

# EPIC Lite offline processing

**Mia.Cizmic@kron.hr**

**Tijana.Vodopija@siemens.hr**

**Julije.Ozegovic@fesb.hr**

# Introduction

---

- EPIC Lite scheme [1] is experimentally implemented on University of Split, FESB Split
- the experience gained for the offline phase will be presented, and improvements proposed
- given improvements are related to decreasing time and memory consumption in offline phase, and have no impact on compression ratio

# EPIC Lite

---

- provides protocol independent specification system for header compression under ROHC [2] framework
- usage is expected in the wireless network area, where error rates and round trip times are high
- includes a simple human readable profile language [3], used to describe header properties in high-level manner
- profile assigns one or more compression methods to each field in the protocol stack to be compressed

# Example of profile writing

---

```
Header      =  Field_A
              new_rule
              Field_C

Field_A     =  STATIC(90%) | IRREGULAR(2,10%)
new_rule    =  Field_B1
              Field_B2

Field_B1    =  STATIC(100%)
Field_B2    =  VALUE(4,1,80%) | VALUE(4,0,20%)
Field_C     =  STATIC(70%) | LSB(8,-1,15%) | LSB(16,1,15%)
```

# EPIC Lite two phases

---

- Offline phase - converts the input profile into one or more sets of compressed header formats. It is run once. Results are stored at compressor and decompressor.
- Online phase - repeated for each packet in the stream. Provides mechanism which apply appropriate encoding method to each header field.

# Offline phase

---

- profile provides multiple header formats for a protocol stack, depending on the choice of toolbox encoding methods for all fields
- each format has a different probability (calculated as product of basic probabilities)
- offline computes only **max\_formats** most probable header formats to be kept and assigns them appropriate indicator flags based on Huffman coding
- indicator flags are in online phase put in front of compressed header to communicate to the decompressor which header format had been used

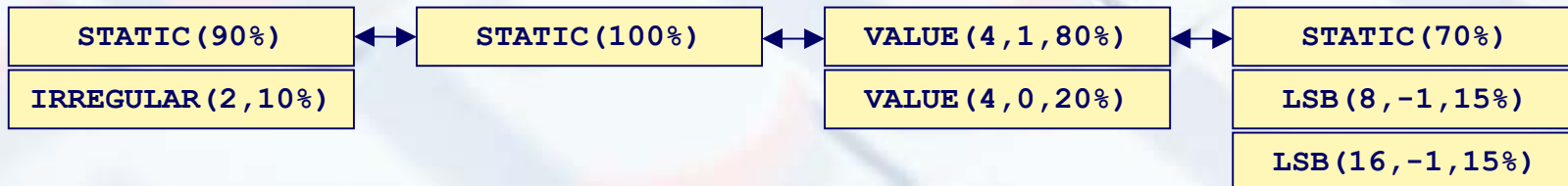
# FESB's offline phase

```
Header      = Field_A
              new_rule
              Field_C

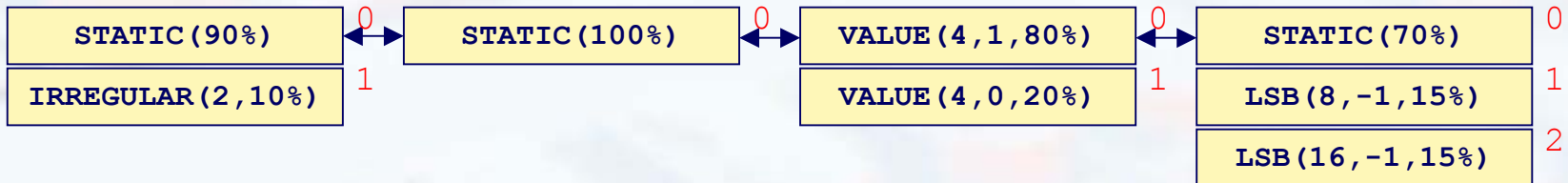
Field_A     = STATIC(90%) | IRREGULAR(2,10%)
new_rule   = Field_B1
              Field_B2

Field_B1    = STATIC(100%)
Field_B2    = VALUE(4,1,80%) | VALUE(4,0,20%)
Field_C     = STATIC(70%) | LSB(8,-1,15%) | LSB(16,1,15%)
```

provides extra functionality →  
transforms tree-structured  
profile to linear form [4]



# Benefits from linear profile



- compression is, in fact, tree traversal where field is encoded only when toolbox method is reached → inner tree nodes are redundant, consuming memory and processing time
- it's easy to implement matching between an integer string and header format:

1010



IRREGULAR(2,10%)

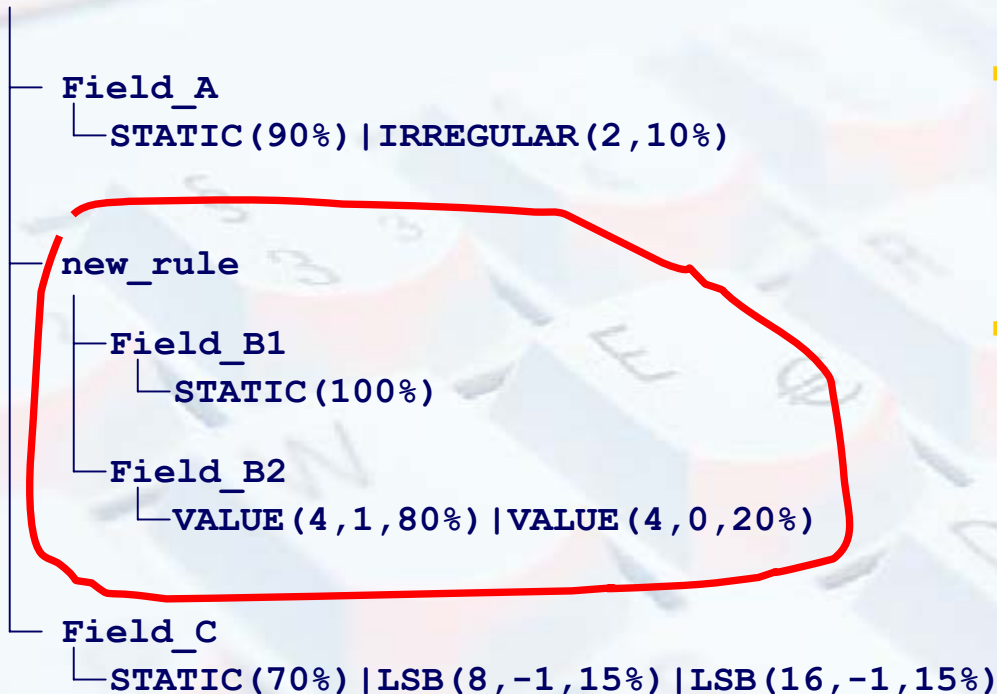
STATIC(100%)

VALUE(4,0,20%)

STATIC(70%)

# Computing `max_formats` formats

## Header



- tree-recursive algorithm [1]
- each step involves list of items computed in earlier step (for lower depth branch)
- when new list exceeds the `max_formats` number, all items with the probability lower than last item probability are discarded


# Why descending order?

---

To decrease the number of operations since:

- to compute first `max_formats` items, it is necessary to have probability-based descending sorted lists of items
- lists are kept in ascending order by [1] → they must be reversed to descending order before discarding procedure which is called at each step → unnecessary swapping operations
- items enter the lists in the order they were written in the profile (dominantly in probability-based descending order) → list is already partially sorted

# Interoperability & descending sort



prob(%)	formats derived by sorting		Huffman codes
	ascending	descending	
0.3	1011	1012	11011111
0.3	1012	1011	11011110
1.2	1001	1002	1101110
1.2	1002	1001	1101101
1.4	1010	1010	1101100
2.7	0011	0012	110101
2.7	0012	0011	110100
5.6	1000	1000	1100
10.8	0001	0002	1011
10.8	0002	0001	1010
12.6	0010	0010	100
50.4	0000	0000	0

- FESB's implementation kept ascending sort for compatibility → proposal the descending sort to enter the standard

# Pre-processing profile

---

- Four parameters that fully describe header format:
  - header format identification
  - compressed header size (in bits)
  - length and
  - value of assigned Huffman code
- those information can be pre-processed on any platform and appended to the profile

# Pre-processed information format

---

```
current_set := 0
```

1011	10	8	DF
1012	18	8	DE
...	...	...	...
0000	0	1	0

```
current_set := 1
```

...	...	...	...
-----	-----	-----	-----

# Conclusion

---

- linear structure of profile → to optimize online operations (compression and decompression) → allows integer string format representation
- descending sort → to spare some item swapping → offline phase can be less time consuming
- pre-processing profile → to spare time and avoid miscalculating formats → offline processing in terminal equipment should only parse BNF file and appended format information to build internal data structures

# References

---

- [1] R. Price, R. Hancock, P. Ollis, A. Surtees, M. West, "*Framework for EPIC-lite*", draft-ietf-rohc-epic-lite-01.txt (work in progress), February 2002
- [2] C. Bormann, C. Burmeister, M. Degermark, H. Fukushima, H. Hannu, L. Jonsson, R. Hakenberg, T. Koren, K. Le, Z. Liu, A. Martensson, A. Miyazaki, K. Svanbro, T. Wiebke, T. Yoshimura, H. Zheng, "*Robust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed*", RFC 3095, July 2001
- [3] D. Crocker, P. Overel, "*Augmented BNF for Syntax Specifications: ABNF*", RFC 2234, November 1997
- [4] L. Viđak, M. Štula, J. Ožegović "*Program structures for EPIC-Lite experimental implementation*", SoftCOM paper, October 2002
- [5] M. Čizmić, T. Vodopija, J. Ožegović "*EPIC Lite offline processing*", SoftCOM paper, October 2002

- all other references from [5]