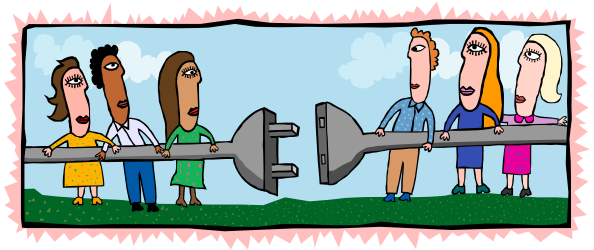
The PPSP Peer Protocol (PPSPP)

Johan Pouwelse, Arno Bakker, Riccardo Petrocco

Delft University of Technology

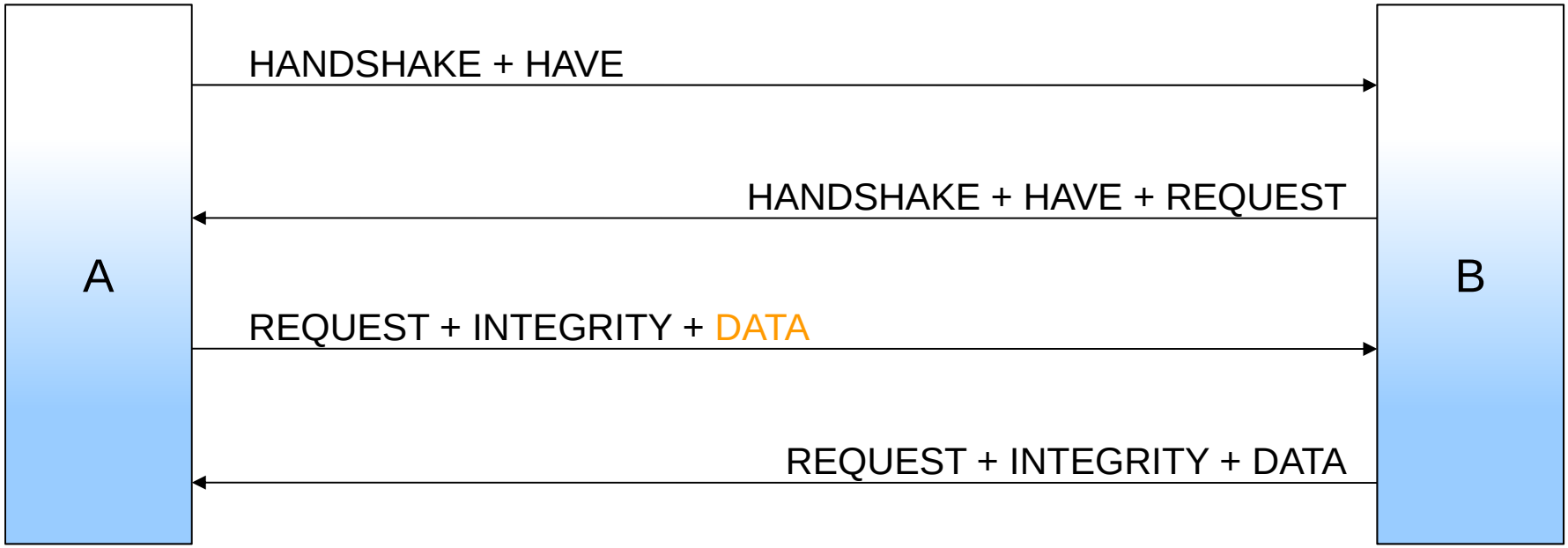
Refresh: PPSPP messages

- Basic unit of communication: Message
  - HANDSHAKE
  - HAVE: convey chunk availability
  - REQUEST: request chunks
  - DATA: actual chunk
  - INTEGRITY: hashes to enable integrity verification
  - …
- Messages are multiplexed together when sent over the wire.



Example PPSPP on the wire

- Peer A and B both have some chunks



- Note: low latency, data transfer already in 3rd datagram.

Changes in -03: Messages

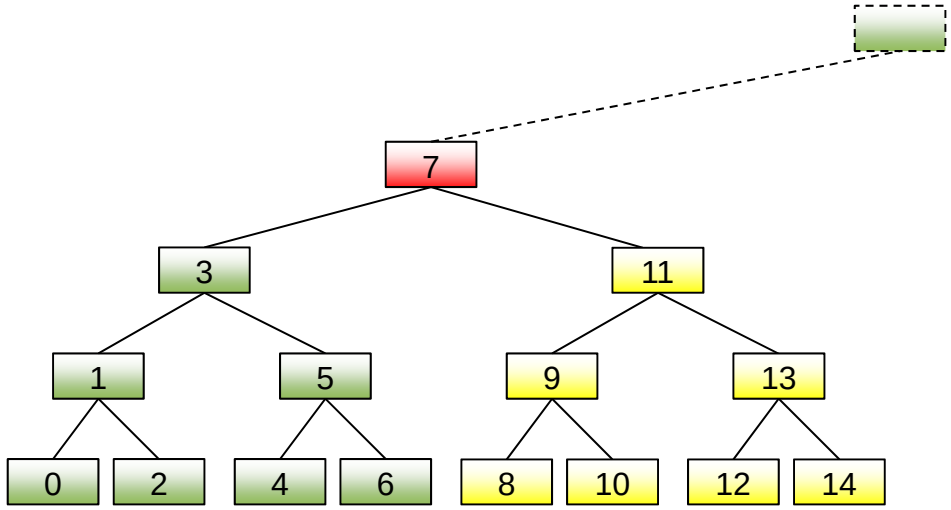
- Resolved Ticket #4: Added explicit CHOKE/UNCHOKE messages:
  - Signal to peer whether it is allowed to download from you
- Resolved Ticket #22, #23, #28: Failure behavior, error codes and dealing with peer crashes:
  - Classify peers as either good (i.e., responding) or bad.
  - Only use the good ones.
  - Handles:
    - slow,
    - crashed, and
    - (silent) malicious peers.

Changes in -03: Chunk Addressing

- Resolved Ticket #13: Chunk ranges are the default chunk addressing scheme that all peers MUST support:
  - REQUEST(start chunk ID, end chunk ID)
  - Bins + byte ranges optional.
- Added a section on compatibility between chunk addressing schemes:
  - Compatible when fixed size.

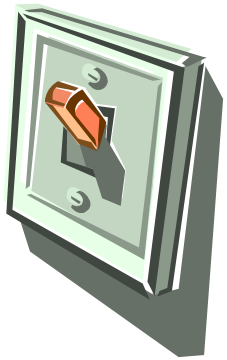
Changes in -03: Live

- Explained Unified Merkle Trees as a method for content integrity protection for live streams.
- Added a section on forgetting chunks in live streaming.



Changes in -03: Protocol Options

- Added "End" option to HANDSHAKE protocol options.
- Added SHA-2 support for Merkle Hash functions.
- Added content integrity protection methods for live streaming to the relevant protocol option:
  - Sign All
  - Unified Merkle Tree
- Added a Live Signature Algorithm protocol option.



Changes in -03: Transport

- Resolved Ticket #24+27: UDP + congestion control chosen as transport:
  - TCP and RTP encapsulations have been removed.
- Resolved Ticket #21+25: LEDBAT congestion control chosen:
  - DATA and ACK messages now contain LEDBAT time fields.
  - If other congestion control algorithms (cf. RMCAT WG) a protocol option will be added.

Changes since -03: Text clean up

- Updated Abstract and Introduction, removing download case.
- Removed directory lists unused in streaming.
- Superfluous parts on Extensibility have been removed.
- Removed appendix with Rationale.

Summary

- Peer protocol now complete for common use cases
- Security analysis complete
- Proposal:
  - Start Working Group Last Call

Demos

https://play.google.com/store/apps/developer?id=TUDelft:+Delft+University+of+Technology

Live Streaming Demo

- From Android:

H.264 Encoder → PPSP Engine →

- To Android:

→ PPSP Engine → VLC →

- No server infrastructure!

Mobile Video Distribution Demo

- Person captures video
- Posts http://ppsp.me/roothash on Twitter
- Follower clicks ppsp link; PPSP player launched
- Peers for roothash looked up via DHT
- Follower plays + seeds video
- Automatic seeding by scraping ppsp links supported

Extra Slides

Mobile Distribution Demo: Boosting

- Booster searches Twitter using official search API:
  - Normal HTTP GET with no authentication
  - JSON results
- Ask for new tweets only
- Search every 10 s
- Extract roothash from Tweet Entities
- Lookup peers in DHT
- Attempt download and seed.

PPSPP Implementation

Arno Bakker

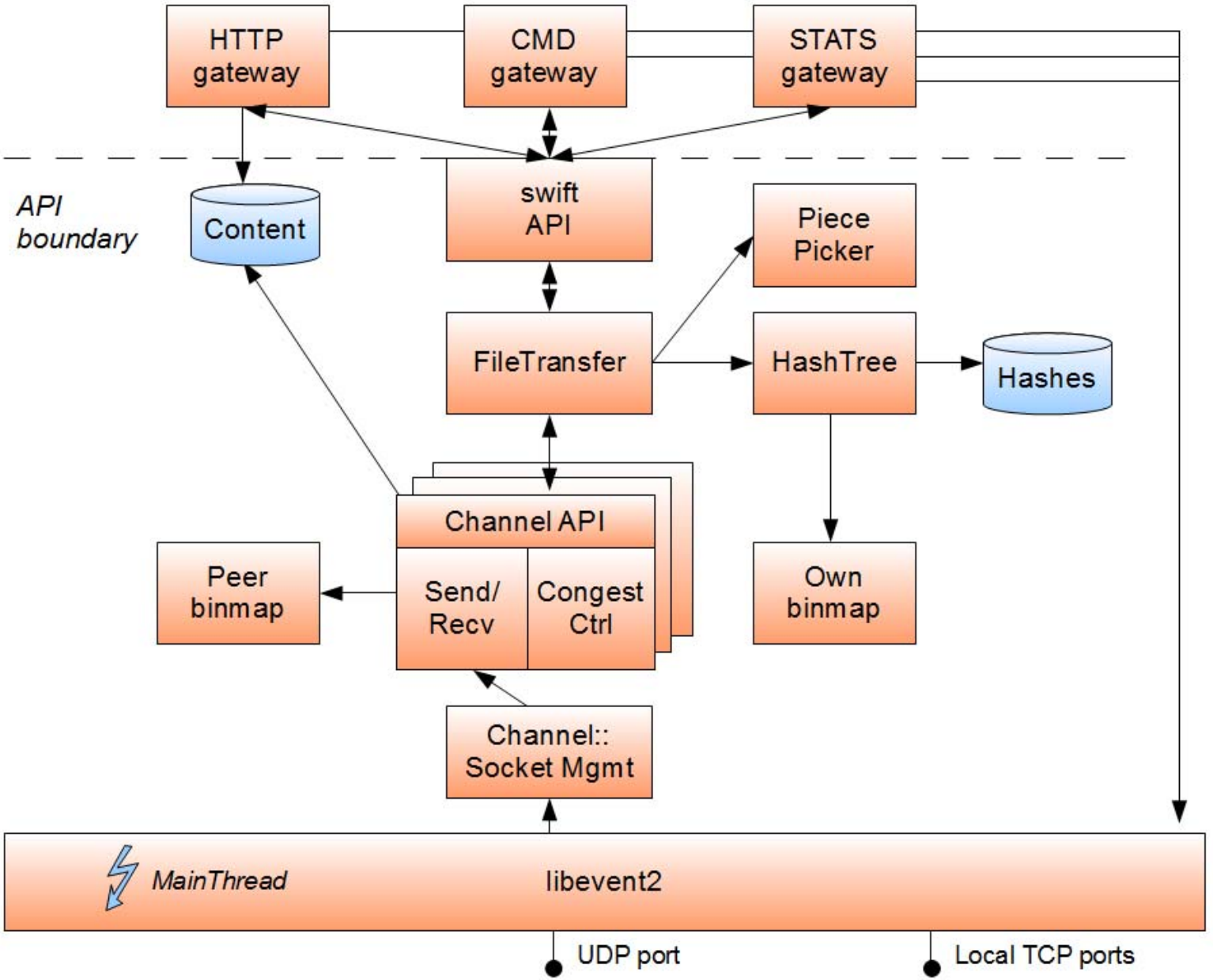
Riccardo Petrocco

Richard Marsh

et al.

Introduction

- Swift implemented in C++
- Libevent2 library for socket communication
- UDP
  - + Multiplexing: Many swarms on same socket
  - + IETF LEDBAT congestion control
- Video-on-demand + live prototype
- Source code:
  - www.libswift.org (GitHub)
  - LGPL License


HTTP gateway
CMD gateway
STATS gateway
API boundary
Content
swift API
Piece Picker
FileTransfer
HashTree
Hashes
Channel API
Peer binmap
Send/ Recv
Congest Ctrl
Own binmap
Channel:: Socket Mgmt
MainThread
libevent2
UDP port
Local TCP ports

Summary

- More info, sources, binaries:
  - www.libswift.org
- Acknowledgements
  - European Community's Seventh Framework Programme in the P2P-Next project under grant agreement no 216217.

Questions?

Arno Bakker (arno@cs.vu.nl)

Riccardo Petrocco <r.petrocco@gmail.com>

Johan Pouwelse (peer2peer@gmail.com)