# P2P Paradigm

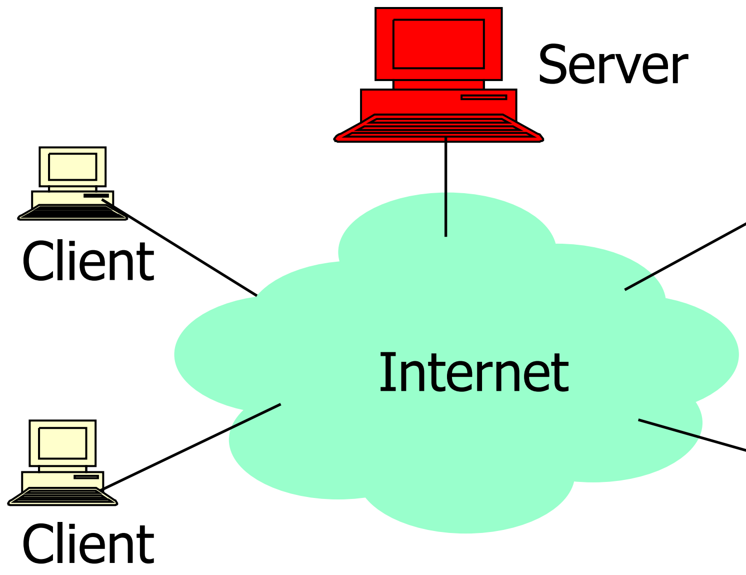
**Lenuta Alboaie**
**adria@info.uaic.ro**

## Content

- **Peer-to-peer(P2P) paradigm**
  - Preliminaries
  - Definitions
  - Characteristics
  - Application types
  - Infrastructures
  - Instruments

# Preliminaries

...let's remember client/server model



# Preliminaries

…let's remember client/server model

- Usually, we look at the client as at a component having low computational capabilities
- The server is maintained and managed centrally

Problems of client / server architecture:

- The lack of robustness
- Lack of safety (Reliability)
- Lack of scalability
- Vulnerability to attack

# Definitions

***Peer*** *= one that is of equal standing with another* (conform Webster)

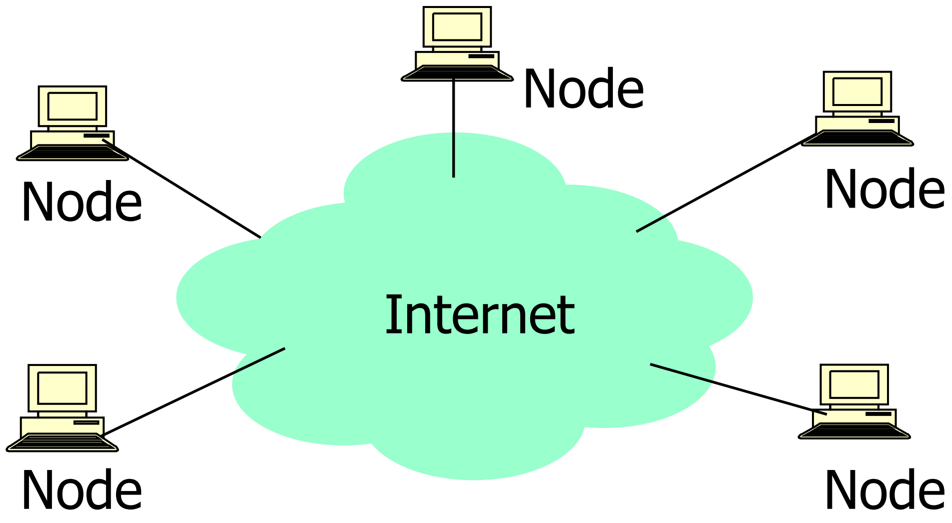
***Peer-to-peer*** (P2P) = network architecture where nodes are relatively equal

- Meaning that each node is capable of performing specific network functions
- In practice, many of the nodes can also perform functions

# Definitions

P2P Systems, in the strict sense, are fully distributed systems

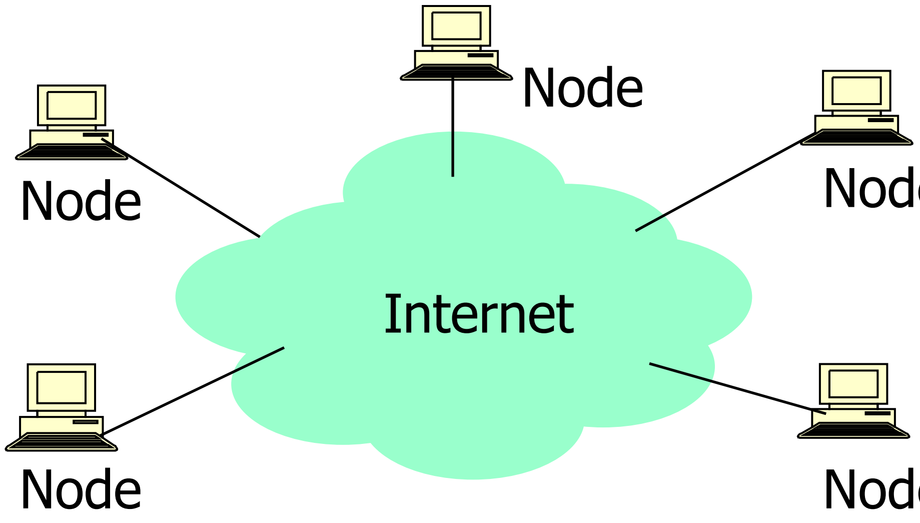
- All nodes are fully equivalent in terms of functionality and activities they can perform



Obs. Pure P2P systems are rare (e.g. Gnutella), most are hybrids, having super-nodes or servers with different roles (data search, control)

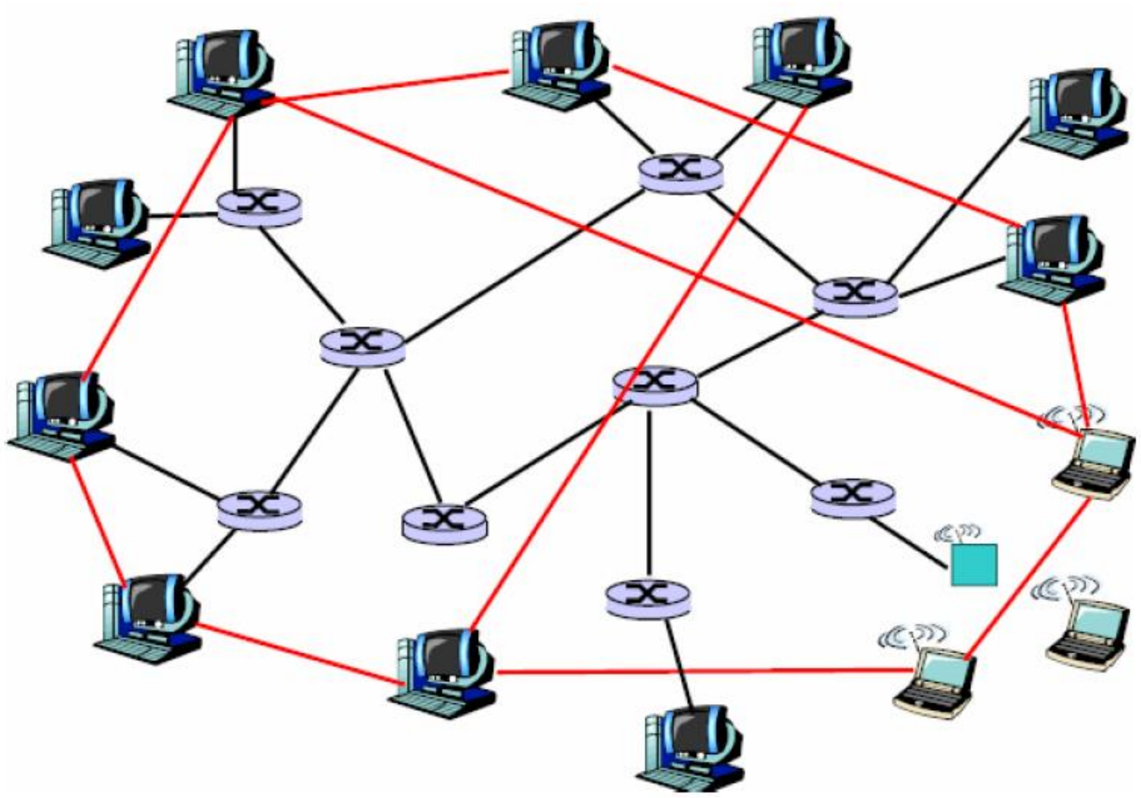
# Definitions

- Nodes
  - Can consume and provide data
    Any node can initiate a connection
- There is no centralized data source =>
  A form of democracy on the Internet
  Copy-right protection threatened



# Definitions

- "**P2P** is the class of applications that rely on the resources (storage, processing, content, human presence) available at the edges of the Internet



*Edges of the Internet* (*overlay networks*)

# Definitions

"**P2P** is a class of applications that take advantage of resources – storage, cycles, content, human presence – available at the edges of the Internet. Because accessing these decentralized resources means operating in an environment of unstable and unpredictable IP addresses, P2P nodes must operate outside the DNS system and have significant, or total autonomy from central servers"

"A distributed network architecture may be called a **P2P** network if the participants share a part of their own resources. These shared resources are necessary to provide the service offered by the network. The participants of such a network are both resource providers and resource consumers"

# Characterization

Characteristics:

- Sharing computational resources through interchange and less directly through mediation offered by a centralized authority (server)
    - Centralized servers can be used to perform specific activities (P2P network initialization, adding new nodes in the network ...)
    - Ideally, nodes participate actively in accomplishing operations such as location & caching nodes / content, routing information, transferred resource management etc.

# Characterization

Characteristics:

- The ability to address instability and variations of network connectivity, with adaptation for the occurred errors or for the node dynamics
  - P2P network topology is adaptive and fault tolerance; nodes are self-organized in order to maintain connectivity and network performance

# Characterization

P2P network is one overlay over the physical network

- It is at Application Level => flexibility
- Virtual edges are TCP connections or pointers to IP addresses
- P2P network maintenance is done by periodically verifying connectivity (ping) or existence (messages "still alive?")
- When a node fails, the P2P system could set new edges
- Nodes proximity (physical) is not important
- P2P network can be structured or not

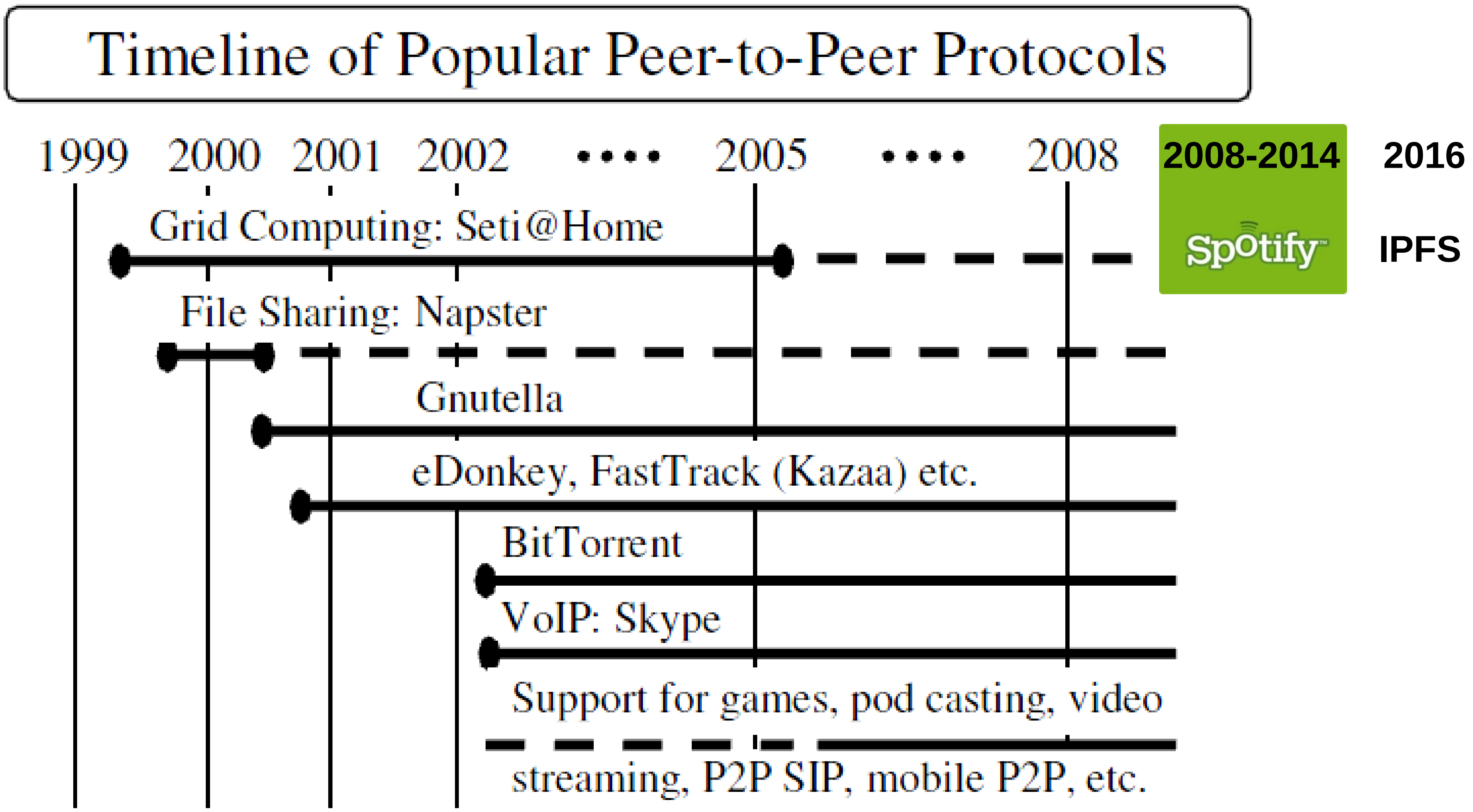
# Aims and benefits

- **Efficient use of resources**
  - Unused bandwidth, storage resources, processing power available to the network edges
- **Scalability**
  - Without centralized information, without bottleneck (communication and computing)
  - Integration of resources is done naturally during the system use
- **Reliability**
  - The existence of copies of data
  - Geographical distribution
  - No more "single point of failure“
- **Easy administration**
  - The nodes are self-organizing
  - Increased fault tolerance and load balancing
  - Increased autonomy
- **Anonymity**
  - Hard to accomplish in a centralized environment
- **Dynamism**
  - Dynamic environment
  - Ad-hoc collaboration and communication

## Disadvantages / Problems

- P2P architectures are probabilistic
  - Unpredictable resource location
  - Resources are volatile
- Nonexistence of centralized control
  - Issues requiring an authority on applications, content and users
  - Difficulties in detecting and identifying users (anti-social aspects)
- Encouraging the use of P2P systems abusive and illegal purposes (e.g. copyright of digital content)
  - Lack of trust in trades, business
  - Security issues (pending future)

# Evolution...


Timeline of Popular Peer-to-Peer Protocols
1999
2000
2001
2002
....
2005
....
2008
2008-2014
2016
Grid Computing: Seti@Home
Spotify
IPFS
File Sharing: Napster
Gnutella
eDonkey, FastTrack (Kazaa) etc.
BitTorrent
VoIP: Skype
Support for games, pod casting, video
streaming, P2P SIP, mobile P2P, etc.

# Types of Applications

- **Communication & collaboration**

  Systems that provide an infrastructure to facilitate direct communication & collaboration, often in real time between nodes

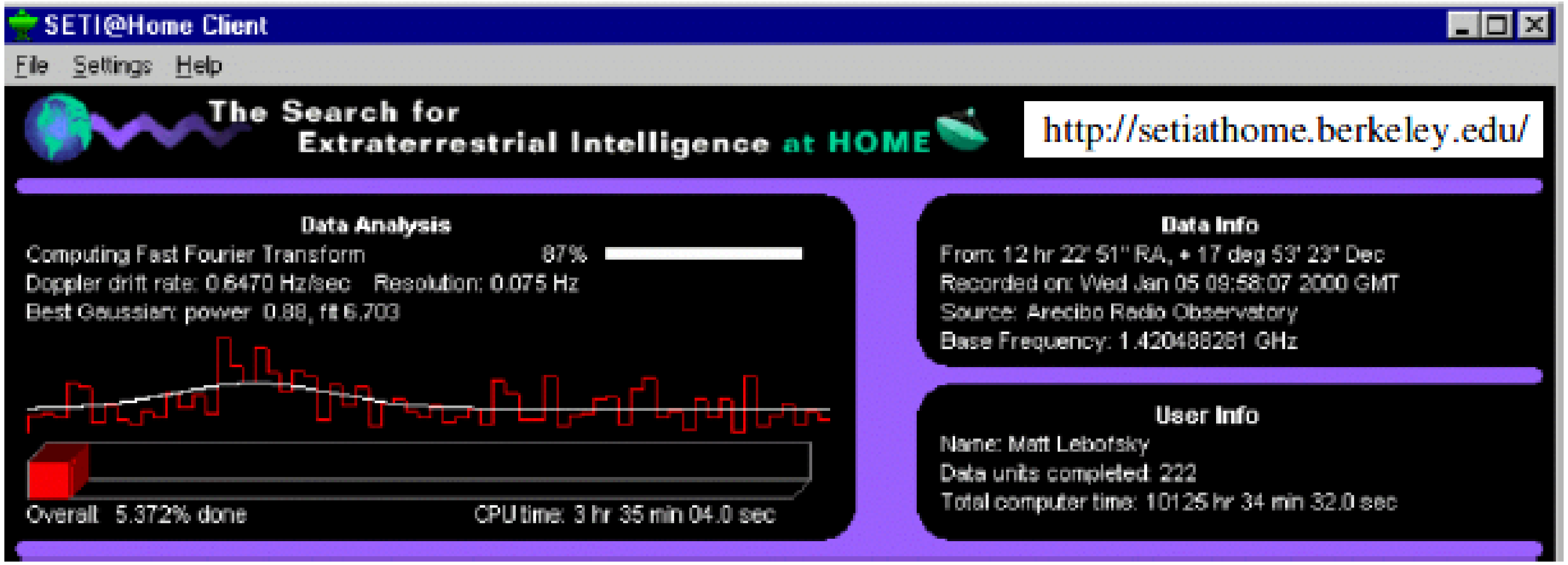
  Conversational systems (chat, instant messaging):

  IRC (Internet Relay Chat), ICQ (1996), YM !, MSN Messenger, Skype, P2P multicast systems (e.g. Cirrus - Adobe Flash http://labs.adobe.com/technologies/cirrus/; WebRTC) ...

- **Distributed Computing**
  - Systems that use computer power of available nodes (processor cycles)
  - Solving problems through *divide-et-impera*: SETI@home (*Search for Extra-Terrestrial Intelligence*-Berkeley), genome@home
    - P2P network is a kind of a computational Grid (… master course)

## Types of Applications| Distributed Computing - Example

SETI@Home: A Public-Resource Computing Experiment


SETI@Home Client
File Settings Help
The Search for Extraterrestrial Intelligence at HOME
http://setiathome.berkeley.edu/

Data Analysis
Computing Fast Fourier Transform 87%
Doppler drift rate: 0.6470 Hz/sec Resolution: 0.075 Hz
Best Gaussian: power 0.88, fit 6.703
Overall: 5.372% done CPU time: 3 hr 35 min 04.0 sec

Data Info
From: 12 hr 22' 51" RA, + 17 deg 53' 23" Dec
Recorded on: Wed Jan 05 09:58:07 2000 GMT
Source: Arecibo Radio Observatory
Base Frequency: 1.420488281 GHz

User Info
Name: Matt Lebofsky
Data units completed: 222
Total computer time: 10125 hr 34 min 32.0 sec

- ❑ Radio telescope signal analysis has insatiable appetite for computing power
- ❑ Usage of computers in homes and offices around the world has provided unprecedented computing power
- ❑ Grid computing application via peer-to-peer approach under central control

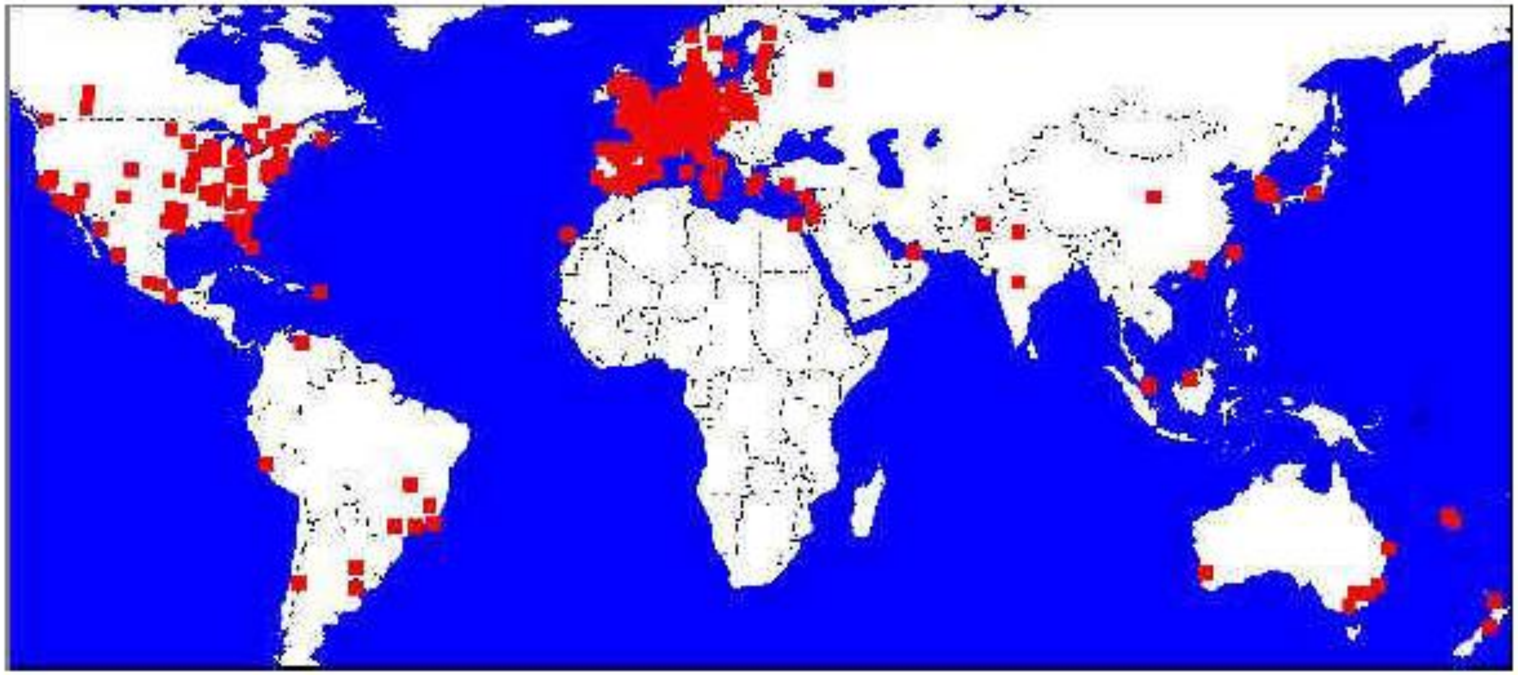
# Types of Applications

- **Storage systems (databases)**
  - Design of distributed database infrastructure based on P2P
    - PIER – a scalable engine for distributed query (http://pier.cs.berkeley.edu/)
    - Edutella –open-source project for queries and *meta-data* storage (P2P for Semantic Web)
- **Distribution of digital content**
  - Systems & Infrastructure to share digital resources (multimedia and other data) among users
    - File sharing applications (e.g. Napster, Gnutella, KaZaA, Freenet, BitTorrent, eDonkey etc.)
    - Distributed storage media for publishing, organizing, indexing, searching and retrieving data in secure & efficient manner (PAST, Chord, Groove, Mnemosyne, Avalanche,…)

## Types of Applications

- **Distribution of digital content| Example**

P2P File-Sharing: Fast distribution of large files



Example: Harry Potter III early propagation after 2 hours on May 28th 2004 (Source: *www.itic.ca/DIC/News/archive.html*)

## Types of Applications

- **Distribution of content through P2P**
  - P2P systems for “interchange files”
    - Nodes transfer files at a time
    - It offers facilities for the achievement of P2P networks and search & transfer files between nodes
    - Security, availability and persistence are not supported
    - Examples: Napster, KaZaA, Gnutella

## Types of Applications

- **Distribution of content through P2P**
  - P2P systems for publishing & content storage
    Users can publish, store and distribute digital content, based on access rights (privileges)
    - Focuses on security and persistence
    - Some systems offer facilities regarding collaboration between users
    - Example: Scan, Groove, Freenet, MojoNation, Tangler

# Types of Applications

- **Distribution of content through P2P**
  - Infrastructure for:
    - Routing & Localization:

      Chord, Can, Pastry, Tapestry, Kademila
    - Anonymity:

      Onion Routing, ZeroKnowledge, Freedom, Tarzan
    - Reputation management:

      EigenTrust, PeerTrust

## Infrastruct.(Routing & Localization)

- Localization and routing mechanism that can be adopted depends on:
  - Topology
  - Structure
  - The degree of centralization of the *overlay network*

# Infrastruct.(Routing & Localization)

- **Aspects regarding centralization**
  - **Pure decentralized architectures**: all nodes perform exactly the same activities, by playing roles of servers and clients simultaneously, without the benefit of a central coordination
    - Nodes are called servents (SERVers + clieENTS)

## Infrastruct.(Routing & Localization)

- **Aspects regarding centralization**
  - **Partially centralized architectures**: some nodes have a more important role (e.g., storing local indexes for shared folders)
    - Nodes become super-nodes in accordance with the policies of each P2P system
    - Super-node role is determined dynamically
  - **Hybrid decentralized architecture**: there is a central server enabling the interaction between nodes, keeping catalogs with metadata of files
    - Servers can identify and verify the storage nodes
    - The systems are called *broker mediated*

# Infrastruct.(Routing & Localization)

- **Aspects regarding network structure:**
  - **Unstructured:** placing content is completely independent of overlay network topology
    - The content must be located
    - Search strategies by "brute force“: flooding the network - requests propagated via BFS / DFS
    - More sophisticated strategies: random path , probabilistic methods etc.
  - **Loosely structured**: although the content location is not completely specified, it is affected by routing
    - Category located between structured and unstructured networks

## Infrastruct.(Routing & Localization)

- **Aspects regarding network structure:**
  - **Structured:** topology is controlled, files are placed in precise locations
    - A mapping  between the content (file identifier) and the location (node address) is performed
      - Like a distributed routing table
    - *exact-match queries* can be performed in a scalable way
    - The structure used to guide message routing is difficult to maintain for transient nodes (with high rates of attachment and disconnection from the network)

## Infrastruct.(Routing & Localization)

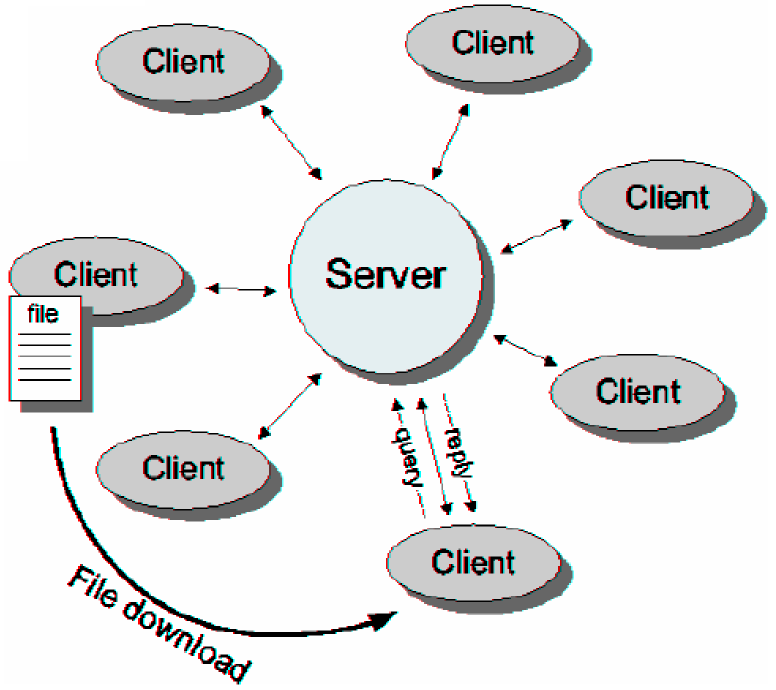
| | Centralizare | | |
|---|---|---|---|
| | **Hibridă** | **Partială** | **Absentă** |
| Nestruc-turată | Napster Publius | KaZaA Morpheus Gnutella Edutella | Gnutella FreeHaven |
| Infrastruc. structurată | | | Chord, CAN Tapestry, Pastry |
| Sisteme structurate | | | OceanStore Scan, PAST Kademlia Tarzan |

# Unstructured Architectures

## Decentralized and hybrid

- Each client computer stores content (files) shared (s)
- The central server keeps a table with registered users (IP, bandwidth, ...) + a table with the list of files for user & meta-data

  Example: **Napster, Publius**



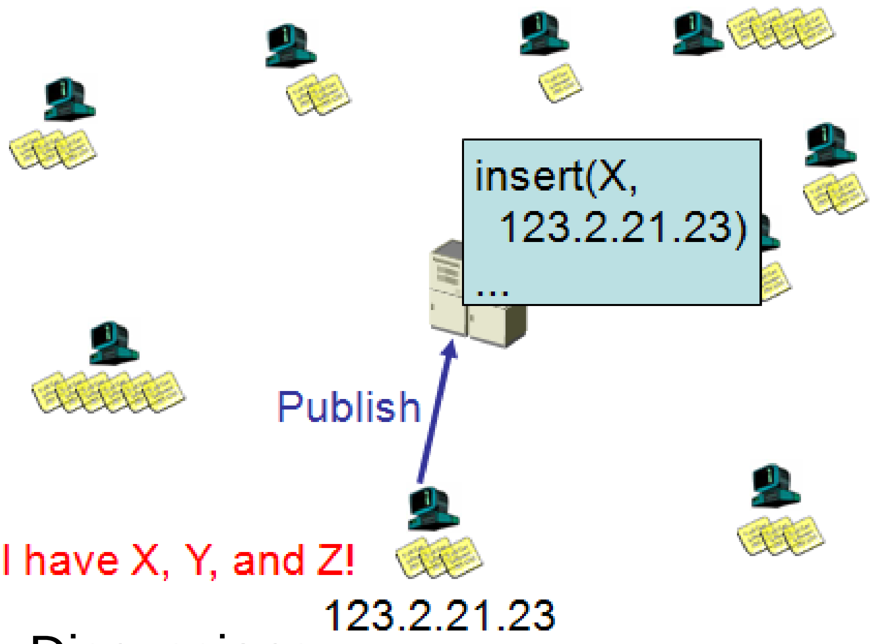
## Unstructured Architectures

### Napster

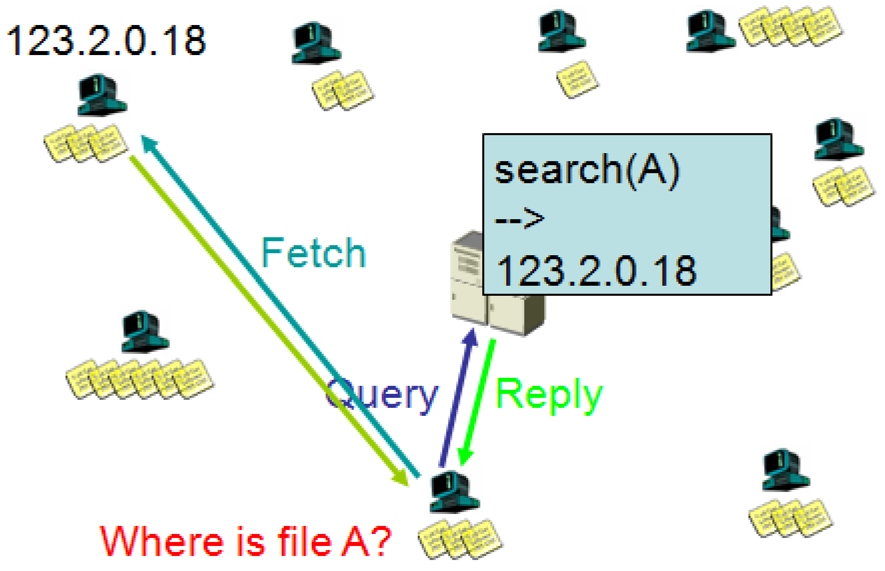
- 1999: Sean Fanning release Napster
- It has reached at 1.5 million concurrent users
- Centralized database - operations:
  - **Join**: the client contacts the central server (via TCP)
  - **Publish**: reporting a list of files to the central server
  - **Search**: query server => it returns one that stores the requested file
  - **Fetch**: take the file directly from peer (the one with the best transfer rate)
- July 2001: Napster was closed

# Unstructured Architectures

**Napster:** Publish



**Napster:** Search



Discussions:

The server does all processing

We have "single point of failure"

Scalability problems, some systems do not allow adding other servers (the list of available servers is static)

# Unstructured Architectures

## Pure Decentralized

- An overlay network is built using routing mechanism based on IP
- There is no central coordination
- Users are connecting via an application that has a dual role – *servent*
- Communication between *servents* is based on a protocol at the application level, with 4 types of messages:
  - Ping – requires a node to announce himself
  - Pong – reply to the *ping* message(IP, port, file size)
  - Query – search request (search string + minimum speed of transfer)
  - Query hints – response (IP, port, speed, file length, file index)

# Unstructured Architectures

## Pure Decentralized

- Searching is done by flooding (flooding)
  - If you don’t have the desired file, ask the n neighbors
  - If they do not have the file, they will ask their neighbors in maximum m hops
  - On the way back answers will be returned (not the files content)
- Each message has a TTL attached
- Example: Gnutella

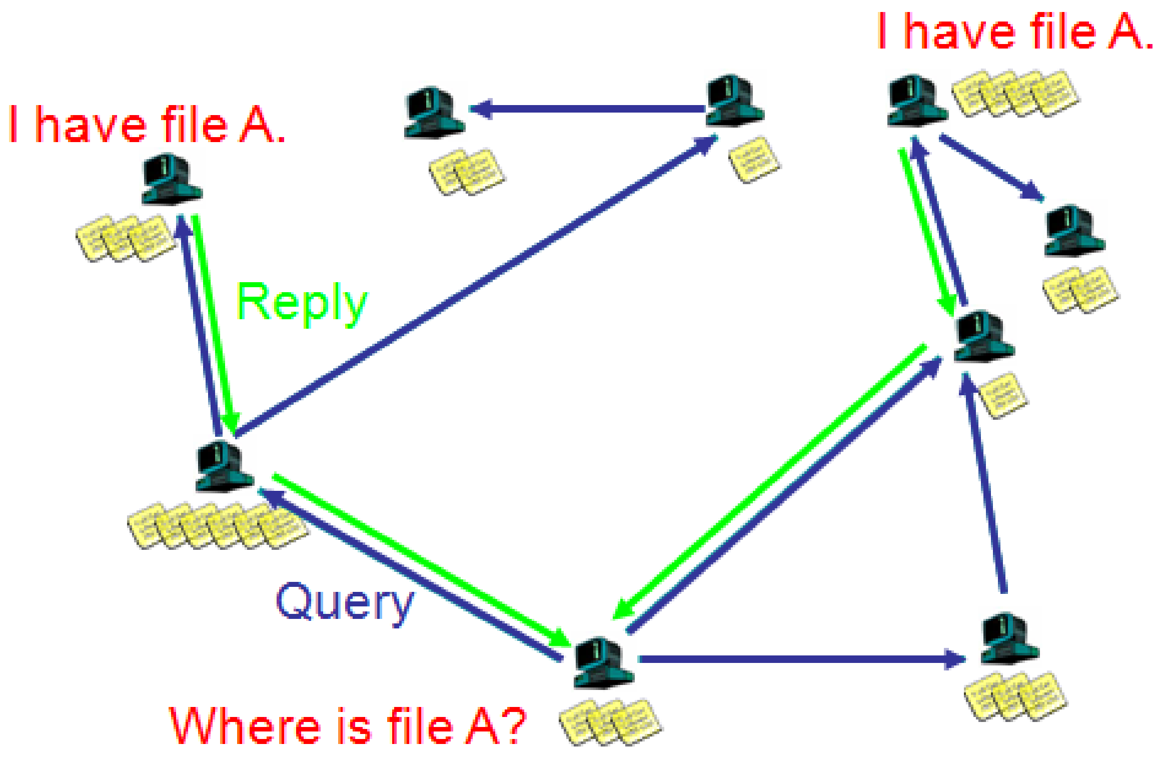
# Unstructured Architectures

## Gnutella

- 2000: L. Frankel and T. Pepper(Nullsoft) launches Gnutella
- Client application comes up: Bearshare, Morpheus, LimeWire
- *Query Flooding*:
  – **Join**: at entrance, the client contacts a few nodes that became his neighbors
  – **Publish**: Not required
  – **Search**: asks neighbors, who ask their neighbors ...
    - There is a TTL limiting the spread
  – **Fetch**: take the file directly from peer

# Unstructured Architectures

**Gnutella**



Issues:

- Search times is… O(?)
- The nodes leave often => unstable network

## Unstructured Architectures

**Partially centralized**

- Use the concept of super-node: that performs various activities in a P2P network (indexing, *caching*)
- The nodes are automatically chosen as super-nodes if they have enough bandwidth and computational power
- All requests are sent initially to super-nodes
- Advantages: resource discovery time is less
- Example: **KaZaA**

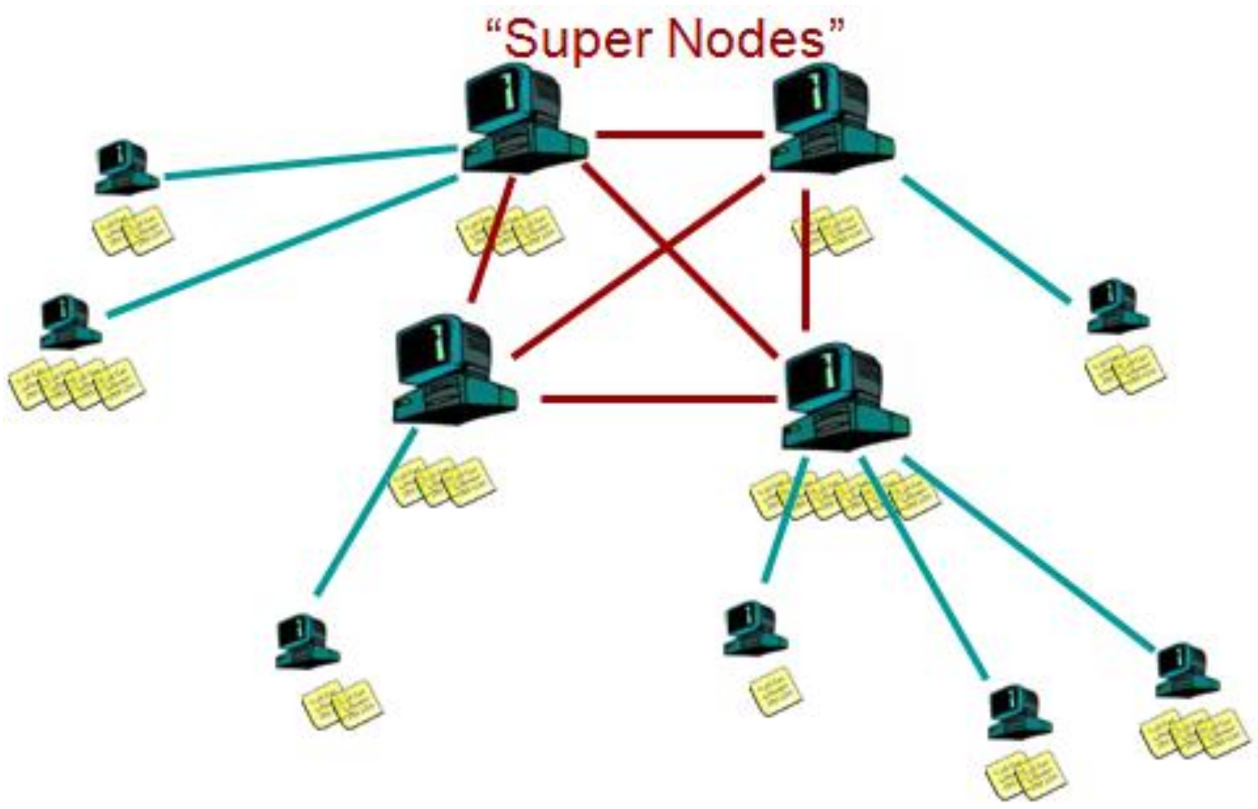
# Unstructured Architectures

## KaZaA

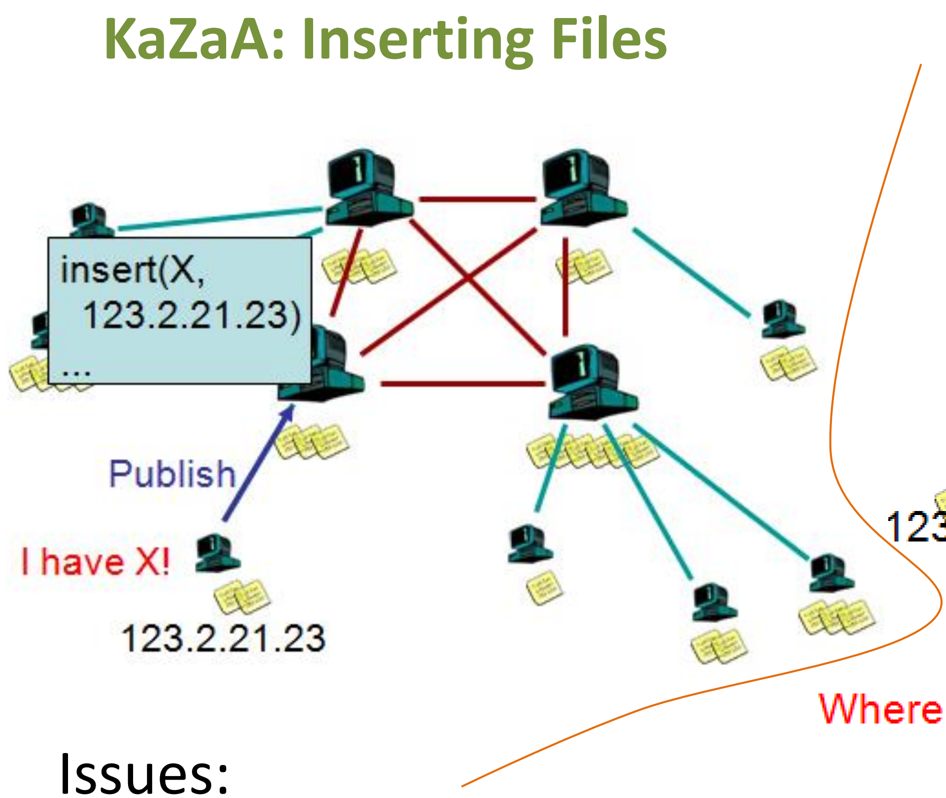
- 2001: KaZaA is launched
- Client application comes up: Morpheus, giFT
- It utilizes a mechanism of “*smart*” *query flooding*:
  - **Join**: at entrance, the client contacts a *super-node* (it can become super-node at a time)
  - **Publish**: sends list of files to super-node
  - **Search**: send a query to the *super-node*, and *super-nodes* interrogate each other
  - **Fetch**: take the file directly from the *peer(s);* the file can be obtained simultaneously from multiple peers

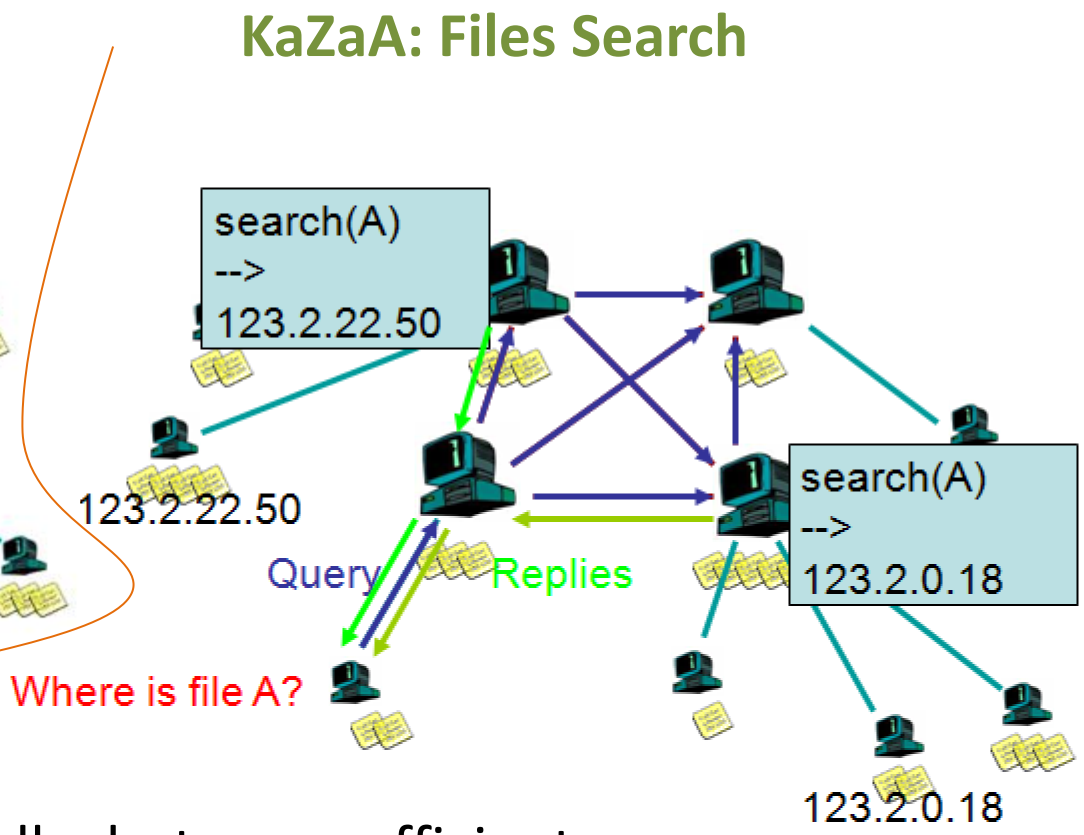
# Unstructured Architectures

## KaZaA: Network design



# Unstructured Architectures





Issues:

- Behaviour similar to Gnutella, but more efficient
- There is no guarantee on the search time or on the search area

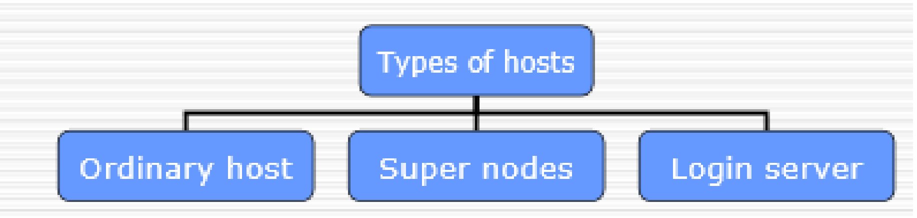
## Unstructured Architectures

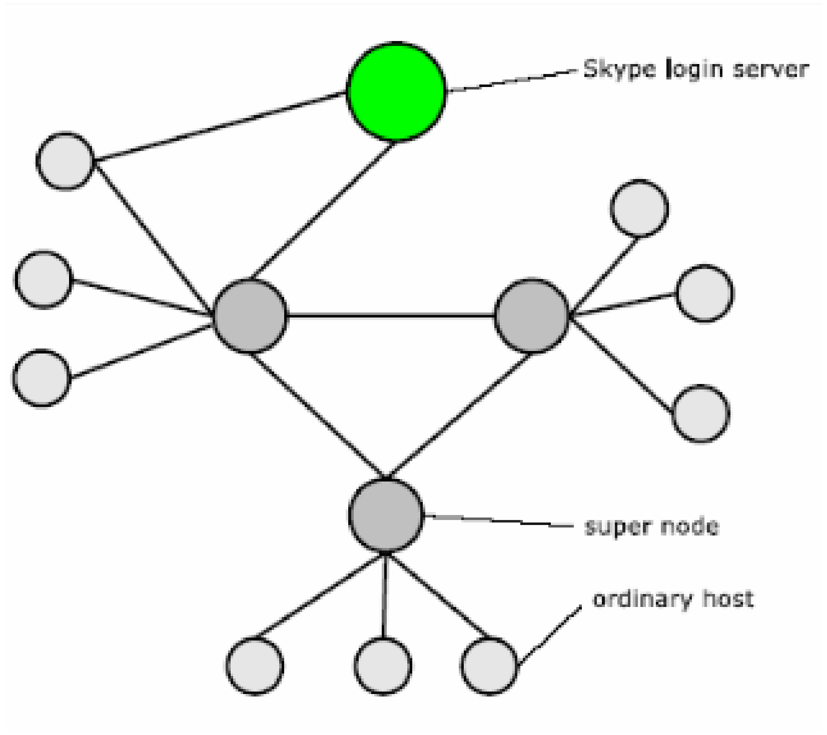
**Partially centralized**

- KaZaA software is a proprietary one
- P2P control data are encrypted
- The messages use HTTP
- A node is either a super-node, or assigned to a super-node
- A super-node has 100-150 child-nodes
- A network can have ~ 30,000 super-nodes
- Each super node has TCP connections with 30-50 super-nodes
- For each file meta-data are maintained (name, size, content hash, file descriptor)
- *The content hash* is used to find another copy of a partially transferred file
- The version without *spyware* and *pop-ups*: **KaZaA-lite**

# Unstructured Architectures

## Skype



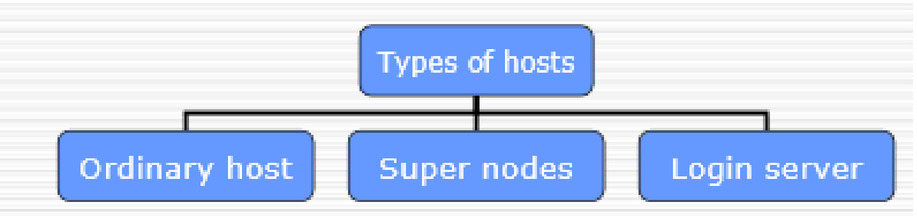


- The first P2P telephony network based on IP
- from June 2014, Microsoft announced the incompatibility with the previous Skype protocol
- Uses Microsoft Notification Protocol 24 ( first use -> MSN Messenger in 1999)
- architecture was similar to KaZaA

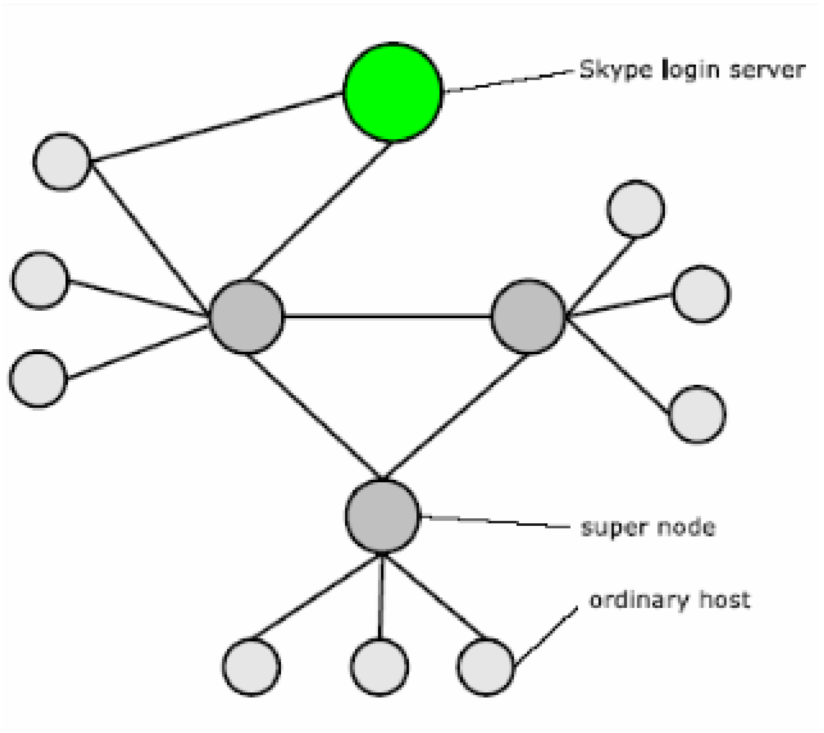
http://www1.cs.columbia.edu/~salman/publications/skype1_4.pdf

# Unstructured Architectures

## Skype



- Each client maintains a *host cache* with IP addresses and the port numbers associated with the accessible super-nodes
- Any customer with bandwidth (and without firewall or NAT restrictions) could become a super-node
- from 2012, Microsoft began super-nodes hosting servers in its data centers



http://en.wikipedia.org/wiki/PRISM_%28surveillance_program%29

# Unstructured Architectures

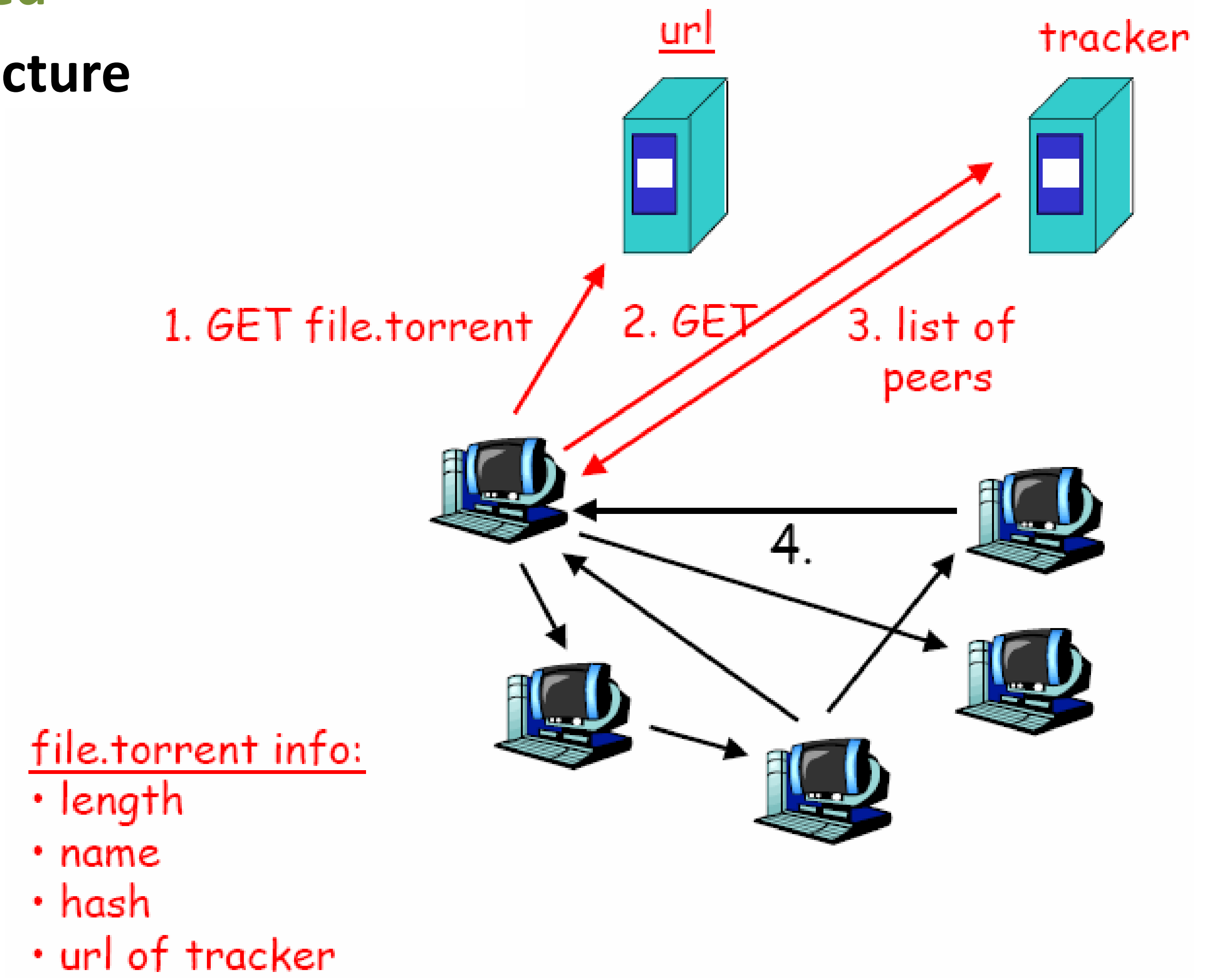
## Partially centralized

- If a file is found on multiple nodes, the transfer can be done in parallel
  - The same copies are identified via *content hash*
- Different chunks of the file is transferred from different nodes
- For interrupted transfers, an automatic recovery is performed
- Example: **BitTorrent**
  - In 2002, B. Cohen released BitTorrent
  - The focus was over *efficient fetching* and not on *searching*
  - Supporters since they have appeared
    - Blizzard Entertainment use BitTorrent to distribute beta versions of new games

# Unstructured Architectures

**Partially centralized**

**BitTorrent - arhitecture**



# Unstructured Architectures

**Partially centralized**

**BitTorrent**

- It is based on *swarming* mechanism:
  - Join: contact a centralized server (*tracker*) and get a list of *peers*
  - Publish: running a *tracker* server
  - Search: e.g. uses Google to find a *tracker* for the desired file
  - Fetch: Take pieces of files from peers;
- Obs. The difference from Napster
  - *File Chunk Downlod*
  - Using the "tit-for-tat" strategy: A  may *download* from other nodes, then A should allow download from it  (*free-rider problem*)

# Unstructured Architectures

## Problems

- **Nodes whose IP addresses are available via NAT (with restrictions)**
  - They can be TCP servers for P2P network
  - Partial solution: *reverse call*
    - A wants to transfer from B and the B uses NAT
    - A and B establish TCP connections with C server (with a routable IP)
    - A may request to B, via C, to create a TCP connection from B to A
    - A can send a request to B, and B gives the answer
  - If A and B use NAT?
- **Flash crowd: an unexpected increase of demand for a particular resource**
  - For the wanted content, there are no enough copies
  - How much time does it take for a user to locate the file?
  - How many messages will receive a node due to searches made by other nodes?
  - A generic protocol, based on TTL can be used

# Structured Architectures

- Represents academic solution for P2P
- Scope:
  - Successful search
  - Search time is performed in known boundaries
  - proven scalability
- Approach: **DHT (Distributed Hash Table)**
  - Pairs (key, value) are stored
    - Key – file names
    - Value – file content or a pointer to a location
  - Each *peer* stores a set (key, value)
  - *Operations: find the node responsible for a Key*
    - Mapping *key – node*
    - Efficient routing to *insert/lookup/delete* operations associated with this node
    - A wide fluctuation of nodes is allowed

# Structured Architectures

- Aspects: **content localization**
- The intuition: The responsibility is distributed to multiple nodes of the coverage network, in an adaptive manner
- To each resource a unique key is associated via a *hash* function: h("Curs Retele")->7929; The range values of the *hash function* is distributed in P2P network
- Each node must "know" the location of at least one copy of a resource for which the hash function has values in its range
- Nodes can maintain its own cache with copies of each resource that they need to "know"

# Structured Architectures

- Aspects: **routing**
- For each resource, a node that “knows” the resource must be accessed through the shortest path
- Structured P2P systems approaches differ from one routing strategy to another
- Nodes from the system forms a distributed structure that can be: a ring, tree, hypercube,… etc.
- It provides an API for distributed hash tables sites (DHT - Distributed Hash Table)
  - Giving a key k, the API will return the IP address of the node responsible for the k key value

## Structured Architectures

- **Implementations**
  - Chord [MIT]
  - Pastry [Microsoft Research UK, Rice University]
  - Tapestry [UC Berkeley]
  - Content Addressable Network (CAN) [UC Berkeley]
  - SkipNet [Microsoft Research US, Univ. of Washington]
  - Kademlia [New York University]
  - Viceroy [Israel, UC Berkeley]
  - P-Grid [EPFL Switzerland]

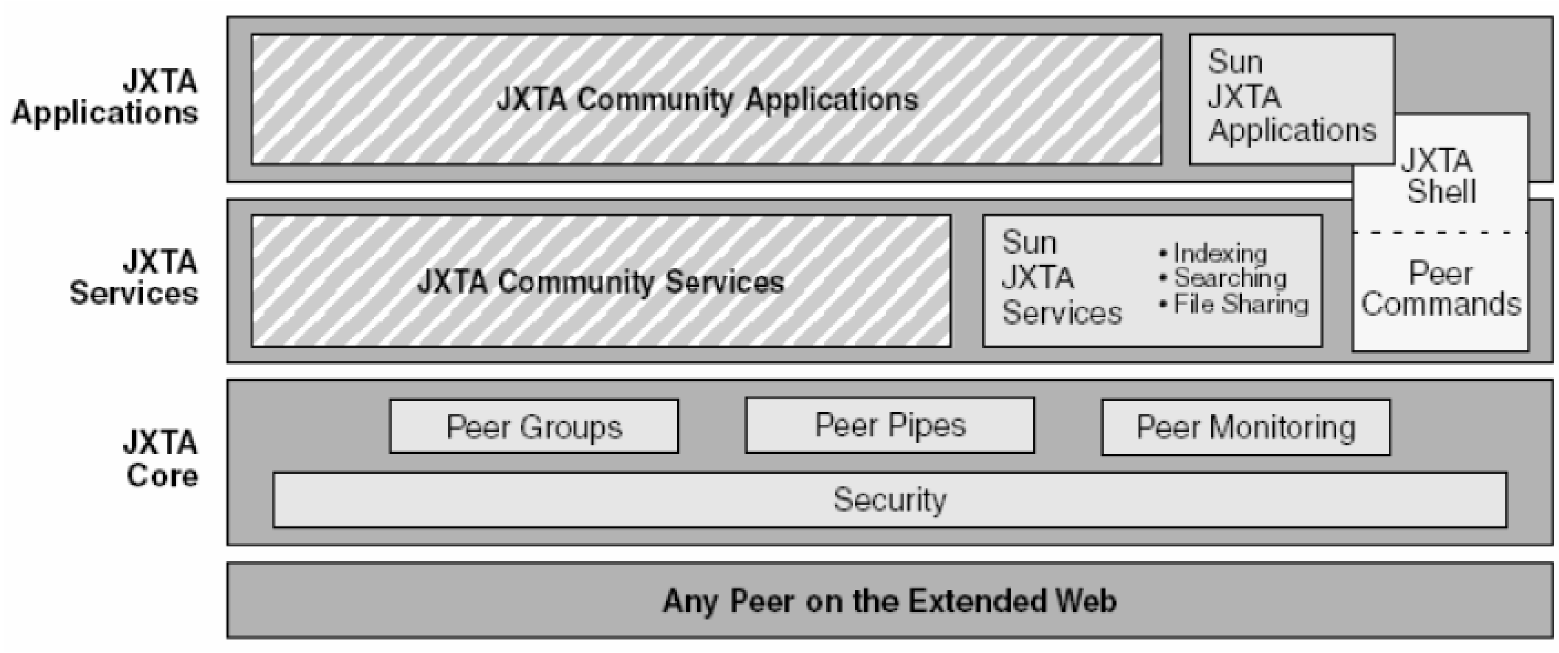
# Structured Architectures

- **Loosely structured**
  - The nodes can estimate which nodes store the search resources
    - Blind broadcast are avoided
    - *Chain mode propagation* mechanism is used: each node takes local decisions regarding who will be the next node which will be queried
  - Search for a file involves using a key and a timeout mechanism
  - Example: **Freenet**

[https://en.wikipedia.org/wiki/Freenet]

# Instruments

- JXTA – www.jxta.org
  - Development environment for P2P systems & applications
  - Based on Java - available under open source



## Instruments

- P2P – framework for Android

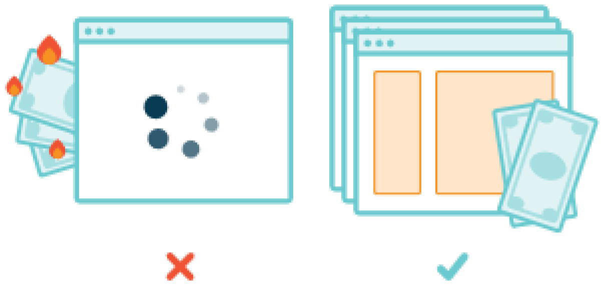
  https://code.google.com/p/p2p-communication-framework-for-android/
- p2psim – simulator for p2p protocol

  http://pdos.csail.mit.edu/p2psim/
- Instruments and protocols for P2P:

  http://en.wikibooks.org/wiki/The_World_of_Peer-to-Peer_%28P2P%29/Networks_and_Protocols/Other_Software_Implementations
- “ IPFS is the Distributed Web” - https://ipfs.io/

# IPFS

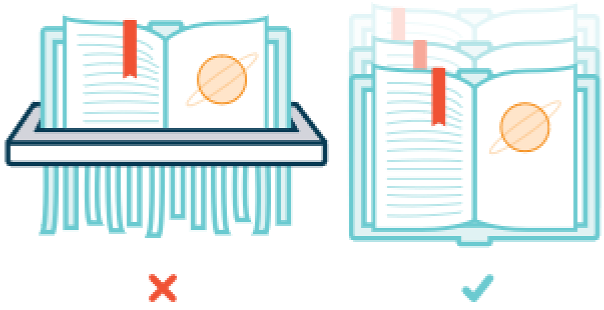
- “ IPFS is the Distributed Web” - https://ipfs.io/
  - –**A peer-to-peer hypermedia protocol to make the web faster, safer, and more open**



**HTTP is inefficient and expensive**

HTTP downloads a file from a single computer at a time, instead of getting pieces from multiple computers simultaneously. With video delivery, a P2P approach could save 60% in bandwidth costs.

IPFS makes it possible to distribute high volumes of data with high efficiency. And zero duplication means savings in storage.



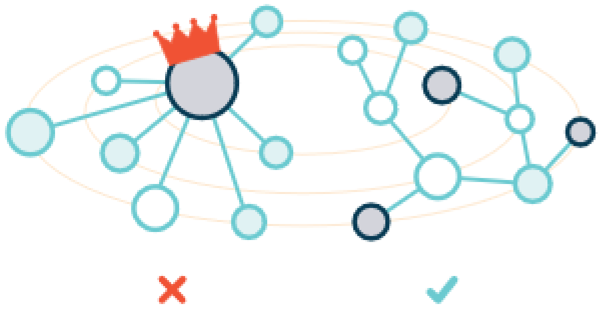
**Humanity's history is deleted daily**

The average lifespan of a web page is 100 days. Remember GeoCities? The web doesn't anymore. It's not good enough for the primary medium of our era to be so fragile.

IPFS provides historic versioning (like git) and makes it simple to set up resilient networks for mirroring of data.

# IPFS

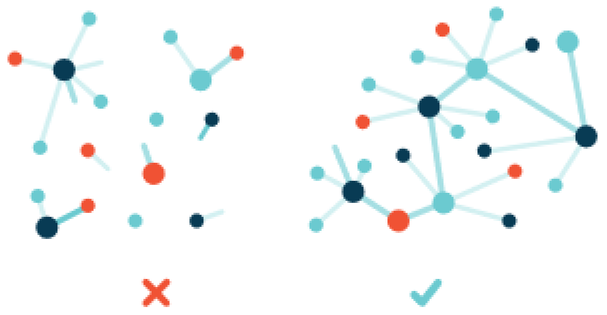
- “ IPFS is the Distributed Web” - https://ipfs.io/
  - A peer-to-peer hypermedia protocol to make the web faster, safer, and more open



**The web's centralization limits opportunity**

The Internet has been one of the great equalizers in human history and a real accelerator of innovation. But the increasing consolidation of control is a threat to that.

IPFS remains true to the original vision of the open and flat web, but delivers the technology which makes that vision a reality.



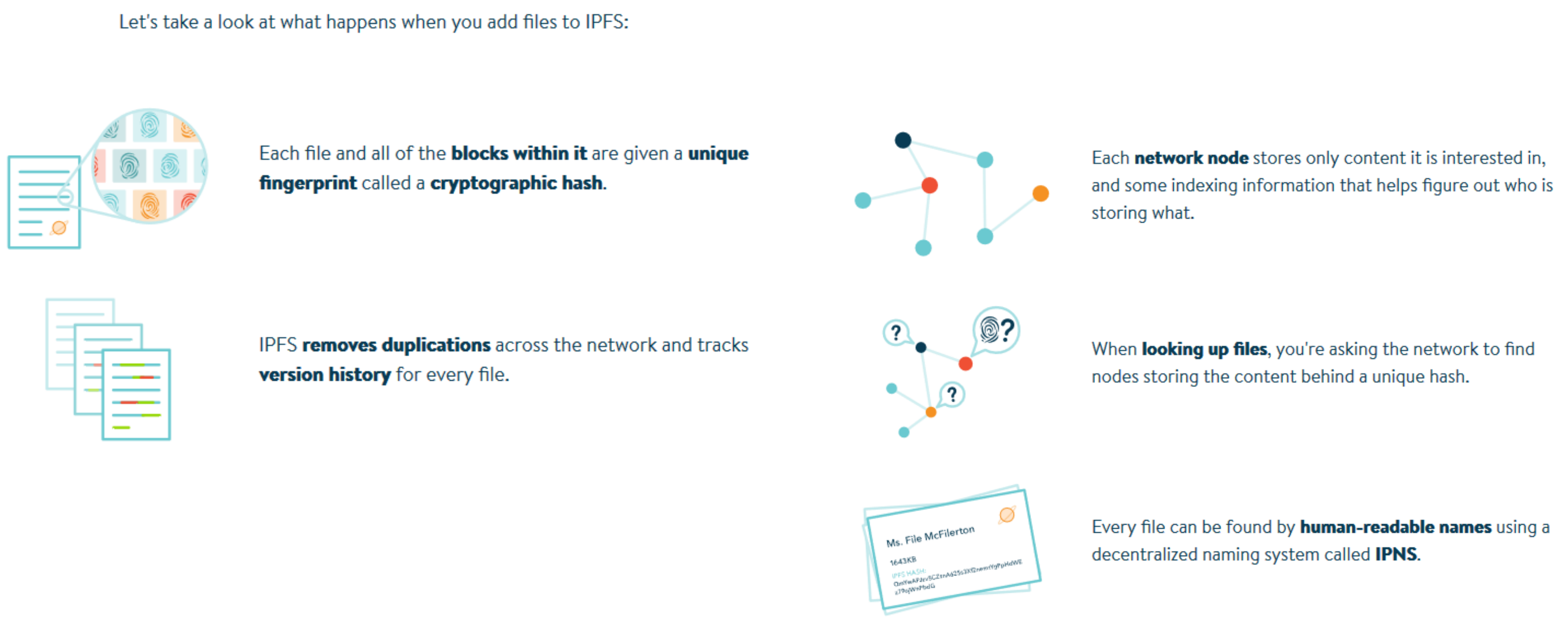
**Our apps are addicted to the backbone**

Developing world. Offline. Natural disasters. Intermittent connections. All trivial compared to interplanetary networking. The networks we're using are so 20th Century. We can do better.

IPFS powers the creation of diversely resilient networks which enable persistent availability with or without Internet backbone connectivity.

# IPFS

- “ IPFS is the Distributed Web” - https://ipfs.io/



Let's take a look at what happens when you add files to IPFS:

Each file and all of the **blocks within it** are given a **unique fingerprint** called a **cryptographic hash**.

IPFS **removes duplications** across the network and tracks **version history** for every file.

Each **network node** stores only content it is interested in, and some indexing information that helps figure out who is storing what.

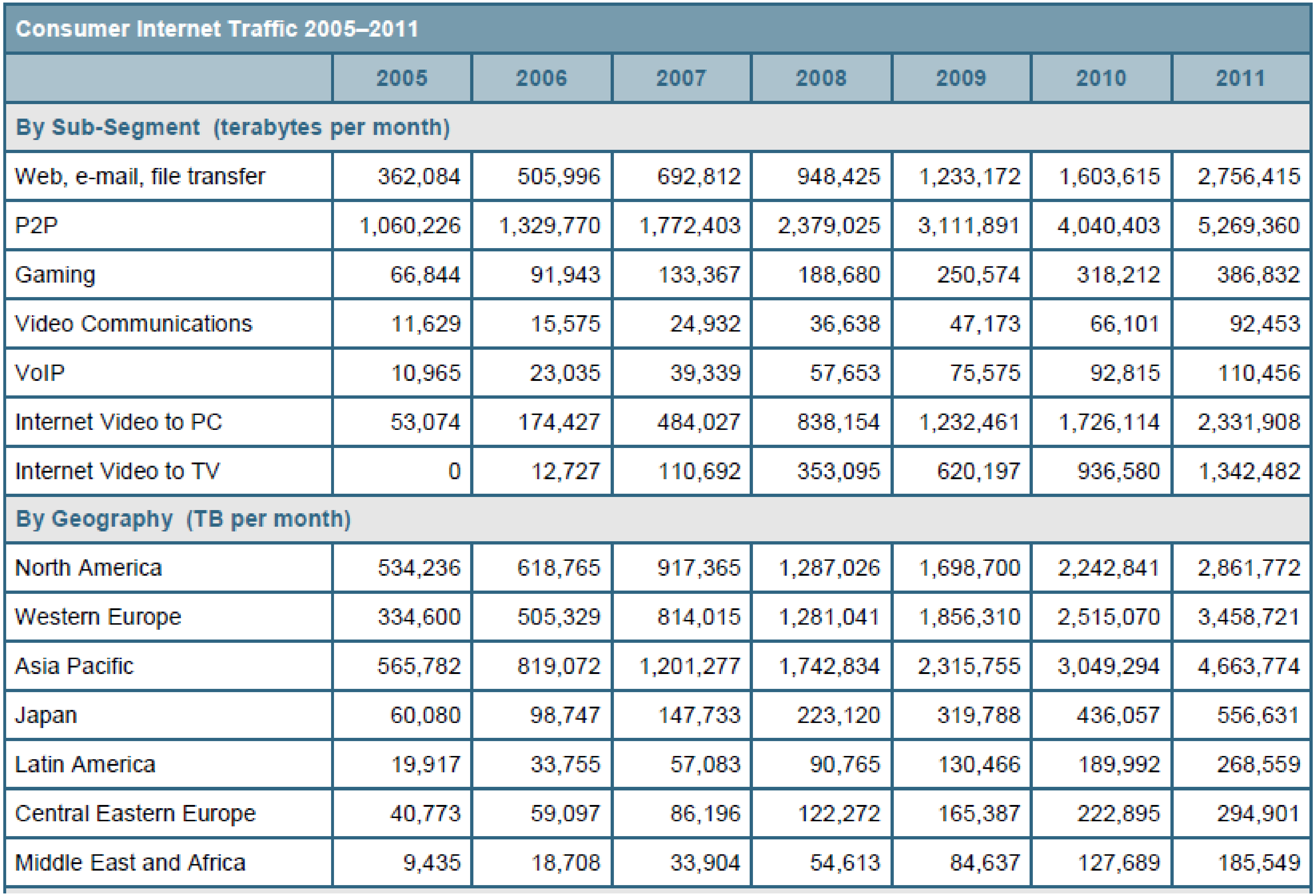
When **looking up files**, you're asking the network to find nodes storing the content behind a unique hash.

Every file can be found by **human-readable names** using a decentralized naming system called **IPNS**.

•https://github.com/ipfs/papers/raw/master/ipfs-cap2pfs/ipfs-p2p-file-system.pdf

## Statistics

Global Consumer Internet Traffic 2005–2011



| Consumer Internet Traffic 2005–2011 | | | | | | | |
|---|---|---|---|---|---|---|---|
| | 2005 | 2006 | 2007 | 2008 | 2009 | 2010 | 2011 |
| **By Sub-Segment (terabytes per month)** | | | | | | | |
| Web, e-mail, file transfer | 362,084 | 505,996 | 692,812 | 948,425 | 1,233,172 | 1,603,615 | 2,756,415 |
| P2P | 1,060,226 | 1,329,770 | 1,772,403 | 2,379,025 | 3,111,891 | 4,040,403 | 5,269,360 |
| Gaming | 66,844 | 91,943 | 133,367 | 188,680 | 250,574 | 318,212 | 386,832 |
| Video Communications | 11,629 | 15,575 | 24,932 | 36,638 | 47,173 | 66,101 | 92,453 |
| VoIP | 10,965 | 23,035 | 39,339 | 57,653 | 75,575 | 92,815 | 110,456 |
| Internet Video to PC | 53,074 | 174,427 | 484,027 | 838,154 | 1,232,461 | 1,726,114 | 2,331,908 |
| Internet Video to TV | 0 | 12,727 | 110,692 | 353,095 | 620,197 | 936,580 | 1,342,482 |
| **By Geography (TB per month)** | | | | | | | |
| North America | 534,236 | 618,765 | 917,365 | 1,287,026 | 1,698,700 | 2,242,841 | 2,861,772 |
| Western Europe | 334,600 | 505,329 | 814,015 | 1,281,041 | 1,856,310 | 2,515,070 | 3,458,721 |
| Asia Pacific | 565,782 | 819,072 | 1,201,277 | 1,742,834 | 2,315,755 | 3,049,294 | 4,663,774 |
| Japan | 60,080 | 98,747 | 147,733 | 223,120 | 319,788 | 436,057 | 556,631 |
| Latin America | 19,917 | 33,755 | 57,083 | 90,765 | 130,466 | 189,992 | 268,559 |
| Central Eastern Europe | 40,773 | 59,097 | 86,196 | 122,272 | 165,387 | 222,895 | 294,901 |
| Middle East and Africa | 9,435 | 18,708 | 33,904 | 54,613 | 84,637 | 127,689 | 185,549 |

# Statistics

Legend:

Web, E-mail, and File Transfer – includes Web, e-mail, instant messaging, newsgroups, and file transfer (excluding P2P and commercial file transfer such as iTunes)

P2P – includes peer-to-peer traffic from all recognized P2P systems such as BitTorrent, eDonkey, etc.

Gaming – includes casual online gaming, networked console gaming, and multiplayer virtual world gaming

Video Communications – includes PC-based video calling, Webcam viewing, and Web-based video monitoring

VoIP – includes traffic from retail VoIP services and PC-based VoIP, but excludes wholesale VoIP transport

Internet Video to PC – free or pay TV or VoD viewed on a PC, excludes P2P video file downloads

Internet Video to TV – free or pay TV or VoD delivered via Internet but viewed on a TV screen using a STB or media gateway

Global Consumer Peer-to-Peer Traffic 2005–2011

| Consumer Peer-to-Peer Traffic 2005–2011 | | | | | | | |
|---|---|---|---|---|---|---|---|
| | 2005 | 2006 | 2007 | 2008 | 2009 | 2010 | 2011 |
| **By Geography (TB per month)** | | | | | | | |
| North America | 381,746 | 378,538 | 462,356 | 560,817 | 673,083 | 852,483 | 1,080,979 |
| Western Europe | 223,519 | 304,988 | 411,057 | 540,032 | 757,818 | 991,817 | 1,330,885 |
| Asia Pacific | 391,235 | 550,664 | 762,276 | 1,074,759 | 1,401,028 | 1,811,094 | 2,327,648 |
| Japan | 28,621 | 42,883 | 58,463 | 87,446 | 117,967 | 154,868 | 206,803 |
| Latin America | 8,732 | 14,358 | 23,247 | 37,284 | 53,587 | 80,043 | 117,731 |
| Central Eastern Europe | 22,075 | 31,009 | 43,117 | 59,928 | 79,589 | 106,543 | 141,282 |
| Middle East and Africa | 4,297 | 7,329 | 11,886 | 18,759 | 28,819 | 43,553 | 64,033 |
| **Total (TB per month)** | | | | | | | |
| **Peer-to-Peer Traffic** | 1,060,226 | 1,329,770 | 1,772,403 | 2,379,025 | 3,111,891 | 4,040,403 | 5,269,360 |

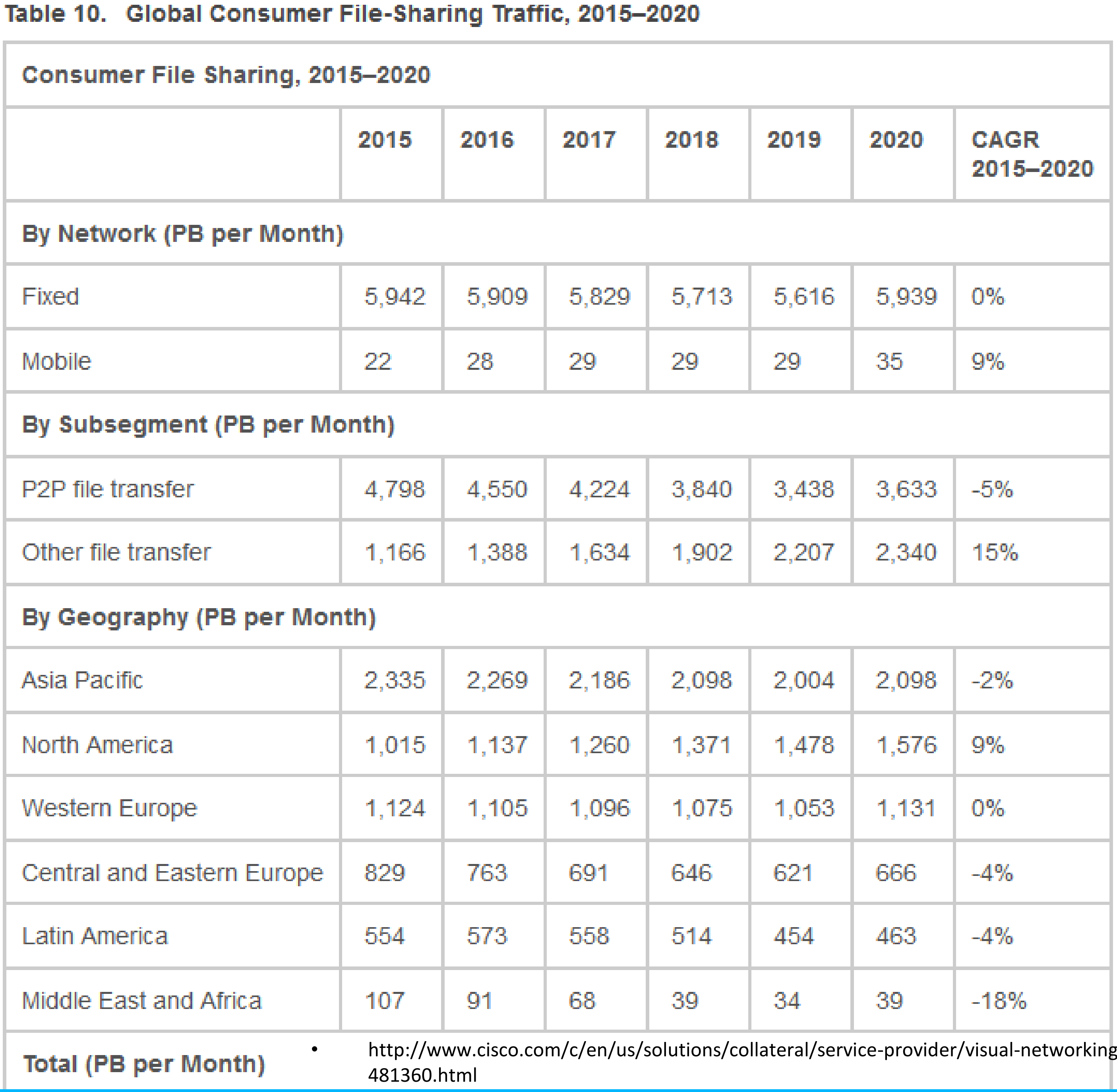


Table 10. Global Consumer File-Sharing Traffic, 2015–2020

| Consumer File Sharing, 2015–2020 | | | | | | | |
|---|---|---|---|---|---|---|---|
| | 2015 | 2016 | 2017 | 2018 | 2019 | 2020 | CAGR 2015–2020 |
| **By Network (PB per Month)** | | | | | | | |
| Fixed | 5,942 | 5,909 | 5,829 | 5,713 | 5,616 | 5,939 | 0% |
| Mobile | 22 | 28 | 29 | 29 | 29 | 35 | 9% |
| **By Subsegment (PB per Month)** | | | | | | | |
| P2P file transfer | 4,798 | 4,550 | 4,224 | 3,840 | 3,438 | 3,633 | -5% |
| Other file transfer | 1,166 | 1,388 | 1,634 | 1,902 | 2,207 | 2,340 | 15% |
| **By Geography (PB per Month)** | | | | | | | |
| Asia Pacific | 2,335 | 2,269 | 2,186 | 2,098 | 2,004 | 2,098 | -2% |
| North America | 1,015 | 1,137 | 1,260 | 1,371 | 1,478 | 1,576 | 9% |
| Western Europe | 1,124 | 1,105 | 1,096 | 1,075 | 1,053 | 1,131 | 0% |
| Central and Eastern Europe | 829 | 763 | 691 | 646 | 621 | 666 | -4% |
| Latin America | 554 | 573 | 558 | 514 | 454 | 463 | -4% |
| Middle East and Africa | 107 | 91 | 68 | 39 | 34 | 39 | -18% |
| **Total (PB per Month)** | | | | | | | |

**File Sharing**
This category includes traffic from P2P applications such as BitTorrent and eDonkey, as well as web-based file sharing. Note that a large portion of P2P traffic is due to the exchange of video files, so a total view of the impact of video on the network should count P2P video traffic in addition to the traffic counted in the Internet video-to-PC and Internet video-to-TV categories. Table 10 shows the forecast for consumer P2P traffic from 2015 to 2020. Note that the P2P category is limited to traditional file exchange and does not include commercial video-streaming applications that are delivered through P2P, such as PPStream or PPLive.

- http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html

# Bibliography

- P2P Networking and Applications, John F. Buford, Heather Yu, Eng Keong Lua ,2009, Elsevier
- http://mtc.sri.com/Conficker/P2P/
- http://www.cisco.com/en/US/docs/cable/serv_exch/serv_control/broadband_app/protocol_ref_guide/01_p2p.pdf
- http://pdos.csail.mit.edu/p2psim/
- Statistici: http://www.sandvine.com/news/pr_detail.asp?ID=312
- Statistici: http://www.hbtf.org/files/cisco_IPforecast.pdf
- http://en.wikibooks.org/wiki/The_World_of_Peer-to-Peer_%28P2P%29/Networks_and_Protocols/Other_Software_Implementations
- https://www.kirsle.net/blog/entry/skype-switched-to-the-msn-messenger-protocol

## Summary

- **Peer-to-peer(P2P) paradigm**
  - Preliminaries
  - Definitions
  - Characteristics
  - Application types
  - Infrastructures
  - Instruments

## Questions?

Questions?