Reţele şi aplicaţii P2P

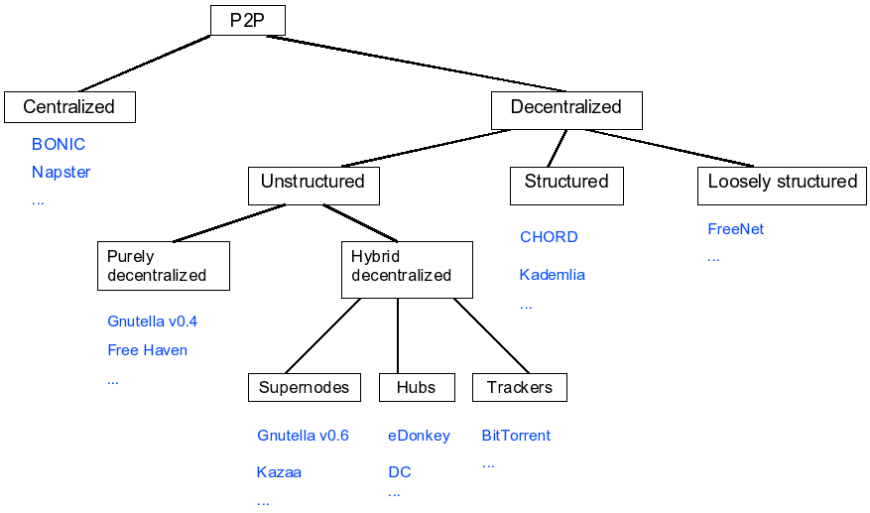
P2P

- File sharing (Napster, Gnutella, Kazaa, Bittorrent,...)
- Multiplayer games (Unreal Tournament, DOOM)
- Aplicaţii colaborative (ICQ, shared whiteboard)
- Distributed computing (Seti@home, platforma BOINC)
- Comunicaţii (IM, Skype)
- Reţele Ad-hoc
- Soft colaborativ (e.g., Magi, Groove, Jabber)
- Platforme P2P (e.g., JXTA) pentru P2P Computing

Clasificare P2P

- P2P Hibrid – conservă o parte din arhitectura tradiţională client/server; un server central stochează indecşii şi e folosit ca link între clienţi
  - ex: Napster
- P2P Nestructurat – nu există control asupra topologiei şi a plasamentului fişierelor
  - ex: Gnutella, Morpheus, Kazaa, etc
- P2P Structurat – topologia şi plasamentul fişierelor sînt bine controlate
  - ex: Chord, CAN, Pastry, Tornado, etc

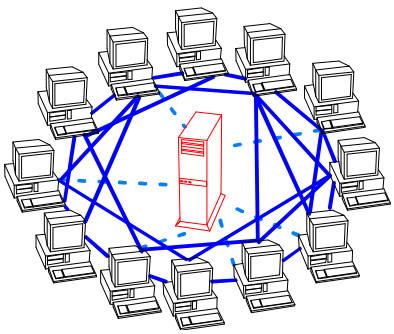
Clasificare P2P



Aplicaţii şi protocoale *File Sharing*

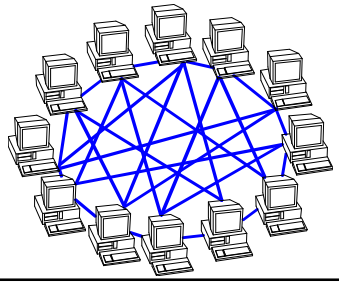
Reţele de *File Sharing* P2P nestructurate

- **P2P centralizat**
  - Toate nodurile conectate la o entitate centrală
  - Nodurile se conectează unul la altul pentru transferul fişierelor
  - Nodul central este strict necesar pt. funcţionare
  - Nodul central are funcţii de indexare şi *lookup-table*
  - Exemple: Napster



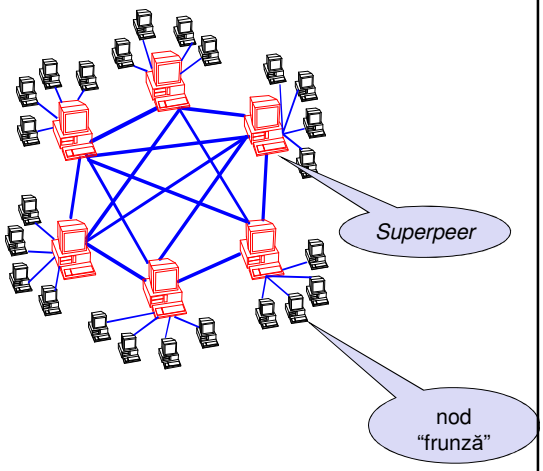
Reţele de *File Sharing* P2P nestructurate

- **P2P "pur"**
  - orice nod poate fi eliminat fără a pierde din funcţionalitate
  - Nu există nod central
  - Stabilirea conexiunilor între *peers* este aleatoare
  - Exemple: Gnutella, FreeNet



Reţele de *File Sharing* P2P nestructurate

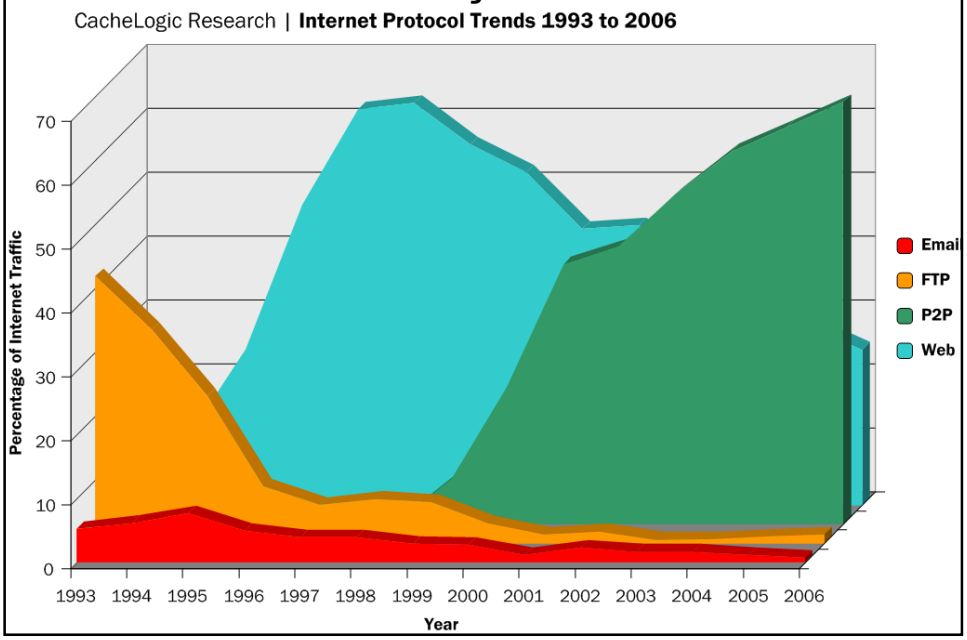
- **P2P hibride**
  - Comparativ cu P2P "pure", se introduce un nivel ierarhic suplimentar
  - există un proces prin care se aleg *superpeers* (supernoduri)
  - Grad de interconectare mare a Superpeers: (grad>>20, în funcţie de dimensiunea reţelei)
  - nodurile "frunze" (normale), conectate la un nr. limitat de Superpeers (grad <7)
  - Exemple: KaZaA (superpeer s.n. supernode)



File-sharing P2P

- Napster
- Gnutella
- Kazaa
- Bittorrent

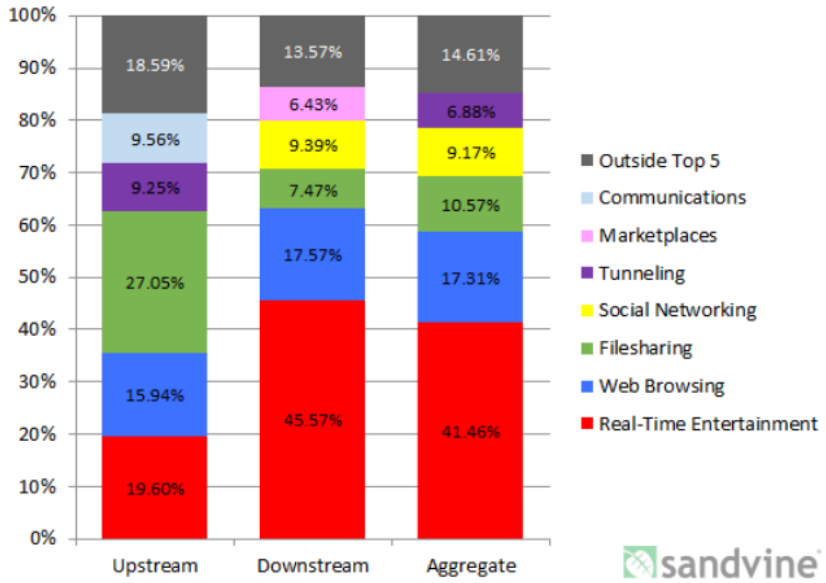
Evoluţie P2P



Evoluţie P2P - SUA

- 2007-2015: trend modificat de apariția *video streaming* (Youtube, Netflix,...), social media (Facebook, Instagram, WhatsApp,...), Windows update etc.
- 2015: traffic P2P în America de Nord redus la 5% din total, *Video Streaming:* 70%

Evoluţie P2P - Europa (trafic fix)



Sursa: Sandvine Global Internet Phenomena, 2015; *Realtime=Video & Audio streaming*

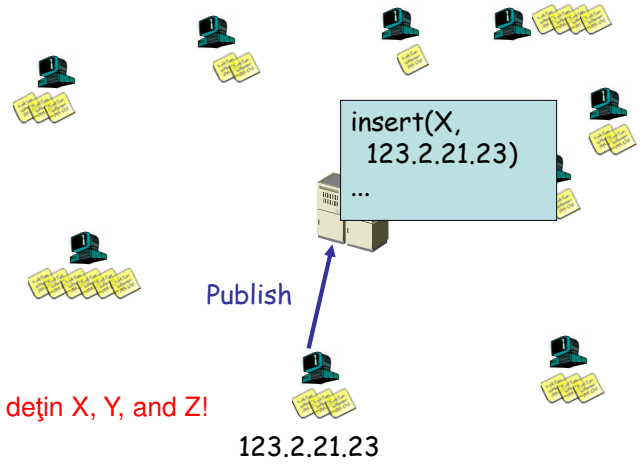
Napster

- Istoria Napster: ascensiunea
  - ianuarie 1999: Napster version 1.0
  - Mai 1999: compania Napster înfiinţată
  - Septembrie 1999: primele acţionări în judecată ale companiei
  - 2000: 80 milioane utilizatori
- Istoria Napster: declinul
  - Mid 2001: Compania falimentată de procese
  - Mid 2001: Zeci de alternative P2P mai eficiente şi mai greu de atins prin procese
  - 2003: dezvoltarea unor servicii contra cost, gen iTunes
- Istoria Napster: revenirea
  - 2003: Napster reconstituit ca serviciu plătit, cu un succes limitat

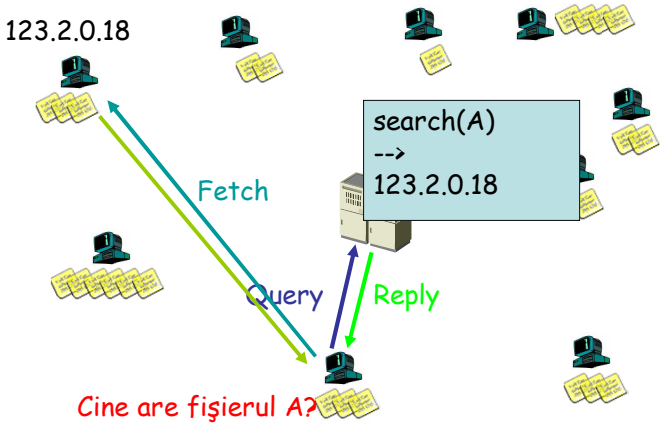
Napster

- **Join**: la pornire, clientul contactează serverul central
- **Publish**: raportează fişierele proprii către serverul central
- **Search**: query către server pentru a primi lista nodurilor care deţin fişierul căutat
- **Fetch**: transfer direct al fişierului de la un nod listat

Napster: Publish



Napster: Search



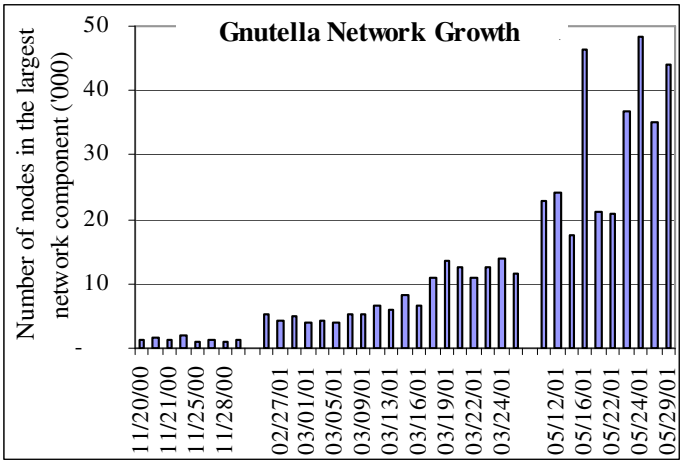
Napster

- Avantaje:
  - Simplu
  - Căutarea se face cu O(1)
  - Controlabil (avantaj sau dezavantaj ?)
- Dezavantaje:
  - Serverul menţine întreaga stare a reţelei
  - Serverul efectuează toate căutările
  - *Single point of failure*
  - Juridic: existenţa serverului = problema principală
  - Atacuri legale asupra serverului, nu clienţilor

Gnutella - istorie

- Gnutella - scris de Justin Frankel, 21 ani, fondator al Nullsoft.
- (Nullsoft cumpărat de AOL, iunie 1999)
- **14 martie 2000**: Nullsoft (producător al WinAmp) postează Gnutella pe web.
- **15 martie 2000**: AOL elimină Gnutella din locaţia postată de Nullsoft, la cererea Time Warner.
- 1 zi a fost de ajuns: 23k utilizatori ai Gnutella
- Gnutella continuă ca program independent de Nullsoft (alte implementări)

Creşterea Gnutella

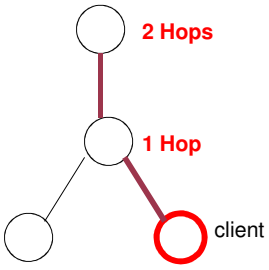


Factori de creştere
- creşterea DSL şi a modemurilor de cablu în acea perioadă
- Disponibilitatea progresivă a mai multe software-client

- 2007: Gnutella, cea mai mare rețea P2P (40% din total)
- 2010: restricții judecătorești afectează cel mai popular client (*Limewire*)
- 2015: Gnutella încă funcțională, dar în scădere masivă față de BitTorrent

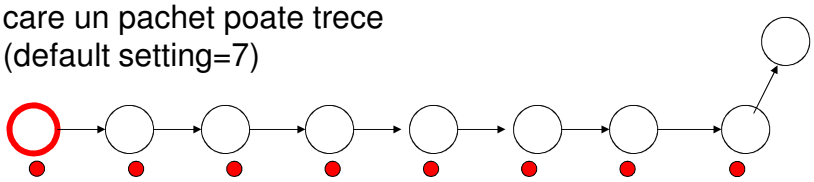
Gnutella

Servent: Nod Gnutella, în acelaşi timp server şi client.

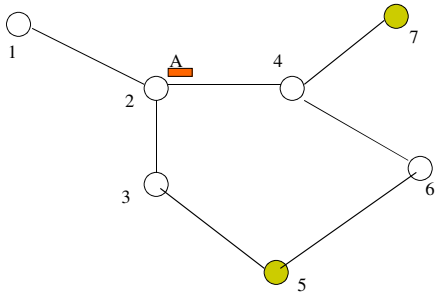


Hops: o trecere printr-un nod intermediar

TTL: nr. Maxim de hopuri prin care un pachet poate trece (default setting=7)

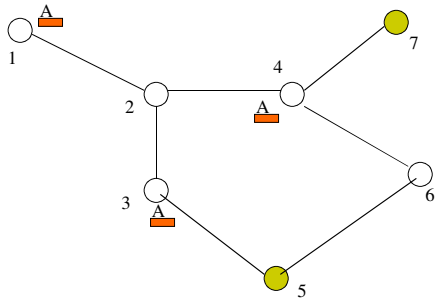


Gnutella: mecanismul de *search*



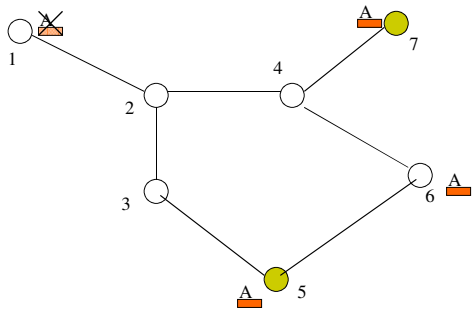
Paşi:
- Nod 2 iniţiază căutare pt fişierul A

Gnutella: mecanismul de *search*



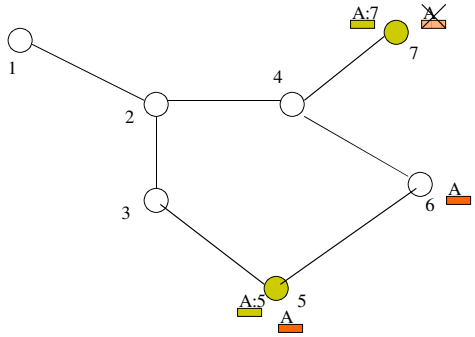
Paşi:
- Nod 2 iniţiază căutare pt fişierul A
- trimite mesajul către toţi vecinii

Gnutella: mecanismul de *search*



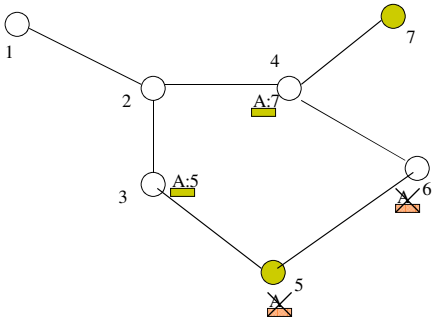
Paşi:
- Nod 2 iniţiază căutare pt fişierul A
- trimite mesajul către toţi vecinii
- vecinii trimit mesajul mai departe

Gnutella: mecanismul de *search*



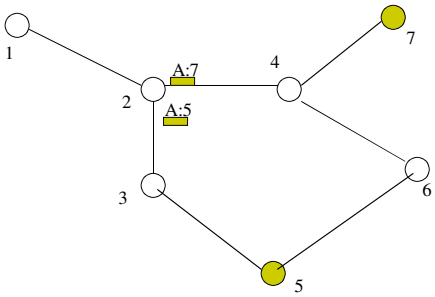
Paşi:
- Nod 2 iniţiază căutare pt fişierul A
- trimite mesajul către toţi vecinii
- vecinii trimit mesajul mai departe
- nodurile care deţin fişierul A iniţiază un mesaj *reply*

Gnutella: mecanismul de *search*



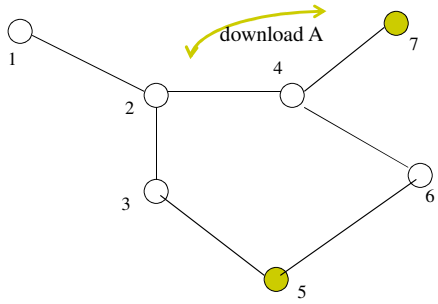
Paşi:
- Nod 2 iniţiază căutare pt fişierul A
- trimite mesajul către toţi vecinii
- vecinii trimit mesajul mai departe
- nodurile care deţin fişierul A iniţiază un mesaj *reply*
- mesajul *reply* este trimis înapoi pe aceeaşi cale

Gnutella: mecanismul de *search*



Paşi:
- Nod 2 iniţiază căutare pt fişierul A
- trimite mesajul către toţi vecinii
- vecinii trimit mesajul mai departe
- nodurile care deţin fişierul A iniţiază un mesaj *reply*
- mesajul *reply* este trimis înapoi pe aceeaşi cale

Gnutella: mecanismul de *search*



- OBS: dacă un nod X este în spatele unui firewall, se poate folosi *push* de la X la cel care cere fişierul

Paşi:

- Nod 2 iniţiază căutare pt fişierul A
- trimite mesajul către toţi vecinii
- vecinii trimit mesajul mai departe
- nodurile care deţin fişierul A iniţiază un mesaj *reply*
- mesajul *reply* este trimis înapoi pe aceeaşi cale
- Download direct al fişierului de la sursă

Gnutella: mecanismul de *search*

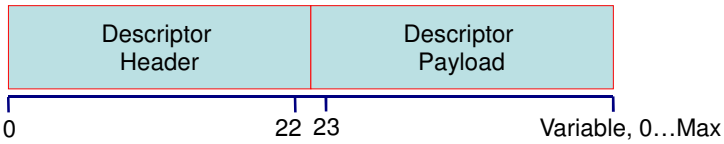
- TTL limitat la 7

- avantaj: se limitează inundarea reţelei cu queries
- dezavantaj: un fişier aflat la mai mult de 7 noduri distanţă nu va fi găsit
  - Gnutella favorizează găsirea fişierelor populare; cele mai neobişnuite şi puţin cerute vor fi şi mai greu de găsit

Gnutella - terminologie

- Mesajele schimbate între noduri = *descriptori*
- Nodurile = *servents*

Descriptori Gnutella

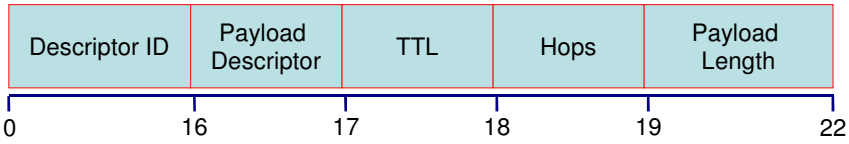


Mesajele schimbate între noduri = *descriptori*

Tipuri de Descriptori:

- **Ping:** descoperă în mod activ serverele în reţea
- **Pong**: răspunsul la *Ping (*include GUID - addresa unui *servent* conectat şi informaţii despre datele puse la dispoziţie de către *servent*)
- **Query**: mecanism de căutrare
- **QueryHit:** răspuns la *Query* (conţine GUID şi info fişier)
- **Push:** mecanism pt *firewalled servents*

Header Descriptori Gnutella



• **Descriptor ID**: identificator unic pt descriptor (16-byte string)

• **Payload Descriptor**: *0x00 = Ping: 0x01 = Pong: 0x40 = Push: 0x80 = Query: 0x81 = QueryHit*

• **TTL:** *Time To Live* sau *Horizon.* Fiecare *servent* decrementează TTL înainte de a-l trimite mai departe; TTL=0 = descriptor eliminat

• **Hops:** nr. de hopuri prin care descriptorul a trecut; hops = TTL(0) cînd TTL a expirat

Payload Length: următorul header de descriptor se află exact la *Payload Length* octeţi de la sfîrşitul headerului curent

Gnutella Payload 1 –Descriptor Ping

• Descriptori **Ping:**

  • fără payload

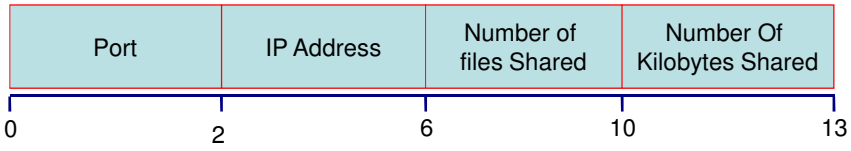
  • = lungime zero

• Ping = ***Descriptor Header*** cu:

  • **Payload_ Length** = 0x00000000.

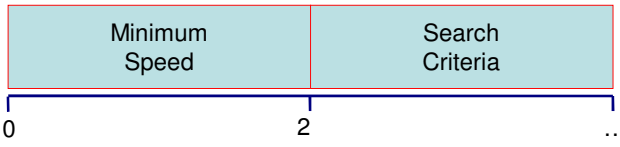
  • ***Payload_Descriptor*** f= 0x00

Gnutella Payload 2 - Pong



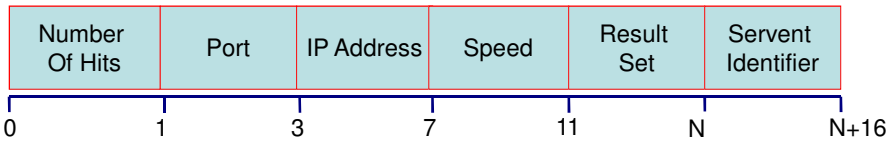
- **Port:** port pe care hostul respondent poate accepta conexiuni *incoming*
- **IP Address:** adr IP a respondentului (big-endian)
- **Number of Files Shared**: număr de fişiere pe care respondentul îl pune la dispoziţia reţelei
- **Number of Kilobytes Shared**: idem în KB

Gnutella Payload 3 - Query



- **Minimum Speed:** viteza minimă (în kb/sec) a *servents* care pot răspunde la mesaj
  - Un servent nu va răspunde dacă nu poate comunica la o viteză egală sau mai mare
- **Search Criteria:** şir ASCIIZ care conţine numele căutat; lungimea maximă este limitată de cîmpul *Payload_Length* a headerului de descriptor
  - ex: “nume_cutare.mp3”

Gnutella Payload 4 - QueryHit



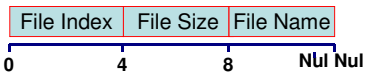
• **Number of Hits:** număr de rezultate pozitive la query (hits) în result set

• **Port:** port pe care respondentul poate accepta conexiuni

• **IP Address:** Adr. IP a respondentului (big-endian)

• **Speed:** viteza (în kb/sec) a respondentului

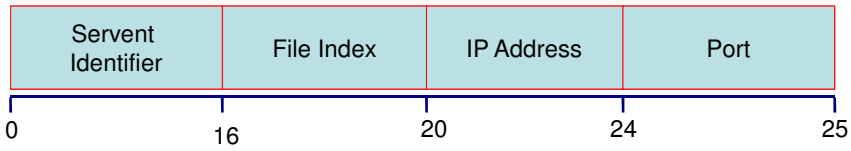
• **Result Set:** mulţimea de răspunsuri *Number_of_Hits* la mesajul Query cu următoarea structură:



• **File Index:** ID al fişierului care satisface query corespunzător – asignat de respondent
• **File Size:** mărimea în octeţi a fişierului
• **File Name:** nume fişier terminat cu 0x0000

• **Servent Identifier:** network ID (16 octeţi) al servent, în funcţie de adresa de reţea al servent; necesar în operaţii cu ***Push Descriptor*** ….

Gnutella Payload 5 - Push



în cele ce urmează *servent* se referă la cel *target* (destinaţie)

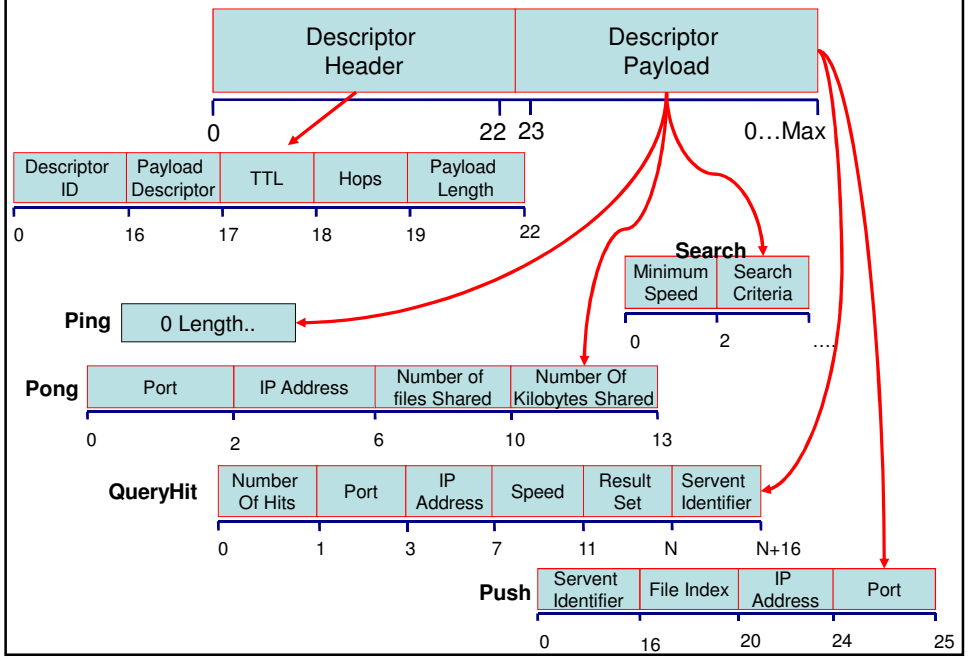
•**Servent Identifier:** *servent* network ID (16 octeţi) căruia i se solicită să efectueze push asupra fişierului (cu indexul *File_Index*)

• **File Index:** ID fişier de mai sus

• **IP Address:** adr. IP a *servent* a cărui fişier trebuie pushed (big-endian)

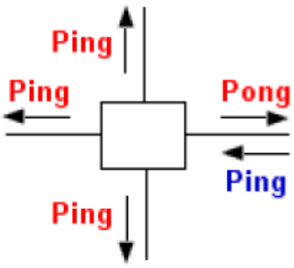
• **Port:** port pe *servent*

Gnutella Descriptor



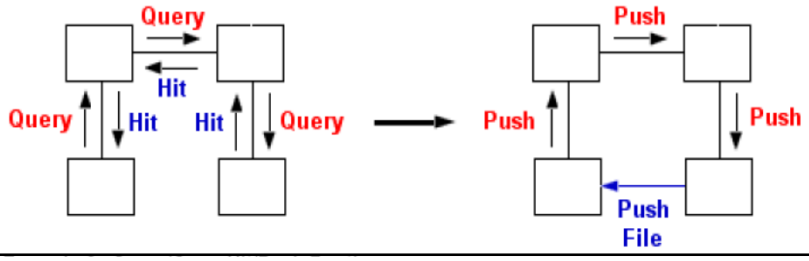
Rutarea descriptorilor

- Necesară pt. a controla traficul în Gnutella
- Descriptorii Pong trimişi pe aceeaşi cale pe care au venit descriptorii Ping



Rutarea descriptorilor

- Descriptorii Query Hit trimişi pe aceeaşi cale ca Query
- Descriptorii Push trimişi pe aceeaşi cale ca Query Hit



Transferul de fişiere

- Conexiune directă între sursă şi destinaţie
- Fişierul nu este trimis prin reţeaua Gnutella
- Reţeaua Gnutella trimite doar descriptorii prezentaţi pînă acum
- Pt. download: se foloseşte HTTP

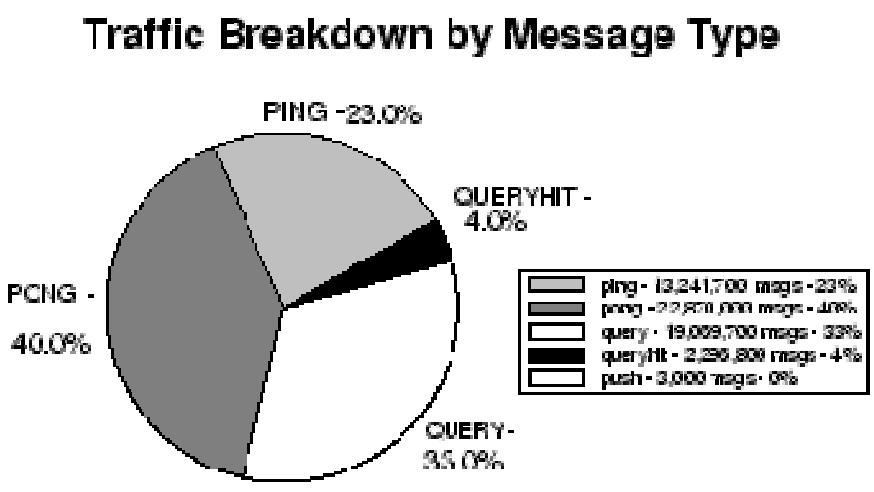
GET /get/<File Index>/<File Name>/ HTTP/1.0\r\n

Exemplu

File Index	*2468*
File Size	*4356789*
File Name	*Foobar.mp3\x00\x00*

GET /get/2468/Foobar.mp3/ HTTP/1.0\r\n

Gnutella - eficienţa ?



- Sursa: “A Quantitative Analysis of the Gnutella Network Traffic”

“Free Riding” în Gnutella

- 70% din utilizatorii Gnutella nu împart (*share*) fişiere
- 90% din utilizatori nu răspund la *queries*
- Mulţi din cei care *share* limitează nr. de conexiuni şi viteza de upload, rezultînd într-o rată mare de pierderi la download
- Efectiv: un număr redus de noduri contribuie la “binele reţelei”; acestea sînt echivalente cu nişte servere centralizate

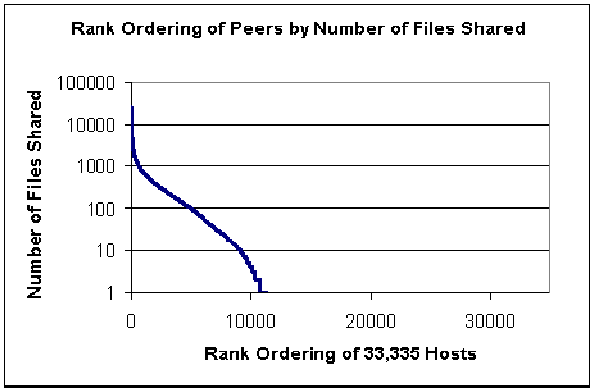


Figure 1

- Adar and Huberman at http://www-2.cs.cmu.edu/~kunwadee/research/p2p/gnutella.html

KaZaA - istorie

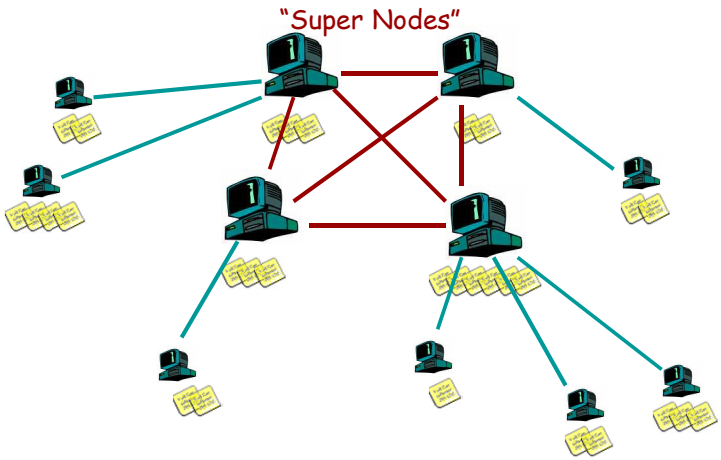
- Creat de compania estoniană Blue Moon Interactive
- Reţeaua se numeşte FastTrack şi softul de client s.n. KaZaA
- Vîndută către companii finlandeze
- martie 2001: vîndută către Consumer Empowerment (companie olandeză)
- sfîrşit 2001: vîndută către Sharman networks
- 2001-2006: multiple acţionări în judecată
- 2003: acţionări în judecată a utilizatorilor KaZaa

- Curent: variantă legală cu taxă lunară de descărcare
- Varianta “ilegală” încă în funcţiune, dar în scădere masivă
  - mai ales din cauza acuzaţiilor că softul proprietar KaZaa instalează componente de tip AdWare şi SpyWare
  - opţiune: Kazaa Lite (dezvoltată independent faţă de Sharman Networks, ne-aprobată de aceştia, dar funcţionînd tot în reţeaua FastTrack, deci compatibilă)

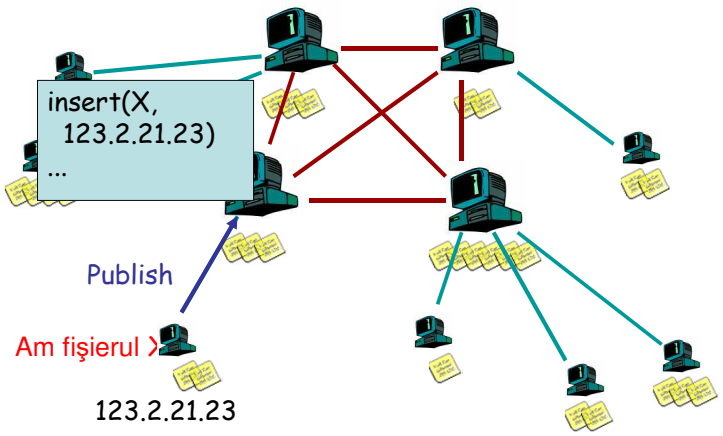
KaZaA - descriere

- Gnutella • Napster (generaţia 2)
  - Fără server dedicat
  - Nu toate nodurile (*peers*) sînt egale
- "Smart" Query *Flooding*:
  - **Join**: la pornire, clientul contactează un "supernod" ... La un moment dat poate deveni el-însuşi supernod
  - **Publish**: trimite lista de fişiere pe care le deţine către supernod
  - **Search**: trimite o cerere către supernod; supernodurile trimit între ele cererea cu flood query.
  - **Fetch**: transferă fişierul direct de la peer; poate transfera simultan de la mai multe peers

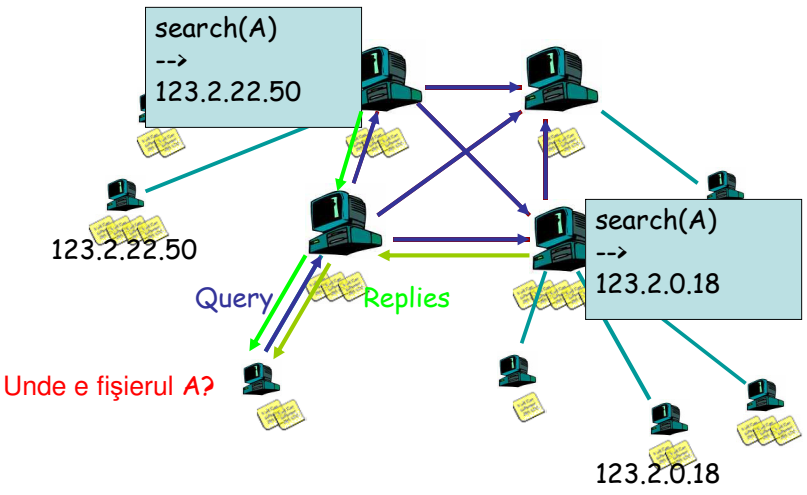
KaZaA: arhitectură reţea



KaZaA: File Insert



KaZaA: File Search



KaZaA: Fetching

- Mai mult de un nod poate deţine fişierul cerut
- Cum să determinăm aceasta?
  - Fişierul cerut poate avea nume diferit
  - Numele poate fi acelaşi dar fişierul nu.
- Folosim un hash al fişierului
  - KaZaA foloseşte UUHash: mai rapid dar mai puţin sigur
  - Alternative: MD5, SHA-1
- Cum se face transferul ?
  - transferă octeţii [0..1000] de la A, [1001...2000] de la B

Stabilitate şi Superpeers (Supernoduri)

- De ce supernoduri?
  - Query consolidation
    - Multe din nodurile simple pot avea un nr. redus de fişiere
    - Propagarea unui query către un astfel de nod consumă timp inutil căci majoritatea queries vor primi răspuns negativ
- Selecţia Superpeer este bazată pe timp
  - Cît timp ai fost în reţea este un bun indicator al cît timp vei mai rămîne în reţea

KaZaA: Discuţie

- Avantaje:
  - Ia în considerare eterogenitatea nodurilor:
    - Bandă
    - Resurse de calcul
    - disponibilitate (?)
- Dezavantaje:
  - Mecanisme uşor de ocoloit
    - “free riding” e posibil
  - Nu se garantează timpul de căutare şi nici măcar aria căutărilor
- Îmbunătăţire a Gnutella

BitTorrent

- Autor: Bram Cohen, 2001
- Adresează doar transferul de fişiere, nu şi căutarea fişierelor
  - căutarea se face pe Web
- Principiu: *Tit-for-tat*
  - reciprocitate: download-ul e permis doar celor care fac upload
  - elimină *Free Riding*
- Fişiere împărţite în bucăţi (*pieces*)
- UL/DL se face la nivel de *piece*
- Un user poate DL *pieces* diferite ale aceluiași fișier, simultan de la mai mulți *peers* (parteneri)

Terminologie

Peer = nod

Seeder = un peer care deţine întregul fişier.

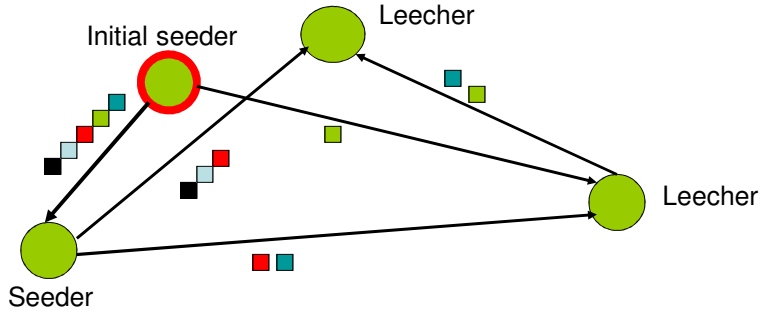
Initial seeder = primul seeder al fişierului

Swarm = Set de peers care UL/DL acelaşi fişier

Leecher = un peer care nu deţine (încă) întregul fişier

Tracker = sistem care conţine lista peers şi ce porţiuni de fişier are fiecare

Choked = stare a conexiunii în care un *peer* nu face UL către alt *peer*



Mesaje

- Mesaje Peer – Peer
  - TCP Sockets
- mesaje Peer – Tracker
  - HTTP Request/Response
- B-encoding
  - format de codare a tipurilor de date:
    - 4:text, 2:ab reprezintă şirurile “text”, “ab”
    - i0e, i-45e, i12345e reprezintă numerele 0, -45, 12345
    - l4:text2:ab:i1e3:abce reprezintă lista “text”,”ab”,1,”abc”
    - dicţionare: d<bencoded string><bencoded element>e

fişier .torrent

Conţine mai multe elemente, în formatul BEncode

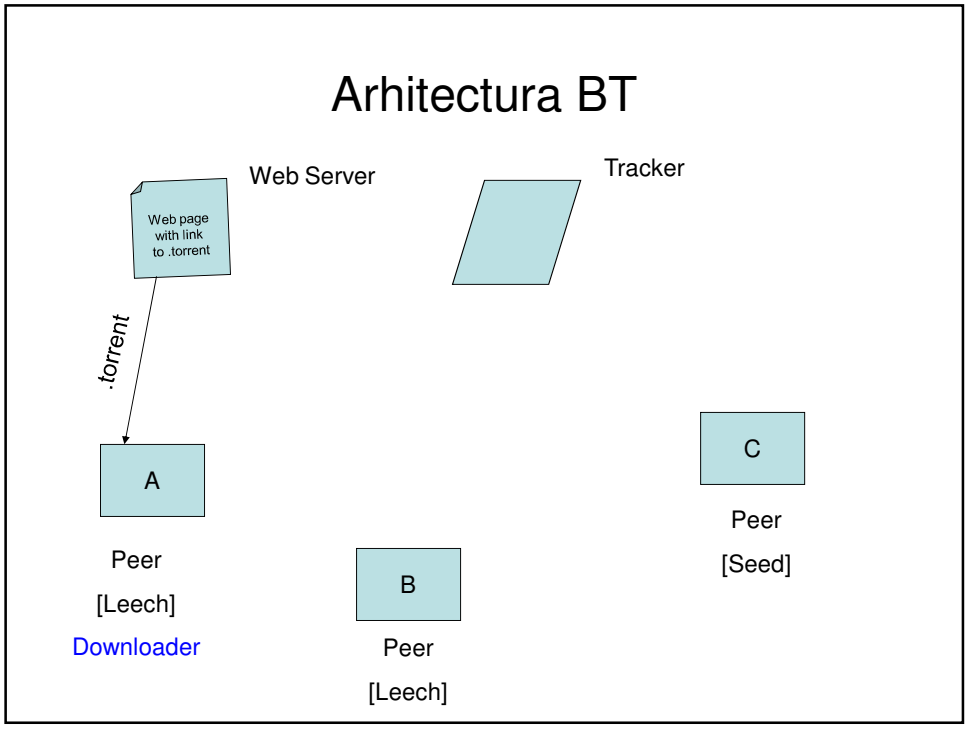
- announce: url al tracker-ului (sau listă de trackers)
- nume *publisher*, comentarii (opţional)
- info: dicţionar care descrie fişierele din .torrent:
  - length, name: lungime şi nume
  - Cale (path) dacă e un torrent compus din mai multe fişiere
  - piece length: lungimea fiecărei *Piece*
  - pieces: şir format prin concatenarea hash-urilor fiecărei *piece* <hash1,hash2,….hash*n*>

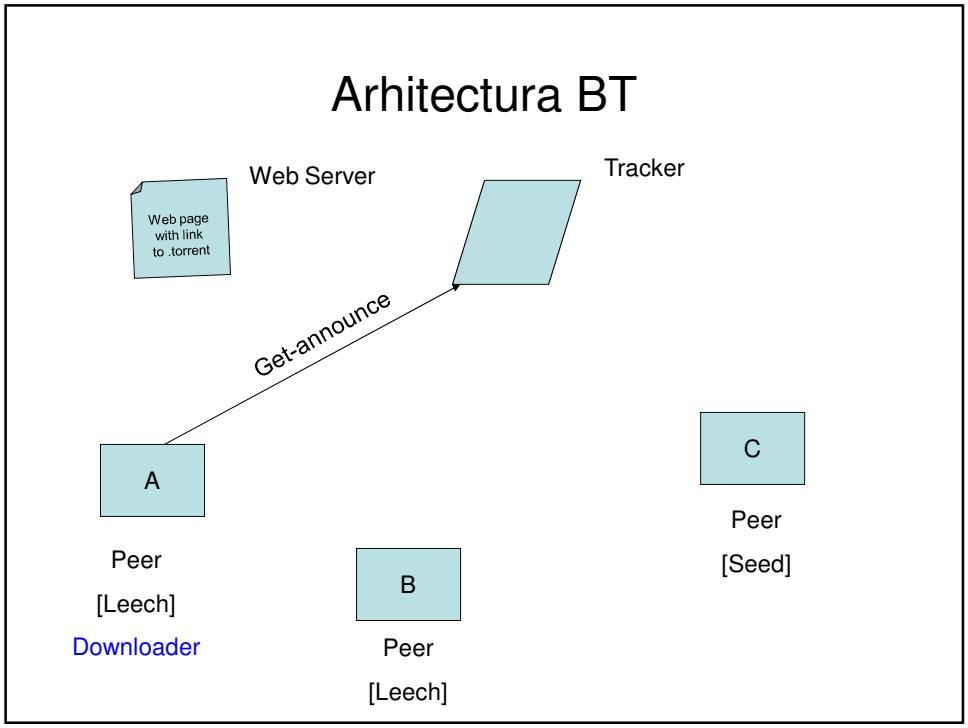
  (fiecare *piece* are un hash SHA-1 de 20 bytes pt. a putea face transferul la nivel de *piece*)

Exemplu fişier .torrent

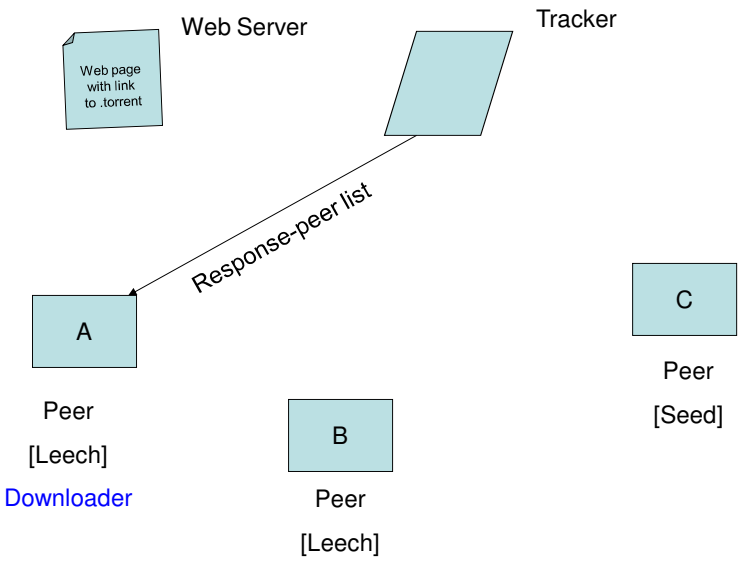
d8:announce40:http://tracker.thepiratebay.org/announce13:announce-listll40:http://tracker.thepiratebay.org/announceel42:udp://tracker.thepiratebay.org:80/announceel44:udp://tracker.openbittorrent.com:80/announceel42:http://tracker.openbittorrent.com/announceee7:comment47:Torrent downloaded from http://thepiratebay.org10:created by13:uTorrent/185013:creation datei1260707746e8:encoding5:UTF-84:infod6:lengthi2362253e4:name31:The Lost Symbol - Dan Brown.pdf12:piece lengthi65536e6:pieces740:h›f>a—®:)ì¤ÍØqo¡Ìh=1…nÑÆ·ª~úmÝõEmcÂ7éŸâD9\ßÍC¦ü‡7-7
zeêxüööG?¬)è`ᵑ||»'å´óiÇ8Ù§YjŒõ/æµ*+Uf¯ôÕàn<ÓîÉLs€·tõ· n¶ƒlŸŒ6*âµF;
%>×•]¿zË.ºdèt3HËÈJ·üè›®ˮ &¬oE>.`¬',õ?'Þ-ðš
·yo9· @½‡§AGåGĢó'^·
·»˜¥æ9ZS%]R/zTÈh^uÐD7
¡ÜjŸ¼Ñ¨Ø·¸åã×»kSu
¸¿gc/· N×Ó·qšú·KĴù¶§úP|{>Ó¤dÆ@“…¶3'/ª,æK‹Â€Üc6/•(5?…Js·s{/ù¦¾Y{©
Ñ·'DÂ¿1o·€ÿìæhŠ‡Ţøìjõv3b²ý·_=Ñ¡·ý¸ìŞÎër·ˆw{Êî·zª–
îŸ·.ï«ÚÍZs·Êlã,¤·B<ËLÖ·3ÁSÍ%re<Ęc:=)Ę·X'7µ·p·äò%g^…ŻWĎYQÎ
¶Ý€Þ›IÜ½'+k_õIĢ„ˆy""þ%àCþá[‹¢²ÎŠæíßÕáCƒKha8¢ÿ#*œ'ÈùÖ=I! >
ÌàT·ñî·>d?eâvbFÙ:ø„åž‰Rd&9PĢ[‡˜@p>DW?®ë60]N~Ẁ'·Miç
põ€Ui `Ì»õ"dþ\}2S*w§˜y"±þ ºæÞJÕà…iã Î\üÐ)f÷—C‹±ùàöĒMÌlýçÑà J ½ý“—
¤2ë+%ÛÙ™·QĒĶ¥1°þþÙÖbõdŠ·µRQJm‹äÔrè ÌB@çäyŠC;-X^EYn£-
çķF5½‡¼œá'¦·ùĒMs½ Ì (ëj¯—¢žó1ĵŸ7•]†·Œ@ê©£±šĻþ‹,-·'· ï$«x…·O¨1…Î
·Nò<Ì¼™¯ %ª ?='·<ákÕz‡J·©r8ØëÑUñ¸Üÿee

(se respectă culorile din slide precedent)

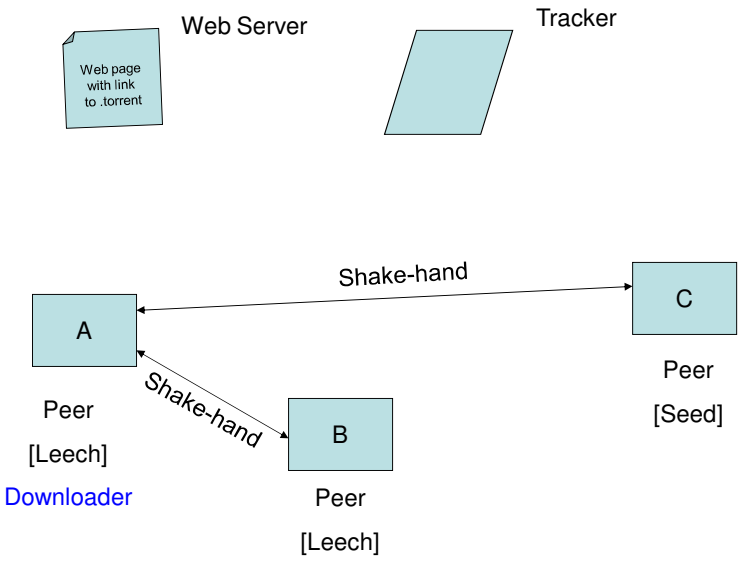

Arhitectura BT
Web Server
Tracker
Web page with link to .torrent
.torrent
A
Peer
[Leech]
Downloader
B
Peer
[Leech]
C
Peer
[Seed]


Arhitectura BT
Web Server
Tracker
Web page with link to .torrent
Get-announce
A
Peer
[Leech]
Downloader
B
Peer
[Leech]
C
Peer
[Seed]

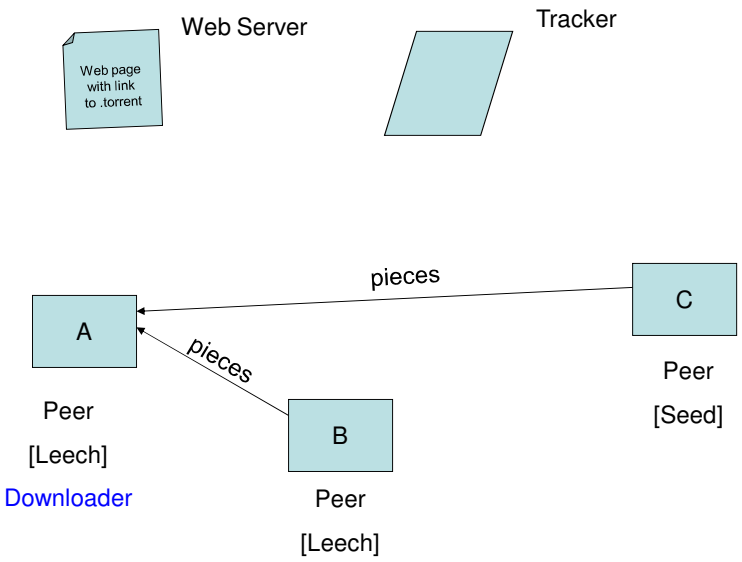
Arhitectura BT



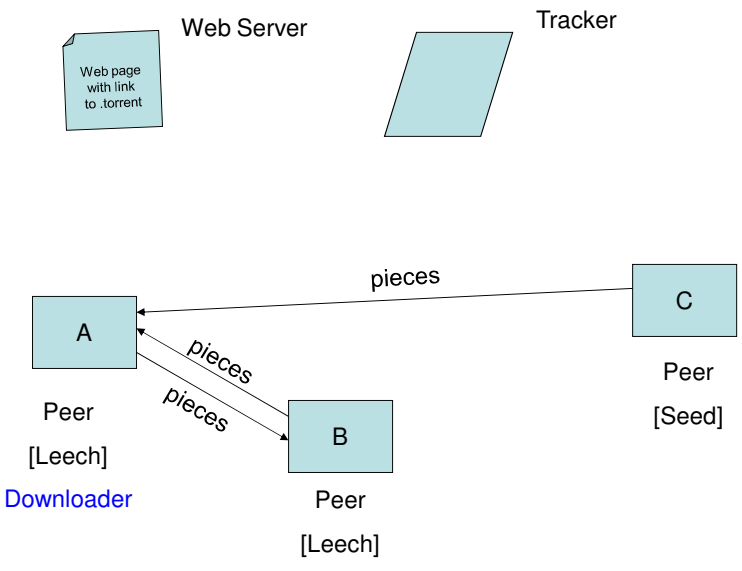
Arhitectura BT



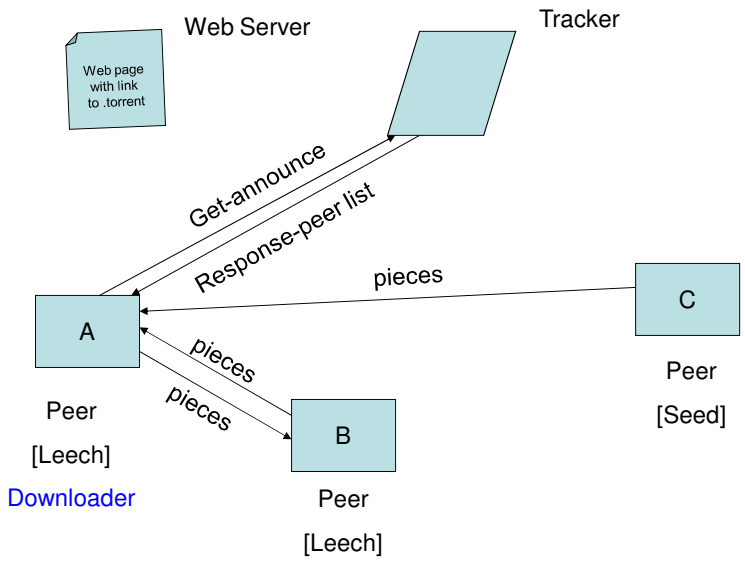
Arhitectura BT



Arhitectura BT



Arhitectura BT



Mesaj *GET-Announce* (Peer → Tracker)

- **Info_hash** - 20 byte SHA1 hash al valorii *info* sub formă *bencoded*.
- **Peer_id** - șir de lungime 20 conținînd un ID al *peer*-ului, generat aleator la începutul operatiei de DL.
- **IP** - IP (sau nume DNS) al *peer*.
- **Port** - număr de port de folosit de peer pentru *listen*, începînd cu 6881, și dacă e ocupat, 6882, etc - pînă la 6889.
- **Uploaded** - cît s-a UL din acel fișier pînă atunci (în ideea că s-a repornit un transfer, UL și DL nu sînt 0)
- **Downloaded** - cît s-a DL din acel fișier pînă atunci.
- **Left** - număr de bytes care mai rămîn de DL de către acel peer
- **Event** - opțional: poate fi *started*, *completed*, sau *stopped.*

Mesaj *Response peer-list* (Tracker → Peer)

- Răspunsurile de la tracker sînt dicționare *bencodate*
- Dacă există o eroare, dicționarul conține o singură cheie, un mesaj de eroare care se transmite ca text pentru a putea fi citit de către user.
- În caz contrar, se transmit două chei:
  - **Interval** numărul de secunde cît *peer* trebuie să aștepte pînă la un nou *request*
  - **Peers** listă de dicționare, fiecare conține 3 chei:
    - **peer ID** (cel trimis de *peer* ,obținut aleator)
    - **IP**
    - **port**

Operaţii Peer-Peer

Un peer:

- Alocă spaţiu pe disc (fişier de 1G = 1G alocat)
- se conectează la alţi peers

- trimite un mesaj inițial Bitfield
  - biți de 1 sau 0 pentru *pieces* pe care le are sau nu
- trimite mesaje *request <index, begin, length>* unde *begin, length* sînt offset-uri exprimate în octeți; tipic, length este 16KB mai puțin dacă se trunchează la sfîrșitul fișierului
- primește mesaje *piece <index, begin, piece>*
- trimite mesaj *Have <index>* cînd o *piece* cu numărul *index* s-a recepționat complet și SHA-1 corespunde
- trimite mesaj *cancel <index, begin, length>*

către un *peer* atunci cînd are deja *piece* pe care o primește și de la acest *peer*;

Mesaje Peer-Peer

primul octet din orice mesaj:

- 0 - choke
- 1 - unchoke
- 2 - interested
- 3 - not interested
- 4 - have
- 5 - bitfield
- 6 - request
- 7 - piece
- 8 - cancel

mesajele *choke, unchoke, interested*, și *not interested* nu mai au cîmp de date *(payload*)

Tracker

- construit pe baza unui server HTTP
- urmărește (*tracks*) toate *pieces* ale unui fișier din rețea
- monitorizează UL/DL pentru acel fișier, din partea userilor
- implementează politica *tit-for-tat* pe baza UL: un user care UL mai mult poate DL mai rapid; în lipsa UL nu se poate DL.

Pieces şi *Sub-Pieces*

- O *piece* este împărţită în *sub-pieces,* tipic de 16KB
- cîtă vreme o *piece* nu este transferată complet, doar *sub-pieces* ale ei sînt transferate
- astfel peers vor acumula rapid *pieces* întregi pe care le vor putea transfera altui peer

BT: algoritmi de selecţie a *pieces*

- **Rarest First**
  - regula uzuală
- **Random First Piece**
  - caz special, la început
- **Endgame Mode**
  - caz special, la sfîrşit

Random First Piece

- Iniţial, un peer nu are ce să ofere la schimb
- Important să deţină o *piece* completă cît mai repede
- Alege o *piece* aleatoare şi o cere

Rarest Piece First

- peers deţin un număr de *pieces*
- determină între ei care sînt *pieces* cele mai rare (deţinute de cei mai puţini *peers* cu care se află în proces de schimb) şi le transferă pe acestea
- *pieces* rare nu se vor pierde dacă cîţiva peers părăsesc *swarm*

Endgame Mode

- Cînd fişierul e aproape complet transferat, un peer trimite request pentru *pieces* care încă îi lipsesc către toţi peers care au acea *piece*
- Cînd o *piece* este transferată complet, cererile către ceilalţi peers pentru acea *piece* sînt anulate
- Aceasta asigură că terminarea DL nu este amînată nedeterminat datorită unor peers cu viteză de transfer prea mică
- Acest proces risipeşte într-o anumită măsură banda disponibilă, dar în practică se dovedeşte că nu prea mult

BT: mecanisme interne

mecanisme de încurajare (**incentive)** de a face upload la fişiere, pentru a preveni *Free Riding*:

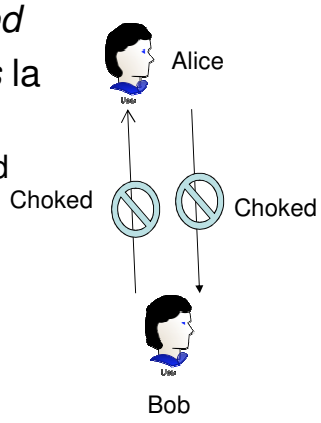
- Algoritm de *Choking*
- *Optimistic Unchoking*

Choking

- **Choking** este o situaţie de refuz temporar de a face *upload* către alţi peers.
- ***Tit-for-tat strategy*** (reciprocitate) este un concept bazat pe teoria jocurilor

Choking

- Peer A *chokes* Peer B atunci cînd A nu face UL către B și B este *interested*
- Peer A *unchokes* maximul 4 *peers* la un moment dat:
  - Cei 3 cu cele mai mari rate de upload către A (tit-for-tat)
  - Încă unul ales aleator (Optimistic Unchoke) - pentru a căuta periodic opțiuni mai bune



Optimistic unchoking

- Un peer are la un moment dat un singur peer numit "optimistic unchoke" căruia îi face UL indiferent de rata de DL de la el. Acest peer se schimbă la fiecare 30s
- Motivaţie:
  - găsirea unor peers mai buni
  - oferirea unor servicii minimale către peers nou-veniţi.

Modul Upload-Only

- Odată ce DL este complet, un peer nu mai are rate de DL de comparat între peers;
- Q: Cum se face selecţia peers către care se face UL în această situaţie ?
- A: Se face UL către peers cu cea mai bună rată de UL. Aceasta încurajează seeders.

Avantaje și dezavantaje BT

Avantaje:

•utilizează eficient fișiere DL parțial

•*tit-for-tat* descurajează *free-riding*

•sistemul *rarest first* ține *swarm*-ul activ un timp cît mai lung posibil, indiferent ce *peer* părăsește *swarm*-ul

Dezavantaje:

•Nu conține partea de *search* implicită, ca celelalte rețele P2P (căutarea se face pe web)

•tracker = *single point of failure*; dispariția *tracker*-ului duce la imposibilitatea ca noi *peers* să intre în *swarm*

•performanța depinde de popularitate; un fișier vechi (ajuns nepopular) poate să nu mai aibă *tracker* și nu va mai fi posibil UL/DL, chiar dacă *pieces* sînt disponibile

Bibliografie

Y.C. Lai, *CS 640: Introduction to Computer Networks*

E. Horowitz, *A look at Peer-to-Peer File Sharing with Gnutella*

G. Tribhuvan, *Gnutella: A Summary Of The Protocol and its Purpose*

B. Cohen, *Incentives Build Robustness in BitTorrent*

Bittorrent: The protocol, its background and uses

http://www.bittorrent.org/beps/bep_0003.html (BitTorrent protocol specification)