GOE FEC schemes
<draft-roca-rmt-goe-fec-01>
<draft-roca-rmt-goe-ldpc-00>

IETF83, March 26th, 2012, Paris

V. Roca, A. Roumy (Inria)
B. Sayadi (ALU-BL)

Alcatel·Lucent

informatiques mathématiques

Outline

1. the two goals for GOE schemes

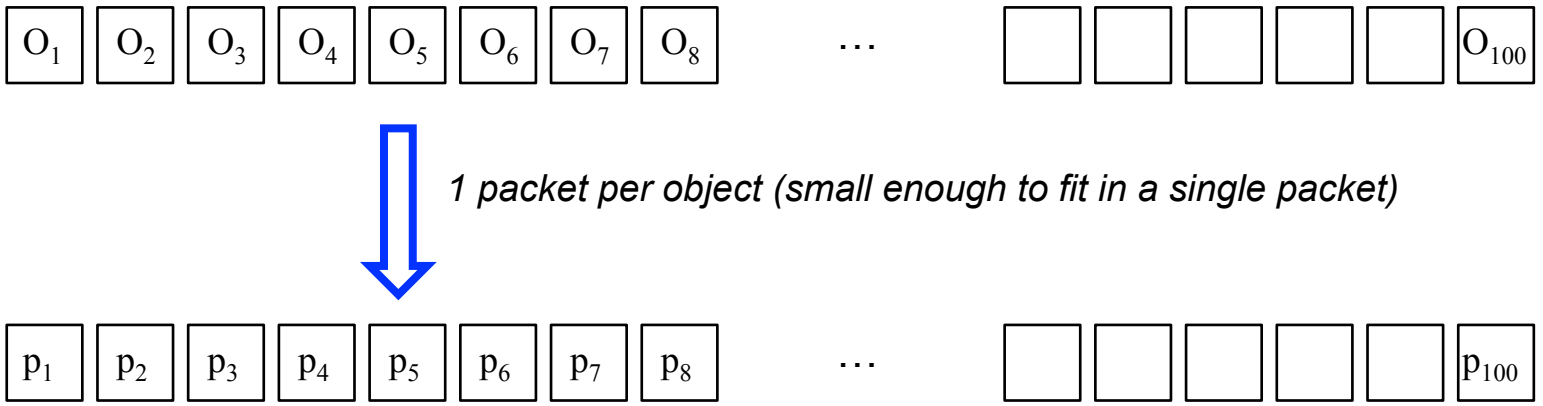
2. Generalized Object Encoding (GOE)
   - **the idea**
   - **a few key results**

Goal 1: provide Unequal Erasure Protection

- with other FEC schemes, all symbols of an object are equally protected…
- UEP is sometimes needed
  - **even with file transfers**
- can be achieved in 3 different ways
  1. **thanks to UEP aware FEC codes**
     - dedicated FEC codes
  2. **thanks to UEP aware packetization** ← **UOD**
     - keep standard FEC codes
  3. **thanks to UEP aware signaling** ← **GOE**
     - keep standard FEC codes

Goal 2: protect a bundle of small files

- imagine you have 100 files of 100 bytes each…
  - **sending (e.g.) twice each packet is not efficient…**
    - neither in terms of protection
    - nor flexibility (code rate is one of {1/2, 1/3, 1/4...})



*send each packet **twice** ⇒ code rate = ½*

… and pray for one of the two packets of each object to be received!

Goal 2: bundle of small files… (cont')

- can be solved in two different ways

1. **thanks to bundle aware packetization** ← **UOD**
2. **thanks to bundle aware signaling** ← **GOE**

Outline

1. the two goals for GOE schemes
2. Generalized Object Encoding (GOE)
   - **the idea**
   - **a few key results**

Generalized Object Encoding (GOE)

- GOE is a pure **signaling** proposal
  - no new FEC code *…but dedicated GOE FEC schemes*
  - no specific packetization *…1 symbol = 1 packet*
  - what GOE I-D does is:
    - **explain what happens to original objects**
    - **explain how Generalized Objects (GO) are created**
    - **explain additional signaling**

and that's all…

GOE in 3 slides 1/3

- **explain what happens to original objects**
- explain how Generalized Objects (GO) are created
- explain additional signaling

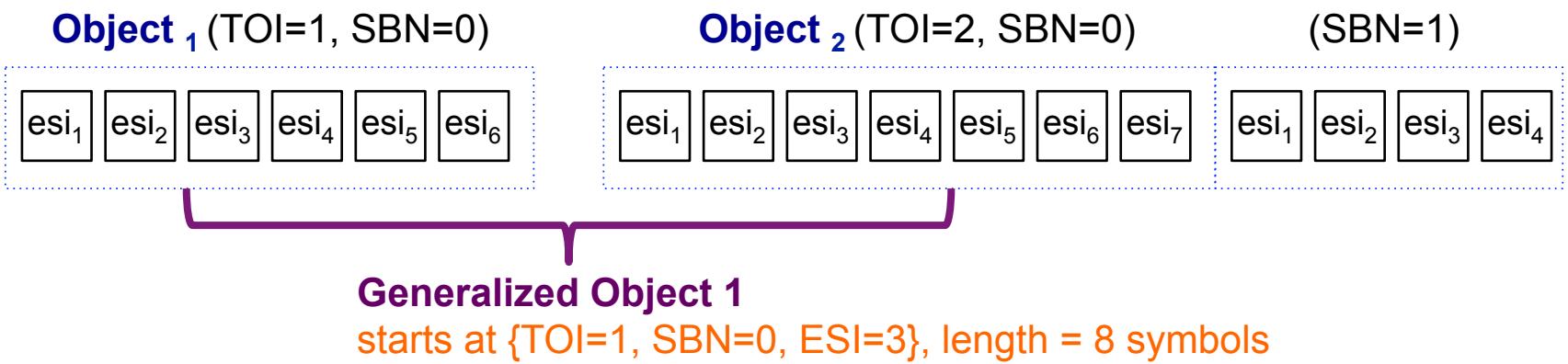
- use a “No-Code FEC” Scheme
  - choose the same symbol size **for all objects**
  - manage TOI in sequence for all objects that need to be considered together (if applicable)
  - **“No-Code FEC” encode** each object
  - **send** “No-Code FEC” encoded symbols
    - **they are source symbols**
  - **nothing new**, FLUTE/FCAST signaling is used as usual

GOE in 3 slides… 2/3

- explain what happens to original objects
- **explain how Generalized Objects (GO) are created**
- explain additional signaling

create GO(s) on top of source objects

- **identify the 1st source symbol of a GO**
  - use the {TOI, SBN, ESI} provided by No-Code FEC encoding
- **identify the number of symbols of a GO**
  - they possibly belong to different objects, it's not an issue



GOE in 3 slides… 3/3

- explain what happens to original objects
- explain how Generalized Objects (GO) are created
- **explain additional signaling**

- signaling aspects
  - **assign a new TOI for each GO**
    - to be easily distinguished from original objects
  - **same FEC Payload ID as original FEC scheme**
    - however only repair symbols are sent
  - **dedicated FEC OTI (carried in EXT_FTI or FLUTE FDT Inst.)**
    - carry the GOE specific metadata
    - identifier for initial source symbol + number of symbols

GOE signaling example

- example: EXT_FTI for GOE Reed-Solomon over GF(2^8)

GOE specific no change

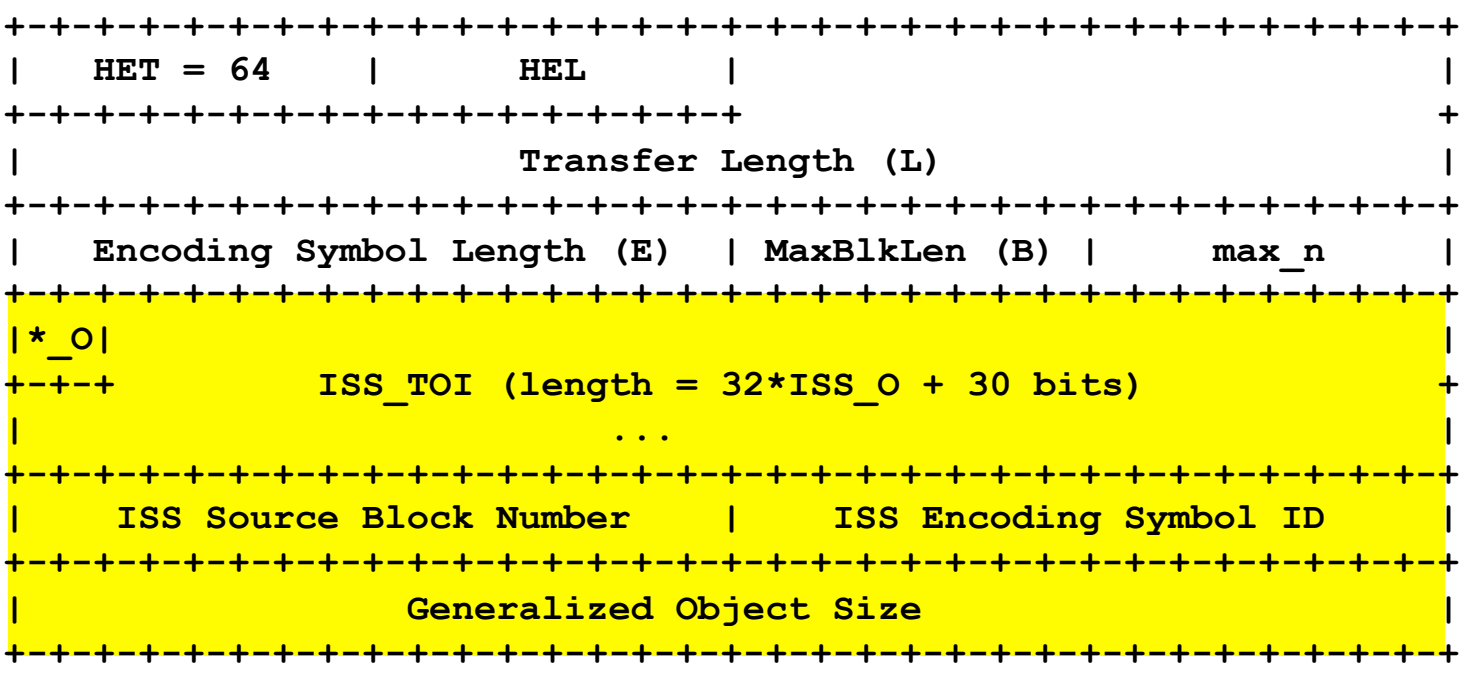
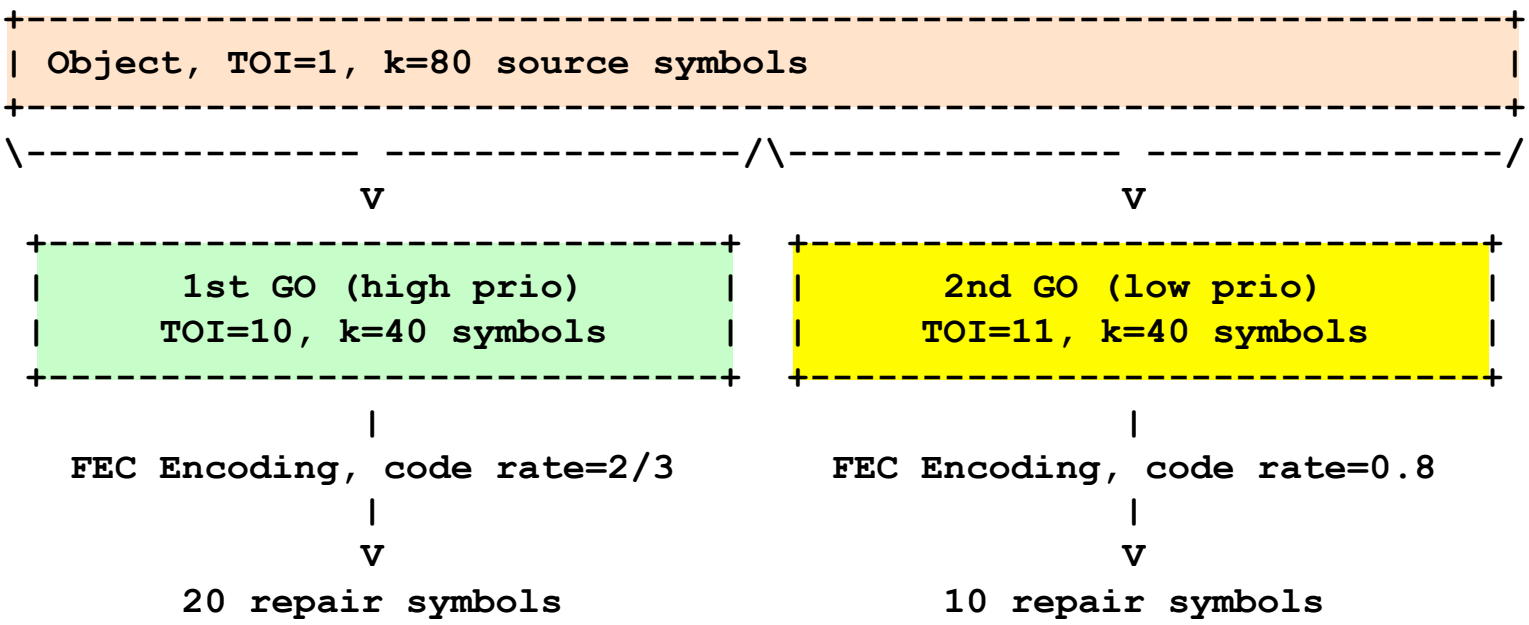


Figure 2: EXT_FTI Header Format with FEC Encoding ID XXX

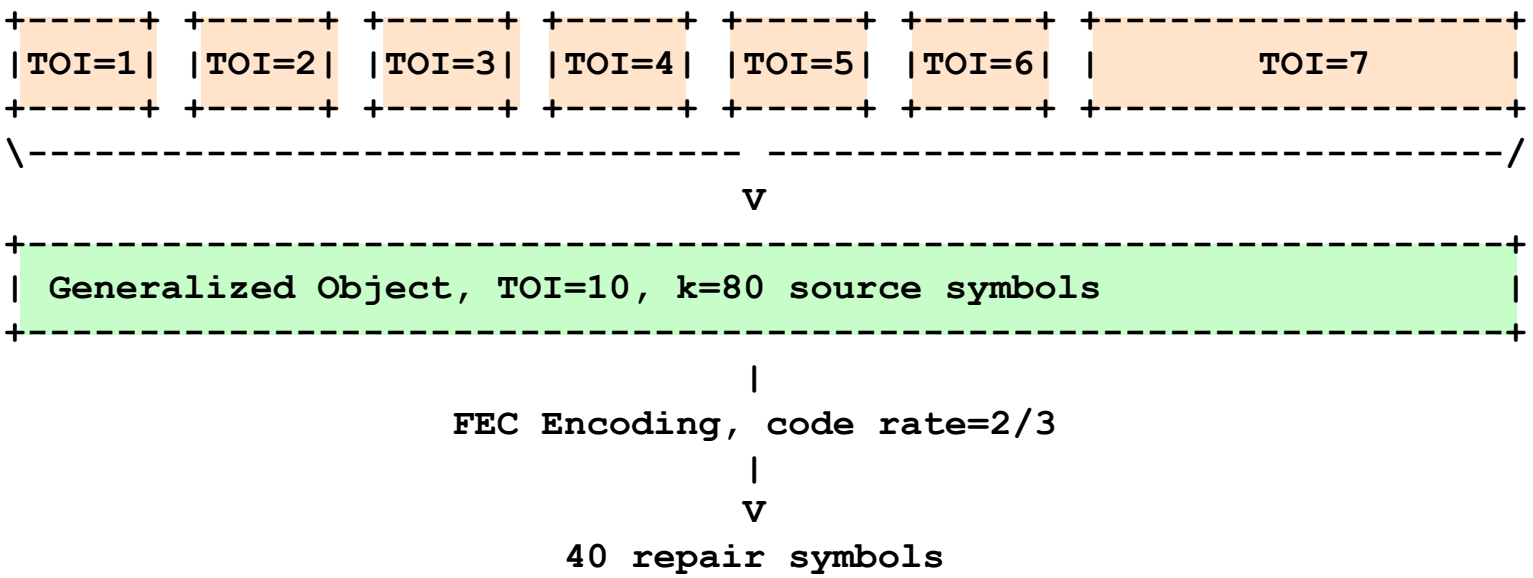
How does GOE address goals 1?

- goal 1: UEP
  - here GO == “subset of a file of a given priority”
  - assign different target code rates to each GO



How does GOE address goals 2?

- goal 2: file bundle
  - here GO == “whole set of files in the bundle”
  - assign the desired code rate to the GO



Does it work? Yes!

- GOE is simple
  - **the "object" ⇔ "GO" mapping is quite natural**
    - … even if it requires some logic to implement it
  - **initialization is trivial**

- GOE is compatible with **all** FEC schemes
  - **GOE Reed-Solomon for GF(2^8) available**
  - **GOE LDPC-Staircase available**
  - **adding others is trivial**

- GOE is backward compatible
  - **a receiver that has no GOE-aware FEC scheme…**
    - can take advantage of "No-Code source symbols"
    - silently drops all "GOE repair symbols" (different TOI and LCT codepoint)

Comparison… (cont')

- GOE is efficient [RRSI11]
  - **we proved [RRSI11][RRS12] that GOE (uniform interleaving) and UOD/PET feature the same UEP protection**
    - no difference, sometimes GOE performs the best, sometimes it's the opposite
  - **less predictable than UOD/PET**
    - is it really an issue?

- GOE features a high flexibility
  - **GOE can be optimized for specific use-cases, by changing the packet transmission order**
    - e.g. to reduce peak memory requirements and decoding delay of high priority GO, while smoothing processing load
    - trade-off to find with robustness in front of erasure bursts

Next steps?

- next steps?
  - **continue standardization within RMT? In TSVWG? As an individual submission?**

- references

[RRSI'11]

A. Roumy, V. Roca, B. Sayadi, R. Imad, *"Unequal Erasure Protection (UEP) and Object Bundle Protection with a Generalized Object Encoding Approach"*, INRIA Research Report 7699, July 2011.

http://hal.inria.fr/inria-00612583/en

[RRS12]

A. Roumy, V. Roca, B. Sayadi, ***"Memory Consumption Analysis for the GOE and PET Unequal Erasure Protection Schemes"***, IEEE International Conference on Communications (ICC'12), June 2012.
http://hal.inria.fr/hal-00668826/en/

Alcatel·Lucent

informatiques mathématiques