



Static Context Header Compression

[sjiek]

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 *anything* to provide an interface to the real world.



Web
with JSON
over HTTP
using OAuth



oneM2M
with JSON
over MQTT
using TLS



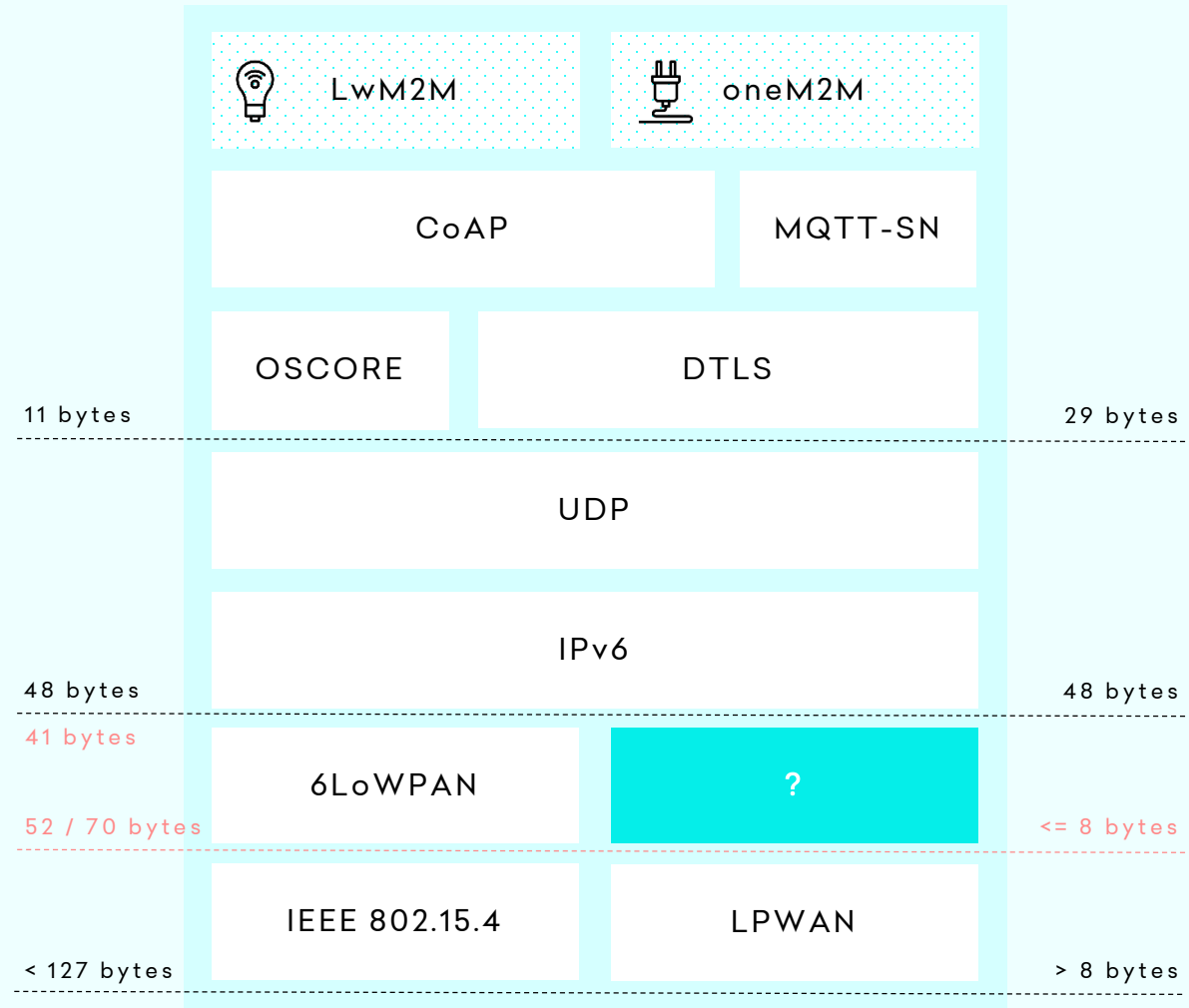
LwM2M
with SenML
over CoAP
using DTLS



Web
with CBOR
over CoAP
using OSCORE

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?



1) SHORT RANGE AND 2) LONG RANGE IoT STACKS

LOW POWER WIDE AREA NETWORKS

Short range technologies do not suffice.



Sigfox
8 - 12 bytes

< 15 km
0.1 kbps



LoRa
51 - 242 bytes

< 15 km
0.3 - 5.5 kbps



NB-IoT
125 bytes

< 10 km
100 kbps

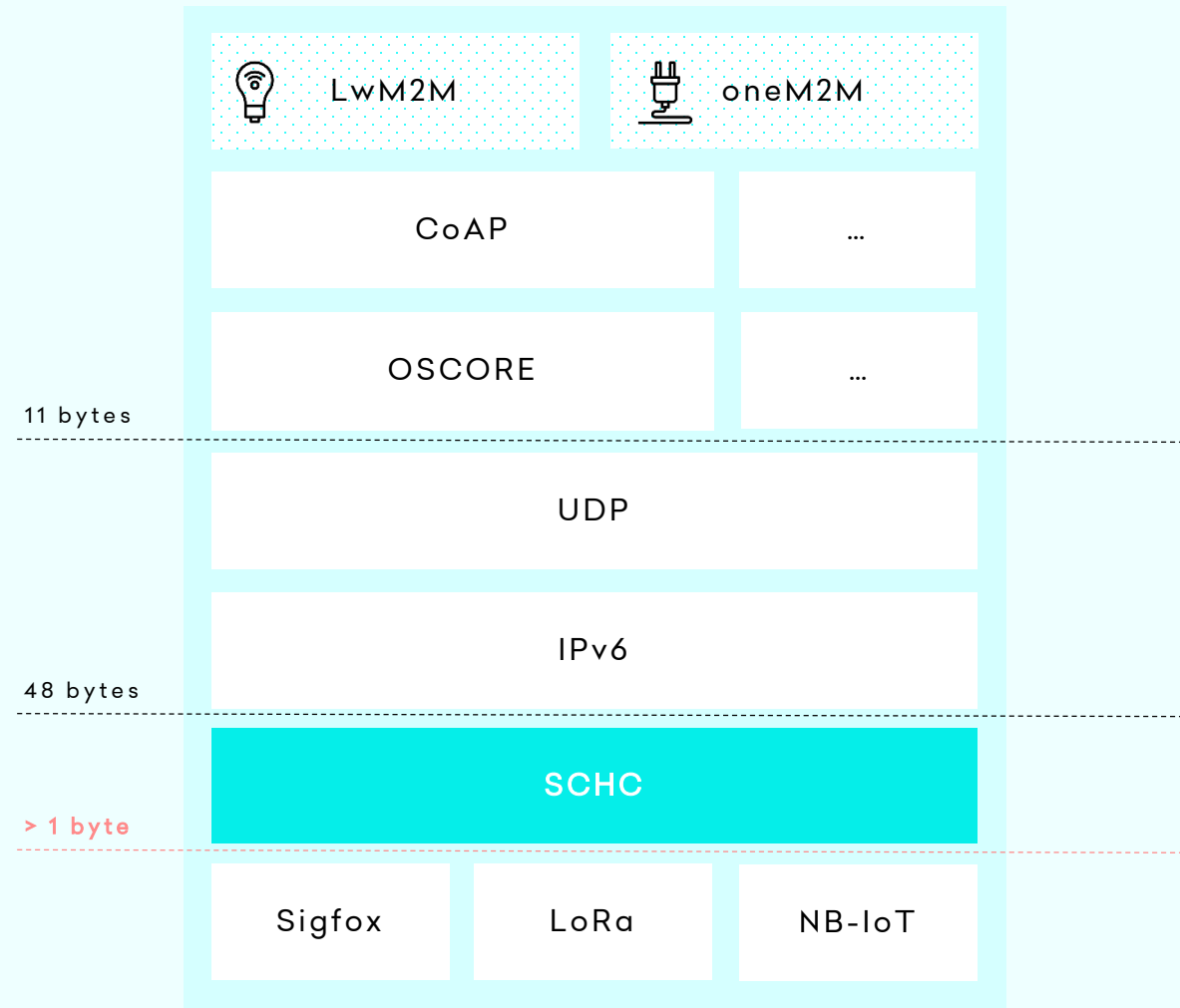


Weightless
> 10 bytes

< 5 km
< 10 Mbps

STATIC CONTEXT HEADER **COMPRESSION**

- Generic Framework for Header compression in LPWANs: **RFC 8724**



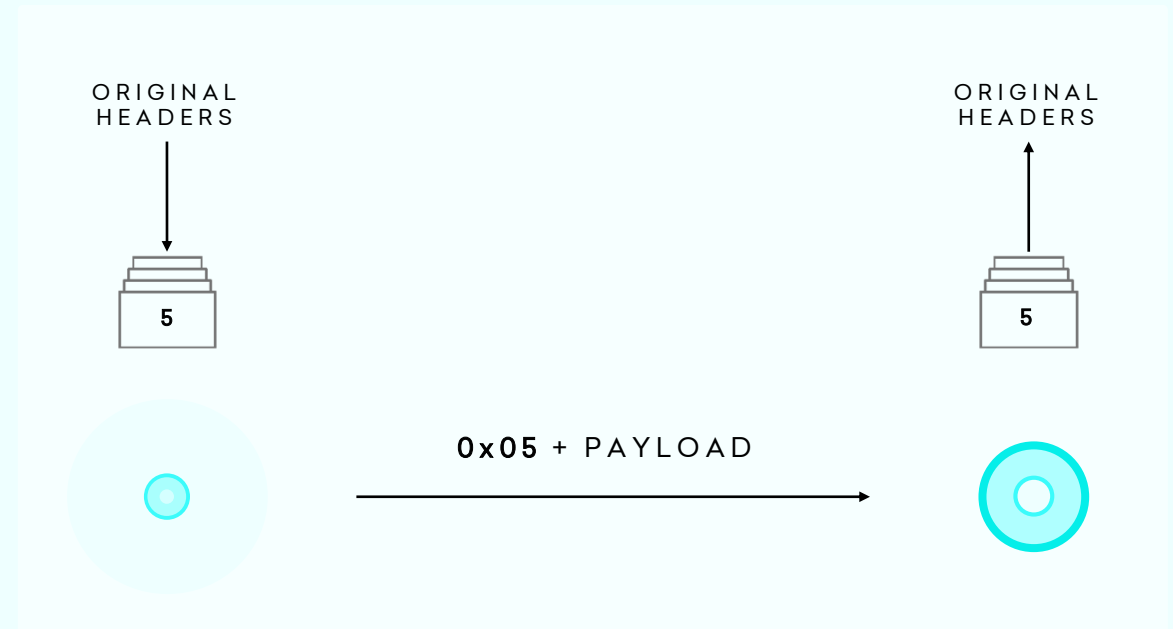
STATIC CONTEXT HEADER **COMPRESSION**

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



SCHC CONTEXT

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

FIELD	FL	DI	TV	MO	CDA	ID 1
src port	16	BI	5683	&equal	not-sent	

STATIC CONTEXT HEADER COMPRESSION 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

Field	FL	DI	TV	MO	CDA	ID 1
Src port	16	BI	5683	&MSB(4)	LSB(12)	
Dst port	16	BI	5683	&equal	not-sent	
Length	16	BI		&ignore	comp-length	
Checksum	16	BI		&ignore	comp-chk	

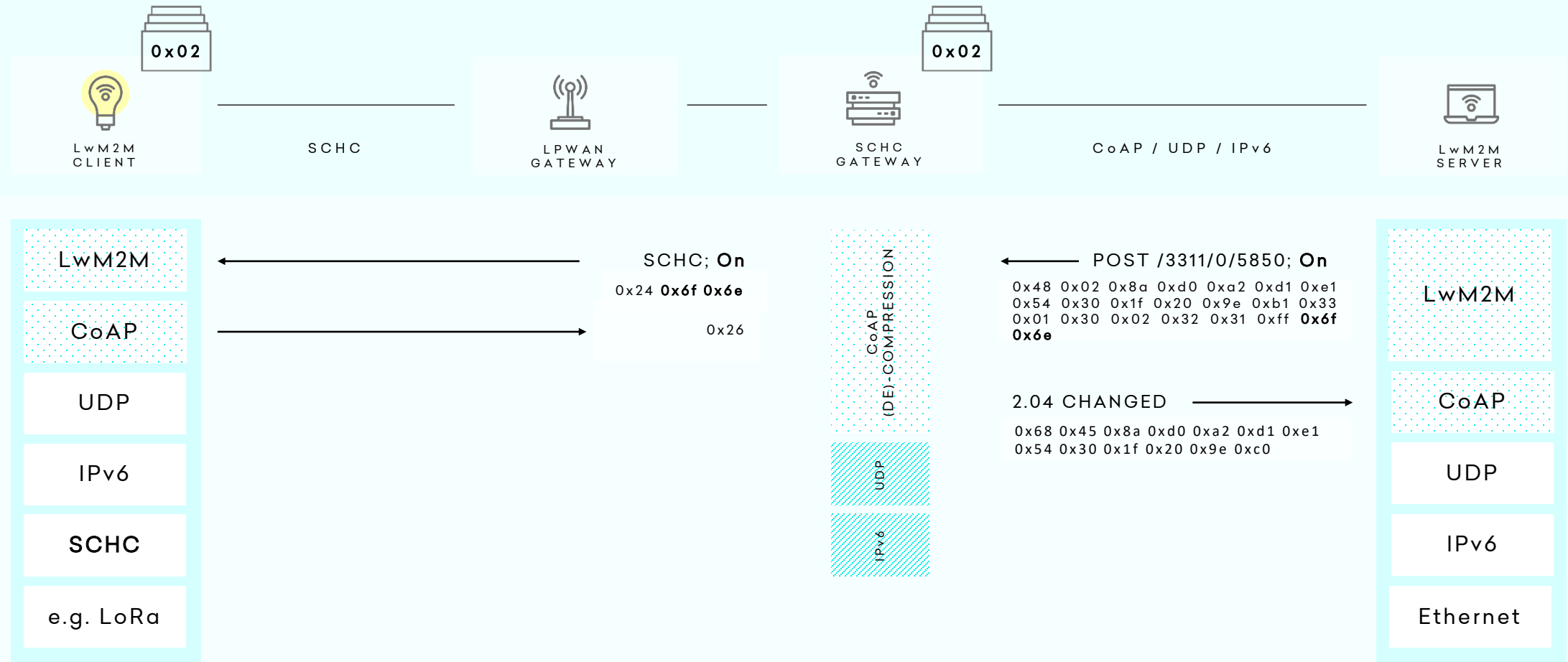
Table 1: UDP RULE1

Field	FL	DI	TV	MO	CDA	ID 2
Version	2	BI	1	&equal	not-sent	
Type	2	DW	CON	&equal	not-sent	
Type	2	BI	[ACK, RST]	&match-map	mapping-sent	
TKL	4	BI	0x00	&equal	not-sent	
Code	8	UP	GET	&equal	not-sent	
Code	8	DW	CONTENT	&equal	not-sent	
MID	16	BI	0x0000	MSB(4)	LSB(12)	
Uri-Path	88	DW	temperature	&equal	not-sent	

Table 2: CoAP RULE1

COMPRESSION EXAMPLE

Sensor response to a CoAP POST request.



LwM2M* EXAMPLE

* or any IPv6 enabled service

The screenshot displays a web browser window titled "Leshan Server Demo" at the URL "192.168.0.198:8080/#/clients". The interface shows a table of clients and a detailed view of a specific client.

Client Endpoint	Registration ID	Registration Date	Last Update
lwm2m-client-1	PhGBaV0tdf	Sep 13, 2020 12:49:13 PM	Sep 13, 2020 12:49:13 PM

Client details for lwm2m-client-1:

- Lifetime: 1200s
- Binding mode: U
- Protocol version: 1.1
- Address: 2a02:1810:2f1e:e600:0:0:3:5683

Below the table, there is a "LwM2M SERVER" icon and a lightbulb icon representing a client.

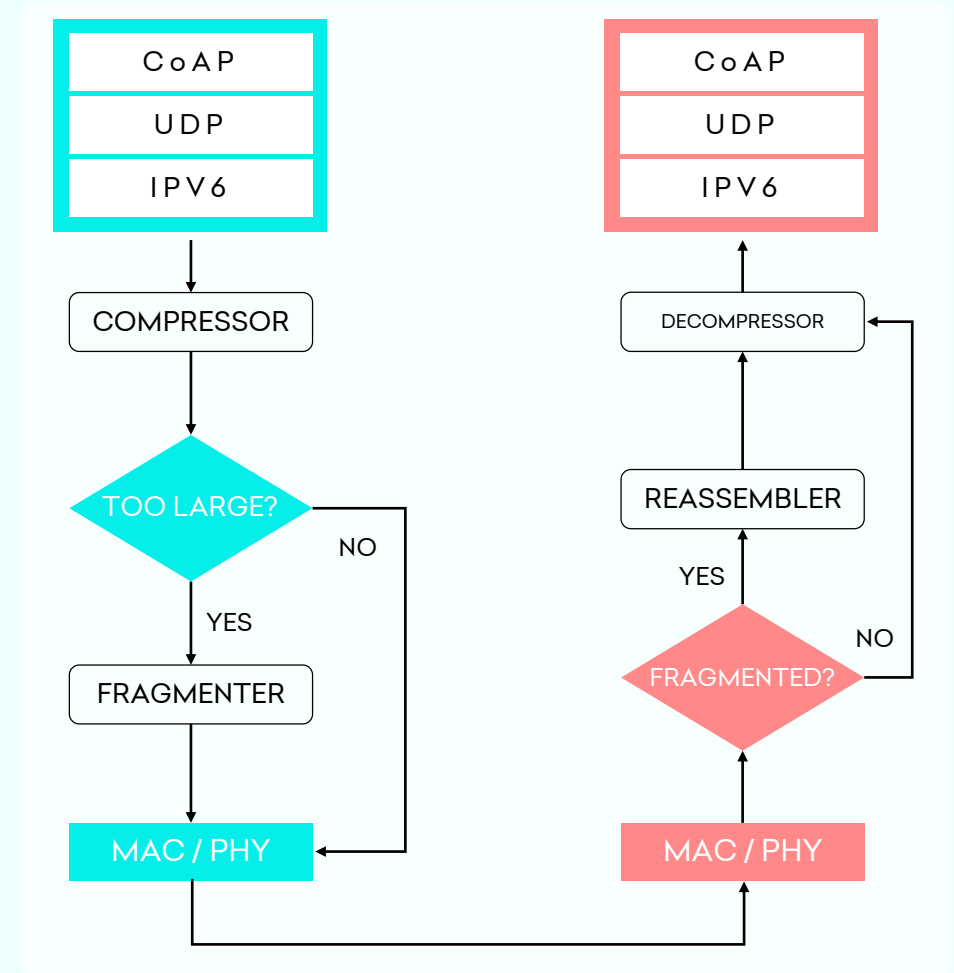
Two terminal windows are overlaid on the bottom half of the image, showing network traffic analysis:

```
boortmans@moussorgsky: ~/Nextcloud/Projects/PortForward/Project/device_managemen...  
File Edit View Search Terminal Help  
-----+-----  
| Original Packet |  
-----+-----  
60 00 00 00 00 1E 11 40 2A 02 18 10  
2F 1E E6 00 BA 27 EB FF FE 5D 14 0A  
2A 02 18 10 2F 1E E6 00 00 00 00 00  
00 00 00 03 16 33 16 33 00 1E 0B 6F  
64 41 89 C4 C4 89 0A 00 B2 72 64 0A  
50 68 47 42 61 56 30 74 64 66  
  
[049] packet_received_callback(): decompressed length is 70 (70), contents:  
[050] 60 00 00 00 00 1E 11 40 2A 02 18 10 2F 1E E6 00 BA 27 EB FF FE 5D 14 0A 2A  
02 18 10 2F 1E E6 00 00 00 00 00 00 03 16 33 16 33 00 1E 0B 6F 64 41 89 C  
4 C4 89 0A 00 B2 72 64 0A 50 68 47 42 61 56 30 74 64 66  
[051] uIP log: udp: udp_input  
[lwm2m_handle_packet:210] Entering  
[lwm2m_handle_packet:214] Parsed: ver 1, type 2, tk1 4, code 2.01, mid 35268, Co  
ntent type: 0  
[lwm2m_handle_packet:216] Payload:  
[transaction_handleResponse:276] Entering  
[lwm2m_handleRegistrationReply:642] 123 Registration successful,  
[lwm2m_data_new:160] size: 1
```

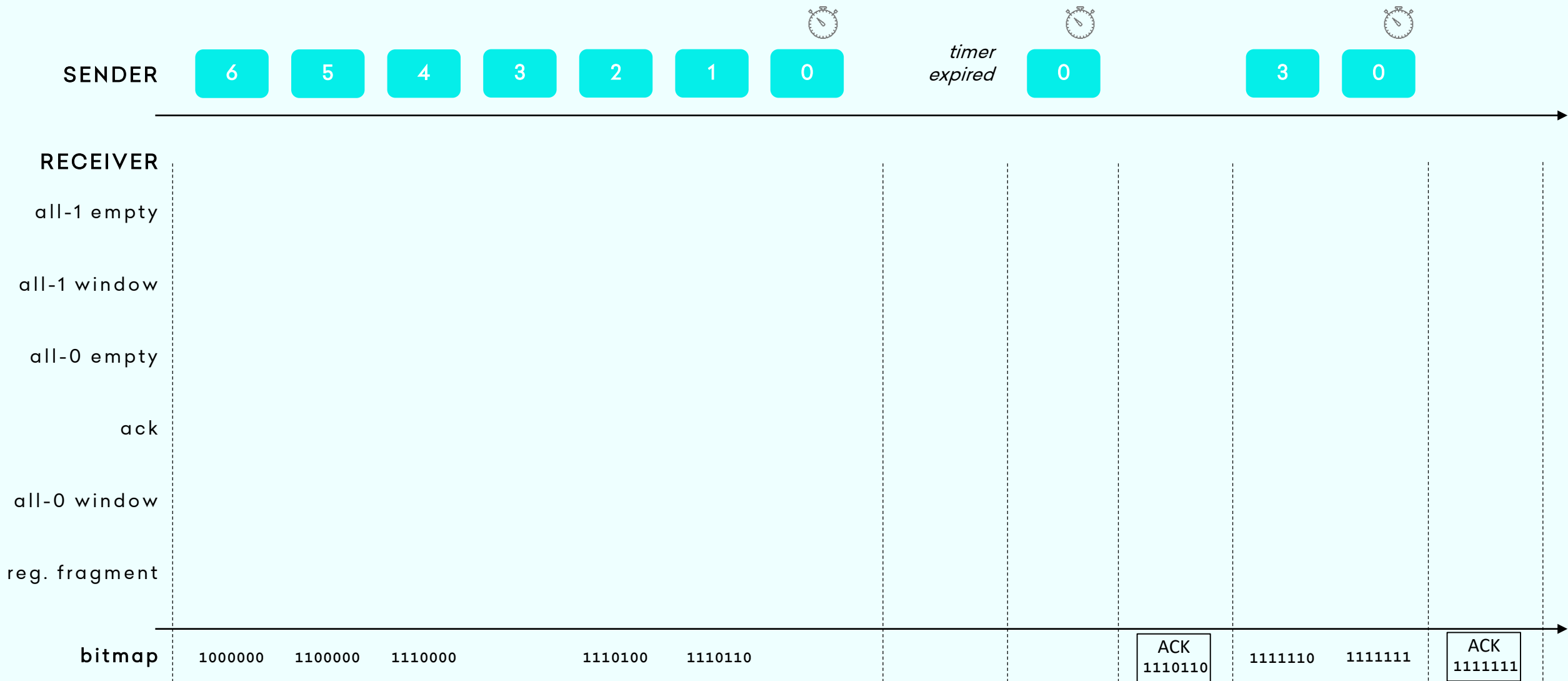
```
boortmans@moussorgsky: ~/Nextcloud/Projects/PortForward/Project/device_managemen...  
File Edit View Search Terminal Help  
-----+-----  
| Original Packet |  
-----+-----  
60 00 00 00 00 5F 11 40 2A 02 18 10  
2F 1E E6 00 00 00 00 00 00 00 00 03  
2A 02 18 10 2F 1E E6 00 BA 27 EB FF  
FE 5D 14 0A 16 33 16 33 00 5F D7 45  
44 02 89 C4 C4 89 0A 00 B2 72 64 11  
28 39 6C 77 6D 32 6D 3D 31 2E 31 0D  
04 65 70 3D 6C 77 6D 32 6D 2D 63 6C  
69 65 6E 74 2D 31 03 62 3D 55 07 6C  
74 3D 31 32 30 30 FF 3C 2F 3E 3B 72  
74 3D 22 6F 6D 61 2E 6C 77 6D 32 6D  
22 2C 3C 2F 31 2F 30 3E 2C 3C 2F 33  
2F 30 3E  
  
mbuf_delete(): mbuf is head, delete head  
mbuf_delete(): clear slot 0 in mbuf pool  
After LPWAN: 135 | 60000000 005f1140 2a021810 2f1ee600 00000000 00000003  
Forward packet from LpWAN to IPv6 network: 2a02:1810:2f1e:e600::3 -> 2a02:1810:2  
f1e:e600:ba27:ebff:fe5d:140a plen 95, next 17, hlim 64  
Received IPv6 packet with length 70 for node 1  
60 a 44 35 0 1e 11 40 2a 2 18 10 2f 1e e6 0 ba 27 eb ff fe 5d 14 a 2a 2 18 10 2f  
1e e6 0 0 0 0 0 0 0 3 16 33 16 33 0 1e 3b 6f 64 41 89 c4 c4 89 a 0 82 72 64 a  
50 68 47 42 61 56 30 74 64 66
```

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



RIOT INTEGRATION

The screenshot shows a network traffic capture tool interface. The main window displays a list of captured packets with columns for No., Time, Source, Destination, Protocol, Length, Data, and Info. The packets include MDNS, Broadcast, and ICMPv6 traffic. Below the packet list, a detailed view of a frame is shown, including Ethernet II, Internet Protocol Version 6, and User Datagram Protocol (UDP) information.

Two terminal windows are open in the foreground, showing the execution of a program. The left terminal window displays the output of a program named 'schc', showing the transmission and reception of a packet. The right terminal window displays the output of a program named 'mbuf_delete', showing the deletion of a packet from a mbuf pool.

```
boortmans@moussorgsky: ~/Nextcloud/RIOT/RIOT/examples/gnrc_schc
File Edit View Search Terminal Help
schc: set timer (5000 msec) for device 1
SEND
schc_fragment(): all-1 window
set_fragmentation_header(): padding bits of last tile 8
compute_mic(): original packet length 176 bits, last tile padding 8 bits, extra padding 0 bits
compute_mic(): MIC for device 1 is E3F54C1A
send_fragment(): sending fragment 4 with length 8 to device 1
0x0C 0x7E 0x3F 0x54 0xC1 0xA3 0x74 0x00
schc: transmitting packet with length 8 for device 1
schc: sent 8 bytes.
set_local_bitmap(): for fcn 7 at index 6
1 1 1 0 0 1
set_retrans_timer(): for 20000 ms
schc: set timer (20000 msec) for device 1
schc: GNRC_NETDEV_MSG_TYPE_RCV received
schc: received packet with length 2
schc: Received packet has no netif header
0000 33 33 00 00 00
0010 b1 fc 00 30 11 WAIT_BITMAP
0020 3b ff fe 36 63
0030 00 00 00 00 00 w == w
0040 00 00 00 01 00 no more fragments, MIC ok
0050 65 79 2d 68 6b schc_fragment(): end transmission cycle
0060 6c 00 00 0c 00 end_tx() callback
schc: waiting for incoming message.
```

```
boortmans@moussorgsky: ~/Nextcloud/RIOT/RIOT/examples/gnrc_schc
File Edit View Search Terminal Help
mbuf_delete(): set head
mbuf_delete(): clear slot 0 in mbuf pool
mbuf_delete(): set head
mbuf_delete(): clear slot 1 in mbuf pool
mbuf_delete(): set head
mbuf_delete(): clear slot 2 in mbuf pool
mbuf_delete(): mbuf is head, delete head
mbuf_delete(): clear slot 3 in mbuf pool
udp: GNRC_NETAPI_MSG_TYPE_RCV
PKTDUMP: data received:
-- SNIP 0 - size: 4 byte, type: NETTYPE_UNDEF (0)
00000000 74 65 73 74
-- SNIP 1 - size: 8 byte, type: NETTYPE_UDP (4)
src-port: 8000 dst-port: 8000
length: 12 cksum: 0xb2de
-- SNIP 2 - size: 40 byte, type: NETTYPE_IPV6 (2)
traffic class: 0x00 (ECN: 0x0, DSCP: 0x00)
flow label: 0x00000
length: 12 next header: 17 hop limit: 64
source address: fe80::7038:f6ff:fe2d:f9d2
destination address: fe80::2c8d:3bff:fe36:639f
-- PKT - 3 snips, total size: 52 byte
```

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

imec-idlab / libschc

<> Code Issues 7 Pull requests 0 Actions Projects 0 Wiki Security 0 Insights Settings

A C implementation of the Static Context Header Compression

lpwan schc compression fragmentation coap udp ipv6 Manage topics

318 commits 1 branch 0 packages 0 releases 3 contributors GPL-3.0

Branch: master New pull request Create new file Upload files Find file Clone or download

Commit	Author	Time
bartmoons git merge		Latest commit 5089baa on Apr 8
docs	fragmenter cc99	2 months ago
examples	fragmenter cc99	2 months ago
rules	fragmenter cc99	2 months ago
.gitignore	rule configuration	3 months ago

README.md

libschc: A C implementation of the Static Context Header Compression

ABOUT LIBSCHC

libschc is a C implementation of the Static Context Header Compression, drafted by the IETF. It is a header compression technique, used in Low Power Wide Area Networks in order to enable tiny low-power microcontrollers to have an end-to-end IPv6 connection. This repository contains both the compression as well as the fragmentation mechanism. For further information related to SCHC, see <https://datatracker.ietf.org/doc/draft-ietf-lpwan-ipv6-static-context-hc/>.

See the [docs](#) for more information on the implementation and configuration.

ACKNOWLEDGEMENT

libschc has been developed partially with the support of VLAIO, FWO and imec. It is also part of the IoT middleware stack, being developed for the European Union's Horizon 2020 PortForward project, where, amongst others, LwM2M will be integrated with this library in order to deliver open standards-based sensor-Cloud connectivity.

CURRENT IMPLEMENTATION

Memory management

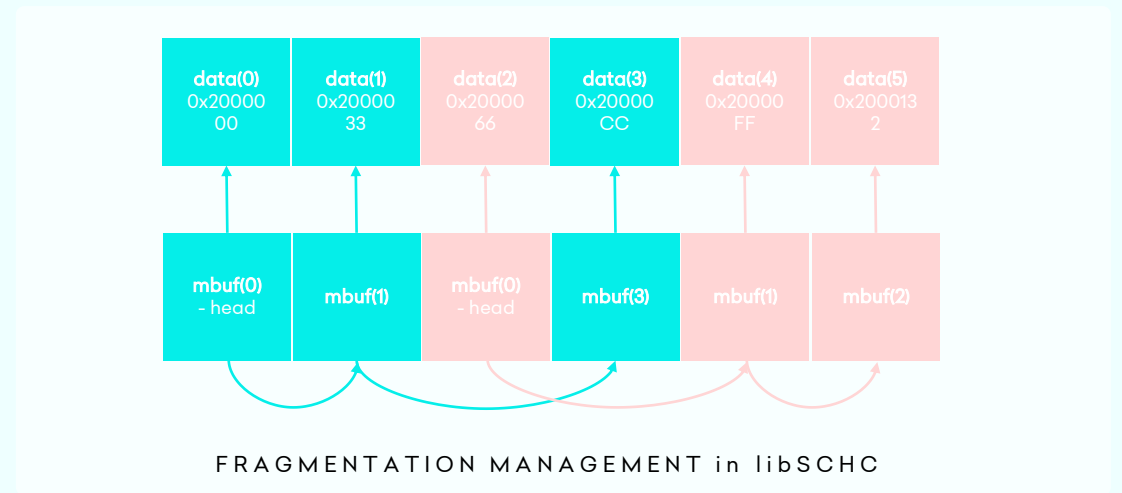
- fragmented packet uses pre-allocated chunk of memory, stored in a **mbuf** (*network memory buffer*) chain
- contains a pointer to headers and payload

```
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;
```


FRAGMENTATION MANAGEMENT

Abstraction required

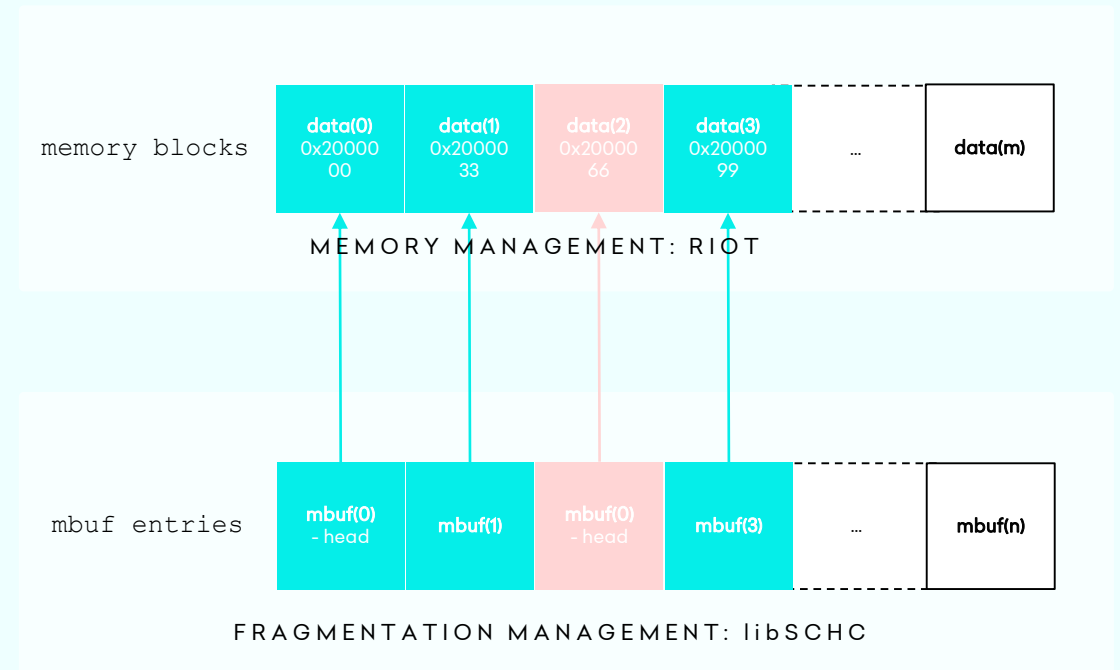
- differentiate between end-devices
- possibility to
 - reorder linked list (missing fragments)
 - take a single payload byte from the chain
 - to calculate MIC



RIOT INTEGRATION

Currently poor memory management in `libSCHC`.

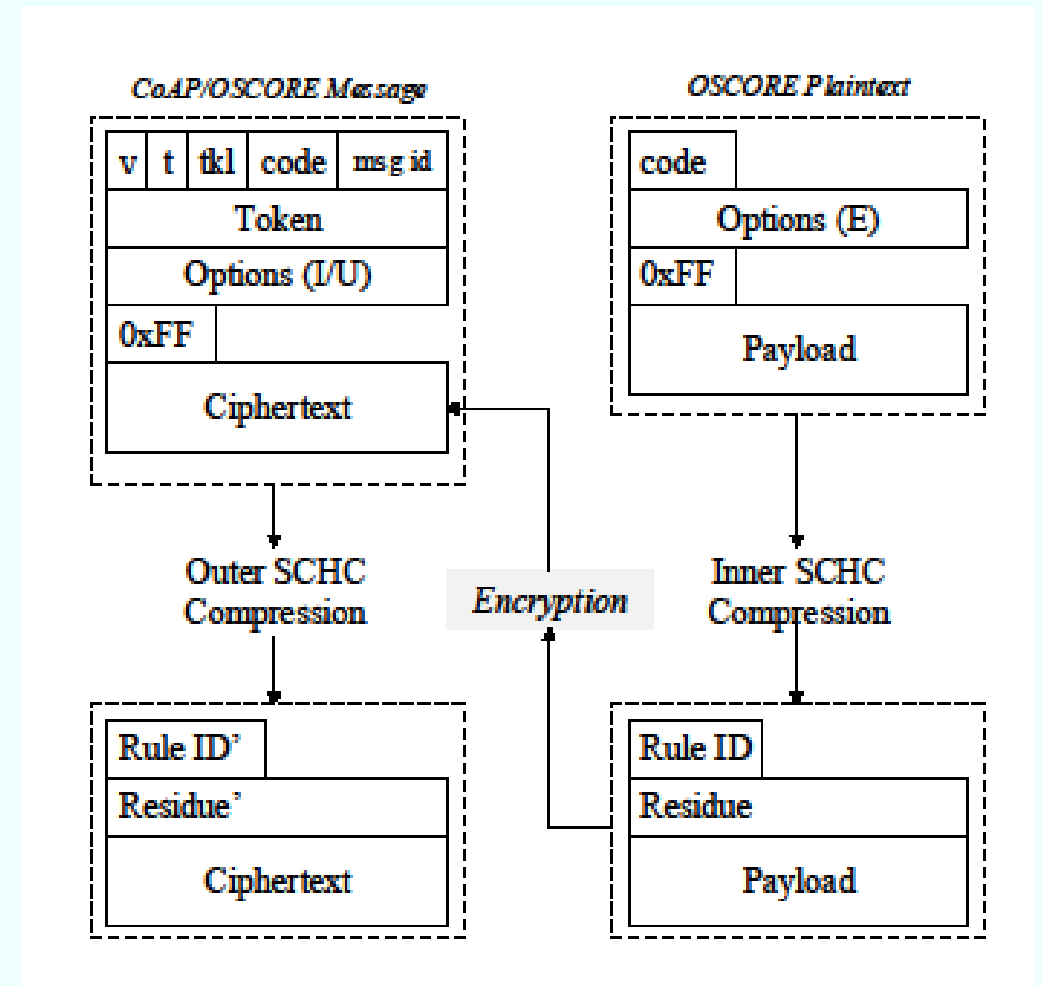
- Separate memory and `mbuf` logic and instead use `pktsnip`, `pktqueue`
- Pass stripped, concatenated fragments to the IPv6 layer without copying



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

Web
with JSON
over HTTP
using OAuth

oneM2M
with JSON
over MQTT
using TLS

LwM2M
with SenML
over CoAP
using DTLS

Web
with CBOR
over CoAP
using OSCORE

Static Context Header Compression [sjiek] in RIOT

BART MOONS.

Ghent University – IDLab – imec

bamoons.moons@ugent.be

idlab.ugent.be



Bart Moons