ROHC: compression of IP, UDP, RTP, TCP headers

Didier Barvaux, Viveris

didier.barvaux@toulouse.viveris.com
didier@rohc-lib.org

slides published under CC BY-NC-SA 4.0

Capitole du Libre, Toulouse, 2018
Didier Barvaux

- 2005: ENSEEIHT diploma
- 2005: daily job at Viveris https://www.viveris.fr/
- 2007: start working on ROHC library
  https://rohc-lib.org/
- ...
- 2018: still working at Viveris ;-)
- 2018: still working on ROHC library ;-)

Viveris
ROHC: compression of IP, UDP, RTP, TCP headers
Agenda

1. Header compression
2. The ROHC protocol
3. The ROHC library
4. Motivations
Header compression: why?

Header size is a concern on network links

- Voice Over IP (VoIP): 40B headers + 20B payload = 60B
- TCP Acknowlegments: 52B headers + 0B payload = 52B
Header compression: why?

**Header size is a concern on network links**
- Voice Over IP (VoIP): 40B headers + 20B payload = 60B
- TCP Acknowlegments: 52B headers + 0B payload = 52B

**Compressed packets**
- Voice Over IP (VoIP): 1B header + 20B payload (65% saved)
- TCP Acknowlegments: 4 bytes (92% saved)
Header compression: why?

Header size is a concern on network links
- Voice Over IP (VoIP): 40B headers + 20B payload = 60B
- TCP Acknowledgments: 52B headers + 0B payload = 52B

Compressed packets
- Voice Over IP (VoIP): 1B header + 20B payload (65% saved)
- TCP Acknowledgments: 4 bytes (92% saved)

Is header compression still useful in 2018?
## Header compression: why?

### Header size is a concern on network links
- Voice Over IP (VoIP): 40B headers + 20B payload = 60B
- TCP Acknowledgments: 52B headers + 0B payload = 52B

### Compressed packets
- Voice Over IP (VoIP): 1B header + 20B payload (65% saved)
- TCP Acknowledgments: 4 bytes (92% saved)

### Is header compression still useful in 2018?
- designed for serial links in 1990, network are now larger
Header compression: why?

Header size is a concern on network links
- Voice Over IP (VoIP): 40B headers + 20B payload = 60B
- TCP Acknowledgments: 52B headers + 0B payload = 52B

Compressed packets
- Voice Over IP (VoIP): 1B header + 20B payload (65% saved)
- TCP Acknowledgments: 4 bytes (92% saved)

Is header compression still useful in 2018?
- designed for serial links in 1990, network are now larger
- content is larger too + links are congested and/or expensive
1. Header compression

2. The ROHC protocol
   - definition
   - protocol

3. The ROHC library

4. Motivations
ROBust Header Compression (ROHC)

A network protocol that compresses away protocol headers
What ROHC is?

**RObust Header Compression (ROHC)**

A network protocol that compresses away protocol headers

**Objectives**

- lossless / transparent
- efficient
- robust to packet loss
- extensible framework IPv4, IPv6, UDP, UDP-Lite, RTP, TCP, ESP, GRE...
What ROHC is?

RObust Header Compression (ROHC)
A network protocol that compresses away protocol headers

Objectives

- lossless / transparent
- efficient
- robust to packet loss
- extensible framework IPv4, IPv6, UDP, UDP-Lite, RTP, TCP, ESP, GRE...

IETF standard https://www.ietf.org/

- ROHCv1 (2001)
- ROHCv2 (2008)
- ROHC-TCP (2013)
Main principles: only headers are compressed

- **IPv4**
- **UDP**
- **RTP**
- **audio codec**

- **IPv4**
- **TCP**
- **TCP options**
- **HTTP**

- **IPv4**
- **ICMP**
- **ICMP Echo payload**

- **IPv4 fragment**
- **TCP**
- **truncated HTTP**

- **IPv4 fragment**
- **TCP**
- **truncated HTTP**

Viveris

ROHC: compression of IP, UDP, RTP, TCP headers
Main principles: packet classification

classify packets by streams to optimize redundancy

incoming packets

- TCP stream #1
- TCP stream #2
- RTP stream
- UDP stream
- ICMP stream
- others

ROHC: compression of IP, UDP, RTP, TCP headers
Main principles: analogy with video compression

- transmit keyframes with full content regularly
- transmit only deltas between 2 keyframes

Main principles: do not transmit at all if possible

Well-known or inferred fields

avoid transmitting well-known or inferred fields

<table>
<thead>
<tr>
<th>Field</th>
<th>IPv4</th>
<th>UDP</th>
<th>RTP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Version</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>IHL</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Type of Service</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Total Length</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Fragment Offset</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Header Checksum</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Identification</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Flags</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Time to Live</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Protocol</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Source Address</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Destination Address</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Source Port</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Destination Port</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Length</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Checksum</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>V</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>P</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>X</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>CC</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>M</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>PT</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>sequence number</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>timestamp</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>synchronization source (SSRC) identifier</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Main principles: transmit as less as possible

### Static fields

Transmit static fields only a few times

<table>
<thead>
<tr>
<th>IPv4</th>
<th>Version</th>
<th>IHL</th>
<th>Type of Service</th>
<th>Total Length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Identification</td>
<td>Flags</td>
<td>Fragment Offset</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Time to Live</td>
<td>Protocol</td>
<td>Header Checksum</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Source Address</td>
<td>Destination Address</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>UDP</th>
<th>Source Port</th>
<th>Destination Port</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Length</td>
<td>Checksum</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RTP</th>
<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></td>
<td>timestamp</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>synchronization source (SSRC) identifier</td>
</tr>
</tbody>
</table>
## Main principles: transmit only deltas

Changing fields (eg. RTP or TCP sequence numbers)

W-LSB algorithm: designed to compress small changes in a robust way

<table>
<thead>
<tr>
<th>uncompressed</th>
<th>compressed</th>
<th>16b</th>
<th>3b</th>
<th>5b</th>
</tr>
</thead>
<tbody>
<tr>
<td>42 0 0 0 0 0 0 0 0 0 1 0 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>43 0 0 0 0 0 0 0 0 0 1 0 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>44 0 0 0 0 0 0 0 0 0 1 0 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>45 0 0 0 0 0 0 0 0 0 1 0 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>46 0 0 0 0 0 0 0 0 0 1 1 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>47 0 0 0 0 0 0 0 0 0 1 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>48 0 0 0 0 0 0 0 0 0 1 1 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ROHC: compression of IP, UDP, RTP, TCP headers
1. Header compression

2. The ROHC protocol

3. The ROHC library
   - history
   - new features
   - library API example

4. Motivations
History

Genesis

- 2003: initial version by Lulea University of Technologies
- 2007: internal fork by TAS, CNES, and Viveris Technologies
- 2009: public version of the fork (GPLv2+)
- 2014: LGPLv2+ license
History

Genesis

- 2003: initial version by Lulea University of Technologies
- 2007: internal fork by TAS, CNES, and Viveris Technologies
- 2009: public version of the fork (GPLv2+)
- 2014: LGPLv2+ license

Features

- 2012: ROHCv1 IP/ESP profile
- 2013: Linux kernel
- 2016: IP/TCP profile
- 2017: Context Replication for IP/TCP profile
New features

Latest version 2.2.0 released on April 2018

- add ROHCv2 IP-only, IP/ESP and IP/UDP profiles (with Developing Solutions)
- improve interoperability on ROHCv1 standards
- Wireshark dissector in Lua
New features

Latest version 2.2.0 released on April 2018
- add ROHCv2 IP-only, IP/ESP and IP/UDP profiles (with Developing Solutions)
- improve interoperability on ROHCv1 standards
- Wireshark dissector in Lua

Next version 2.3.0 to be released
- partial ROHCv2 IP/UDP/RTP profile
- improve interoperability on ROHCv1 standards
- improve robustness on lossy mediums
- better CPU and memory performances
Compressing one IP/UDP/RTP packet

```
struct rohc_comp *compressor;
...
compressor =
    rohc_comp_new2 (ROHC_SMALL_CID, ROHC_SMALL_CID_MAX,
                    gen_random_num, NULL);
rohc_comp_enable_profile (compressor, ROHC_PROFILE_RTP);
...
rohc_compress4 (compressor, ip_packet, &rohc_packet);
...
rohc_comp_free (compressor);
```

API documentation, tutorials and examples on
https://rohc-lib.org/support/documentation/
1. Header compression
2. The ROHC protocol
3. The ROHC library
4. Motivations

ROHC: compression of IP, UDP, RTP, TCP headers
A way to improve skills

Skills I learned with the project

- C language and compilers, Python module, Lua, shell...
- dev tools: autotools, Bzr, Git, vim
- continuous integration: Travis CI, Buildbot
- static analysis: clang, cppcheck, coverity
- unit tests: 88% LOC coverage computed by gcov/lcov
- runtime analysis: valgrind, asan/ubsan, AFL fuzzer
- debug tools: gdb, strace, perf, tcpdump, wireshark, tshark
- software hardening: capabilities, seccomp-bpf, namespaces
- documentation: Doxygen, man pages
- packaging: ebuild, RPM, DEB

Skills I would like to learn in the future

- Rust
- Meson