ELT-53206 Peer-to-Peer Networks

P2P File Sharing: BitTorrent in Detail

Mathieu Devos
Tampere University of Technology
Department of Electronics & Communications Engineering
mathieu.devos@tut.fi – TG406

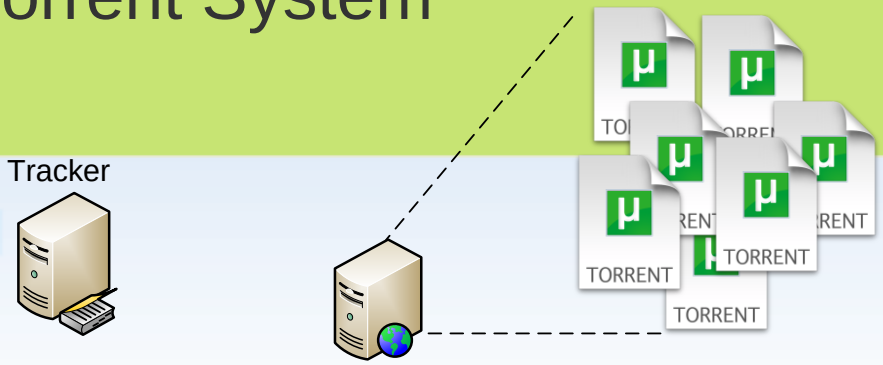
Outline

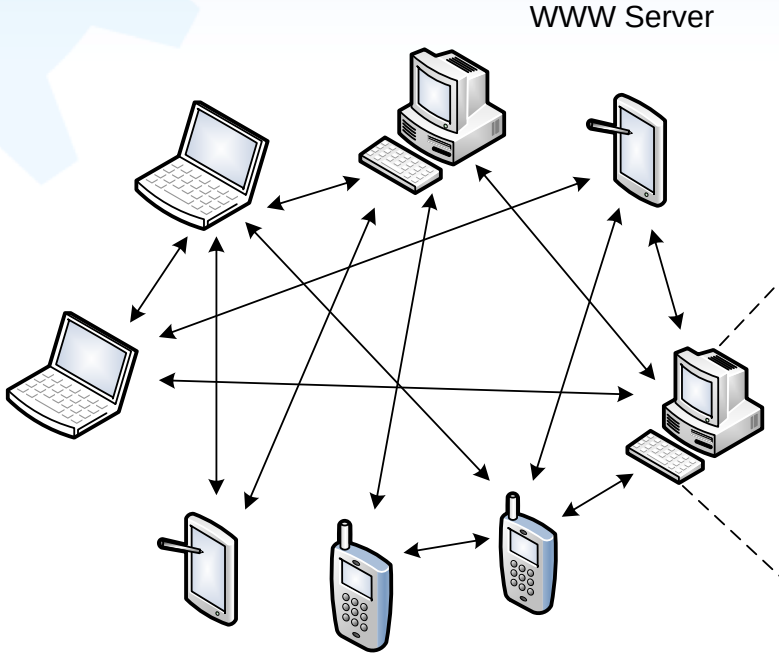
1. BitTorrent System
    1. Torrent file
    2. Tracker
    3. Download protocol
2. Piece Selection
    1. Random first piece
    2. Rarest piece first
    3. Endgame mode
3. Choking Mechanism
4. BitTorrent Enhancement Proposals
    1. Multitracker metadata extension
    2. Peer exchange
    3. DHT protocol
5. Clustered BitTorrent Concept
6. Some statistics
7. Conclusions

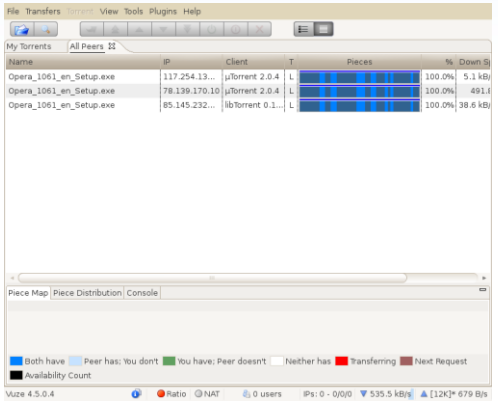
Outline

1. BitTorrent System
   1. Torrent file
   2. Tracker
   3. Download protocol
2. Piece Selection
   1. Random first piece
   2. Rarest piece first
   3. Endgame mode
3. Choking Mechanism
4. BitTorrent Enhancement Proposals
   1. Multitracker metadata extension
   2. Peer exchange
   3. DHT protocol
5. Clustered BitTorrent Concept
6. Some statistics
7. Conclusions

BitTorrent System







BitTorrent System

- Complete BitTorrent file distribution system consists of following entities
  - A static metadata file (torrent file) constructed of the content to be shared
  - A WWW server for publishing torrent files (or published by other means)
  - A BitTorrent tracker
  - An original peer publishing content (an original seed) which can leave the network if the whole content is available among other peers in the system
  - WWW browser for finding torrent files
  - Downloader application which can handle torrent files and operate in the BitTorrent network

- Features
  - Multisource downloading (simultaneous downloading from multiple sources)
  - Uploading the content while still downloading it
    - Together with tit-for-tat policy provides efficient spreading of the content
  - Integrity checks done at the piece level
    - One corrupted piece will not ruin the whole download process

Torrent File

- Used for finding actual content (via tracker) and verifying integrity of the content
- Includes piece hashes (SHA-1)
  - Content is partitioned into pieces (typically 256KB) and hash values are calculated from them
- Other information: piece length, file name, content length, the URL of the tracker, etc.
- Torrent file is **bencoded** (BitTorrent's own encoding)

```
<?xml version="1.0" encoding="UTF-8"?>
<tor:TORRENT
   xmlns:tor=http://azureus.sourceforge.net/files
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://azureus.sourceforge.net/files
            http://azureus.sourceforge.net/files/torrent.xsd">
 <ANNOUNCE_URL>http://torrent.opera.com:6969/announce</ANNOUNCE_URL>
 <CREATION_DATE>1281694453</CREATION_DATE>
 <TORRENT_HASH>2EEAB52F0242C42AC788D17924CCC197DC3D2F0C</TORRENT_HASH>
 <INFO>
  <NAME encoding="utf8">Opera_1061_en_Setup.exe</NAME>
  <PIECE_LENGTH>262144</PIECE_LENGTH>
  <LENGTH>10809256</LENGTH>
  <PIECES>
   <BYTES>54E6CCEC9EB3BEA12711FB9F383890023650D5AF</BYTES>
   <BYTES>D134F302346EFB60E57BA240F8F38BD27A408C35</BYTES>
   <BYTES>73811C5D1313F5A16A18BFF77DCAD18CADFB3F97</BYTES>
   .
   .
   .
   <BYTES>6AE8353393A66804FF95C98624331824A6F61D62</BYTES>
  </PIECES>
 </INFO>
</tor:TORRENT>
```

Bencoding

- Strings are length-prefixed (base ten) followed by a colon and the string itself
  - 10:bittorrent encodes the string ‘bittorrent’
- Integers are represented by an ‘i’ followed by the number in base ten followed by an e
  - i23e encodes the integer 23
- Lists are encoded as an ‘l’ followed by their elements (also bencoded) followed by an ‘e’
  - l10:bittorrenti23ee encodes the string “bittorrent” and the integer 23
- Dictionaries are encoded as a ‘d’ followed by a list of alternating keys and their corresponding values followed by an ‘e’
  - d3:agei30e4:name5:james5likesl4:food5:drinkee encodes the structure

    age: 30

    name: james

    likes: {food, drink}

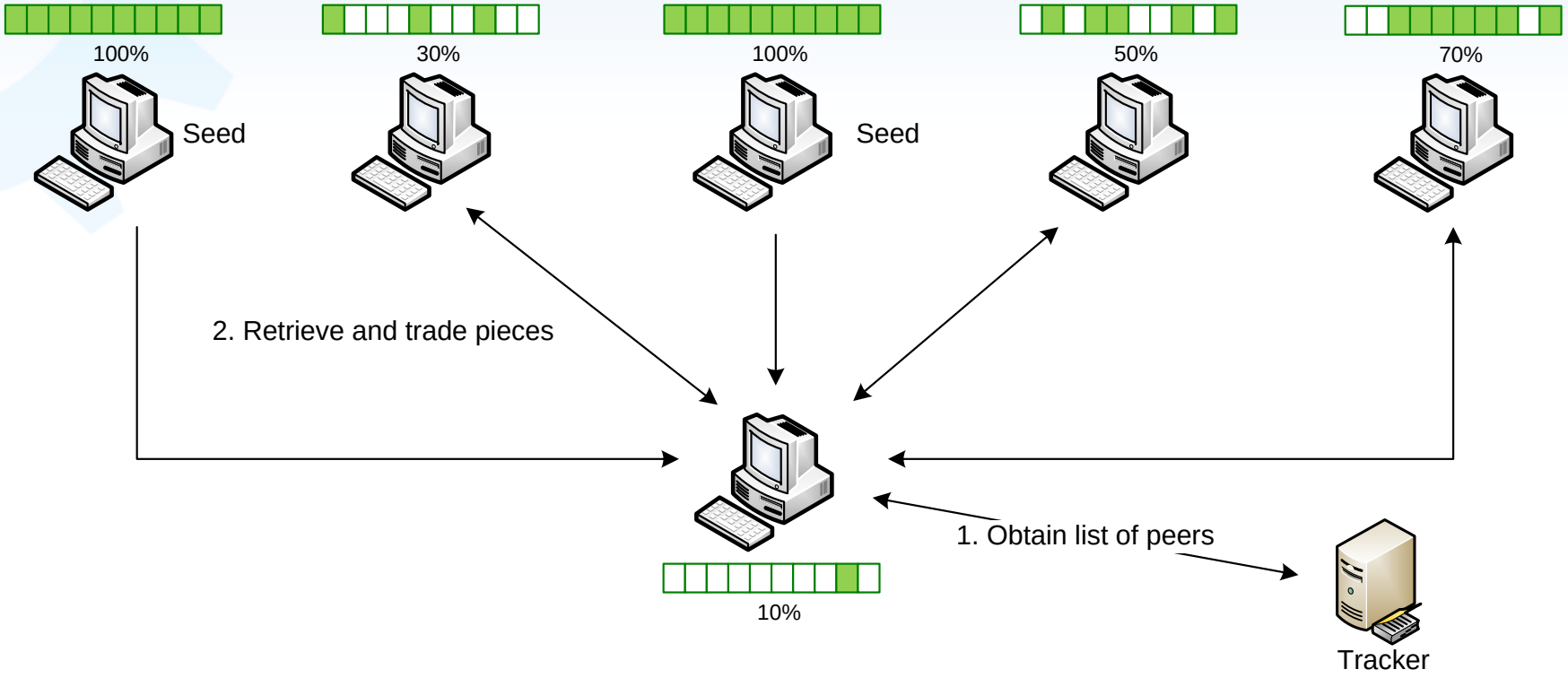
Tracker

- Used to find other peers' contact information who are interested of the same torrent
- Stores peers' contact information by the torrent bases
- Private and public trackers
  - Anybody can use public trackers for publishing content
  - Private flag in info part (adopted but not officially accepted in protocol)
- Peer uses HTTP protocol for tracker communication
  - 160 bit info hash is used to identify torrents (and content) in the network
  - Calculated from the info part of the torrent file (SHA-1)
- Tracker is connected when starting, refreshing, stopping or completing a torrent
  - Starting and re-announcing both return a peer list for the asked info hash (torrent)
  - Re-announcements are done at regular intervals
  - Tracker must have timeout for the peers to remove non-existing peers
- Tracker can keep different kind of statistics for the torrents
- Single point of failure
  - Target for lawsuits (TBP tracker forever down)
  - Crawling trackers (seize servers = know who downloads what)

Download Protocol

- Peers communicate with each other using BitTorrent protocol on top of TCP
- Peers interested for the same torrent form an independent mesh overlay network where each peer work together for a common target to complete the file download as fast as possible
- Once a peer has the entire file, it may (selfishly) leave or (altruistically) remain as a seed
- Messages
  - `choke`, `unchoke`, `interested`, `not interested`, `have`, `bitfield`, `request`, `piece`, `cancel`, `port`, `suggest`, `have all`, `have none`, `reject request`, `allowed fast`, `extended`, ...

Download Protocol



Outline

1. BitTorrent System
   1. Torrent file
   2. Tracker
   3. Download protocol
2. Piece Selection
   1. Random first piece
   2. Rarest piece first
   3. Endgame mode
3. Choking Mechanism
4. BitTorrent Enhancement Proposals
   1. Multitracker metadata extension
   2. Peer exchange
   3. DHT protocol
5. Clustered BitTorrent Concept
6. Some statistics
7. Conclusions

Piece Selection

- The order in which pieces are downloaded by different peers is important for good performance
- A poor algorithm can end up in a situation where every peer has all the pieces that are currently available and none of the missing ones
  - If the original seed is taken down, the file cannot be completely downloaded
- **First policy** for piece selection is that once a single sub-piece has been requested, the remaining sub-pieces from that particular piece are requested before sub-pieces from any other piece
  - This does a good job of getting complete pieces as quickly as possible

Random First Piece

- When downloading starts, a peer has nothing to trade, so it is important to get a complete piece as quickly as possible
- Rare pieces are typically only present on one peer, so downloading a rare piece initially is not a good idea
  - It would be downloaded slower than pieces which are present on multiple peers for which it's possible to download sub-pieces from different places
- **Policy:** Pieces to download are selected at random until the first complete piece is assembled, and then the strategy changes to rarest first

Rarest Piece First

- **Policy:** When selecting which piece to start downloading next, peers generally download pieces which the fewest of their own peers have first
  - Usually referred to as ’rarest first’
- This technique does a good job of making sure that peers have pieces which all of their peers want, so uploading can be done when wanted
- It also makes sure that pieces which are more common are left for later
  - The likelihood that a peer which currently is offering upload will not later have anything of interest is reduced

Endgame Mode

- Sometimes a piece will be requested from a peer with very slow transfer rates
  - Not a problem in the middle of a download, but could potentially delay a download's finish
- **Policy:** To keep that from happening, once all sub-pieces which a peer does not have are actively being requested it sends requests for all sub-pieces to all peers
- `cancel` messages are sent for sub-pieces which arrive to keep too much bandwidth from being wasted on redundant sends
- In practice not much bandwidth is wasted this way, since the endgame period is very short, and the end of a file is always downloaded quickly

Outline

1. BitTorrent System
    1. Torrent file
    2. Tracker
    3. Download protocol
2. Piece Selection
    1. Random first piece
    2. Rarest piece first
    3. Endgame mode
3. Choking Mechanism
4. BitTorrent Enhancement Proposals
    1. Multitracker metadata extension
    2. Peer exchange
    3. DHT protocol
5. Clustered BitTorrent Concept
6. Some statistics
7. Conclusions

Choking Mechanism

- One of BitTorrent's most powerful idea is the choking mechanism which ensures that nodes cooperate and eliminates the free-rider problem
- Each BitTorrent peer always unchokes a fixed number of other peers, so the issue becomes which peers to unchoke
  - This approach allows TCP's built-in congestion control to reliably saturate upload capacity
- Peers reciprocate uploading to top four leecher peers who they themselves have successfully downloaded from
  - Decision to choke/unchoke is performed every 10 seconds based on current download rates, which are evaluated on a rolling 20 seconds average
- Choking is a temporary refusal to upload; downloading occurs as normal
  - Connection is kept open so that setup costs are not borne again and again

Optimistic Unchoking

- Simply uploading to the peers which provide the best download rate would suffer from having no method of discovering if currently unused connections are better than the ones being used
- To fix this, a BitTorrent peer has a single ‘optimistic unchoke’, a peer which is unchoked regardless of the current download rate from it
- Optimistic unchoking is performed every 30 seconds to possibly become top four uploader for some random leecher peer, in which case the random leecher peer would start to send data back

Anti-snubbing

- Occasionally a BitTorrent peer will be choked by all peers which it was formerly downloading from
  - In such cases it will usually continue to get poor download rates until the optimistic unchoke finds better peers
- **Policy:** When over a minute goes by without getting a single sub-piece from a particular peer, do not upload to it except as an optimistic unchoke
- **Policy:** If choked by everyone, increase the number of simultaneous optimistic unchokes to more than one (an exception to the exactly one optimistic unchoke rule mentioned earlier), which causes download rates to recover much more quickly when they falter

Upload Only

- Once a peer is done downloading, it no longer has useful download rates to decide which peers to upload to
- The question is, which nodes to upload to?
- **Policy:** Upload to those with the best upload rate
- This ensures that pieces get replicated faster
- Also, peers that have good upload rates are probably not being served by others

Outline

1. BitTorrent System
   1. Torrent file
   2. Tracker
   3. Download protocol
2. Piece Selection
   1. Random first piece
   2. Rarest piece first
   3. Endgame mode
3. Choking Mechanism
4. BitTorrent Enhancement Proposals
   1. Multitracker metadata extension
   2. Peer exchange
   3. DHT protocol
5. Clustered BitTorrent Concept
6. Some statistics
7. Conclusions

BitTorrent Enhancement Proposals

- To overcome a single point of failure type of problem with one central tracker, **multitracker metadata extension** allows to define more than one tracker in the torrent file
- **UDP tracker protocol** is proposed to reduce the overhead in the tracker communication
- Peer discovery without tracker communication
  - **Peer exchange (PEX)** can be used to trade lists of other peers after two peers are connected
  - **DHT** is another way to discover peers without tracker communication
    - When DHT is enabled, BitTorrent client will connect to some bootstrap node and from there receive a list of nodes which are also DHT enabled
    - As a result, each DHT enabled peer becomes a **tracker** and other DHT enabled peers can use them to request addresses of peers when joining to some trackerless torrent
    - Magnet links
    - Key = 160bit infohash (SHA-1), Value = Peer list
    - Node-id = 160bit “random” value (same address space as keys)

Multitracker Metadata Extension

- In addition to the standard ‘announce’ key, a new key ‘announce-list’ will be used to describe a list of lists of tracker URLs
- If the client is compatible with the multitracker metadata extension, and if the ‘announce-list ‘key is present, the client will ignore the ‘announce’ key and only use the URLs in announce-list

```
<?xml version="1.0" encoding="UTF-8"?>
<tor:TORRENT
  xmlns:tor="http://azureus.sourceforge.net/files"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://azureus.sourceforge.net/files
            http://azureus.sourceforge.net/files/torrent.xsd">
 <ANNOUNCE_URL>http://torrent.ubuntu.com:6969/announce</ANNOUNCE_URL>
 <ANNOUNCE_LIST>
  <ANNOUNCE_ENTRY>
   <ANNOUNCE_URL>http://torrent.ubuntu.com:6969/announce</ANNOUNCE_URL>
  </ANNOUNCE_ENTRY>
  <ANNOUNCE_ENTRY>
   <ANNOUNCE_URL>http://ipv6.torrent.ubuntu.com:6969/announce</ANNOUNCE_URL>
  </ANNOUNCE_ENTRY>
 </ANNOUNCE_LIST>
 <COMMENT encoding="utf8">Ubuntu CD releases.ubuntu.com</COMMENT>
 <CREATION_DATE>1286702723</CREATION_DATE>
 <TORRENT_HASH>BCF2E587AFD4D3B1BDD8ECE5150D9FB4D2958AF4</TORRENT_HASH>
 <INFO>
  <NAME encoding="utf8">ubuntu-10.10-desktop-i386.iso</NAME>
  <PIECE_LENGTH>524288</PIECE_LENGTH>
  <LENGTH>726827008</LENGTH>
  <PIECES>
   <BYTES>2F7C363C74A83FCDE79FF41F62BEF2582A8DE138</BYTES>
   .
   .
```

UDP Tracker Protocol

- Binary protocol that does not require a complex parser and connection handling
  - Reduce the complexity of tracker code and increase the performance of the tracker
- Four types of messages
  - `connect request`, `connect response`, `announce request`, `announce response`
  - **connection_id** will be used to prevent IP spoofing
    - The tracker calculates a value and sends it to the client in the `connect response` message
    - The client will include this value to the `announce request` message and the tracker verifies the connection_id and ignores the request if the value does not match
- Since UDP is an unreliable protocol, application is responsible for retransmissions
  - If a response is not received after **15 * 2 ^ n** seconds, the client should retransmit the request (n starts at 0 and is increased up to 8 after every retransmission)
  - It is also necessary to re-request a connection_id when it has expired (validity time is 60s)

Peer Exchange

- PEX leverages the knowledge of peers that a user is connected to by asking them for the addresses of peers that they are connected to
  - Faster and more efficient than relying solely on a tracker and reduces the processing load on the tracker
- PEX cannot be used on its own to introduce a new peer to a file sharing session
  - To make initial contact, each peer must either connect to a tracker using a torrent file, or use a router computer (a DHT bootstrap node) to get a initial list of peers in the file sharing session
- PEX is typically implemented using one of two common extension protocols: **Azureus Extended Messaging Protocol** or **Extension Protocol**
  - Messages contain a group of peers to be added to the swarm and a group of peers to be removed

DHT Protocol

- BitTorrent uses DHT for storing peer contact information for trackerless torrents
- The DHT protocol is based on Kademlia and is implemented over UDP
- A **peer** is a client/server listening on a TCP port that implements the BitTorrent protocol
- A **node** is a client/server listening on a UDP port implementing the distributed hash table protocol
- Each node has a globally unique identifier known as the Node ID
  - Node IDs are chosen at random from the same 160 bit space as BitTorrent info hashes
  - XOR metric is used to compare two node IDs or a node ID and an info hash for mathematical closeness
  - Nodes must maintain a routing table containing the contact information for a small number of other nodes (finger table)
  - The routing table becomes more detailed as IDs get closer to the node's own ID > O(log n) lookup speed

DHT Protocol

- When a node wants to find peers for a torrent, it uses the distance metric to compare the info hash of the torrent with the IDs of the nodes in its own routing table
  - It then contacts the nodes it knows about with IDs closest to the info hash and asks them for the contact information of peers currently downloading the torrent
  - If a contacted node knows about peers for the torrent, the peer contact information is returned with the response
  - Otherwise, the contacted node must respond with the contact information of the nodes in its routing table that are closest to the info hash of the torrent
- The original node iteratively queries nodes that are closer to the target info hash until it cannot find any closer nodes
- After the search is exhausted, the client then inserts the peer contact information for itself onto the responding nodes with IDs closest to the info hash of the torrent

DHT Protocol

- Torrent file extension
  - A trackerless torrent dictionary does not have an **announce** key
    - Instead, a **nodes** key should be set to the K closest nodes in the torrent generating client's routing table
    - Alternatively, the key could be set to a known good node such as one operated by the person generating the torrent
- The KRPC protocol is a simple RPC mechanism consisting of bencoded dictionaries sent over UDP
  - One query, one response, no retry
- Three message types
  - `query`, `response`, `error`
- Four types of queries
  - `ping`, `find_node`, `get_peers`, `announce_peer`

Outline

1. BitTorrent System
    1. Torrent file
    2. Tracker
    3. Download protocol
2. Piece Selection
    1. Random first piece
    2. Rarest piece first
    3. Endgame mode
3. Choking Mechanism
4. BitTorrent Enhancement Proposals
    1. Multitracker metadata extension
    2. Peer exchange
    3. DHT protocol
5. Clustered BitTorrent Concept
6. Some statistics
7. Conclusions

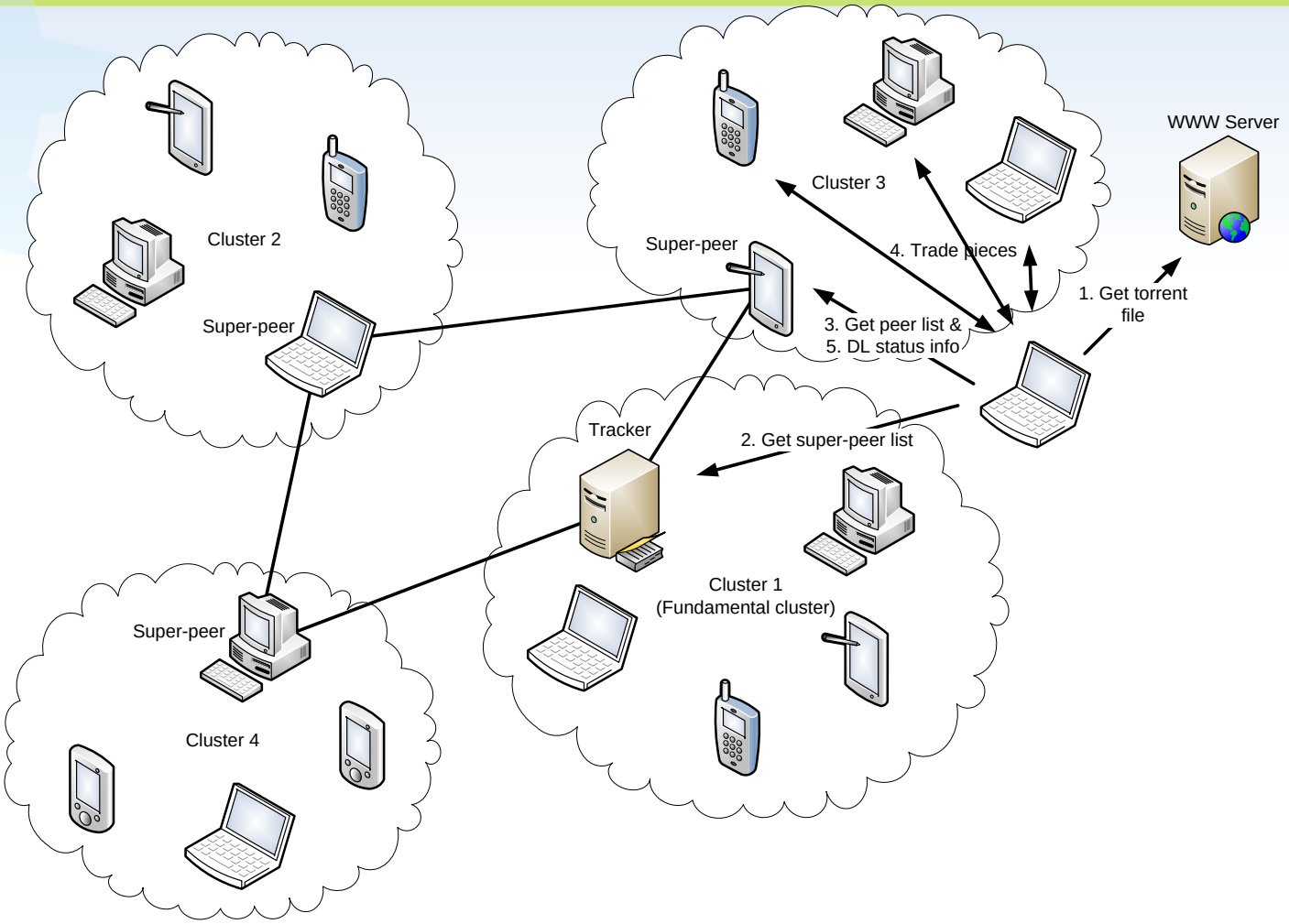
Clustered BitTorrent Concept

- To improve the file sharing performance, a hierarchical architecture to group peers into clusters according to their proximity in the underlying overlay network is proposed
  - Proximity is measured using RTT and TTL
  - Clusters are evenly distributed
    - Split and merge if needed
  - Peers in the cluster are relatively close to each other
- Tracker and three types of peers: seed, downloader (leecher) and new super-peer

Clustered BitTorrent Concept

- Tracker
  - Normal BitTorrent functionalities
  - Super-peer in the fundamental cluster
    - Fundamental cluster exists always
  - Monitor periodically the super-peer backbone network
- Super-peer
  - Cluster head
  - Collects and maintains state information of all peers in a cluster
- File download process
  1. Join to the cluster
     - If only fundamental cluster peer joins to it
     - Otherwise peer gets list of super-peers (including torrent tracker) from the tracker and joins the cluster which super-peer is nearest to it
  2. Exchange data with other peers in the cluster
     - Random peer list from the selected super-peer
     - Normal BitTorrent operation inside the cluster

Clustered BitTorrent Concept



Outline

1. BitTorrent System
    1. Torrent file
    2. Tracker
    3. Download protocol
2. Piece Selection
    1. Random first piece
    2. Rarest piece first
    3. Endgame mode
3. Choking Mechanism
4. BitTorrent Enhancement Proposals
    1. Multitracker metadata extension
    2. Peer exchange
    3. DHT protocol
5. Clustered BitTorrent Concept
6. Some statistics
7. Conclusions

Some statistics

- Who is using it:
  - Several content providers use bittorrent protocol for distribution
  - Software users:
    - Blizzard uses it to distribute content and patches
    - CCP uses it for EVE Online
  - Academic uses:
    - Florida State University shares large files among researchers
    - Academic Torrents to share large datasets
  - Clustered servers:
    - Facebook distributes patches among their servers using Bittorrent
    - Twitter similar
- How much is it consuming:
  - Estimated over 100 million users
  - Network bandwidth > Netflix (~35%)
  - Over 40% of traffic
  - Might cause issues with home routers (filling NAT tables fast)

Outline

1. BitTorrent System
    1. Torrent file
    2. Tracker
    3. Download protocol
2. Piece Selection
    1. Random first piece
    2. Rarest piece first
    3. Endgame mode
3. Choking Mechanism
4. BitTorrent Enhancement Proposals
    1. Multitracker metadata extension
    2. Peer exchange
    3. DHT protocol
5. Clustered BitTorrent Concept
6. Some statistics
7. Conclusions

Summary

- BitTorrent has almost gained de facto standard status when speaking about P2P file sharing
- BitTorrent has proven its strength for deploying large-scale P2P file delivery, but there are still scalability issues with the original approach
  - A single point of failure type of problem with one central tracker
  - Randomly selected peers among all peers in the network might create a long delay to the file sharing between two peers
- BitTorrent enhancement proposals improve the core protocol
- Clustered BitTorrent concept has been also proposed to improve the file sharing performance
  - Simulations have shown that the Clustered BitTorrent system can achieve faster download speed and higher file availability compared with the original system

Learning Outcomes

- Things to know
  - How BitTorrent works
    - Piece selection policies
    - Choking mechanism
  - Protocol enhancements

- Be ready for BitTorrent lab assignment

More Information

- B. Cohen, “Incentives Build Robustness in BitTorrent,” in Proceedings of the Workshop on Economics of Peer-to-Peer Systems (P2PECON2003), Jun. 2003, pp. 116–121.
- Di Wu, Prithula Dhungel, Xiaojun Hei, Chao Zhang and Keith W. Ross, "Understanding Peer Exchange in BitTorrent Systems,“ in Proceedings of the 10th IEEE International Conference on Peer-to-Peer Computing (IEEE P2P'10), Aug. 2010, pp. 1–8.
- R.L. Xia and J.K. Muppala, "A Survey of BitTorrent Performance," IEEE Communications Surveys & Tutorials, vol.12, no.2, pp.140–158, Second Quarter 2010.
- J. Yu and M. Li, “CBT: A proximity-aware peer clustering system in large-scale BitTorrent-like peer-to-peer networks,” Computer Communications, vol. 31, no. 3, pp. 591–602, Feb. 2008.
- The BitTorrent Community Forum
  - http://bittorrent.org/

Any questions?

mathieu.devos@tut.fi