

# A Peer-to-Peer Swarm Creation and Management Framework

Adriana Drăghici, Marius Sandu-Popa, **Răzvan Deaconescu**,  
Nicolae Țăpuș

1st Workshop of Software Services

September 23-26, 2010, Timișoara, Romania

- 1 Context
- 2 Swarm Management Framework
- 3 Usage
- 4 Conclusion
- 5 Questions

1 Context

2 Swarm Management Framework

3 Usage

4 Conclusion

5 Questions

# Peer-to-Peer and BitTorrent

- P2P traffic dominating current Internet traffic (more than 50%)
- BitTorrent – number one protocol in the Internet
- “recent” research interests: streaming, social networking, trackerless swarms, reputation, privacy

# BitTorrent

- peer (seeder, leecher)
- torrent file (metafile)
- tracker
- swarm
- optimistic unchoking (tit-for-tat)
- rarest piece first

# Measurements and Simulations in Peer-to-Peer Systems

- simulation-based
  - simulators
  - + scalability, flexibility
  - - realism, BitTorrent swarms
- real-world environments
  - tracker log files, probing
  - PlanetLab
  - compromise between scalability and relevant information
  - + realism

1 Context

**2 Swarm Management Framework**

3 Usage

4 Conclusion

5 Questions

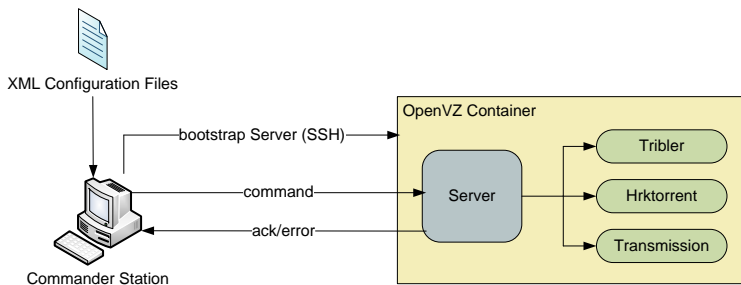
# Design Goals

- BitTorrent clients behaviour and protocol analysis
- automation
- complete control
- full client information
- flexibility

# Why?

- lack of a BitTorrent scenario deployment framework
- tool for BitTorrent experimenters
- complete information for subsequent analysis

# Architectural Overview



# Architectural Overview (2)

- Commander Station
  - experimenter
  - automation/swarm control
  - client status and swarm “logic”
- “hardware” node – OpenVZ container
- “dumb” BitTorrent client interaction service
- (instrumented) BitTorrent supported clients

# Configuration Files

- swarm configuration file
  - metafile (.torrent file)
  - limitations
  - verbosity
  - churning
- nodes configuration file
  - IP address
  - listening ports
  - SSH credentials
  - local paths

# Communication Protocol

- one BitTorrent client for each “hardware” node
- commands from Commander Station to BitTorrent client through “hardware” node server
- start/stop client, get list of running clients
- archive – archive client logging/status file
- status – monitor client state
- cleanup – remove log files

# Implementation Details

- Python
- XML
- client instrumentation as required
- current support for libtorrent-rasterbar, Tribler, Transmission

- 1 Context
- 2 Swarm Management Framework
- 3 Usage**
- 4 Conclusion
- 5 Questions

# Physical Setup

- OpenVZ based containers
- UPB NCIT cluster
- 10 identical commodity hardware systems (x10 OpenVZ containers)
- iptables/tc
- SSH (Cluster SSH, Parallel SSH)
- NFS

# Use Case

- setup nodes/swarm configuration files
- start commander
- start swarm
- wait for scenario completion
- collect information
- parse, analyse, disseminate

- 1 Context
- 2 Swarm Management Framework
- 3 Usage
- 4 Conclusion**
- 5 Questions

# Conclusion and Further Work

- automation, full control, extensible, client information
- additional BitTorrent clients (NextShare, Vuze)
- scheduling facility
- web interface
- deployment on PlanetLab
- `http://koala.cs.pub.ro/git/?p=cs-p2p-next.git;a=tree;f=ppf;hb=HEAD`

# Acknowledgements

- Alex Herişanu (NCIT cluster – University Politehnica of Bucharest)
- P2P-Next team
- Tribler team (TU Delft)
- Arvid Norberg (libtorrent-rasterbar)
- Henrik Friedrichsen (hrktorrent)

- 1 Context
- 2 Swarm Management Framework
- 3 Usage
- 4 Conclusion
- 5 Questions**