

# Optimal syntax

for packet header compression specification

Julije Ožegović  
University of Split, FESB Split  
[julije.ozegovic@fesb.hr](mailto:julije.ozegovic@fesb.hr)

# Introduction

---

- packet header compression is attractive technique to preserve radio bandwidth in the “last hop” wireless link
- formal specification of compression schemes is introduced recently
- experience is gained through the experimental implementation of the EPIC Lite scheme
- new possibly optimal syntax is proposed

# Compression schemes overview

---

- **CSLIP 1990**: ergonomic response time for TELNET users
  - requirements similar to voice-data integration constrains
  - no robustness
- **DEFLATE 1996**: LZ77 usage for standardized compressed data transfer over Internet, not actually header compression
- **IPHC 1997-1999**: for IP/TCP, introduces concepts
  - context and context identifier (CID)
  - encoding method
  - robustness (twice algorithm and feedback)

# Compression schemes overview

---

- **C RTP 1999**: extension of IPHC for IP/UDP/RTP
- **ROHC 2001**: IETF working group, work in progress
  - **ROHC framework**: packet formats for common transport
  - **profiles**: set of rules to compress particular protocol header
  - **robustness**: r compressor contexts, LSB method
  - **robustness**: three modes of operation, uni and bi-directional
- **EPIC Lite 2002**: profile specification language for ROHC
  - defines implicitly compression - decompression engine

# Header compression specification

---

## Ad hoc specification syntax:

- **CSLIP 1990**: text, C language reference implementation
- **DEFLATE 1996**: text, C samples, pseudo code
- **IPHC 1997-1999**: text, pseudo code
- **CRTP 1999**: text and tables
- **ROHC 2001**: text and tables

# Header compression specification

---

## Formal specification syntax:

- **UHCF 2000**: Unified header Compression Framework, Pearl
- **EPIC Lite 2002**: ABNF notation
- **Extended EPIC Lite 2002**: stack operations hidden
- **Generic notation 2002**: hierarchical packet structure

# UHCF syntax

---

- **UHCF 2000**: Formal specification for CSLIP model
  - Pearl like specification language
  - **field**, **rule** and **action** instructions
- **FIELD instruction**: specify header field
  - length, method, parameter
- **RULE instruction**: specify rules to filter packets out
- **ACTION instruction**: specify action to execute on event
- **UHCF usage**: IP/TCP profile without options presented

# UHCF syntax (cont)

---

```
class IPCompress {
  Compressed_ID 0x0081;
  UnCompressed_ID 0x0083;

  PField protover, fsize=4, ptype=NOCHANGE;
  PField hdrlen, fsize=4, ptype=NOCHANGE;
  PField tos, fsize=8, ptype=NOCHANGE;
  PField totlen, fsize=16, ptype=INFERRED,
    formula=[length];
  PField packetid, fsize=16, ptype=DELTA,
    encoding=VARONETHREE, negatives=DISALLOWED;
  PField fragments, fsize=16, ptype=NOCHANGE;
  .....

  PRule SendAsIP, ruletext=[protover]!=4; # IPv4 only
  StateVar expireTime, type=int, initvalue=0;
  .....
}
```

# EPIC Lite syntax

---

- **EPIC Lite 2002**: Formal specification for ROHC framework
  - provides complete compression system
  - protocol independent
  - formal language used to specify profile
- **ABNF**: Augmented BNF to define new rules
  - **AND** and **OR** of previously defined or primitive rules
  - **stack machine** implicitly defined
- **CHOICE "OR" operator**: rule choice possible
  - choose best rule
  - multiple formats possible
  - use **ID\_bits** to identify format

# EPIC Lite syntax (cont)

---

- **HUFFMAN encoding**: optimal ID\_bits usage
  - calculate probability of format
  - keep **max\_formats** most probable formats
  - use **format sets** for better ID\_bits usage
- **EPIC Lite packet outline**: ID\_bits, then compressed fields

1101100	Compressed Field 1	Compressed Field 2	...	payload...
---------	--------------------	--------------------	-----	------------

# EPIC Lite syntax (cont)

---

**TCP-IP-CO** = **INFERRED-IP-CHECKSUM (IPv4-co-header)**  
**TCP-co-header**  
**CRC (8,100%)**

**IPv4-co-header** = **version**  
**header-len**  
**tos-co**  
**.....**

**version** = **VALUE (4,4,100%)**  
**header-len** = **VALUE (4,5,100%)**  
**tos-co** = **STATIC (99%) | IRREGULAR (6,1%)**  
**.....**

# Extended EPIC Lite syntax

---

- Extended EPIC Lite 2002: stack machine hidden
  - LABEL** method used instead of stack manipulations
  - NEXT\_FIELD** used to access labeled data
  - labels can be reused

```
eg_1 = LABEL(8,a_label)
      .....           ; some_other_fields
      NEXT_FIELD(a_label) ; compress the labelled value
      IRREGULAR(8,100%)
      LABEL(16,a_label)  ; re-use the label
```

# Generic notation

---

- **GENERIC NOTATION 2002**: alternative specification language
  - based on **ABNF** and **EPIC Lite**
  - introduces new compression machine
  - profile split in two parts: header and compression specification
- **HEADER SPECIFICATION**: header fields are defined
  - field as a **variable**
  - only field length is specified, declaration **must** follow header structure
  - field access is random, but impractical for option fields
  - field names are hierarchical, e.g. **IP.version**

# Generic notation (cont)

---

- **COMPRESSION SPECIFICATION**: format of compressed header
  - Huffman coded ID\_bits used to identify format
  - instead of format sets, **subheaders** are introduced
- **SUBHEADER** concept
  - compressed header is **hierarchical** structure of subheaders
  - each subheader includes ID\_bits of its own
- **PROFILE** writing
  - **HUFFMAN** and **P**(robability) methods used
  - ID\_bits generation under control of profile writer
  - options processed with **SUBTREE** and **LIST** methods, complex

# Generic notation (cont)

---

```
tcp      := sport
           .....
           PADDING-T(options, 32, 0, 8)

sport    := BITS(16)
           .....
options  := LIST-0(opt-end, opt-gen, ..... )
           .....

ctcp     := ctcp-flags
           STATIC(tcp.sport)
           STATIC(tcp.dport)
           .....

ctcp-flags := HUFFMAN(pseqnack, ..... )

pseqnack := P(45%, LSB(tcp.seqno, 14, 4096), ..... )
```

# Syntax drawbacks

---

- **UHCF 2000**: no choice operators
  - one format only
  - TCP options not covered
- **EPIC Lite 2002**: explicit usage of stack machine, no field definition
  - profile not readable and (human) error prone
  - complex stack manipulations needed to postpone field access
- **Extended EPIC Lite 2002**: stack machine hidden
  - LABEL method enables field access postponing
  - no major improvement
- **GENERIC NOTATION 2002**: complex and not verified
  - still lacks full field definition concept

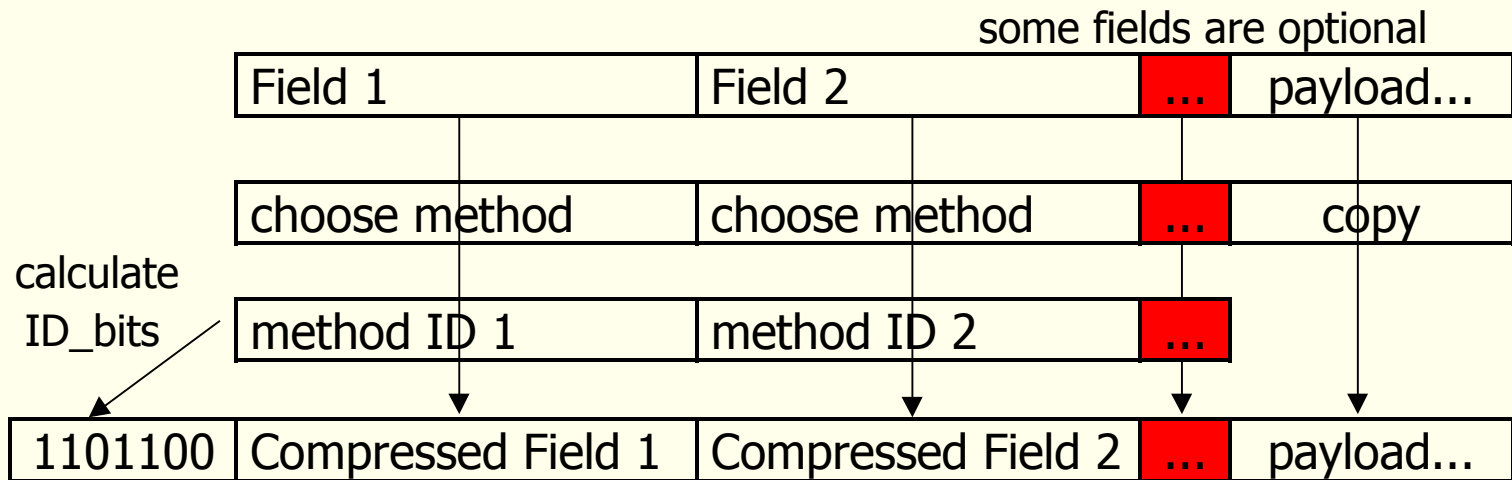
# EPIC Lite FESB implementation

---

- **ABNF USAGE:** application generator or loadable profile
  - **loadable profile** concept adopted
  - outer and inner interpreter implemented
- **OUTER INTERPRETER:** offline processing
  - profile parsing, format generation and probability calculation
  - format sort and discard, ID\_bits calculation
  - data structures generation
- **INNER INTERPRETER:** online processing
  - actual compression and decompression
  - file to file experimental functionality
  - traverses data structures AND packet header
  - makes choices of optimal coding

# EPIC Lite FESB implementation

- **LINEAR PROFILE:** logical approach to header compression
  - **optimized** compression and decompression
  - **simple** profile preprocessing and ID\_bits distribution



# EPIC Lite FESB experience

---

- EPIC Lite: **complete** and **flexible**
  - several test profiles generated successfully
  - interoperability achieved on IP/TCP and IP/UDP RTP flows
- ABNF notation: **problems** in profile writing
  - ABNF freedom prevents formal verification
  - implicit and explicit stack manipulations induce errors
  - order of processing follows header structure
  - OR-ed rules failure require complex stack recovery
- TOOLBOX: implementation specific issues
  - choice can be more or less “intelligent”
  - unc\_fields concept is complex

# Optimal syntax goals

---

- Declare format of original header explicitly
- Random access to all fields except optional ones
- Declare method parameters explicitly
- Declare more parameters in profile header
  - context check rules
  - protocol ID values
- Optimize inter method and rule communication
- Introduce control structures
- Keep data structures simple, no unneeded hierarchy
- Provide safer writing and formal verification

# Optimal syntax proposition

---

- Compression ordering
- Field declarations
- Profile declaration
- Method parameters
- OR-ed rules
- Optional fields
- Variable length fields
- Multiple format sets
- Payload compression

# Compression ordering

---

- **0 orders of freedom:** EPIC Lite
  - fields declared and processed in order of appearance
  - compressed header in order of field processing
  - EPIC Lite offers only field postponing (LABEL or stack)
- **1 order of freedom:** Generic notation
  - fields declared in order of appearance
  - fields accessed randomly
  - compressed header in order of field processing
- **2 orders of freedom:** Optimal syntax
  - fields declared randomly
  - fields accessed randomly
  - compressed header in order of field processing
- **More freedom:** not required

# Field declarations

---

- Declare field: **offset** from beginning of header AND **length**:

```
<field_name> = FIELD(<offset>, <length>)
```

- Fields with predefined values: use **value** for context check

```
<field_name> = FIELD-N(<offset>, <length>, <value>)
```

- Calculated offset: to be used **after** optional fields
  - use available header length **IHL** field:

```
IHL    = FIELD(4, 4)
```

```
ackno  = FIELD(IHL*32+64, 32)
```

# Field declarations (cont)

---

- Optional field: offset from beginning of options:

```
<field_name> = FIELD-O(<offset>, <length>)
```

- General variable: 32 bits, but actual length of data tracked

```
<variable_name> = VARIABLE
```

- Variable initiation:

```
<rule_name> = STORE(<variable_name> , <expression>)
```

# Profile declaration

---

- More parameters: context check, protocol IDs:

```
.....  
CO_packet      = <CO_packet_rule>  
IR_DYN_packet  = <IR_DYN_packet_rule>  
IR_packet      = <IR_packet_rule>  
profile_check  = <profile_check_rule>  
context_check  = <context_check_rule>  
protocol_unc   = <value>  
protocol_comp  = <value>  
<field_name>  = FIELD(<offset>, <length>)  
.....  
<variable_name> = VARIABLE  
.....  
<profile_check_rule> = version protocol  
<context_check_rule> = saddr daddr sport dport
```



# OR-ed rules

---

- **Random access fields:** no stack recovery needed
  - OR operation expects **equal length** branches
  - **different length** branches to be reconsidered
- **OR choice operation:** implementation specific
  - **less intelligent:** simple trying
  - **more intelligent:** keep track of profile probability to avoid discarded profiles
- **Explicit choice:** control structures to be considered
  - give profile writer **explicit choice control**

# OR-ed rules (cont)

---

- **Control structure:** IF/ELSEIF/ELSE/ENDIF as simple CASE
  - explicit or ID\_bit choice signaling possible
  - **ID\_bit signaling** chosen for robustness
  - test for **true** or **value**

```
<rule_name> = IF(<field>, <rule_true>, <rule_false>)
```

```
<rule_name> = IF(<field> = <value>, <rule_true>,  
                <rule_false>)
```

```
<rule_name> =  
  IF(<field> = <value>, <rule_true>)  
  ELSEIF(<field> = <value>, <rule_true>)  
  ELSEIF(<field> = <value>, <rule_true>)  
  .....  
  ELSE(<rule_false>)  
  ENDIF
```

# Optional fields

---

- **Format signaling**: problem with optional fields
  - options present or not in arbitrary order
  - large number of possible formats
- **EPIC Lite**: **unc\_fields** concept, U method provided
  - header always of the same length (format)
  - optional fields packed **after** the fixed part of format
  - **presence** and **order** fields included in fixed part
  - **low** compression efficiency when no options are present
- **Generic notation**: **subheader** concept
  - **complex, not verified yet**

# Optional fields (cont)

---

- EPIC Lite concept kept
- Options length available: process until the end of list
  - length calculated, e.g.  $(data\_offset - 5) * 32$
  - `<field>` contains option identifier to be compared with `<value>`
  - FIELD-O fields should be used, LIST calculates relative offset
  - `<v_order>` and `<v_present>` variables are updated

```
<rule_name> = LIST(<length>, <field>,  
                  <v_order>, <v_present>,  
                  1*(OPTIONAL(<rule>, <value>))  
                  .....)
```

# Optional fields (cont)

---

- Options end code available: process until the **end** code
  - declare **end** code

```
<rule_name> = LIST-C(<end>, <field>,  
                    <v_order>, <v_presence>,  
                    1*(OPTIONAL(<rule>, <value>)))
```

- Options ID not used: use length, e.g. SACK blocks

```
<rule_name> = LIST(<length>, <v_order>, <v_presence>,  
                  1*(OPTIONAL-N(<rule>)))
```

# Variable length fields

---

- EPIC Lite concept: **unc\_fields**
  - variable length fields pushed **after** the fixed part of the header
  - **<v\_length>** can be field or variable, compressed **later**

```
<rule_name> = UNCOMPRESSED(<v_length>, <field>)
```

- **unc\_fields** redundant: can be **part of compressed** header

# Multiple format sets

---

- EPIC Lite concept kept: choice algorithm implementation specific
  - choice result stored in variable

```
<rule_name> = FORMAT(<v_selection>, <rule>, 1*<rule>)
```

# Payload compression

---

- **Payload as packet field:** concept of packet compression
  - length can be calculated, compress next bits
  - provide appropriate method
  - store compressed length for later compression

```
<rule_name> = LZ77 (<length>, <v_length>)
```

- Payload can be declared as field with variable length

```
<rule_name> = LZ77 (<field>, <length>, <v_length>)
```

# Conclusion

---

- Optimal header compression specification is proposed, based on EPIC Lite concept
- Optimal compression order is provided by random access to all header fields except optional ones
- Compressed header format is defined by compression order, which is convenient for decompression and supports linear profile
- Methods are redefined to use random access fields
- Control structures are proposed with possible variable length branches

# Conclusion (cont)

---

- Optional fields encoded using EPIC Lite concept
- EPIC Lite unc\_fields stack is possibly redundant
- Declared goals are satisfied, human readability is improved
- **Future work:** proposed syntax to be elaborated in full standard
- Formal and experimental validation is needed
- Optimal balance between efficiency and complexity should be achieved

# References

---

- [1] Bormann C, et al, 2001, "Robust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed", RFC 3095, 2001.
- [2] Casner S, Jacobson V, "Compressing IP/UDP/RTP Headers for Low-Speed Serial Links", RFC 2508, 1999.
- [3] Cizmic, M., Vodopija, T., Ozegovic, J.: "EPIC Lite offline processing", SoftCOM 2002.
- [4] Crocker D, et al: "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, 1997.
- [5] Degermark, M., Engan, M., Nordgren B, Pink S, "Low-loss TCP/IP Header Compression for Wireless Networks" ACM/Baltzer Journal on Wireless Networks, vol 3, no 5, 1997.
- [6] Degermark M, Nordgren B, Pink S.: "IP Header Compression", RFC 2507, 1999.
- [7] Deutsch, P. "Deflate", RFC 1951, 1996.
- [8] Jacobson, V. "Compressing TCP/IP Headers for Low-Speed Serial Links", RFC 1144, 1990.
- [9] Liao, H., Zhang, Q., Zhu, W.: "Generic Header Compression Notation for ROHC", draft-liao-rohc-notation-00.txt, 2002.
- [10] Lilley, J., Yang, J., Balakrishnan, H., Seshan, S.: "A unified header compression framework for low-bandwidth links", Proceedings of the sixth annual international conference on Mobile computing and networking, p.131-142, Boston, 2000.
- [11] Mornar, M., Pezelj, A., Ozegovic, J.: " Testbed for header compression implementation ", SoftCOM 2002.

# References

---

- [12] Price, R., Hancock, R., McCann, S., Surtees, A., Ollis, P., West, M.: "Framework for EPIC-LITE", draft-ietf-rohc-epic-lite-01.txt, 2002.
- [13] Price, R., Surtees, A., West, M.: "A Formal Notation for Header Compression", draft-west-rohc-formal-notation-00.txt, 2002.
- [14] Price, R., Surtees, A., McCann, S., West, M., Hancock, R., Findlay, D.: "EPIC Provably Optimal Format Encoding for Compression in the Internet", SoftCOM 2002.
- [15] Price R, et al, "Signalling Compression (SigComp)", RFC 3320, 2002.
- [16] Stula, M., Vidjak, L., Ozegovic, J.: " Program structures for EPIC-LITE experimental implementation ", SoftCOM 2002.
- [17] Ziv J., Lempel A., "A Universal Algorithm for Sequential Data Compression", IEEE Transactions on Information Theory, Vol. 23, No. 3, pp. 337-343.