The GOE FEC schemes
<draft-roca-rmt-goe-fec-00>
&
UOD-RaptorQ vs. GOE

IETF81, 29th July 2011, Québec City

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

Alcatel·Lucent

informatiques mathématiques
Inria

- This presentation is a summary…
- For the details, see:

[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).

Outline

1. the two goals for UOD and GOE schemes

2. close up on UOD
   - why we think this is not a good practical solution

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

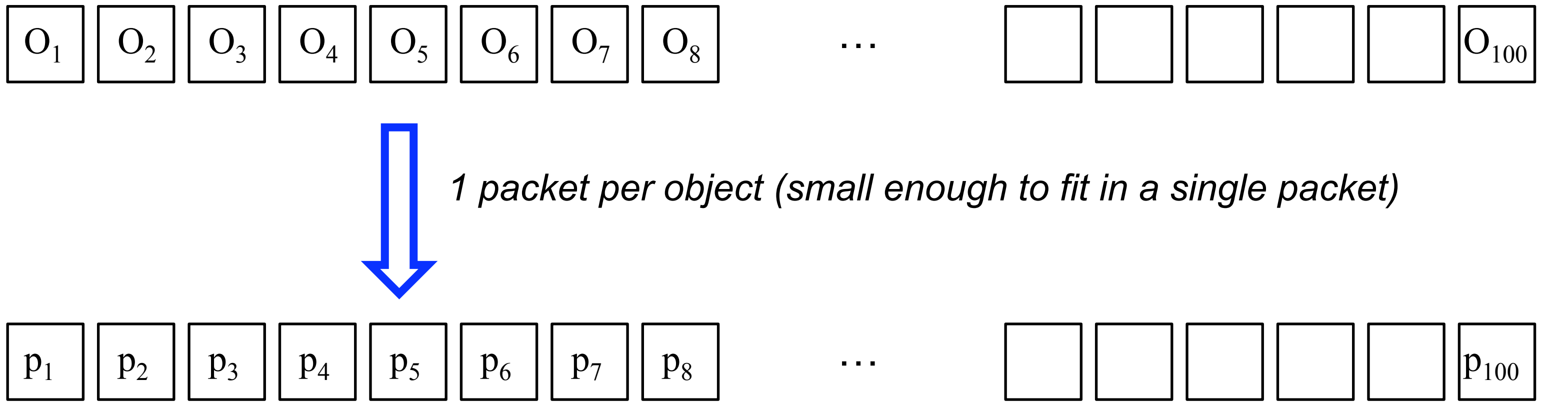
Goal 1: provide Unequal Erasure Protection

- with other FEC schemes, all symbols of an object are equally protected…
- UEP is sometimes needed
    - ❍ **even with file transfers (e.g. file containing scalable video)**
- 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**

- **NB: forget upper-level solutions (e.g. submit a tar archive)**
  - objects may be produced on the fly, they are not necessarily files in a hierarchy of directories

Outline

1. the two goals for UOD and GOE schemes

2. close up on UOD
   - **why we think this is not a good practical solution**

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

UOD (Universal Object Delivery using RaptorQ)

- UOD is a UEP-aware packetization technique
  - inherits from PET [PET96] its packetization mechanism

    each packet is an aggregate of symbols coming from the various "objects"

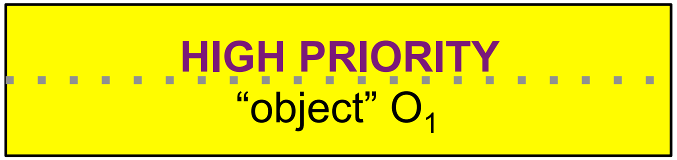
    - **we'll see what "object" means later on**

    - **let's look a bit more at the details…**

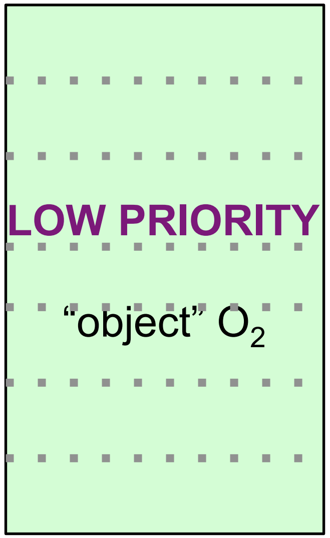
[PET96]

A. Albanese, J. Blomer, J. Edmonds, M. Luby, M. Sudan, "Priority encoding transmission", IEEE Trans. on Information Theory, Vol. 42 Issue 6, Nov. 1996.

UOD sender example: part 1



ex: segmented into 2 “large” symbols



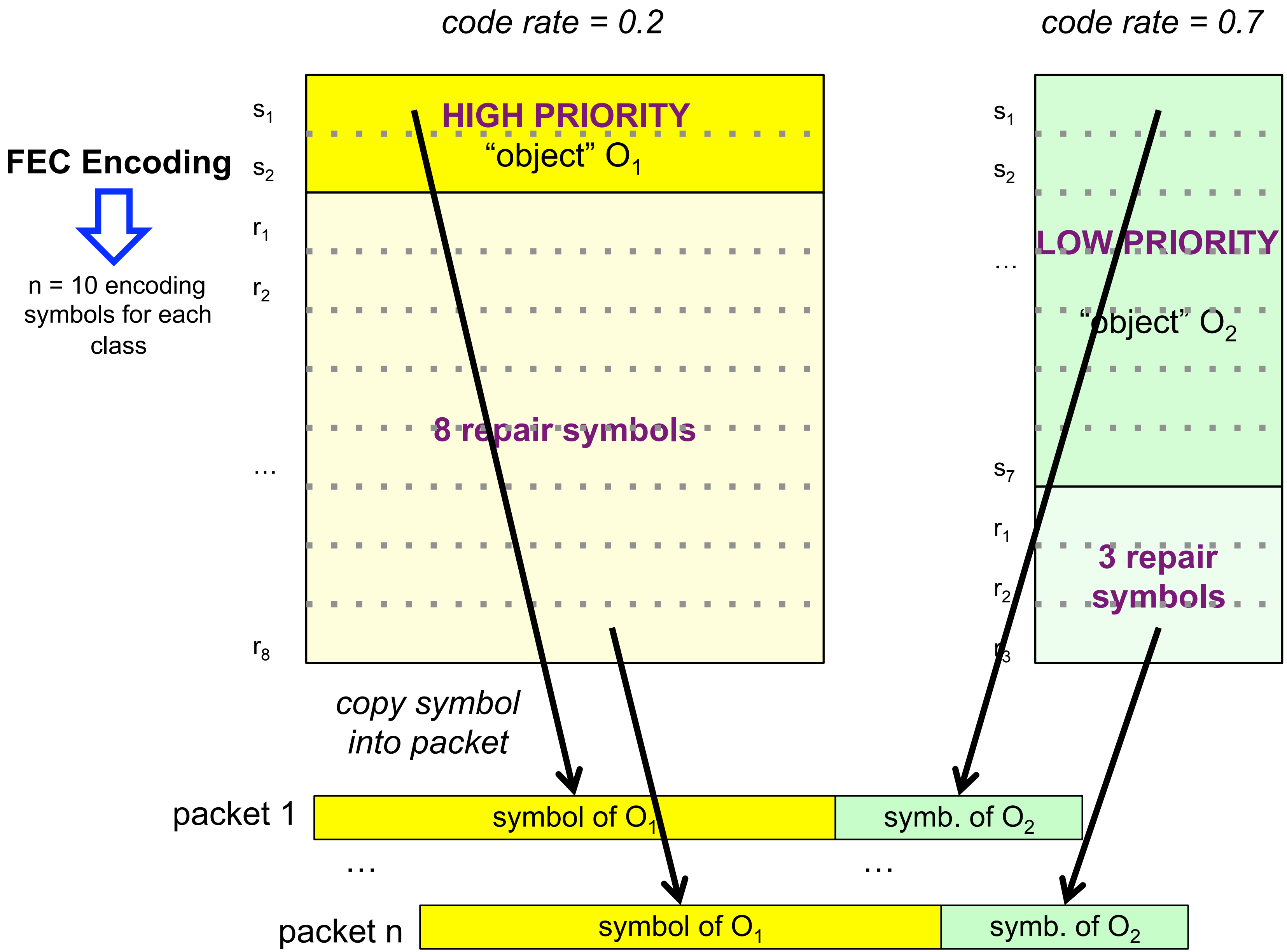
ex: segmented into 7 “small” symbols

Given:
- **2 objects of different priority**
- **target packet size**
- **target code rate for each object**

Calculate (see [PET96]):
- **n, number of packets**
- **number of symbols for each object**
- **symbol size for each object**

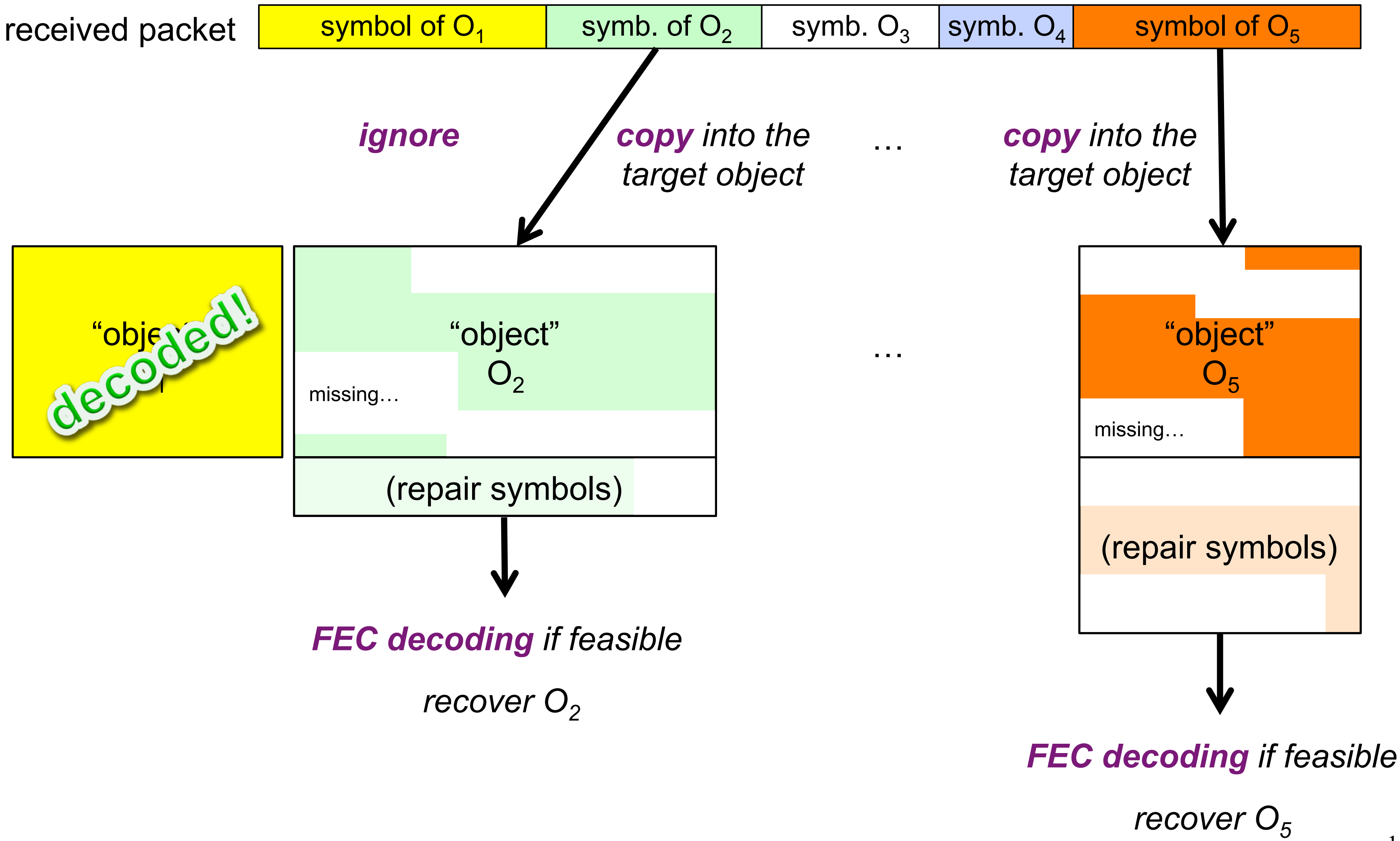
NB: due to rounding effects:
- **the actual packet size is ≤ target**
- **the actual code rate of each object is ≥ target**

UOD sender: part 2, FEC + packet creation



UOD receiver example:

Packet processing at a receiver



How UOD addresses goals 1 and 2

- goal 1: UEP
  - ❍ **here "object" == "subset of a file of a given priority"**
  - ❍ **assign different target code rates to each object**

- goal 2: file bundle
  - ❍ **here "object" == "file"**
  - ❍ **each packet contributes to each object decoding**
    - since each packet contains a symbol of each encoding object

UOD analysis

- inherent **complexity** due to its packetization
  - **each incoming packet MUST be processed as long as there's at least one non decoded object**
    - with GOE, a receiver does not look inside packets for decoded/undesired objects ☺
  - **extra memory copies to/from packets**
    - otherwise memory consumption would be too high
    - no such burden with GOE ☺
  - **with a bundle of 100 objects, you perform 100 FEC encodings and 100 FEC decodings**
    - GOE schemes need only 1 ☺
  - **understanding UOD is challenging**
    - to the complexity of PET it adds the complexity of UOSI and RaptorQ features (sub-symbols/blocks, Al alignment)
    - understanding GOE is a matter of 5mn ☺

UOD analysis… (cont')

- UOD is far too **inflexible**
  - **symbol size is determined by {D, object sizes, target code rates, target packet size, AI}**
    - e.g. with D=255 objects, 1024 byte packets, you have no choice but using 4 byte long symbols!!!
    - with GOE, this size usually corresponds to the PMTU, but other choices are possible too, up to the CDP ☺
  - **a small symbol size has significant impacts on decoding complexity**
    - it increases the number of symbols in a block, and the size of the linear system a receiver has to decode!
    - big impact on the Gaussian elimination scheme described in Raptor/RaptorQ RFC!
    - with GOE, the number of symbols is kept minimum, as well as the linear system size ☺

UOD analysis… (cont')

- **NB: error in the I-D**
  - saying the symbol size is determined by the CDP is wrong. It's determined by the UOD scheme, using a specific algorithm that should be described, even if it is complex

UOD analysis… (cont')

- certain situations are not well addressed
  - **UOD bundle example at IETF80 and add a small file**
    - 32 files of size 32 KB, and 1 file of size 10 bytes
    - target code rate ½ for all files, target packet size is 1 KB
    - it follows there are n = 2049 encoding packets

object size	# source symbols	symbol size	target code rate	actual code rate	target pkt size	actual pkt size
32 KB	1171	28 B (32 is too large)	0.5	**0.571**	1024 B	**900 B**
10 byte	3	4 B		**0.00146**		

less protected

protection far too important ☹

sub-optimal packet size ☹

UOD analysis… (cont')

- from a situation where all targets were perfectly achieved
    - see bundle example at IETF80
- **…adding a single small file can have catastrophic consequences ☹**
- reason
    - **AI=4 bytes is the minimum symbol size.**
    - **If the object sizes differ significantly, UOD cannot fill each packet while complying with all the targets**
        - it would require a finer, bit-level, AI granularity

- to summarize

> - **UOD/PET is an excellent idea on the paper...**
> - **…but I wouldn't recommend its use for practical realizations**

Outline

1. the two goals for UOD and GOE schemes

2. close up on UOD
   - why we think this is not a good practical solution

3. 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

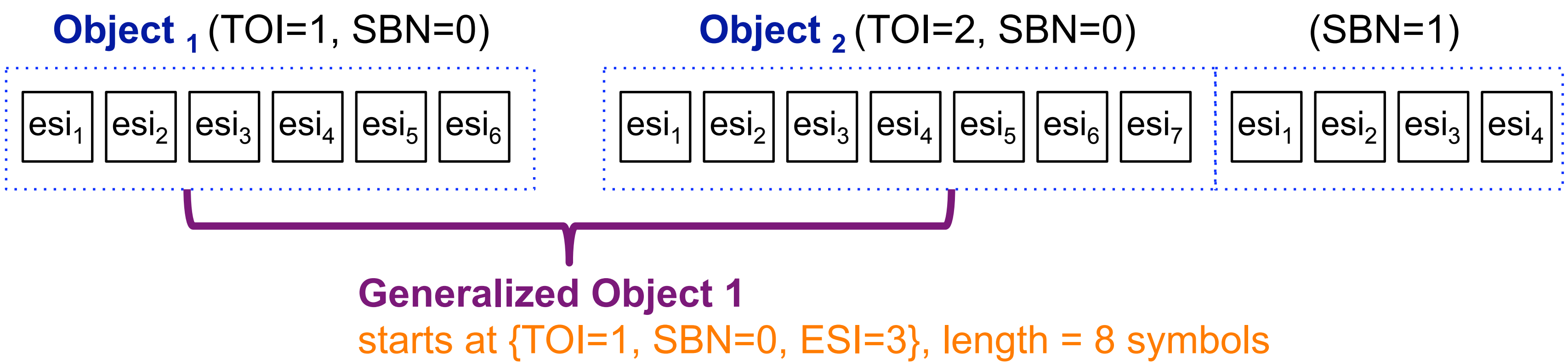
- **explain what happens to original objects**
- **explain how Generalized Objects (GO) are created**
- **explain additional signaling**

- use a No-Code FEC Scheme
  - choose a symbol size **valid for all objects**
  - manage TOI in sequence for all objects
  - No-Code FEC encode each object
  - **send** No-Code encoded symbols
  - nothing new, FLUTE/FCAST signaling is 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 "Generalized Objects" (GO) on top of it
  - **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
  - **dedicated FEC OTI (carried in EXT_FTI or FLUTE FDT Inst.)**
    - carry the GOE specific metadata
    - identifier for initial source symbol + number of symbols
  - **same FEC Payload ID as original FEC scheme, with restrictions on valid ESI**
    - …since only repair symbols are sent

Comparison

- GOE is simple
  - the “object” ⇔ “generalized object” mapping is quite natural
    - … even if it requires some logic to implement it
  - initialization is trivial unlike UOD/PET

- GOE is compatible with all FEC schemes
  - GOE Reed-Solomon for GF(2^8) available
  - GOE LDPC Staircase proposal to come...

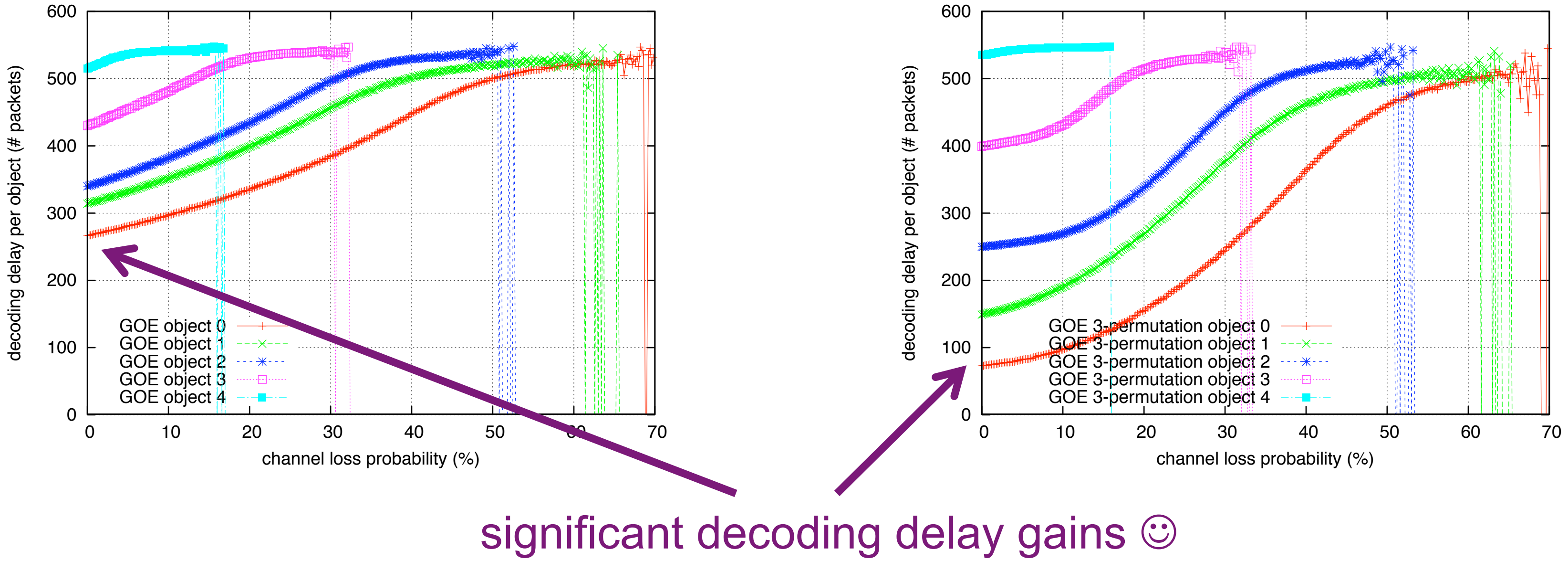
- GOE is backward compatible
  - a receiver that has no GOE-aware FEC scheme…
    - can take advantage of “No-Code source symbols”
    - silently drops all “GOE repair symbols” (different TOI and LCT codepoint)

Comparison… (cont')

- GOE is efficient [RRSI11]
  - **less predictable than UOD/PET**
    - is it really an issue?
  - **same UEP protection as UOD/PET in general**
    - no major difference, sometimes GOE performs the best, sometimes it's the opposite
  - **less processing at a receiver than UOD/PET**
    - no "deep packet processing" unlike UOD/PET
- these features are easily controlled by the sender
  - **GOE can be optimized for specific use-cases**
    - e.g. to reduce peak memory requirements, decoding delay of high priority GO, while smoothing processing load
    - trade-off to find between robustness in front of erasure bursts and gains

Comparison… (cont')

❍example: from "uniform interleaving" to a "3-permutation"



- all details in [RRSI'11]
  - ❍compares PET/UOD versus GOE
  - ❍n-truncated negative binomial distribution model (PET+GOE)
  - ❍theoretical + simulation results for
    - decoding delay
    - number successful decodings
    - max. memory consumption
    - number packets processed

Next steps?

- we have use-cases that need GOE
  - continue standardization within RMT? In TSVWG? As an individual submission?
  - our intent:
    - split current I-D into "GOE FEC Scheme Concept"
    - …and "Reed-Solomon for GF(2^8) GOE FEC Scheme" I-D
    - add an "LDPC-Staircase GOE FEC Scheme" I-D

- 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).

[PET96]

A. Albanese, J. Blomer, J. Edmonds, M. Luby, M. Sudan, "Priority encoding transmission", IEEE Trans. on Information Theory, Vol. 42 Issue 6, Nov. 1996.

Alcatel·Lucent

informatiques mathématiques Inria