ProGres, Part III: Peer-to-Peer

Fabien Mathieu
(fabien.mathieu@normalesup.org)

Master ProGres
Sorbonne Université(s)

November 22, 2024

Roadmap

1. Part I: Connectivity
2. Part II: Web services
3. Part III: Peer-to-peer
   - A brief overview
   - Distributed Hash Tables
   - Bonuses: Bandwidth and BitTorrent (if time)

Online resources

Main entry point	`https://www-npa.lip6.fr/~tixeuil/m2r/pmwiki.php?n=Main.PROGRES`
GitHub	`https://github.com/balouf/progres`

Part III

A) What is P2P?

Definition by enumeration

"Everyone" knows what P2P is:

FileSharing	BitTorrent, eMule,...
Streaming	TVants, PPLive,...
VoIP	Skype

Definition by enumeration

"Everyone" knows what P2P is:

FileSharing	BitTorrent, eMule,...
Streaming	TVants, PPLive,...
VoIP	Skype

"Everyone" knows what P2P is not:

Unplugged	Photoshop, Office...
Plugged	Apache, SSH, browsers...

Definition by enumeration

"Everyone" knows what P2P is:

FileSharing	BitTorrent, eMule,. . .
Streaming	TVants, PPLive,. . .
VoIP	Skype

"Everyone" knows what P2P is not:

Unplugged	Photoshop, Office. . .
Plugged	Apache, SSH, browsers. . .

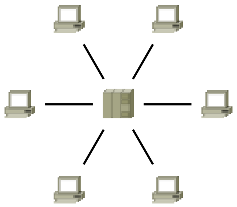
The blurry frontier:

Distributed computing	BOINC projects (Seti@home, Folding@home. . .), BitCoin. . .
Clouds/datacenters	Amazon EC2/AWS, Microsoft Azure, Dropbox, . . .

Network definition

Most network apps follow a client/server paradigm:

- On the one hand, servers supply resources,
- On the other hands, clients uses these resources.

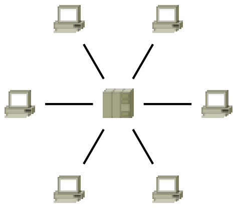


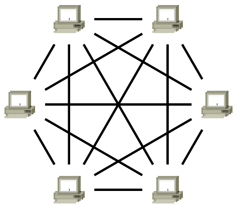
Network definition

Most network apps follow a client/server paradigm:

- On the one hand, servers supply resources,
- On the other hands, clients uses these resources.

Network definition → break the distinction between client and server: every peer can act as a client and/or as a server.





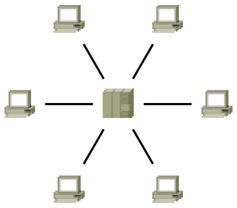
Network definition

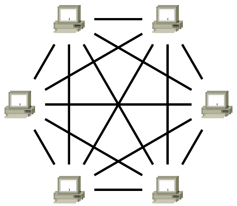
Most network apps follow a client/server paradigm:

- On the one hand, servers supply resources,
- On the other hands, clients uses these resources.

Network definition → break the distinction between client and server: every peer can act as a client and/or as a server.

Is it a good definition?





Network definition: what about Skype?

Back in the days, Skype was branded as a P2P application. Why should Skype be P2P?

- Interaction between two peers?

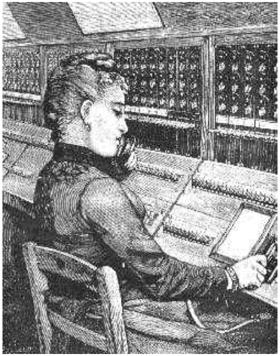
A very, very old tech indeed

Network definition: what about Skype?

Back in the days, Skype was branded as a P2P application. Why should Skype be P2P?

- Interaction between two peers?
- Interactions between any pair of peers?

An old tech



Network definition: what about Skype?

Back in the days, Skype was branded as a P2P application. Why should Skype be P2P?

- Interaction between two peers?
- Interactions between any pair of peers?
- From the makers of KaZaA and Joost (more to come later)?

Back to enumeration

Network definition: what about Skype?

Back in the days, Skype was branded as a P2P application. Why should Skype be P2P?

- Interaction between two peers?
- Interactions between any pair of peers?
- From the makers of KaZaA and Joost (more to come later)?
- Even the network definition is not perfect??

Alternate definitions: methodology, structure

A bit of fun: legal definition in France

P2P can be used for illegal stuff. The DADVSI law (2006) tries to forbid any «software obviously intended for the unauthorized making available to the public of protected works or objects».

A bit of fun: legal definition in France

P2P can be used for illegal stuff. The DADVSI law (2006) tries to forbid any «software obviously intended for the unauthorized making available to the public of protected works or objects».

- Covers P2P sharing (data and multimedia)
- Does not cover Skype (much)
- May cover most network apps (client/server included) : Apache, SSH. . .

A bit of fun: legal definition in France

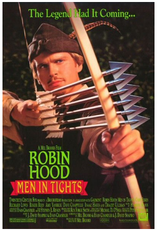
DADVSI hard to use in practice. Awareness of the difficulty of defining (legally) illegal P2P use.
Move to Hadopi (2009).

A bit of fun: legal definition in France

DADVSI hard to use in practice. Awareness of the difficulty of defining (legally) illegal P2P use.
Move to Hadopi (2009).

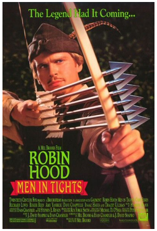
- Monitor target content (honeypot)
- Proving actual download by a physical person is impossible on a large scale
- Implementation of the characteristic negligence offence
- Technical failure, political success?

Methodology definition



P2P uses available resources (CPU, storage, bandwidth, content) wherever they are and redistributes them to those who need it.

Methodology definition

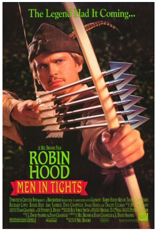


P2P uses available resources (CPU, storage, bandwidth, content) wherever they are and redistributes them to those who need it.

Two variants depending on what «wherever» means:

- At any user (covers BOINC)
- Anywhere in the network (CDN, EC2, . . .)

Methodology definition



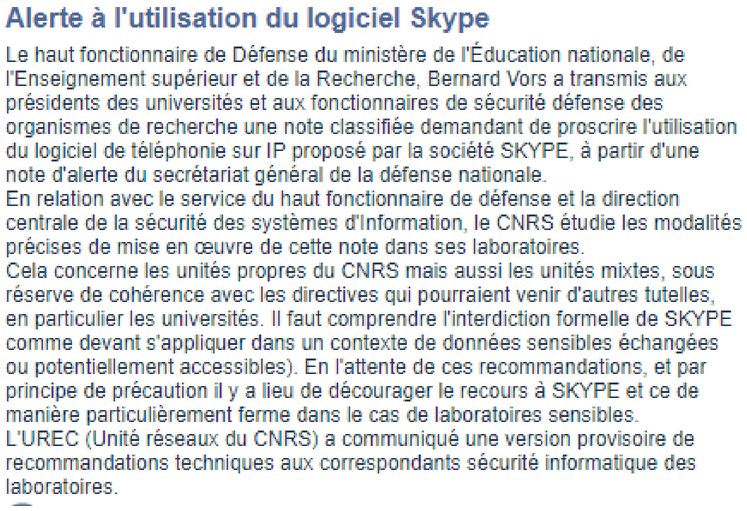
P2P uses available resources (CPU, storage, bandwidth, content) wherever they are and redistributes them to those who need it.

Two variants depending on what «wherever» means:

- At any user (covers BOINC)
- Anywhere in the network (CDN, EC2, . . .)

Skype: borderline

A bit of fun: Skype ban (2005)



Alerte à l'utilisation du logiciel Skype

Le haut fonctionnaire de Défense du ministère de l'Éducation nationale, de l'Enseignement supérieur et de la Recherche, Bernard Vors a transmis aux présidents des universités et aux fonctionnaires de sécurité défense des organismes de recherche une note classifiée demandant de proscrire l'utilisation du logiciel de téléphonie sur IP proposé par la société SKYPE, à partir d'une note d'alerte du secrétariat général de la défense nationale.
En relation avec le service du haut fonctionnaire de défense et la direction centrale de la sécurité des systèmes d'Information, le CNRS étudie les modalités précises de mise en œuvre de cette note dans ses laboratoires.
Cela concerne les unités propres du CNRS mais aussi les unités mixtes, sous réserve de cohérence avec les directives qui pourraient venir d'autres tutelles, en particulier les universités. Il faut comprendre l'interdiction formelle de SKYPE comme devant s'appliquer dans un contexte de données sensibles échangées ou potentiellement accessibles). En l'attente de ces recommandations, et par principe de précaution il y a lieu de décourager le recours à SKYPE et ce de manière particulièrement ferme dans le cas de laboratoires sensibles.
L'UREC (Unité réseaux du CNRS) a communiqué une version provisoire de recommandations techniques aux correspondants sécurité informatique des laboratoires.

Structural definition

Structure: how the elements of a system are arranged, and by extension, interact.
In networks, we can define by structure the answer to the question

Who gives what to whom?

Clients/servers, telephones, etc. generally have trivial structures.
Structural definition: P2P is characterized by non-trivial network structures.
Excludes trivial interactions (Skype?)

Definitions: take-away

Many definitions of P2P exist:

Enumeration	We all know what it is
Law	Mistake the “what” and the “how”
Network	No client/server distinction
Methodology	Tap unused resources
Structural	Non-trivial interactions

Recommendations

- There is no best answer (but there are bad ones)
- Use your brain!
- Is Skype P2P ? Who cares!

Definition for the course

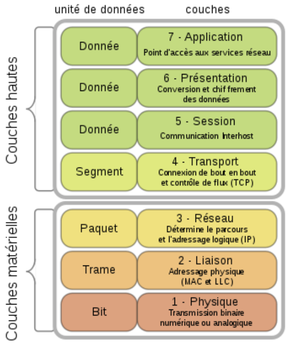
- Part III is mainly about (illegal?) content distribution
- Most non-stupid definitions are OK

Overlay

Re-re-reminder: OSI model

Network is split in 7 layers:

- Layer 7 (Application)
- Layer 4 (Point-to-point flow)
- Layer 3 (Routing)

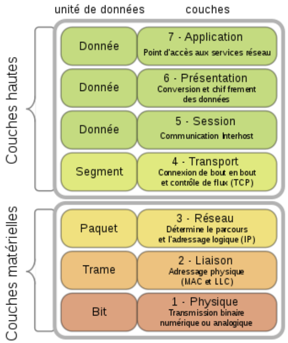

unité de données
couches
Couches hautes
Couches matérielles
Donnée
7 - Application
Point d'accès aux services réseau
Donnée
6 - Présentation
Conversion et chiffrement des données
Donnée
5 - Session
Communication Interhost
Segment
4 - Transport
Connexion de bout en bout et contrôle de flux (TCP)
Paquet
3 - Réseau
Détermine le parcours et l'adressage logique (IP)
Trame
2 - Liaison
Adressage physique (MAC et LLC)
Bit
1 - Physique
Transmission binaire numérique ou analogique

Overlay

Re-re-reminder: OSI model

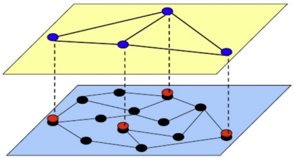
Network is split in 7 layers:

- Layer 7 (Application)
- Layer 4 (Point-to-point flow)
- Layer 3 (Routing)


unité de données
couches
Couches hautes
Couches matérielles
Donnée
7 - Application
Point d'accès aux services réseau
Donnée
6 - Présentation
Conversion et chiffrement des données
Donnée
5 - Session
Communication Interhost
Segment
4 - Transport
Connexion de bout en bout et contrôle de flux (TCP)
Paquet
3 - Réseau
Détermine le parcours et l'adressage logique (IP)
Trame
2 - Liaison
Adressage physique (MAC et LLC)
Bit
1 - Physique
Transmission binaire numérique ou analogique

P2P

- Creates a virtual network on top (overlay) of the physical network
- Overlay → graph
- Emulate routing / connectivity / . . . on the virtual topology
- A physical neighbor is not necessarily a virtual neighbor (and vice versa!)



Overlay: two main approaches

Structured graphs

Maintain a <u>shape</u> with good properties

- Performance is ensured by the graph structure
- Low maintenance for stable networks
- Fragile, high maintenance for dynamic networks
- Mostly unfit to heterogeneous networks

Unstructured graphs

Rely on randomness and self-adaptation

- Robust
- Adapted to heterogeneous networks
- Constant maintenance (costly on stable networks)
- No straightforward performance guarantee

Scalable

Client/server

- The capacity of servers is fixed
- Too many requests $\rightarrow$ DoS

P2P

- The capacity increases with the number of clients
- $\rightarrow$ P2P can in theory handle an arbitrary number of clients: scalability property

In practice, scalability is a real thing but it's not perfect

- Underlying physical network has limitations
- QoS constraints
- Size matters in practice (maintenance, logarithmic growth)

Fairness

Ideally

With P2P each peer can serve and be served by anyone: isotropic principle.

In practice

- It is not always possible to serve anything to anyone
- "All peers are equal, but some peers are more equal than others" (Orwell, the Server Farm)

$\rightarrow$ a peer is isotropic a priori when it enters the system

Resources and goals

Resources

- CPU
- Storage
- Content
- Bandwidth

Goals

- Distributed computed
- Content distribution
- Communication (Skype)
- Gaming

Resources and goals

Resources

- CPU
- Storage
- **Content**
- **Bandwidth**

Goals

- Distributed computed
- **Content distribution**
- Communication (Skype)
- Gaming

Content distribution

Three main branches

Filesharing the generic solution

- Primary goal: retrieve the content
- Secondary goals: availability, download time

Content distribution

Three main branches

On-demand Anything anywhen

- Catalog size (similar to filesharing availability)
- Minimize (start-up delay)
- Maximize quality

Content distribution

Three main branches

Live streaming

- Everyone wants to watch the same thing at the same time
- Minimize play-out-delay
- Maximize quality

Content distribution

Three main branches

Branch	FileSharing	On-demand	Live
Streaming?	No	Yes	Yes
Challenge	Minimal	Anything Anywhen	Small delay
Technical edge	Elastic traffic	Known content	Homogeneous needs
Technical difficulty	Minimal	Heterogeneous needs	Impossible anticipation

Indexation vs Distribution

All P2P content distribution systems must answer two questions:

Where is what I want?

- Build / maintain an index
- On-the-fly resolution
- Rely on a centralized solution

How do I get what I want?

- Just download from someone
- Be smart

Prehistory: communications

A few key dates

-200000?	Speech (network layer 1)
-30000?	Paintings (storage)
-7000?	Writing
-600$\rightarrow$ -100	Long distance (horses/pigeons/semaphores)
1454	Gutenberg
1876	Phone
1969	Arpanet
1993	MP3
1999	DSL (in France), DivX ;) 3.11

Prehistory: communications

A few key dates

-200000?	Speech (network layer 1)
-30000?	Paintings (storage)
-7000?	Writing
-600$\rightarrow$ -100	Long distance (horses/pigeons/semaphores)
1454	Gutenberg
1876	Phone
1969	Arpanet
1993	**MP3**
1999	**DSL (in France), DivX ;) 3.11**

A technical threshold is reached

- Around 2000, we can retrieve content in a reasonable time

	Hand	Print	Modem	DSL 512k	DSL 20M	FTTH
Book	2-3d	$\frac{1}{2}$h	1m	6s	0.15s	10ms
WAV	$\frac{1}{2}$y	1-2d	1h	6m	10s	0.5s
DVD	100y	1y	10d	1d	<1h	4m
MP3	20d	3h	6m	30s	1s	50ms
H.265	12y	40d	1j	3h	5m	30s

A technical threshold is reached

- Around 2000, we can retrieve content in a reasonable time
- A simple metric: can we “stream”?

	Hand	Print	Modem	DSL 512k	DSL 20M	FTTH
Book	No	Yes	Yes	Yes	Yes	Yes
WAV	No	No	No	Almost	Yes	Yes
DVD	No	No	No	No	Yes	Yes
MP3	No	No	Almost	Yes	Yes	Yes
DivX	No	No	No	Almost	Yes	Yes

A technical threshold is reached

Around 2000

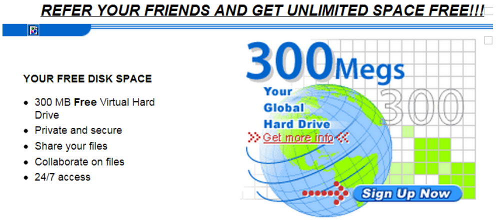
- Digital content becomes mainstream
  - CDs, MP3
  - DVDs, DeCSS, DivX
- High bandwidth enables download through Internet
- No YouTube, Netflix, Disney+, Prime, . . .
- A few solutions for the geeks
  - Newsgroups (`alt.binaries.*`)
  - FTPz
  - IRC
  - Myspace

A technical threshold is reached

Around 2000

- Digital content becomes mainstream
  - CDs, MP3
  - DVDs, DeCSS, DivX
- High bandwidth enables download through Internet
- No YouTube, Netflix, Disney+, Prime, . . .
- A few solutions for the geeks
  - Newsgroups (`alt.binaries.*`)
  - FTPz
  - IRC
  - **Myspace**

Intermezzo: Myspace, Dropbox before Dropbox



REFER YOUR FRIENDS AND GET UNLIMITED SPACE FREE!!!

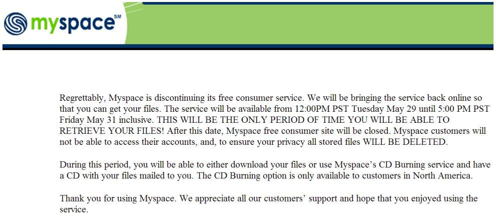
YOUR FREE DISK SPACE

- 300 MB **Free** Virtual Hard Drive
- Private and secure
- Share your files
- Collaborate on files
- 24/7 access

A success story

- MySpace offered 300MB «in the cloud»
- Unlimited account creation
- No Captcha
- $\rightarrow$ Creation of an unofficial API for automation
  - Split content in 300Mb chunks, create accounts, upload
  - Reconstruction infos spread through small files

Intermezzo: Myspace, Dropbox before Dropbox


Regrettably, Myspace is discontinuing its free consumer service. We will be bringing the service back online so that you can get your files. The service will be available from 12:00PM PST Tuesday May 29 until 5:00 PM PST Friday May 31 inclusive. THIS WILL BE THE ONLY PERIOD OF TIME YOU WILL BE ABLE TO RETRIEVE YOUR FILES! After this date, Myspace free consumer site will be closed. Myspace customers will not be able to access their accounts, and, to ensure your privacy all stored files WILL BE DELETED.

During this period, you will be able to either download your files or use Myspace's CD Burning service and have a CD with your files mailed to you. The CD Burning option is only available to customers in North America.

Thank you for using Myspace. We appreciate all our customers' support and hope that you enjoyed using the service.

The fall

- Income:
  - Ads: no money in Internet ads back then
  - Fremium model: didn't work for end users
- Costs:
  - Storage: expensive
  - Bandwidth: very, very, expensive
  - Quad damage from automation
- $\rightarrow$ business model was not sustainable

Intermezzo: Myspace, Dropbox before Dropbox

What can we learn from the MySpace story?

- Some people want to download (legal ?) content
- Too early ($\sim$10 years) for a viable commercial solution to emerge

A perfect setting

Circa 2000, the world is "P2P-ready"

- Means: plenty of downlink, untapped uplink
- Motive: retrieve content
- Opportunity: no competition

P2P: Key softwares

1998	Napster: P2P content (MP3) distribution
2000	Gnutella: decentralized search
2000	KaZaA/eDonkey: hierarchical indexing
2003	BitTorrent: efficient distribution
2006	R.I.P. Razorback 2.0 and rise of DHTs
2006-2009	PPLive/UUSee/TVAnts/. . . : P2P streaming

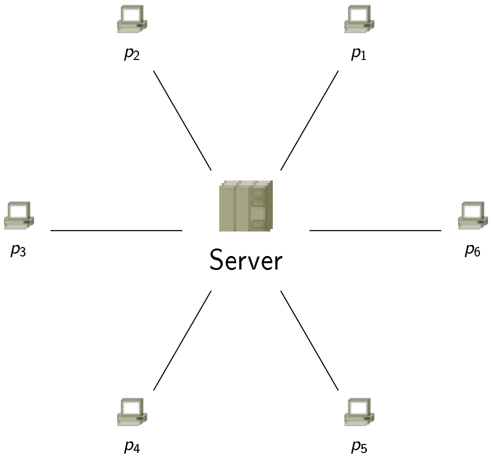
P2P: Key softwares

1998	**Napster: P2P content (MP3) distribution**
2000	Gnutella: decentralized search
2000	KaZaA/eDonkey: hierarchical indexing
2003	BitTorrent: efficient distribution
2006	R.I.P. Razorback 2.0 and rise of DHTs
2006-2009	PPLive/UUSee/TVAnts/. . . : P2P streaming

Napster

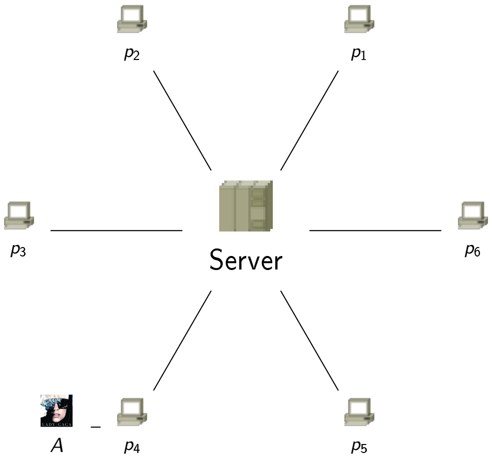
- Created in 1998 by Shawn "Napster" Fanning (The Italian Job) et Seam Parker (appears in The social network)
- Trial in 1999 (MPAA) - closed in July 2001
- Three steps system:
  1. Indexation: users upload the meta-data of their MP3 (catalog) to a server, `napster.com`
  2. Request: ask the server for some file (by name) ; receives a list of Napster clients (IP adresses) that claim hosting a compatible file
  3. Distribution: Download the MP3 from one client from the received list

Napster



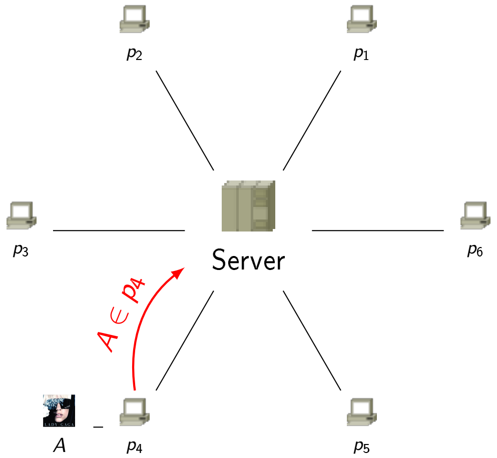
1. Indexation

Napster



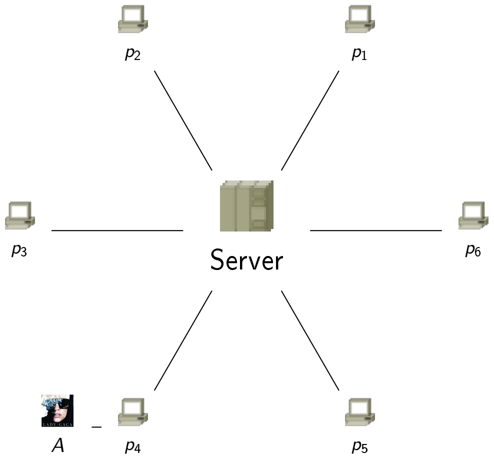
1. Indexation
   - p_4 owns A

Napster



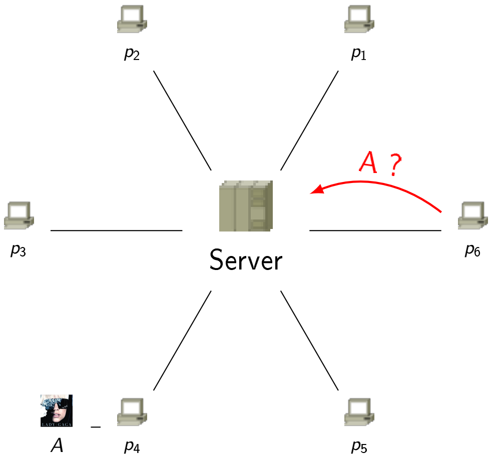
1. Indexation
   - p_4 owns A
   - p_4 notifies s

Napster



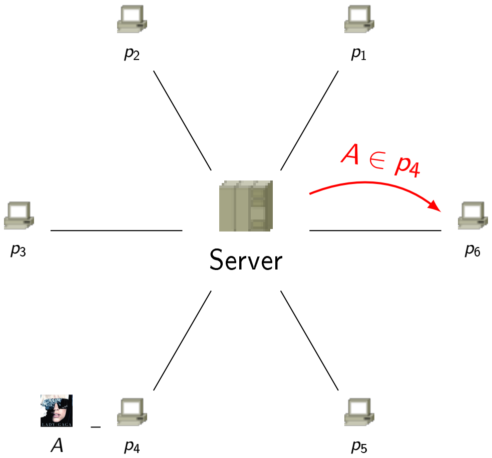
1. Indexation
   - p_4 owns A
   - p_4 notifies s
2. Request

Napster



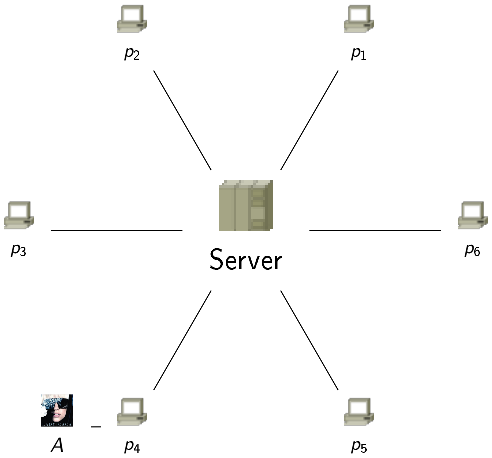
1. Indexation
   - p_4 owns A
   - p_4 notifies s
2. Request
   - p_6 asks s who owns A

Napster



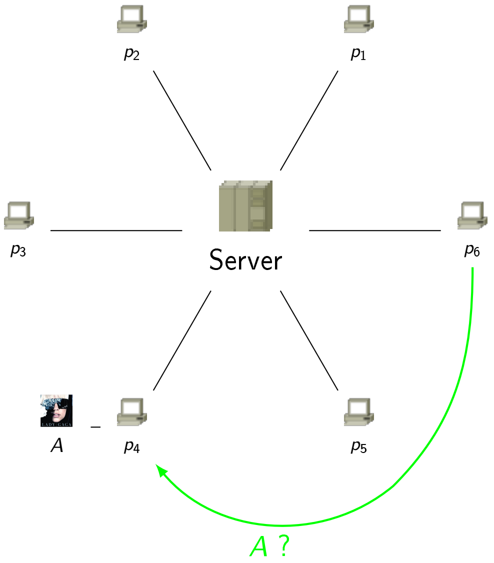
1. Indexation
   - p_4 owns A
   - p_4 notifies s
2. Request
   - p_6 asks s who owns A
   - s responds p_4 (IP)

Napster



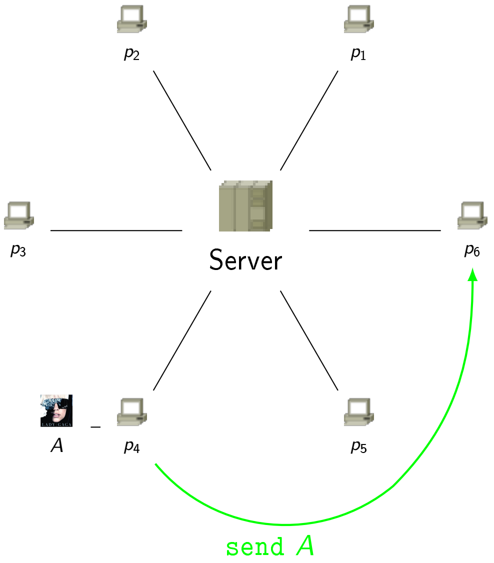
1. Indexation
   - p_4 owns A
   - p_4 notifies s
2. Request
   - p_6 asks s who owns A
   - s responds p_4 (IP)
3. Distribution

Napster



1. Indexation
   - p_4 owns A
   - p_4 notifies s
2. Request
   - p_6 asks s who owns A
   - s responds p_4 (IP)
3. Distribution
   - p_6 contacts p_4

Napster



1. Indexation
   - p_4 owns A
   - p_4 notifies s
2. Request
   - p_6 asks s who owns A
   - s responds p_4 (IP)
3. Distribution
   - p_6 contacts p_4
   - p_4 sends to p_6

Napster

Innovation

- Central indexation $\rightarrow$ complex requests (SQL queries, regex) are feasible
- Distribution from peer to peer (literally)

Limits

- Central index $\rightarrow$ single point of failure (indeed...)
- "Stupid" distribution (one-to-one, whole file)

P2P: Key softwares

1998	Napster: P2P content (MP3) distribution
2000	**Gnutella: decentralized search**
2000	KaZaA/eDonkey: hierarchical indexing
2003	BitTorrent: efficient distribution
2006	R.I.P. Razorback 2.0 and rise of DHTs
2006-2009	PPLive/UUSee/TVAnts/. . . : P2P streaming

Gnutella

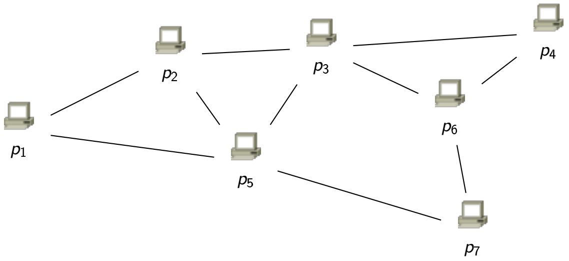
Histoiry

- 1st client : 2000 (nullsoft-AOL). Pulled-off after 1 day (!)
- Code available $\rightarrow$ open-source clones (LimeWire, Morpheus...)
- Biggest IP of the year in its time (quasi-extinct today)

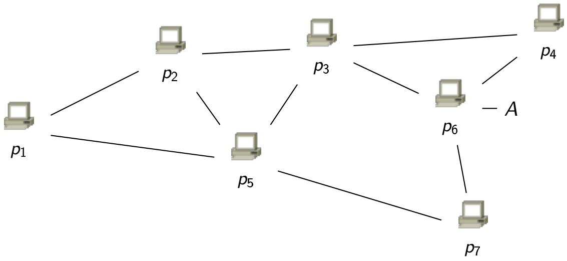
Protocol idea (0.4)

- Fully decentralized: no central index
- Searches are made by **flooding**
- Many enhancements afterwards (hash, ultrapeers, chunks...)

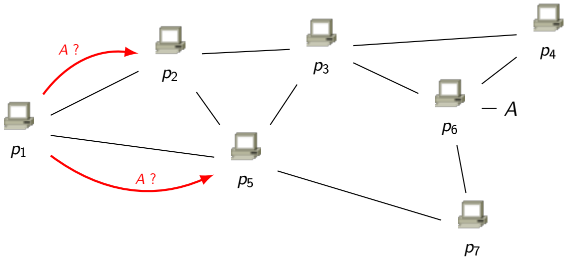
Gnutella



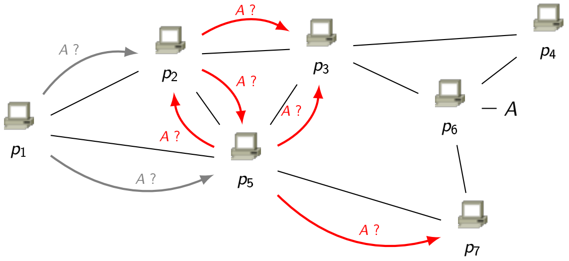
Gnutella



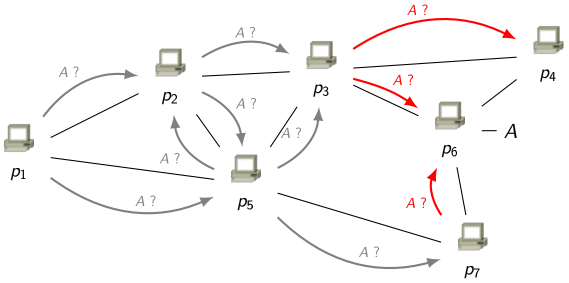
Gnutella



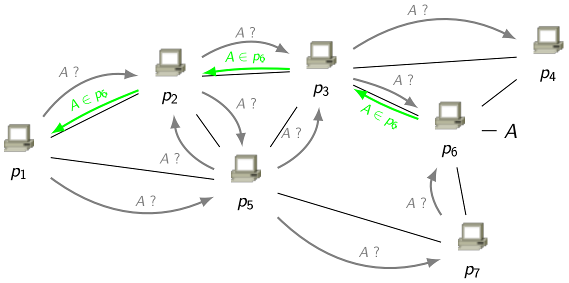
Gnutella



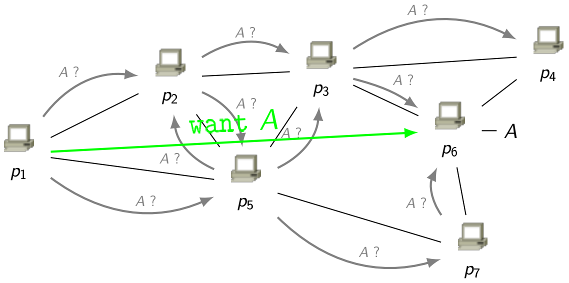
Gnutella



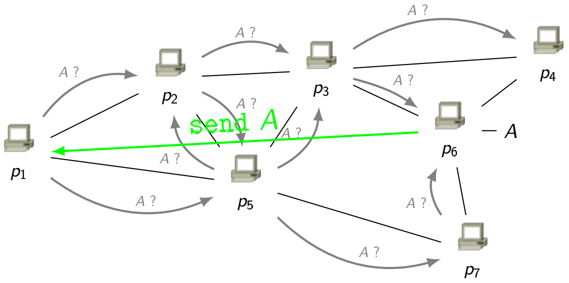
Gnutella



Gnutella



Gnutella



Gnutella

Innovation

No index, regex still feasible

Limits

- Flooding is BAD: for every request, only a fraction of the peers receives query
  - Too small fraction: partial response
  - Too large fraction: overhead (measured proportion of request messages: more than half traffic)
- Distribution is still "stupide"

P2P: Key softwares

1998	Napster: P2P content (MP3) distribution
2000	Gnutella: decentralized search
2000	**KaZaA/eDonkey: hierarchical indexing**
2003	BitTorrent: efficient distribution
2006	R.I.P. Razorback 2.0 and rise of DHTs
2006-2009	PPLive/UUSee/TVAnts/...: P2P streaming

KaZaA

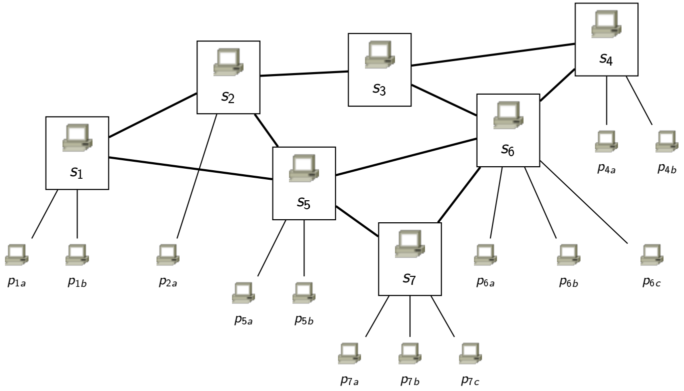
History

- 2000, Niklas Zennström et Janus Friis (Skype, Joost)
- Relies on the FastTrack overlay network (proprietary)
- Biggest traffic of the year, extinction

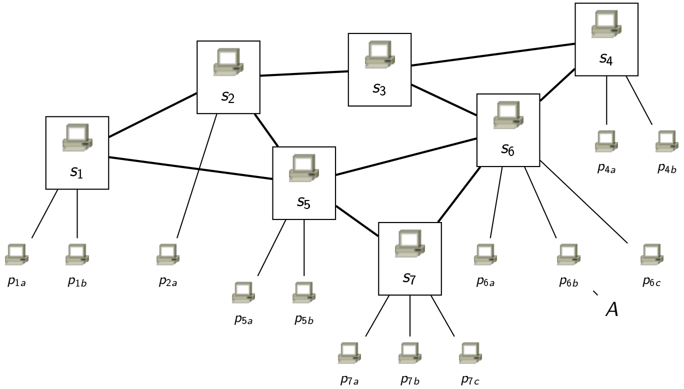
Key idea: **hierarchical index**

- Supernodes: peers with high bandwidth/availability
- Each ordinary node is attached to one supernode
- $\sim 100 - 1000$ ordinary nodes for one supernode
- Supernodes manage their nodes (indexation, requests) and communicate through flooding (Gnutella-style)
- Also, chunk-based distribution that relies on a reputation mechanism

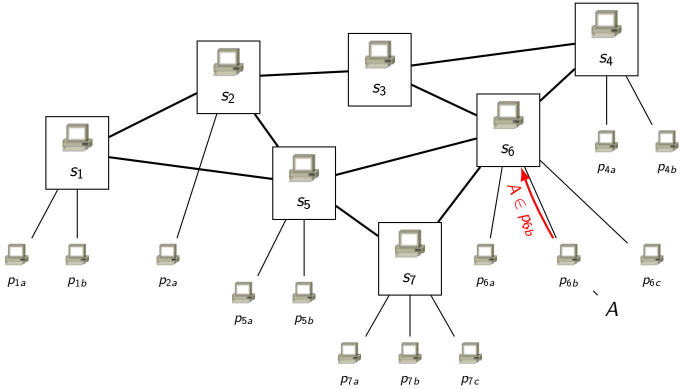
KaZaA


s_1
s_2
s_3
s_4
s_5
s_6
s_7
p_{1a}
p_{1b}
p_{2a}
p_{4a}
p_{4b}
p_{5a}
p_{5b}
p_{6a}
p_{6b}
p_{6c}
p_{7a}
p_{7b}
p_{7c}

KaZaA

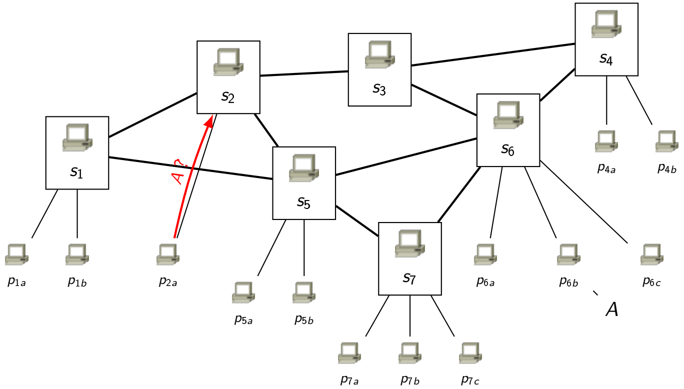

s_1
s_2
s_3
s_4
s_5
s_6
s_7
p_{1a}
p_{1b}
p_{2a}
p_{5a}
p_{5b}
p_{7a}
p_{7b}
p_{7c}
p_{6a}
p_{6b}
p_{6c}
p_{4a}
p_{4b}
A

KaZaA

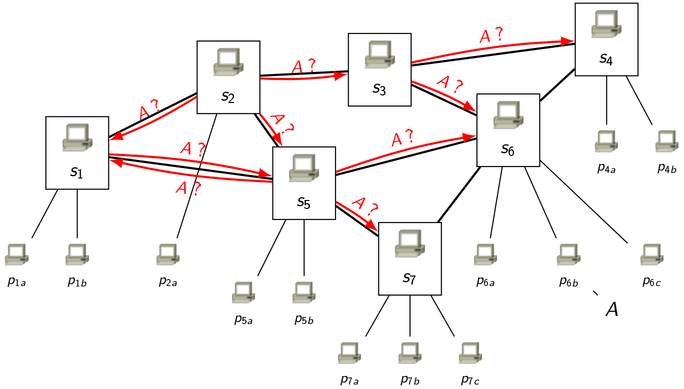


s_1
s_2
s_3
s_4
s_5
s_6
s_7
p_{1a}
p_{1b}
p_{2a}
p_{5a}
p_{5b}
p_{7a}
p_{7b}
p_{7c}
p_{6a}
p_{6b}
p_{6c}
p_{4a}
p_{4b}
$A \in p_{6b}$
A

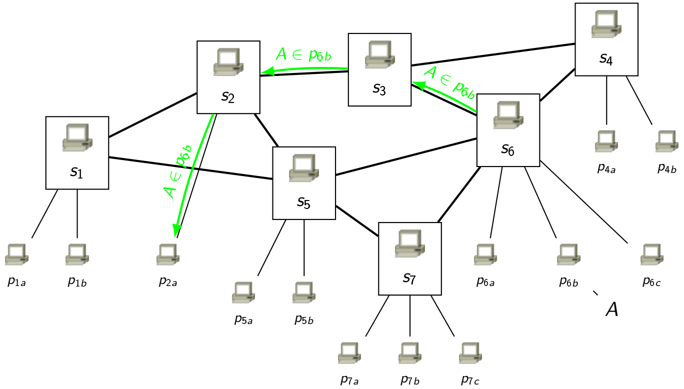
KaZaA


s_1
s_2
s_3
s_4
s_5
s_6
s_7
p_{1a}
p_{1b}
p_{2a}
$A?$
p_{5a}
p_{5b}
p_{7a}
p_{7b}
p_{7c}
p_{6a}
p_{6b}
p_{6c}
A
p_{4a}
p_{4b}

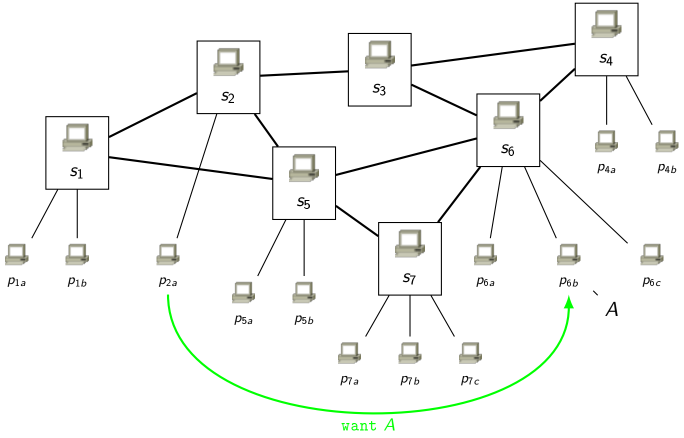
KaZaA


s_1
s_2
s_3
s_4
s_5
s_6
s_7
A ?
p_{1a}
p_{1b}
p_{2a}
p_{5a}
p_{5b}
p_{7a}
p_{7b}
p_{7c}
p_{6a}
p_{6b}
p_{6c}
p_{4a}
p_{4b}
A

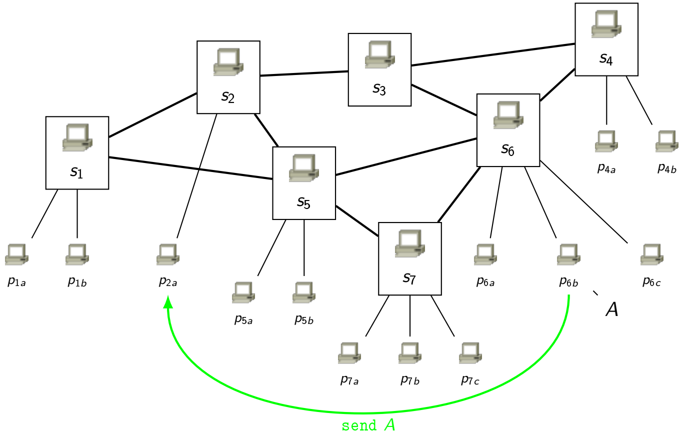
KaZaA


s_1
s_2
s_3
s_4
s_5
s_6
s_7
$A \in p_{6b}$
$A \in p_{6b}$
$A \in p_{6b}$
p_{1a}
p_{1b}
p_{2a}
p_{4a}
p_{4b}
p_{5a}
p_{5b}
p_{6a}
p_{6b}
p_{6c}
p_{7a}
p_{7b}
p_{7c}
A

KaZaA


s_1
s_2
s_3
s_4
s_5
s_6
s_7
p_{1a}
p_{1b}
p_{2a}
p_{4a}
p_{4b}
p_{5a}
p_{5b}
p_{6a}
p_{6b}
p_{6c}
p_{7a}
p_{7b}
p_{7c}
A
want A

KaZaA


s_1
s_2
s_3
s_4
s_5
s_6
s_7
p_{1a}
p_{1b}
p_{2a}
p_{4a}
p_{4b}
p_{5a}
p_{5b}
p_{6a}
p_{6b}
p_{6c}
p_{7a}
p_{7b}
p_{7c}
A
send A

eMule - eDonkey

History

- eDonkey2000 client released in. . . 2000
- MetaMachine corp., closed in 2005 (RIAA)
- Documented protocol → open-source clones (eMule)
- Biggest traffic. . . (still active today)

Principle

- eDonkey servers: machines dedicated to indexation ("professional" supernodes)
- Chunk-based distribution that relies on a queuing system

KaZaA/eDonkey

Innovation

- Hierarchical indexing
- Chunk (overlay data “packet”)

Limitations

- KaZaA: sequential chunk scheduler $\rightarrow$ “missing chunk” syndrome
- KaZaA Lite K++: fake reputation, always supernodes
- eDonkey: Saturation of queues $\rightarrow$ very large delays
- Servers / supernodes $\rightarrow$ points of failures

P2P: Key softwares

1998	Napster: P2P content (MP3) distribution
2000	Gnutella: decentralized search
2000	KaZaA/eDonkey: hierarchical indexing
2003	BitTorrent: efficient distribution
2006	**R.I.P. Razorback 2.0 and rise of DHTs**
2006-2009	PPLive/UUSee/TVAnts/...: P2P streaming

KAD vs Razorback 2.0

Kad

RazorBack 2.0

The circle of life

KAD vs Razorback 2.0

Kad

- Goal: avoid (semi-)centralized indexing AND flooding
- Solution: Distributed Hash Tables, introduced in 2001 (research papers)
- The index is distributed over all peers
- One DHT, Kad, is implemented in eMule quite quickly

RazorBack 2.0

The circle of life

KAD vs Razorback 2.0

Kad

RazorBack 2.0

- A race for the best eDonkey server begins in the early 2000s
- One clear winner: Razorback 2.0
- Other servers become useless (spam, small catalog and low reactivity)

The circle of life

KAD vs Razorback 2.0

Kad

RazorBack 2.0

The circle of life

- At start, Kad is not popular (R2 is much faster)
- February 21, 2006, forced shutdown of Razorback2.X
- Lots of users switch to Kad, the only viable alternative. . .
- . . . and **it works!** (supports the load explosion)

Distributed Hash Tables

Innovation

Fully decentralized index

Limitation

- Only simple combinations of exact requests (~~regexp~~)
- Logarithmic price (delay and maintenance)

P2P: Key softwares

1998	Napster: P2P content (MP3) distribution
2000	Gnutella: decentralized search
2000	KaZaA/eDonkey: hierarchical indexing
2003	**BitTorrent: efficient distribution**
2006	R.I.P. Razorback 2.0 and rise of DHTs
2006-2009	PPLive/UUSee/TVAnts/. . .: P2P streaming

BitTorrent

History

- Founding paper and client proposed by Bram Cohen in 2003
- Documented protocol
- Lots of implementations (vanilla, Azureus/Vuze, μtorrent...)
- Biggest traffic (by years) up to Youtube take-over. Still biggest P2P traffic

Key idea: focus on the content to distribute

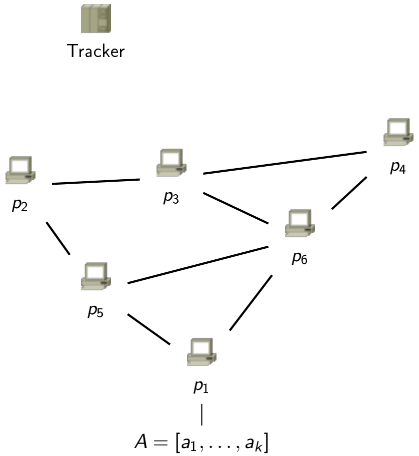
- One dedicated overlay per content: the swarms
- A swarm is managed by a tracker
- Tit-for-Tat incentive policy: Give and you shall receive
- + 2/3 others simple yet efficient tricks

BitTorrent

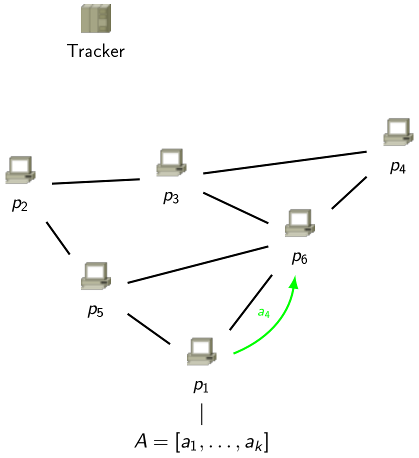
Protocol overview

- The tracker monitors participating peers
- On arrival, a peer receives neighbors
- Chunks are exchanged until download completion

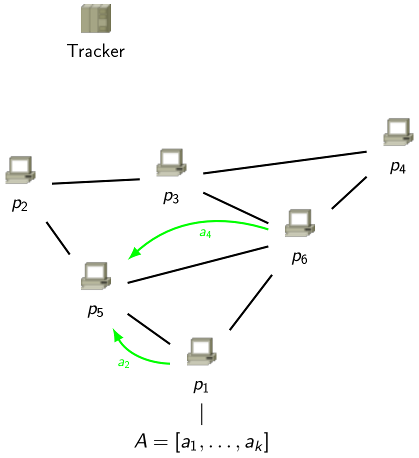
BitTorrent



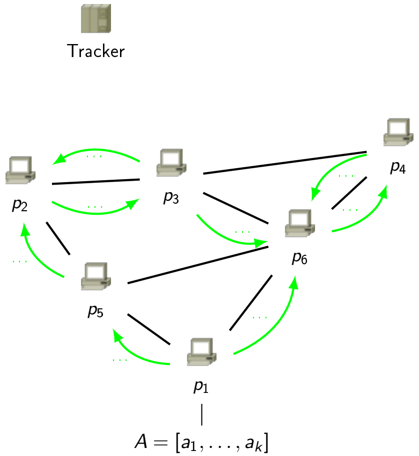
BitTorrent



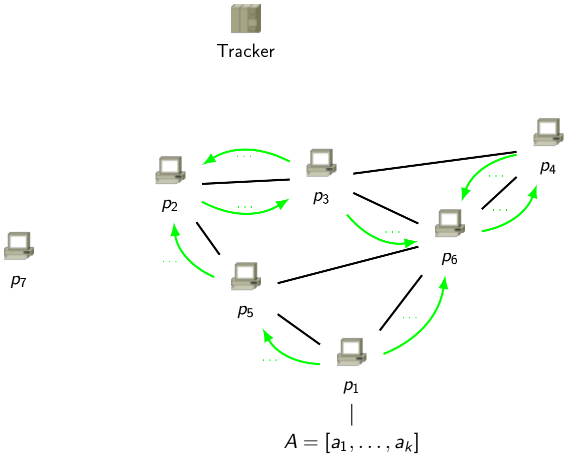
BitTorrent



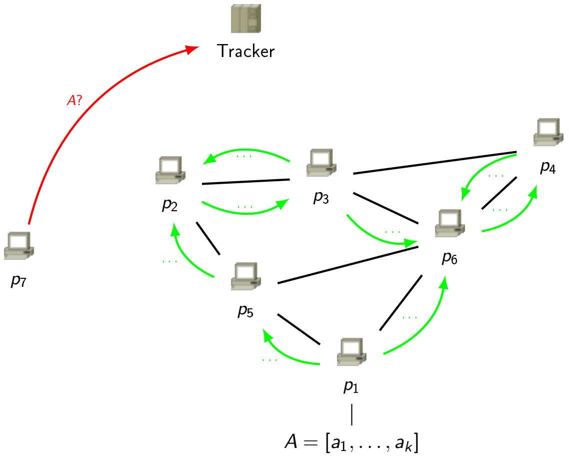
BitTorrent



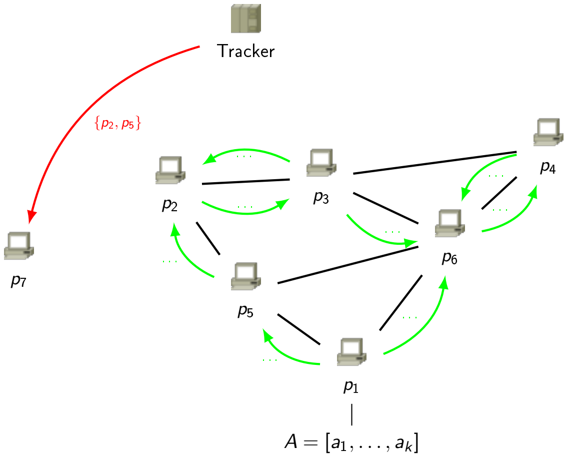
BitTorrent



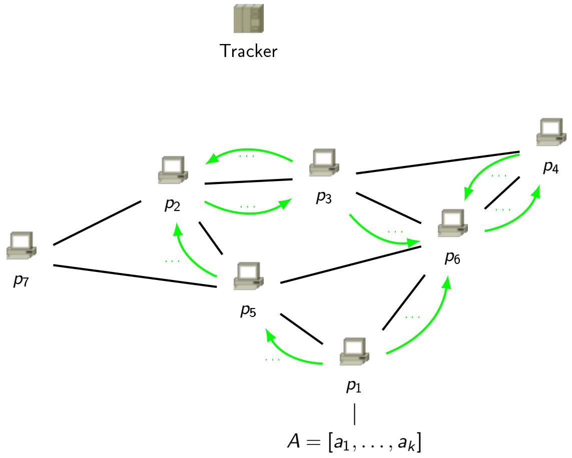
BitTorrent



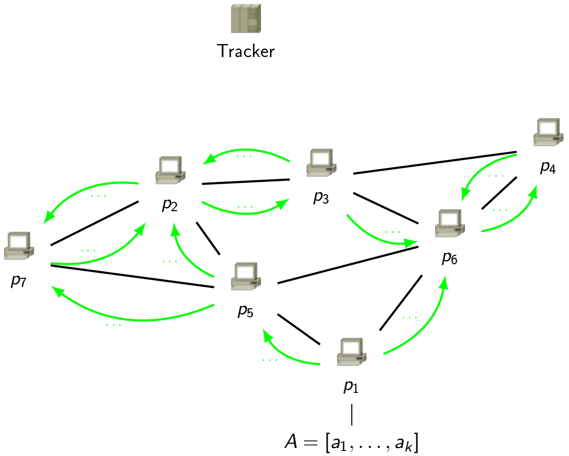
BitTorrent



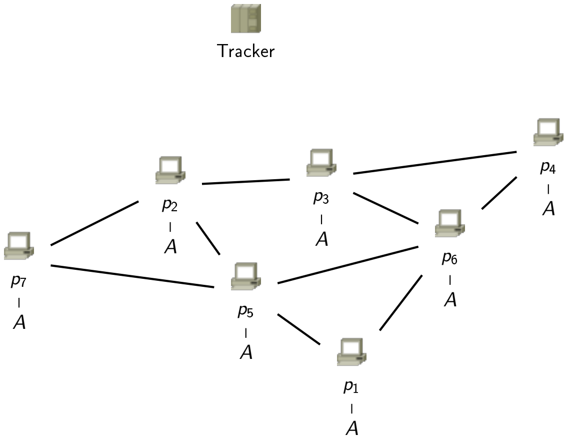
BitTorrent



BitTorrent



BitTorrent



BitTorrent

Innovation

- Content-centric instead of user-centric
- Top distribution algorithms

Limitations

- Trackers $\rightarrow$ points of failures
- Swarm lifespan is smaller than eMule retention

P2P: Key softwares

1998	Napster: P2P content (MP3) distribution
2000	Gnutella: decentralized search
2000	KaZaA/eDonkey: hierarchical indexing
2003	BitTorrent: efficient distribution
2006	R.I.P. Razorback 2.0 and rise of DHTs
2006-2009	**PPLive/UUSee/TVAnts/. . . : P2P streaming**

P2PTV

History

- Goal: sport events
- Many proprietary incompatible solutions (UUSee, PPLive, TVAnts,. . .)
- Growth killed by YouTube and CDN-like solutions
- Mostly used today in Asia and “expat” communities

Idea: lower the delay

- Minimize buffer length
- Optimize chunk diffusion
- Boost the system with actual servers

P2PTV

Innovation

Algorithms for delay optimization

Limitations

- The logarithmic price
- Youtube, Akamai, Netflix

Part III

B) Distributed Hash Tables (DHTs)

Distributed Hash Tables

Motivation

Find where a content is at low cost

Problems to solve

Decentralisation	the burden must be shared among all peers
Scalability	search must work on very large systems
Dynamicity	peers arrive and leave, nicely. . . or not!

Distributed Hash Tables

Hash function

Function $h : E \mapsto F$
E = anything ! (file contents, file names, IP addresses...)
F = key space (circle $[0, ..2^n[$, square, hypercube...)

Principle of a DHT (Distributed Hash Table)

- Each peer has an identifier (key / hash)
- Each data has an identifier (key / hash)
- Each peer is in charge of an area of F (things whose key is inside the area)
- Find a key $\rightsquigarrow$ find a route to the peer in charge of that key
- Lots of variants depending on F and the routing mechanism
  - CAN, Pastry/Tapestry, Chord,...

One important thing

Managing a key does not mean possessing the content!

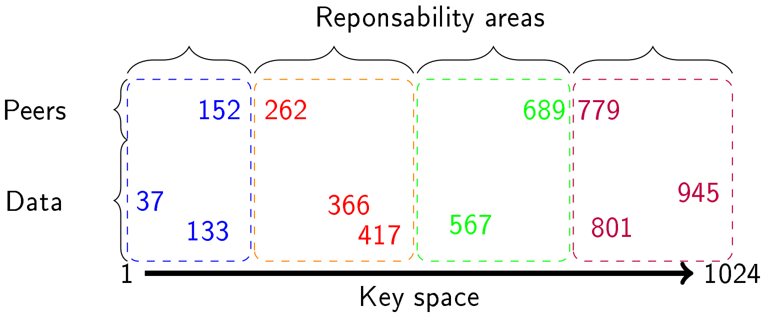
One important thing

Managing a key does not mean possessing the content!

Managing a key is knowing:

- what corresponds to the key
- associated meta-data (IP addresses, . . .)

Toy example: nearest neighbor indexation



Peer with key 152 manages data with keys 37 and 133...

Request example: “House of the Dragon”

1. h(“House”) = 1234
2. Search the peer nearest to 1234
3. It is the peer with hash 1100
4. Ask for a list of contents
5. h(“Dragon”) = 5678
6. Nearest peer: 5600 $\rightarrow$ request $\rightarrow$ contents
7. Intersection of contents
8. Display
9. User chooses content with hash 6666
10. Peer with hash nearest to 6666: 6667
11. Ask list of IPs to 6667
12. Download from the IPs

Routing by key

Principle of the routing algorithm

- Peer P looks for data with key c
- Data is indexed at peer Q
- P knows a few neighbors
- P transmits the request to the neighbor that is «closest» to c
- The neighbor re-transmits, etc.
- Until Q is reached

Performance

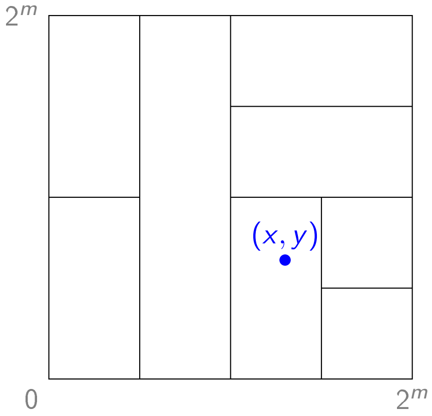
- We should try to make it in $O(\log n)$ forwards (jumps)
- Looks a bit like dichotomic search
- No more flooding!

Routing by key

How to do that?

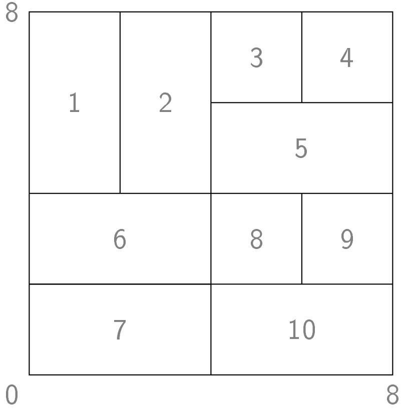
- The overlay <u>tries</u> to have some **topology** :
  - CAN : multi-dimensional torus
  - Chord : circle (with... chords)
  - Viceroy : butterfly graph
  - Tapestry/Pastry/Kademlia : hypercube
  - D2B : de Bruijn graph
- We use the topology to navigate the graph
  - From neighbor to neighbor
  - At each step we get closer to destination
- If the graph diameter is $O(\log n)$...
- ... We should reach destination in $O(\log n)$ steps!

Content-Addressable Network (CAN)

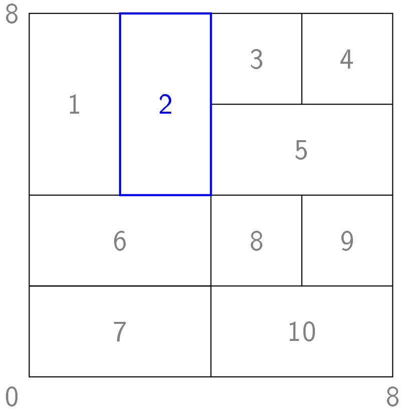


- F : hypercube with d dimensions ($d = 2$)
- Each peer manages a rectangle
- d hash functions $\rightsquigarrow$ each key is d-uplet (a pair) of hashs
- Routing to the adjacent neighbor of closest area for Manhattan distance

Example



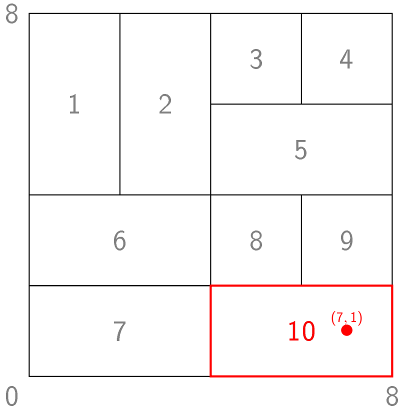
Example



Routing table of 2

Peer	Area
2	$[2,4[\times[4,8[$
1	$[0,2[\times[4,8[$
3	$[4,6[\times[6,8[$
5	$[4,8[\times[4,6[$
6	$[0,4[\times[2,4[$

Example

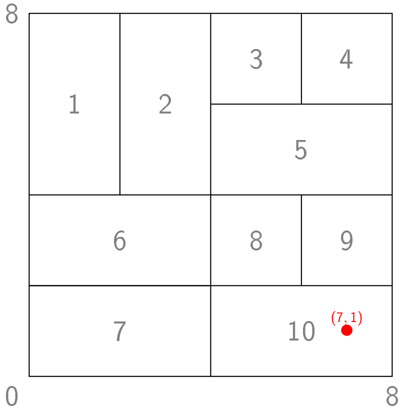


Routing table of 2

Peer	Area
2	$[2,4[\times[4,8[$
1	$[0,2[\times[4,8[$
3	$[4,6[\times[6,8[$
5	$[4,8[\times[4,6[$
6	$[0,4[\times[2,4[$

- Peer 2 looks for key $(7,1)$, possessed by peer 10.

Example

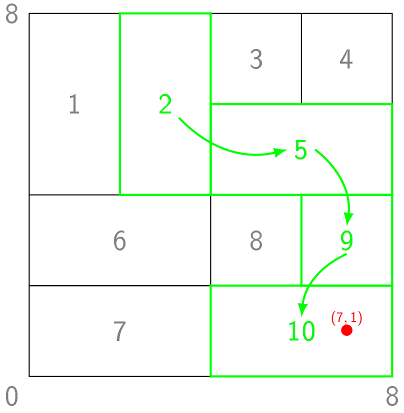


Routing table of 2

Peer	Area
2	$[2, 4[\times[4, 8[$
1	$[0, 2[\times[4, 8[$
3	$[4, 6[\times[6, 8[$
5	$[4, 8[\times[4, 6[$
6	$[0, 4[\times[2, 4[$

- Peer 2 looks for key $(7, 1)$, possessed by peer 10.
- Distance to destination : $2 \rightarrow 4 + 3 = 7, 1 \rightarrow 6 + 3 = 9, 3 \rightarrow 2 + 5 = 7, 5 \rightarrow 0 + 3 = 3, 6 \rightarrow 4 + 1 = 5$.

Example



Routing table of 2

Peer	Area
2	$[2,4[\times[4,8[$
1	$[0,2[\times[4,8[$
3	$[4,6[\times[6,8[$
5	$[4,8[\times[4,6[$
6	$[0,4[\times[2,4[$

- Peer 2 looks for key $(7,1)$, possessed by peer 10.
- Distance to destination : $2 \rightarrow 4+3=7, 1 \rightarrow 6+3=9, 3 \rightarrow 2+5=7, 5 \rightarrow 0+3=3, 6 \rightarrow 4+1=5$.
- Route from 2 to $(7,1)$: $2 \rightarrow 5 \rightarrow 9 \rightarrow 10$.

Arrivals and departures

- Arrival of A : A takes a key (X, Y) and contacts an entry peer B (bootstrap)
  - A contacts through B the peer C in charge of (X, Y)
  - C splits its area with A (they adjust their routing tables)
  - A and C contact the neighbors for updating their tables
- Departure of A : A determines who should take over and contact its neighbors
  - All neighbors are notified / updated
  - The area may be dispatch among several neighbors

Properties

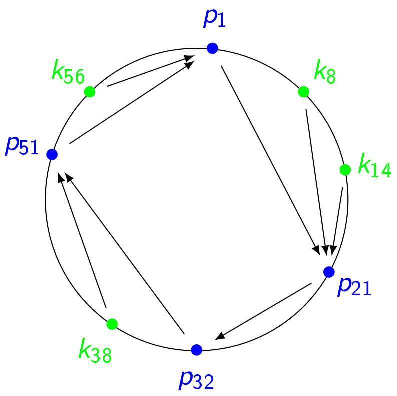
- Easy to understand, easy to extend
  - Overlapping areas for more robustness
  - Dimension d can be larger
  - Dynamic reshaping of areas to balance the load
- Size of routing table: $O(d)$ ☺
- Number of jumps: $O(dn^{1/d})$ ☹
- In the end: useless (costly in messages, slow)

The Chord protocol

- Always the same recipe...
- Hash function: SHA-1
- $F = [0, ..2^m[(m = 160)$ seen as a circle modulo 2^m: the Chord ring
- Routing building block: successor(k) = first peer with key strictly greater than k modulo 2^m

Successors

A peer manages all the keys of which it is the successor

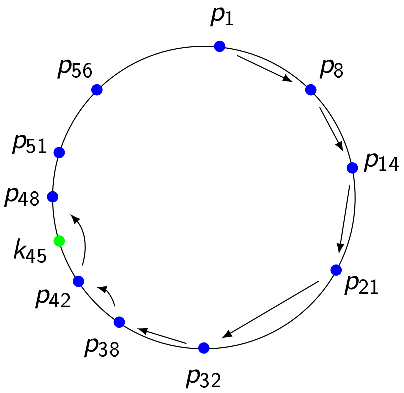


The maths corner

With high probability, for N peers and K keys:

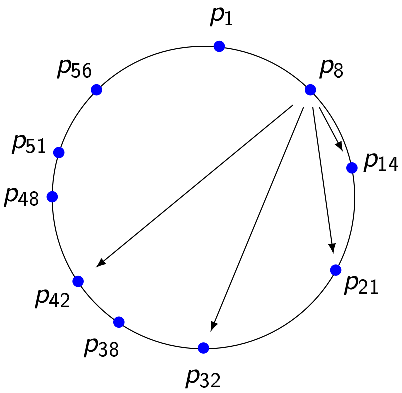
- Each peer manages at most $\frac{(1+C)K}{N}$ keys
- $C = O(\ln(N))$
- Is a peer arrives or leaves, at most $O(\frac{K}{N})$ keys need to change successor

Successor routing



- Example: 1 looks 45 (managed by 48)
- ☹☹☹

Solution : finger table

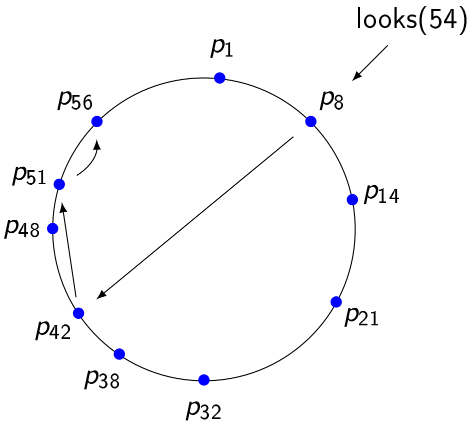


- finger(k) : successor of $n + 2^{k-1}$ modulo 2^m

Finger table of p_8

$S(8 + \{2^0, 2^1, 2^2\})$	p_{14}
$S(8 + \{2^3\})$	p_{21}
$S(8 + \{2^4\})$	p_{32}
$S(8 + \{2^5\})$	p_{42}

Finger table routing



A fast routing

- Each jump halves the distance (more or less)
- Theorem: with high probability, the number of jumps to locate the successor of a key is $O(\ln(N))$
- Theorem: with high probability, the number of entries in the finger table is $O(\ln(N))$

Departure and arrivals

- On arrival, a peer N contacts an entry point A (bootstrap)
  - A looks B, predecessor of N
  - N becomes the successor of B
  - N contacts B, retrieve the table
  - N contacts the table so all can adjust entries
  - N contacts its successor to share the keys
- On departure, a peer contacts its predecessors (finger included) to notify the leave. Keys are sent to the successor. Predecessor is connected to successor.

Chord properties

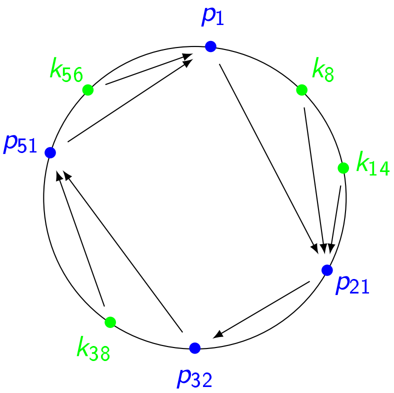
- Correct even if finger are corrupted (worst case: successor routing)
- Redundancy feasible to avoid data loss (e.g. host copy of the keys of predecessor and successor)
- Possibility to cache request results to accelerate future requests
- Relatively easy to understand (yes, yes!)

Viceroy

- Goal: Chord with constant degree
- $F = [0, ..2^m[$ as a ring modulo 2^m, like Chord
- Same base routing: $\underline{\text{successor}(k)}$ and $\underline{\text{predecessor}(k)}$

Successors

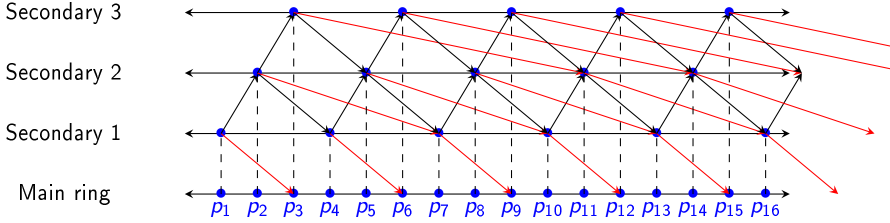
A peer manages all the keys of which it is the successor



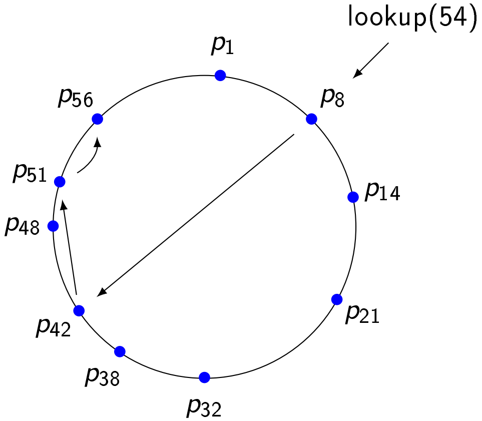
Viceroy approach: butterfly graph

- Each node belongs, in addition to the main ring, to one of the $\log(n)$ secondary rings
- A node of level p has 7 neighbors :
  - 2 neighbors in the main ring ;
  - 2 neighbors in the secondary ring ;
  - Two elevators ;
  - One long link to $succ(x + 2^p)$ in the next lower ring.

“Butterfly” graph


Secondary 3
Secondary 2
Secondary 1
Main ring
p_1 p_2 p_3 p_4 p_5 p_6 p_7 p_8 p_9 p_{10} p_{11} p_{12} p_{13} p_{14} p_{15} p_{16}

Reminder: Chord routing

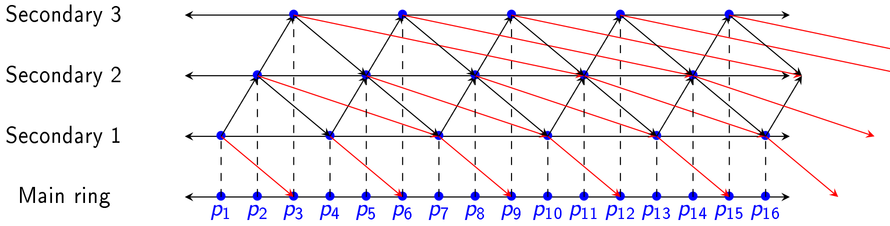


Butterfly routing

Idea: mimick Chord

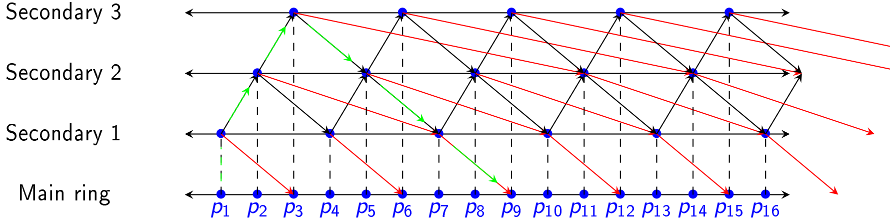
- Always: if one risks to exceed, one goes on the main ring
- One starts by going up to the most powerful level;
- Then we go down the levels one by one, using
  - The long link if possible ;
  - The normal elevator if not.
- Total cost: some $\log(n)$.

Butterfly routing



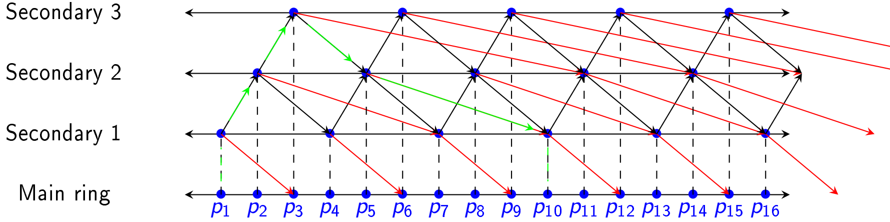
Butterfly routing

$p_1 \rightarrow p_9$



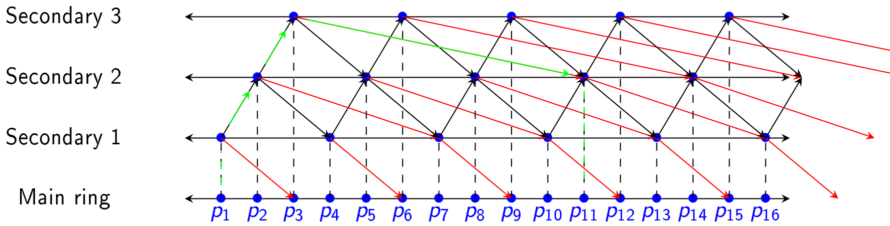
Butterfly routing

$p_1 \rightarrow p_{10}$



Butterfly routing

$p_1 \rightarrow p_{11}$



Secondary 3
Secondary 2
Secondary 1
Main ring
p_1 p_2 p_3 p_4 p_5 p_6 p_7 p_8 p_9 p_{10} p_{11} p_{12} p_{13} p_{14} p_{15} p_{16}

Properties

- Easy to understand... if you know Chord well
- Table size: $O(1)$ ☺☺
- Number of jumps: $O(\log(n))$ ☺
- So, it's perfect! Is it?

Hypercube topology

Idea present in Tapestry, Pastry and Kademlia

Main idea

- Each peer P has a unique hashcode $H(P) = h_1 h_2 ... h_m$ (e.g. $m = 160$: SHA-1)
- Each peer P tries to know one neighbor by prefix:
  - $\overline{h_1}$
  - $h_1 \overline{h_2}$
  - $h_1 h_2 \overline{h_3}$
  - ...
  - $h_1 h_2 ... h_{v-1} \overline{h_v}$
- $v - 1$: longest common prefix with P
- $v = O(\log N)$ in average (thanks to uniform hash distribution)

Hypercube topology

Idea present in Tapestry, Pastry and Kademlia

Example

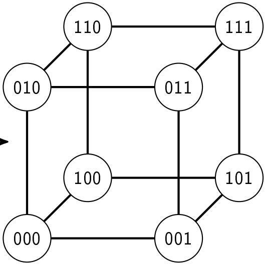
- Peer with hashcode 01000100100011010110100101 01...01 has neighbors voisins :
  - **1**100100100111011010101010...10
  - **00**00100111101001101010011...11
  - **011**101000101010011110010 0...11
  - ...
  - **01000101**01001001000101000...00
- $v = 8$: 7 common bits with the nearest neighbor
- About 2^8 peers in the system

Hypercube topology

Cube toy example:
If v is the same for all, the routing graph is the hypercube in dimension v

- $p_1 = \mathbf{010}101001001$
- $p_2 = \mathbf{011}011101010$
- $p_3 = \mathbf{000}000110101$
- $p_4 = \mathbf{110}110110010$
- $p_5 = \mathbf{111}001111000$
- $p_6 = \mathbf{001}010101001$
- $p_7 = \mathbf{101}110010100$
- $p_8 = \mathbf{100}011001110$

$\Rightarrow$

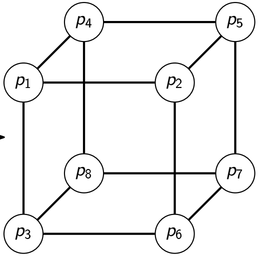


Hypercube topology

Cube toy example:
If v is the same for all, the routing graph is the hypercube in dimension v

- $p_1 =$ **010**101001001
- $p_2 =$ **011**011101010
- $p_3 =$ **000**000110101
- $p_4 =$ **110**110110010
- $p_5 =$ **111**001111000
- $p_6 =$ **001**010101001
- $p_7 =$ **101**110010100
- $p_8 =$ **100**011001110

$\Rightarrow$



Cube dimension?

v is not the same everywhere!

- Average: $v_{moy} = \log N$
- Worst case: $v_{max} = 2 \log N$

Plaxton Mesh routing algorithm

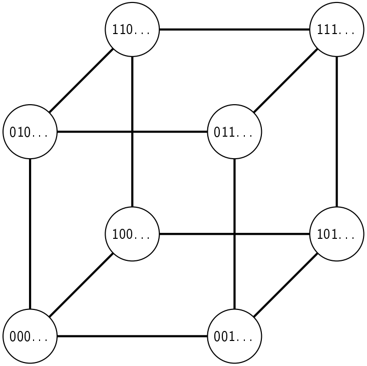
- Intuition: “dial” the destination
- Incremental search, one digit at a time
- Get closer to the destination at each step
- If you cannot get closer, you are in charge!

Routing algorithm

Peer P with hashcode $H(P) = h_1 h_2 ... h_m$ looks for D with hashcode $H(D) = d_1 d_2 ... d_m$

$H(P)$ XOR $H(D)$ tells the first diverging digit

Forward to neighbors : move in the hypercube

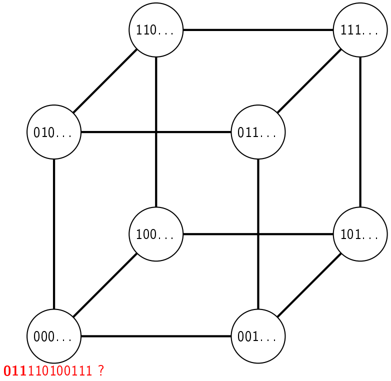


Routing algorithm

Peer P with hashcode $H(P) = h_1 h_2 ... h_m$ looks for D with hashcode $H(D) = d_1 d_2 ... d_m$

$H(P)$ XOR $H(D)$ tells the first diverging digit

Forward to neighbors : move in the hypercube

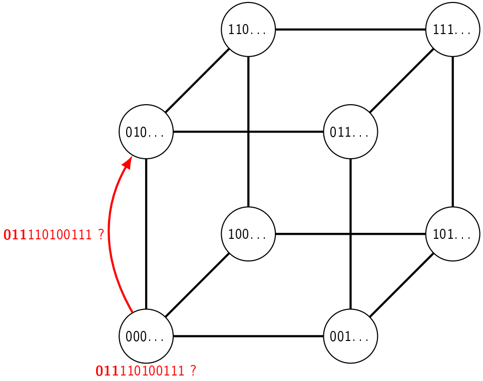


Routing algorithm

Peer P with hashcode $H(P) = h_1 h_2 ... h_m$ looks for D with hashcode $H(D) = d_1 d_2 ... d_m$

$H(P)$ XOR $H(D)$ tells the first diverging digit

Forward to neighbors : move in the hypercube

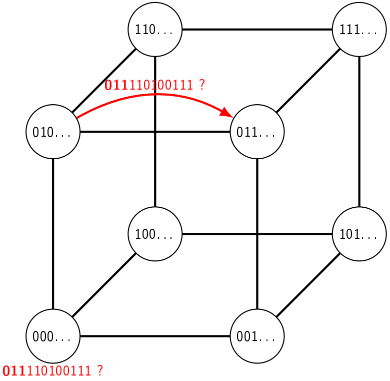


Routing algorithm

Peer P with hashcode $H(P) = h_1 h_2 ... h_m$ looks for D with hashcode $H(D) = d_1 d_2 ... d_m$

$H(P)$ XOR $H(D)$ tells the first diverging digit

Forward to neighbors : move in the hypercube

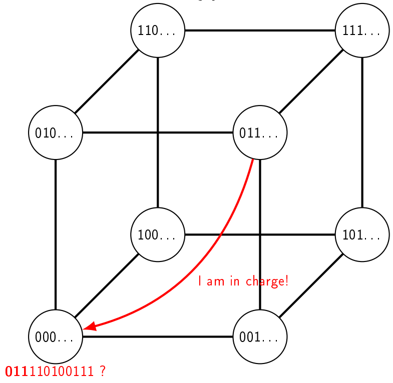


Routing algorithm

Peer P with hashcode $H(P) = h_1 h_2 ... h_m$ looks for D with hashcode $H(D) = d_1 d_2 ... d_m$

$H(P)$ XOR $H(D)$ tells the first diverging digit

Forward to neighbors : move in the hypercube



Routing algorithm

Peer P with hashcode $H(P) = h_1 h_2 ... h_m$ looks for D with hashcode $H(D) = d_1 d_2 ... d_m$
$H(P)$ XOR $H(D)$ tells the first diverging digit
Forward to neighbors : move in the hypercube

Recap:

for N peers

- Size of routing tables: $O(\log N)$ peers
- Look up in $O(\log N)$ jumps

Speeding up lookup

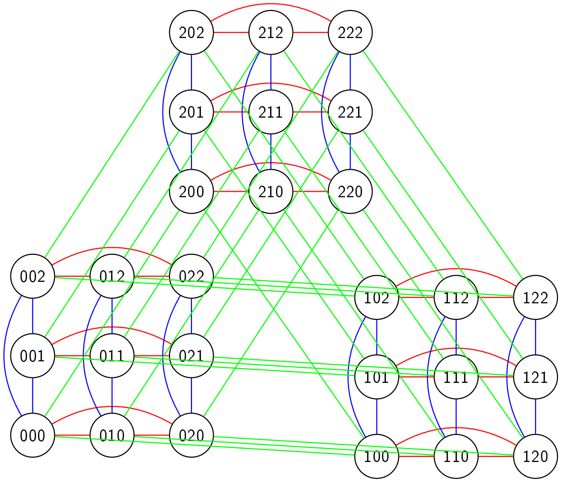
Bigger digit base

Instead of binary digits (base 2) we can use base b

- Benefit: routing in $\log_b(N)$ instead of $\log_2(N)$: speed gain of $\frac{\ln b}{\ln 2}$
- Issue: routing table sizes are multiplied by a factor b ($b - 1$ neighbors against 1 for each dimension)

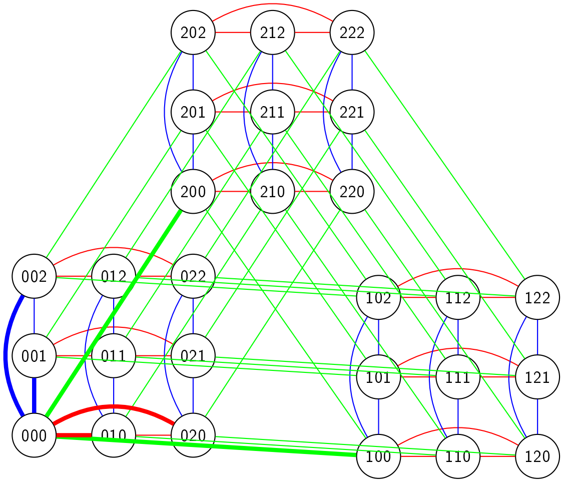
Speeding up lookup

Example in base 3, dimension 3:



Speeding up lookup

Example in base 3, dimension 3:



Better than $\log N$?

How to break the $\log N$ frontier?

If one chooses $b = \ln N$, assume redundancy k

- Lookups in $O(\frac{\ln N}{\ln \ln N})$ (5,2 for $N = 1000000$)
- $k \times O(\frac{(\ln N)^2}{\ln \ln N})$ contacts per peer (72×k pour $N = 1000000$)

Speeding up lookup

Number of neighbors

Instead of 1 neighbor per digit, we keep k for redundancy

- Benefit: cope with dynamics
- Issue: tables grow with factor k

Multiple publication

Data is managed by the k closest peers (not just the closest)

Table size

$N = 2\ 000\ 000$ pairs, $b = 8$ (3 bits), $k = 20$ copies :
$\frac{\ln 2^{21}}{\ln 2^3} * (2^3 - 1) * 20 = 7 * 7 * 20 = 980$ neighbors. Doable.

Insertion?

Algorithm

1. Draw your hashcode H in $[0..2^m[$
2. Look up H by dialing...
3. ...and get your neighbors in the process!
4. Contact your neighbors to initiate connections and possibly update tables
5. Retrieve the indexes you are in charge of

Bootstrap

Always the same problem!
You need to know one entry point

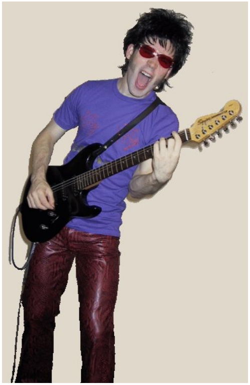
Analysis

Same complexity than a lookup: $O(\log_b(N))$

Kademlia [Maymounkov Mazières 2002]

Full (end-to-end) implementation of the hypercube: Overnet, Kad.

Petar Maymounkov
(MIT)



David Mazières
(Stanford)

XOR distance

A XOR B ($A \oplus B$) is seen as a number.

XOR distance

- **distance:** $(A \oplus B) + (B \oplus C) \geq (A \oplus C)$
  (car $(A \oplus B) \oplus (B \oplus C) = A \oplus C$ et $X + Y \geq X \oplus Y$)
- **unidirectional** : the sphere with center A and radius r contains **one unique** point $B = A \oplus r$

Bucket

Set of k points at distance $[2^i, 2^{i+1}[\simeq$ neighbors through one edge in the hypercube

Routing table

Routing table

Each pair has 160 drawers. Those of number $\geq \log N$ are empty in general (and those $\geq 2 \log N + cst$ empty w.h.p)
Content of an entry: triple (IP, UDP port, hashcode of the peer)

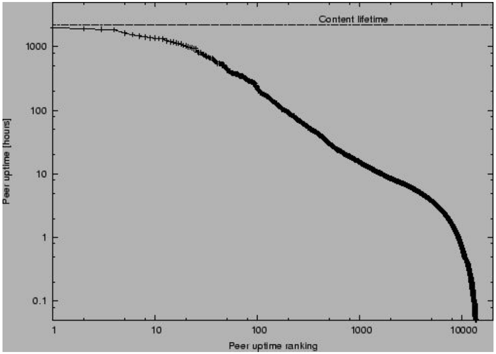
Bucket maintenance: enhanced LRU

- A drawer contains at most k addresses of peers
- Fill the drawer (know new peers)
  - answers to requests
  - requests transmitted
- Discard overflowing peers
  - Keep the old peers rather than the recent ones
  - However remove those that no longer respond
  - One ping per hour

Why keep the old peers in the bucket?

Uptime law

Probability to stay alive for T is a power law



Uptime from 53833 pairs on a "Beyond Good and Evil" torrent [Pouwelse 2004]

Complete Protocol

4 types of packets (UDP)

1. ping
2. store : order to store (key, value)
3. lookup: must return k closest peers known
4. key search: return the key or the k closest key known

Parallel requests to dial

- First shot: 3/20, in parallel
- If failure: 17/20 (in parallel)
- Continue on the first answer

Other perks

Request caching

- Benefit: faster response, load balancing on popular keys
- Issue: handling cache size!

Solution: expiration date exponential with distance (keep twice longer a key that is 1 bit closer to you)

Republishing

To avoid stale data:

- republication by the peer in charge every hour (during the ping to neighbors)
- republication by the owner every 24 hours

Conclusion on Kademlia

Provable

Performance is provable

Robust to attacks

Bombing with new (fake) peers difficult because of enhanced LRU

Strength

- XOR metric: routing by dialing makes routing, caching, and insertion easy
- Parallel requests (3 then 17)
- Efficient use of the uptime

DHT: conclusions

Existing topologies

- Chord [Stoica & alii, 2001] : circle
- CAN [Ratnasamy et al., 2001] : multidimensional torus
- Pastry [Rowstron et Druschel 2001] : hypercube
- Viceroy [Malkhi et al., 2002] : butterfly
- Kademlia [Maymounkov Mazières 2002] : hypercube
- D2B [Fraigniaud et Gauron, 2003] : de Bruijn

DHT: take-away

Benefits

- Fast routing: $O(\log N)$, possibly $O(\frac{\ln N}{\ln \ln N})$ ($b = \ln N$ on the hypercube)
- Compact routing tables: $O(\log N)$, $O(\frac{(\ln N)^2}{\ln \ln N})$, or even constant.
- Much harder to destroy than a single server
- True P2P, true scalability

Issues

- Anything goes through the hash function $\rightarrow$ no complex request !
- Slower than a traditional server
- Protocols can be complex, bugs are hard to address

Part III

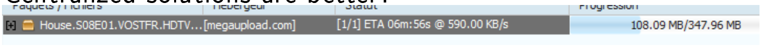
Bonus - Bandwidth conservation law

P2P = Bad performance?

- P2P has a low cost reputation

P2P = Bad performance?

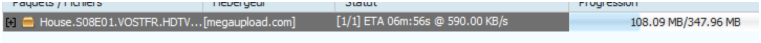
- P2P has a low cost reputation
- Centralized solutions are better?



Paquets / Fichiers	Hébergeur	Statut	Progression
House.S08E01.VOSTFR.HDTV...	[megaupload.com]	[1/1] ETA 06m:56s @ 590.00 KB/s	108.09 MB/347.96 MB

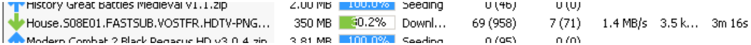
P2P = Bad performance?

- P2P has a low cost reputation
- Centralized solutions are better?



```
Paquets / Fichiers        Hébergeur        Statut                                   Progression
[+] House.S08E01.VOSTFR.HDTV…[megaupload.com]  [1/1] ETA 06m:56s @ 590.00 KB/s    108.09 MB/347.96 MB
```

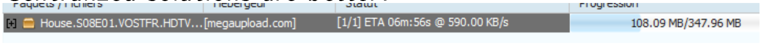
- Really?



```
History Great Battles Medieval v1.1.zip      2.00 MB   100.0%  Seeding   0 (46)    0 (0)
House.S08E01.FASTSUB.VOSTFR.HDTV-PNG…        350 MB    30.2%   Downl…    69 (958)  7 (71)   1.4 MB/s   3.5 k…   3m 16s
Modern Combat 2 Black Pegasus HD v3.0.4.zip  3.81 MB   100.0%  Seeding   0 (95)    0 (0)
```

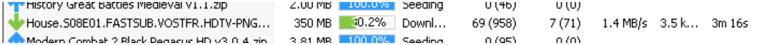
P2P = Bad performance?

- P2P has a low cost reputation
- Centralized solutions are better?



Paquets / Fichiers	Hébergeur	Statut	Progression
House.S08E01.VOSTFR.HDTV...	[megaupload.com]	[1/1] ETA 06m:56s @ 590.00 KB/s	108.09 MB/347.96 MB

- Really?

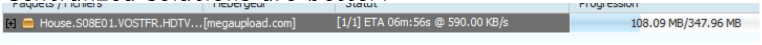


History Great Battles Medieval v1.1.zip	2.00 MB	100.0%	Seeding	0 (46)	0 (0)			
House.S08E01.FASTSUB.VOSTFR.HDTV-PNG...	350 MB	30.2%	Downl...	69 (958)	7 (71)	1.4 MB/s	3.5 k...	3m 16s
Modern Combat 2 Black Pegasus HD v3.0.4.zip	3.81 MB	100.0%	Seeding	0 (95)	0 (0)			

- Can we understand why it works?

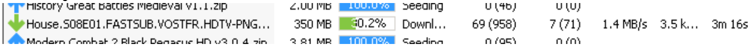
P2P = Bad performance?

- P2P has a low cost reputation
- Centralized solutions are better?



Paquets / Fichiers	Hébergeur	Statut	Progression
House.S08E01.VOSTFR.HDTV…	[megaupload.com]	[1/1] ETA 06m:56s @ 590.00 KB/s	108.09 MB/347.96 MB

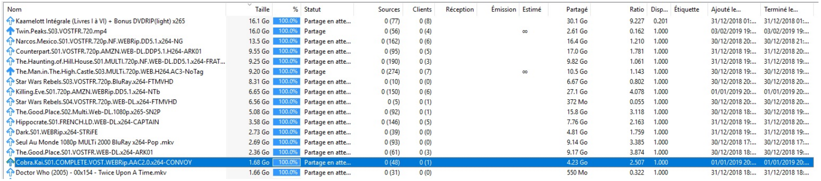
- Really?



History Great Battles Medieval v1.1.zip	2.00 MB	100.0%	Seeding	0 (46)	0 (0)			
House.S08E01.FASTSUB.VOSTFR.HDTV-PNG…	350 MB	30.2%	Downl…	69 (958)	7 (71)	1.4 MB/s	3.5 k…	3m 16s
Modern Combat 2 Black Pegasus HD v3.0.4.zip	3.81 MB	100.0%	Seeding	0 (95)	0 (0)			

- Can we understand why it works?
- Content-oriented analysis (BitTorrent)

A BitTorrent client



Nom	Taille	%	Statut	Sources	Clients	Réception	Émission	Estimé	Partagé	Ratio	Disp...	Étiquette	Ajouté le...	Terminé le...
Kaamelott Intégrale (Livres I à VI) + Bonus DVDRIP(light) x265	16.1 Go	100.0%	Partage en atte...	0 (77)	0 (8)				30.1 Go	9.227	0.201		31/12/2018 01:...	31/12/2018 01:...
Twin.Peaks.S03.VOSTFR.720.mp4	16.0 Go	100.0%	Partage	0 (56)	0 (4)			∞	2.61 Go	0.162	1.000		03/02/2019 19:...	03/02/2019 19:...
Narcos.Mexico.S01.VOSTFR.720p.NF.WEBRip.DD5.1.x264-NG	13.5 Go	100.0%	Partage en atte...	0 (162)	0 (6)				16.4 Go	1.210	1.000		30/12/2018 20:...	30/12/2018 21:...
Counterpart.S01.VOSTFR.720p.AMZN.WEB-DL.DDP5.1.H264-ARK01	9.55 Go	100.0%	Partage en atte...	0 (95)	0 (5)				17.0 Go	1.781	1.000		31/12/2018 19:...	31/12/2018 20:...
The.Haunting.of.Hill.House.S01.MULTI.720p.NF.WEB-DL.DD5.1.x264-FRAT...	9.25 Go	100.0%	Partage en atte...	0 (190)	0 (3)				9.82 Go	1.061	1.000		31/12/2018 19:...	31/12/2018 19:...
The.Man.in.The.High.Castle.S03.MULTi.720p.WEB.H264.AC3-NoTag	9.20 Go	100.0%	Partage	0 (274)	0 (7)			∞	10.5 Go	1.143	1.000		30/12/2018 19:...	30/12/2018 19:...
Star Wars Rebels.S03.VOSTFR.720p.BluRay.x264-FTMVHD	8.31 Go	100.0%	Partage en atte...	0 (10)	0 (0)				6.67 Go	0.802	1.000		30/12/2018 20:...	30/12/2018 20:...
Killing.Eve.S01.720p.AMZN.WEBRip.DD5.1.x264-NTb	6.65 Go	100.0%	Partage en atte...	0 (150)	0 (6)				27.1 Go	4.078	1.000		01/01/2019 20:...	01/01/2019 20:...
Star Wars Rebels.S04.VOSTFR.720p.WEB-DL.x264-FTMVHD	6.56 Go	100.0%	Partage en atte...	0 (5)	0 (1)				372 Mo	0.055	1.000		30/12/2018 20:...	30/12/2018 20:...
The.Good.Place.S02.Multi.Web-DL.1080p.x265-SN2P	5.08 Go	100.0%	Partage en atte...	0 (92)	0 (1)				15.8 Go	3.118	1.000		30/12/2018 18:...	30/12/2018 19:...
Hippocrate.S01.FRENCH.LD.WEB-DL.x264-CAPTAIN	3.58 Go	100.0%	Partage en atte...	0 (146)	0 (5)				7.76 Go	2.163	1.000		31/12/2018 19:...	31/12/2018 20:...
Dark.S01.WEBRip.x264-STRiFE	2.73 Go	100.0%	Partage en atte...	0 (39)	0 (0)				4.81 Go	1.759	1.000		31/12/2018 19:...	31/12/2018 19:...
Seul Au Monde 1080p MULTi 2000 BluRay x264-Pop .mkv	2.69 Go	100.0%	Partage en atte...	0 (93)	0 (0)				9.14 Go	3.385	1.000		30/12/2018 17:...	30/12/2018 17:...
The.Good.Place.S01.VOSTFR.WEB-DL.x264-ARK01	2.36 Go	100.0%	Partage en atte...	0 (61)	0 (3)				9.17 Go	3.874	1.000		30/12/2018 18:...	30/12/2018 19:...
Cobra.Kai.S01.COMPLETE.VOST.WEBRip.AAC2.0.x264-CONVOY	1.68 Go	100.0%	Partage en atte...	0 (48)	0 (1)				4.23 Go	2.507	1.000		01/01/2019 20:...	01/01/2019 20:...
Doctor Who (2005) - 00x154 - Twice Upon A Time.mkv	1.66 Go	100.0%	Partage en atte...	0 (31)	0 (0)				550 Mo	0.322	1.000		31/12/2018 18:...	31/12/2018 18:...

- Seeders, leechers, upload, download, ratio...
- What does this all mean?

Full hybrid model

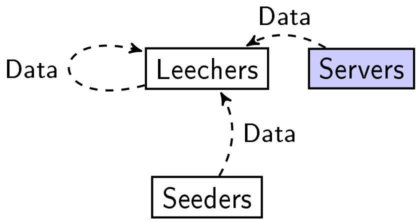
We usually consider three types of "peers":

Centralized servers(C)	Non-P2P component of the system (e.g. seedboxes)
Leechers (L)	Want content ; may provide some content to other leechers
Seeders (S)	Possess content and can provide it to leechers

Two variants

- Static / closed (aka "snapshot")
- Dynamic / open (aka "evolutive")

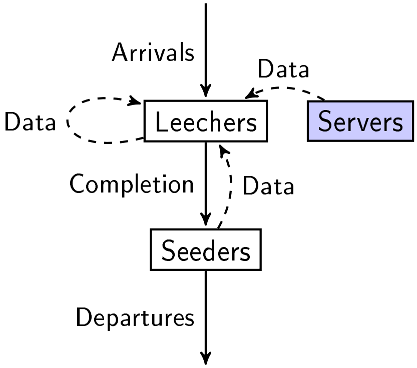
Static model



Leechers can receive from everyone:

- Other leechers
- Seeders
- Servers

Dynamic model



Life cycle:

- Peer starts as leecher
- Upon completion it becomes seeder
- After some time it leaves
- Les servers are static

Notation

- Servers/leechers/seeders: C, $s \in S$, $l \in L$
- Number of peers: N_S, N_L
- Upload capacity: U_S, U_C, u_l, $\bar{u_S}$
- Download: d_l, $d_{\max}$

Bandwidth conservation

- Nothing disappears, nothing is created, all is transferred

$$\sum_{l \in L} d_l \leq U_L + U_S + U_C$$

- Example for infinite download capacity,

$$\sum_{l \in L} d_l = \eta_L U_L + \eta_S U_S + \eta_C U_C$$

- η: Efficiency coefficient
- Perfect systems: $\eta = 1$

Bandwidth conservation

- Nothing disappears, nothing is created, all is transferred

$$\sum_{l \in L} d_l \leq U_L + U_S + U_C$$

- Example for infinite download capacity,

$$\sum_{l \in L} d_l = \eta_L U_L + \eta_S U_S + \eta_C U_C$$

- η: Efficiency coefficient
- **Perfect systems:** $\eta = 1$

Bandwidth conservation

- Si bandwidth is evenly split among the leechers

$$d = \min\left(d_{\mathrm{max}}, \bar{u_L} + \beta \bar{u_S} + \frac{U_C}{N_L}\right), \text{ avec } \beta = \frac{N_S}{N_L}$$

- Works for closed on open systems
- Works for streaming and filesharing
- Many upgrades possible (η, d_{max}, uneven BW sharing. . .)

Streaming example (closed system)

- Target rate r, let $\alpha_X = \frac{\bar{u_X}}{r}$ and $N_C = \frac{U_C}{r}$
- Streaming requires $d \geq r$, otherwise…
- $\alpha_L + \beta\alpha_S$: normalized capacity of the P2P system
- Case 1: $\alpha_L + \beta\alpha_S \geq 1$
  - System is scalable (N_C is not in the formula)
  - Servers are not mandatory for raw bandwidth (but useful for bootstrap/delay/robustness)
- Case 2: $\alpha_L + \beta\alpha_S < 1$
  - N_L cannot go above $\frac{N_C}{1-(\alpha_L+\beta\alpha_S)}$
  - $\rightarrow$ Leveraging effect: server capacity is multiplied by $\frac{1}{1-(\alpha_L+\beta\alpha_S)}$

Interlude: Joost

joost™

BETA

Interlude: Joost

An epic fail story

- After KaZaA (2000), Niklas Zennström and Janus Friis want to become kings of the (P2P) world.
- World is not ready, so they make a small garage project: Skype
- 2005: Skype sold to eBay. Back to world conquest (with a lot of money).
- 2007: Launch of Joost, the ultimate P2P software (P2P-powered Netflix)
- 2008: P2P-part abandoned; browser solution (Youtube-like)...
- 2009: Effective shutdown("official" shutdown: 2012)

Interlude: Joost

- Joost was proprietary software
- Upload/download ratio could be measured: $\alpha = \frac{1}{6}$
- If $\beta = 0$, leverage effect of 20%
- If $\beta = 1$, leverage effect of 50% (most likely behavior)
- If $\beta = 2$, 100%
- ...
- Scaling requires $\beta \geq 5$ (even at peak hours)
- Conclusion: Joost only used P2P as a booster (like BBC)

Conclusions on bandwidth

- Rules valid for all types of content distribution
- First order only (feasibility / order of magnitude)
- Second order is context specific
  - Latency
  - Core bottlenecks, packet losses
  - Underlying network protocols (TCP/μTP)
  - Propagation / availability of data (chunks)
  - Overhead
  - ...

Part IV

Bonus: a zoom on BitTorrent

Three parts in a torrent's lifetime

Birth (flashcrowd)

- One unique owner (initial seeder)
- Burst of arrivals

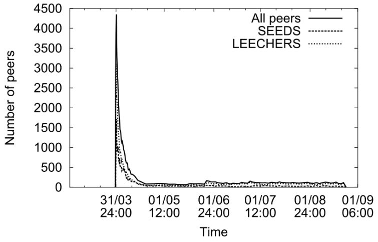
Stationary regime

- Departures and arrivals are roughly the same
- BCL laws are (roughly) respected

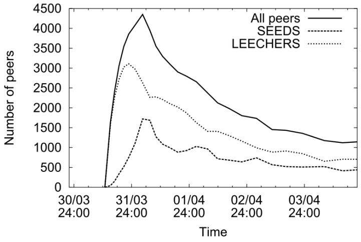
Death

- Less and less arrivals
- At some point, availability failure (missing chunk)

Three parts in a torrent's lifetime



(a) Complete trace



(b) Zoom on the first five days

Source : Dissecting BitTorrent : Five Months in a Torrent's Lifetime

Flashcrowd

Where's the bottleneck?

During a flashcrowd, the bottleneck can mainly be at two places.

- If the seeder is fast enough, the bottleneck is the leechers' bandwidths $\rightarrow$ vanilla BCL.
- If the seeder is slow, the bottleneck is content availability, i.e. the seeder's bandwidth.
- The frontier depends mainly on the arrival rate.
- Seeder bottleneck = risk of instability

Flashcrowd: transition phase

When seeder's content is the bottleneck, many things happen when the last chunk is distributed

- Just before the last chunk is sent
  - Everyone has almost every chunk but one
  - Huge deviation on chunk availability
- If all goes well
  - The deviation slowly diminishes
  - Soft transition towards stationary regime
- If all goes wrong
  - As soon as someone gets the new chunk, she leaves
  - As soon as someone arrives, she quickly retrieves all chunks but one
  - Only the seed is guaranteed to possess the last chunk: missing chunk syndrome

Missing chunk syndrome

- Attack used IRL by some majors to slow down some torrents
- Simple cure
  - Seeder: stealth mode (a.k.a. superseed)
  - Leechers: rarest first policy + seed afterwards

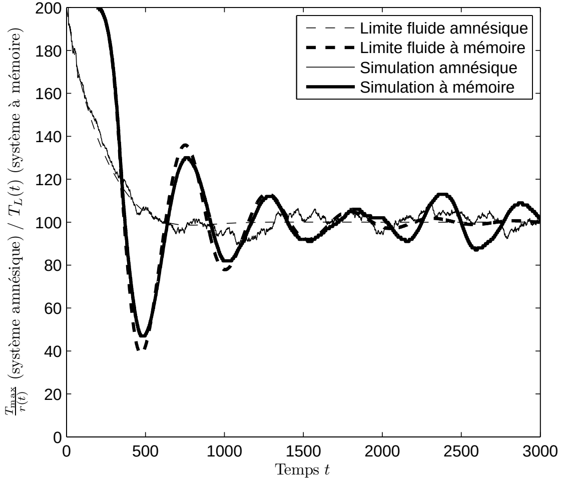
Transition to a stationary regime

FYI, the phenomenon can be modeled by differential equations.

$$\begin{pmatrix} \dot{n}_L(t) \\ \dot{n}_S(t) \end{pmatrix} = \begin{pmatrix} -\sigma_L & -\sigma_L \\ \sigma_L & \sigma_L - \sigma_S \end{pmatrix} \begin{pmatrix} n_L(t) \\ n_S(t) \end{pmatrix} + \begin{pmatrix} \lambda - C\sigma_L \\ C\sigma_L \end{pmatrix}$$

Transition to a stationary regime

FYI, the phenomenon can be modeled by differential equations.



Death

- At least one chunk becomes unavailable
- Possibly a (fake) flashcrowd can appear
- Two options: reseed or game over

Free-riders issue

To who should I send? And WHY?

- Two problems to solve:
  - "Server"-side of a peer. How to prioritize incoming requests? Waiting queue is a fair solution but there is... waiting!...
  - How to give some incentive to upload?
    - Upload bandwidth is limited (especially in the old days): upload costs
    - Legal issues
    - $\rightarrow$ Why sharing ?? $\rightarrow$ free-riders: just receive don't give
- Bram Cohen $\rightarrow$ two birds with one stone

Prisoner's dilemma

- Two gangsters, Al et Vito, are arrested
- They are separated in isolated cells
- Only circumstantial evidence available
- Testimony required ! Attorney gives incentives:

Al /Vito	Al is silent	Al talks
Vitois silent	1 year / 1 year	free / 20 years
Vito talks	20 ans / free	15 years / 15 years

Prisoner's dilemma

What are the options?

- Whatever Vito chooses, Al gains if he defects
- Same for Al: defecting always means less jailtime
- But the best solution for both of them is to keep quiet

Al /Vito	Al is silent	Al talks
Vitois silent	1 year / 1 year	free / 20 years
Vito talks	20 ans / free	15 years / 15 years

Prisoner's dilemma

Nash equilibrium

When no player can improve things for her by changing her decisions (all other things being equal)

Dilemma

The Nash equilibrium is not Pareto-optimal: $(15, 15) > (1, 1)$

Al /Vito	Al is silent	Al talks
Vitois silent	1 year / 1 year	free / 20 years
Vito talks	20 ans / free	15 years / 15 years

Do you know Golden Balls?

Prisoner's dilemma: free-riders' edition!

Satisfaction

- Download a file: +10
- Legal risk, upload cost: -1

Alice /Bob	Alice shares	Alice free-rides
Bob shares	9 / 9	10 / -1
Bob free-rides	-1 / 10	0 / 0

Yes but....

P2P is actually an iterated version of the dilemma
(you play multiple games)

Empirical observations

"Nice" tactics pay
[JP Delahaye "L'altruisme récompensé ?" Pour la Science 1992]

A very strong, very simple tactic: Tit For tat : play the other's previous move
Start wit cooperation (generous tit-for-tat): pat to earn trust

Remarks

A frequent dilemma

Prisoner's dilemma appears in many real situations

- Terror equilibrium (MAD)
- Tax rate in states
- Politics

An active research field

Relatively recently (2012), Freeman Dyson discovered a whole new class of dominant strategies.

Bram Cohen's idea

P2P is like iterated prisoner's dilemma.

Don't share with those who wait but share with those who cooperate! Event better: those that cooperate with me!

The Tit-for-Tat paradigm

Tit for tat

Give and ye shall receive

BitTorrent

- c upload connections ($c = 4$, then $c = 0.8\sqrt{u}$)
- Every 10s, compute the best peers (that have sent the most)
- Give an upload slot to the $c - 1$ best ones
- Give one upload slot to a random peer (optimistic unchoke)
  - Helps to establish new bilateral connections
  - Gains trust

The Tit-for-Tat paradigm

Link with BCL

$$d(u) \approx \frac{1}{c}\bar{u_L} + \frac{c-1}{c}u + \dots \text{ (seeders/serveurs)}$$

- Idea: law of demand and supply (stock exchange)
- Formally: preference-based networks (not here)
- There is a formula!
- Doesn't work for bootstrap (getting first chunk)
- Rich are faster $\rightarrow$ skewed observed bandwidth distribution

The Tit-for-Tat paradigm

Benefits

- You get something when you share
- Hence, average upload BW shared increases
  - Network QoS increases!
  - Peer QoS increases: upload $\simeq$ download
- No need to compute global cooperation
  - Very hard to cheat
  - Just value quality of a link over a short period of time

The Tit-for-Tat paradigm

Cons

- Better global QoS $\rightarrow$ better for free-riders...
- Bootstrap (when you have no chunk)
- No reward for pure servers (seeds) $\rightarrow$ inefficient if many seeds are available

The Tit-for-Tat paradigm

Free-riders tolerance

Formula: as long as

$$p_f \leq 1/c + \frac{c-1}{c}\left(\frac{\bar{T}_S \bar{u_S}}{k} + \frac{U_C}{\lambda k}\right),$$

the system can cope with free-riders. Beyond, the system explodes (peers stay forever in the system).

Share Ratio : Tit-for-Tat for seeders

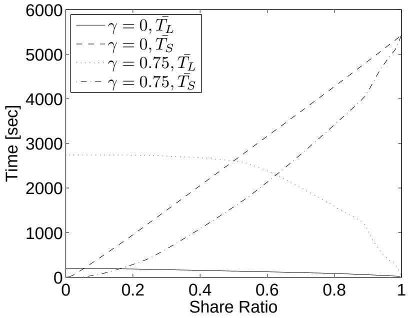
- Goal: have seeders for the torrent
- Share ratio : upload/download ratio
- Share ratio policy : punish (at tracker level) the peers that are below a given share ratio SR
- Generousity: initial quota of virtual upload for bootstrap
- What's a good SR ?

Share Ratio : Tit-for-Tat for seeders

- Goal: have seeders for the torrent
- Share ratio : upload/download ratio
- Share ratio policy : punish (at tracker level) the peers that are below a given share ratio SR
- Generousity: initial quota of virtual upload for bootstrap
- What's a good SR ?
  - Better pick $SR < 1$ (because the average SR is 1 by definition)
  - The closer to 1, the bigger the swarms

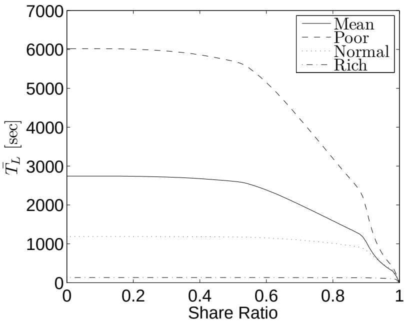
Share Ratio: global impact

- $T_S(u) = \max(0, \frac{SRF}{u} - T_L(u))$ (F : filesize)
- $\gamma = \frac{k-1}{k}$: TfT ratio
- T_S is impacted by TfT policy
- Remark: without SR, TfT takes longer than pure random?

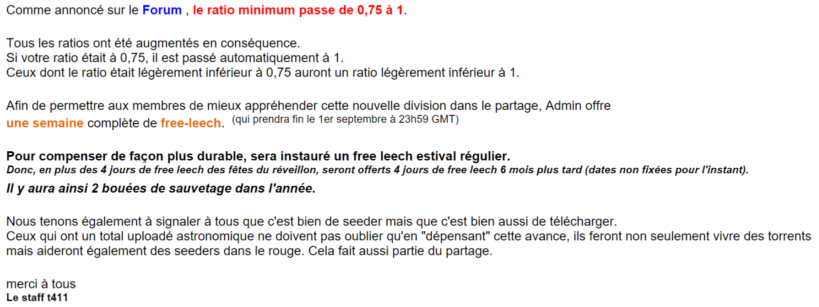


Share Ratio: per class impact

- Assume three types of peers
- SR asks by thresholds on the different classes
- First threshold: 0.6
- Second threshold at 0.9 :risky



SR=1?



Comme annoncé sur le **Forum** , **le ratio minimum passe de 0,75 à 1**.

Tous les ratios ont été augmentés en conséquence.
Si votre ratio était à 0,75, il est passé automatiquement à 1.
Ceux dont le ratio était légèrement inférieur à 0,75 auront un ratio légèrement inférieur à 1.

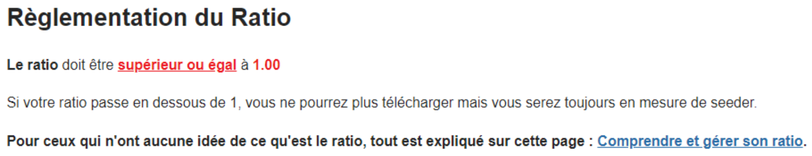
Afin de permettre aux membres de mieux appréhender cette nouvelle division dans le partage, Admin offre **une semaine** complète de **free-leech**. (qui prendra fin le 1er septembre à 23h59 GMT)

Pour compenser de façon plus durable, sera instauré un free leech estival régulier.
Donc, en plus des 4 jours de free leech des fêtes du réveillon, seront offerts 4 jours de free leech 6 mois plus tard (dates non fixées pour l'instant).
Il y aura ainsi 2 bouées de sauvetage dans l'année.

Nous tenons également à signaler à tous que c'est bien de seeder mais que c'est bien aussi de télécharger.
Ceux qui ont un total uploadé astronomique ne doivent pas oublier qu'en "dépensant" cette avance, ils feront non seulement vivre des torrents mais aideront également des seeders dans le rouge. Cela fait aussi partie du partage.

merci à tous
Le staff t411

SR=1?



Règlementation du Ratio

Le ratio doit être **supérieur ou égal** à **1.00**

Si votre ratio passe en dessous de 1, vous ne pourrez plus télécharger mais vous serez toujours en mesure de seeder.

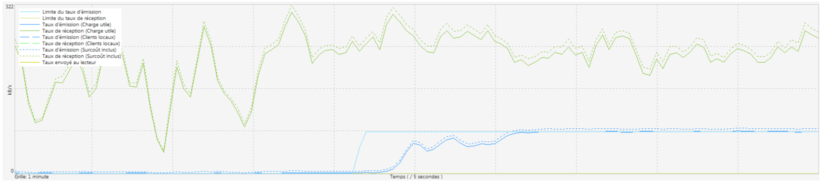
Pour ceux qui n'ont aucune idée de ce qu'est le ratio, tout est expliqué sur cette page : Comprendre et gérer son ratio.

(yggtorrent)

Good practice: optimize your BitTorrent

Ground rules for fast download

- Tit-for-tat $\rightarrow d = f(u)$: you should upload



Good practice: optimize your BitTorrent

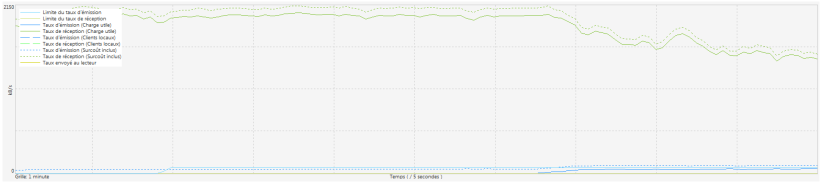
Ground rules for fast download

- Tit-for-tat $\rightarrow d = f(u)$: you should upload
- Seeders have a big impact $\rightarrow$ private trackers have higher performance

Good practice: optimize your BitTorrent

Ground rules for fast download

- Tit-for-tat $\rightarrow d = f(u)$: you should upload
- Seeders have a big impact $\rightarrow$ private trackers have higher performance
- Over-provisioning? Cut the upload (temporarily)!



Good practice: optimize your BitTorrent

Ground rules for fast download

- Tit-for-tat $\rightarrow d = f(u)$: you should upload
- Seeders have a big impact $\rightarrow$ private trackers have higher performance
- Over-provisioning? Cut the upload (temporarily)!
- Remark: High download speed doesn't guarantee low streaming latency

Acknowledgments

Part III owns a lot to Laurent Viennot and Fabien de Montgolfier.