Static Context Header Compression

[sjiekJ]

Where do we want to go in RIOT?
WHERE ARE WE GOING TO?

“an entirely new reality in which our interface to the Internet will no longer be predominantly a screen, but rather the objects of the cyber-physical system embodied by the Internet of Things.”
TOWARDS A WEB OF THINGS

Connecting any *thing* to provide an interface to the real world.
WHAT DEFINES A THING?

1. Services and semantics
   • What do I measure?
   • What is on/off?

2. Networking
   • How do I send a packet?

3. Communication
   • What frequency do I use?
## Low Power Wide Area Networks

Short range technologies do not suffice.

<table>
<thead>
<tr>
<th>Technology</th>
<th>Bytes</th>
<th>Range</th>
<th>Data Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sigfox</td>
<td>8 - 12 bytes</td>
<td>&lt; 15 km</td>
<td>0.1 kbps</td>
</tr>
<tr>
<td>LoRa</td>
<td>51 - 242 bytes</td>
<td>&lt; 15 km</td>
<td>0.3 - 5.5 kbps</td>
</tr>
<tr>
<td>NB-IoT</td>
<td>125 bytes</td>
<td>&lt; 10 km</td>
<td>100 kbps</td>
</tr>
<tr>
<td>Weightless</td>
<td>&gt; 10 bytes</td>
<td>&lt; 5 km</td>
<td>&lt; 10 Mbps</td>
</tr>
</tbody>
</table>
• Generic Framework for Header compression in LPWANs: RFC 8724
Static Context Header Compression employs the fact that

- UDP/IP flows remain static in LPWANs
- some fields can be calculated from L2

These static flows are stored on both sides

- the id of the flow is sent to the other side
The list of flows contains a Target Value (TV) for every header field
• Can be matched using the Matching Operator (MO)
• Can be (de)compressed using the Compression/Decompression Action (CDA)

Send the rule id and possible residue
Matching Operator provides flexibility to match a header field with a rule field:

- **equal** matches the header field with the target value
- **match-map** sends the index of the matched target value
- **MSB** sends the last bits of the original header field
COMPRESSION EXAMPLE

Sensor response to a CoAP POST request.

POST /3311/0/5850; On
0x48 0x02 0x8a 0x00 0x02 0x0d 0x61
0x54 0x30 0x1f 0x20 0x9e 0x01 0x33
0x01 0x30 0x02 0x32 0x31 0xff 0x6f
0x6e

2.04 CHANGED
0x68 0x45 0x8a 0xda 0xda 0xe1
0x54 0x30 0x1f 0x20 0x9e 0xc0
LwM2M* EXAMPLE
* or any IPv6 enabled service
FRAGMENTATION

- Packet remains too large
- Cope with IPv6 MTU value for routers (1280 bytes): fragmentation
  - No-ack: no reliability
  - Ack-On-Error: acknowledge an erroneous window
  - Ack-Always: total reliability
### Fragmentation Example

#### Bitmap

<table>
<thead>
<tr>
<th>SENDER</th>
<th>RECEIVER</th>
</tr>
</thead>
<tbody>
<tr>
<td>all-1 empty</td>
<td>all-1 empty</td>
</tr>
<tr>
<td>all-1 window</td>
<td>all-1 window</td>
</tr>
<tr>
<td>all-0 empty</td>
<td>all-0 empty</td>
</tr>
<tr>
<td>ack</td>
<td>ack</td>
</tr>
<tr>
<td>all-0 window</td>
<td>all-0 window</td>
</tr>
<tr>
<td>reg. fragment</td>
<td>reg. fragment</td>
</tr>
</tbody>
</table>

**Bitmap:**
- 1000000
- 1100000
- 1110000
- 1110100
- 1110110

**ACK:**
- 1111110
- 1111111
- 1111111
- 1111111

**Timer Expired:**
- 0

**Receiver:**
- 1111110
- 1111111
- 1111111
- 1111111

**Sender:**
- 6
- 5
- 4
- 3
- 2
- 1
- 0

**SCHC Ack-Always / Ack-On-Error Reliability**
RIOT INTEGRATION
LIBSCHC

Open source implementation of

- Rule context
- Compression
  - Interoperability tested using test vectors of the LPWAN WG
- Fragmentation/reassembly
  - Tiles, windows, bitmaps, timers
  - Integrity checking
  - No-Ack, Ack-Always, Ack-on-Error
- Padding management

https://github.com/imec-idlab/libschc
CURRENT IMPLEMENTATION

Memory management

• fragmented packet uses pre-allocated chunk of memory, stored in a mbuf (network memory buffer) chain
• contains a pointer to headers and payload

```c
typedef struct schc_mbuf_t {
    ...  
    /* the length of the fragment */
    uint16_t len;
    /* pointer to the chunk of memory */
    uint8_t* data;
    /* pointer to the next mbuf */
    struct schc_mbuf_t next;
} schc_mbuf_t;
```
Abstraction required

- differentiate between end-devices
- possibility to
  - reorder linked list (missing fragments)
  - take a single payload byte from the chain
- to calculate MIC
RIOT INTEGRATION

Currently poor memory management in libSCHC.

- Separate memory and mbuf logic and instead use pktsnip, pktqueue
- Pass stripped, concatenated fragments to the IPv6 layer without copying
WHAT’S NEXT?

SCHC specifies CoAP w/ OSCORE compression for end-to-end security.

• Integrate libOSCORE and libSCHC for end-to-end security in LPWANs
WHAT'S NEXT?

- SCHC is a generic framework:
  - HTTP
  - CoAP
  - MQTT
  - ...
  - NDN?
- RIOT/LoRaWAN gateway w/o MQTT
- Firmware updates for LPWANs

---

RIOT SUMMIT 2020
Static Context Header Compression [sjiek] in RIOT

BART MOONS.
Ghent University – IDLab – imec
bamoons.moons@ugent.be
idlab.ugent.be