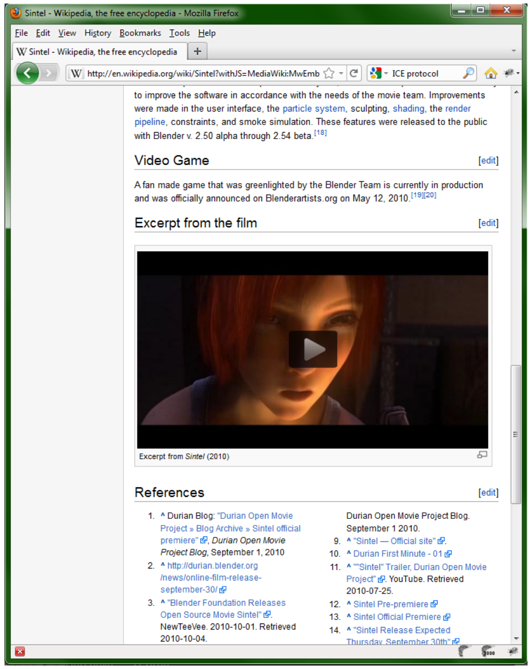
The Swift Multiparty Transport Protocol As PPSP

Arno Bakker, Victor Grischenko, Johan Pouwelse

P2P-Next / Delft University of Technology

Status

- Implemented in C++
  - Video-on-demand over UDP
- Running in Firefox:
  - <video src="swift://…
  - Via 100 KB plugin
  - Hooks on en.wikipedia.org
- Running on:
  - iPad
  - Android
  - set-top box
- Works with P2P caches


Sintel - Wikipedia, the free encyclopedia - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Sintel - Wikipedia, the free encyclopedia
http://en.wikipedia.org/wiki/Sintel?withJS=MediaWiki:MwEmb
ICE protocol

to improve the software in accordance with the needs of the movie team. Improvements were made in the user interface, the particle system, sculpting, shading, the render pipeline, constraints, and smoke simulation. These features were released to the public with Blender v. 2.50 alpha through 2.54 beta.[18]

Video Game [edit]

A fan made game that was greenlighted by the Blender Team is currently in production and was officially announced on Blenderartists.org on May 12, 2010.[19][20]

Excerpt from the film [edit]

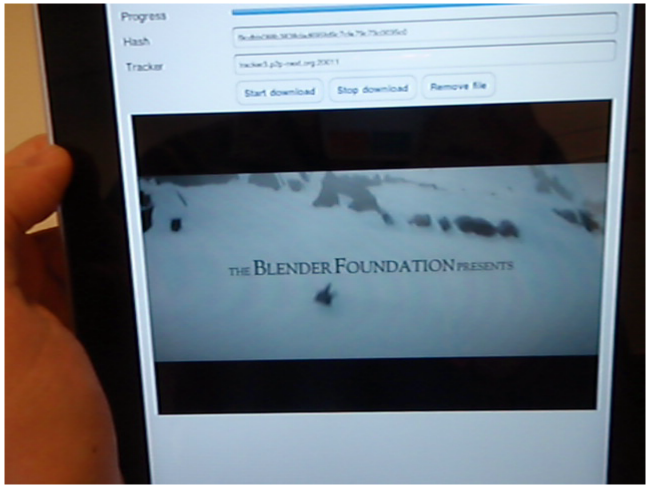
Excerpt from *Sintel* (2010)

References [edit]

1. ^ Durian Blog: "Durian Open Movie Project » Blog Archive » Sintel official premiere", *Durian Open Movie Project Blog*, September 1, 2010
2. ^ http://durian.blender.org/news/online-film-release-september-30/
3. ^ "Blender Foundation Releases Open Source Movie Sintel". NewTeeVee. 2010-10-01. Retrieved 2010-10-04.

Durian Open Movie Project Blog. September 1 2010.

9. ^ "Sintel — Official site".
10. ^ Durian First Minute - 01
11. ^ ""Sintel" Trailer, Durian Open Movie Project". YouTube. Retrieved 2010-07-25.
12. ^ Sintel Pre-premiere
13. ^ Sintel Official Premiere
14. ^ "Sintel Release Expected Thursday, September 30th"

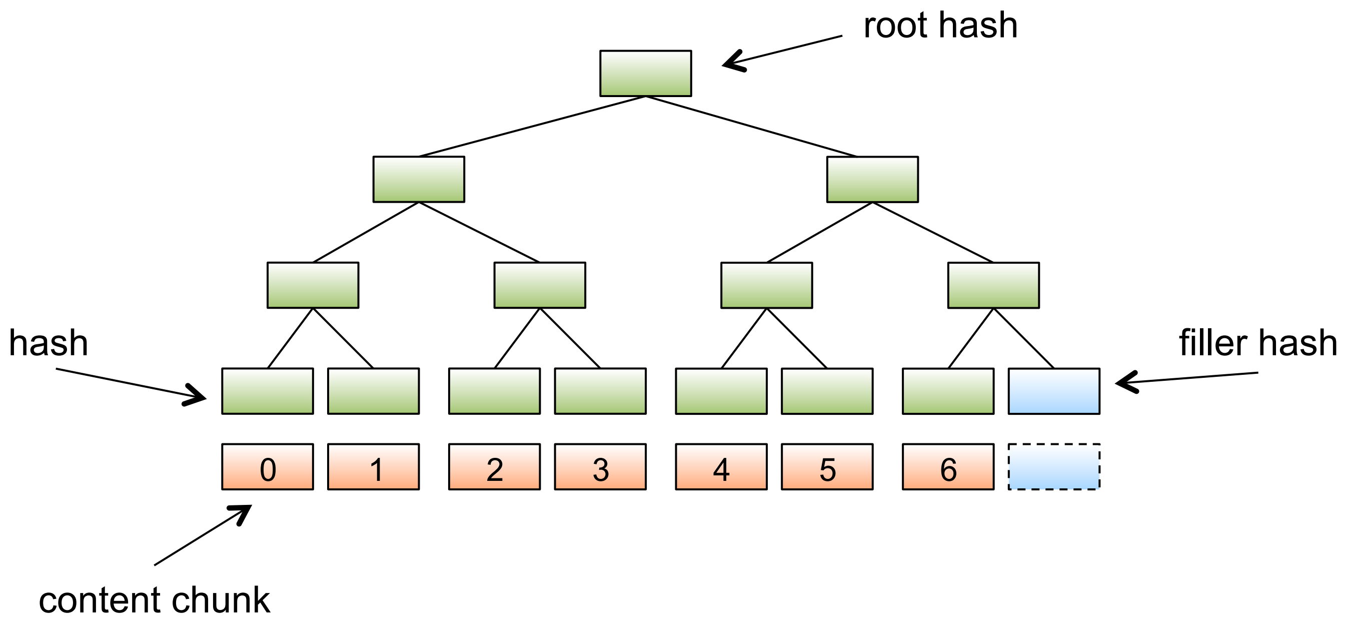


Swift design goals

1. Kernel-ready, low footprint
2. Generic protocol that covers 3 use cases (dl, vod, live)
3. Have short prebuffering times
4. Traverse NATs transparently
5. Be extensible:
   - Different congestion control algorithms (LEDBAT)
   - Different reciprocity algorithms (tit4tat, Give-to-Get)
   - Different peer-discovery schemes

Swift metadata

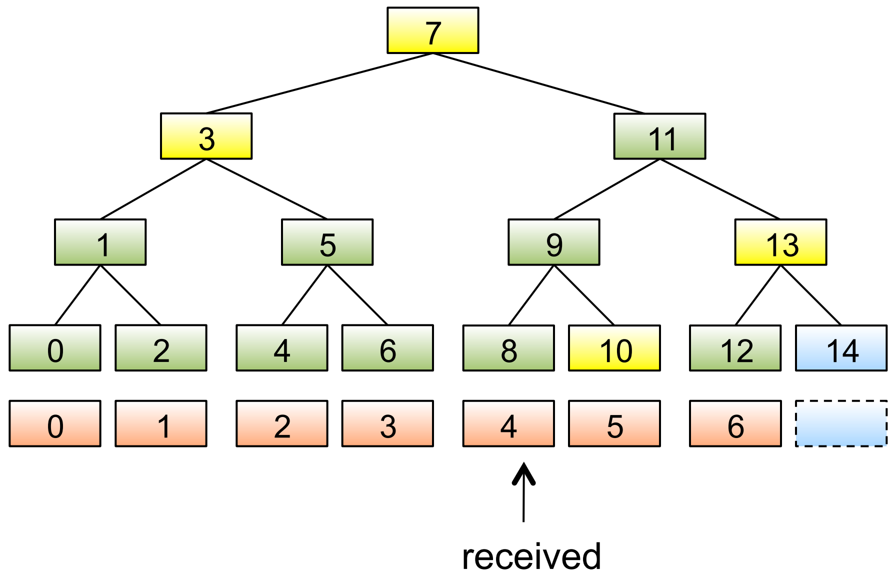
- Content identified by single root hash
- Root hash is top hash in a Merkle hash tree



- Information-centric addressing: small enough for URLs

Swift integrity checking

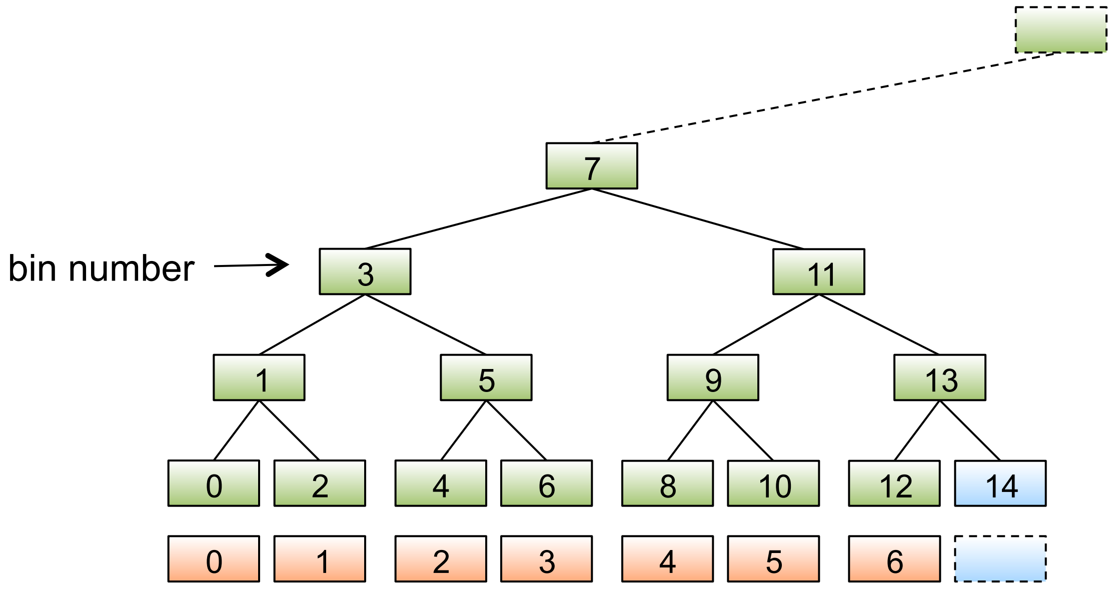
- Atomic datagram principle:
  - Transmit chunk with uncle hashes
  - Allows independent verification of each datagram



  - Protection against malicious peers

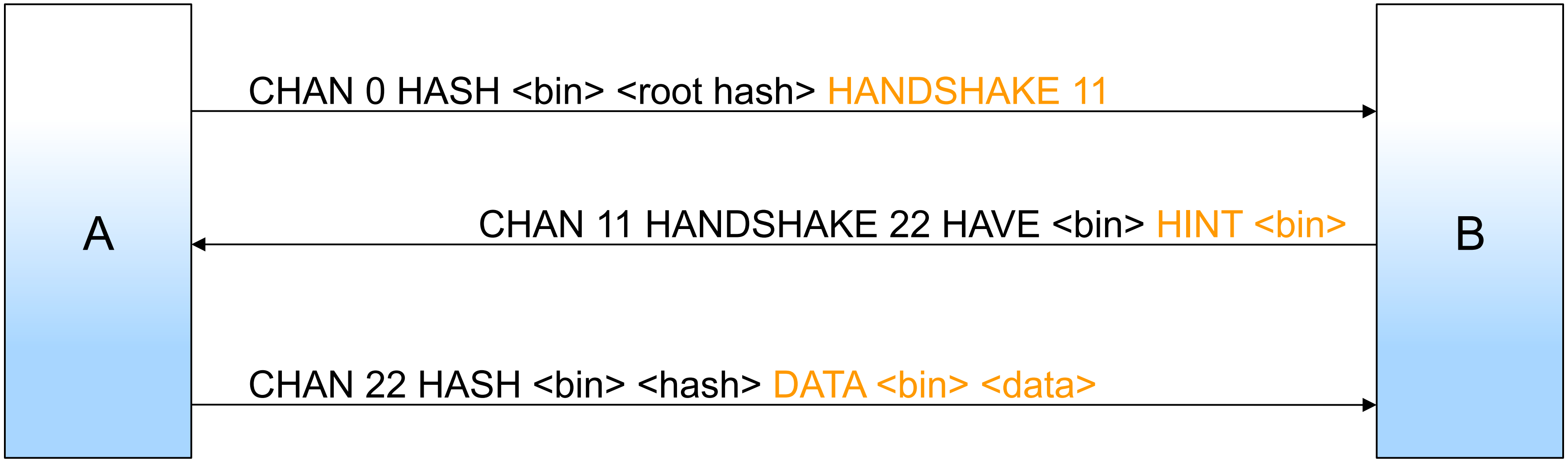
Swift chunk IDs and live trees

- Nodes in tree denote chunk ranges: bins
  - Used for scalable acknowledgements + low footprint
- Dynamically growing & pruned trees for live



Swift wire format

- Datagram consists of channel ID + multiple messages
- Message is fixed length, first byte message ID
- E.g.


A
B
CHAN 0 HASH <bin> <root hash> HANDSHAKE 11
CHAN 11 HANDSHAKE 22 HAVE <bin> HINT <bin>
CHAN 22 HASH <bin> <hash> DATA <bin> <data>

- Data after 1 roundtrip -> short prebuffering times

PPSP Basic Requirements

√ = *Done*
√ = *Some work needed*

REQ-1	√	PEX message as basis for tracker proto
REQ-2	√	Extra protection may be needed for RT P2P
REQ-3	√	Peer ID is open, self-certification proposed
REQ-4	√	Swarm ID is root hash or public key
REQ-5	√	Chunk is 1K, or variable
REQ-6	√	Chunk ID is bin number
REQ-7	√	Carrier can be UDP or TCP, RTP or HTTP
REQ-8	√	Protocol is extensible for QoS info

See draft and PPSP materials

PPSP Peer Protocol Requirements

PP.REQ-1	√	HAVE message+GET_HAVE if push insufficient
PP.REQ-2	√	HAVE message are bidirectional
PP.REQ-3	√	PEX message + GET_PEX if push insufficient
PP.REQ-4	√	HAVE message for updates
PP.REQ-5	√	Protocol is extensible for status info
PP.REQ-6	√	Transmission and chunk requests integrated

See draft and PPSP materials

PPSP Security Requirements

SEC.REQ-1	√	P2P-Next Closed Swarms design suitable
SEC.REQ-2	√	Inherit from carrier proto, think of caching
SEC.REQ-3	√	Compatible with existing solutions
SEC.REQ-4	√	Merkle tree limits propagation bad content
SEC.REQ-5	√	Peer ignores bad senders
SEC.REQ-6	√	Secure tracking against injector Eclipse
SEC.REQ-7	√	Enabled by PEX or DHT with self-certification
SEC.REQ-8	√	Merkle tree is founded on BitTorrent hashing
SEC.REQ-9	√	Detection easy, reporting hard

See draft and PPSP materials

Relationship to other IETF work

- LEDBAT
  - Implemented
- ALTO
  - Integration possible
- DECADE
  - Swift designed for in-network caches
- draft-dannewitz-ppsp-secure-naming-02
  - Orthogonal, sign root hashes
- NAT traversal
  - Orthogonal

airspeed velocity
$\sqrt{5\left[\left(\frac{Q_c}{P_o}+1\right)^{\frac{2}{7}}-1\right]}$
unladen swallow

Summary

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

Questions?

Arno Bakker (arno@cs.vu.nl)

Johan Pouwelse (peer2peer@gmail.com)

Swift over RTP

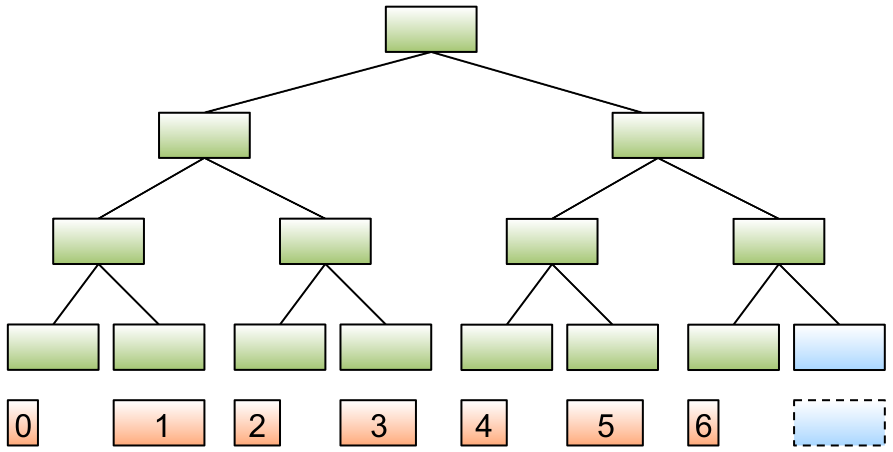
- RTP packet

V P X CC M PT	Sequence Number
Timestamp	
SSRC Identifier	
Extension ID	Extension header length
HINT+HAVE+HASHES	
DATA	

- Problem: Header fields not protected

RTP over Swift

- Carry RTP packet as chunk over Swift
- Header protected
- Merkle tree can handle variable-sized chunks



Swift over HTTP

GET /7c462ad1d980ba44ab4b819e29004eb0bf6e6d5f HTTP/1.1

Host: peer481.example.com

Range: bins 11 <- "I want bin 11"

Accept-Ranges: bins 3 <- "I have bin 3"

…

HTTP/1.1 206 Partial Content

Content-Range: bins 8

Content-Merkle: (10,*hash10*),(13,*hash13*) ;h=SHA1;b=1K <- hashes

Accept-Ranges: bins 7 <- "seeder"

…

Chunk 8

The Internet today

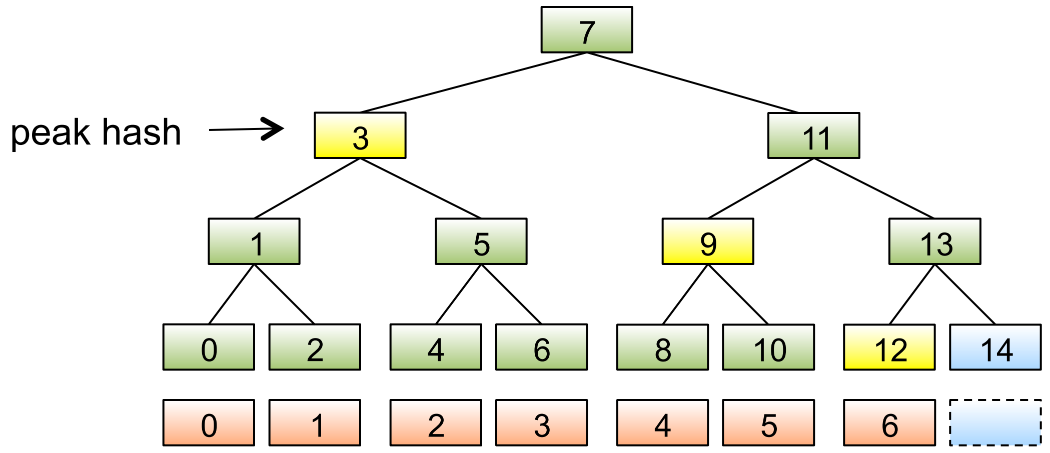
- Dominant traffic is content dissemination:
  - One-to-many
    - Download (ftp)
    - Video-on-demand (YouTube)
    - Live (Akamai, Octoshape, PPLive)
- Dominant protocol was designed for one-to-one:
  - TCP

What's wrong with TCP?

- TCP's functionality not crucial for content dissemination:
  - Don't need Reliable delivery
  - Don't need In-order delivery
- High per-connection memory footprint
  - Aim for many connections to find quick peers
- Complex NAT traversal
- Fixed congestion control algorithms
- I.e. not designed for "The Cloud"

Swift Peak Hashes

- Used to securely calculate content size


peak hash
7
3
11
1
5
9
13
0
2
4
6
8
10
12
14
0
1
2
3
4
5
6