FACULTY
OF INFORMATION
TECHNOLOGY

Monitoring of BitTorrent Traffic in LAN
PDS project
Academic year 2022/2023

Doc. Ing. Petr Matoušek, PhD., MA
Brno University of Technology, Faculty of Information Technology
Bozetechova 1/2, 612 66 Brno - Kralovo Pole
matousp@fit.vutbr.cz

BitTorrent Architecture

- Peer to peer (P2P) network for distributed sharing of resources, e.g., files.

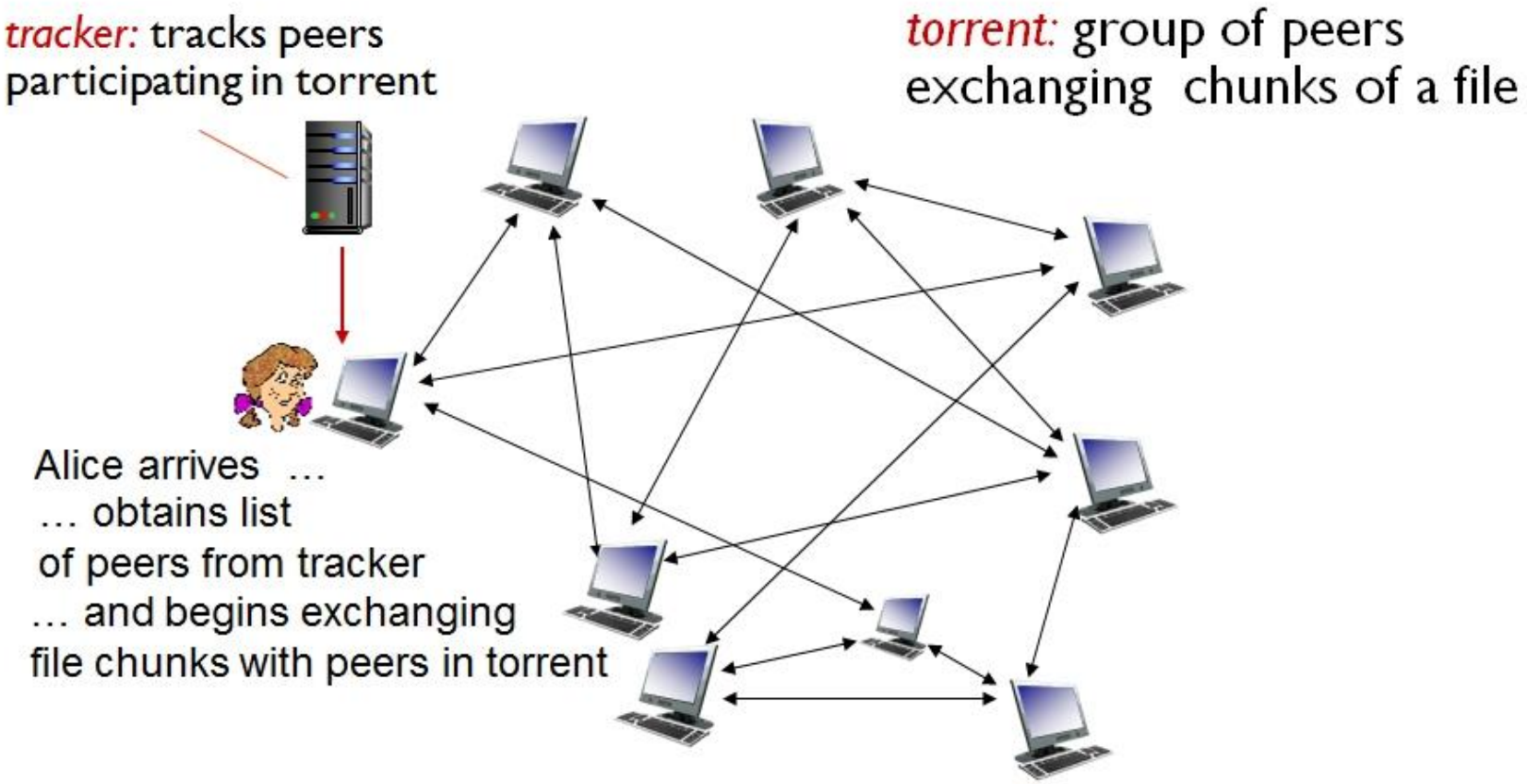
- *Peers* provide resources (bandwidth, memory) for sharing a file(s).
- *Trackers* register peers that distributes a specific file.
- *Torrent* = a set of nodes participating in distribution of a particular file.
- A resource (file) identified by a *info_hash* in the *metainfo* file:

```
{ ”announce ”: ”http://bttracker.debian.org:6969/announce”,     # reference to the tracker that keeps a list of
                                                                  active peers for the  torrent
  ”info”: {
    ”length”: 678301696,                                        # length of the file
    ”name”: ”debian-503-amd64-CD-1.iso”,                        # file name
    ”piece length”: 262144,                                     # size of the chunk
    ”pieces”: <binary SHA1 hashes> }                            # a list of chunks (hashes)
}
```

- Communication in BitTorrent is implemented using *trackers* and/or *distributed hash table* (DHT).

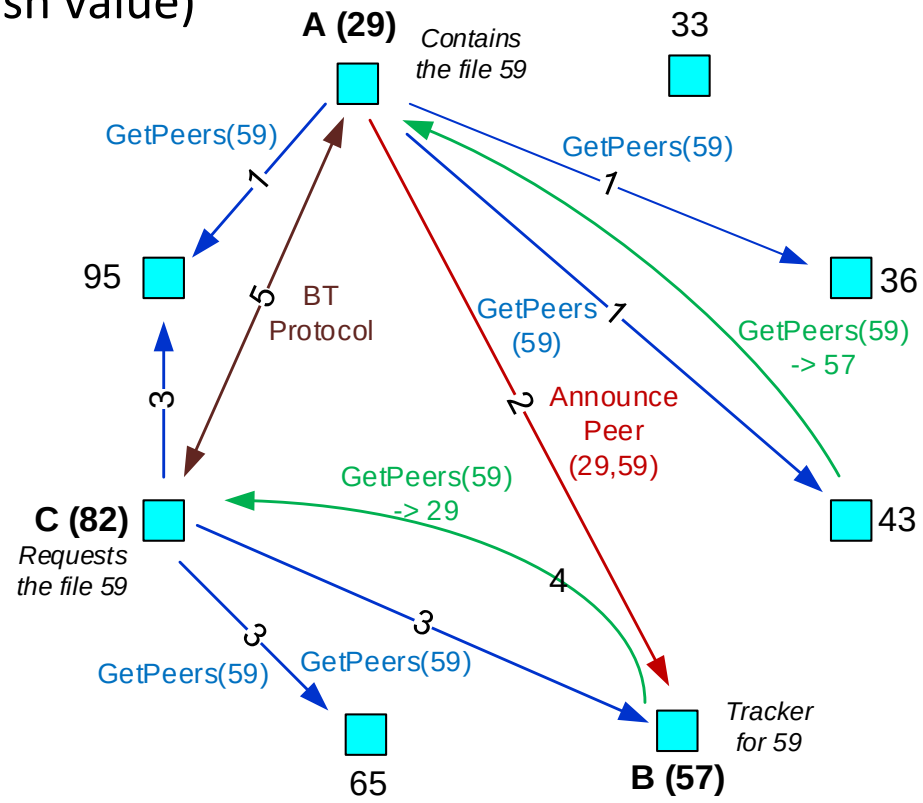
BitTorrent Communication using Trackers

- Example of communication [1]
- Communication implemented using HTTP and BitTorrent protocol



BitTorrent Communication using DHT [2]

- Decentralized systems based on Kademlia Protocol (BT-DHT)
- Each peer maintains its own routing table of neighbor peers
- The network saves a file to a node with *Node ID* closest to the *File ID* (160 bits hash value)



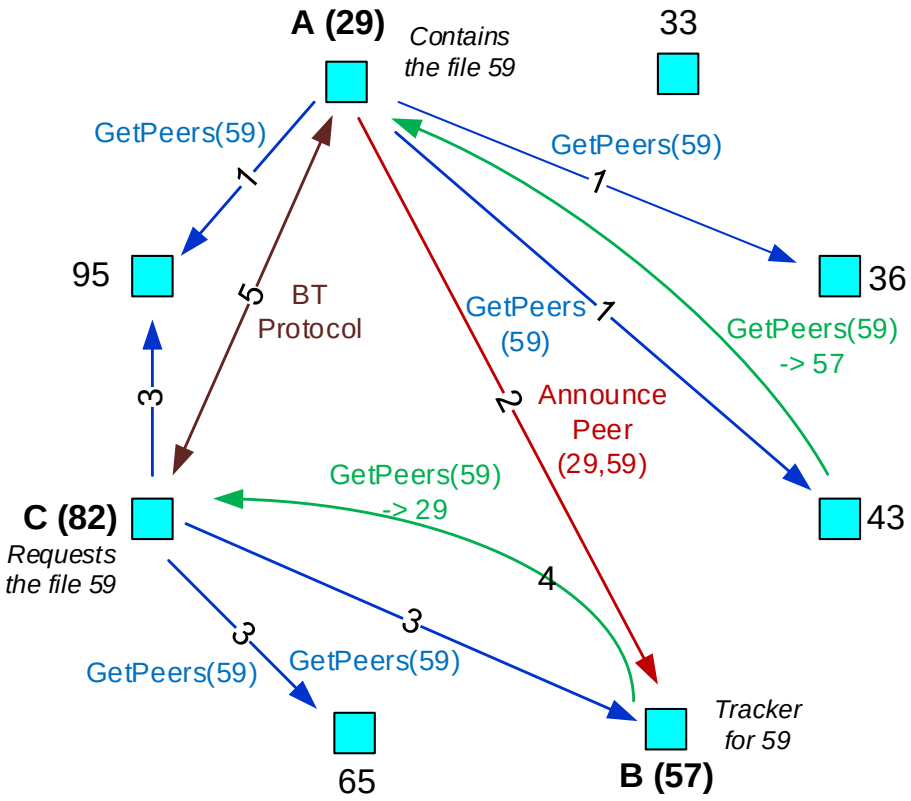
BT-DHT Commands:

- Ping
- Find_node
- Get_peers
- Announce_peers

BitTorrent for File Transfer:

- Handshake
- Have
- Request
- Piece

BitTorrent Communication using DHT [2] – example



Node A (ID=29) contains a file with ID=59.

Node C (ID=82) requests such a file.

1. Node A finds a nearest node to file *x* using GET_PEERS. The nearest is B.
2. Node A informs B that has file *x* using ANNOUNCE_PEER. B keeps this information.
3. Node C requests file *x* (ID=59) in his neighbors.
4. Node B sends a list of a nodes having file *x*, i.e., node A.
5. Node C downloads a file using BT protocol.

PDS Project

Goal: Implement detection and monitoring of BitTorrent traffic in LAN.

- The project can be conducted through the following steps:
  1. Study BitTorrent architecture and communication including protocols BitTorrent, BT-DHT and uTP. Focus on Mainline DHT implementation, see References.
  2. Install a BitTorrent client, e.g., BitTorrent Classic (Win), qBittorrent (Win, MacOS, Linux).
  3. Provide experiments with the client. Analyze the following communication:
     a) How an initial connection to the P2P network is established using bootstrap servers?
     b) To what peers the node is connected using keep-alive monitoring (GetPeers)?
     c) How a resource is found and downloaded?
     d) Based on observed communication, compute the routing table of a BT client (optional).
  4. Create a tool that monitors and analyzes BitTorrent communication (PCAP).
     a) Detect BitTorrent initialization -> provide a list of IP addresses + ports.
     b) Detect active BitTorrents peers -> a list of peers, node_id, IP+port, # of connections.
     c) Detect file transmission(s) -> info_hash, contributors, length, duration.
  5. Write the project report (see the recommended structure below).
  6. Submit the project (source code + PCAP files + document) via BUT IS.

PDS Project

Project: Monitoring of BitTorrent Traffic in LAN.

- Project deadline: 24th April 2023 (hard deadline).
- Individual registration to the project required using BUT IS until 28th Feb.
- Maximum points: 25
- Online consultations available using Moodle News.
- Individual project – each student creates its own solution.
- A partial solution is accepted. Not-implemented parts must be stated in Readme.txt.
- Required outcomes:
  1. Monitoring tool *bt-monitor* -> running as CLI application on a Unix system.
  2. PCAP file with captured BT communication.
  3. Report in PDF format with the required structure.
- Plagiarism prohibited – see copyrights and the publication policy.

1) Monitor BitTorrent client communication

1. Install a BitTorrent client on your system or a virtual system.
2. Capture communication using Wireshark/tshark or tcpdump.
3. Analyze communication of BitTorrent client.
   a) Find out how a client is connected to the BitTorrent networks – analyze DNS or/and HTTP communication with bootstrap servers, observe how IP addresses and ports to BitTorrent network are open.
   b) Observe how client is initialized and find neighbors using Find_Node and Get_Peers requests and responses.
4. Download a file using a BitTorrent client.
   a) Observe file ID (info hash) and peers where the file is downloaded from.
   b) Find out how file chunks are downloaded from what peers.
5. Describe your findings in the report.
6. Propose a method how to detect above mentioned activities by monitoring network traffic (using Wireshark/tshark/tcpdump).

2) Create a tool for detection of BT communication

1. Use your preferable programming language (e.g., Python, C, Java).
2. Tool *bt-monitor* will be running in the Unix command line (CLI).
3. The tool reads PCAP files or CSV files and detects BT communication.
   - In case of using CSV, please, provide a script that converts a PCAP to CSV.
4. PCAP files can be preprocessed using tshark first.

   tshark -r file.pcapng -T fields -E separator=";" -d udp.port==47222,bt-dht -e frame.time_relative -e ip.src -e ip.dst -e bt-dht.ip -e bt-dht.port -e bt-dht.bencoded.string "bt-dht“ > file.csv, see Wireshark Display Filters.
5. Tool has the following syntax (obligatory format):

   bt-monitor -pcap <file.pcap>|-csv <file.csv> -init | -peers | -download

   <file.pcap>: input PCAP file or <file.csv> input CSV file

   -init: returns a list of detected bootstrap nodes (IP, port)

   -peers: returns a list of detected neighbors (IP, port, node ID, # of conn)

   - download: returns file info_hash, size, chunks, contributes (IP+port)

   - rtable: returns the routing table of the client (node IDs, IP, ports)
6. *Readme.txt ->* describes how to compile and run the tool + required libraries

2) Create a tool for detection of BT communication

How to detect BT communication?

- IP address and/or port analysis -> well-known trackers, port numbers, the same ports used for both TCP/UDP connection.
- Deep Packet Inspection (DPI): HTTP commands: GET /announce, GET /scrape, BitTorrent handshake: “0x13BitTorrent protocol”, DHT commands: ping, find_nodes, get_peers, announce_peers in bencoding.
- Statistical features: no. of connections among nodes, directions, size of transmitted data in each direction, frequency of opening new connections.
- DNS analysis: A requests for domain names with “torrent”
- Time features: flow inter-arrival time, duration.
- Machine-learning approach: K-means clustering.
- See [6-9] for inspiration.

3) Write the report (5-10 pages)

Recommend document structure:

1. Description of BitTorrent architecture and communication.
2. Your experiments with BitTorrent client and findings.
3. Description of detection method(s) for BitTorrent traffic.
4. Description of implemented application, i.e., how application operates, how is compiled, what parameters expects, how can be run.
5. Testing of the application on created datasets, results, evaluation.
6. Discussion of the results.
7. Conclusion and contribution.

For creating a document, use BSc/MSc template, see https://www.fit.vut.cz/study/theses/bachelor-theses/.

4) Project submission

1. Submit a zip file with name *xlogin.zip* that includes following files:
   - Readme.txt – with your name, login, a list of files in the ZIP archive, description how your tool can be run.
   - The project report in PDF format (file *xlogin.pdf*).
   - Source code of your tool *bt-monitor* you developed with a possible preprocessing script.
   - Input data that you captured and used for testing.

Optional extension (extra points)

- Analyze BT communication regarding client initialization and write down the clients local routing table: *bt-monitor -pcap <file.pcap> -rtable*
- Extension is considered only if a basic functionality is working

Concluding remarks

- The goal of the project is to present your ability to analyze behavior of a network application based on networking data.
- Focus is on individual reasoning, data processing and analysis.
- The project includes
  - experiment part,
  - analysis part,
  - proposal of a detection method,
  - implementation part,
  - testing, evaluation, and documentation.
- Innovative approach in any of these parts is highly appreciated.

- Any external tools, code, sources of information must be properly referenced, otherwise the work is considered as plagiarism.

References

[1] James F. Kurose and Keith W. Ross. Computer Networking: *A Top-Down Approach Featuring the Internet*. Addison-Wesley, 6th edition, 2012.

[2] L. Wang and J. Kangasharju. *Measuring large-scale distributed systems: case of BitTorrent Mainline DHT*. In IEEE P2P 2013 Proceedings, pages 1-10, Sept 2013.

[3] Jahn Arne Johnsen, Lars Erik Karlsen, Sebjørn Sæther Birkeland: *Peer-to-peer networking with BitTorrent*, 2005, Available at https://web.cs.ucla.edu/classes/cs217/05BitTorrent.pdf.

[3] Cohen, B. *The BitTorrent Protocol Specification* [online]. Available at https://www.bittorrent.org/beps/bep_0003.html.

[4] Loewenstern, A. a Norberg, A. *DHT Protocol* [online]. Available at https://www.bittorrent.org/beps/bep_0005.html.

[5] Norberg, A. *uTorrent transport protocol* [online]. Available at: https://www.bittorrent.org/beps/bep_0029.html.

[6] Florek Daniel: *Detekce provozu protokolu BitTorrent*, BP FIT VUT v Brně, 2018.

[7] Richard Wanner: *Detecting Torrents Using Snort*, SANS Whitepaper, 2009.

[8] Raymond Wong: *BitTorrent Traffic Detection with Deep Packet Inspection and Deep Flow Inspection*, Master Thesis, 2011, San Jose State University

[9] Kemp: *Detecting BitTorrent using Flowmon ADS*, 2022, available online at https://demo.flowmon.com/doc/adsplug/?file=47221124.html

[10] Examples of BitTorrent communication at: https://cshark.fit.vutbr.cz/captures/0182830e8bfb, https://cshark.fit.vutbr.cz/captures/a72a02e192fc.