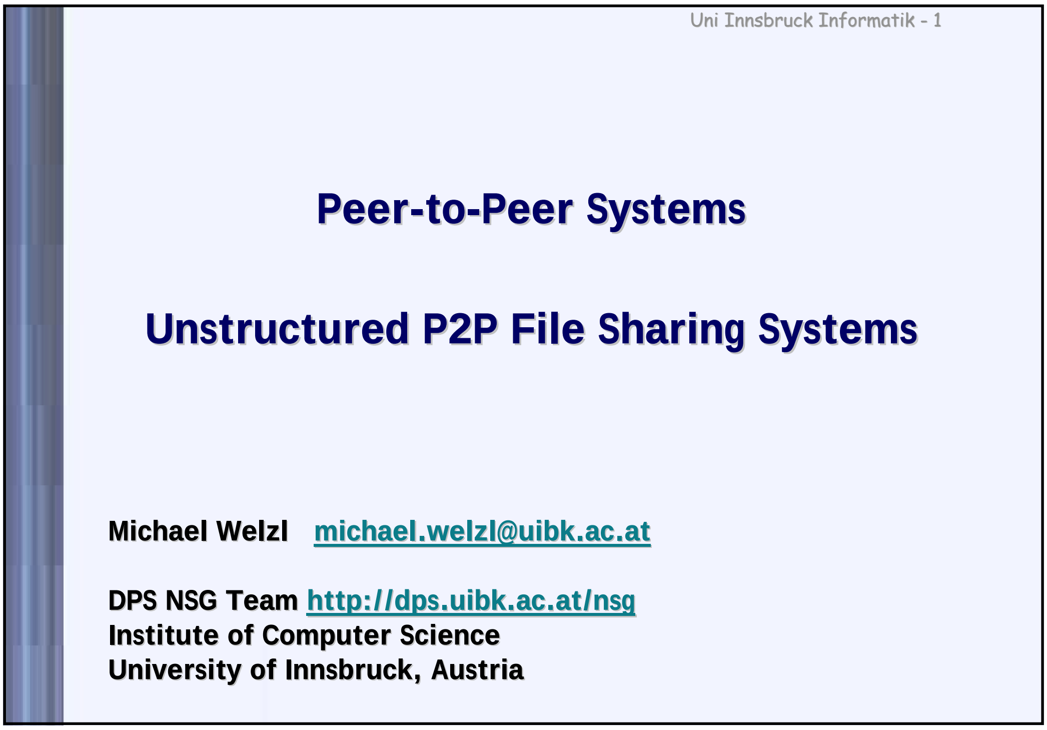
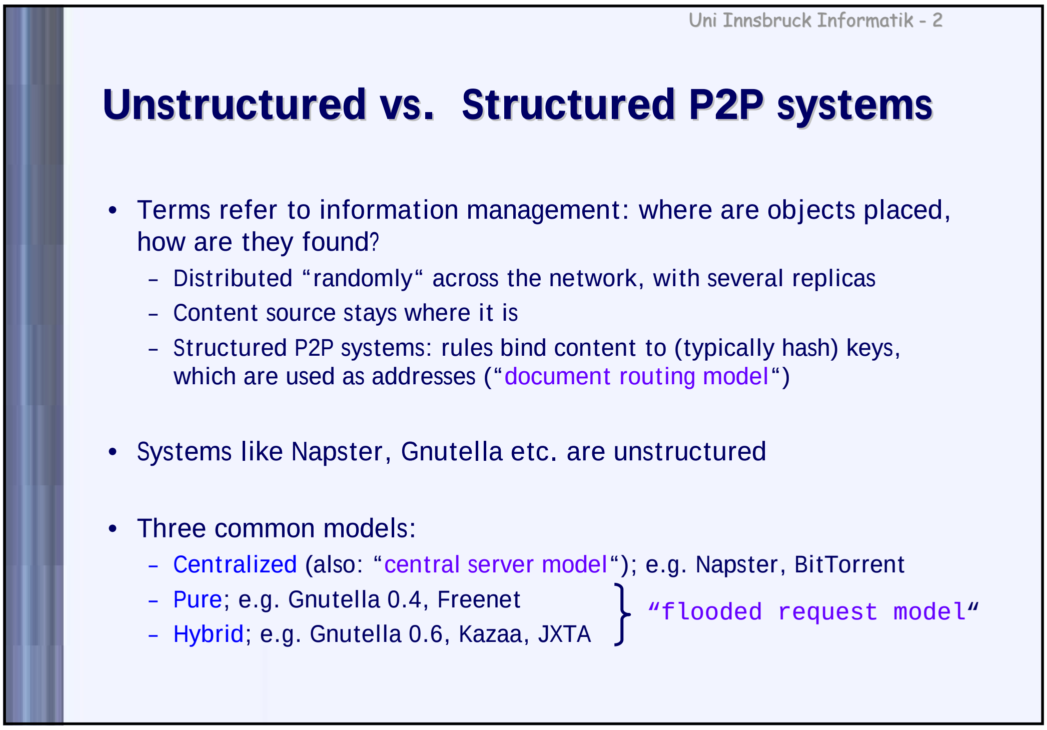


Peer-to-Peer Systems

Unstructured P2P File Sharing Systems

Michael Welzl michael.welzl@uibk.ac.at

DPS NSG Team http://dps.uibk.ac.at/nsg
Institute of Computer Science
University of Innsbruck, Austria

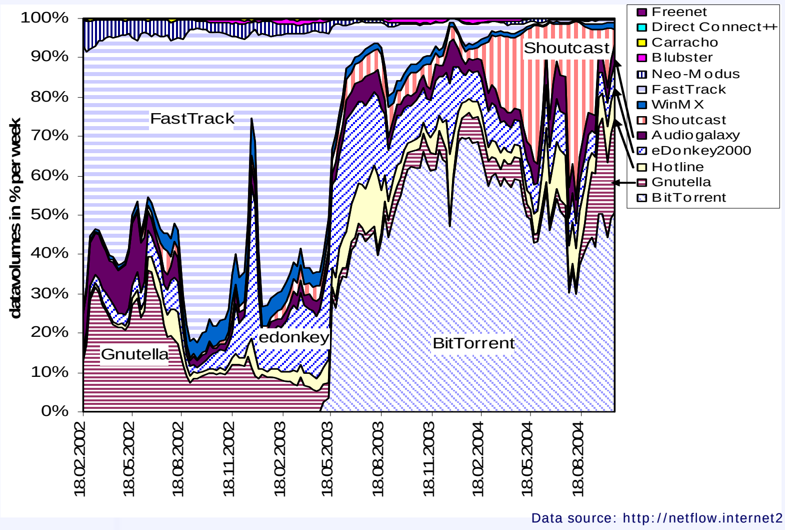


Unstructured vs. Structured P2P systems

- Terms refer to information management: where are objects placed, how are they found?
  - Distributed "randomly" across the network, with several replicas
  - Content source stays where it is
  - Structured P2P systems: rules bind content to (typically hash) keys, which are used as addresses ("document routing model")
- Systems like Napster, Gnutella etc. are unstructured
- Three common models:
  - Centralized (also: "central server model"); e.g. Napster, BitTorrent
  - Pure; e.g. Gnutella 0.4, Freenet
  - Hybrid; e.g. Gnutella 0.6, Kazaa, JXTA

} "flooded request model"

Development of P2P Applications

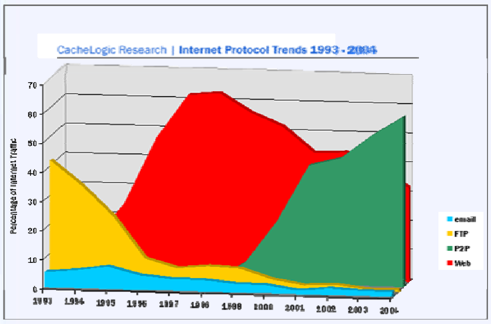


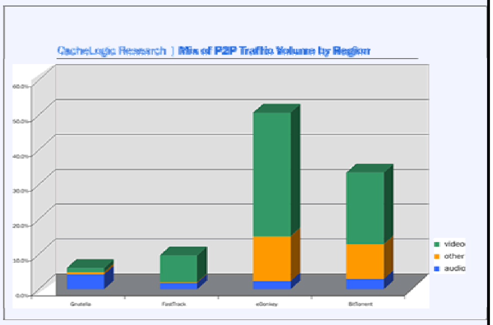
Data source: http://netflow.internet2.edu/weekly/

Traffic portions of the different P2P applications and protocols from the traffic measured per week in the Abilene backbone from 18.02.2002 until 18.010.2004

Development of P2P applications /2

- Main protocols in 2004: eDonkey, BitTorrent, FastTrack, Gnutella





- Source: http://www.cachelogic.com/research/2005_slide07.php# (2004)

Current P2P Content Distribution Systems

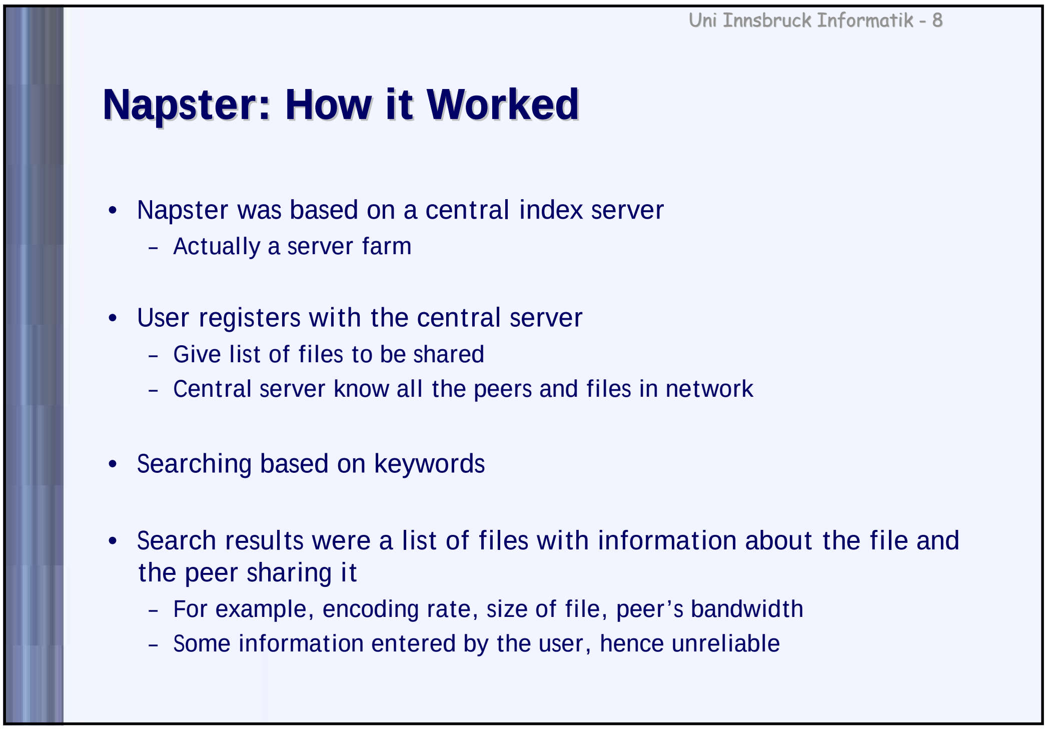
- Most current P2P content distribution systems targeted at one application: File sharing
- Users share files and others can download them
- Content typically music, videos, or software
  - Also often illegally shared
- Content distribution has made P2P popular
- Note: Distinguish between name of network (e.g., Gnutella) and name of client (e.g., LimeWire)

Napster

- Napster was the first P2P file sharing application
  - created by Shawn Fanning
  - “Napster” was Shawn’s nickname
- Only sharing of MP3 files was possible
- Napster made the term “peer-to-peer” known
- Do not confuse original Napster and current Napster
  - Latter is an online music store, nothing to do with P2P
  - Uses Napster name mainly to attract people

History of Napster

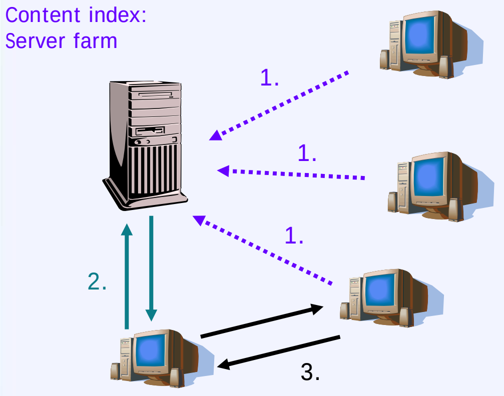
- Napster started in 1999
  - Beta released in June, “Download of the year” in fall ‘99
  - First lawsuit: December 1999 from several major recording companies
- Napster grew in popularity
  - Peaked at 13.6 million users in February 2001
- July 2001 judge ordered Napster to shut down
  - Case partially settled on September 24, 2001
  - Napster paid $26 million for past and future damages
- Bertelsmann AG bought Napster on May 17, 2002
  - Napster filed bankruptcy protection
  - On September 3, 2002, Napster forced to liquidate
- On October 29, 2003 Napster came back as an online music store



Napster: How it Worked

- Napster was based on a central index server
  - Actually a server farm
- User registers with the central server
  - Give list of files to be shared
  - Central server know all the peers and files in network
- Searching based on keywords
- Search results were a list of files with information about the file and the peer sharing it
  - For example, encoding rate, size of file, peer’s bandwidth
  - Some information entered by the user, hence unreliable

Napster: Queries

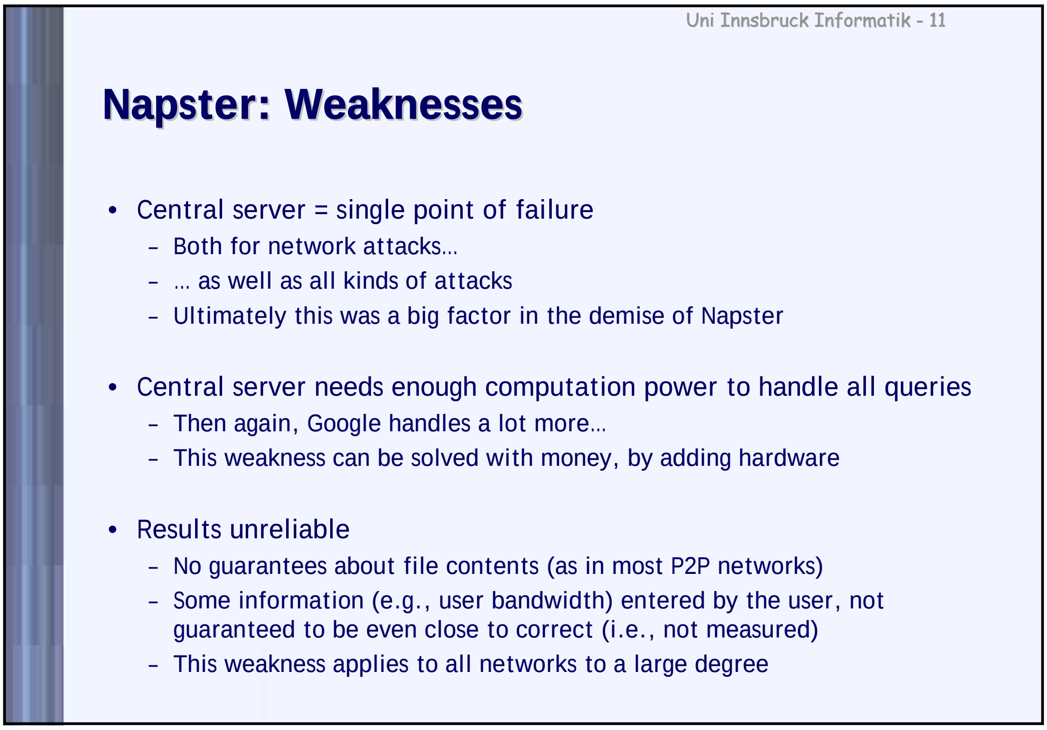


1. Peers register with central server, give list of files to be shared
2. Peers send queries to central server which has content index of all files
3. File transfers happen directly between peers

Last point is common to all P2P networks and is their main strength as it allows them to scale well

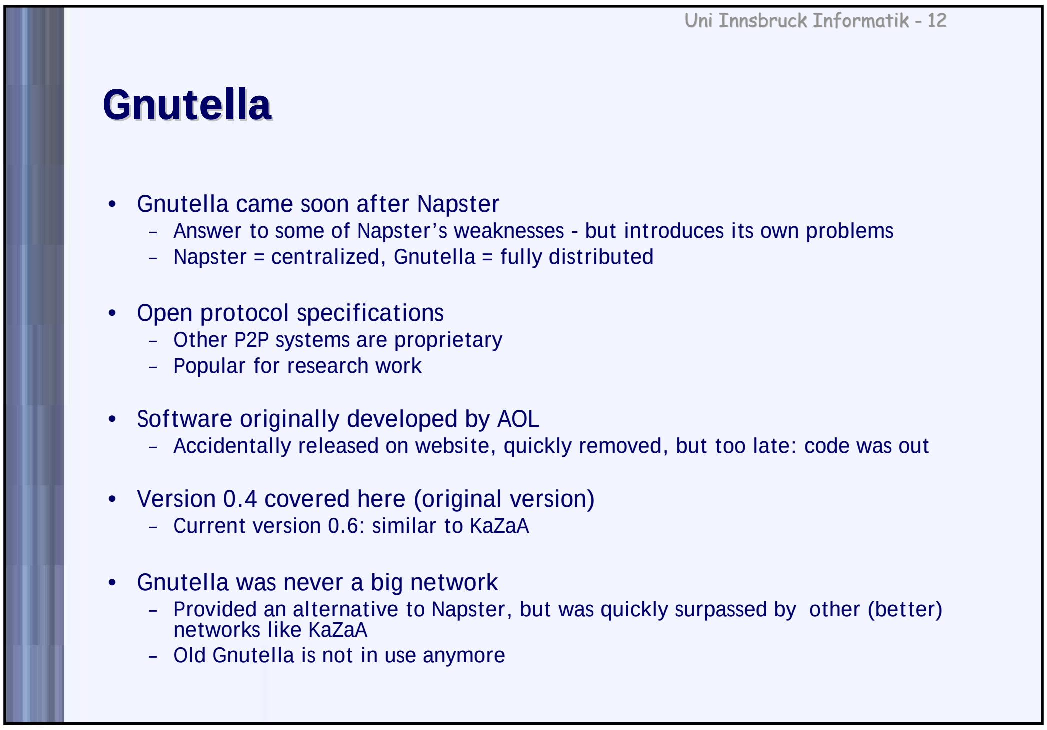
Napster: Strengths

- Consistent view of the network
  - Central server always knows who is there and who is not
- Fast and efficient searching
  - Central server always knows all available files
  - Efficient searching on the central server
  - Complicated queries possible
- Answer guaranteed to be correct
  - “Nothing found” means none of the currently on-line peers in the network has the file
  - Not true for all P2P systems



Napster: Weaknesses

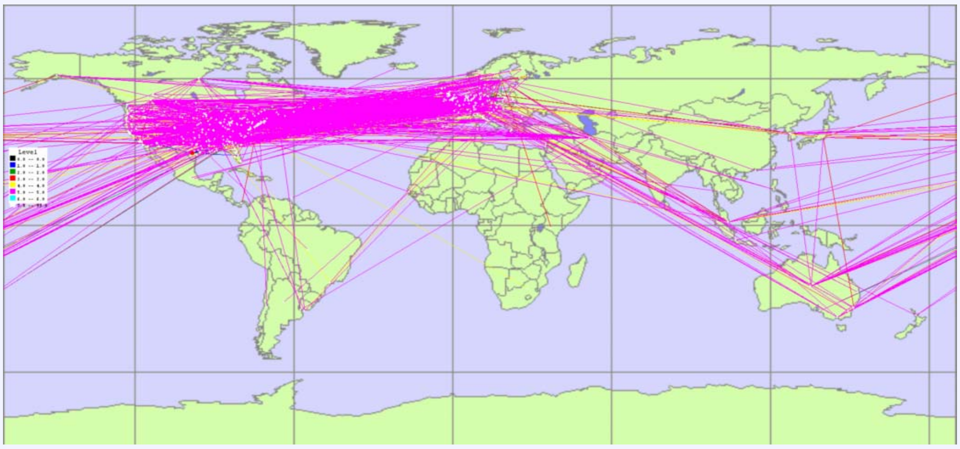
- Central server = single point of failure
  - Both for network attacks…
  - … as well as all kinds of attacks
  - Ultimately this was a big factor in the demise of Napster
- Central server needs enough computation power to handle all queries
  - Then again, Google handles a lot more…
  - This weakness can be solved with money, by adding hardware
- Results unreliable
  - No guarantees about file contents (as in most P2P networks)
  - Some information (e.g., user bandwidth) entered by the user, not guaranteed to be even close to correct (i.e., not measured)
  - This weakness applies to all networks to a large degree



Gnutella

- Gnutella came soon after Napster
  - Answer to some of Napster's weaknesses - but introduces its own problems
  - Napster = centralized, Gnutella = fully distributed
- Open protocol specifications
  - Other P2P systems are proprietary
  - Popular for research work
- Software originally developed by AOL
  - Accidentally released on website, quickly removed, but too late: code was out
- Version 0.4 covered here (original version)
  - Current version 0.6: similar to KaZaA
- Gnutella was never a big network
  - Provided an alternative to Napster, but was quickly surpassed by other (better) networks like KaZaA
  - Old Gnutella is not in use anymore

The Gnutella Network

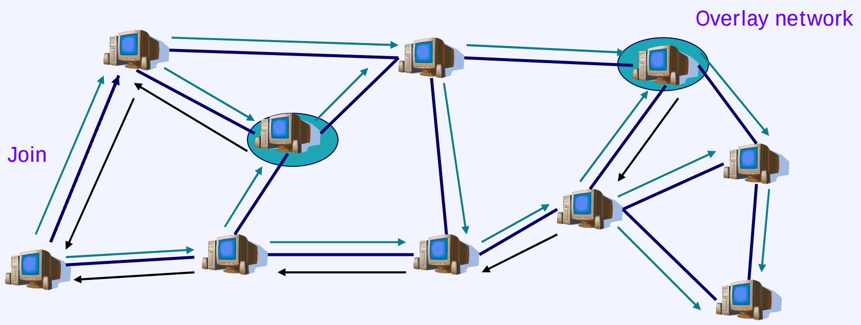


Measurements taken at the
LKN in May 2002

Gnutella: how it works

- Gnutella network has only peers
  - All peers are fully equal
  - Peers connected via TCP form overlay network
  - Peers called servents (**serv**er + cli**ent**)
- To join the network, peer needs the address of another peer that is currently a member
  - Bootstrapping needed: out-of-band channel, e.g., get it from a website
  - Often: cached peer lists
- Once a peer joins the network, it learns about other peers and learns about the topology of the network
- Queries are flooded across network
- Downloads directly between peers

Gnutella in a nutshell



- To join, peer needs address of one member, learn others
- Queries are sent to neighbors
- Neighbors forward queries to their neighbors (flooding)
  - Yes, via TCP!
- Replies routed back via query path to querying peer

Gnutella: Joining the Network

- A peer who wants to join the Gnutella network, needs the address of one peer who is already a member
- New peer sends connect message to existing peer
  - GNUTELLA CONNECT
- Reply is simply "OK"
  - No state involved at this point
- The point of this message is not very clear…
  - Receiving peer can deny the join (denial of service)
  - Several other protocol oddities in Gnutella 0.4

Gnutella: PING/PONG

- A peer discovers other peers in Gnutella with the PING and PONG messages

- PING
  - Used to actively discover hosts on the network

- PONG
  - The response to a Ping
  - Includes the address of a connected Gnutella servent and information regarding the amount of data it is making available to the network

- PONGs sent back along the path that PING took

Gnutella: QUERY/QUERYHIT

- For finding content

- QUERY
  - A servent receiving a Query descriptor will respond with a QueryHit if a match is found against its local data set

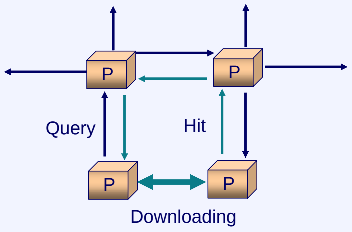
- QUERYHIT
  - The response to a Query
  - Provides recipient with enough information to acquire the data matching the corresponding Query

- Servents receiving QUERY messages forward them to their neighbors (unless TTL expired)

- Replies returned along the same path

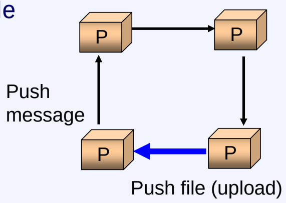
Gnutella: Download

- Peer sends QUERY to find files it wants
  - If peer with matching content reached, the querying peer will receive QUERYHIT
  - QUERYHIT contains IP address and port number of peer who sent it
- Contact that peer directly
- Download via HTTP GET request
  - Use given port number, but HTTP syntax
- Donwload directly between peers
  - Overlay network not involved in downloads
- May not work if contacted peer is behind firewall

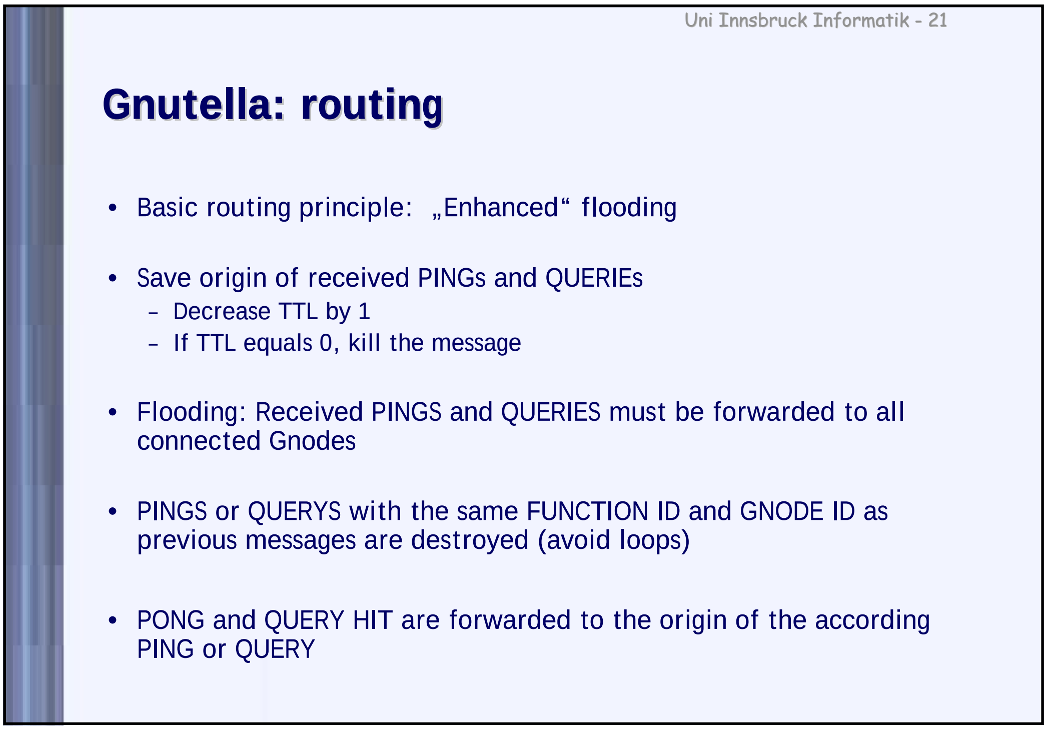


Gnutella: PUSH

- PUSH message
  - Peer outside firewall sends PUSH to peer inside firewall
    - Assumption: Peer inside firewall keeps TCP connection open to some neighboring peers in the overlay network
  - Peer inside firewall initiates upload of requested file
  - Does not work if both peers are behind a firewall

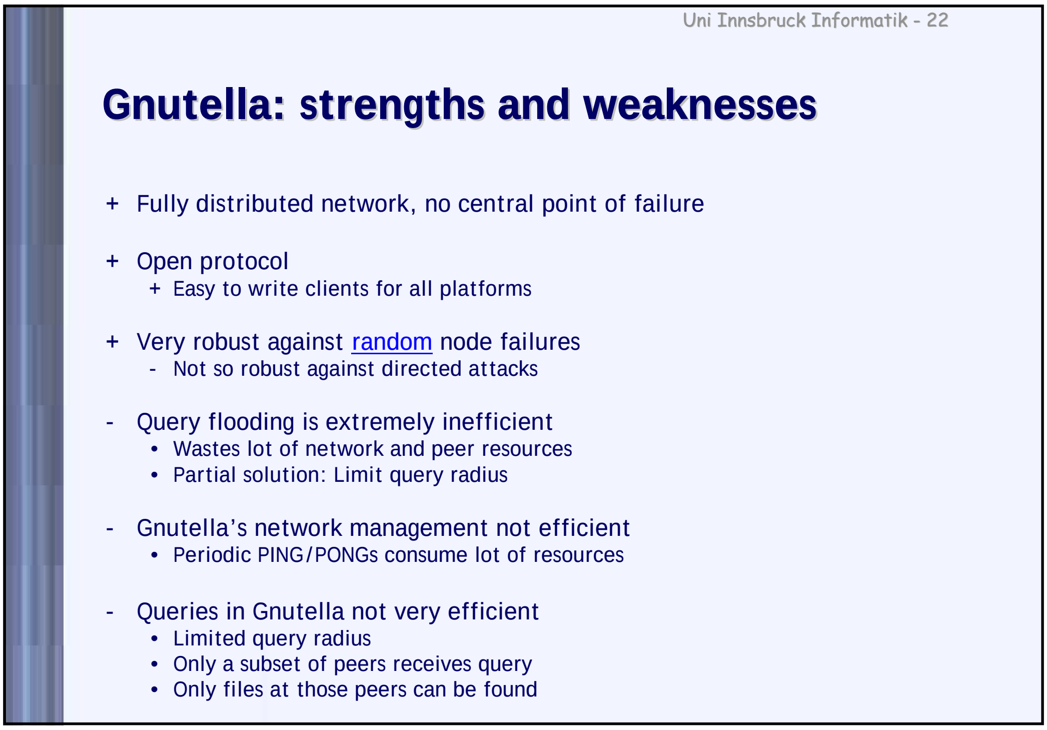


- Problem: exploitation for DDoS attack
  - *Spoof* of source address by using address of site to be attacked
  - Send such messages to many peers (reflectors)
  - Reflectors try to set up data connections to point of attack
  - Traceback very difficult in Gnutella network



Gnutella: routing

- Basic routing principle: „Enhanced“ flooding
- Save origin of received PINGs and QUERIEs
  - Decrease TTL by 1
  - If TTL equals 0, kill the message
- Flooding: Received PINGS and QUERIES must be forwarded to all connected Gnodes
- PINGS or QUERYS with the same FUNCTION ID and GNODE ID as previous messages are destroyed (avoid loops)
- PONG and QUERY HIT are forwarded to the origin of the according PING or QUERY



Gnutella: strengths and weaknesses

+ Fully distributed network, no central point of failure

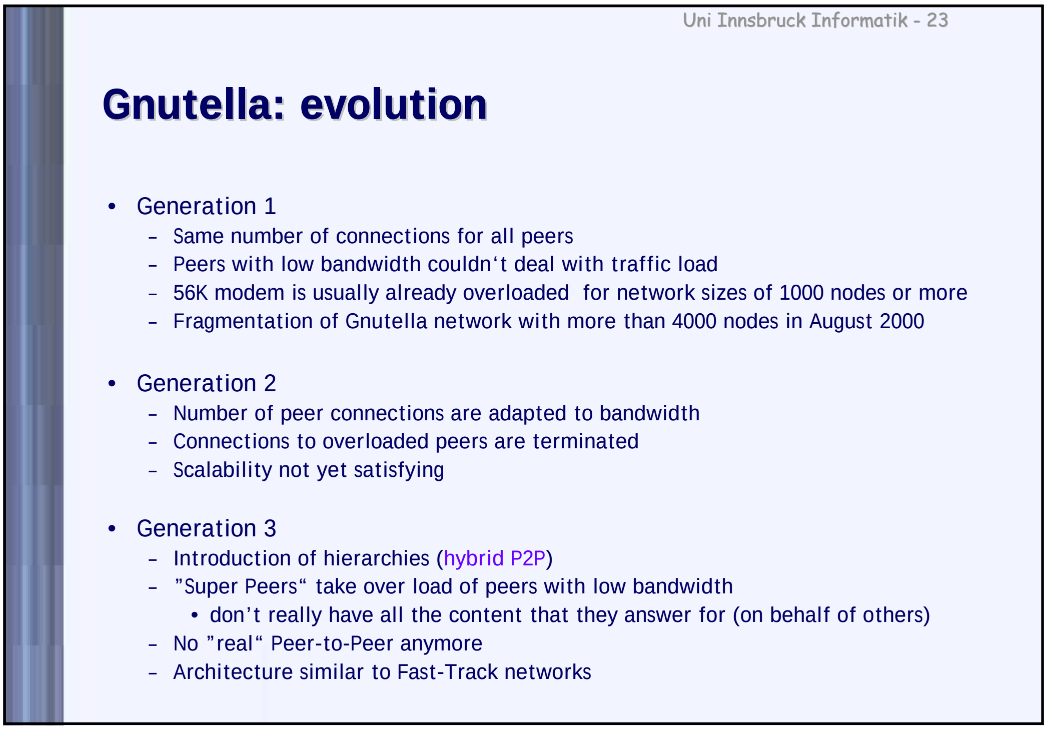
+ Open protocol
  + Easy to write clients for all platforms

+ Very robust against random node failures
  - Not so robust against directed attacks

- Query flooding is extremely inefficient
  - Wastes lot of network and peer resources
  - Partial solution: Limit query radius

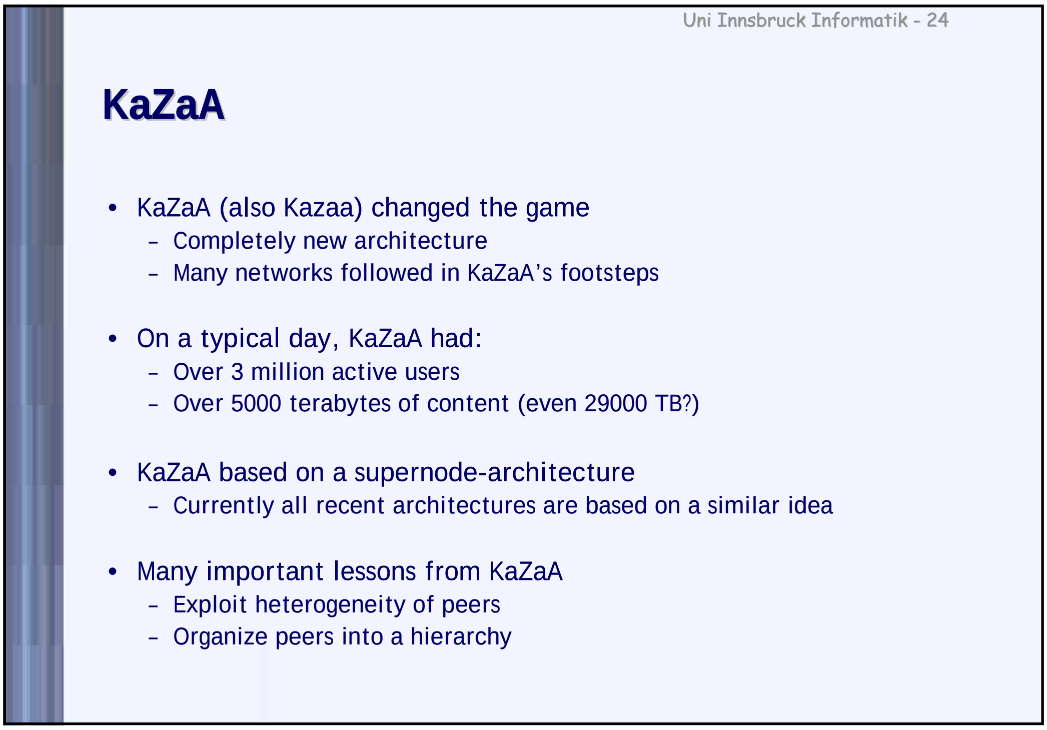
- Gnutella’s network management not efficient
  - Periodic PING/PONGs consume lot of resources

- Queries in Gnutella not very efficient
  - Limited query radius
  - Only a subset of peers receives query
  - Only files at those peers can be found



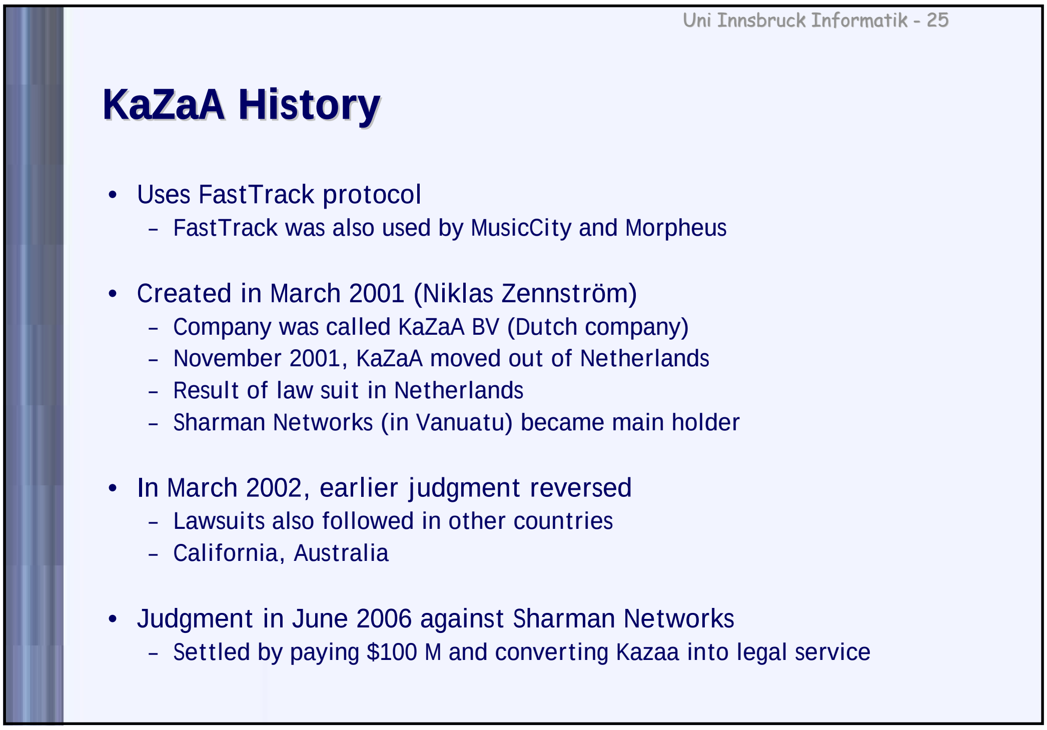
Gnutella: evolution

- Generation 1
  - Same number of connections for all peers
  - Peers with low bandwidth couldn‘t deal with traffic load
  - 56K modem is usually already overloaded for network sizes of 1000 nodes or more
  - Fragmentation of Gnutella network with more than 4000 nodes in August 2000
- Generation 2
  - Number of peer connections are adapted to bandwidth
  - Connections to overloaded peers are terminated
  - Scalability not yet satisfying
- Generation 3
  - Introduction of hierarchies (hybrid P2P)
  - ”Super Peers“ take over load of peers with low bandwidth
    - don’t really have all the content that they answer for (on behalf of others)
  - No ”real“ Peer-to-Peer anymore
  - Architecture similar to Fast-Track networks



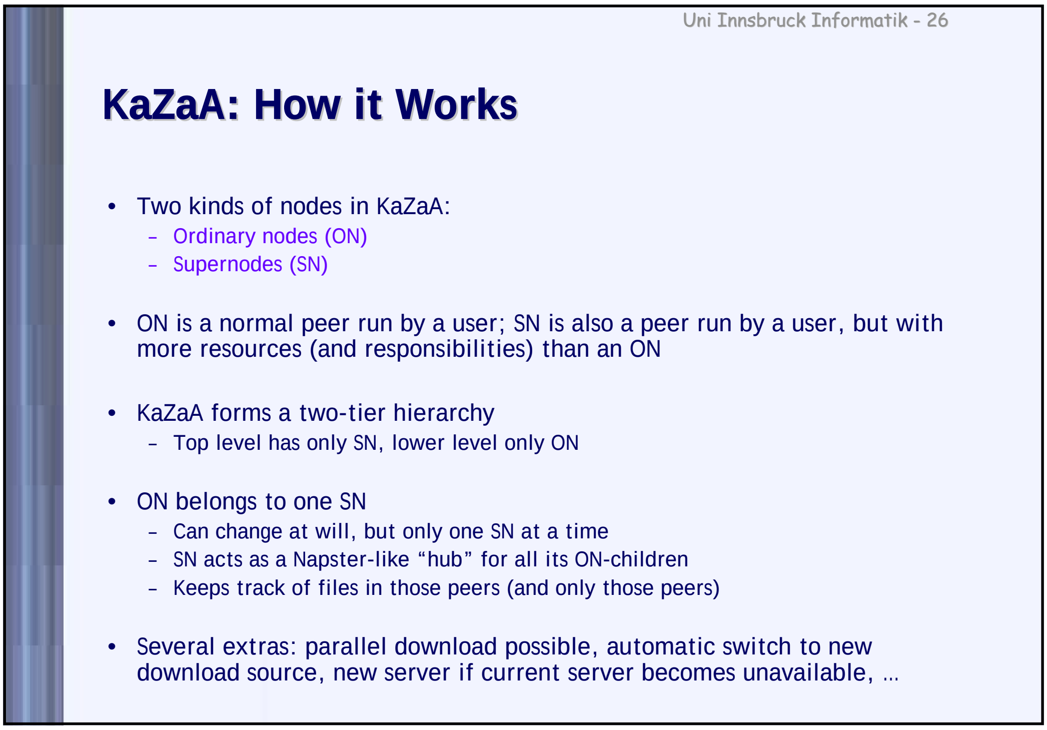
KaZaA

- KaZaA (also Kazaa) changed the game
  - Completely new architecture
  - Many networks followed in KaZaA’s footsteps
- On a typical day, KaZaA had:
  - Over 3 million active users
  - Over 5000 terabytes of content (even 29000 TB?)
- KaZaA based on a supernode-architecture
  - Currently all recent architectures are based on a similar idea
- Many important lessons from KaZaA
  - Exploit heterogeneity of peers
  - Organize peers into a hierarchy



KaZaA History

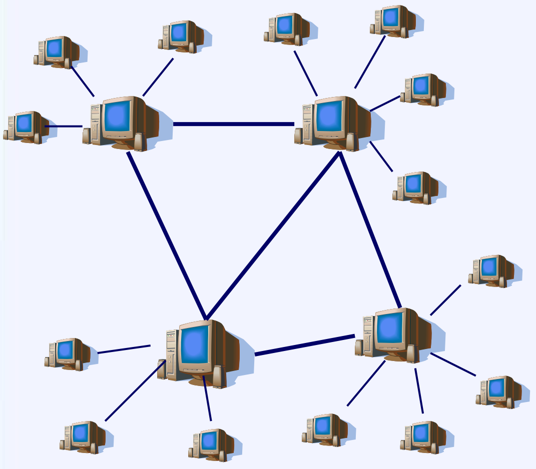
- Uses FastTrack protocol
  - FastTrack was also used by MusicCity and Morpheus
- Created in March 2001 (Niklas Zennström)
  - Company was called KaZaA BV (Dutch company)
  - November 2001, KaZaA moved out of Netherlands
  - Result of law suit in Netherlands
  - Sharman Networks (in Vanuatu) became main holder
- In March 2002, earlier judgment reversed
  - Lawsuits also followed in other countries
  - California, Australia
- Judgment in June 2006 against Sharman Networks
  - Settled by paying $100 M and converting Kazaa into legal service



KaZaA: How it Works

- Two kinds of nodes in KaZaA:
  - Ordinary nodes (ON)
  - Supernodes (SN)
- ON is a normal peer run by a user; SN is also a peer run by a user, but with more resources (and responsibilities) than an ON
- KaZaA forms a two-tier hierarchy
  - Top level has only SN, lower level only ON
- ON belongs to one SN
  - Can change at will, but only one SN at a time
  - SN acts as a Napster-like “hub” for all its ON-children
  - Keeps track of files in those peers (and only those peers)
- Several extras: parallel download possible, automatic switch to new download source, new server if current server becomes unavailable, …

KaZaA Hierarchy

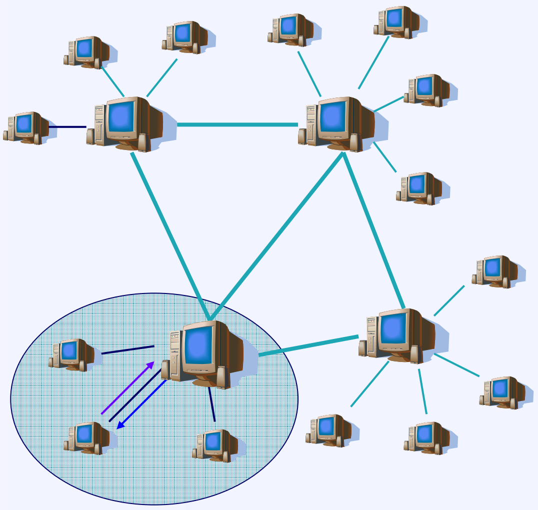


Ordinary Node

SuperNode

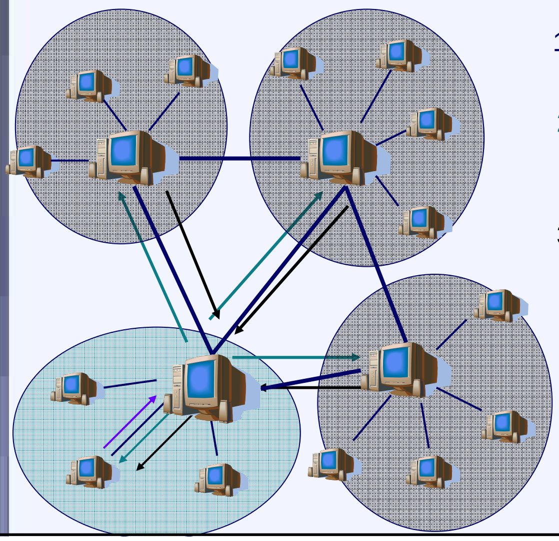
- Ordinary nodes belong to one Supernode
  - Can change SN, but not common (Kazaa-Lite)
- Supernodes exchange information between themselves
- Supernodes do not form a complete mesh

KaZaA: Building the Network



- Peer obtains address of SN from “somewhere”
  - Bootstrap server or included in software
- Peer sends request to SN, gives list of files to share
- SN starts keeping track of this peer
- Other SN not aware of the new peer

KaZaA: Finding Stuff



1. Peer sends query to its own supernode
2. Supernode answers for all of its peers and forwards query to other supernodes
3. Other supernodes reply for all of their peers

KaZaA: Operation

- ON can be promoted to SN if it demonstrates sufficient resources (bandwidth, time on-line)
  - User can typically refuse to become a SN
  - Typical bandwidth requirement for SN 160-200 kbps
    - OK for cable and universities, but problem for DSL!
- SN change connections to other SN on a time scale of tens of minutes
  - Allows for larger range of network to be explored
  - Average lifetime of SN 2.5 hours, but variance is high
- SN does not cache information from disconnected ON
- Estimated 30,000 SN at any given time
  - One SN has connections to 30-50 other SN
- 13% of ON responsible for 80% of uploads

KaZaA: Spyware

- KaZaA includes/included spyware in their program
- Spyware does things like:
  - Sends all DNS queries to a tracking server
  - Monitors visited websites
  - Additional popup windows on “partner” sites
- KaZaA originally denied existence of spyware
- In theory, possible to disable spying functions
  - But removal software often fails…

KaZaA: strengths and weaknesses

+ Combines good points from Napster and Gnutella
  - Efficient searching under each supernode
  - Flooding restricted to supernodes only
  - Result: Efficient searching with “low” resource usage

+ Most popular network (globally)
  - A lot of content, many users
  - Some networks more popular in some areas (e.g., eDonkey in Germany)
  - Currently most big file sharing networks have been shut down

- Queries not comprehensive
  - Can still miss a file even though it exists
  - But better reach than Gnutella

- Single point of failure?
  - Lawsuits against KaZaA eventually successful
  - Software comes with list of “well-known” supernodes
    - Increases robustness?
    - More targets for lawyers?

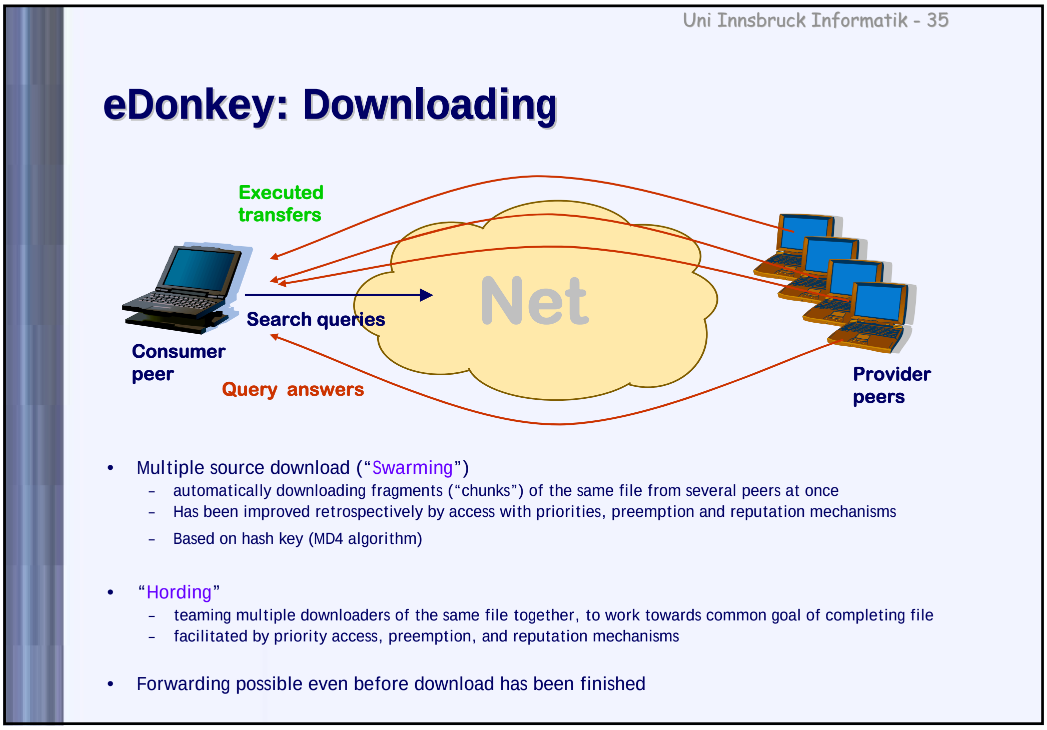
Napster vs. Gnutella vs. KaZaA

	Napster	Gnutella	KaZaA
Type of Network	Centralized	Distributed	Hybrid
Efficient Searching	+++	---	+
Resilience to Attacks	---	++?	+
Open Protocol	N	Y	N
Spyware-free	Y	Y	Y/N?
Popularity	+++	-	+++

eDonkey

- P2P file sharing system developed by “Meta Machine”, NYC, 2000-2001
  - Mainly for sharing of (very) large files: CD images, videos, games, …
- Hybrid P2P architecture
  - Client-server structure: servers don’t store data, just references
  - Clients find servers via server list or web links (ed2k://…)
  - Everybody can set up a server; shut down is more or less “useless”
- Server
  - Stores file references, forwards queries, distributes server lists
  - Originally closed-source, proprietary, no longer maintained
  - Eserver: from scratch rewrite, closed-source, proprietary, free of charge for several OS, in active development, widely used (2008)

- Client
  - eDonkey 2000 from Meta Machine closed source, freeware
  - Most prominent open source implementation: eMule (open source, GPL)
    - Now with support for Kad network: Kademlia DHT used for locating content without servers (structured P2P system!)

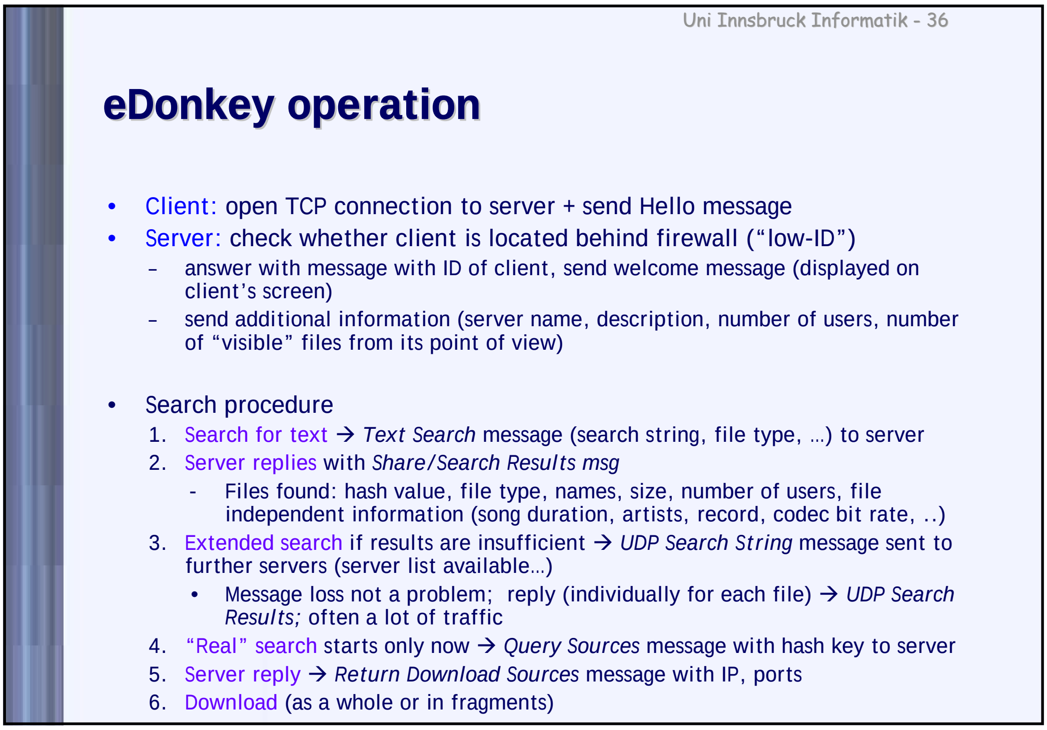


eDonkey: Downloading

- Multiple source download ("Swarming")
  - automatically downloading fragments ("chunks") of the same file from several peers at once
  - Has been improved retrospectively by access with priorities, preemption and reputation mechanisms
  - Based on hash key (MD4 algorithm)

- "Hording"
  - teaming multiple downloaders of the same file together, to work towards common goal of completing file
  - facilitated by priority access, preemption, and reputation mechanisms

- Forwarding possible even before download has been finished

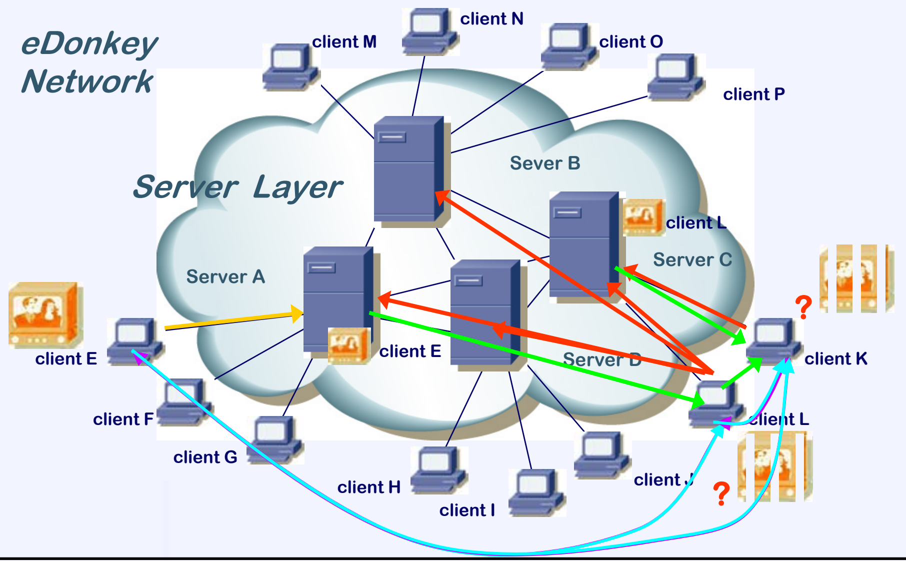


eDonkey operation

- Client: open TCP connection to server + send Hello message
- Server: check whether client is located behind firewall ("low-ID")
  - answer with message with ID of client, send welcome message (displayed on client's screen)
  - send additional information (server name, description, number of users, number of "visible" files from its point of view)

- Search procedure
  1. Search for text → *Text Search* message (search string, file type, …) to server
  2. Server replies with *Share/Search Results msg*
     - Files found: hash value, file type, names, size, number of users, file independent information (song duration, artists, record, codec bit rate, ..)
  3. Extended search if results are insufficient → *UDP Search String* message sent to further servers (server list available…)
     - Message loss not a problem; reply (individually for each file) → *UDP Search Results;* often a lot of traffic
  4. "Real" search starts only now → *Query Sources* message with hash key to server
  5. Server reply → *Return Download Sources* message with IP, ports
  6. Download (as a whole or in fragments)

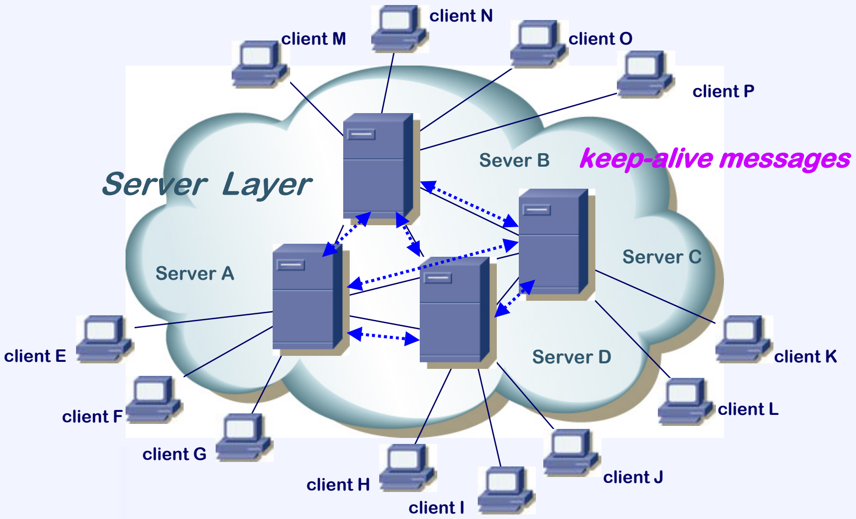
3.5 Search and Download in eDonkey2000



Search and Download in eDonkey2000 /2

- Client – client traffic:
  - Multiple TCP connections setup to clients that have been found through *Query Sources*
    - Chunks have size up to 9.28 MB
    - Composition of file from fragments
    - Available fragments are offered immediately (!)
      - Popular content spreads quickly
      - In case of fragments with error, alternative clients can be selected
  - If client is behind firewall:
    - The client receives a much smaller ID during registration (~download priority)
    - If that client only has data → upload possibility included in protocol

Overlay maintenance in eDonkey2000



Problems with eDonkey2000

1. Fakes ("~Lord of the Rings")
   - Misleading file names chosen intentionally
   - Some fragments may even correspond to original
   - ...usefulness is limited after long download.. ;-)
   - → Record/media companies? Self protection?
   - Web sites (e.g, http://www.sharereactor.com) with names and hash values of fakes plus pointers to the "real" content

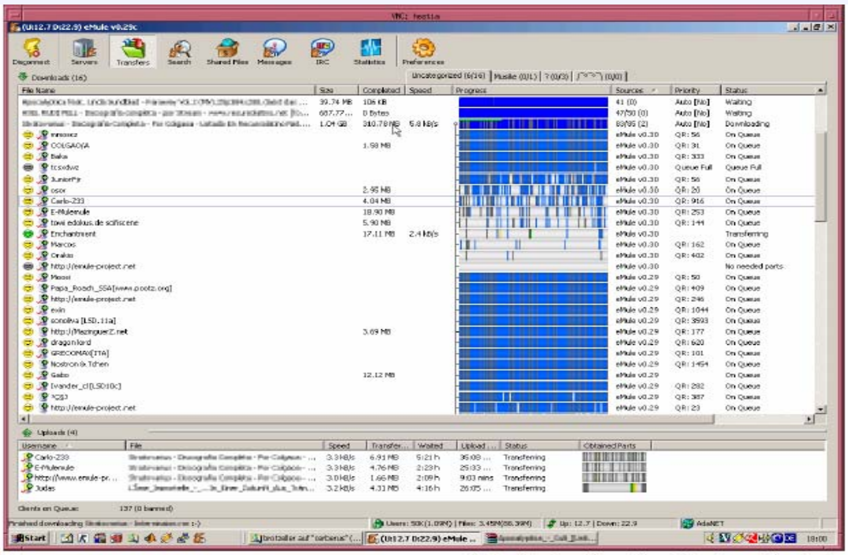
2. UDP traffic
   - "No" control
   - Simultaneous search on many servers creates high load in the network (relation UDP:TCP ca. 9:1)
   - Servers often overloaded but the traffic does not slow down
   - Path of a server may continue to exist for many hours after shutdown

Problems with eDonkey2000 /2

3. DDoS – attacks
   - UDP source address can easily be faked
   - Addresses of servers are openly available...

- Free riders (also a major problem in Gnutella!)
  - Do not provide any resources
  - Only interested in downloading
  - Appropriate methods are investigated for making them more “active”
    - To forbid downloads?
    - Imposing an upper limit on the relation of inbound to outbound traffic?
    - To calculate priorities based on outbound traffic?

eDonkey2000 in action



File Name	Size	Completed	Speed	Progress	Sources	Priority	Status
[illegible]	39.74 MB	106 KB			41 (0)	Auto [No]	Waiting
[illegible]	687.77...	0 Bytes			47/50 (0)	Auto [No]	Waiting
[illegible]	1.04 GB	310.78 MB	5.8 kB/s		83/85 (2)	Auto [No]	Downloading
mmoncz					eMule v0.30	QR: 56	On Queue
COLGAOfA		1.58 MB			eMule v0.30	QR: 31	On Queue
Baka					eMule v0.30	QR: 333	On Queue
tcsxdwe					eMule v0.30	Queue Full	Queue Full
JuniorPjr					eMule v0.30	QR: 56	On Queue
osor		2.95 MB			eMule v0.30	QR: 20	On Queue
Carlo-233		4.04 MB			eMule v0.30	QR: 916	On Queue
E-Mulemule		18.90 MB			eMule v0.30	QR: 253	On Queue
towi edolus.de scifiscene		5.90 MB			eMule v0.30	QR: 144	On Queue
Enchantment		17.11 MB	2.4 kB/s		eMule v0.30		Transferring
Marcos					eMule v0.30	QR: 162	On Queue
Orakio					eMule v0.30	QR: 402	On Queue
http://emule-project.net					eMule v0.30		No needed parts
Mooni					eMule v0.29	QR: 50	On Queue
Papa_Roach_SSA[www.pootz.org]					eMule v0.29	QR: 409	On Queue
http://emule-project.net					eMule v0.29	QR: 246	On Queue
exin					eMule v0.29	QR: 1044	On Queue
sonoliva [LSD.11a]					eMule v0.29	QR: 3593	On Queue
http://MazinguerZ.net		3.69 MB			eMule v0.29	QR: 177	On Queue
dragonlord					eMule v0.29	QR: 620	On Queue
GRECOMAX[ITA]					eMule v0.29	QR: 101	On Queue
Nostron & Tdhen					eMule v0.29	QR: 1454	On Queue
Gabo		12.12 MB			eMule v0.29		On Queue
[vander_cl[LSD10c]					eMule v0.29	QR: 282	On Queue
?2SJ					eMule v0.29	QR: 387	On Queue
http://emule-project.net					eMule v0.29	QR: 23	On Queue

Username	File	Speed	Transfer...	Waited	Upload ...	Status	Obtained Parts
Carlo-233	[illegible]	3.3 kB/s	6.91 MB	5:21 h	35:08 ...	Transferring	
E-Mulemule	[illegible]	3.3 kB/s	4.76 MB	2:23 h	25:33 ...	Transferring	
http://www.emule-pr...	[illegible]	3.0 kB/s	1.66 MB	2:09 h	9:03 mins	Transferring	
Judas	[illegible]	3.2 kB/s	4.31 MB	4:16 h	26:05 ...	Transferring	

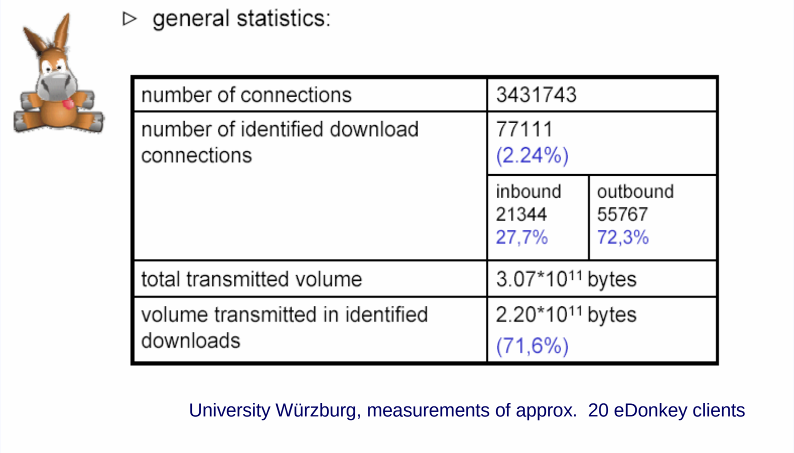
Clients on Queue: 137 (0 banned)

Finished downloading [illegible] :-)

Users: 50K(1.09M) | Files: 3.45M(55.39M) | Up: 12.7 | Down: 22.9 | AdaNET

A Traffic Profile of the EDonkey Filesharing Service

Some statistics of eDonkey2000



▷ general statistics:

number of connections	3431743	
number of identified download connections	77111 (2.24%)	
	inbound 21344 27,7%	outbound 55767 72,3%
total transmitted volume	$3.07*10^{11}$ bytes	
volume transmitted in identified downloads	$2.20*10^{11}$ bytes (71,6%)	

University Würzburg, measurements of approx. 20 eDonkey clients

BitTorrent

- New approach for sharing large files
- BitTorrent also widely used for legal content
  - For example, Linux distributions, software patches
  - Official movie distributions are also planned (WB)
- Main goal: Quickly replicate file to large number of clients
  - File sharing networks attempt to provide as much content as possible for download
- BitTorrent more appropriately called peer-to-peer content distribution
  - BitTorrent has also had its share of litigation

P2P Content Distribution

- BitTorrent builds a network for every file that is being distributed

Big advantage of BitTorrent:

- Can send "link" to a friend
- "Link" always refers to the same file

- Not really feasible on Napster, Gnutella, or KaZaA
  - These networks are based on searching, hard to identify a particular file
  - Downside of BitTorrent: No searching possible
    - Websites with "link collections" and search capabilities exist

BitTorrent History

- BitTorrent developed by Bram Cohen in 2001
  - Written in Python, available on many platforms

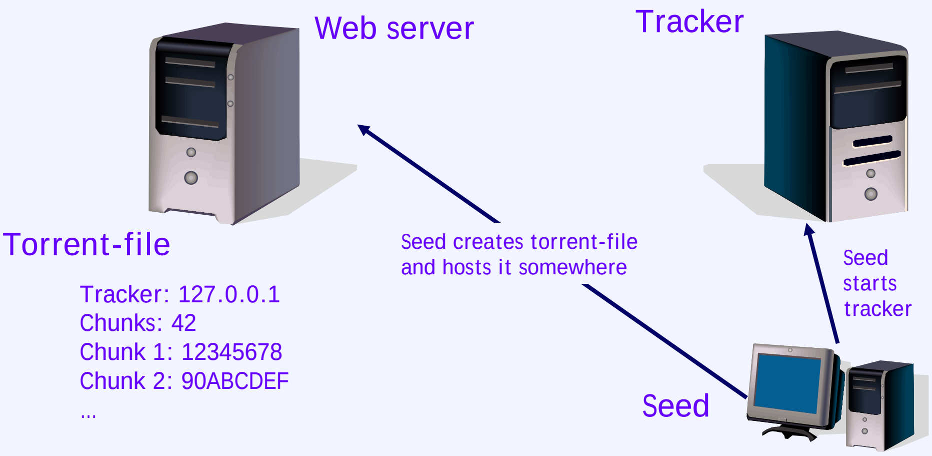
- Uses old upload/download-ratio concept from BBSs
  - "The more you give, the more you get"
  - Participation enforced in protocol
  - Other P2P systems have adopted similar ideas

- Why was there little content on BitTorrent for a while?
  - No search functionality?
  - Original source easily identified?
  - Currently lots of illegal content on BitTorrent too…

BitTorrent: operation

- For each file shared on BitTorrent, there is (initially) one server which hosts the original copy
  - File broken into chunks
- A “torrent” file which gives metadata about the file
  - Torrent file hosted typically on a web server
- Client downloads torrent file
  - Metadata indicates the sizes of chunks and their checksums
  - Metadata identifies a tracker
- Tracker is a server which tracks the currently active clients
  - Tracker does not participate in actual distribution of file
  - Law suits against people running trackers have been successful, even though tracker holds no content

BitTorrent: Players



Web server

Tracker

Torrent-file

Tracker: 127.0.0.1
Chunks: 42
Chunk 1: 12345678
Chunk 2: 90ABCDEF
…

Seed creates torrent-file and hosts it somewhere

Seed starts tracker

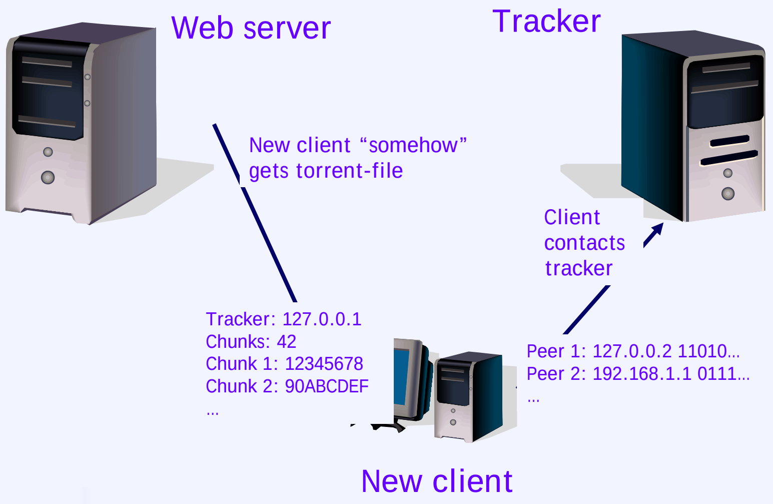
Seed

- 3 entities needed to start distribution of a file

BitTorrent: operation

- Terminology:
  - Seed: Client with a complete copy of the file
  - Leecher: Client still downloading the file
- Client contacts tracker and gets a list of other clients
  - Gets list of 50 peers
- Client maintains connections to 20-40 peers
  - Contacts tracker if number of connections drops below 20
  - This set of peers is called peer set
  - Client downloads chunks from peers in peer set and provides them with its own chunks
    - Chunks typically 256 KB
    - Chunks make it possible to use parallel download

BitTorrent: Starting Up



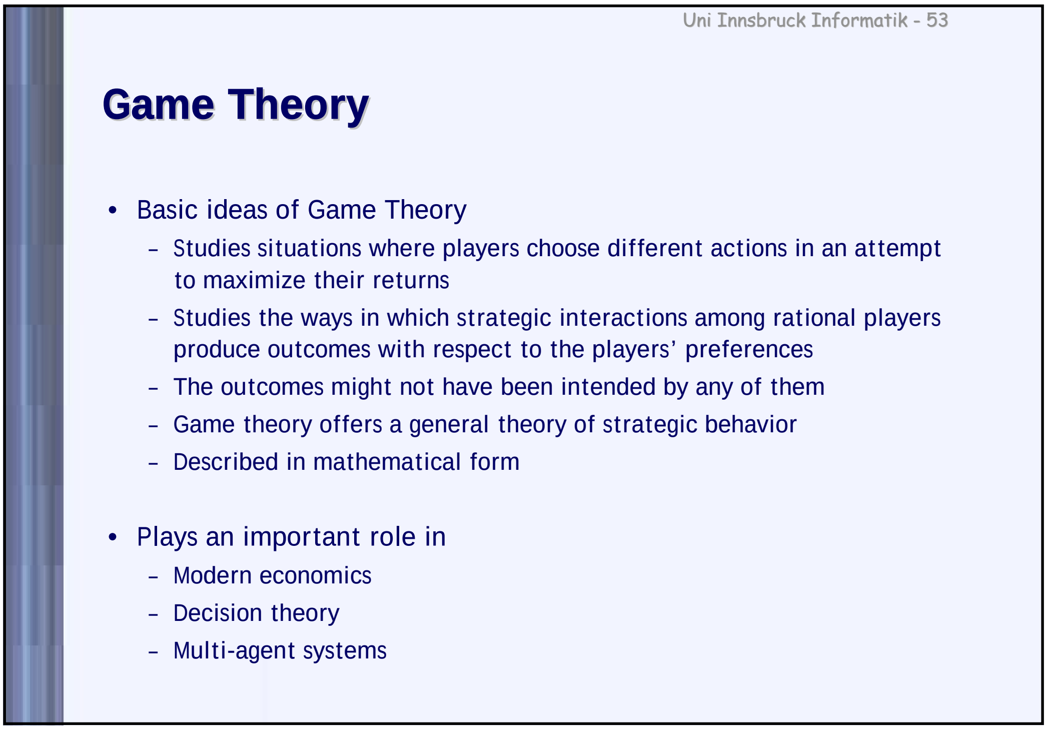
- New client gets torrent-file and gets peer list from tracker

BitTorrent: Tit-for-Tat policy

- A peer serves peers that serve it
  - Encourages cooperation, discourage free-riding
- Peers use rarest first policy when downloading chunks
  - Having a rare chunk makes peer attractive to others
  - Others want to download it, peer can then download the chunks it wants
  - Goal of chunk selection is to maximize entropy of each chunk
- For first chunk, just randomly pick something, so that peer has something to share
- Finishing active chunks: strict priority policy
- Endgame mode
  - Send requests for last sub-chunks to all known peers
  - End of download not stalled by slow peers

Choking

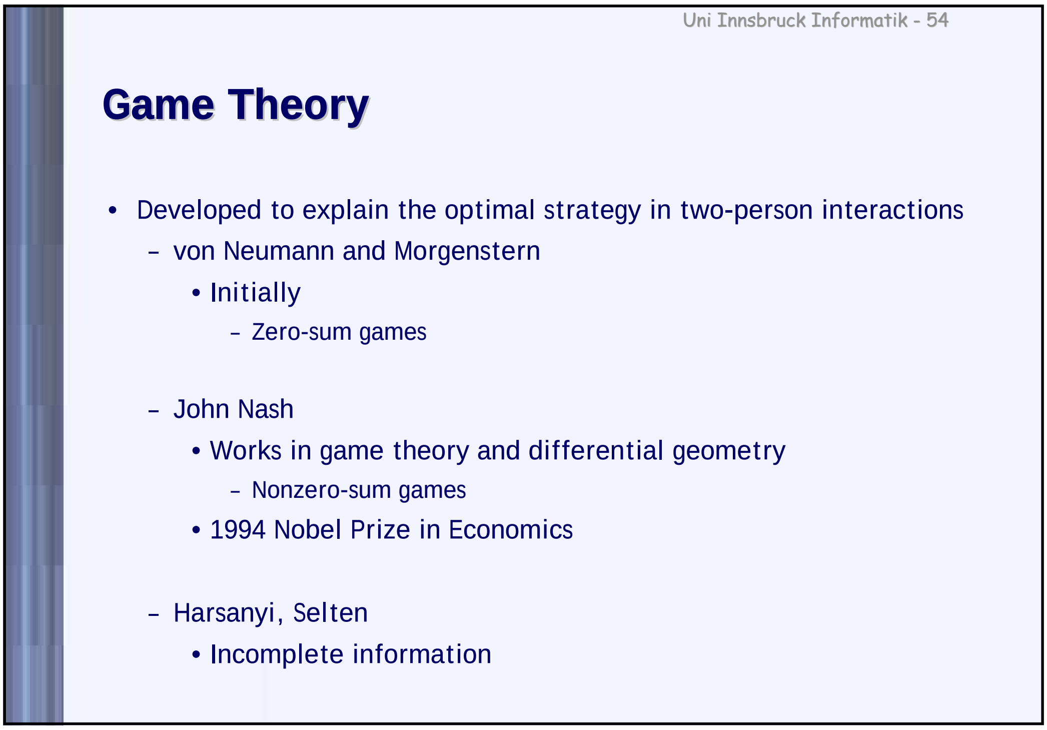
- Choking mechanism
  - Ensures that nodes cooperate
  - Eliminates the free-rider problem
  - Cooperation involves uploaded sub-pieces that you have to your peer
- Choking
  - Temporary refusal to upload
  - Downloading occurs as normal
  - Connection is kept open
    - No Setup costs
    - Fairness between connections due to TCP congestion control
- Based on game-theoretic concepts
  - Tit-for-tat strategy in repeated games



Game Theory

- Basic ideas of Game Theory
  - Studies situations where players choose different actions in an attempt to maximize their returns
  - Studies the ways in which strategic interactions among rational players produce outcomes with respect to the players’ preferences
  - The outcomes might not have been intended by any of them
  - Game theory offers a general theory of strategic behavior
  - Described in mathematical form

- Plays an important role in
  - Modern economics
  - Decision theory
  - Multi-agent systems



Game Theory

- Developed to explain the optimal strategy in two-person interactions
  - von Neumann and Morgenstern
    - Initially
      - Zero-sum games
  - John Nash
    - Works in game theory and differential geometry
      - Nonzero-sum games
    - 1994 Nobel Prize in Economics
  - Harsanyi, Selten
    - Incomplete information

Definitions

- Games
  - Situations are treated as games
- Rules
  - The rules of the game state who can do what, and when they can do it
- Player's Strategies
  - Plan for actions in each possible situation in the game
- Player's Payoffs
  - Is the amount that the player wins or looses in a particular situation
- Dominant Strategy
  - If players best strategy doesn't depend on what other players do
- Nash equilibrium
  - A situation where no player has anything to gain by changing only her own strategy
- Pareto efficient (Pareto optimal)
  - A situation where any change that makes at least one player better off also makes at least one other player worse off

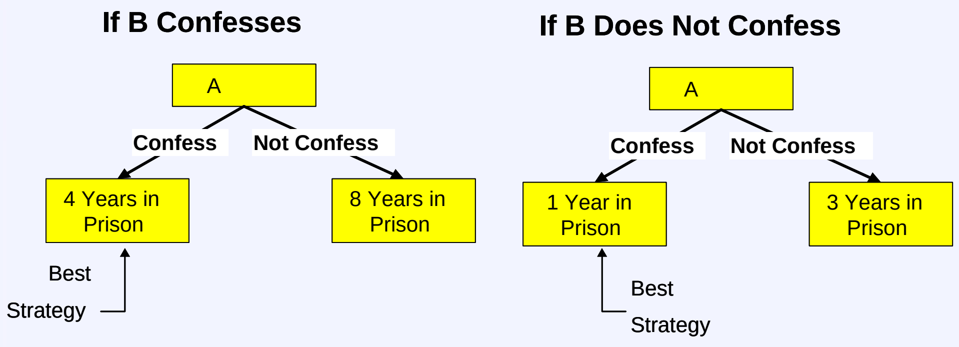
Prisoner's Dilemma

- Famous example of game theory
- A and B are arrested by the police
  - They are questioned in separate cells
    - Unable to communicate with each other
  - They know how it works
    - If they both resist interrogation and proclaim their mutual innocence, they will get off with a three year sentence for robbery
    - If one of them confesses to the entire string of robberies and the other does not, the confessor will be rewarded with a light, one year sentence and the other will get a severe eight year sentence
    - If they both confess, then the judge will sentence both to a moderate four years in prison

Prisoner's Dilemma

		B	
		Confess	**Not Confess**
A	**Confess**	4 years each	1 year for Bonnie and 8 years for Clyde
	Not Confess	8 years for Bonnie and 1 year for Clyde	3 years each

A’s Decision Tree



- The dominant strategy for A is to confess

⇒ Prisoner’s dilemma has a non-Pareto-optimal Nash equilibrium

Repeated Games

- A repeated game
  - Game that the same players play more than once
  - Differ form one-shot games because people's current actions can depend on the past behavior of other players
  - Cooperation is encouraged
- Interesting effects can arise… e.g. Shubik Dollar Auction:
  - Auction with a simple change of rules: highest bidder wins, second-highest bidder gets nothing but still has to pay
  - Better to pay $ 1,10 and get $1 than to pay $0,90 and get nothing!
  - Known to work, and make people angry…
- Repeated prisoner’s dilemma: famous experiment by Robert Axelrod
  - Tournament between computer programs with different strategies
  - Tit-for-tat won

Tit for tat

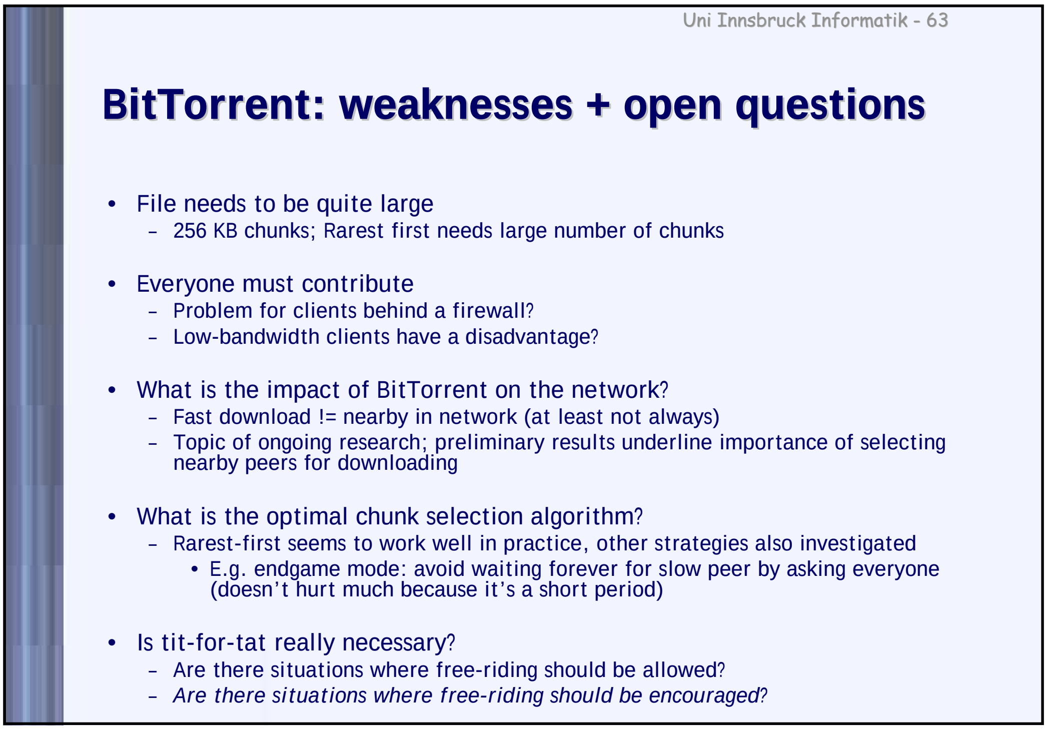
- Tit for tat
  - Highly effective strategy
  - An agent using this strategy will initially cooperate
  - Then respond in kind to an opponent's previous action
  - If the opponent previously was cooperative, the agent is cooperative
  - If not, the agent is not
- Dependent on four conditions
  - Unless provoked, the agent will always cooperate
  - If provoked, the agent will retaliate
  - The agent is quick to forgive
  - The agent must have a good chance of competing against the opponent more than once

BitTorrent: Choke/Unchoke

- Peer serves e.g. 4 (default value) peers in peer set simultaneously
  - Seeks best (fastest) downloaders if it's a seed
  - Seeks best uploaders if it's a leecher
- Choke is a temporary refusal to upload to a peer
  - Leecher serves 4 best uploaders, chokes all others
  - Every 10 seconds, it evaluates the transfer speed
    - Based on 20-second moving average
    - 10 second decision interval: chosen to avoid rapidly choking/unchoking peers
  - If there is a better peer, choke the worst of the current 4
- Every 30 seconds peer makes an "optimistic unchoke"
  - Randomly unchoke a peer from peer set
  - Idea: Maybe it offers better service
- Seeds behave exactly the same way, except they look at download speed instead of upload speed

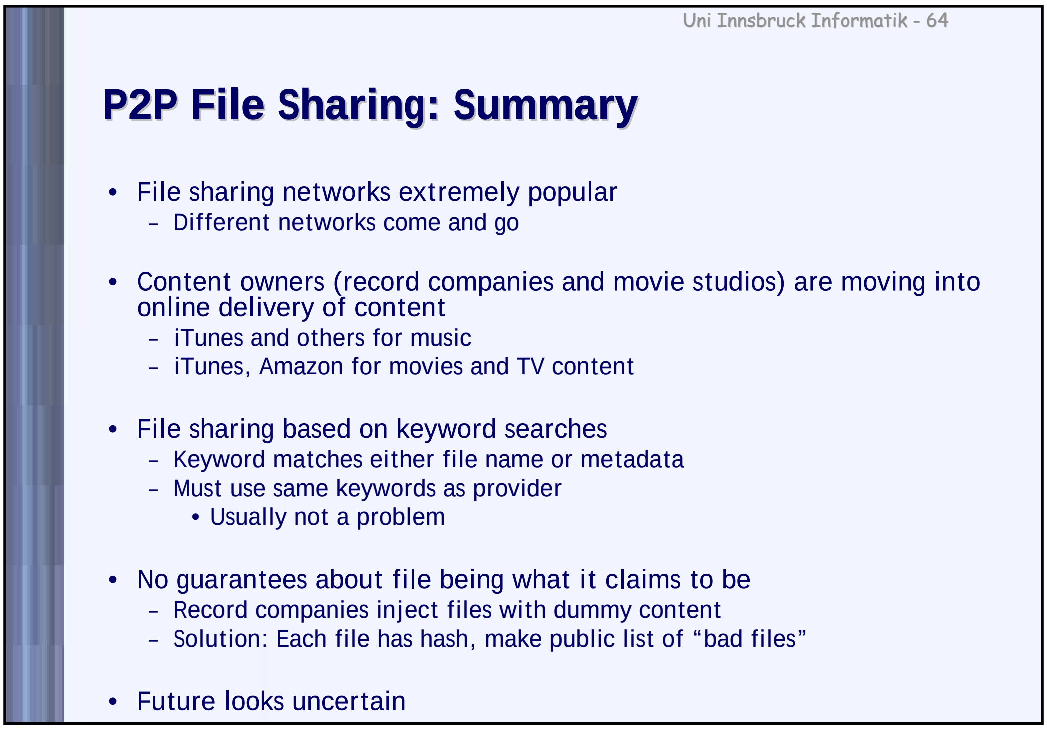
BitTorrent: Strengths

- Works quite well
  - Download a bit slow in the beginning, but speeds up considerably as peer gets more and more chunks
- Users keep their peers connected as seeds
  - Legal content, so no need to worry?
  - Large download, leave running over night?
  - How necessary is this?
- Those who want the file must contribute
  - Attempts to minimize free-riding
- Efficient mechanism for distributing large files to many clients
  - Popular software, updates, …
  - See also Avalanche from Microsoft Research



BitTorrent: weaknesses + open questions

- File needs to be quite large
  - 256 KB chunks; Rarest first needs large number of chunks
- Everyone must contribute
  - Problem for clients behind a firewall?
  - Low-bandwidth clients have a disadvantage?
- What is the impact of BitTorrent on the network?
  - Fast download != nearby in network (at least not always)
  - Topic of ongoing research; preliminary results underline importance of selecting nearby peers for downloading
- What is the optimal chunk selection algorithm?
  - Rarest-first seems to work well in practice, other strategies also investigated
    - E.g. endgame mode: avoid waiting forever for slow peer by asking everyone (doesn't hurt much because it's a short period)
- Is tit-for-tat really necessary?
  - Are there situations where free-riding should be allowed?
  - *Are there situations where free-riding should be encouraged?*



P2P File Sharing: Summary

- File sharing networks extremely popular
  - Different networks come and go
- Content owners (record companies and movie studios) are moving into online delivery of content
  - iTunes and others for music
  - iTunes, Amazon for movies and TV content
- File sharing based on keyword searches
  - Keyword matches either file name or metadata
  - Must use same keywords as provider
    - Usually not a problem
- No guarantees about file being what it claims to be
  - Record companies inject files with dummy content
  - Solution: Each file has hash, make public list of "bad files"
- Future looks uncertain