes
- Ιστορία του Napster: η επάνοδος
  - 2003: επανίδρυση του Napster ως υπηρεσίας με πληρωμή
  - 2007: γίνεται ακόμα πολύ file sharing

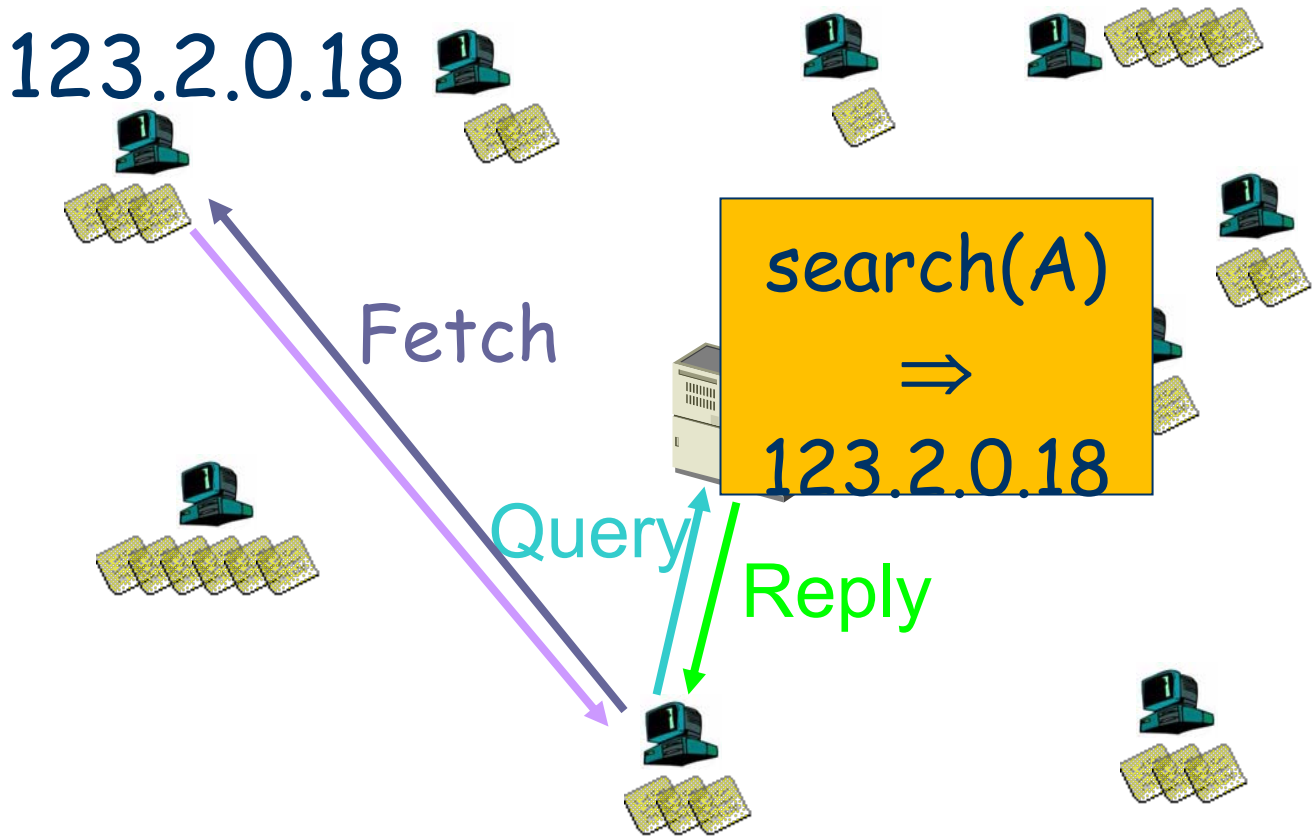
Napster: Σύνοψη λειτουργίας

- Κεντρική βάση δεδομένων
  - **Join**: ξεκινώντας, ο client επικοινωνεί με τον κεντρικό server
  - **Publish**: αναφέρει τη λίστα αρχείων στον κεντρικό server
  - **Search**: ερώτηση στον server για κάποιο αρχείο => απάντηση για κάποιον ομότιμο που έχει το ζητούμενο αρχείο
  - **Fetch**: λήψη του αρχείου απευθείας από ομότιμο (peer)

Napster: Publish



Napster: Search και fetch



Napster: Υπηρεσία καταλόγου

- Εγκατάσταση λογισμικού από τον χρήστη
  - Κατέβασμα του προγράμματος client
  - Εγγραφή: name, password, local directory, κλπ.
- Ο Client συνδέεται στο Napster (με TCP)
  - Παρέχει λίστα μουσικών αρχείων που θα μοιραστεί
  - … και ο κεντρικός server του Napster ενημερώνει τον κατάλογό του
- Ο Client ψάχνει έναν τίτλο ή τραγουδιστή
  - Το Napster προσδιορίζει online clients με το αρχείο
  - … και παρέχει διευθύνσεις IP
- Ο Client ζητάει το αρχείο από τον επιλεγέντα πάροχο
  - Ο πάροχος μεταδίδει το αρχείο στον client
  - Και ο client και ο πάροχος αναφέρουν την κατάσταση στο Napster

Napster: Ιδιότητες

- Ο κατάλογος του Server ενημερώνεται διαρκώς
  - Πάντα γνωρίζει τι μουσική είναι διαθέσιμη κάθε στιγμή
  - Ευάλωτο σημείο για νομικές διώξεις
- Μεταφορά αρχείου peer-to-peer
  - Δεν φορτώνεται στον server
  - Εύλογη δυνατότητα άρνησης για λόγους νομικής δίωξης (αλλά δεν αρκεί)
- Ιδιωτικό (proprietary) πρωτόκολλο
  - Λειτουργίες login, search, upload, download, και status
  - Όχι ασφάλεια: cleartext password
- Θέματα εύρους ζώνης
  - Οι πάροχοι κατατάσσονται από το φαινομενικό εύρος ζώνης και τον χρόνο απόκρισης

Napster: Υπέρ και κατά

- Υπέρ:
  - Απλό
  - Το πεδίο αναζήτησης είναι O(1)
  - Ελέγξιμο (υπέρ ή κατά;)
- Κατά:
  - O server διατηρεί O(N) καταστάσεις
  - O server κάνει όλη την επεξεργασία και αποτελεί σημείο συμφόρησης, όσον αφορά την επίδοση
  - O server αποτελεί μοναδικό σημείο αποτυχίας
  - Δικαιώματα copyright

η μεταφορά αρχείων είναι αποκεντρωμένη, αλλά ο εντοπισμός του περιεχομένου είναι πολύ κεντρικός

P2P: Query flooding

- πλήρως κατανεμημένος κατάλογος
  - όχι κεντρικός server
- πρωτόκολλο public domain
- κάθε ομότιμος έχει κατάλογο των αρχείων που διαθέτει προς διανομή
- πολλοί clients υλοποιούν το πρωτόκολλο

υπερκείμενο δίκτυο: γράφος

- θεωρείται ότι υπάρχει ακμή μεταξύ των ομότιμων X και Y, αν υπάρχει σύνδεση TCP
- όλοι οι ενεργοί ομότιμοι και οι ακμές σχηματίζουν υπερκείμενο δίκτυο
- ακμή: νοητή (*όχι* φυσική) ζεύξη
- ένας ομότιμος συνδέεται συνήθως με < 10 overlay γείτονες

P2P Query flooding: Gutella

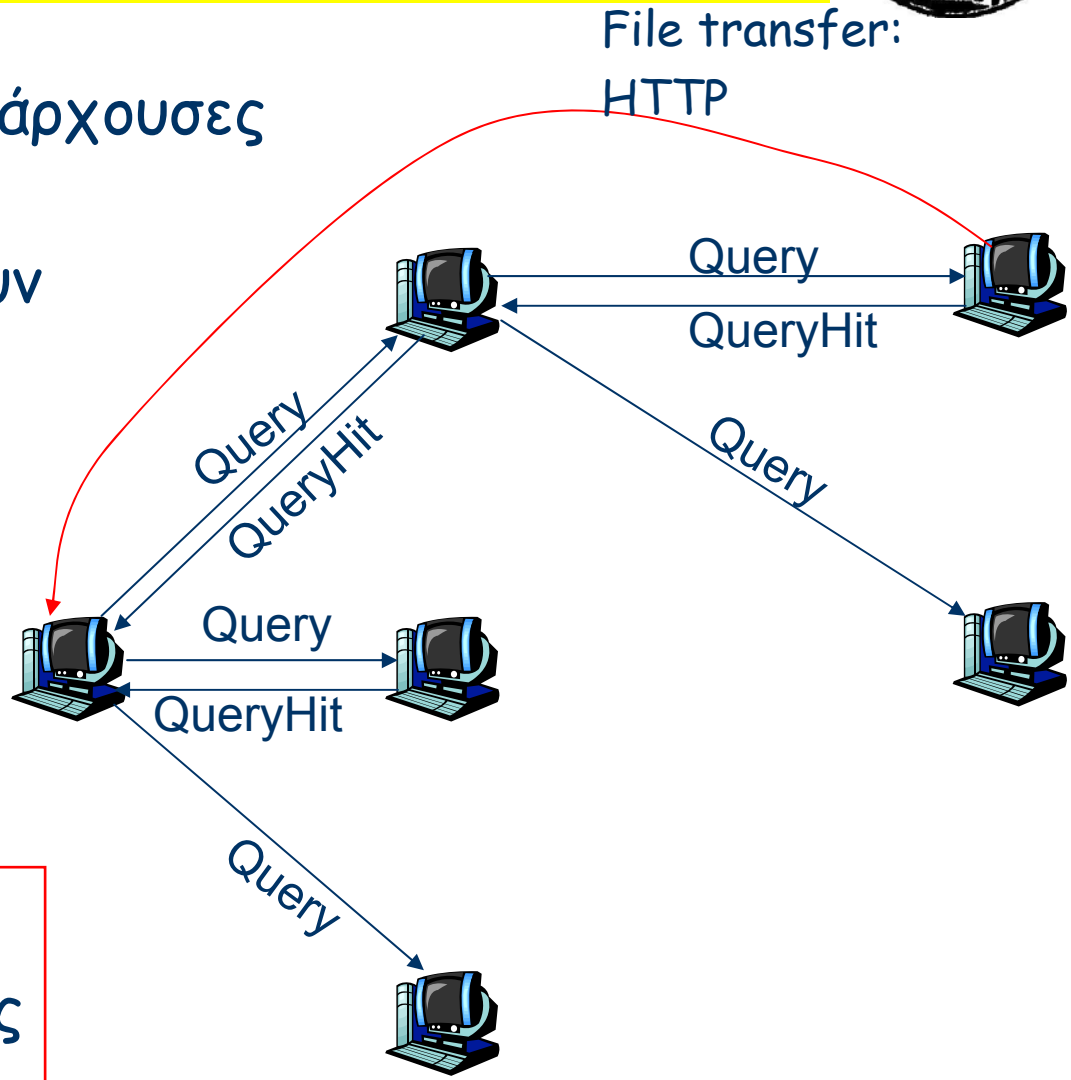
- Ιστορία του Gnutella
  - 2000: J. Frankel & T. Pepper ανακοινώνουν το Gnutella
  - Πολύ σύντομα μετά: πολλοί άλλοι clients (π.χ., Morpheus, Limewire, Bearshare)
  - 2001: βελτιώσεις στο πρωτόκολλο, π.χ., "ultra-peers"
- Query flooding
  - **Join**: έναρξη επικοινωνίας με λίγους κόμβους για να γίνουν γείτονες
  - **Publish**: δεν χρειάζεται!
  - **Search**: ερώτηση στους γείτονες, οι οποίοι ρωτούν τους γείτονές τους κοκ... και όταν/αν βρεθεί απάντηση στον αποστολέα
  - **Fetch**: κατέβασμα του αρχείου απ' ευθείας από τον ομότιμο κόμβο

Gnutella: ένταξη ομοτίμων

1. ο εντασσόμενος ομότιμος Χ πρέπει να βρει έναν άλλο ομότιμο στο δίκτυο Gnutella: χρησιμοποιεί λίστα υποψήφιων ομότιμων
2. ο Χ επιχειρεί διαδοχικές συνδέσεις TCP με υποψήφιους ομότιμους μέχρι να συνδεθεί με κάποιον άλλον ομότιμο τον Υ
3. μετά την εγκατάσταση σύνδεσης TCP μεταξύ Χ και Υ, ο Χ μπορεί να στείλει ένα μήνυμα "ping" στον Υ (με μετρητή βημάτων)
4. *πλημμύρα:* ο Υ προωθεί το μήνυμα "ping" στους overlay γείτονές του (που στη συνέχεια το προωθούν στους δικούς τους γείτονες....)
   - οι ομότιμοι που λαμβάνουν μήνυμα ping απαντούν στον Χ με μήνυμα "pong"
5. ο Χ λαμβάνει πολλά μηνύματα "pong" και μπορεί να εγκαταστήσει επιπρόσθετες συνδέσεις TCP με άλλους ομότιμους

Gnutella: Search

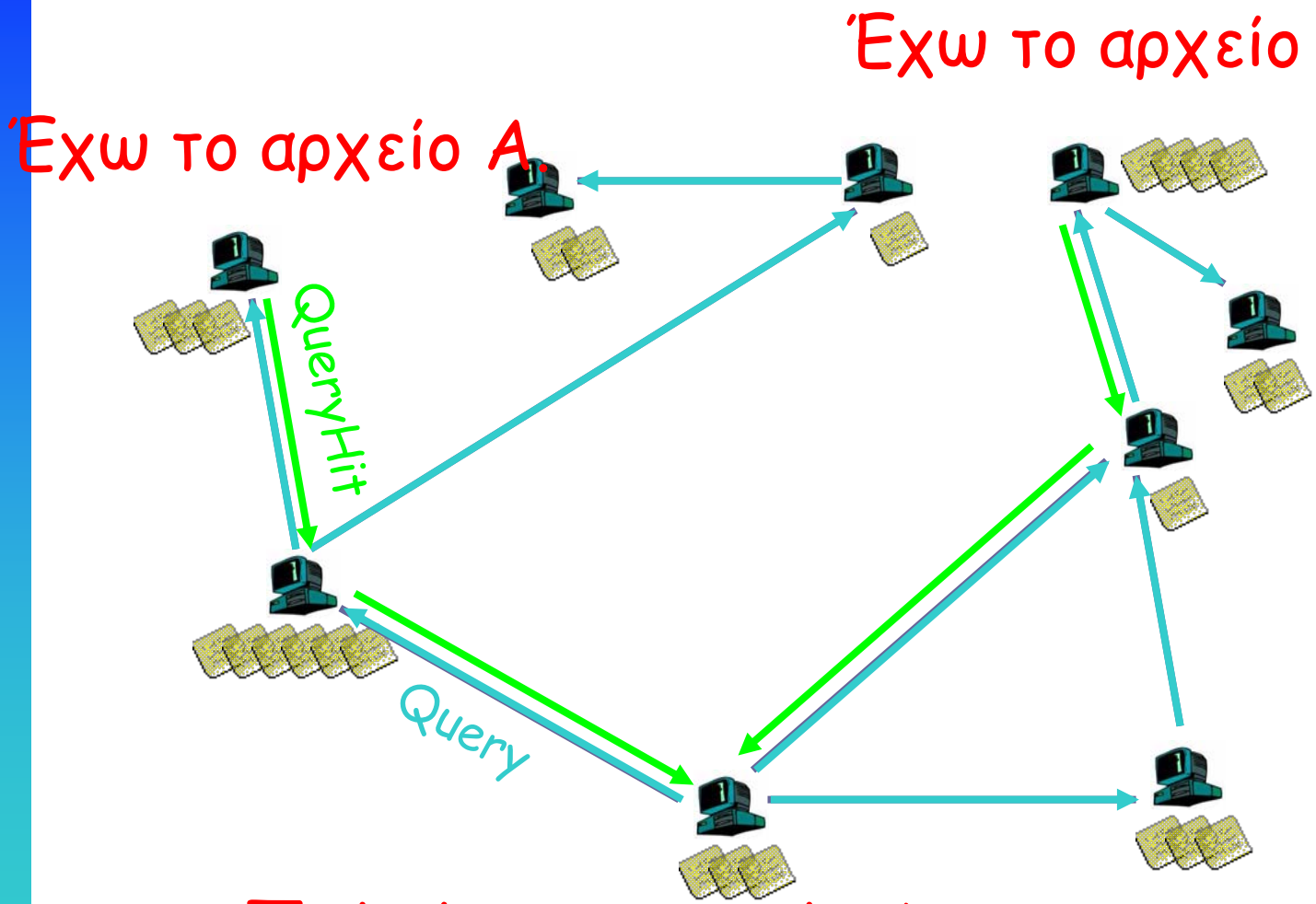
- μήνυμα Query στέλνεται από τις υπάρχουσες συνδέσεις TCP
- οι ομότιμοι προωθούν το μήνυμα Query
- QueryHit αποστέλλεται στην ανάστροφη διαδρομή



Κλιμάκωση:

περιορισμένης έκτασης πλημμύρα

Gnutella: Search



Gnutella: Υπέρ και κατά

- Πλεονεκτήματα
  - Πλήρως αποκεντρωμένο
  - Κατανεμημένο κόστος αναζήτησης
- Μειονεκτήματα
  - Το πεδίο αναζήτησης μπορεί να είναι πολύ μεγάλο
  - Ο χρόνος αναζήτησης μπορεί να είναι πολύ μεγάλος
  - Οι κόμβοι μπαινοβγαίνουν συχνά, ασταθές δίκτυο

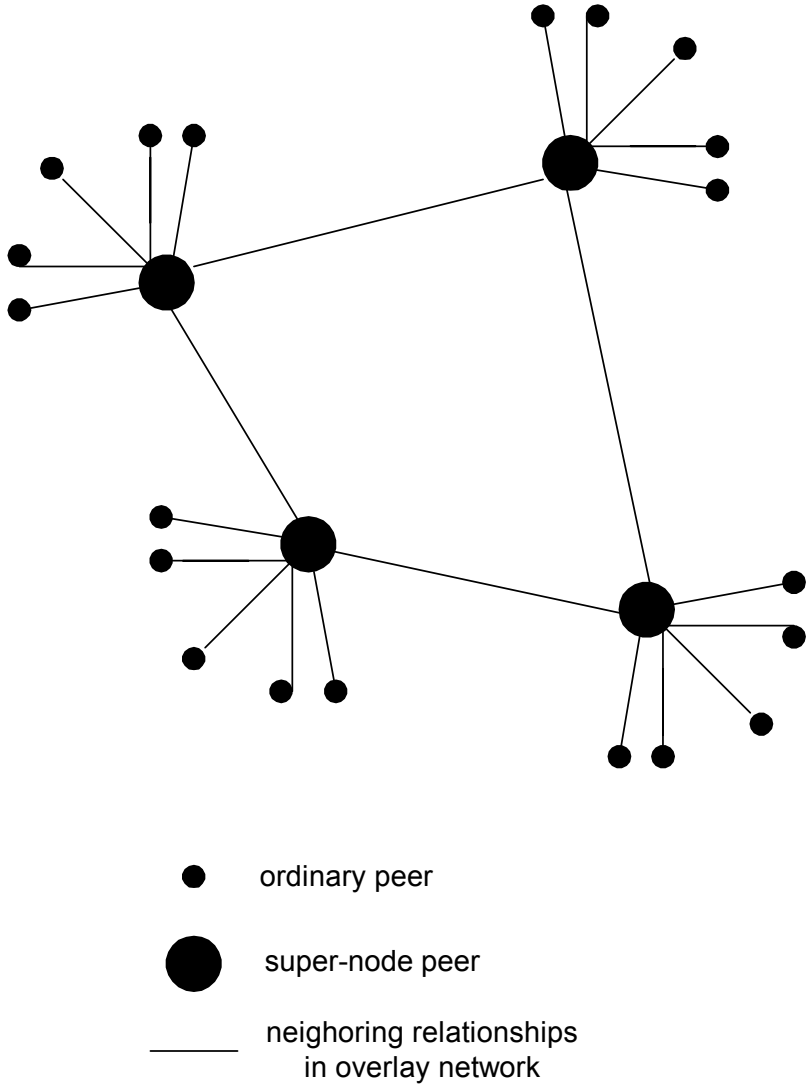
P2P Ιεραρχική διάρθρωση: KaZaA

- Ιστορία του KaZaA
  - 2001: δημιουργία από τον Ολλανδική εταιρία (Kazaa BV)
  - Ένα δίκτυο ονομαζόμενο FastTrack χρησιμοποιείται και από άλλους clients (Morpheus, giFT, κλπ.)
  - Τελικά, το πρωτόκολλο άλλαξε, ώστε οι άλλοι clients δεν μπορούν πλέον να μιλούν με αυτό

- Έξυπνο query flooding
  - **Join**: αρχικά, ο client επικοινωνεί με έναν super-node (και μπορεί αργότερα να γίνει τέτοιος)
  - **Publish**: ο client στέλνει λίστα αρχείων στον super-node του
  - **Search**: αποστολή query στον super-node και οι super-nodes πλημμυρίζουν μεταξύ τους τις queries
  - **Fetch**: λήψη αρχείου απευθείας από τον(τους) peer(s). Μπορεί να παίρνει ταυτόχρονα από πολλούς peers

KaZaA: Αξιοποίηση της ετερογένειας

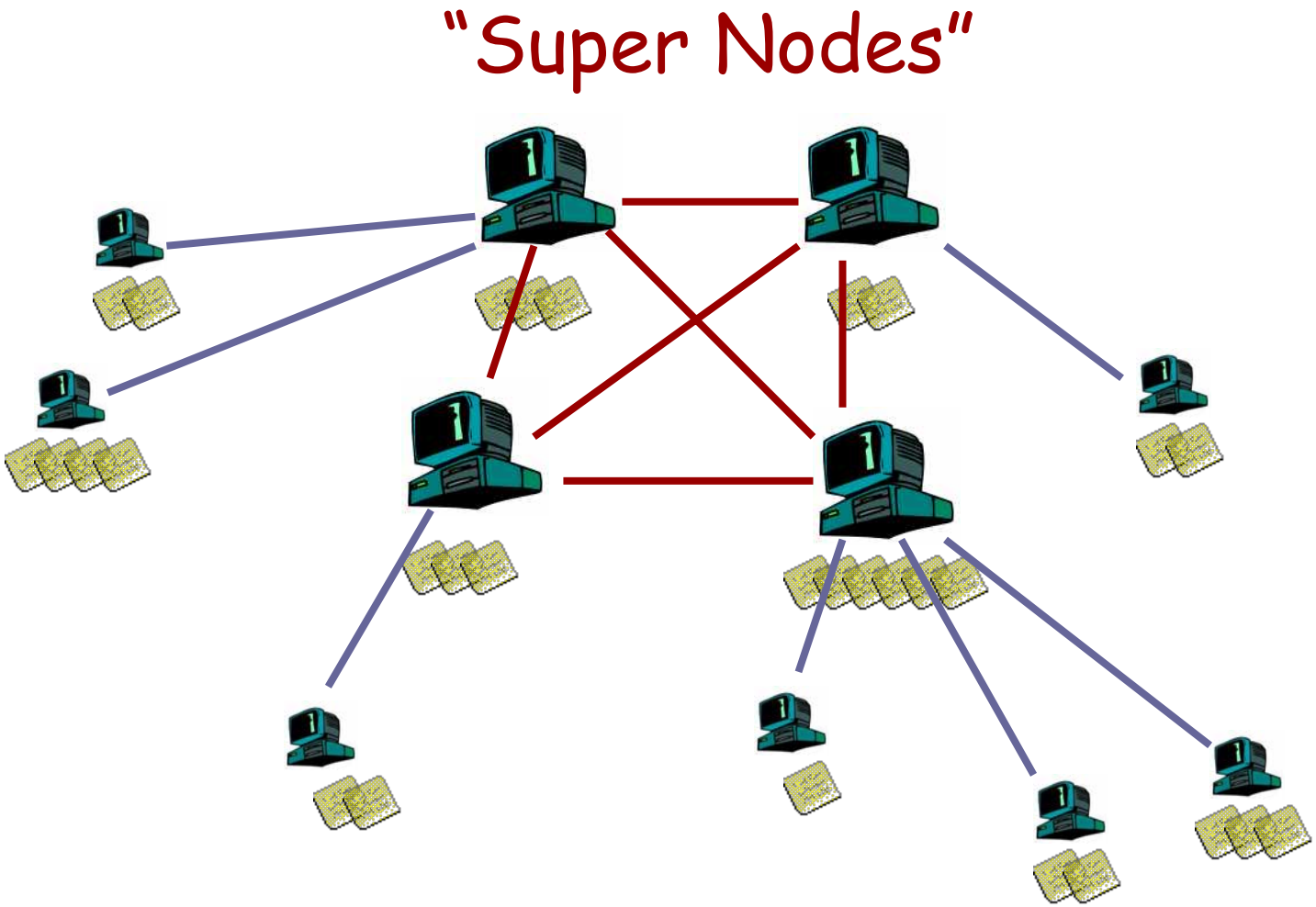
- Αρχιτεκτονική μεταξύ κεντρικού καταλόγου και query flooding
- Κάθε peer είναι είτε super-node ή assigned σε έναν super-node
  - σύνδεση TCP μεταξύ του peer και του super-node του
  - συνδέσεις TCP μεταξύ μερικών ζευγών super-nodes
- Ο super-node παρακολουθεί το περιεχόμενο των παιδιών του



KaZaA: Κίνητρα για Super-Nodes

- Συγχώνευση query
  - Πολλοί συνδεδεμένοι κόμβοι μπορεί να έχουν μόνο λίγα αρχεία
  - Η διάδοση μιας query σε έναν κοινό κόμβο μπορεί να παίρνει περισσότερο χρόνο από το να απαντήσει ο ίδιος ο super-node
- Ευστάθεια
  - Η επιλογή του super-node ευνοεί τους κόμβους που είναι πολύ ώρα παρόντες
  - Το πόση ώρα ήταν παρών ένας κόμβος είναι καλή πρόβλεψη για το πόση ώρα θα είναι παρών στο μέλλον

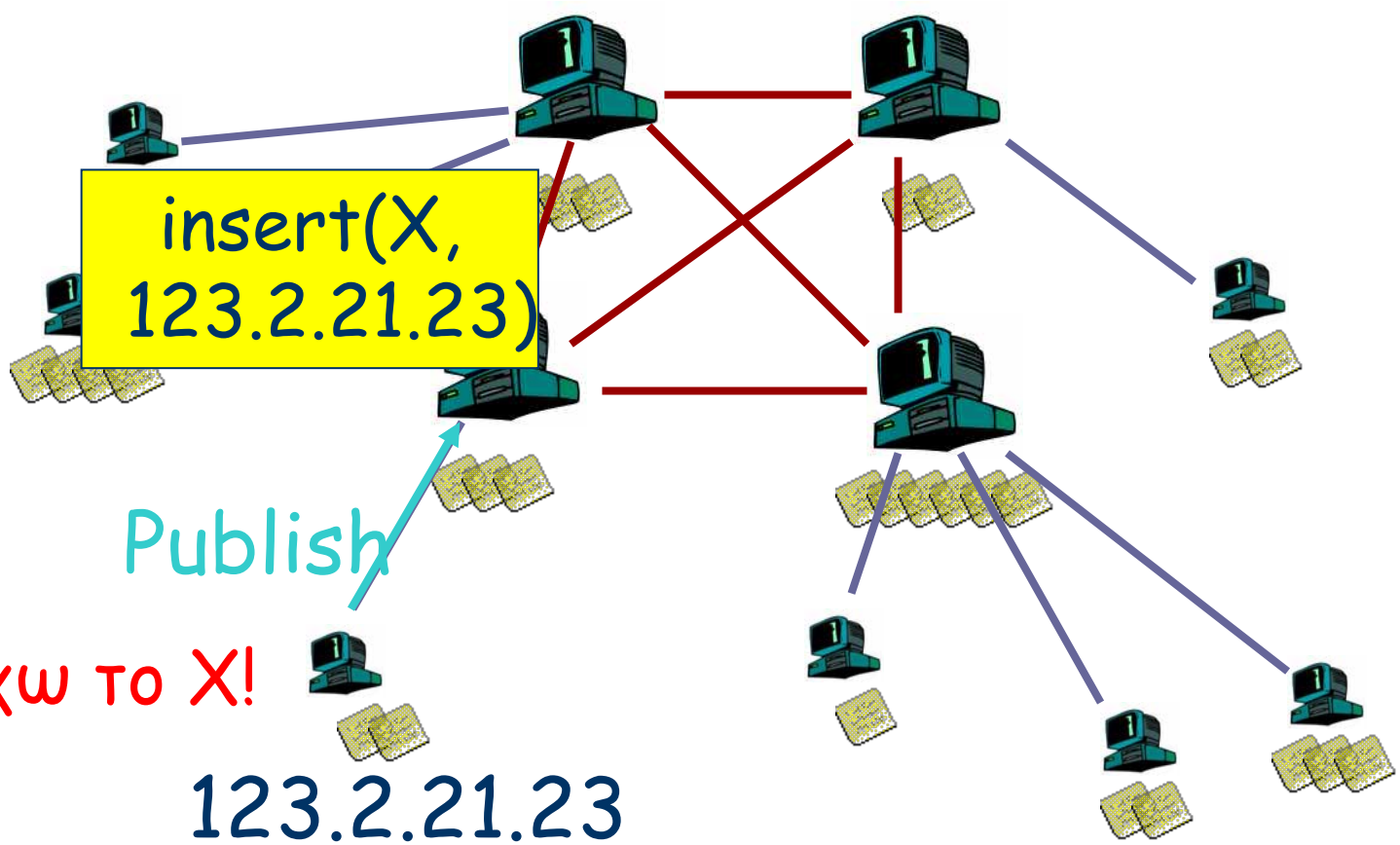
KaZaA: δομή δικτύου



KaZaA: Συντήρηση υπερκείμενου δικτύου

- Λίστα εν δυνάμει super-nodes περιλαμβάνεται στο κατέβασμα του software
- Ο νέος peer διατρέχει τη λίστα μέχρι να βρει super-node σε λειτουργία
  - Ο peer στέλνει ping σε (5-6) super-nodes της λίστας και συνδέεται με τον πρώτο που απαντάει
  - Συνδέεται και λαμβάνει πιο ενημερωμένη λίστα, με 200 εγγραφές
  - Οι super-nodes στην ενημερωμένη λίστα είναι "κοντά" στον peer
- Αν ο super-node κλείσει, ο peer ανατρέχει στην ενημερωμένη λίστα και βρίσκει νέο super-node

KaZaA: Publish



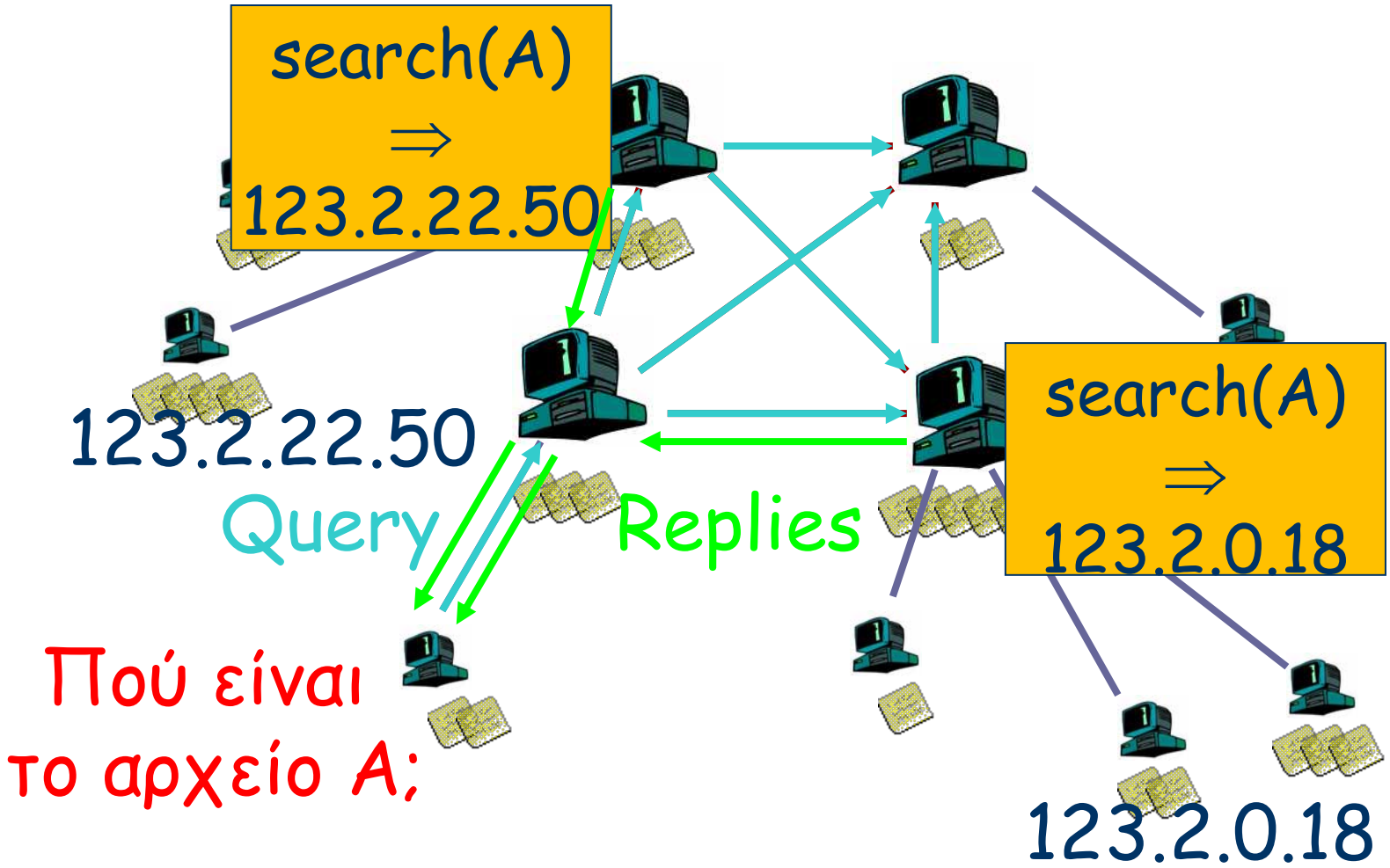
KaZaA: Metadata

- Όταν ένας κοινός κόμβος συνδέεται σε super-node, ανεβάζει τα metadata του
- Για κάθε αρχείο:
  - File name
  - File size
  - Content Hash (UUHash, MD5, SHA-1)
  - File descriptors: χρησιμοποιούνται για ταίριασμα των keyword κατά την αναζήτηση
- Content Hash:
  - Όταν ο peer A επιλέγει αρχείο του peer B, ο A στέλνει το ContentHash στο HTTP request
  - Αν το κετέβασμα κάποιου αρχείου αποτύχει (ολοκληρωθεί μερικώς), χρησιμοποιείται το ContentHash για την αναζήτηση νέου αντιγράφου του υπόψη αρχείου

KaZaA: Queries

- Ένας κόμβος στέλνει πρώτα query στον super-node
  - Ο Super-node απαντά με matches
  - Αν βρεθούν x matches, τελειώνει
- Αλλιώς, ο super-node προωθεί το query σε υποσύνολο από super-nodes
  - Αν βρεθούν συνολικά x matches, τελειώνει
- Αλλιώς, το query προωθείται περαιτέρω
  - Πιθανώς από τον αρχικό super-node, παρά διαδοχικά

KaZaA: search



KaZaA: Fetching -Παραλληλισμός

- Αν το αρχείο βρεθεί σε πολλούς κόμβους, ο χρήστης μπορεί να επιλέξει παράλληλο κατέβασμα
- Τα ίδια αντίγραφα αναγνωρίζονται από το ContentHash
- Η επέκταση byte range retrieval του HTTP για τη ζήτηση διαφορετικών τμημάτων του αρχείου από διαφορετικούς κόμβους
  - Server: Accept-Ranges: bytes
  - Client: Range: bytes=0-500, 5000-
- Εναλλακτική λύση: Erasure codes
- Αυτόματη επανεύρεση, όταν ο server peer σταματήσει να στέλνει το αρχείο
  - Το ContentHash χρησιμοποιείται για την αναζήτηση νέου αντιγράφου του αρχείου

KaZaA: Υπέρ και κατά

- Υπέρ:
  - Προσπαθεί να λάβει υπόψη την ετερογένεια:
    - Bandwidth
    - Host Computational Resources
    - Host Availability (?)
  - Ισχυρισμοί ότι λαμβάνει υπόψη την τοπικότητα
- Κατά:
  - Εύκολα καταστρατηγούμενοι μηχανισμοί
  - Δεν υπάρχει ακόμα εξασφάλιση ως προς τον χώρο αναζήτησης και τον χρόνο αναζήτησης

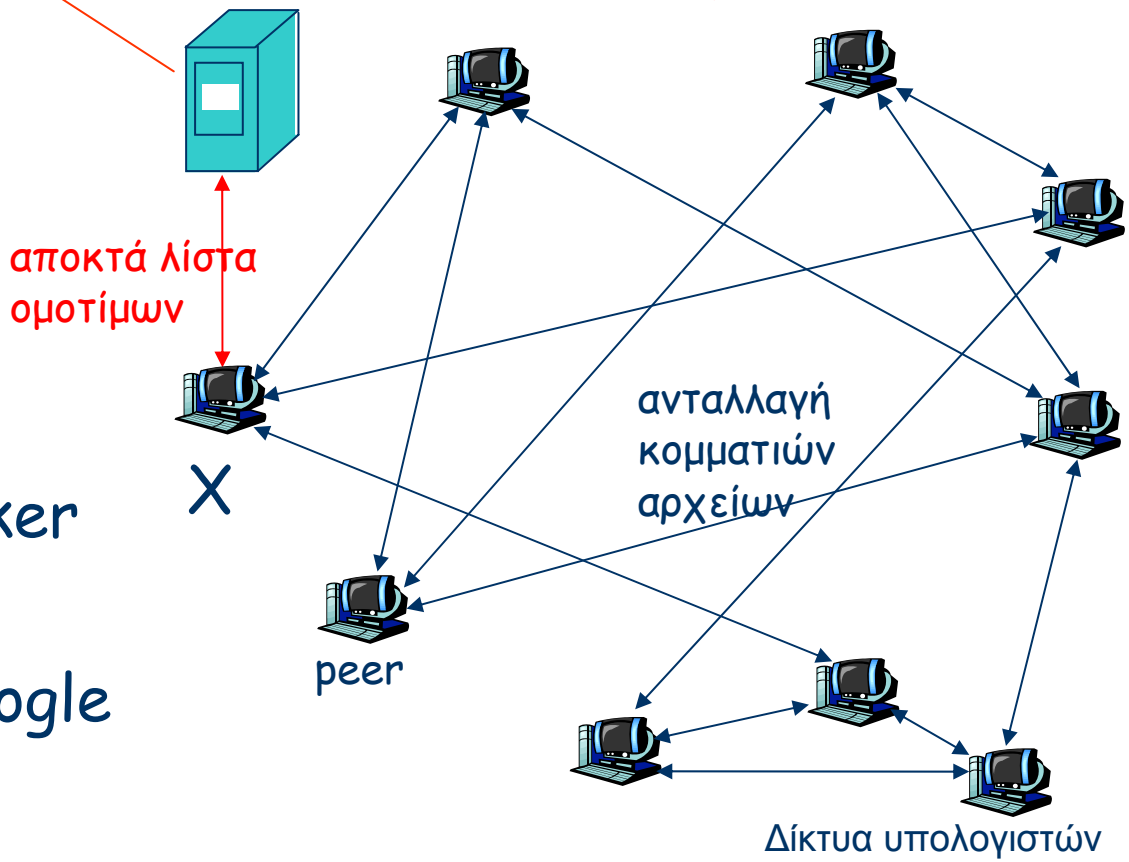
Δίκτυα P2P: BitTorrent

- Ιστορία και κίνητρα για το BitTorrent
  - 2002: B. Cohen εισήγαγε το BitTorrent
  - Εστιάζει στην αποτελεσματική προσκόμιση (*fetching*), όχι στην αναζήτηση (searching)
    - Διανομή του ίδιου αρχείου σε πολλούς peers
    - Ένας εκδότης, πολλοί host που κατεβάζουν
  - Παρεμπόδιση ελεύθερου κατεβάσματος

Δίκτυα P2P: BitTorrent

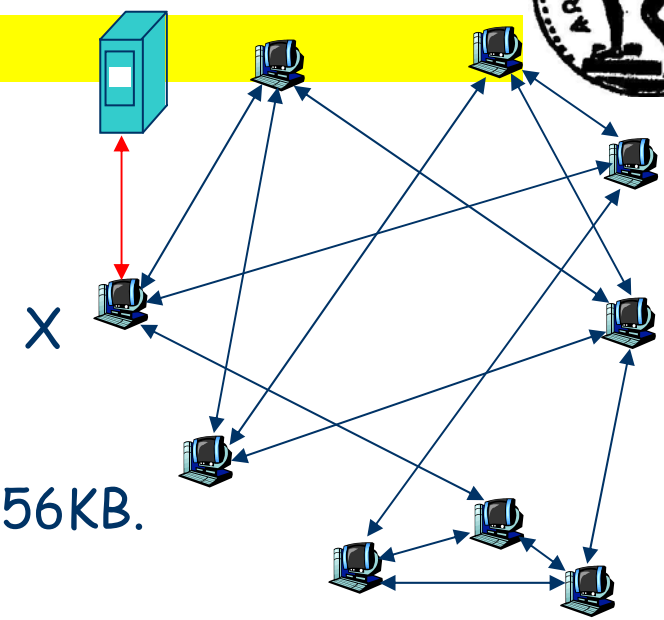
tracker: ανιχνεύει ομότιμους που συμμετέχουν στο torrent

torrent: ομάδα ομότιμων που ανταλλάσσουν μεγάλα κομμάτια ενός αρχείου (τυπικό μέγεθος 256KB)



- Κεντρικοποιημένο δίκτυο, λόγω tracker
- .torrent αρχεία
- Αναζήτηση στο Google για .torrents

BitTorrent



- αρχείο διαιρούμενο σε κομμάτια των 256KB.
- ομότιμος εντασσόμενος στο torrent:
  - δεν έχει κομμάτια, αλλά τα συγκεντρώνει με τον χρόνο
  - εγγράφεται στον tracker για να λάβει λίστα ομοτίμων, συνδέεται σε υποσύνολο ομοτίμων ("γείτονες")
- ενώ κάνει download, ο ομότιμος κάνει upload κομμάτια σε άλλους ομότιμους.
- ομότιμοι μπορεί να μπαίνουν και να βγαίνουν
- μόλις ένας ομότιμος έχει όλο το αρχείο, μπορεί (ατομιστικά) να φύγει ή (αλτρουϊστικά) να μείνει

BitTorrent: Σύνοψη λειτουργίας

- Μαζική παρουσία
  - **Join**: επικοινωνία με τον "tracker" server, λήψη λίστας από peers.
  - **Publish**: Τρέξε έναν tracker server.
  - **Search**: Out-of-band. Π.χ., χρήση του Google για να βρεθεί tracker για το αρχείο που θέλουμε.
  - **Fetch**: Κατέβασμα κομματών του αρχείου από τους σχετικούς peers. Ανέβασμα κομματιών που έχεις για αυτούς.

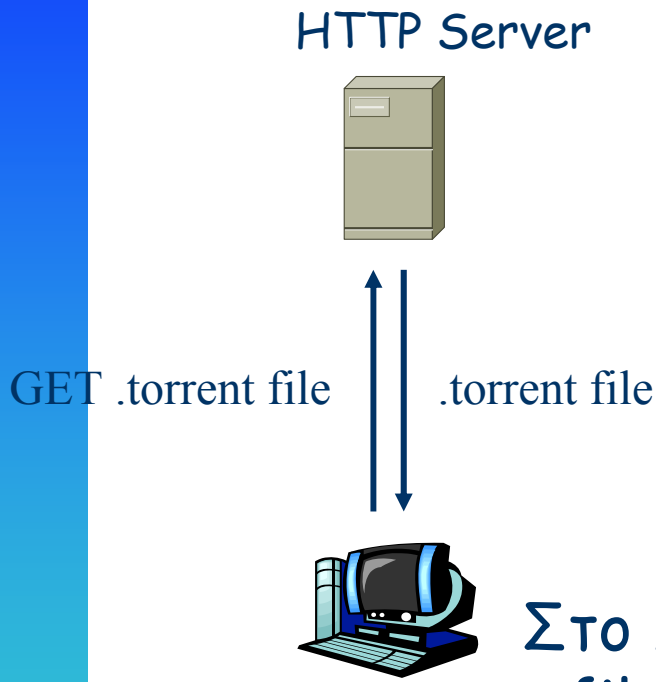
BitTorrent: Tracker

- Κόμβος υποδομής
  - Ανιχνεύει peers που συμμετέχουν στο torrent
- Οι peers γράφονται στον tracker
  - Ο peer γράφεται μόλις εμφανιστεί
  - Ο peer πληροφορεί περιοδικά τον tracker ότι είναι ακόμη παρών
- Ο tracker επιλέγει peers για κατέβασμα
  - Επιστρέφει μια τυχαία ομάδα από peers μαζί με τις IP διευθύνσεις τους
  - Οπότε, ο νέος peer γνωρίζει σε ποιον θα απευθυνθεί για δεδομένα

BitTorrent: Chunks

- Ένα μεγάλο αρχείο διαιρείται σε μικρότερα κομμάτια (chunks)
  - Κομμάτια σταθερού μεγέθους
  - Τυπικό μέγεθος κομματιού 256 Kbytes
- Επιτρέπονται ταυτόχρονες μεταφορές
  - Κατέβασμα κομματιών από διάφορους γείτονες
  - Αποστολή κομματιών σε άλλους γείτονες
- Πληροφόρηση για τα κομμάτια που έχουν οι γείτονες
  - Περιοδικές ερωτήσεις προς αυτούς για τη λίστα
- Το αρχείο αποκτάται όταν καταβαστούν όλα τα κομμάτια

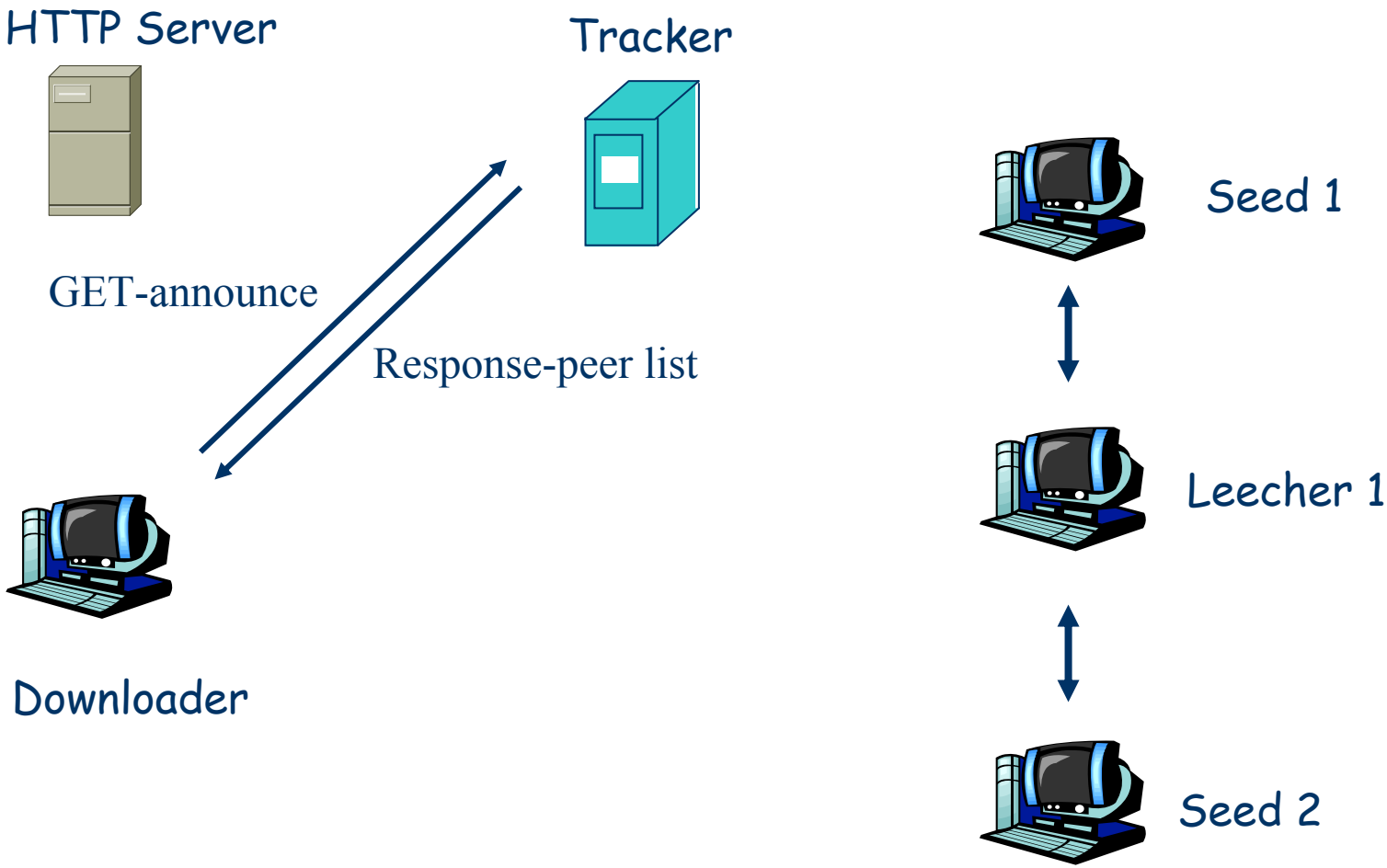
BitTorrent: εύρεση .torrent αρχείων



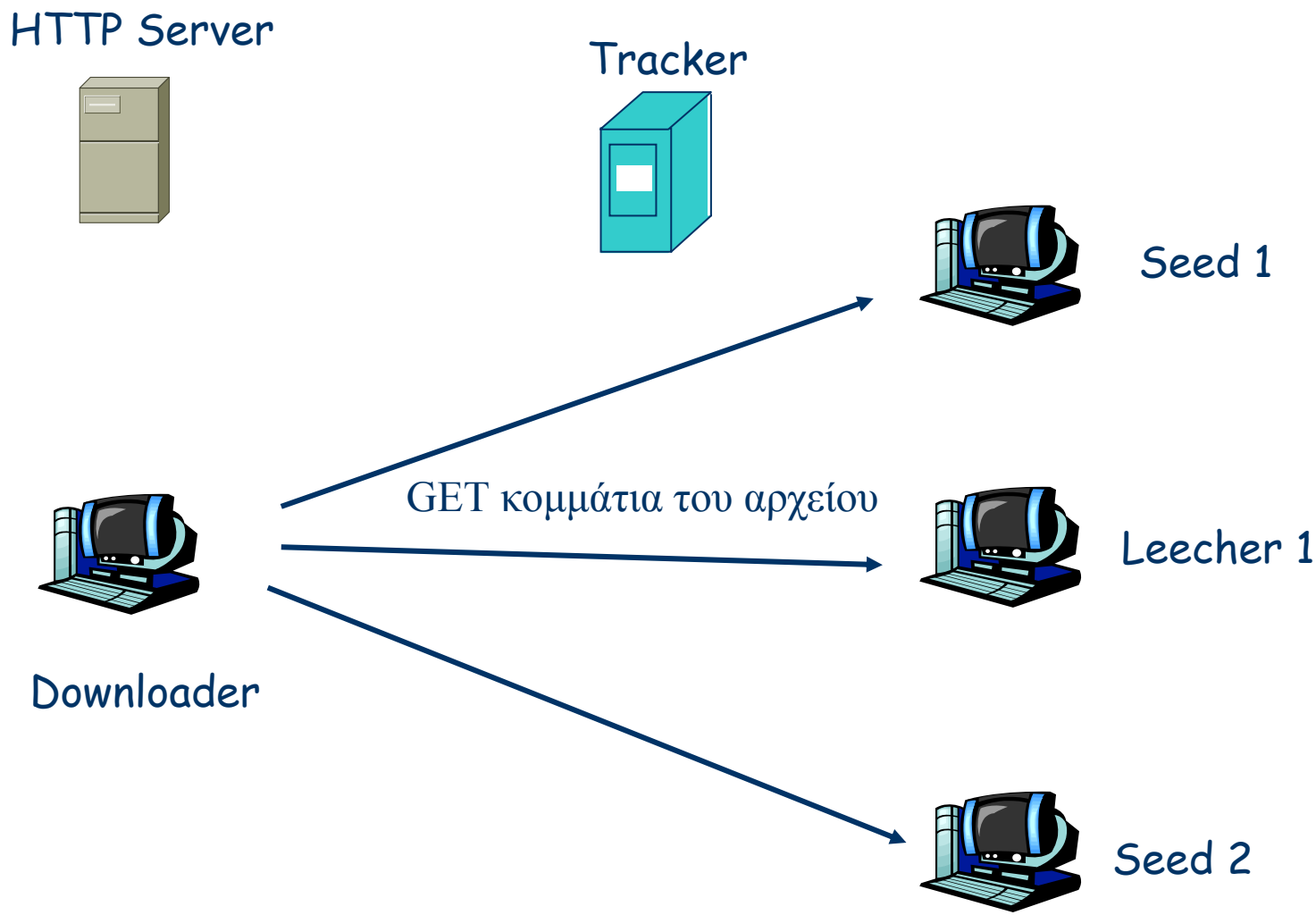
Στο .torrent file περιέχονται:

- file size
- file name
- hash του αρχείου (SHA1)
- URL του tracker

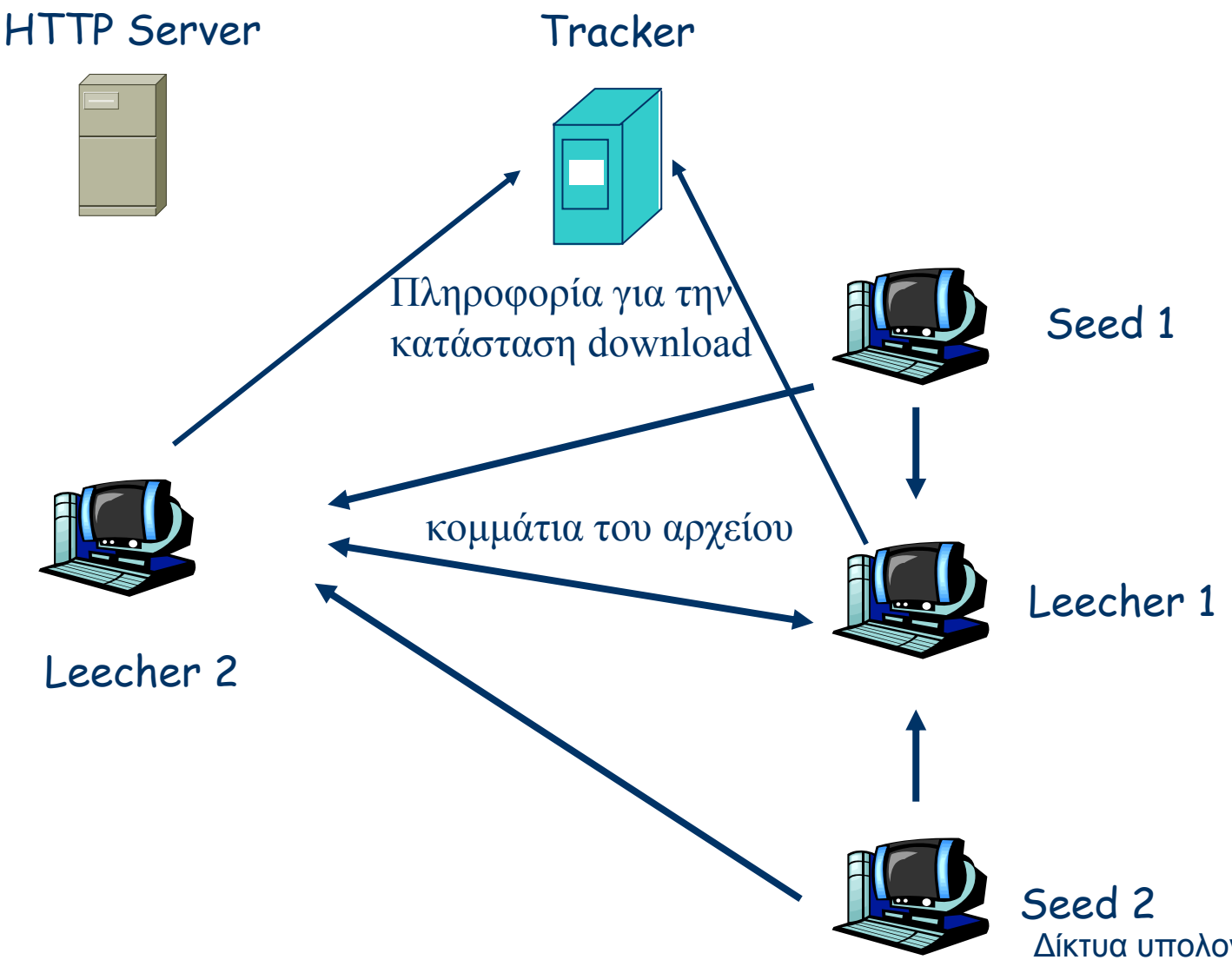
BitTorrent: εύρεση λίστας ομοτίμων



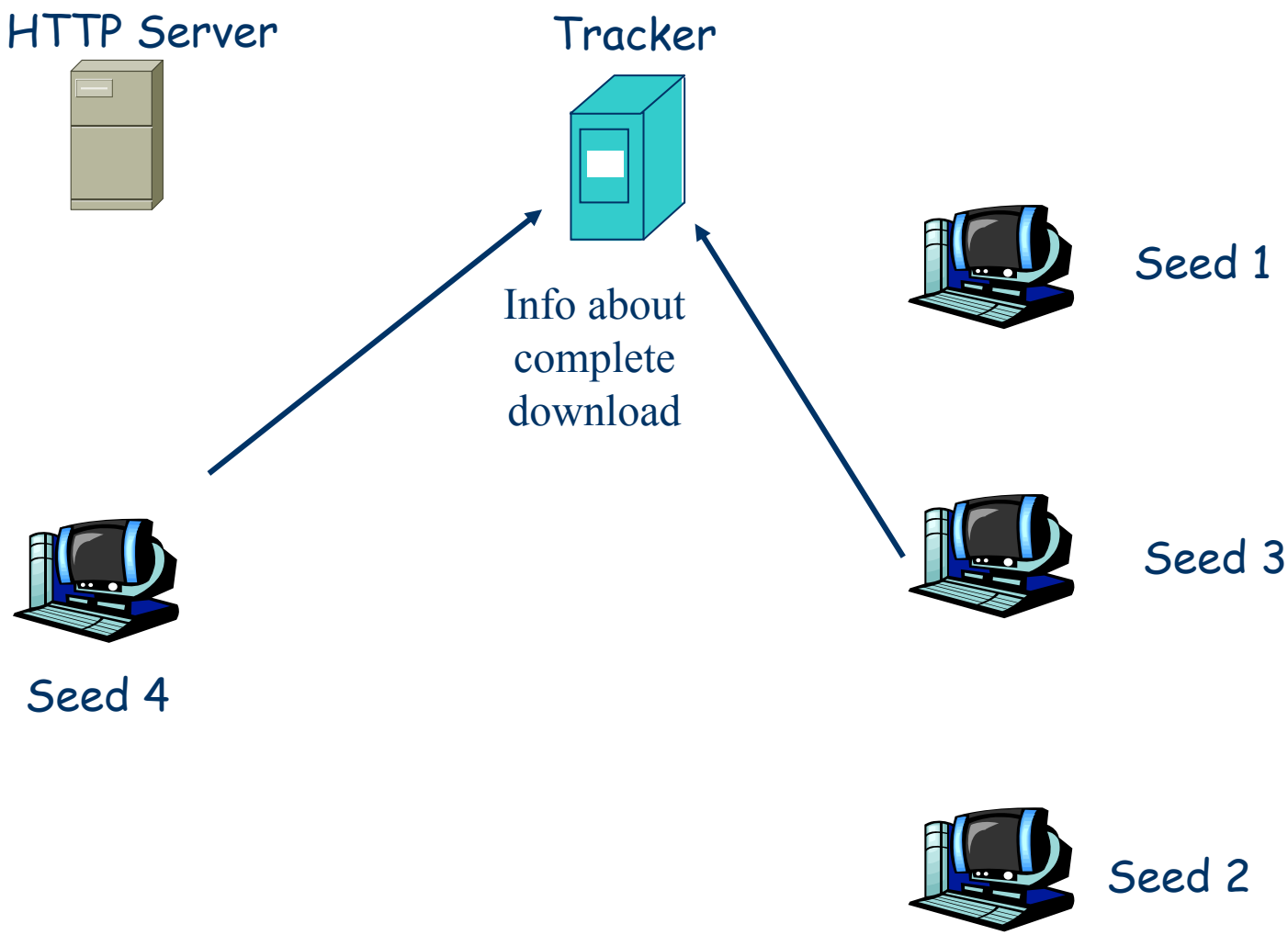
BitTorrent: αναζήτηση κομματιών



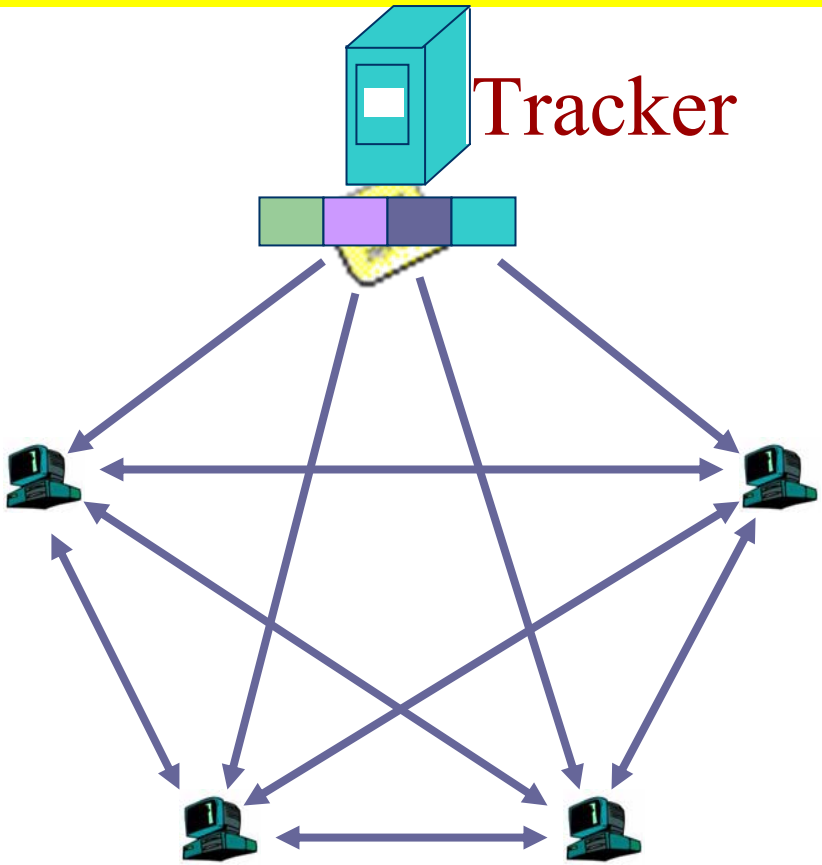
BitTorrent: κατέβασμα κομματιών



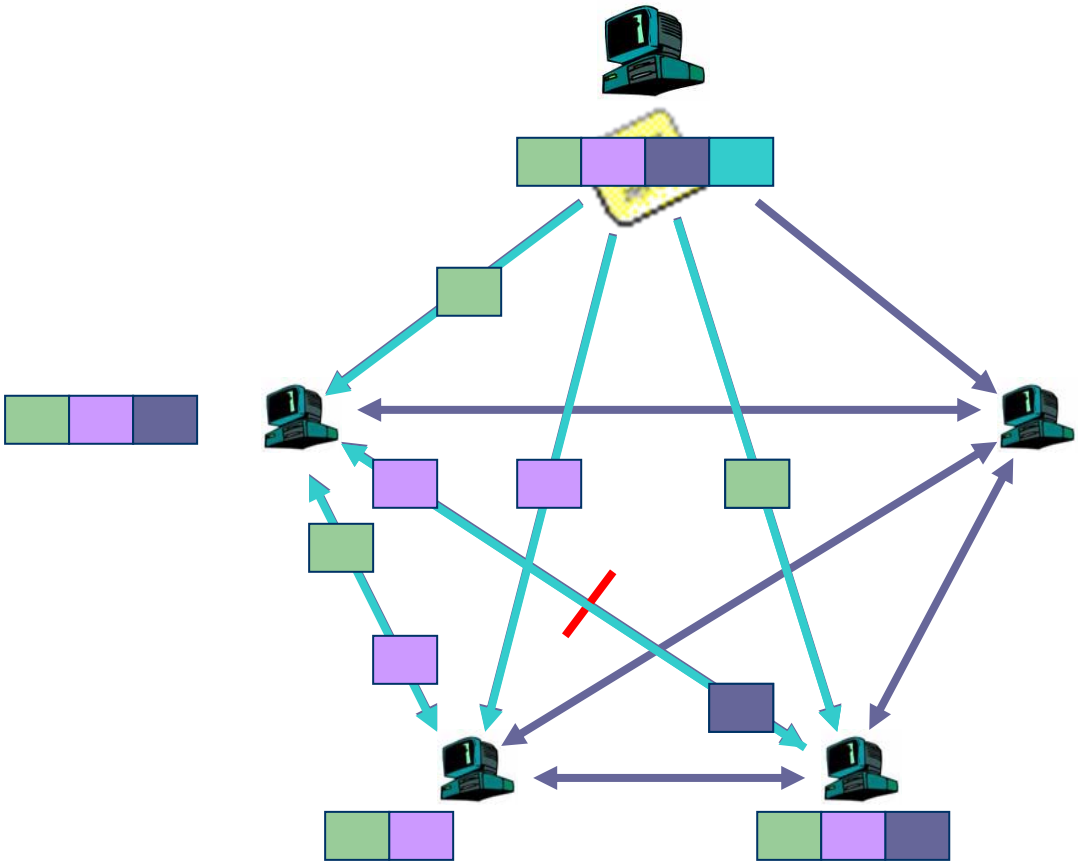
BitTorrent: πληροφόρηση κατάστασης



BitTorrent: Join/Publish



BitTorrent: Fetch



BitTorrent: σειρά ζήτησης κομματιών

- Ποια κομμάτια να ζητηθούν;
  - Μπορούν κα κατεβούν με τη σειρά, όπως π.χ. κάνει ένας HTTP client;
- Πρόβλημα: πολλοί peers έχουν τα αρχικά κομμάτια
  - Οι peers έχουν λίγα να μοιραστούν μεταξύ τους
  - Περιορισμός της κλιμάκωσης του συστήματος
- Πρόβλημα: κατά σύμπτωση κανείς δεν έχει σπάνια κομμάτια
  - π.χ., τα κομμάτια του τέλους του αρχείου
  - Περιορισμός της ικανότητας περάτωσης του κατεβάσματος
- Λύσεις: τυχαία επιλογή και τα σπάνια πρώτα

BitTorrent: πρώτα το σπάνιο κομμάτι

- Ποια κομμάτια να ζητηθούν πρώτα;
  - Τα κομμάτια με τα λιγότερα διαθέσιμα αντίγραφα
  - π.χ., το σπανιότερο κομμάτι πρώτο
- Ωφέλη για τον peer
  - Αποφυγή σταματήματος κατεβάσματος όταν μερικοί peers αποχωρήσουν
- Ωφέλη για το σύστημα
  - Αποφυγή σταματήματος κατεβάσματος για όλους τους peers που περιμένουν ένα αρχείο
  - Ισοστάθμιση φορτίου εξισώνοντας τους αριθμούς των αντιγράφων των τεμαχίων

BitTorrent: παρεμπόδιση free-riding

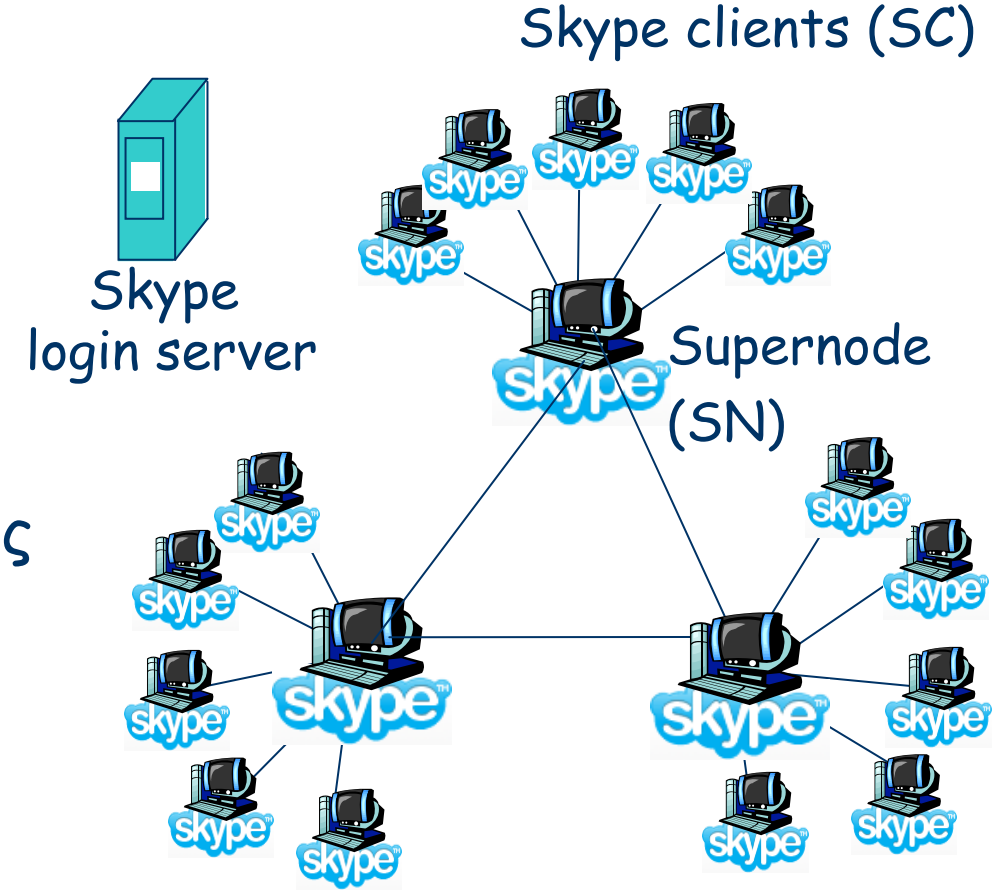
- Μεγάλο πλήθος χρηστών είναι free-riders
  - Οι περισσότεροι δεν μοιράζονται αρχεία και δεν απαντούν σε ερωτήσεις
  - Άλλοι περιορίζουν τον αριθμό των συνδέσεων ή την ταχύτητα ανεβάσματος
- Λίγοι peers λειτουργούν ουσιατσικά ως servers
  - Λίγοι συνεισφέρουν στο κοινό καλό
  - Καθίστανται hubs που βασικά δρουν ως servers
- Το BitTorrent εμποδίζει το free riding
  - Επιτρέπει τους πιο γρήγορους peers να κατεβάζουν από άλλον peer
  - Περιστασιακά επιτρέπει σε μερικούς free loaders να κατεβάζουν

Bit-Torrent: παρεμπόδιση free-riding

- Ο peer έχει περιορισμένο εύρος ζώνης ανεβάσματος
  - Και πρέπει να το μοιράζει σε πολλούς peers
- Προτεραιότητα ανάλογα με εύρος ζώνης ανεβάσματος
  - Υποστήριξη των γειτόνων που ανεβάζουν με τον υψηλότερο ρυθμό
- Επιβράβευση των 4 πρώτων γειτόνων
  - Μετράει την ταχύτητα κατεβάσματος για κάθε γείτονα
  - Ανταποκρίνεται στέλνοντας στους 4 πρώτους peers
  - Επαναϋπολογίζει τους top4 κάθε 10 sec
- Οπτιμιστική επανεπιλογή
  - Δοκιμάζει τυχαία έναν νέο γείτονα κάθε 30 sec
  - Έτσι, ο νέος γείτονας έχει μια ευκαιρία να συμμετάσχει

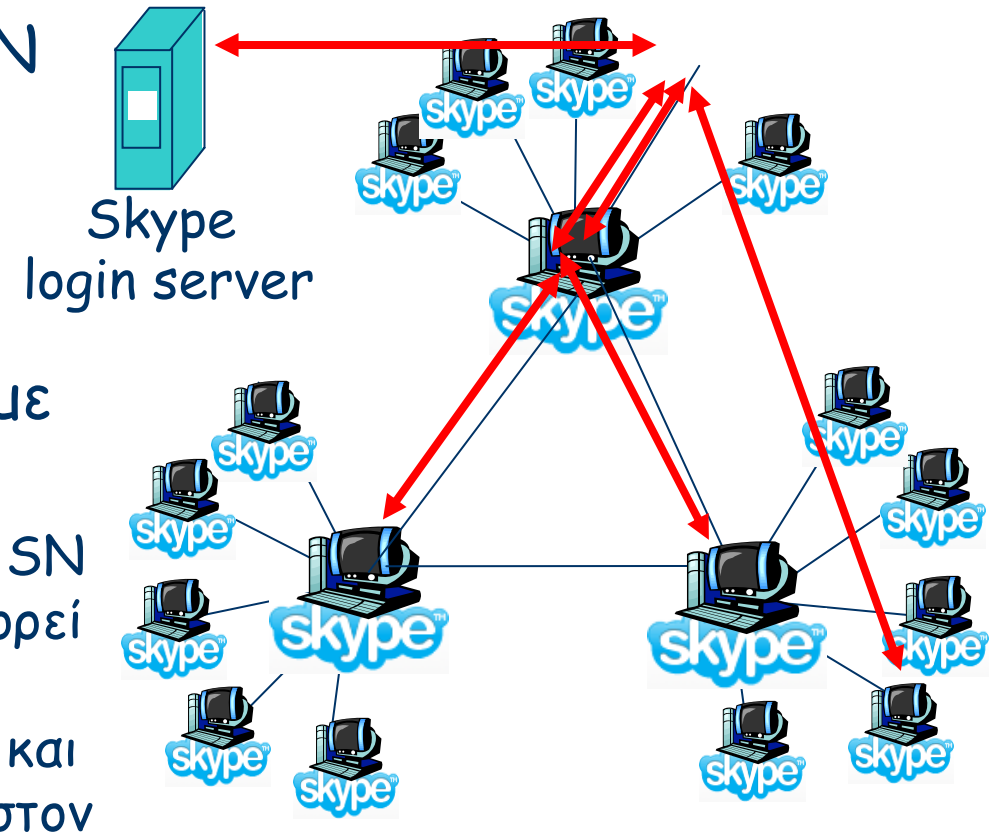
P2P: Skype

- P2P (pc-to-pc, pc-to-phone, phone-to-pc) Voice-Over-IP (VoIP) εφαρμογή
  - επίσης IM
- proprietary πρωτόκολλο στρώματος εφαρμογής
- ιεραρχική διάρθρωση



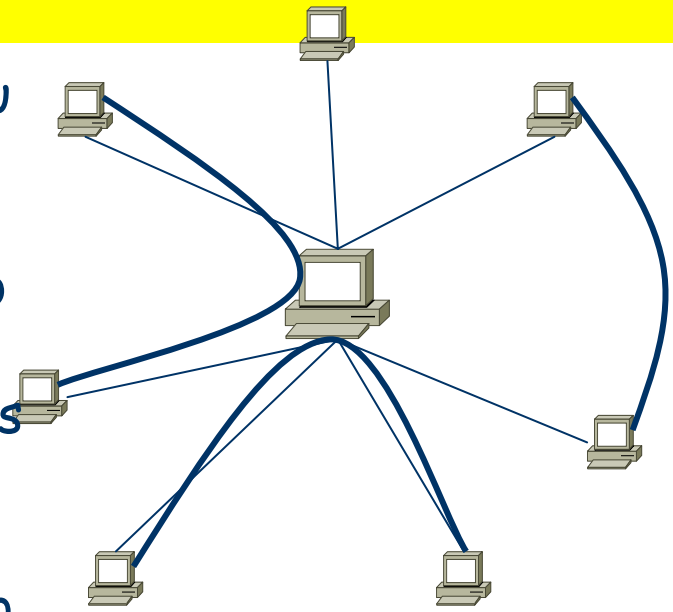
Skype: πραγματοποίηση κλήσης

- Ο χρήστης ξεκινά το Skype
- Ο SC εγγράφεται στον SN
  - λίστα από bootstrap SNs
- Ο SC κάνει log in (πιστοποίηση αυθεντικότητας
- Κλήση: SC καλεί τον SN με την ID του καλούμενου
  - SN επικοινωνεί με άλλους SN (άγνωστο πρωτόκολλο, μπορεί πλημμύρα) για να βρει τη διεύθυνση του καλούμενου και επιστρέφει την διεύθυνση στον SC
- Ο SC επικοινωνεί άμεσα με τον καλούμενο πάνω από TCP



Διέλευση μέσω NAT και Firewall

- Αν ο καλών είναι πίσω απο NAT, η σηματοδοσία κλήσης (TCP) προωθείται από κάποιον κόμβο που έχει public IP address



- Αν και ο καλών και ο καλούμενος έχουν public IP address, η σηματοδοσία της κλήσης (TCP) και η κίνηση φωνής (UDP) πραγματοποιείται απευθείας μεταξύ τους

- Αν είτε ο καλών είτε ο καλούμενος ή και οι δύο είναι πίσω από NAT η κίνηση φωνής (UDP) προωθείται μέσω του ίδιου κόμβου

- Αν και ο καλών και ο καλούμενος είναι πίσω από NAT και από UDP-restricted firewall, τότε η κίνηση σηματοδοσίας και φωνής προωθούνται μέσω ενός άλλου κόμβου με TCP