belledonne communications.

ENHANCING THE VIDEO CALL EXPERIENCE WITH FLEXIBLE FORWARD ERROR CORRECTION

Jehan Monnier & Flore Harlé

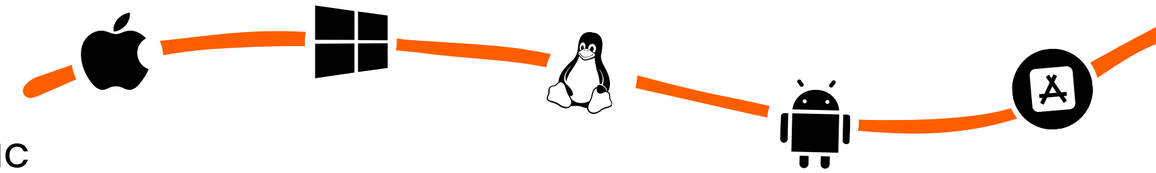
FOSDEM 2024

linphone

open source VOIP project

Quick intro

Linphone



- VOIP SIP client around since 2001
- Is available on GNU/Linux, android, iOS, Windows, Mac
- Uses SIP standards for audio, video and instant messaging
- Secure group messaging using a Signal protocol derivative

Linphone's team also provides

- Flexisip, an open source SIP Proxy
- A free SIP service sip.linphone.org

I. Principles of Forward Error Correction

II. Why the Linphone team chose the Flexible FEC scheme

III. Flexible FEC scheme algorithm described in RFC 8627

IV. Implementation challenges

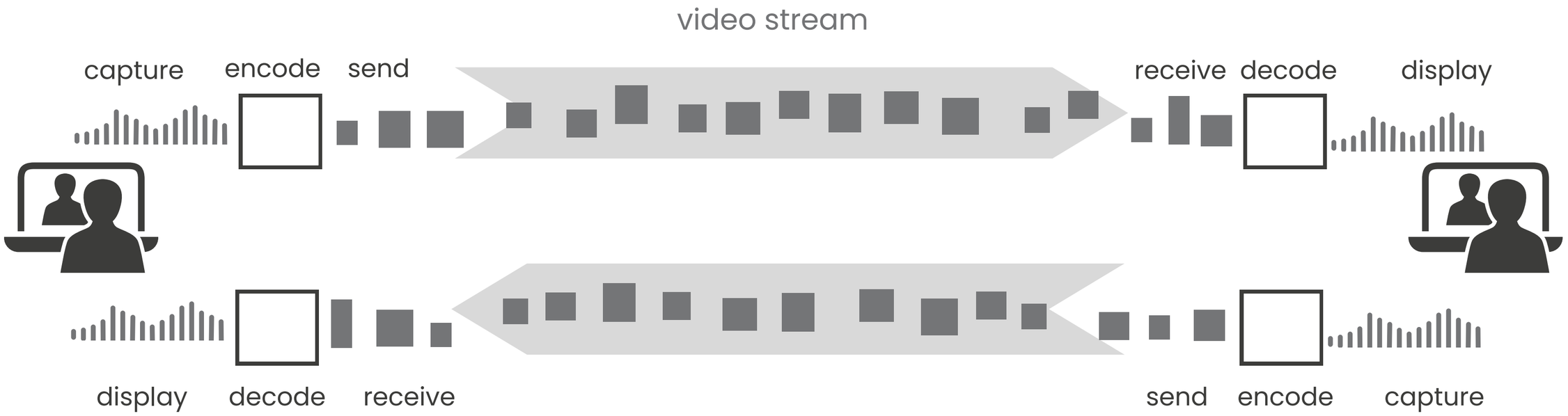
V. Results

I. PRINCIPLES OF FORWARD ERROR CORRECTION

PACKET LOSS IN VIDEO CALL

Real-time Transport Protocol (RTP)

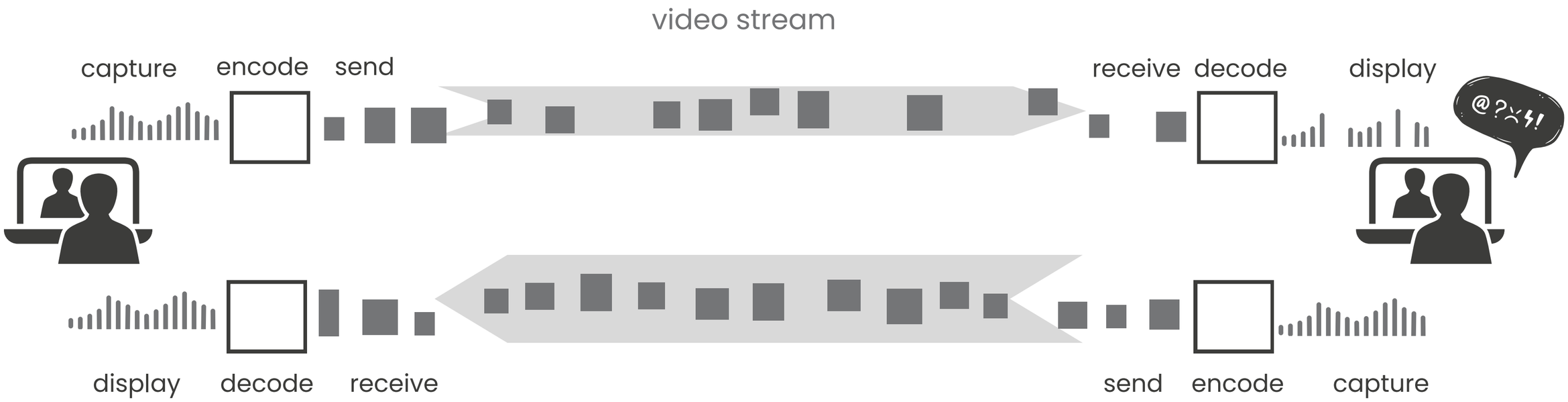
- internet protocol for video/audio transmission
- packets format
- more reliable than UDP



PACKET LOSS IN VIDEO CALL

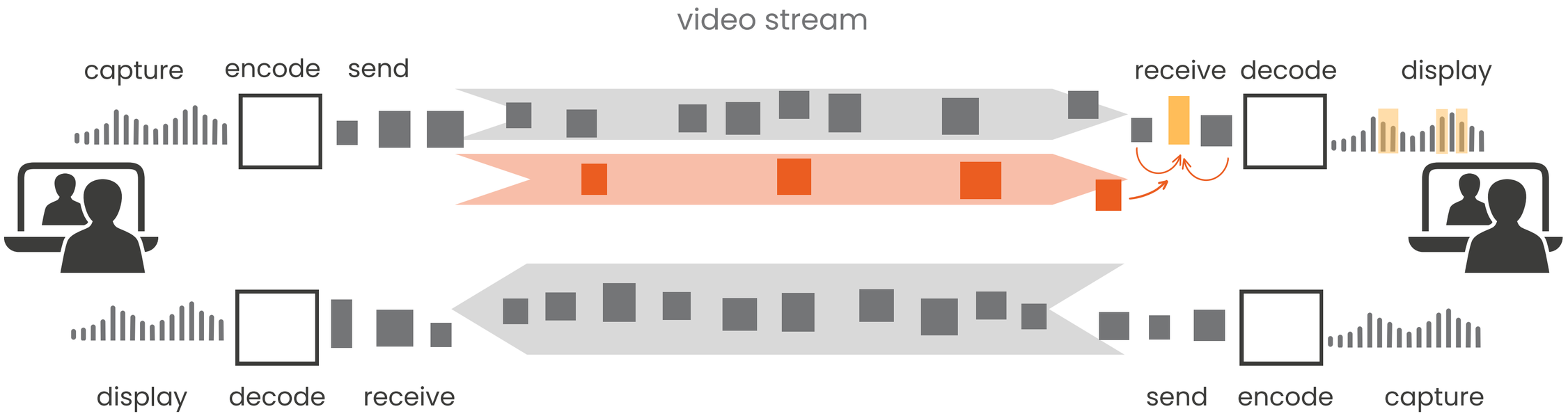
Real-time Transport Protocol (RTP)

- internet protocol for video/audio transmission
- packets format
- more reliable than UDP



PACKET LOSS IN VIDEO CALL

Real-time Transport Protocol (RTP)
- internet protocol for video/audio transmission
- packets format
- more reliable than UDP



II. WHY THE LINPHONE TEAM CHOSE THE FLEXIBLE FEC SCHEME

STRATEGIES TO RECOVER LOST PACKETS

- Loss detected
  - ask to send the packet again
  - send preventively the packet twice
  - recover the packet

- Forward Error Correction
  - Low Density Parity Check codes
  - Flexible FEC
    - simplicity: based on packet combinaison with XOR
    - free solution
    - recent standard
    - interoperability with webrtc

RFC 8627

Internet Engineering Task Force (IETF)
Request for Comments: 8627
Category: Standards Track
ISSN: 2070-1721

M. Zanaty
Cisco
V. Singh
callstats.io
A. Begen
Networked Media
G. Mandyam
Qualcomm Inc.
July 2019

RTP Payload Format for Flexible Forward Error Correction (FEC)

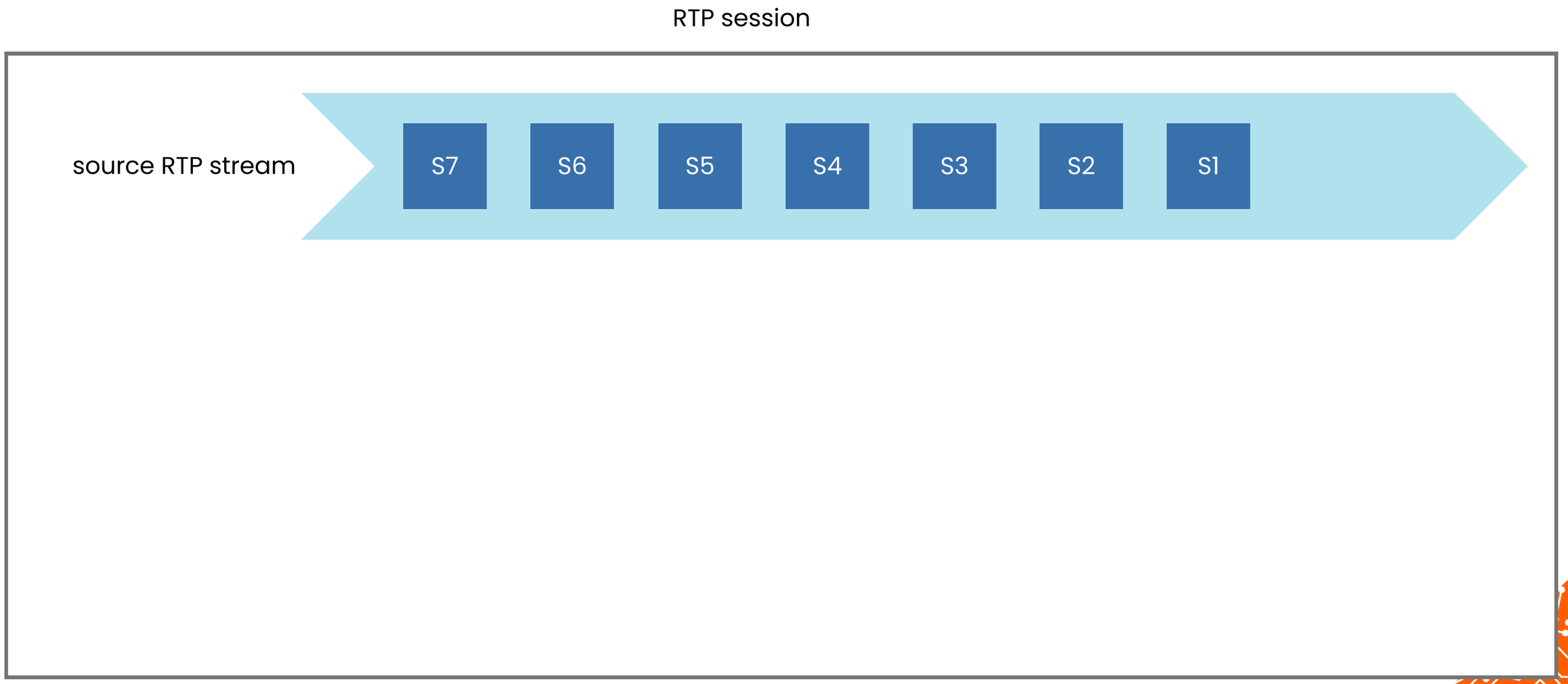
Abstract

This document defines new RTP payload formats for the Forward Error Correction (FEC) packets that are generated by the non-interleaved and interleaved parity codes from source media encapsulated in RTP. These parity codes are systematic codes (Flexible FEC, or "FLEX FEC"), where a number of FEC repair packets are generated from a set of source packets from one or more source RTP streams. These FEC repair packets are sent in a redundancy RTP stream separate from the source RTP stream(s) that carries the source packets. RTP source packets that were lost in transmission can be reconstructed using the source and repair packets that were received. The non-interleaved and interleaved parity codes that are defined in this specification offer a good protection against random and bursty packet losses, respectively, at a cost of complexity. The RTP payload formats that are defined in this document address scalability issues experienced with the earlier specifications and offer several improvements. Due to these changes, the new payload formats are not backward compatible with earlier specifications; however, endpoints that do not implement this specification can still work by simply ignoring the FEC repair packets.

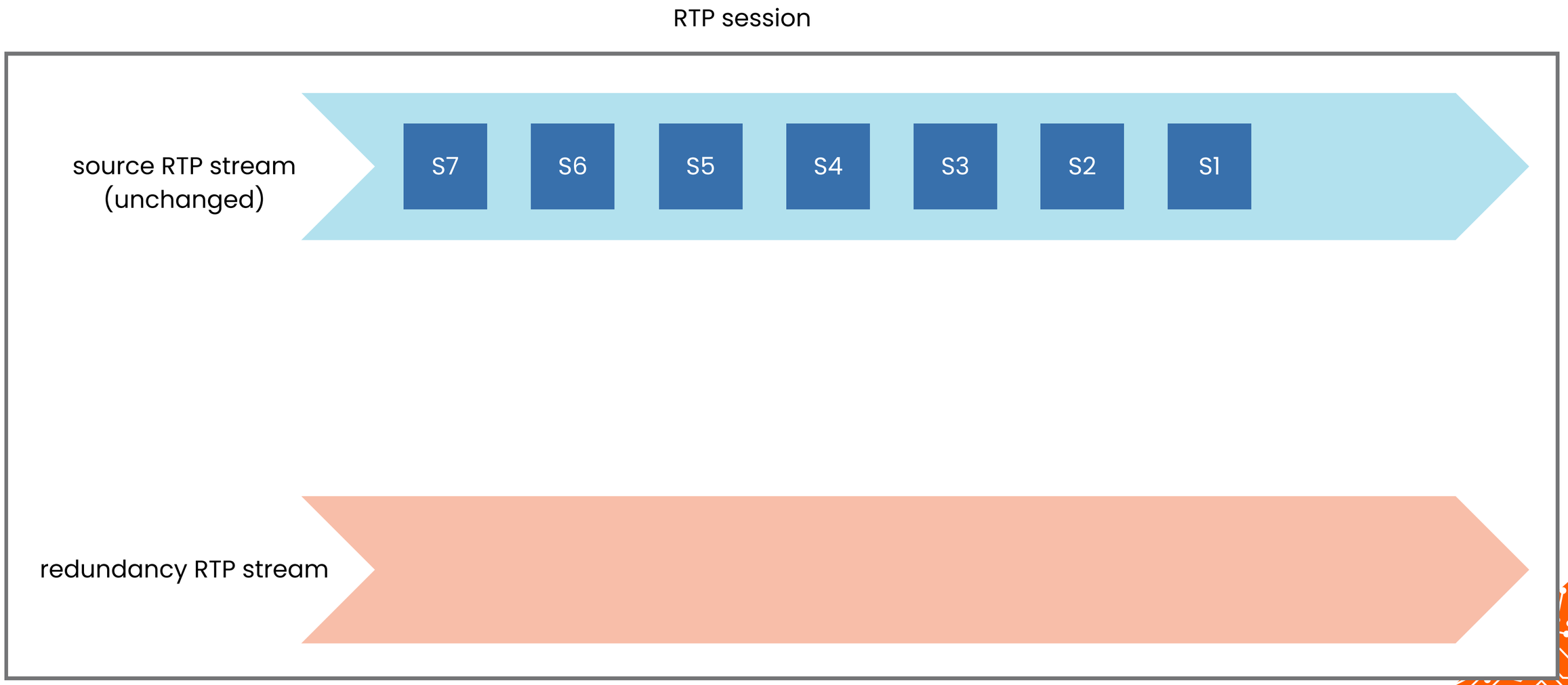
- Full description of how a RTP stream is protected
- RTP payload and header format for FEC packets
- Parity codes for reconstruction
- All media: video, audio, text, application

III. FLEXIBLE FEC SCHEME ALGORITHM DESCRIBED IN RFC 8627

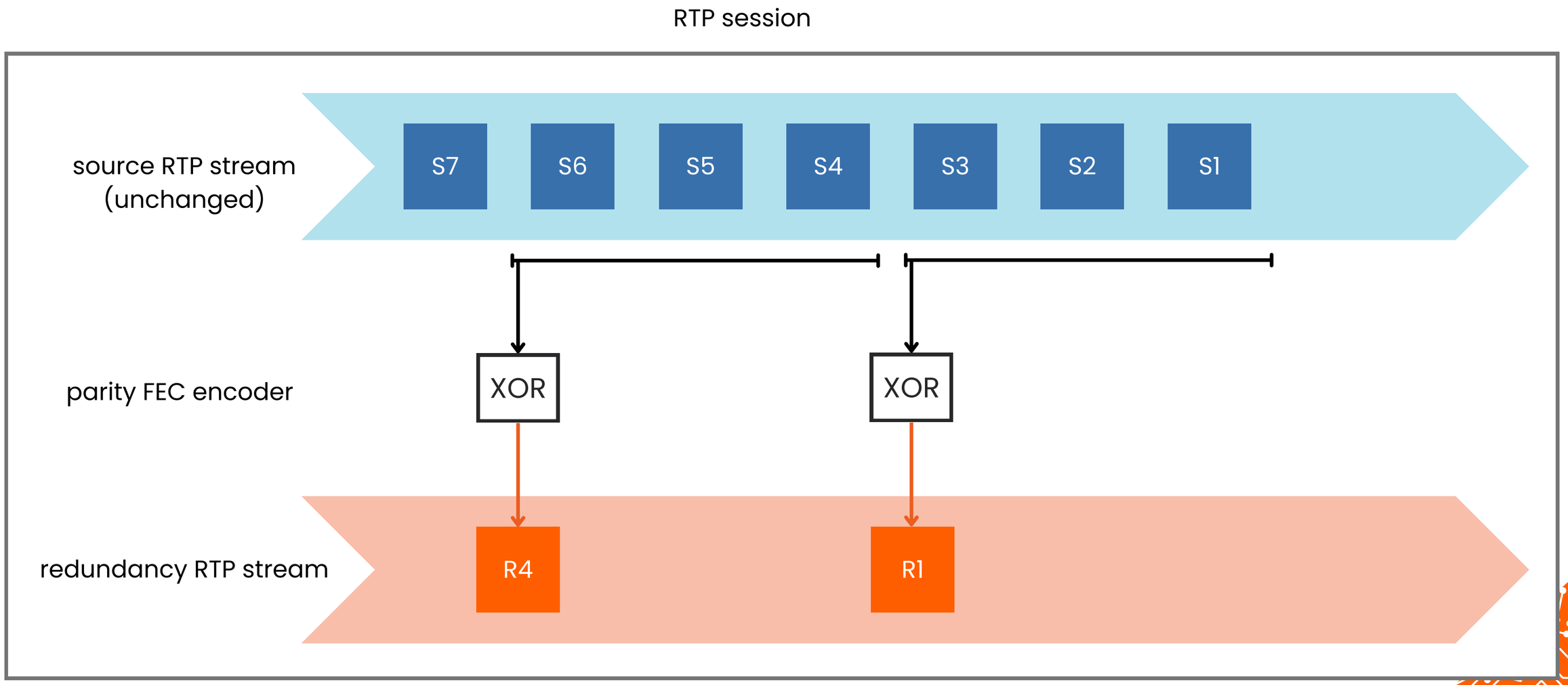
FEC PROTECTION



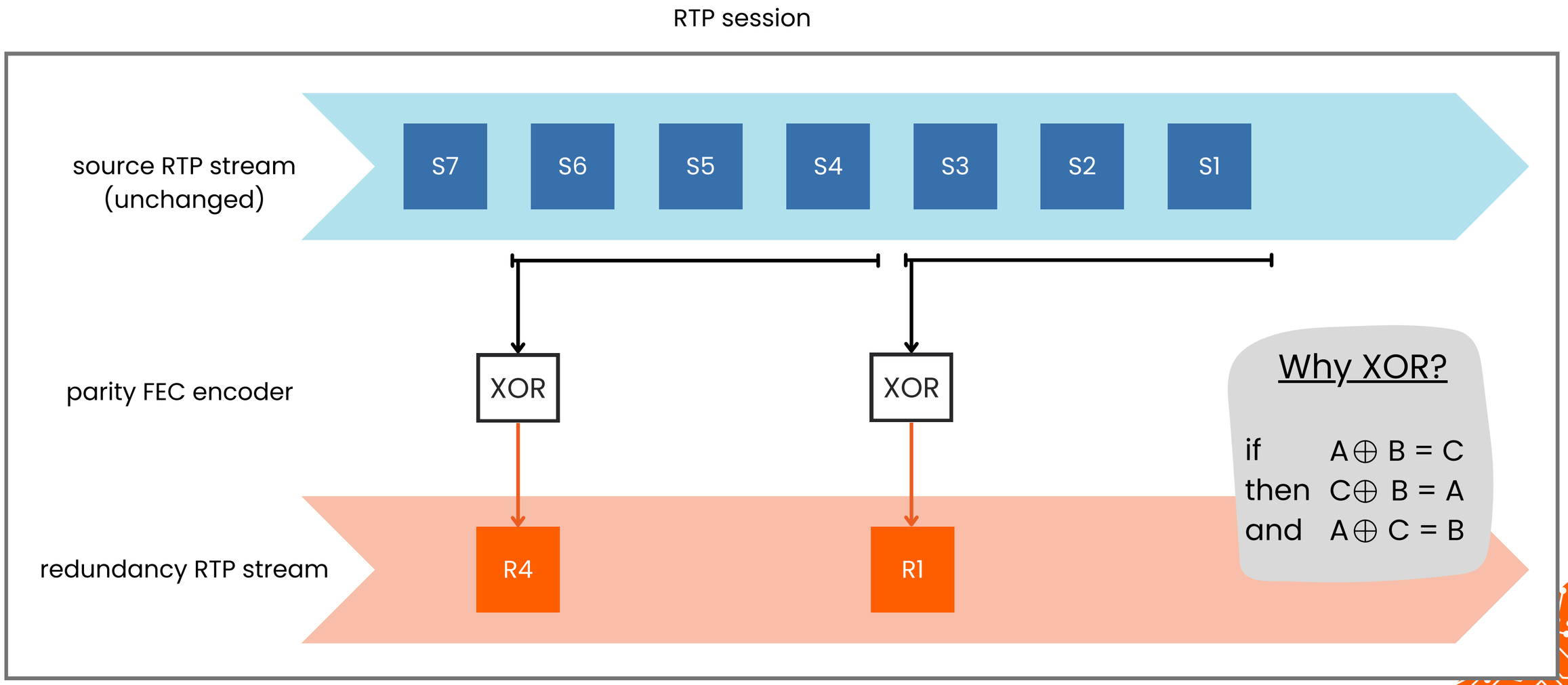
FEC PROTECTION



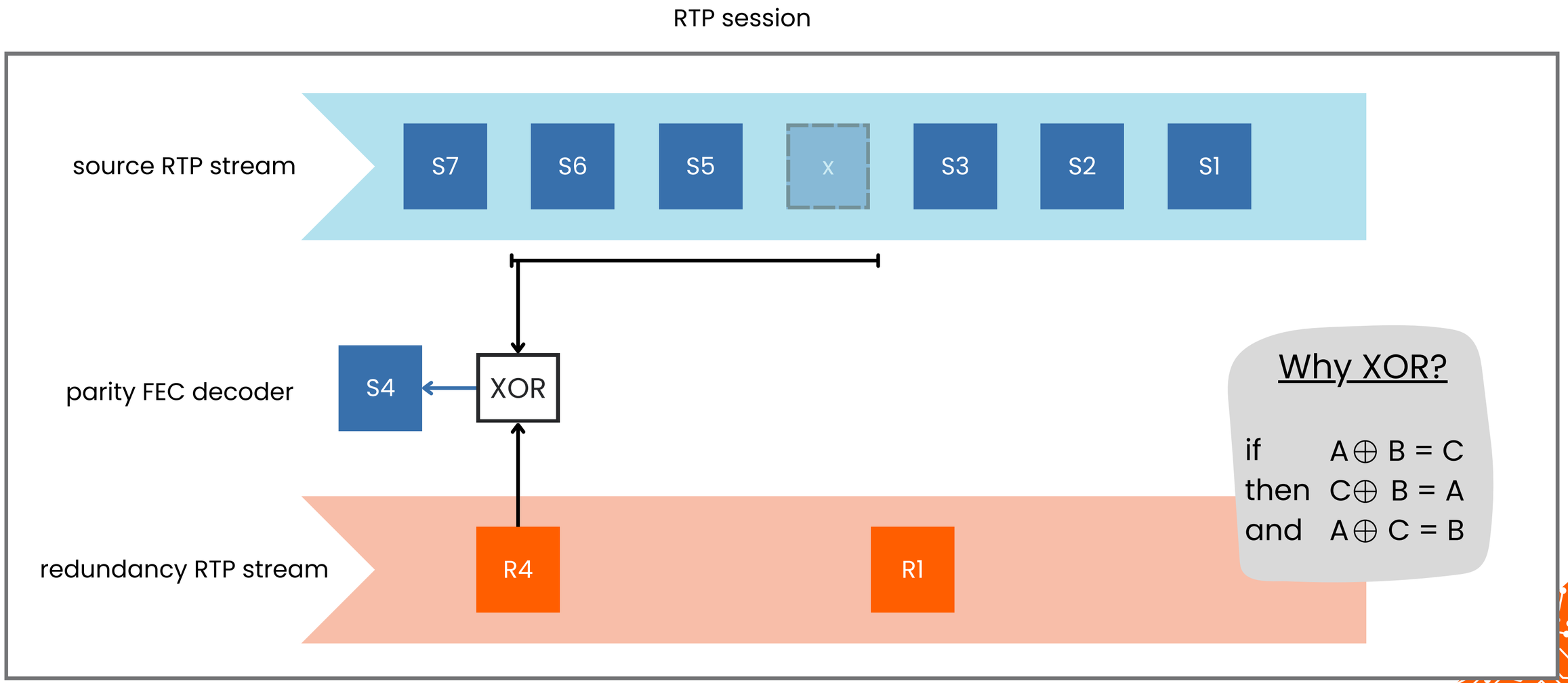
FEC PROTECTION: SENDER



FEC PROTECTION: SENDER

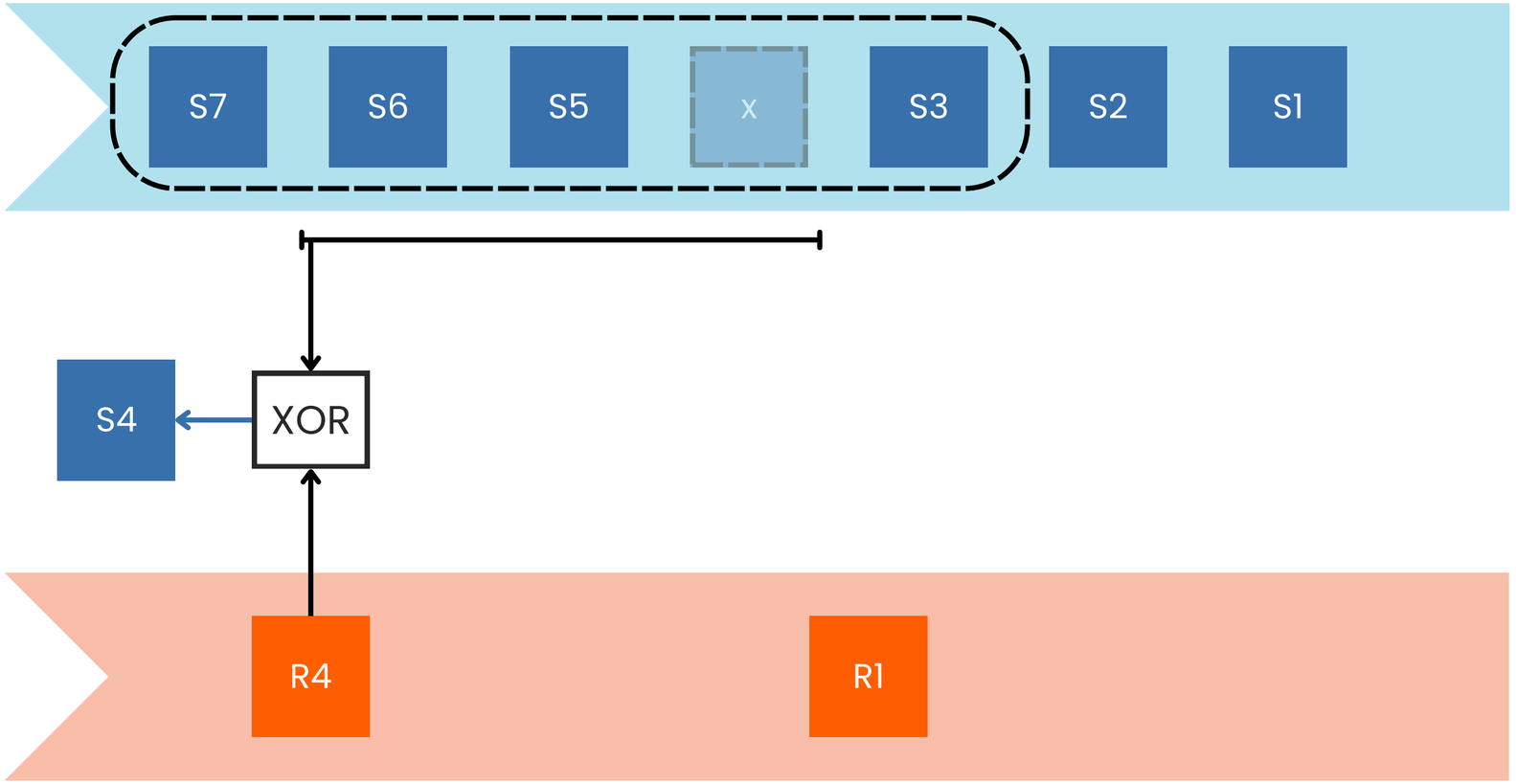


FEC PROTECTION: RECEIVER

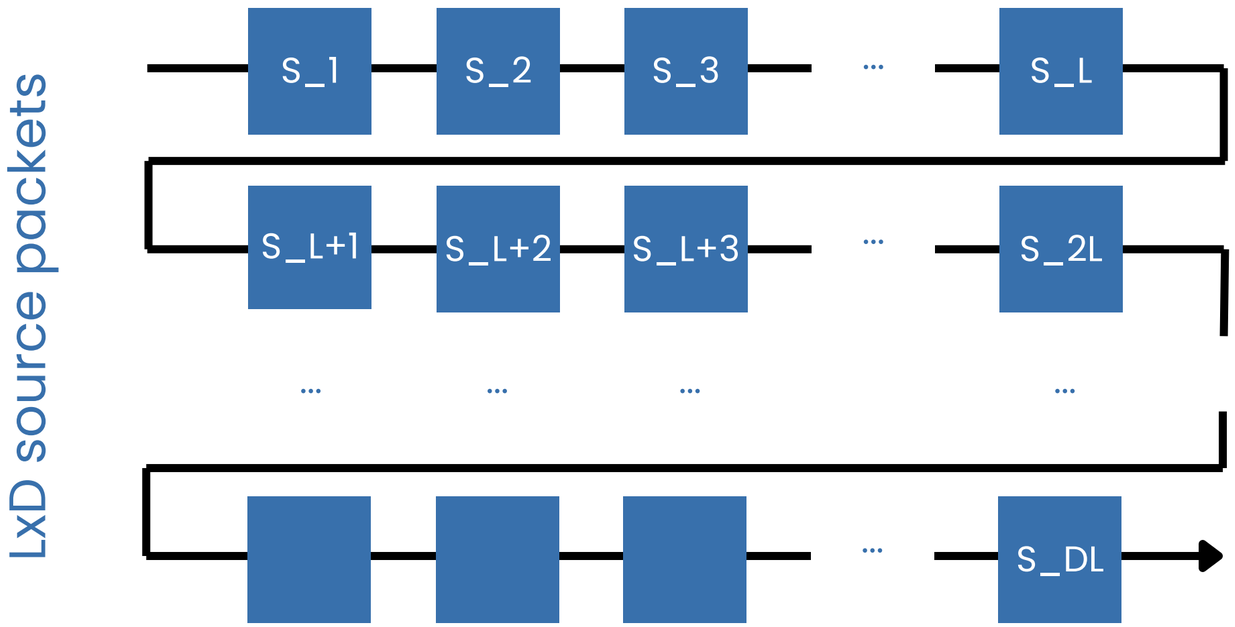


FEC PROTECTION: PARAMETERS

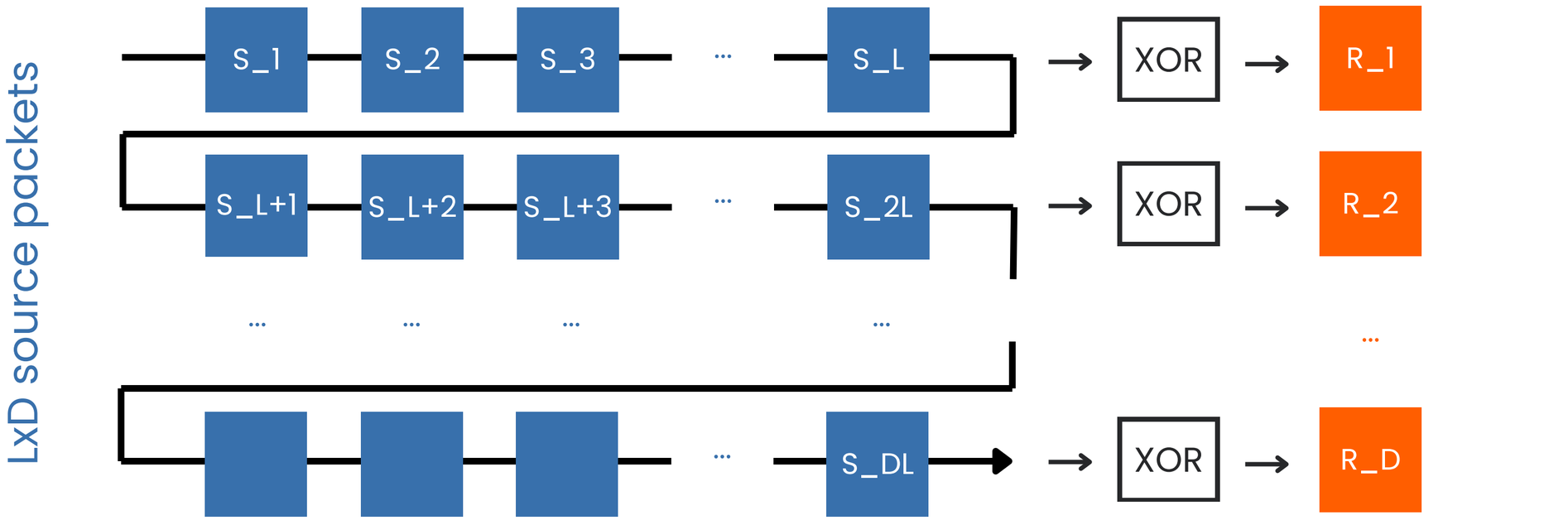
- Repair window
- Protection pattern



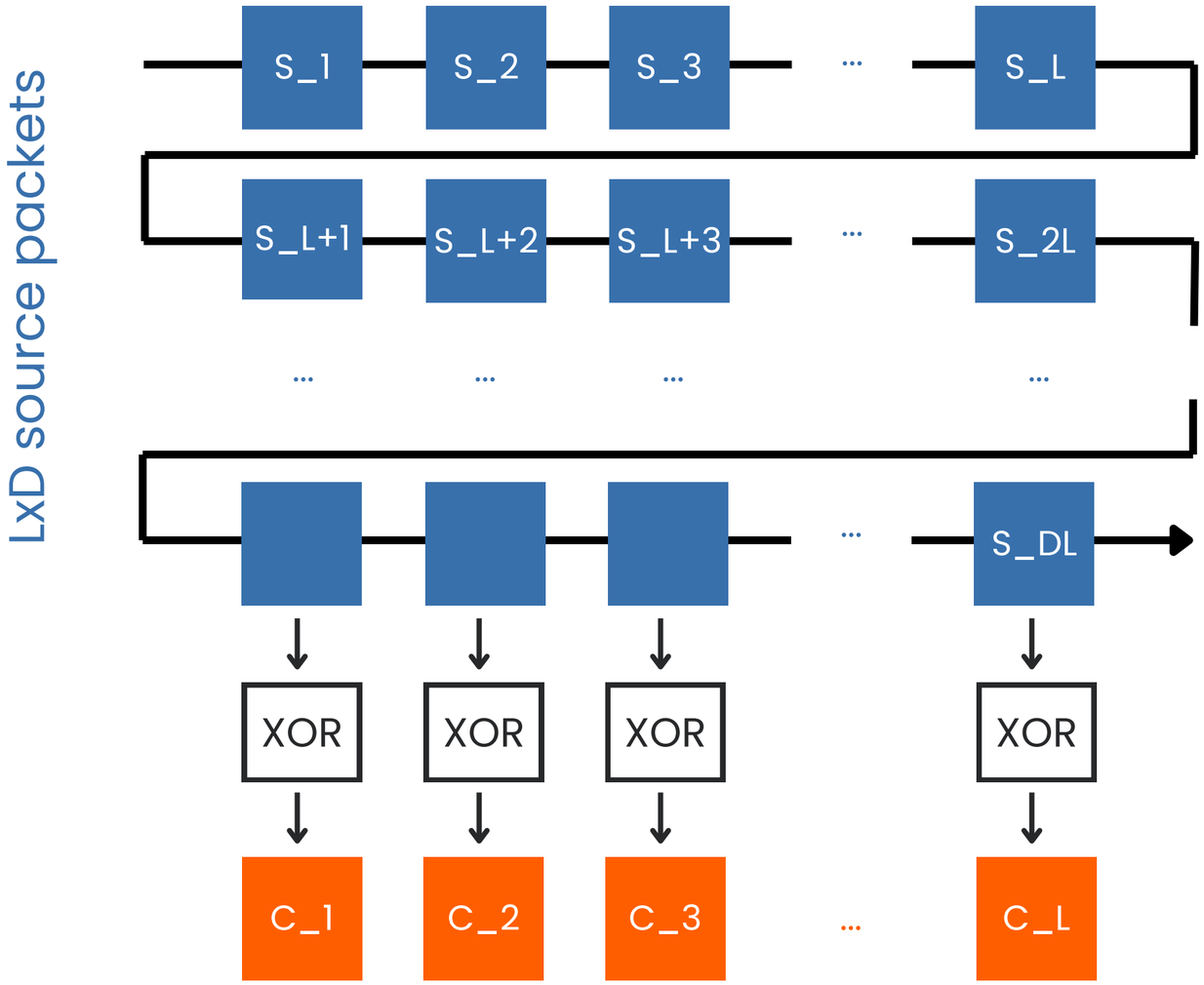
1D NON INTERLEAVED FEC PROTECTION



1D NON INTERLEAVED FEC PROTECTION



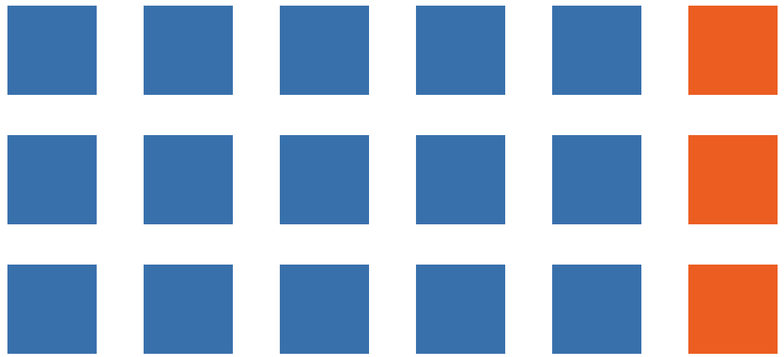
1D INTERLEAVED FEC PROTECTION



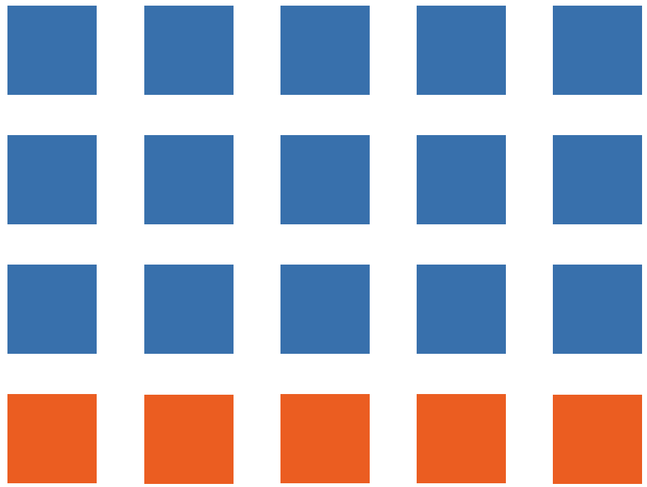
depth D

L repair packets

DECODING ALGORITHM: 1D



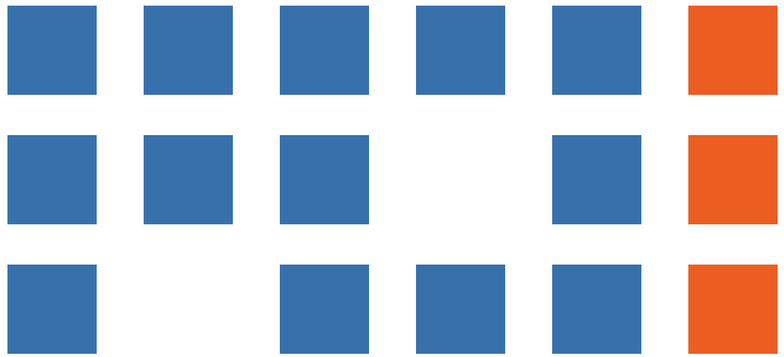
1D non interleaved
L = 5



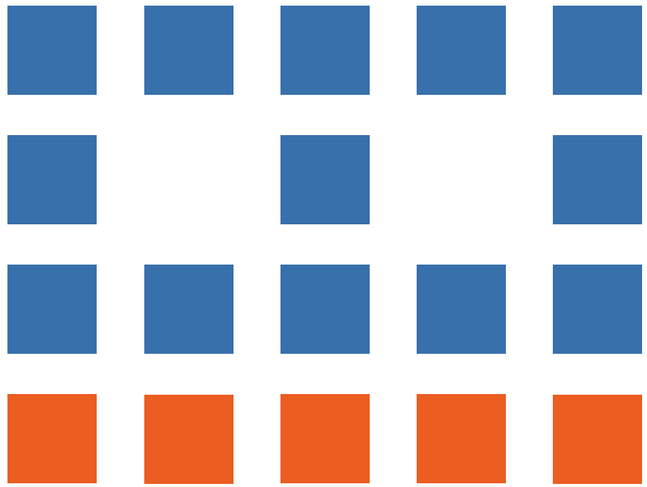
1D interleaved
D = 3

DECODING ALGORITHM: 1D

random packet loss



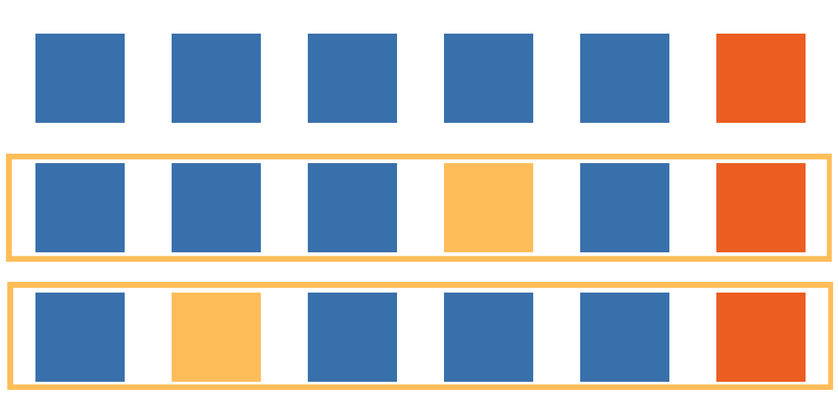
1D non interleaved
L = 5

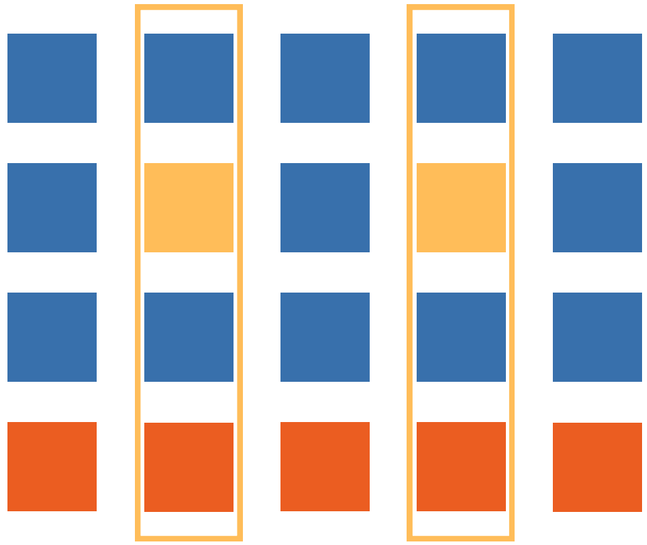


1D interleaved
D = 3

DECODING ALGORITHM: 1D

random packet loss -> full error recovery



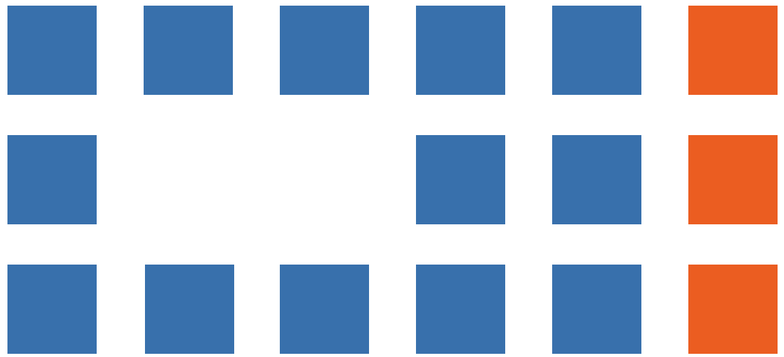


1D non interleaved
L = 5

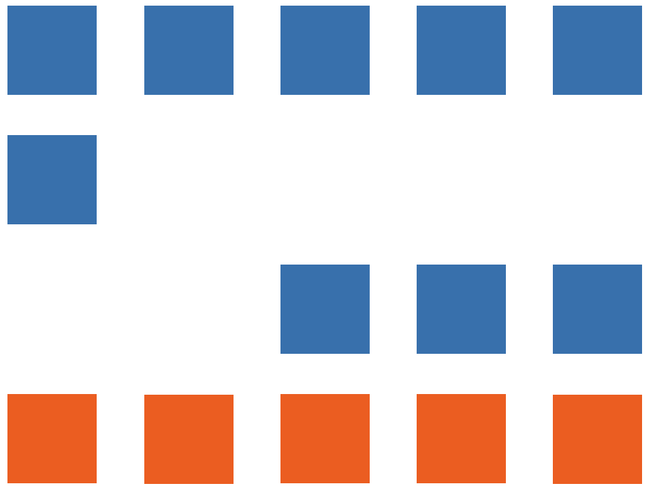
1D interleaved
D = 3

DECODING ALGORITHM: 1D

burst: consecutive packet loss



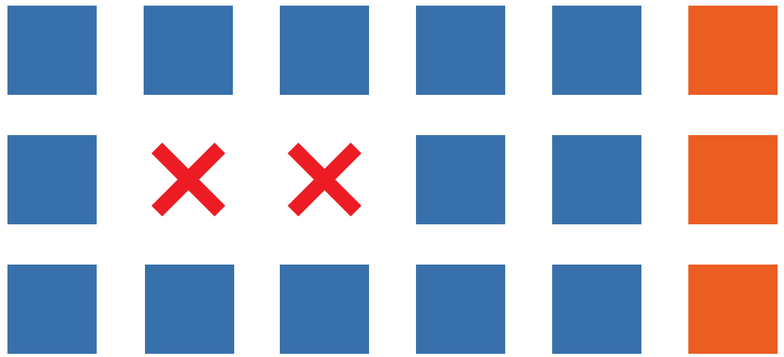
1D non interleaved
L = 5

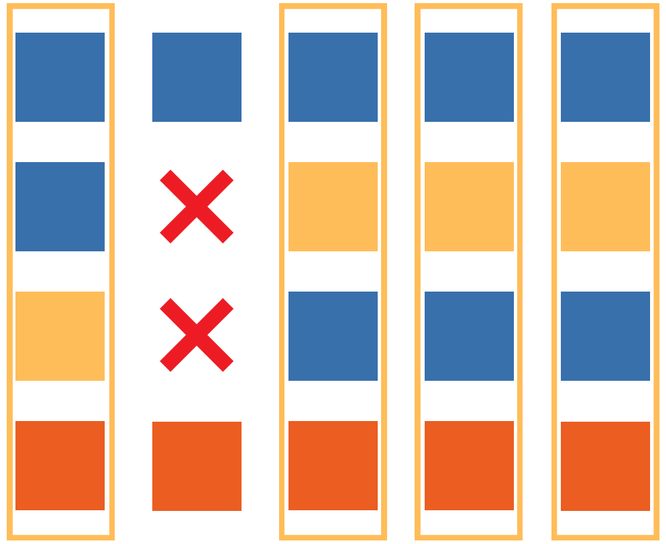


1D interleaved
D = 3

DECODING ALGORITHM: 1D

burst: consecutive packet loss -> recovery fail



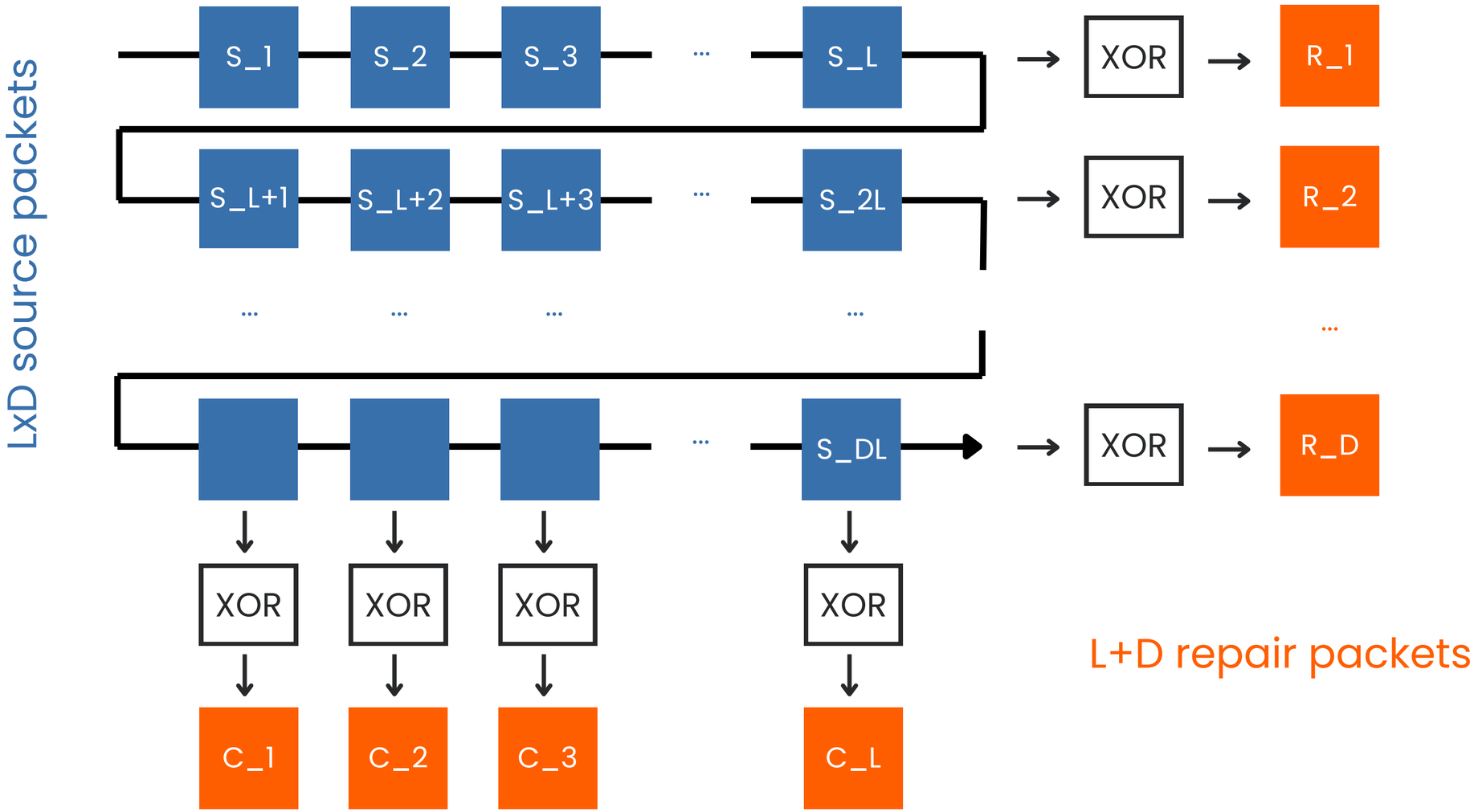


1D non interleaved
L = 5

1D interleaved
D = 3

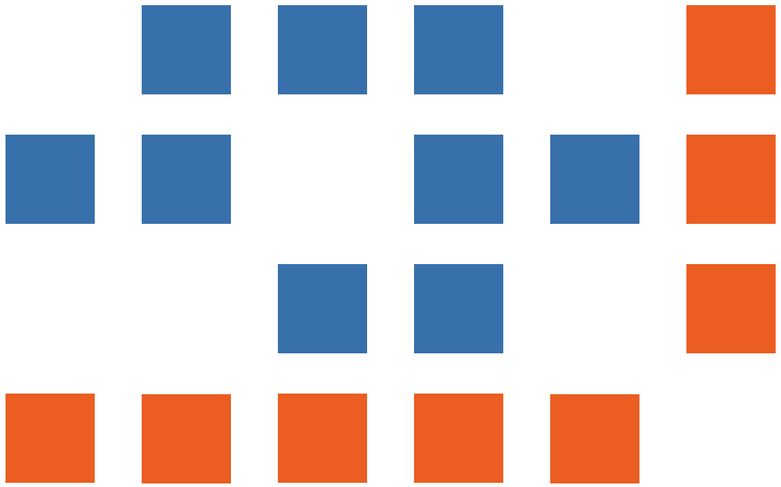
2D PARITY FEC PROTECTION

2D interleaved



L+D repair packets

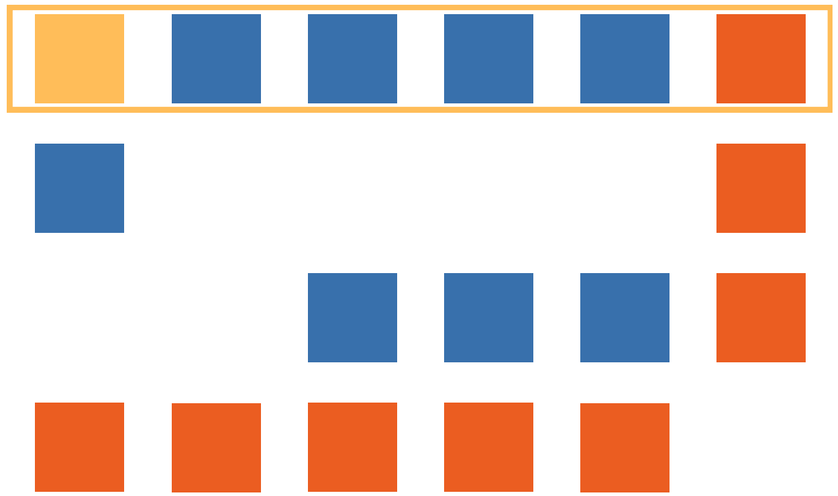
DECODING ALGORITHM: 2D, ITERATIVE APPROACH

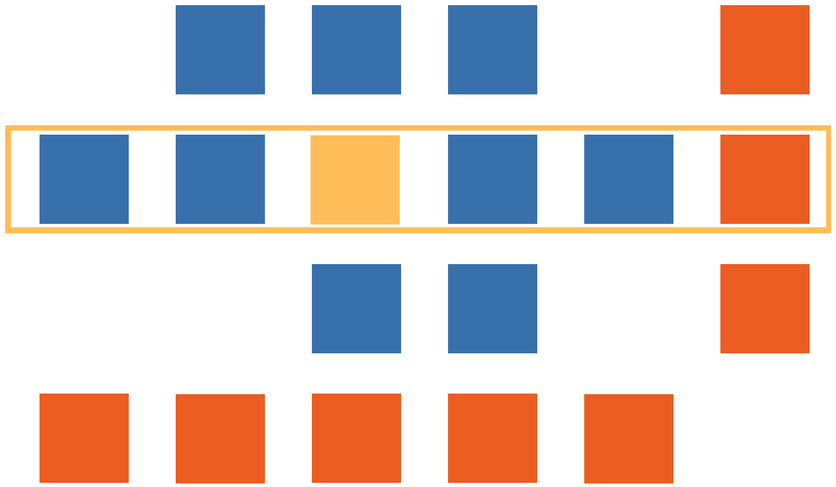


L = 5

D = 3

DECODING ALGORITHM: 2D, ITERATIVE APPROACH



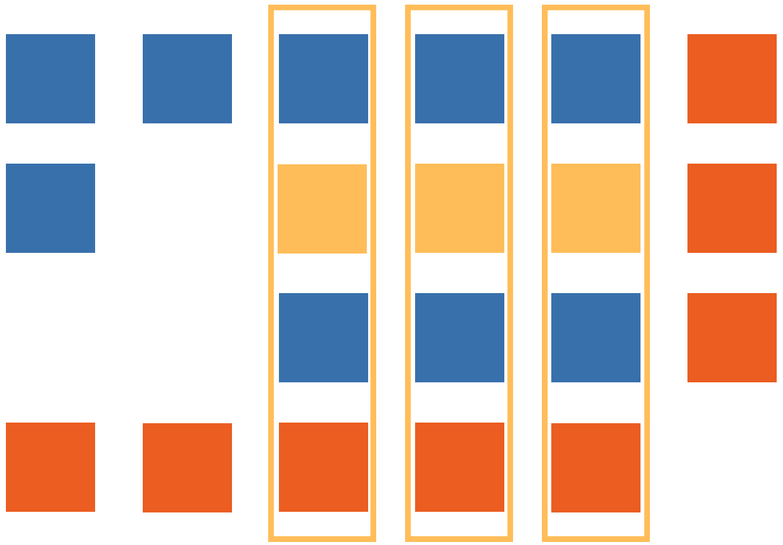


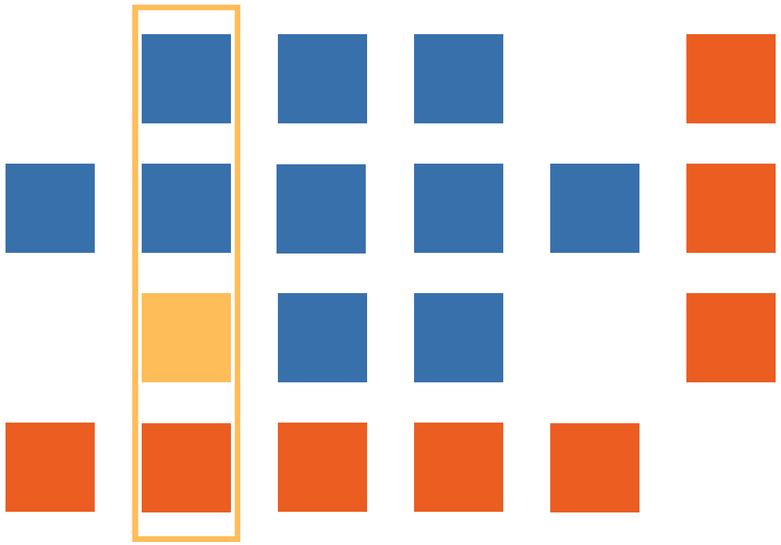
repair rows

L = 5

D = 3

DECODING ALGORITHM: 2D, ITERATIVE APPROACH

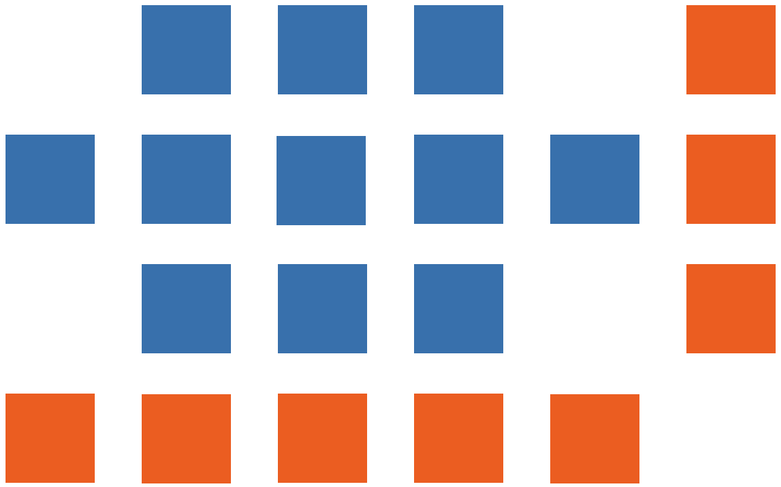




repair rows
repair columns

L = 5
D = 3

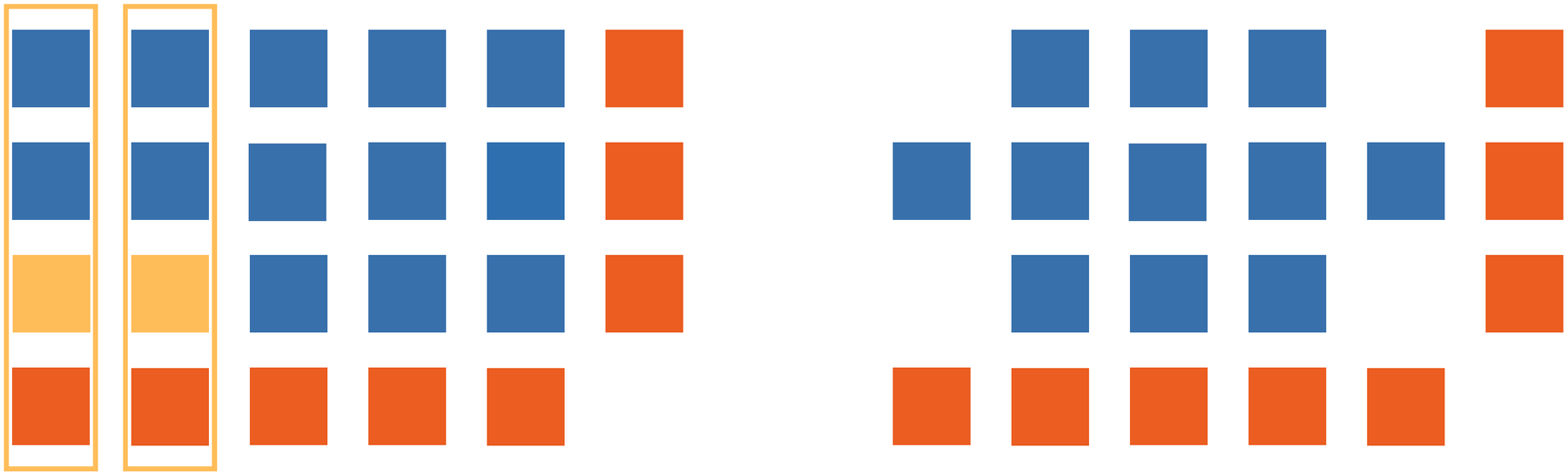
DECODING ALGORITHM: 2D, ITERATIVE APPROACH



repair rows
repair columns
repeat

L = 5
D = 3

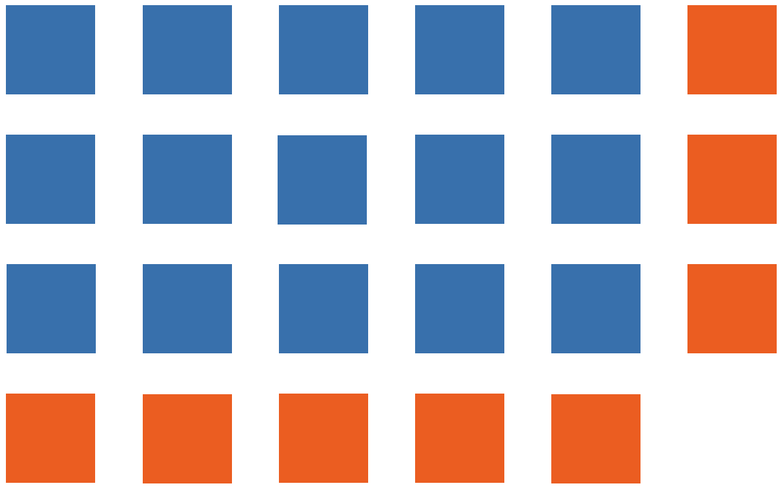
DECODING ALGORITHM: 2D, ITERATIVE APPROACH

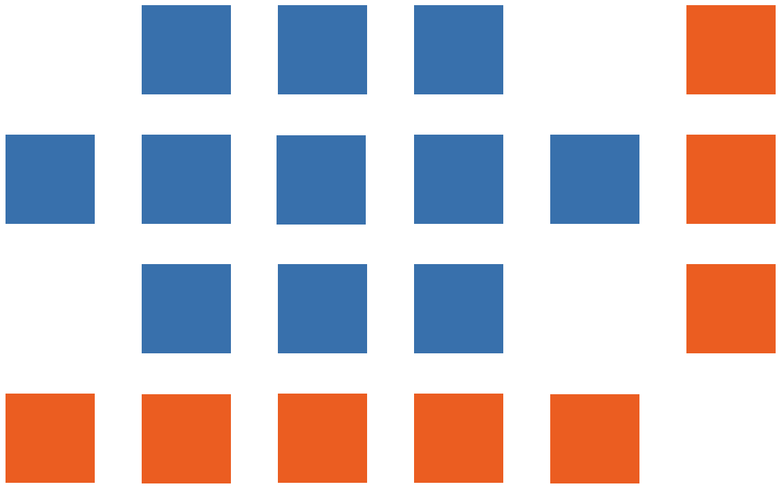


repair rows
→ repair columns
repeat

L = 5
D = 3

DECODING ALGORITHM: 2D, ITERATIVE APPROACH



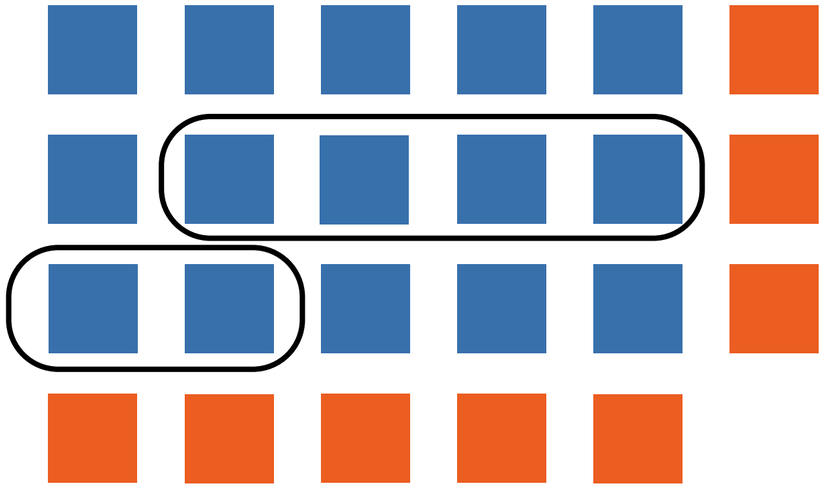


repair rows
repair columns
repeat
stop

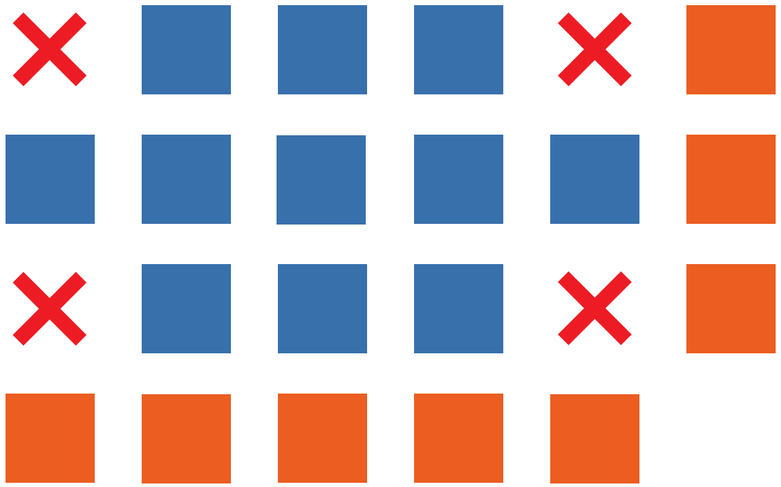
L = 5
D = 3

DECODING ALGORITHM: 2D, ITERATIVE APPROACH

burst -> full error recovery



cyclic loss pattern -> recovery fail



repair rows
repair columns
repeat
stop

- some loss pattern fail
- more robust to bursts

L = 5
D = 3

OVERHEAD

$$\text{overhead} = \frac{\text{number of byte of the repair packets}}{\text{number of byte of the protected source packets}}$$

OVERHEAD

$$\text{overhead} = \frac{\text{number of byte of the repair packets}}{\text{number of byte of the protected source packets}}$$

- size(repair packet) > size (source packet)

OVERHEAD

$$\text{overhead} = \frac{\text{number of byte of the repair packets}}{\text{number of byte of the protected source packets}}$$

- size(repair packet) > size (source packet)
- if all source packets have the same size :
  - 1D non interleaved $\text{overhead} = \frac{1}{L}$
  - 1D interleaved $\text{overhead} = \frac{1}{D}$
  - 2D $\text{overhead} = \frac{1}{L} + \frac{1}{D}$

OVERHEAD

$$\text{overhead} = \frac{\text{number of byte of the repair packets}}{\text{number of byte of the protected source packets}}$$

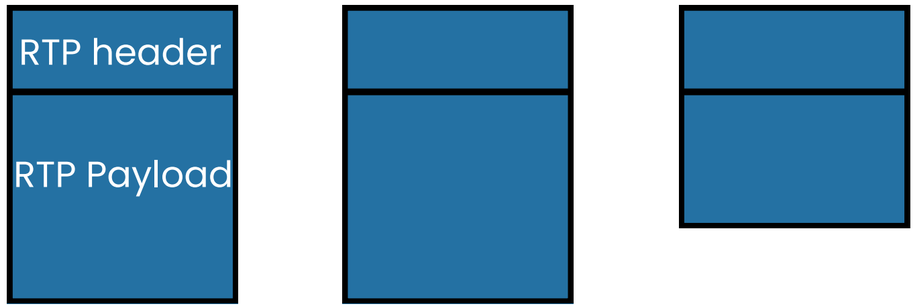
- size(repair packet) > size (source packet)
- if all source packets have the same size :
  - 1D non interleaved $\text{overhead} = \frac{1}{L}$
  - 1D interleaved $\text{overhead} = \frac{1}{D}$
  - 2D $\text{overhead} = \frac{1}{L} + \frac{1}{D}$

L	D	overhead
10	0	0.10
5	0	0.20
5	5	0.40
3	3	0.67

PACKETS FORMAT

Source packets

- unchanged
  - RTP header
  - RTP payload



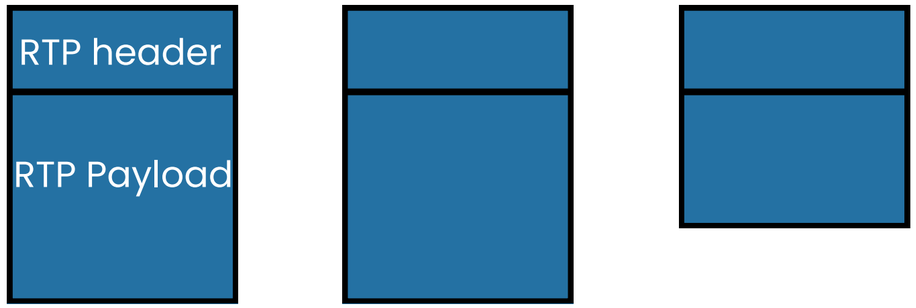
PACKETS FORMAT

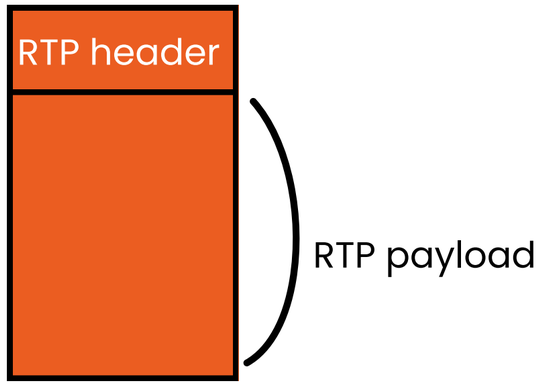
Source packets

- unchanged
  - RTP header
  - RTP payload

Repair packet

- RTP header
- RTP payload





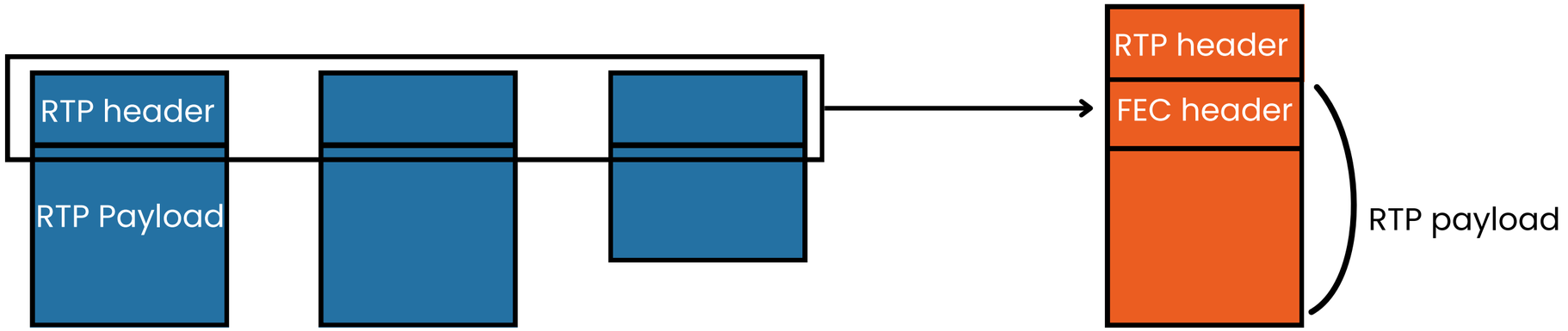
PACKETS FORMAT

Source packets

- unchanged
  - RTP header
  - RTP payload
- unique sequence number in RTP header

Repair packet

- RTP header
- payload
  - FEC header: identification of the source packets protected



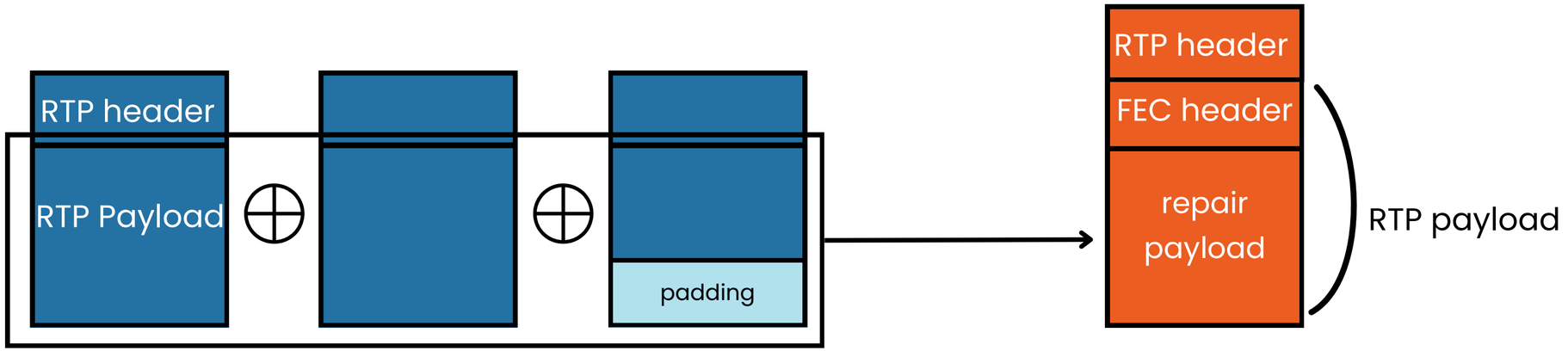
PACKETS FORMAT

Source packets

- unchanged
  - RTP header
  - RTP payload
- unique sequence number in RTP header

Repair packet

- RTP header
- payload
  - FEC header: identification of the source packets protected
  - repair payload: XOR result

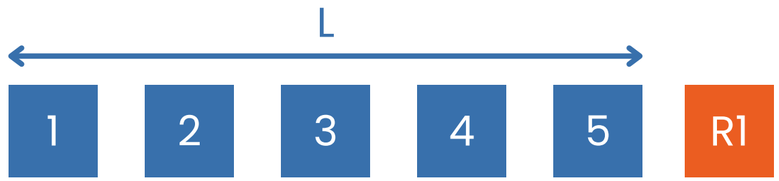


REPAIR PACKET

- A **single repair packet** carries information about
  - the set of source packets protected
  - the protection configuration with L and D

R1

REPAIR PACKET



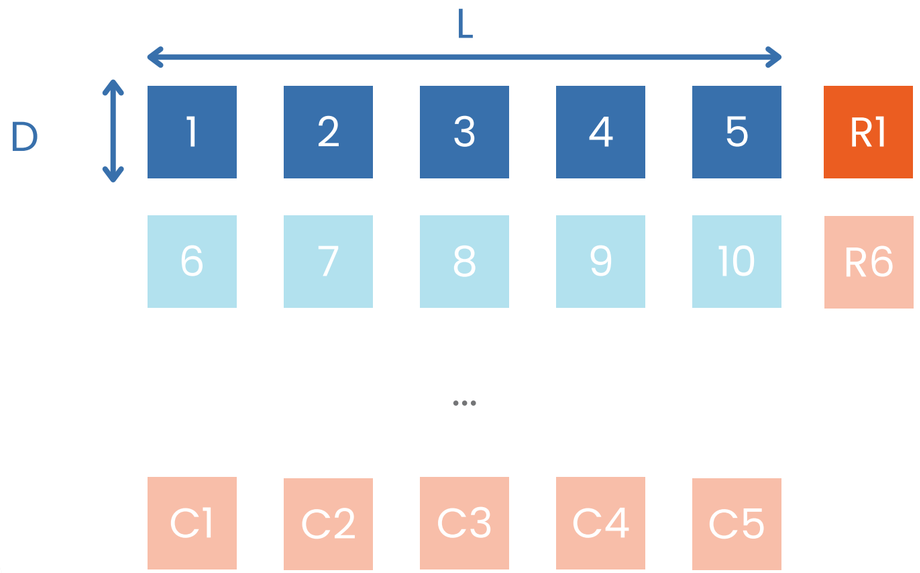
- A **single repair packet** carries information about
  - the set of source packets protected
  - the protection configuration with L and D

- For fixed L columns and D rows

  - **L > 0, D = 0: row FEC**, source packets: SN, SN+1, SN+(L-1)

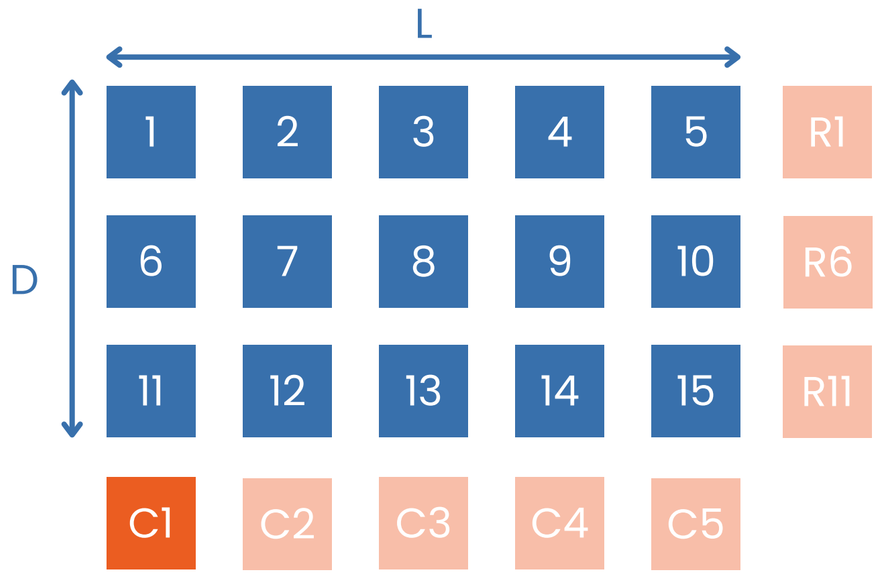
REPAIR PACKET

- A **single repair packet** carries information about
  - the set of source packets protected
  - the protection configuration with L and D



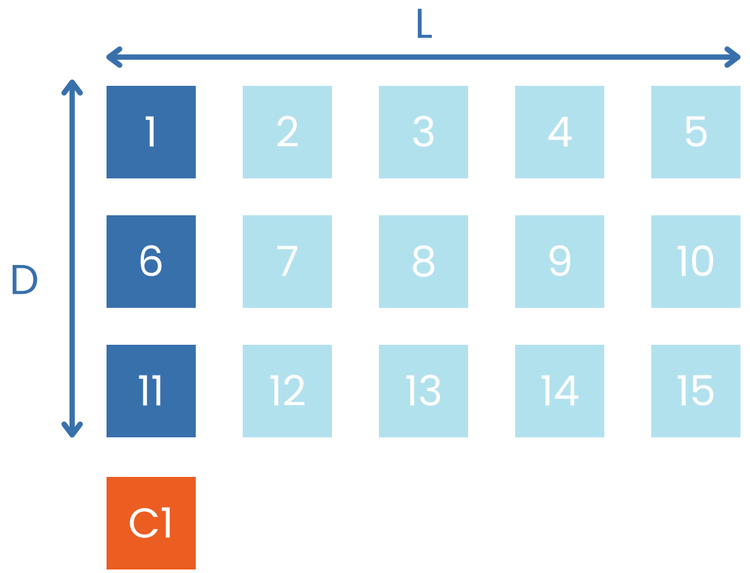
- For fixed L columns and D rows
  - L > 0, D = 0: row FEC, source packets: SN, SN+1, SN+(L-1)
  - **L > 0, D = 1: row FEC followed by column**, all columns following all rows repair packets

REPAIR PACKET



- A **single repair packet** carries information about
  - the set of source packets protected
  - the protection configuration with L and D
- For fixed L columns and D rows
  - L > 0, D = 0: row FEC, source packets: SN, SN+1, SN+(L-1)
  - L > 0, D = 1: row FEC followed by column, all columns following all rows repair packets
  - **L > 0, D > 1: column FEC**, source packets: SN, SN+L, ..., SN+(D-1)*L

REPAIR PACKET



- A **single repair packet** carries information about
  - the set of source packets protected
  - the protection configuration with L and D

- For fixed L columns and D rows
  - L > 0, D = 0: row FEC, source packets: SN, SN+1, SN+(L-1)
  - L > 0, D = 1: row FEC followed by column, all columns following all rows repair packets
  - **L > 0, D > 1: column FEC**, source packets: SN, SN+L, ..., SN+(D-1)*L

IV. IMPLEMENTATION CHALLENGES

IN LINPHONE

- Parameters
  - (L, D): (10, 0), (5, 0), (5, 5), (3, 3)
    - ideally 2D parity protection
    - adaptation to the loss rate and the network capabilities
  - repair window: 200 ms
    - contain full source block, given any value of L, D
    - do not cause any delay in the video
- Implementation in Linphone SDK in C and C++
  - oRTP: library implementing the RTP protocol
  - Mediastreamer2: streaming engine for voice/video

OUR STRATEGY FOR VIDEO QUALITY

Best possible use of the bandwidth

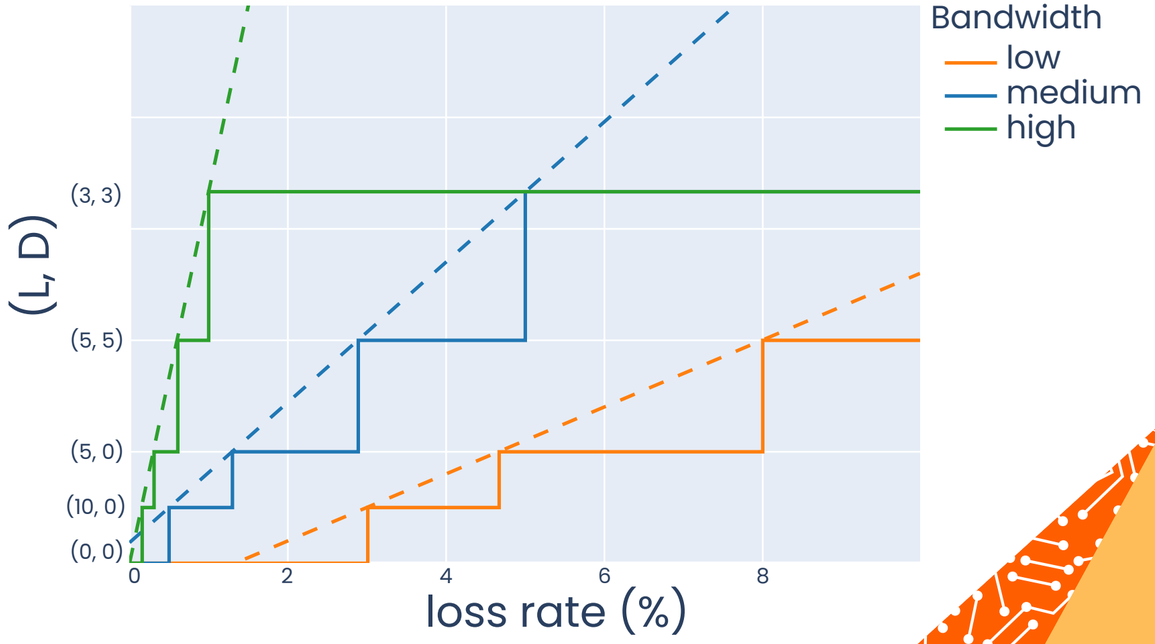
- Optimal video setting with flexFEC
  - definition, bitrate, frame rate
  - no freeze

  packet protection > high encoding setting

- Adaptatiblity: periodic control
  - available bandwidth
  - loss rate
  - bandwidth dedicated to FEC

- Congestions: disabling FEC without delay

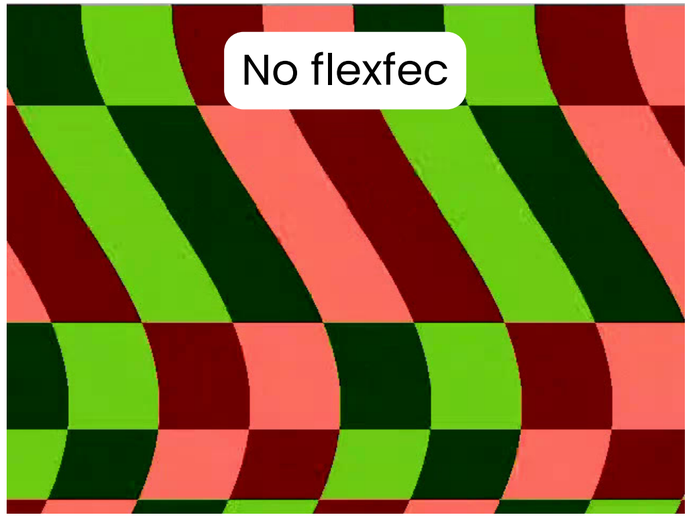
FEC protection adaptation to available bandwidth

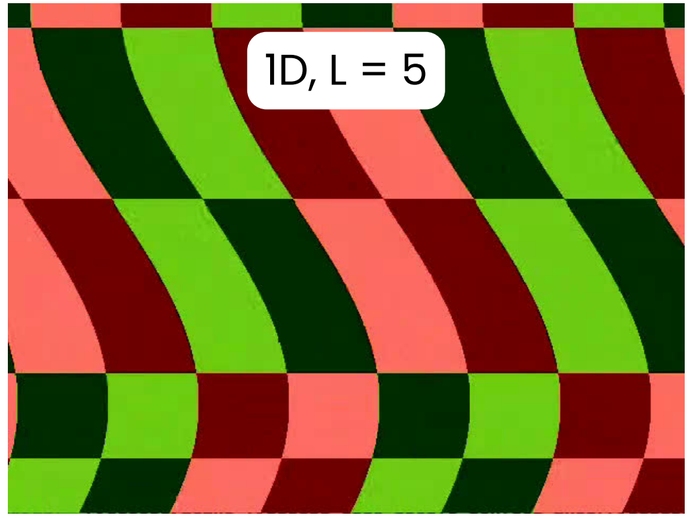


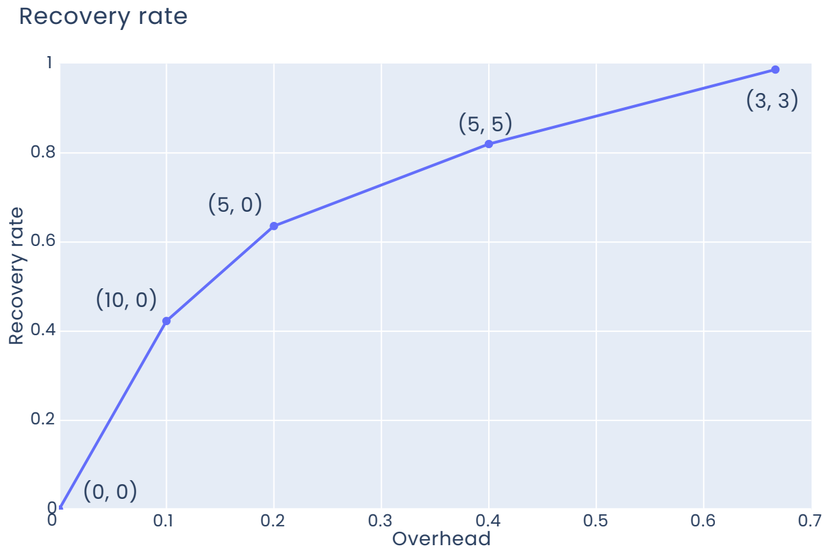
V. RESULTS

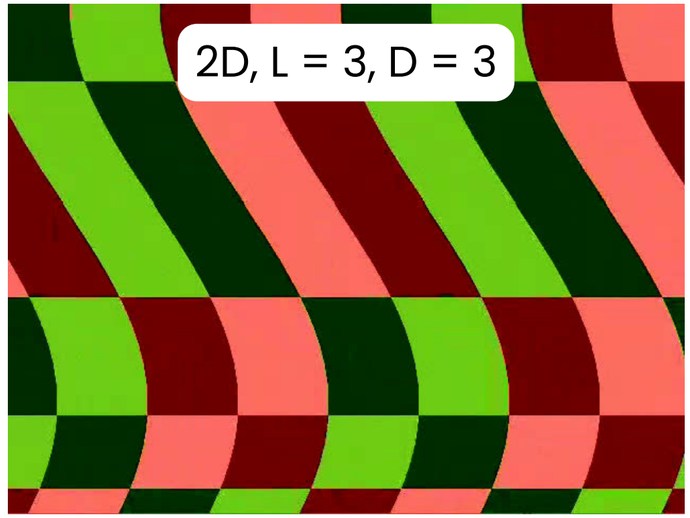
6% LOSS

AV1,
640x480
400kbits/s
30fps



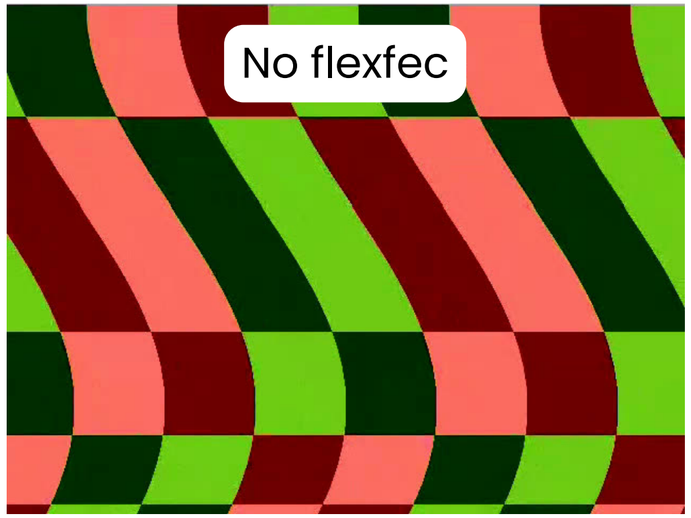


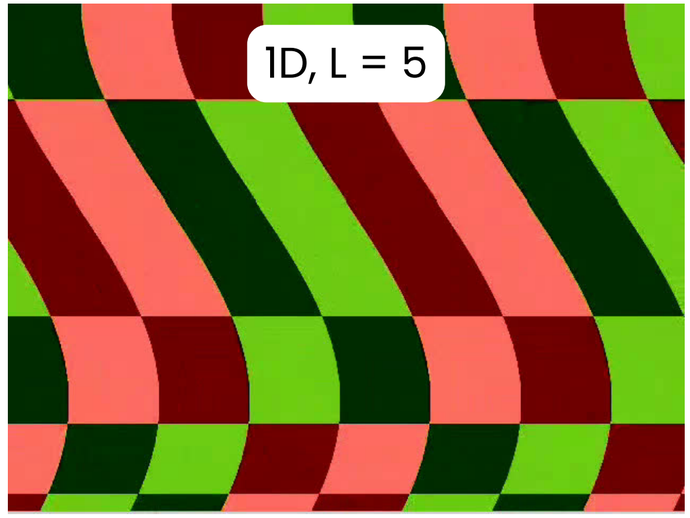


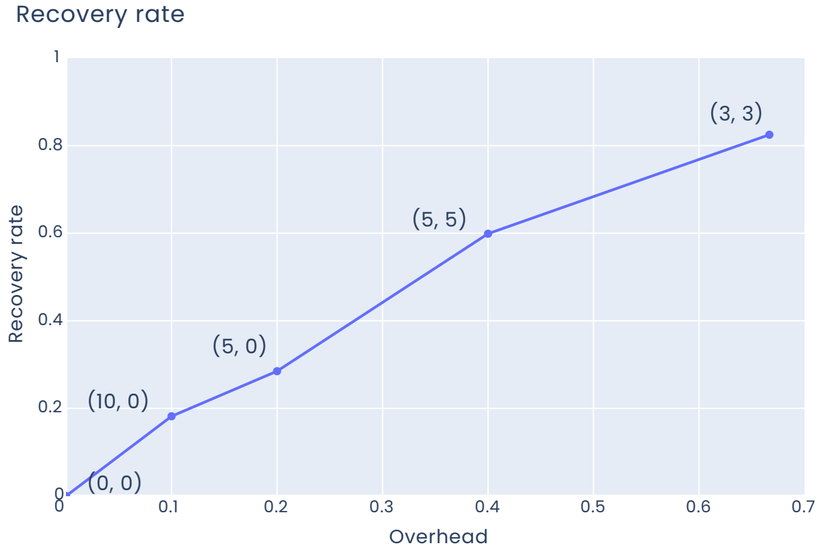


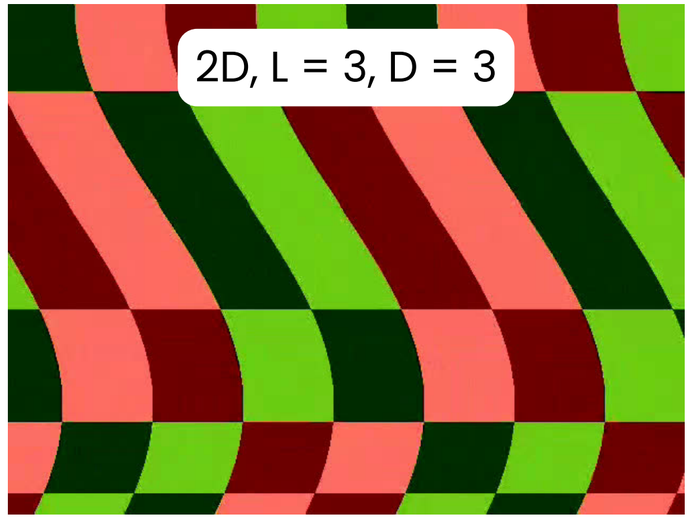
6% LOSS, BURST

AV1,
640x480
400kbits/s
30fps









CONCLUSIONS

- Flexible FEC
  - Simple and efficient way to improve resiliency against packet loss
    - send redondant information
    - adaptable
    - short delay
  - Significant bandwidth needed ❗
  - Complete description in RFC 8627, backward compatible
  - Real improvement in video quality

- FlexFEC coming soon in Linphone!
  - first for simple video call in 2024
  - then for video conference, audio...

THANK YOU FOR YOUR ATTENTION!

ANY QUESTIONS?

linphone

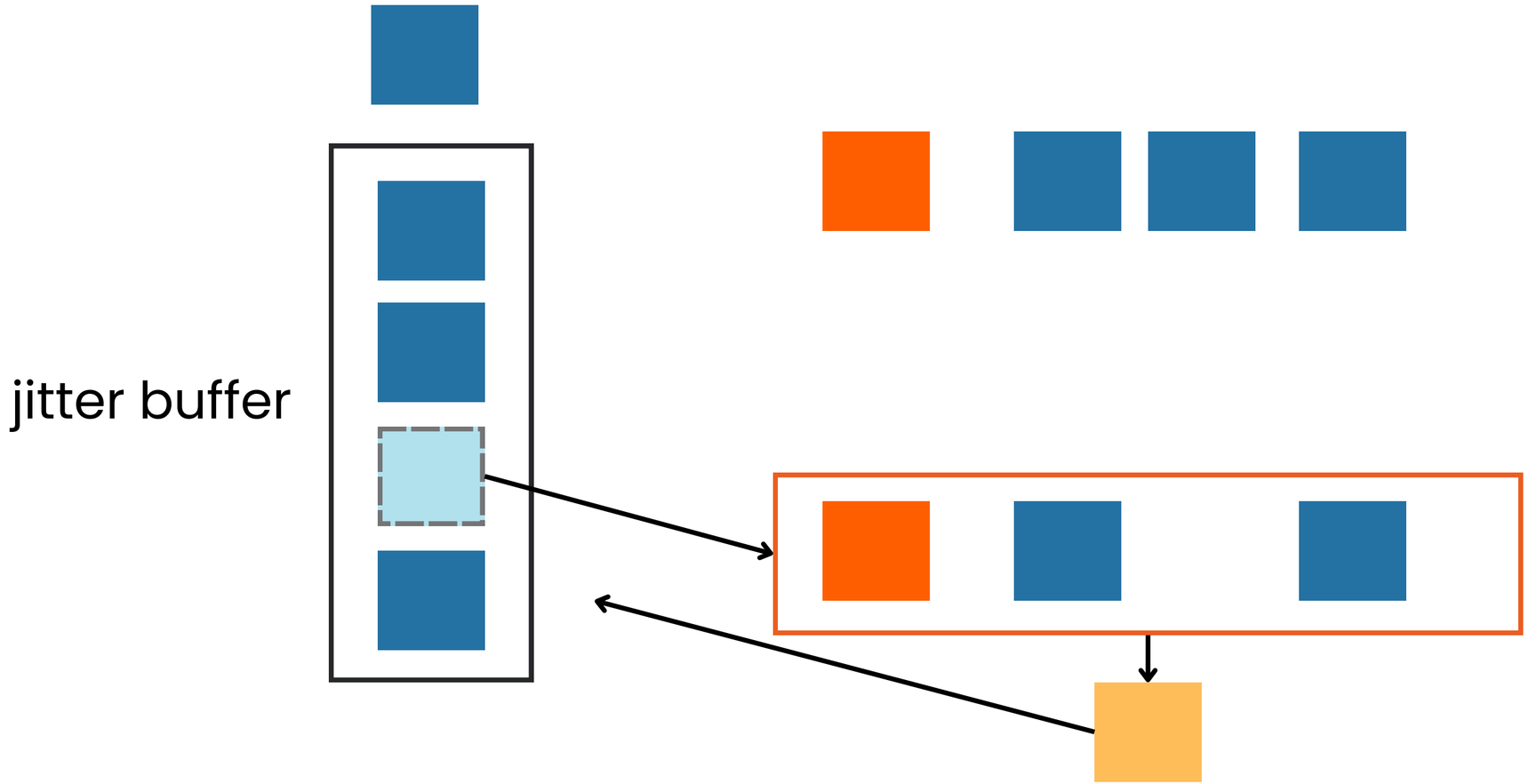
open source VOIP project

REFERENCES

- Linphone: https://www.linphone.org
- RFC 8627, RTP Payload Format for Flexible Forward Error Correction (FEC), 2019, https://datatracker.ietf.org/doc/html/rfc8627
- RFC 3550, RTP: A Transport Protocol for Real-Time Applications, 2003, https://datatracker.ietf.org/doc/html/rfc3550
- RFC 8854, WebRTC Forward Error Correction Requirements: https://datatracker.ietf.org/doc/html/rfc8854
- Source code linphone SDK: https://gitlab.linphone.org/BC/public/linphone-sdk

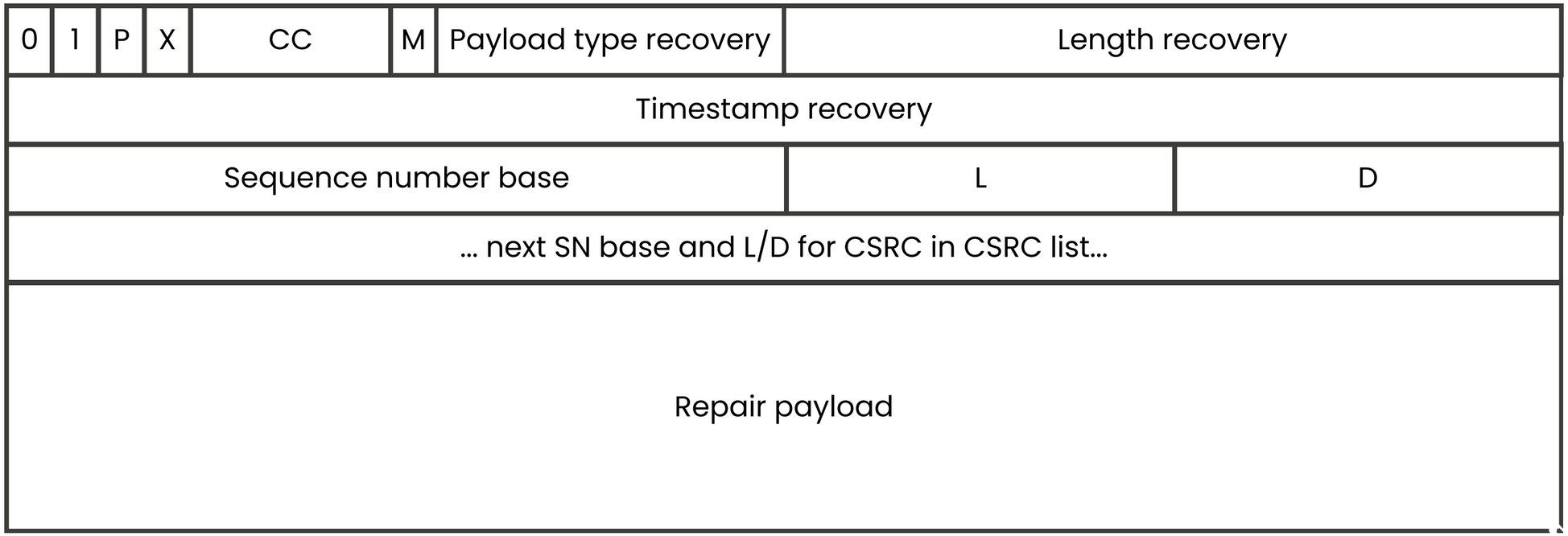
APPENDIX

RECEIVING FEC STREAM IN LINPHONE



FEC HEADER

- fixed L columns and D rows



L and D can vary for different FEC blocks

ITERATIVE DECODING ALGORITHM (RFC 8627)

1. Set num_recovered_until_this_iteration to zero

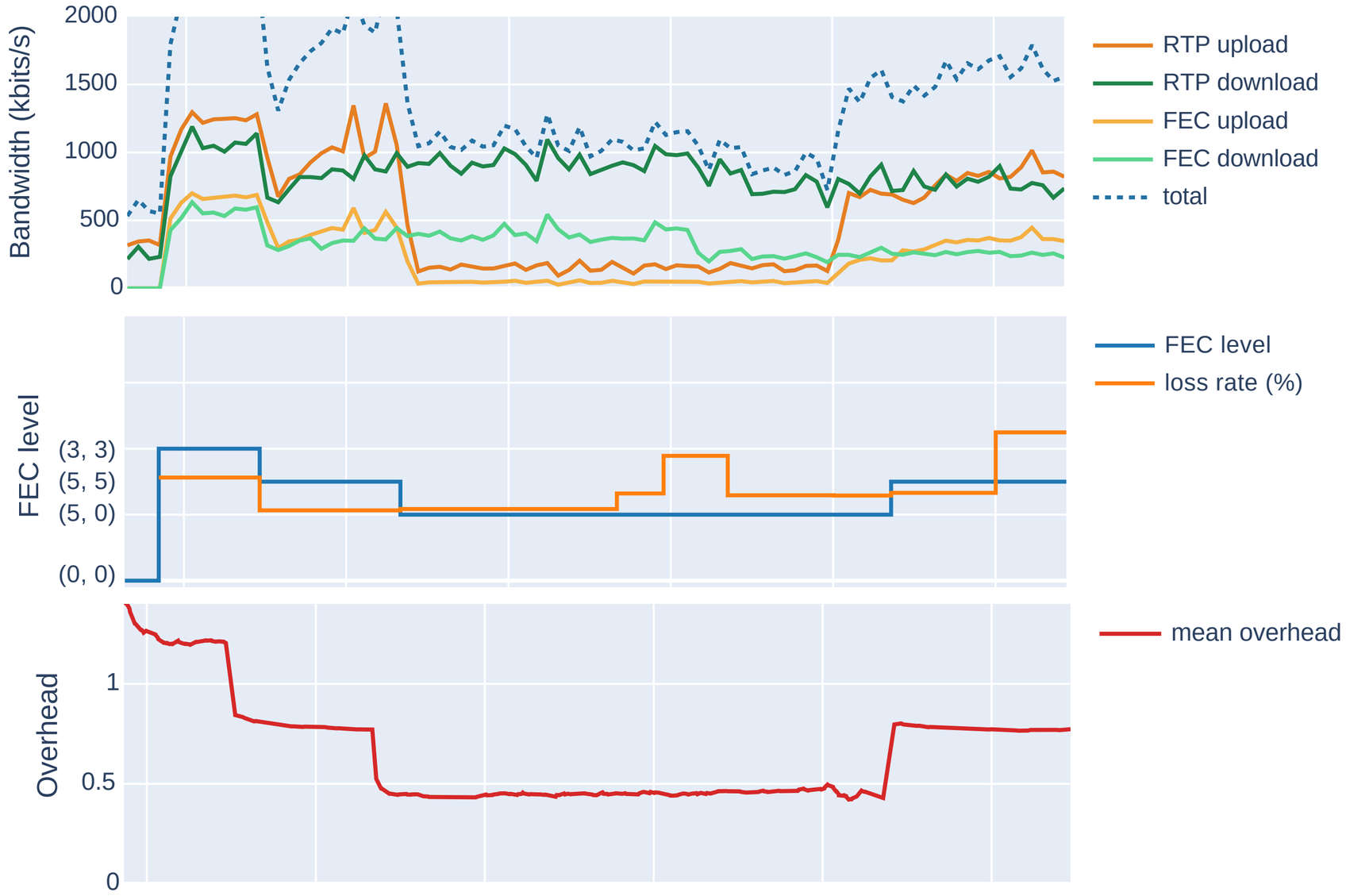
2. Set num_recovered_so_far to zero

3. Recover as many source packets as possible by using the non-interleaved FEC repair packets and increase the value of num_recovered_so_far by the number of recovered source packets.

4. Recover as many source packets as possible by using the interleaved FEC repair packets and increase the value of num_recovered_so_far by the number of recovered source packets.

5. If num_recovered_so_far > num_recovered_until_this_iteration
       num_recovered_until_this_iteration = num_recovered_so_far
       Go to step 3.
   Else
       Terminate

ADAPTABILITY

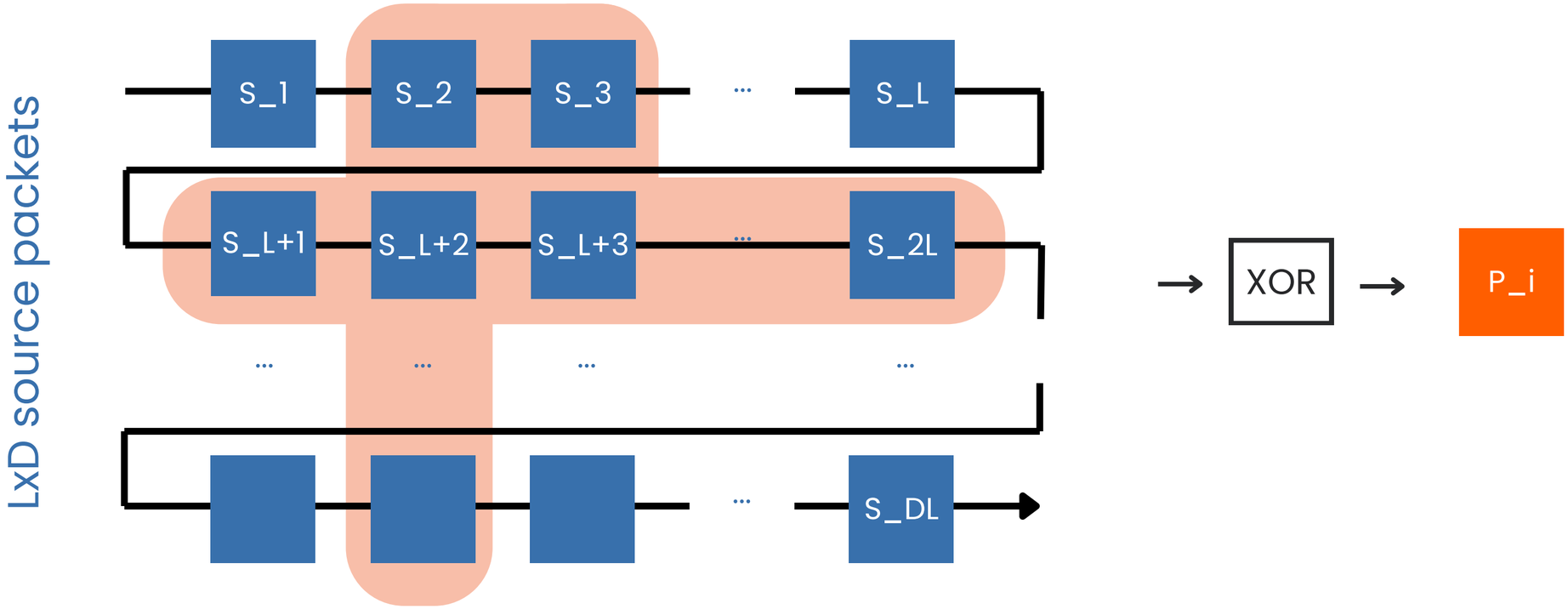


CALL

- Repair window: impacts the buffering
  - sender proposes a value in microseconds
  - rejected by receiver if it can’ t support the value
- Any unknown option can be ignored in the answer and FEC deleted

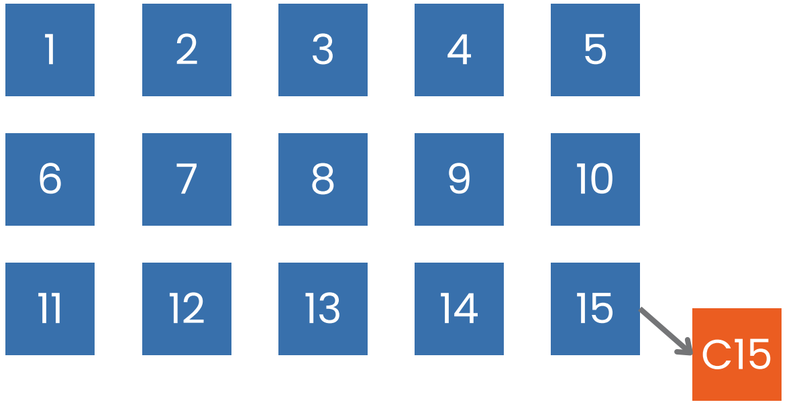
```
v=0
o=marie%20laroueverte_6jr06r~ 153 2733 IN IP6
s=Talk
c=IN IP6 2a01:e0a:27e:8210:fc05:3486:429e:714a
t=0 0
a=rtcp-xr:rcvr-rtt=all:10000 stat-summary=loss
a=group:BUNDLE as vs
m=audio 35787 RTP/AVP 0 101
a=rtpmap:101 telephone-event/8000
a=mid:as
a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid
a=rtcp:58774
a=rtcp-fb:* trr-int 5000
a=rtcp-fb:* ccm tmmbr
m=video 51437 RTP/AVP 96 97
a=rtpmap:96 AV1/90000
a=rtpmap:97 flexfec/90000
a=fmtp:97 repair-window=200000
```

FEC PROTECTION WITH FLEXIBLE MASK



- must send a bitmap with the mask
- dynamic and flexible protection

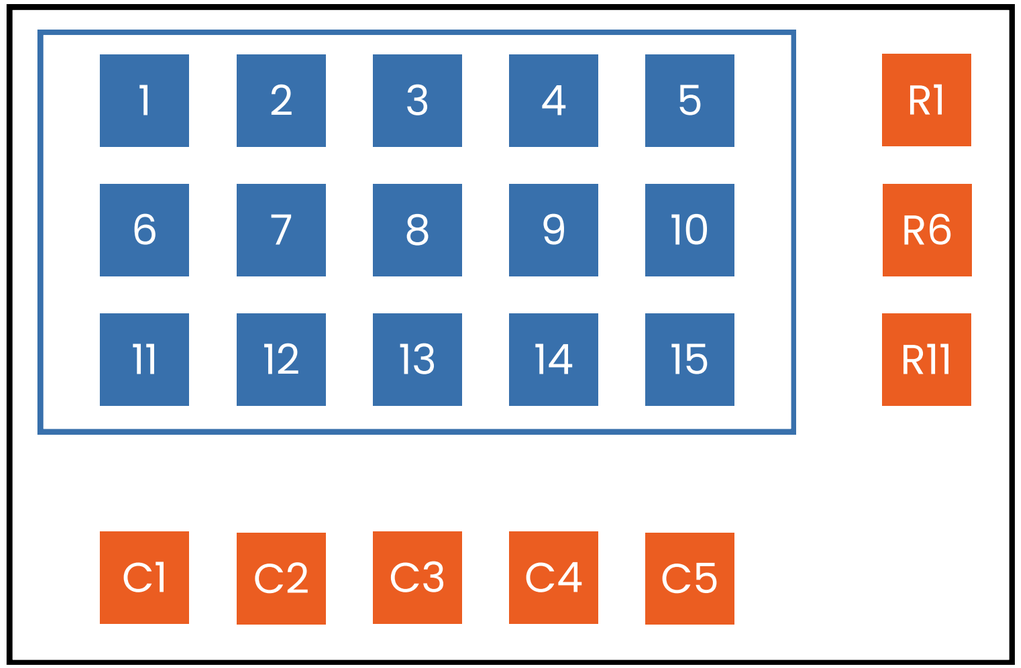
REPAIR PACKET CONSTRUCTION



- Fixed L columns and D rows
  - L = 0, D = 0: ignore
  - L > 0, D = 0: row FEC, source packets: SN, SN+1, SN+(L-1)
  - L > 0, D = 1: row FEC followed by column, all columns following all rows repair packets
  - L > 0, D > 1: column FEC, source packets: SN, SN+L, ..., SN+(D-1)*L
  - L = 1, D = 0: retransmission of a single source packet

BLOCKS

source block



FEC block