CCN and NDN TLV encodings in 802.15.4 packets

Marc Mosko
Palo Alto Research Center
marc.mosko@parc.com

Christian Tschudin
University of Basel
christian.tschudin@unibas.ch

IRTF ICNRG interim meeting, Boston, Jan 13+14, 2015

Contributions

- A new TLV encoding called 1+0:
  - 1 byte for T *and* L
  - assumes «contextual type values»
- Embeddable in fixed-length CCNx1.0 as well as variable size NDN codes
- Concrete IoT example, emphasizing security *important for IoT, think door locks etc*
- Take home message: size matters
  - permit for enough security bits
  - reduce air time (battery life)
  - avoid fragmentation

Overview

- 802.15.4 intro, packet examples
- The case for 1-byte IoT TLV encoding
- Example using 1+0 encoding

802.15.4 PHY MTU of 127 bytes

- Same problems as IPv6 (RFC 4944)

	2-byte addr	8-byte addr
Maximum Payload	127	127
802.15.4 MAC header	-11	-23
802.15.4 Security header	-5	-5
AES-CCM-16 Encrypted MAC*	-16	-16
802.15.4 FCS	-2	-2
Available Payload Size	**93**	**81**

* Encrypted Message Authentication Code

See also Sastry & Wagner, "Security considerations for IEEE 802.15.4", http://www.cs.berkeley.edu/~daw/papers/15.4-wise04.pdf

802.15.4 Packet Assumptions

- Use worst case 8-byte addresses with PAN ID
- AES-CCM-16 encryption with authentication
- Content Object/Data uses 16-byte HMAC sig
- Name `/abcd/efgh/ijkl` (4/4/4)
- Only mandatory fields
- 32-bytes of user payload
- No fragmentation! Fit in one packet.

Disclaimer

- You can always twiddle fields or use less overhead, different names, etc.. If you hand craft CCN/NDN packets for 802.15.4, you can obviously do better – we wanted to stick with TLV.
- The 32-byte payload was picked before creating the packets to see if we could make that fit.
- One could use the 802.15.4 AES-CCM-16 signature and encryption instead of a CCN/NDN Signature on the Data -- has drawbacks.
- Comparing 1+0 with: 2+2 CCN, 1+1 CCN, 1+1 NDN

CCN Encoding OCTETS

	PHY	Fixed	Data	2+2	1+1	1+0
802.15.4 GFSK PHY header	6					
802.15.4 64-bit address		23				
802.15.4 Security header		5				
Fixed Header		8				
ContentObjectMessage TL				4	2	1
Name TL				4	2	1
Name Component TL				4	2	1
Name /abcd			4			
Name Component TL				4	2	1
Name /efgh			4			
Name Component TL				4	2	1
Name /ijkl			4			
Payload TL				4	2	1
Payload			32			
Validator Alg TL				4	2	1
Validator HMAC				4	2	1
KeyId TL				4	2	1
KeyId			2			
Validator Payload TL				4	2	1
Validator Payload (128-bit HMAC)			16			
802.15.4 AES-CCM-128 Auth		16				
802.15.4 FCS		2				
SUBTOTAL	**6**	**54**	**62**	**40**	**20**	**10**
TOTAL 802.15.4 PHY Payload				**156**	**136**	**126**
OVERHEAD				**65%**	**32%**	**16%**

We kept 8 byte fixed header, this is obvious place to save

A 2+2 or 1+1 CCN Encoding with Fixed Header is too large

Overhead = encoding / data (e.g. 40 / 62 = 0.65)

	NDN Encoding OCTETS				
	PHY	Fixed	Data	NDN	1+0
802.15.4 GFSK PHY header	6				
802.15.4 64-bit address		23			
802.15.4 Security header		5			
Data Packet TL				2	1
Name TL				2	1
Name Component TL				2	1
Name /abcd			4		
Name Component TL				2	1
Name /efgh			4		
Name Component TL				2	1
Name /ijkl			4		
Content TL				2	1
Contents			32		
Signature Info TL				2	1
Signature Type TL				2	1
Signature Type			1		
KeyLocator TL				2	1
KeyId TL				2	1
KeyId			2		
Signature Value TL				2	1
Signature (128-bit HMAC)			16		
802.15.4 AES-CCM-16 Auth		16			
802.15.4 FCS		2			
SUBTOTAL	**6**	**46**	**63**	**22**	**11**
TOTAL 802.15.4 PHY Payload				**131**	**120**
OVERHEAD				**35%**	**17%**

Note: No fixed header, no nonce (it's a Data packet)

A 1+1 NDN encoding is too large

Overhead = encoding / data (e.g. 22/63 = 0.35)

Maximum payload, Gain
when changing the encoding while keeping name and crypto bits fixed

	absolute (octets)	relative increase
CCN 2+2	3	
CCN 1+1	23	667%
CCN 1+0	33	43%
NDN 1+1	28	
NDN 1+0	39	39%

Increase = (current – previous)/previous

The case for 1+0 Encoding

- There are very few fields needed. You cannot really fit more anyway.
- Can mix 1+0 with other encodings when need more types or longer lengths (see next slides)
- It saves a lot of bytes.
- Requires a separate specification on packet format, as there are only 4 available "T"s per container in the 1+0 format.

Embedding 1+0 in NDN: Encoding

Approach:

- Reserve some type code space for IoT encoding (four type values)
- Reserve some codes for overflow (announcing length of T)

```
(y = type bit, x = length bit)
001yyyyy <5-bit type> VAR-NUMBER(length)
00111101 2-byte(type) VAR-NUMBER(length)
00111110 4-byte(type) VAR-NUMBER(length)
00111111 8-byte(type) VAR-NUMBER(length)
000xxxxx type 0 length 0xxxxx (5-bit length)
01xxxxxx type 1 length xxxxxx (6-bit length)
10xxxxxx type 2 length xxxxxx (6-bit length)
11xxxxxx type 3 length xxxxxx (6-bit length)
```

Embedding 1+0 in NDN: Pseudocode

```
if (type_val & 0b11100000 == 0b00100000) {
    // VAR-NUMBER type processing
    if (type_val < 0x3D ) {
        type = type_val & 0x1F;
    } else if ( type_val == 0x3D ) {
        // 2-byte VAR-NUMBER type follows
    } else if ( type_val == 0x3E ) {
        // 4-byte VAR-NUMBER type follows
    } else {
        // 8-byte VAR-NUMBER type follows
    }
    // VAR-NUMBER length follows
} else {
    // IOT processing
    type = type_val >> 6;
    length = type_val & 0b0011111;
}
```

Embedding 1+0 in CCN 2+2: Encoding & Pseudocode

```
(y = type bit, x = length bit)
001yyyyy yyyyyyyy xxxxxxxx xxxxxxxx (8K types, 64K length)
000xxxxx type 0 length 0xxxxx (5-bit length)
01xxxxxx type 1 length xxxxxx (6-bit length)
10xxxxxx type 2 length xxxxxx (6-bit length)
11xxxxxx type 3 length xxxxxx (6-bit length)
```

```
if (type_val & 0b11100000 == 0b00100000) {
    type = (uint16_t) type_val << 8 | next_byte;
    // 2-byte length follows
} else {
    // IOT processing
    type = type_val >> 6;
    length = type_val & 0b0011111;
}
```

```
+--------+--------+--------+--------+--------+--------+--------+--------+
|                           PHY HEADER                  |     Control     |
+--------+--------+--------+--------+--------+--------+--------+--------+
|   SN   |       Dest      |     Source      |     Aux. Sec. Hdr      ~
+--------+--------+--------+--------+--------+--------+--------+--------+
~     Aux. Sec. Hdr        |   Ver  |PktType |  Packet Length  |Reserved|
+--------+--------+--------+--------+--------+--------+--------+--------+
|Reserved| Flags  | HdrLen |
+--------+--------+--------+
|01110001|                                     // Content Object TL (49 bytes)
+--------+--------+
         |00001111|                           // 1+0 Name TL (15 bytes)
         +--------+--------+--------+--------+--------+--------+
                  |00000100|   a        b        c        d    |
                  +--------+--------+--------+--------+--------+
                  |00000100|   e        f        g        h    |
                  +--------+--------+--------+--------+--------+
                  |00000100|   i        j        k        l    |
         +--------+--------+--------+--------+--------+--------+
         |10100000|                          // 1+0 Payload TL
         +--------+--------+--------+--------+--------+--------+--------+--------+--------+
         ~                             (32 bytes of payload)                               ~
+--------+--------+--------+--------+--------+--------+--------+--------+--------+--------+
|10000011|                                   // 1+0 ValidatorAlg TL
+--------+--------+--------+--------+--------+
         |01000011|01000010|  (2-byte keyid) | // 1+0 HAMC TL + KEYID TL
+--------+--------+--------+--------+--------+
|11010000|
+--------+--------+--------+--------+--------+--------+--------+--------+--------+
         ~                   16-byte CCN HMAC signature                          ~
+--------+--------+--------+--------+--------+--------+--------+--------+--------+
~                  16-byte 802.15.4 AES-CCM-16 encrypted MAC         ~
+--------+--------+--------+--------+--------+--------+--------+--------+
|  802.15.4 FCS   |
+--------+--------+
```

Conclusions

- The examples stimulate discussion – not absolute judgments on encodings
- 2+2 and 1+1 have a lot of overhead for IoT
- 802.15.4 case:
  - 1+1 formats (CCN and NDN) slightly too large for 32-byte payload with AES-CCM-16
  - 1+0 format works for 32-byte payload
- Graceful overflow: 1+0 format can be combined with existing NDN- and CCN-style encodings