10th Slide Set Cloud Computing

Prof. Dr. Christian Baun

Frankfurt University of Applied Sciences
(1971–2014: Fachhochschule Frankfurt am Main)
Faculty of Computer Science and Engineering
christianbaun@fb2.fra-uas.de

Agenda for Today

- Peer-to-Peer (P2P)
  - Fundamentals
  - Fields of application
  - Centralized P2P
    - Napster
  - Pure P2P
    - Gnutella version 0.4
  - Hybrid P2P
    - Gnutella version 0.6
    - FastTrack
  - BitTorrent
  - Distributed hash table
  - Bitcoins
  - P2PTV
  - Wuala
  - P2P and Cloud

Peer-to-Peer – Fundamentals

- A Peer-to-Peer system is a network of equal nodes
  - Node are called Peers
  - Nodes provide access to resources to each other
  - Each node is Client and Server at the same time
- Can be used to build up decentralized working environments

Andy Oram

„A Peer-to-Peer system is a self-organizing system of equal, autonomous entities (peers) which aims for the shared usage of distributed resources in a networked environment avoiding central services."

Spelling

A standard spelling does not exist. Often, Peer-to-Peer is also written P2P or Peer-2-Peer (with or without dashes)

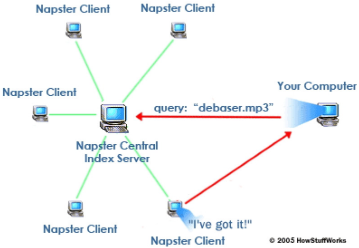
Fields of Application of Peer-to-Peer

- File Sharing
  - Objective: Distribution of large amounts of data as quickly as possible to many users
  - Applications: BitTorrent, Napster, Gnutella, Kazaa, Overnet
- Communication and collaboration
  - Instant messaging systems for text messages and chat
  - Secure and anonymous data exchange (Freenet)
    - `https://freenetproject.org`
  - Applications: ICQ, AIM, MSN, Jabber, Freenet, usw.

Napster

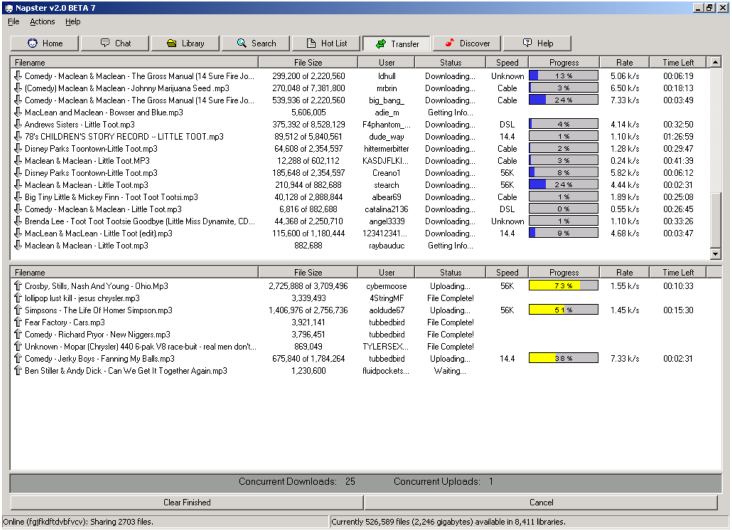
- Pioneer of file-sharing
  - Developed by Shawn Fanning (a student in Boston)
  - Active: June 1999 – July 2001
- Napster implemented centralized P2P
  - Napster contained centralized indexing
  - Clients uploaded their file list to the Napster server
  - Clients search in the file list at the index server and receive a provider list from the index server
  - Exchange of files was carried out directly between Peers

- Since December 1999: Massive legal trouble with music industry and artists
- 2001: Napster has 77 million users
- July 2001: Napster is shut down



Napster (Frequent screen of the days. . .)

Image Source: Wikipedia



Filename	File Size	User	Status	Speed	Progress	Rate	Time Left
Comedy - Maclean & Maclean - The Gross Manual (14 Sure Fire Jo...	299,200 of 2,220,560	ldhull	Downloading...	Unknown	13 %	5.06 k/s	00:06:19
(Comedy) Maclean & Maclean - Johnny Marijuana Seed .mp3	270,048 of 7,381,800	mrbrin	Downloading...	Cable	3 %	6.50 k/s	00:18:13
Comedy - Maclean & Maclean - The Gross Manual (14 Sure Fire Jo...	539,936 of 2,220,560	big_bang_	Downloading...	Cable	24 %	7.33 k/s	00:03:49
MacLean and Maclean - Bowser and Blue.mp3	5,606,005	adie_m	Getting Info...				
Andrews Sisters - Little Toot.mp3	375,392 of 8,528,129	F4phantom_...	Downloading...	DSL	4 %	4.14 k/s	00:32:50
78's CHILDREN'S STORY RECORD -- LITTLE TOOT.mp3	89,512 of 5,840,561	dude_way	Downloading...	14.4	1 %	1.10 k/s	01:26:59
Disney Parks Toontown-Little Toot.mp3	64,608 of 2,354,597	hittermerbitter	Downloading...	Cable	2 %	1.28 k/s	00:29:47
Maclean & Maclean - Little Toot.MP3	12,288 of 602,112	KASDJFLKI...	Downloading...	Cable	3 %	0.24 k/s	00:41:39
Disney Parks Toontown-Little Toot.mp3	185,648 of 2,354,597	Creano1	Downloading...	56K	8 %	5.82 k/s	00:06:12
Maclean & Maclean - Little Toot.mp3	210,944 of 882,688	stearch	Downloading...	56K	24 %	4.44 k/s	00:02:31
Big Tiny Little & Mickey Finn - Toot Toot Tootsi.mp3	40,128 of 2,888,844	albear69	Downloading...	Cable	1 %	1.89 k/s	00:25:08
Comedy - Maclean & Maclean - Little Toot.mp3	6,816 of 882,688	catalina2136	Downloading...	DSL	0 %	0.55 k/s	00:26:45
Brenda Lee - Toot Toot Tootsie Goodbye (Little Miss Dynamite, CD...	44,368 of 2,250,710	angel3339	Downloading...	Unknown	1 %	1.10 k/s	00:33:26
MacLean & MacLean - Little Toot (edit).mp3	115,600 of 1,180,444	123412341...	Downloading...	14.4	9 %	4.68 k/s	00:03:47
Maclean & Maclean - Little Toot.mp3	882,688	raybauduc	Getting Info...				

Filename	File Size	User	Status	Speed	Progress	Rate	Time Left
Crosby, Stills, Nash And Young - Ohio.Mp3	2,725,888 of 3,709,496	cybermoose	Uploading...	56K	73 %	1.55 k/s	00:10:33
lollipop lust kill - jesus chrysler.mp3	3,339,493	4StringMF	File Complete!				
Simpsons - The Life Of Homer Simpson.mp3	1,406,976 of 2,756,736	aoldude67	Uploading...	56K	51 %	1.45 k/s	00:15:30
Fear Factory - Cars.mp3	3,921,141	tubbedbird	File Complete!				
Comedy - Richard Pryor - New Niggers.mp3	3,796,451	tubbedbird	File Complete!				
Unknown - Mopar (Chrysler) 440 6-pak V8 race-buit - real men don't...	869,049	TYLERSEX...	File Complete!				
Comedy - Jerky Boys - Fanning My Balls.mp3	675,840 of 1,784,264	tubbedbird	Uploading...	14.4	38 %	7.33 k/s	00:02:31
Ben Stiller & Andy Dick - Can We Get It Together Again.mp3	1,230,600	fluidpockets...	Waiting...				

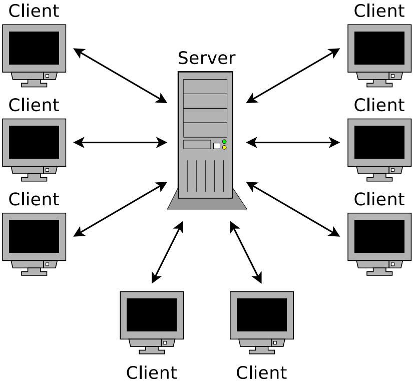
Concurrent Downloads: 25 Concurrent Uploads: 1

Online (fgjfkdftdvbfvcv): Sharing 2703 files.

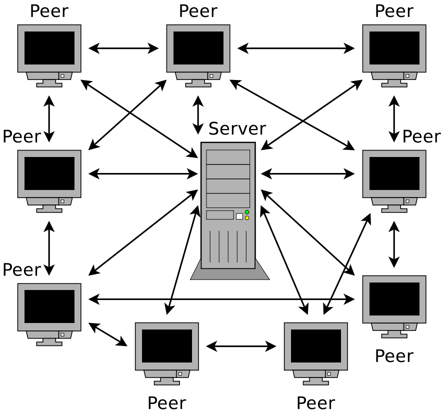
Currently 526,589 files (2,246 gigabytes) available in 8,411 libraries.

Client-Server compared with Centralized P2P

Client-Server



Centralized Peer-to-Peer



- What are advantages and drawbacks of centralized P2P?

Advantages and Drawbacks of Centralized P2P

- Advantages:
  - Peers communicate directly with each other
    - Server has only a few functions
  - Excellent extensibility
    - Each additional Peer causes only little load on the server
  - Failure of one or more Peers does not damage the network and the availability of services
  - Centralized services provide high performance
  - Centralized service is a well-known entry point
    - New Peers can easily become a part of the system
- Drawbacks:
  - Servers cause costs (electricity, space, administration,. . .)
  - Without the centralized services, the system does not work
  - Centralized services are always a point of attack
    - Easy to attack via lawsuits

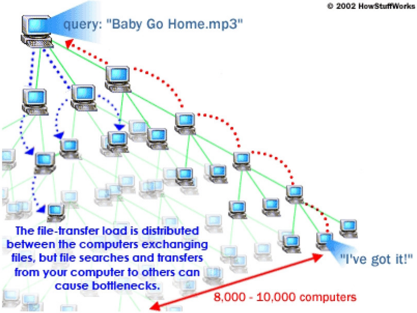
Gnutella

- Developed by Justin Frankel and Tom Pepper at Nullsoft (Winamp)
- Licensed under the terms of the GNU General Public License (GPL)
- March 2000: Initial release
- Reason for the development:
  - Growing trouble of Napster in 1999/2000
    - Music industry initiated by lawsuits
  - Napster used centralized index servers
    - Central server make a service vulnerable!
  - Objective: Development of an alternative, which does not require servers
- AOL – owner of Nullsoft – withdraw the software
  - But the source code was already released under the terms of the GPL...
- Popular Clients: LimeWire, Morpheus, Shareaza, Bearshare
- Today, Gnutella is a synonymous for the protocol and not for a specific software

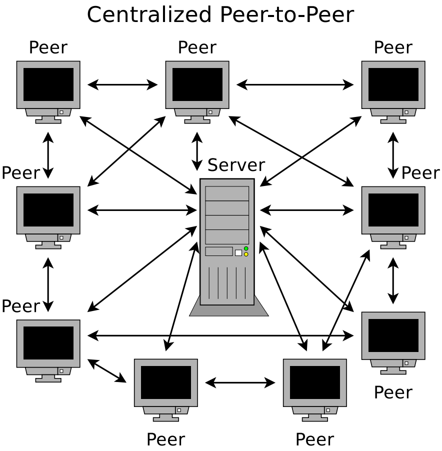
Gnutella Protocol Version 0.4

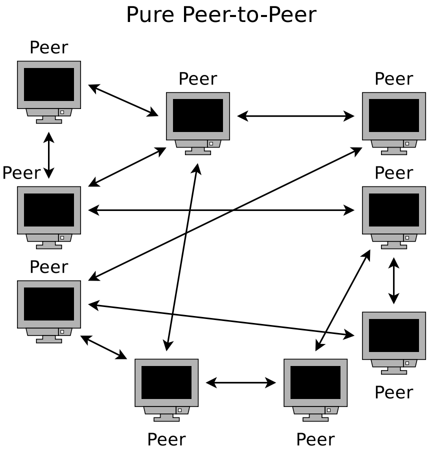
- `http://rfc-gnutella.sourceforge.net/developer/stable/index.html`
- All Peers are equal participants in the network
- No centralized services (servers) ⟹ **pure P2P**
- Each Peer needs to know at least 1 additional node
- Each Peer maintains a list of the peers, known to it
- Each Peer is actively connected to up to 5 other Peers

- Each Peer sends search requests to all other Peers, to which it is actively connected with
  - Forwardings are called **Hops**
  - Maximum number of Hops per search request: 7
- Drawbacks:
  - Search requests take a long time
  - Network gets flooded with search requests


© 2002 HowStuffWorks
query: "Baby Go Home.mp3"
The file-transfer load is distributed between the computers exchanging files, but file searches and transfers from your computer to others can cause bottlenecks.
"I've got it!"
8,000 - 10,000 computers

Centralized P2P compared with Pure P2P





- What are advantages and drawbacks of pure P2P?

Advantages and Drawbacks of Pure P2P

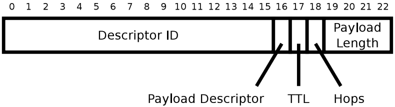
- Advantages:
  - Independent from servers
    - Peers communicate directly with each other
  - No central points of attack
    - System (network) is hard to destroy
    - Failure of one or more Peers does not damage the network and the availability of services
  - No servers $\Longrightarrow$ no administration effort
- Drawbacks:
  - Difficult to access the system
    - At least a single Peer must be already known
  - Search requests stress the network a lot
  - Lists of data and Peers must be spread (high overhead)
- Examples of pure P2P: Gnutella (version 0.4) and Freenet

Technical Details of Gnutella 0.4 (1/2)

- The Gnutella 0.4 protocol works with TCP/IP connections
- Connection establishment:
  - Connection request, sent by a new client:
    `GNUTELLA CONNECT/0.4\n\n`
  - If the desired Peer wants to respond, it sends:
    `GNUTELLA OK\n\n`
- Afterwards Gnutella messages can be exchanged

Technical Details of Gnutella 0.4 (2/2)

- Structure of the header of Gnutella messages:
  - Descriptor ID (16 Byte). Unique identifikation of the transmitting Peer
  - Payload Descriptor (1 Byte). Can be. . .
    - 0x00 = **Ping** (Discover the presence of other nodes)
    - 0x01 = **Pong** (Response to a Ping message)
    - 0x40 = **Push** (Download files)
    - 0x80 = **Query** (Search request in the Gnutella network)
    - 0x81 = **Query Hit** (Response to a Query message)
  - TTL (1 Byte). Specifies the maximum number of Hops for a message
    - Is decremented with every step in the network by value 1
    - If the value is 0, the message is thrown away
  - Hops (1 Byte). Counter for the Hops
    - Is incremented with every step in the network by value 1
  - Payload Length (4 Byte). Payload length in Bytes
    - Determines where the message stops and a new message starts



0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22

Descriptor ID

Payload Length

Payload Descriptor

TTL

Hops

Gnutella 0.4 Messages (1/3)

- The existence of other nodes is discovered via the **Ping** message
  - Ping contains no further information
- If a node has received a Ping message, it can respond with a **Pong** message
  - The reply (Pong message) contains these fields:
    - Port number and IP address of the responding Peer
    - Number of files, which can be downloaded by everybody
    - Size [kB] of the available files
- The **Query** message is used for search requests in the Gnutella network
  - The Query message contains these fields:
    - Minimum download bandwidth (in kB/sec), the requesting peer will accept (0 for any bandwidth)
    - The search string, usually filenames, which may contain wildcards (e.g. $*$)

Source: `http://krum.rz.uni-mannheim.de/inet-2004/sess-404.html`

Gnutella 0.4 Messages (2/3)

- A **QueryHit** message contains the response to a query message
  - The QueryHit message contains these fields:
    - Number of files, to which the search request matches
    - Port number and IP address of the responding Peer
    - Download bandwidth (in kB/sec), which the responding peer can provide ($\geq$ the requested download bandwidth)
    - For each file found: Index, file size in kB and file name
    - Information, if the responding peer is behind a firewall
    - Identification of the responding peers
    - *Vendor Code*, used to distinguish the different Gnutella implementations, e.g. `BEAR` = BearShare, `GNUC` = Gnucleus, `LIME` = LimeWire, `RAZA` = Shareaza, `MRPH` = Morpheus
  - If a Peer has no matching files, it does not create a QueryHit message

Source: `http://krum.rz.uni-mannheim.de/inet-2004/sess-404.html`

Transport of Files with Gnutella 0.4

- For the transport of files, Gnutella uses the HTTP 1.0 protocol
  - The HTTP connection is established directly between client and server, independent from the Gnutella network
    - The QueryHit message contains the IP address and port number
- HTTP `GET` is used to download a file, which is specified by a QueryHit response

```
GET /get/<Index in QueryHit>/<DateiName>/ HTTP/1.0\r\n
Connection: Keep-Alive\r\n
Range: bytes=0-\r\n
User-Agent: Gnutella\r\n
\r\n
```

- The responding Peer (server) sends the file via HTTP

```
HTTP 200 OK\r\n
Server: Gnutella\r\n
Content-type: application/binary\r\n
Content-length: xxx\r\n
\r\n
<xxx Bytes file content>
```

- Problem: What is done, if the server is behind a firewall? ⟹ **Push**

Gnutella 0.4 Messages (3/3)

- The **Push** message is used to download files, if the server is located behind a firewall
    - The Push message contains these fields:
        - Identification of the Peer, which provides the file (receiver of the message)
        - Port number and IP address of the requesting Peer
        - Index of the desired file, which is specified by a QueryHit response
    - Unlike HTTP `GET`, where the client establishes the TCP/IP connection to the server, the server establishes the connection to the client
        - The server sends a `GIV` Header

```
GIV <Index>:<PeerID>/<DateiName>\r\n
```

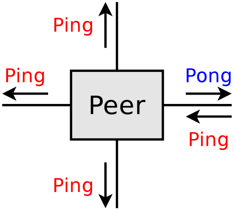
- Next, the client sends its HTTP `GET` request

```
GET /get/<Index>/<DateiName>/ HTTP/1.0\r\n
...
HTTP 200 OK\r\n
...
```

Source: `http://krum.rz.uni-mannheim.de/inet-2004/sess-404.html`

Forwarding (Routing) of Gnutella Messages (1/2)

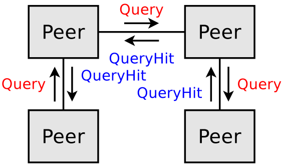
- When forwarding Gnutella messages. . .
  - the TTL field of each message is decremented (reduced)
    - If the TTL field has value 0, the message is discarded
  - the Hops field of each message is incremented (increased)
- If a Peer detects that a message has been received once again, within a (short) period of time, it discards the message
  - This is used for loop detection



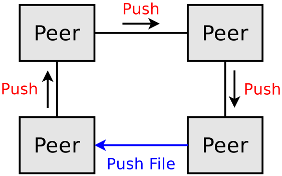
- Incoming Ping and Query messages are answered with Pong or a QueryHit messages
  - Pong messages contain the IP address
  - QueryHit messages contain the IP address and a list of matches
- Incoming Ping and Query messages are forwarded to all connections (except the the incoming one), if the TTL value is > 0

Forwarding (Routing) of Gnutella Messages (2/2)

- Incoming Pong and QueryHit messages are forwarded only on the connection, on which the corresponding Ping or Query messages arrived
  - Otherwise, they are discarded



- Push messages are forwarded only on the connection, on which the corresponding QueryHit message arrived
  - Otherwise, they are discarded



Source: http://krum.rz.uni-mannheim.de/inet-2004/sess-404.html

Pure P2P has Limitations

- For a high number of Peers, pure P2P systems become unusable, because of the network load
  - Slow
  - High network overhead
- Alternative: Centralized services with dedicated servers
  - Not desired
- Alternative: **Hybrid Peer-to-Peer**

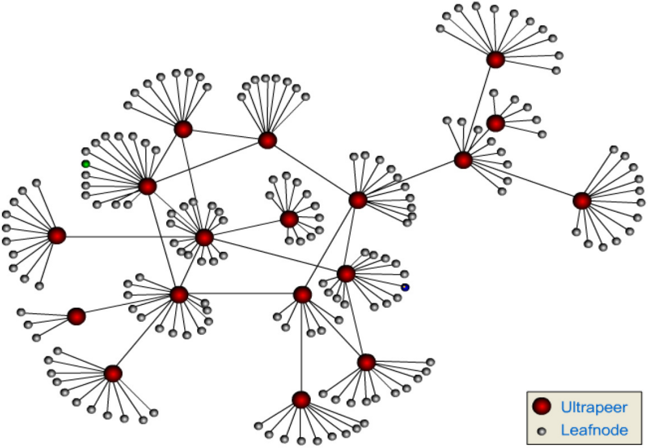
Hybrid P2P: Gnutella Protocol Version 0.6

- Distinction of the nodes into leaf nodes and Ultrapeers
  - Each leaf node is connected to a maximum of 3 Ultrapeers
  - Each Ultrapeer is connected to max. 16 Ultrapeers and 16 leaf nodes
    - Maximum number of Hops per search request: 4
- Gnutella v0.6 implements the Query Routing Protocol (QRP)
  - Each leaf node sends its Ultrapeers a list of the files' names it offers
  - Search requests are only forwarded to Peers, which offer files with matching file names
  - The Ultrapeers exchange the lists of files names between themselves
- Hybrid P2P implements a kind of dynamic, centralized service
  - Advantages of pure P2P remain

The FastTrack protocol of KaZaA is based on the Gnutella protocol v0.6

- Supernodes act as temporary index servers for slower Peers – this increases the scalability of the network
- The client software contains a list of IPs of some Supernodes
- Supernodes communicate with other Supernodes to satisfy queries of clients
- If a requested file is found, the client establishes a direct connection to the Peer, which offers the file, and the file is transferred via HTTP

Gnutella: Ultrapeers and Leaf Nodes



Source: Jörg Eberspächer and Rüdiger Schollmeier. *First and Second Generation of Peer-to-Peer Systems* (2005). LNCS 3485

Peer-to-Peer (File Sharing) Networks – Small Selection

Peer-to-Peer Network	Clients
Ares	Ares
Blubster	Blubster
FileSpree	FileSpree
Filetopia	Filetopia
Gnutella	AquaLime, BearShare, FileNavigator, FreeeWire, Gnucleus, LimeWire, Phex, Shareaza, Xolox Ultra
Direct Connect	Direct Connect, DC++. Koala DC
eDonkey2000	eDonkey2000, eMule
FastTrack	Morpheus, KaZaA, KaZaA Lite, Grokster, iMesh
OpenNap	Napster, Shuban, File Navigator, Rapigator, Spotlight, StaticNap, SunshineUN, Swaptor, WinMX OpenNap
Overnet	Overnet, eDonkey2000, MLDonkey
Piolet	Piolet
MinMX	MinMX
Freenet	Freenet, Entropy, Frost
Entropy	Entropy, Frost, Freenet
WASTE	WASTE

Source: `http://www.umkc.edu/is/security/p2p_explanation.asp`

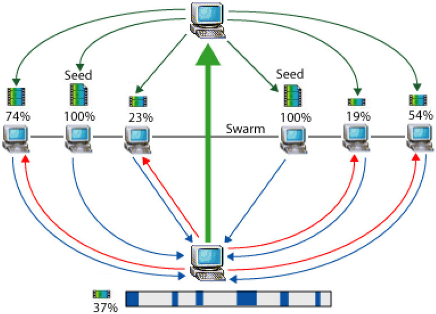
BitTorrent (1/3)

- System for rapid distribution of large amounts of data
- Developed by Bram Cohen
- July 2001: Initial release
- If a Peer wants to download files, the Peer needs the appropriate **Torrent file**
    - Contains the IP addresses of the **Trackers**, file size, and a list of checksums of segments
    - Torrent files have a size of just a few kB
    - Torrent files are often collected and offered on web pages (e.g. The Pirate Bay)
- A tracker manages for each Torrent a list of Peers, which provide parts of the file
    - A BitTorrent client receives this list from the Tracker
    - With this list, the client (Peer) can send download requests to other Peers, which provide (offer) parts of the requested file

BitTorrent (2/3)

- **Chunks**: Segments of a file
- **Swarm**: All Peers, which are interested in the same Torrent
- **Seeds**: Peers, which provide the complete Torrent
  - These Peers only provide Segments and do not download any more Segments of the file
- **Leechs**: Peers, which do not yet have the complete Torrent
  - These Peers download Segments from other Peers

BitTorrent tracker identifies the swarm and helps the client software trade pieces of the file you want with other computers.



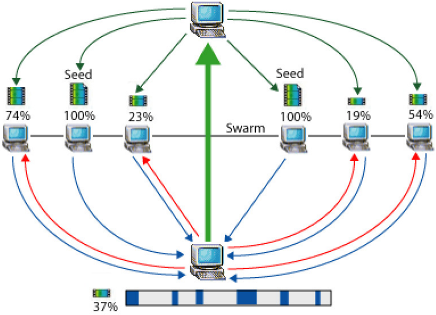
Computer with BitTorrent client software receives and sends multiple pieces of the file simultaneously.

©2005 HowStuffWorks

BitTorrent (3/3)

- If the Tracker is no longer available, the Torrent file becomes useless
- Without a working Tracker, the current list of Seeds and Leeches cannot be obtained
  - With this List, the Leeches can request missing chunks from other Seeds and Leeches
- If a participant wants to offer a file, he must run a Tracker and offer/spread the corresponding Torrent file

BitTorrent tracker identifies the swarm and helps the client software trade pieces of the file you want with other computers.



Computer with BitTorrent client software receives and sends multiple pieces of the file simultaneously.

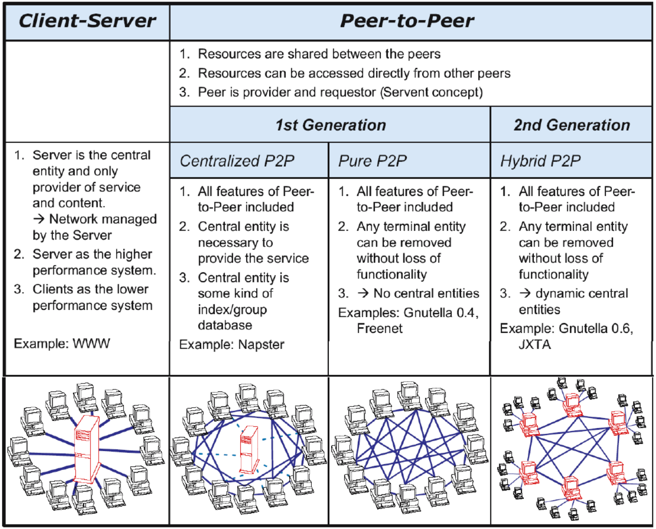
©2005 HowStuffWorks

BitTorrent in Cloud Infrastructures

- BitTorrent is a highly efficient solution for distributing large amounts of data
- Example, where large amounts of data must to be transmitted as quickly as possible:
  - Distribution of images to the physical nodes in cloud infrastructures
- In 2010, the CERN operated a virtual Cluster, based on OpenNebula
  - The Cluster was dynamically extended with up to 16,000 virtual machines
  - A maximum number of 400,000 computing jobs have been processed in a reasonable time by the LSF scheduler
  - Distribution of the images and data in the cluster was carried out best via BitTorrent

Source: Ignacio M. Llorente. CERN CLOUD SCALING TO 16,000 VMS! (November 2010)
`http://blog.opennebula.org/?p=983`

P2P Summary



Client-Server	***Peer-to-Peer***		
	1. Resources are shared between the peers 2. Resources can be accessed directly from other peers 3. Peer is provider and requestor (Servent concept)		
	1st Generation		***2nd Generation***
	Centralized P2P	*Pure P2P*	*Hybrid P2P*
1. Server is the central entity and only provider of service and content. → Network managed by the Server 2. Server as the higher performance system. 3. Clients as the lower performance system Example: WWW	1. All features of Peer-to-Peer included 2. Central entity is necessary to provide the service 3. Central entity is some kind of index/group database Example: Napster	1. All features of Peer-to-Peer included 2. Any terminal entity can be removed without loss of functionality 3. → No central entities Examples: Gnutella 0.4, Freenet	1. All features of Peer-to-Peer included 2. Any terminal entity can be removed without loss of functionality 3. → dynamic central entities Example: Gnutella 0.6, JXTA

Source: Jörg Eberspächer und Rüdiger Schollmeier. *First and Second Generation of Peer-to-Peer Systems* (2005). LNCS 3485

BitTorrent can be used without a dedicated Tracker

BitTorrent 4.1.0 beta, Trackerless support

May 19th 2005

BitTorrent Goes Trackerless: Publishing with BitTorrent gets easier!

As part of our ongoing efforts to make publishing files on the Web painless and disruptively cheap, BitTorrent has released a „trackerless“ version of BitTorrent in a new release.

. . .

Many of you have blogs and websites, but dont have the resources to set up a tracker. In the new version, we've created an optional „trackerless“ method of publication. Anyone with a website and an Internet connection can host a BitTorrent download!

While it is called trackerless, in practice it makes every client a lightweight tracker. A clever protocol, based on a Kademlia distributed hash table or „DHT“, allows clients to efficiently store and retrieve contact information for peers in a torrent.

. . .

Source: `http://forums.degreez.net/viewtopic.php?f=1&t=5277`

Distributed Hash Table (DHT)

- Data structure, which can be used to **store the locations** of files in a P2P system
  - Only references to the files are stored inside the DHT and not the files itself
  - Objective: Decentralized and efficient data storage and localization
- Each storage node corresponds to an entry in the hash table
- Positive characteristics of the DHT
  - Self-organization: No manual configuration is required
  - Scalability: The system can handle a large number of nodes
  - Load distribution: Uniform distribution of keys across the existing nodes
  - Fault tolerance: Even if nodes fail or leave the system

Distributed Hash Table (DHT)

- Known DHT implementations: **Chord** and **Kademlia**
- Popular software, which uses the DHT:
  - Modern BitTorrent clients (e.g. Azureus, KTorrent, rTorrent, μTorrent)
  - Freenet (anonymous data storage)
  - File Sharing Tools (LimeWire, MLDonkey)
  - GlusterFS (distributed file system)
  - Apache Cassandra (distributed database management system)
  - YaCy (distributed search engine)

Keys?

- The following slides, often contain the term *key*
- Keys are checksums (hash values) of files
- The hash values are generated via **hash functions** $\Longrightarrow$ see next slide

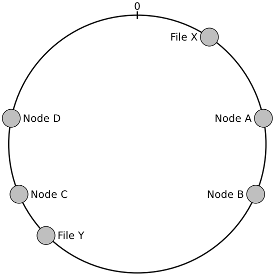
Hash Function

- Functions, which generate a short, fixed-length output data (the **hash code**, **hash value** or **checksum**) from a large input data
  - The name originates from the verb *to hash* = „hack into smaller pieces“
  - „Good“ hash functions generates for different input data different output data
    - The hash value of a file is similar to a fingerprint
  - Hash codes are used among others to detect transmission errors
  - In cryptography, hash functions are used for signing
- Popular (and „good“) hash algorithms are MD5 and SHA-1
  - These procedures have practically free of collision
    $\Longrightarrow$ The probability of duplicate records is minimal
  - MD5 generates 128-bit checksums
  - SHA-1 generates 160-bit checksums

```
$ md5sum slides_cloud_computing_lecture_10_WS1314_english.pdf
96900c79555c251533801f68a89a4014
$ sha1pass slides_cloud_computing_lecture_10_WS1314_english.pdf
$4$zLdj04up$rDEi2kYGXwqlLwaD5Hk1+32U9us$
```

Distributed Hash Table – Short Summary

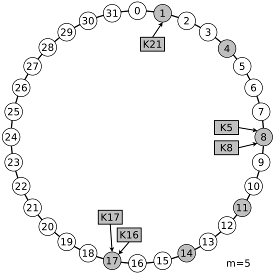
- With a hash function, keys are assigned to files in a linear range of values (key space)
  - The key space can be imagined as a ring
  - **The files are *hashed* into the key space**
- Objective: The key space should be distributed as uniformly as possible across the nodes



- Procedure: **The nodes are *hashed* into the key space of the same ring**
  - For this, a unique feature (e.g. the IP address) is required
- **The clockwise next node is responsible for a file**
  - If that node fails, the clockwise next available node becomes responsible

Distributed Hash Table – Functioning of Chord

- Protocol for the implementation of a distributed lookup service
  - A lookup service is a service to locate data
- For each **node** and every **key**, an **ID** is calculated with a hash function (e.g. SHA-1)
  - The ID is *m*-bits long (often 160 bits)
  - Maximum number of nodes in the system: 2^m



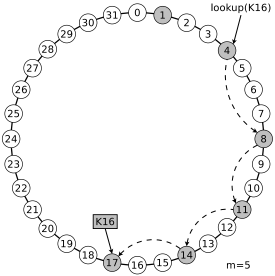
Key distribution in the Chord Ring

- The nodes are arranged in a ring
  $\Longrightarrow$ **Chord Ring**
- A key *k* is assigned to the node *n*, whose ID is $\geq$ the ID of the key *k*
  - Node *n* is called **successor** of *k*
- Chord provides the following function: *find for key k the responsible node n*
  - This should happen as efficiently as possible

Source: `http://sarwiki.informatik.hu-berlin.de/Chord`

Functioning of Chord – Linear Search

- Most simple form of looking for a node: **Linear search**
- Each node can identify its successor node
- If a node receives a search request, it checks, if it's successor node is responsible for the requested key
  - The search request is forwarded from one node to another, until the destination is reached



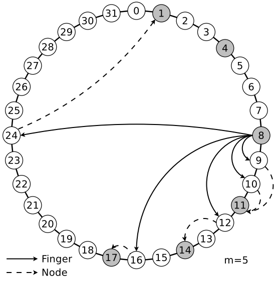
Linear search in the Chord Ring

- Drawback: Linear search is **not efficient**
  - The length of the search path grows linear with the number of nodes

Source: `http://sarwiki.informatik.hu-berlin.de/Chord`

Functioning of Chord – Binary Search

- Scalable form of looking for a node: **Binary search**
  - This requires the nodes to store more information about the ring
- Every node maintains its table with references to m nodes
  - m is the length of the IDs used in bits
  - References are called **Chord Fingers** and the table is called **Finger Table**



Fingers in the Chord Ring

Finger Table of node $n = 8$

Entry	Start	Node
1	9	11
2	10	11
3	12	14
4	16	17
5	24	1

The table has 5 entries, because m contains the length of the ID in bits and $m = 5$

- The **Start** value of entry i of the table on node n is $(n + 2^{i-1}) \bmod 2^m$
- The **Node** value of entry i points to the first node, which follows to n at a distance of at least 2^{i-1}

The **Node** value points to the clockwise next node after the **Start** value

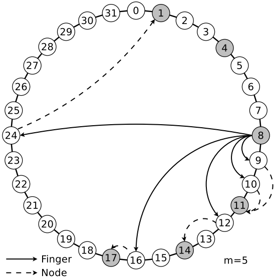
Functioning of Chord – Search Requests

- With each hop, the distance to the destination can be halved
- If a search request is sent to node n for key k, node n searches its finger table from **bottom to top** until the first entry i (**Node** value!) is found, whose node is **not** located between n and k
  - Entry i is the **closest predecessor node of key** k, which is known by node n
  - Entry $i + 1$ is the **closest successor node of key** k, which is known by node n
- The search request is forwarded to the closest predecessor node
- This is repeated until the node is identified, whose successor node is responsible for k

Source: `http://sarwiki.informatik.hu-berlin.de/Chord`

Functioning of Chord – Example

- Example: The node with ID 8 receives a search request for a key with ID 3



Finger Table of node $n = 8$

Entry	Start	Node
1	9	11
2	10	11
3	12	14
4	16	17
5	24	1

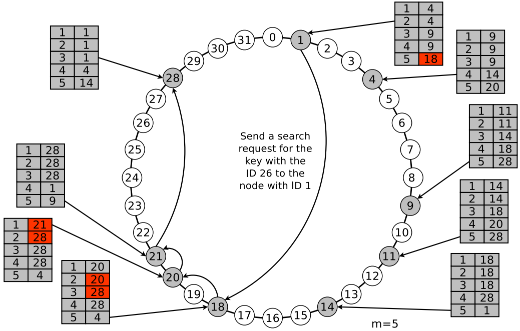
The table has 5 entries, because m contains the length of the ID in bits and $m = 5$

- Node 8 searches in its Finger Table for the closest located predecessor node of key 3 (node 1) and sends the request to it
    - Note: It is impossible to search backwards in the Ring!
- Node 1 detects that its successor node (node 4) is responsible for key 3
- Node 1 sends the address of node 4 to node 8

Source: http://sarwiki.informatik.hu-berlin.de/Chord

Functioning of Chord – Example from Tanenbaum (1/2)

- Example: The node with ID 1 receives a search request for the key with ID 26
- Node 1 checks its Finger Table for key 26
- 26 is greater than the value of the last entry (18) in the Finger Table
- The search request is forwarded to node 18


0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
Send a search request for the key with the ID 26 to the node with ID 1
m=5

Node 1:

1	4
2	4
3	9
4	9
5	18

Node 4:

1	9
2	9
3	9
4	14
5	20

Node 9:

1	11
2	11
3	14
4	18
5	28

Node 11:

1	14
2	14
3	18
4	20
5	28

Node 14:

1	18
2	18
3	18
4	28
5	1

Node 18:

1	20
2	20
3	28
4	28
5	4

Node 20:

1	21
2	28
3	28
4	28
5	4

Node 21:

1	28
2	28
3	28
4	1
5	9

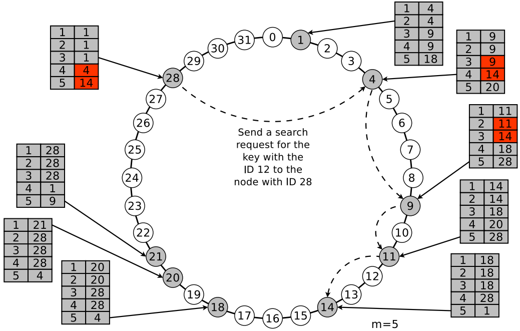
Node 28:

1	1
2	1
3	1
4	4
5	14

- Node 18 selects node 20, because node 20 $<$ key 26 $\leq$ node 28
- Node 20 selects node 21, because node 21 $<$ key 26 $\leq$ node 28
- Node 21 selects node 28, because this node is responsible for key 26
  - Reason: Node 28 is the direct successor of node 21 and a key k is assigned to the node n, whose ID is $\geq$ the ID of the key k
- Node 21 sends the address of node 28 to node 1

Functioning of Chord – Example from Tanenbaum (2/2)

- Example: The node with ID 28 receives a search request for the key with ID 12
- Node 28 checks its Finger Table for key 12


0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
Send a search request for the key with the ID 12 to the node with ID 28
m=5

1	1
2	1
3	1
4	4
5	14

1	4
2	4
3	9
4	9
5	18

1	9
2	9
3	9
4	14
5	20

1	11
2	11
3	14
4	18
5	28

1	28
2	28
3	28
4	1
5	9

1	14
2	14
3	18
4	20
5	28

1	21
2	28
3	28
4	28
5	4

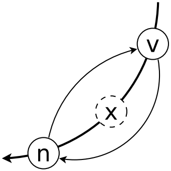
1	20
2	20
3	28
4	28
5	4

1	18
2	18
3	18
4	28
5	1

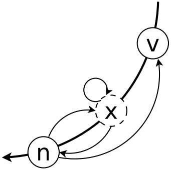
- Node 28 selects node 4, because node 4 $<$ key 12 $\leq$ node 14
- Node 4 selects node 9, because node 9 $<$ key 12 $\leq$ node 14
- Node 9 selects node 11, because node 11 $<$ key 12 $\leq$ node 14
- Node 11 selects node 14, because this node is responsible for key 12
  - Reason: Node 14 is the direct successor of node 11 and a key k is assigned to the node n, whose ID is $\geq$ the ID of the key k
- Node 11 sends the address of node 14 to node 28

Functioning of Chord – Insert new Nodes (1/2)

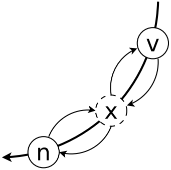
- If a node wants to enter the ring, it needs an entry point
  - Usually, *well-known nodes* are used
- The ID of the new node is calculated
  - For this, the hash function usually uses the IP address or MAC address as input data
- Next, the new node (x) sends to the well-known node a search query for the key with the calculated ID
  - This way, the new node finds its successors (n) node



- The new node sets its successor pointer
  - Next, it informs its successor node (n) about its existence
- The successor node (n) checks, if the ID of the new node is greater than the ID of the old predecessor (v) node and then adjusts its predecessor pointer



Functioning of Chord – Insert new Nodes (2/2)



- During the periodic execution of the stabilization protocol, the old predecessor node (v) detects, that the new node (x) is its new successor node and adjusts its successor pointer
  - Next, node (v) informs the new node (x), that it is its new successor node
- Next, the new node (x) adjusts its predecessor pointer
  - Now, the ring is again in stable state
- The new node still needs to fill its Finger Table
  - To do this, the new node sends a search request for each table entry
  - A more easy way is to copy the Finger Table of the direct successor node
    - Its Finger Table does not differ much from that of the new node
    - The stabilization protocol will update the Finger Table entries
    - The correctness of the Finger Table is only essential for the duration of search requests, but not for the correctness of search results

Bitcoins – P2P-based Money

- Bitcoin is **electronic money**, which is generated in a decentralized way
- The maximum amount of Bitcoins has a fixed upper limit
  - The last possible Bitcoins **block** (No. 6,929,999), will be generated around the year 2140
    - Then the total number of Bitcoins will stop at approximately 21 million
  - Currently (November 2013), approx. 11.9 million Bitcoins available
    - Source: `http://blockexplorer.com/q/totalbc`
  - The growth is halved every 4 years
- Each Bitcoin can be subdivided to eight decimal digits
  - Thus, the smallest possible value is 0.00000001 BTC
- The number of existing Bitcoins results from the number of **blocks**, multiplied by the coin value of a block
  - The coin value for each one of the first 210,000 blocks is 50 BTC
  - The coin value for each one of the next 210,000 blocks is 25 BTC
  - The coin value for each one of the next 210,000 blocks is 12.5 BTC, ...
- As a result, the increase of Bitcoins slows down

Bitcoins – Wallet File

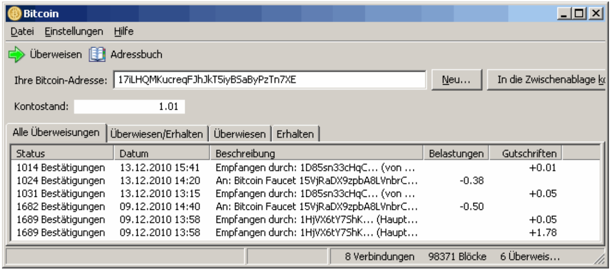
- Every user of the Bitcoin software requires a **Wallet file** `/home/<username>/.bitcoin/wallet.dat`
  - This file contains a public key and a private key
    - The public key is used as **Bitcoin address** to send and receive Bitcoins
    - The private key is used to authorize money transactions
  - Bitcoin address contain no information about its owner
  - For better exchange, a human-readable form of 33 characters exist
  - Creating a new key pair is equal to creating a new account
    - Thus, each user can have a theoretically unlimited number of accounts
- One major advantage of Bitcoin: No **transaction fee** is charged
  - Theoretically, a transfer of 0.00000001 BTC is possible
    - For the sake of simplicity, the the smallest accepted unit is the second digit after the decimal point

Well written article (in German language)

Florian Hofmann, Jörn Loviscach. **Virtuelles Vermögen**. c't 17/2011. P.74-79

Bitcoins – Anonymity of Transactions (Transfers)

- Payments are partially anonymous
  - A payment address can not identify the receiver
  - The anonymity is similar with cash money
- If a money transfer was executed, the Bitcoin client automatically creates a new Bitcoin address
  - Users should use for each money transaction a different address, to make it impossible to track money movements



Ihre Bitcoin-Adresse: 17iLHQMKucreqFJhJkT5iyBSaByPzTn7XE

Kontostand: 1.01

Status	Datum	Beschreibung	Belastungen	Gutschriften
1014 Bestätigungen	13.12.2010 15:41	Empfangen durch: 1D85sn33cHqC... (von ...		+0.01
1024 Bestätigungen	13.12.2010 14:20	An: Bitcoin Faucet 15VjRaDX9zpbA8LVnbrC...	-0.38	
1031 Bestätigungen	13.12.2010 13:15	Empfangen durch: 1D85sn33cHqC... (von ...		+0.05
1682 Bestätigungen	09.12.2010 14:40	An: Bitcoin Faucet 15VjRaDX9zpbA8LVnbrC...	-0.50	
1689 Bestätigungen	09.12.2010 13:58	Empfangen durch: 1HjVX6tY7ShK... (Haupt...		+0.05
1689 Bestätigungen	09.12.2010 13:58	Empfangen durch: 1HjVX6tY7ShK... (Haupt...		+1.78

8 Verbindungen 98371 Blöcke 6 Überweis...

Image source: http://kalingeling.wordpress.com/2010/12/19/bitcoin-tutorial/

The status display provides information on how many Peers the client is connected, how many blocks have been processed, and how many money transactions have been made or received

Bitcoins – Transactions (Transfers)

- Money transactions are validated by the nodes
- All transactions are stored in blocks and they are constantly checked and confirmed blockwise by all connected clients
- Each client has access to the current block and all generated blocks
    - During startup, the client fetches all blocks, which have been created in the meantime
- Payments may be carried out, although a client is offline
    - Transmitted transactions are written into the block, which is currently processed by the network and checked by all clients
        - This prevents double payments
    - If the receiver starts its client, it fetches all blocks from the network
        - If one of the blocks contains an address, which is assigned to the user, and the associated transaction has already been confirmed by several other clients, the client adds the amount to the users account
        - The account balance is increased, and the transaction gets listed in the list of transfers (`http://blockexplorer.com`)
        - Transaction not yet confirmed ⟹ the amount is listed under reservation

Create/Compute Bitcoins

- A problem of money in general is the initial distribution of assets
  - As a new currency, Bitcoin enjoys initially no trust
  - Users can buy Bitcoins e.g. via Paypal, but who wants that?
    - Changing back to established money is not guaranteed by any authority
- Users can **create/compute blocks, which contain Bitcoins**
  - Components of a block are:
    - The checksum of the previous block
    - All completed transactions in a certain period of time
    - A security number, which is only used once
  - In order to finish (to solve!) a block, a Bitcoin client needs to guess (brute force!) a system-defined checksum
    - Next, the block is marked as solved and a new block is started
    - The client gets as reward the value of the block credited
- It takes approx. 1 year to calculate a block with a CPU
  - A modern GPU is $>$ 100 times faster
- One option is pool mining
  - Multiple clients work together and share the credit

What can be done with Bitcoins

- Several service providers change Bitcoins to € or $
  - The exchange rate is determined by the individual service providers
  - The market regulates itself
- List of companies and service providers which accept Bitcoins: `https://en.bitcoin.it/wiki/Trade`
- In Berlin, several pubs and hotels accept Bitcoins

Satoshi Nakamoto. **Bitcoin: A Peer-to-Peer Electronic Cash System**. 2008
`http://www.bitcoin.org/bitcoin.pdf`

Risks/Problems with Bitcoins (1/2)

- June 2011: Symantec discovers a **Bitcoins trojan**
  - `Infostealer.Coinbit` searches for the wallet files to steal them
  - The trojan sends the files via email to a server in Poland
  - There, the crackers download the files to use them for themselves
  - Solution: Encrypting the wallet file(s)
- June 2011: 25,000 BTC (~ €500,000) are **stolen** from a user
- June 2011: 60,000 login credentials and email addresses from the Bitcoin exchange market Mt. Gox are **stolen and published**
  - Result: High exchange rate losses
    - Temporarily the value of a 1 BTC dropped from $17.5 to a few cents
  - Several exchange markets temporarily discontinued the Bitcoin trading
- March 2012: Several credentials of Linode users, who stored their Bitcoin wallet files at the cloud service, are hacked
  - Most of the damage has the Bitcoin exchange market Bitcoinica
    - It gets > 40,000 BTC (~ €140,000) **stolen**

Risks/Problems with Bitcoins (2/2)

- April 2012: 923 BTC (∼ €108,000) are **stolen** from the Bitcoin mining pool Ozcoin
- April 2012: Several hundred BTC are **stolen** from the Bitcoin exchange market Bitcoin-Central
- May 2012: 18,547 BTC (∼ €70,000) are **stolen** during a hacker attack on the Bitcoin exchange market Bitcoinica
- September 2012: 24,000 BTC (∼ €207,000) are **stolen** from the Bitcoin exchange market Bitfloor
    - Reason: The attackers had access to unencrypted backups of wallet files
- January 2013: BTC assets of undisclosed quantity are **stolen** from the Bitcoin exchange market Vircure because of a vulnerability of the free web framework Ruby on Rails

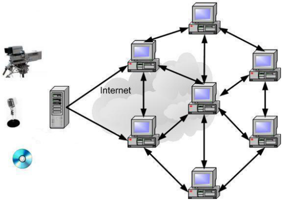
Transferring Television over the Internet

- Distributing television signals over the Internet is cost intensive
- Ideal: Distribution via broadcast to all participants
  - There, the number of audience is independent from the bandwidth, which is blocked by a channel
  - This requires a multicast-enabled network infrastructure on the complete path from the provider to the receiver
    - This is the internet nearly impossible
  - Multicast exists so far only in enclosed IPTV systems (e.g. T-Home)
- Reality in the Internet: Providers must provide an individual stream (Unicast) to each user
- Interesting approach: Transmission of television signals via P2P

Television via P2P

Image Source: Luis Eduardo Melendez Campis

- During P2P streaming, each user not only downloads the data stream, but also forwards it to other users
  - Each user contributes to the downstream bandwidth
- Advantages: Scalability
  - The more Peers contribute to the distribution, the better are the performance and reliability



- P2PTV systems are practically **real-time versions of BitTorrent**
  - If a user wants to watch a specific channel, the client requests from the index server (Tracker) the addresses of the Peers, which spread the channel
  - Next, the client establishes connections to these Peers to initiate the streaming
  - The tracker stores the address of the new Peer as a potential source for additional users, who want to watch the channel

Television via P2P in Reality – Zattoo

http://zattoo.com

- Zattoo (Japanese for *crowd*, *crush*, *swarm*)
    - Legal P2P streaming
    - Client software for Linux, MacOS X and Windows
- During connection establishment, the service checks in which country the user resides
    - Because of the broadcast rights, it must be ensured, that TV programs can be received only in the permitted territory
- Financing through advertising, which is displayed during the channel switching delay

- Switching channels takes about 5 sec.
- Time delay: Approximately 20 sec.
- Summer 2009: Switch to a web client
    - Zattoo is no longer based on P2P, but uses Flash streams
      ⟹ Improved quality

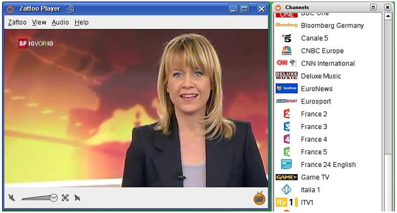


Image source: http://freie-software.blogspot.com

Television via P2P in Reality – Joost

http://www.joost.com

- Established in 2006 by Niklas Zennstrom and Janus Friis
  - Both are the founders of Skype and Kazaa
- 2007: $45 million venture capital
  - Legal P2P streaming
    - Offer: Streaming of channels and individual movies/series
    - Problem: Broadcast rights have to be renegotiated regularly for all countries

- Client software for MacOS X and Windows
- Financing through advertising
- December 2008: Switch to a purely web-page-based service with Flash
- Inactive since April 30th 2012

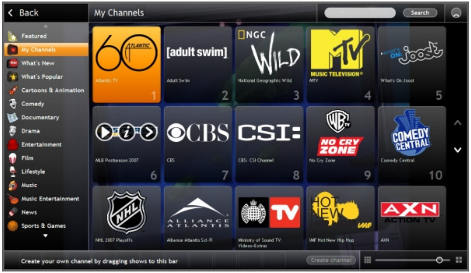


Image source: http://www.gizmodo.com.au

More once popular Projects, which have failed

- Babelgum
  - `http://www.babelgum.com`
  - P2P client from 2007 to March 2009
  - Concept similar to Joost
  - Problem: Insufficient high quality content

Image source: `http://www.nettv.cc`

- BBC iPlayer
  - `http://www.bbc.co.uk/iplayer/`
  - 2003/4: The BBC wanted tho make their archives freely accessible with a P2P-based client
  - Long development time
    - The service was available not until October 2005
  - December 2008: Switch from P2P model to HTTP downloads

Existing Projects

- Miro Media Player
  - `http://www.getmiro.com`
  - Previously *Democracy Player*
  - Free P2P client (free software)
  - Users can automatically download videos from subscribed channels
- PPLive, QQlive and UUSee
  - Proprietary P2P systems from China
    - `http://www.pplive.com`
    - `http://live.qq.com`
    - `http://www.uusee.com`
  - All these services focus on the Asian market
  - German and English versions of PPLive exist
    - `http://www.chip.de/downloads/PPLive_17375588.html`
- Octoshape
  - `http://www.octoshape.com`
  - Proprietary P2P client
  - e.g. DW-TV (Deutsche Welle) uses Octoshape

Reasons for Failure of P2P Streaming Projects (1/2)

- In contrast to platforms like YouTube, users cannot upload own videos
- Providers offer only professional content
  - The content must be licensed for each country, where it can be received
  - Without a large number of users, the providers do not have a strong negotiating position with the rights holders
  - Without a large number of users, the providers cannot finance the service via advertising
- Since the existence of YouTube, it is hard to explain to users why they need to install a client software only for watching video streams
- A dedicated client must be developed and maintained for many operating systems and hardware platforms
  - The development effort is high

Reasons for Failure of P2P Streaming Projects (2/2)

- The services were mostly proprietary
  - An open protocol can be optimized better, but is difficult to realize because of the place because of the required broadcast rights
- Time delays during connection establishment and channel switching are considered as annoying by the users
  - These time delays cannot be avoided in principle
- The upload data rate of most users is much lower compared with the download data rate
  - Especially with DSL, the upload speed is often lower by a factor of 1:10 or even 1:20
  - This limits the possibility to re-distribute by the Peers
  - Workaround: High performance CDN to support the distribution
    - CDN = Content Distribution Network
    - The nodes of the CDN act as Supernodes

P2P way to store distributed Data in Clouds

- The P2P paradigm can help to build up virtual clusters with common data in EC2-compatible infrastructures
- Different ways of cloud-based data storage
  - Instance storage (block-based, non-persistent)
  - EBS volumes (block-based, persistent)
  - S3 buckets (object-based, persistent)
  - Also database services exist
    - These services are not important here. They are primarily used for special applications, because they operate table oriented
    - Block-based and object-based storage services can be used for general purposes
- An EBS volume can only be attached to a single EC2 instance per time
- EBS volumes can be combined with distributed file systems, such as Ceph, GlusterFS, HDFS or PVFS2
  - These file systems operate completely distributed ($\Longrightarrow$ **P2P**)
  - Ceph, GlusterFS and HDFS even provide internal redundancy