The Swift Multiparty Transport Protocol
As PPSP

Arno Bakker, Victor Grishchenko, Riccardo Petrocco, 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
Swift design goals

1. Generic protocol that covers 3 use cases (vod, live, dl)
2. Have short prebuffering times
3. Be extensible:
   - Different congestion control algorithms (LEDBAT)
   - Different reciprocity algorithms (tit4tat, Give-to-Get)
   - Different peer-discovery schemes (tracker, DHT)
4. Can be carried over different transport protocols (UDP, TCP, RTP profile, HTTP)
5. Traverse NATs transparently
6. Low footprint
Swift messages

- Basic unit of communication: **Message**
  - **HANDSHAKE**
  - **HAVE**: convey chunk availability
  - **HINT**: request chunks
  - **DATA**: actual chunk
  - **HASH**: MDCs to enable integrity verification
  - ...

- Messages are *multiplexed* together when sent over the wire.
Swift on the wire: Example 1

- Peer A and B both have some chunks

- Note: *low latency*, data transfer already in 3rd datagram.
Swift on the wire: Example 2

- Peer A and B both have some chunks
- Are receiving chunks from others in parallel

Note: Chunk availability always up-to-date by pushing
Chunk availability and Rarest first

- **Rarest-first** is common element in chunk selection policies:
  - Peers download chunk that least peers have
    - Low supply
  - Peers can upload that to many peers
    - High demand

- Result: **Upload capacity of peers exploited!**

- Requires:
  - Peers have **good view** of neighbours’ chunk availability
  - Hence: Swift **pushes** HAVE messages
Swift on the wire: Example 3

- Peer A is starting leecher, peer B is seeder

- Note: Receiver controls flow

Arno Bakker, Delft University of Technology, IETF 82 PPSP WG  15.11.2011, Taipei
Swift on the wire: Example 4

- Peer A is leecher, peer B is seeder,
- Peer A requests peer list

**Diagram:**
```
A -> B
<table>
<thead>
<tr>
<th>HANDSHAKE</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
</tr>
<tr>
<td>HANDSHAKE + HAVE</td>
</tr>
<tr>
<td>A</td>
</tr>
<tr>
<td>HINT + PEX_REQ</td>
</tr>
<tr>
<td>B</td>
</tr>
<tr>
<td>PEX_ADD + HASH + DATA</td>
</tr>
</tbody>
</table>
```
Swift in detail

- Common set of messages across transports
- Novel method of content integrity protection:
  - Merkle hash trees
- Novel method of chunk addressing:
  - Bins
  - = Address range of chunks with single integer
- Novel method of privacy protection
  - Work in progress (ask Riccardo, for SVC too)
Swift integrity checking

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

Arno Bakker, Delft University of Technology, IETF 82 PPSP WG  15.11.2011, Taipei
Swift integrity checking (cont’d)

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

- Root hash + some peer addresses enough to start download!
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
Transport protocols

- Swift over UDP
  - Implemented
- Swift as RTP profile (charter hint)
- Swift over HTTP (charter hint)
Swift over UDP

- Datagram consists of **channel ID** + multiple messages
  - Channels allow different swarms on single UDP port
- Message is fixed length, first byte message ID
- IETF **LEDBAT** congestion control
- Simple NAT traversal via protocol itself
Swift as RTP profile

- cf. Secure Real-time Transport Protocol (SRTP)
  - “layer residing between RTP app and transport layer”
- Chunk = RTP packet

<table>
<thead>
<tr>
<th>V</th>
<th>P</th>
<th>X</th>
<th>CC</th>
<th>M</th>
<th>PT</th>
<th>Sequence Number</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Timestamp</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SSRC Identifier</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Extension ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Extension header length</td>
</tr>
<tr>
<td>Data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HINT+HAVE+HASH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Length of swift messages</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Swift as RTP profile (cont’d)

- RTP header protected against malicious modification
- Merkle tree can handle variable-sized chunks (if req)
- Advantages of UDP
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”

...  <- hashes

Chunk 8
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.