Streaming Erasure Codes under Mismatched Source-Channel Frame Rates

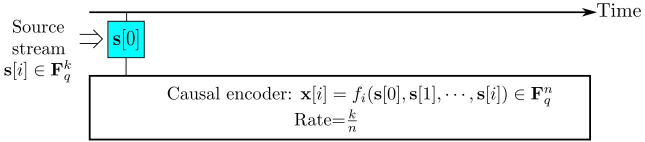
Pratik Patil
University of Toronto

Joint work with Ahmed Badr (U of T) and Ashish Khisti (U of T)

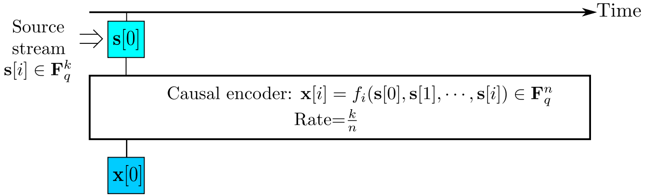
CWIT 2013

Basic delay-sensitive streaming setup (matched scenario)

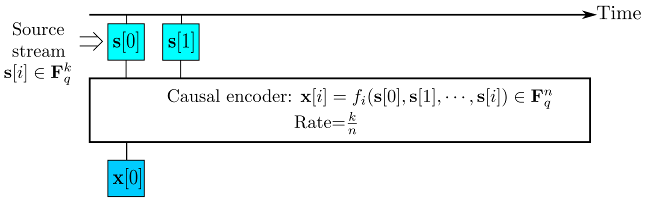
Basic delay-sensitive streaming setup (matched scenario)



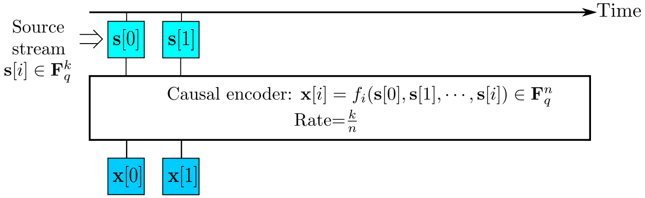
Basic delay-sensitive streaming setup (matched scenario)



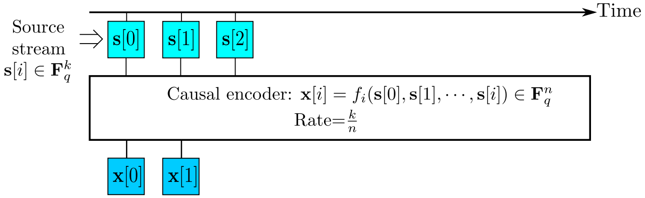
Basic delay-sensitive streaming setup (matched scenario)



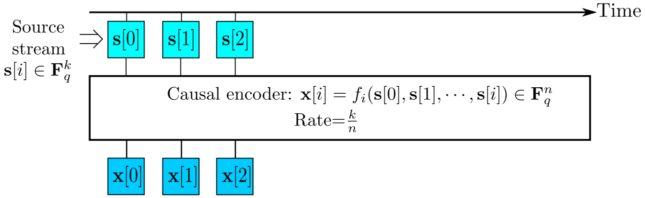
Basic delay-sensitive streaming setup (matched scenario)



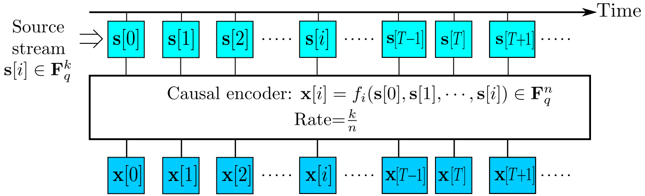
Basic delay-sensitive streaming setup (matched scenario)



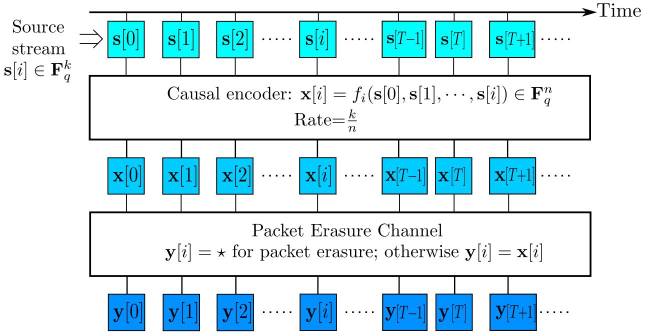
Basic delay-sensitive streaming setup (matched scenario)



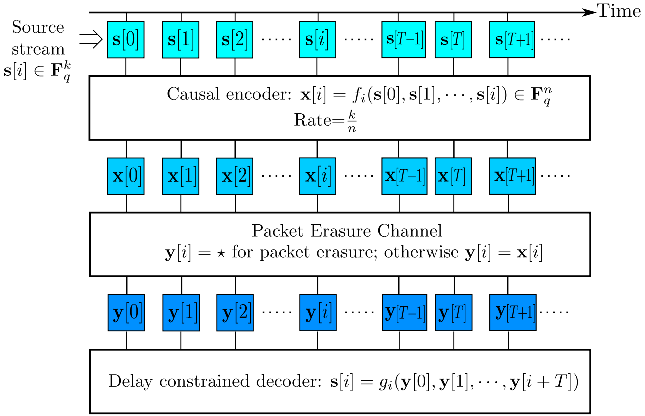
Basic delay-sensitive streaming setup (matched scenario)



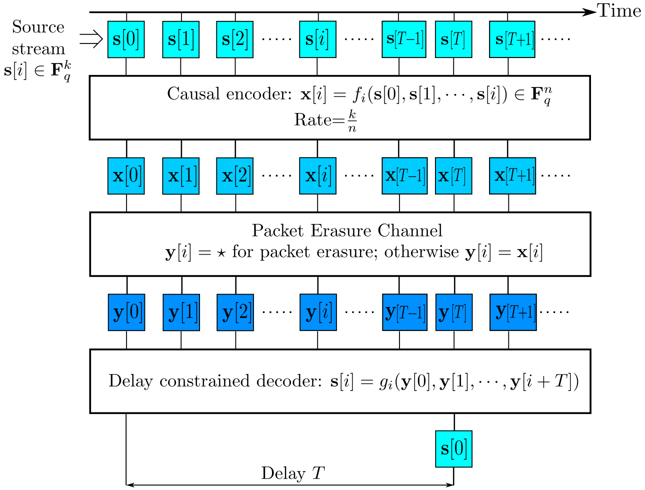
Basic delay-sensitive streaming setup (matched scenario)



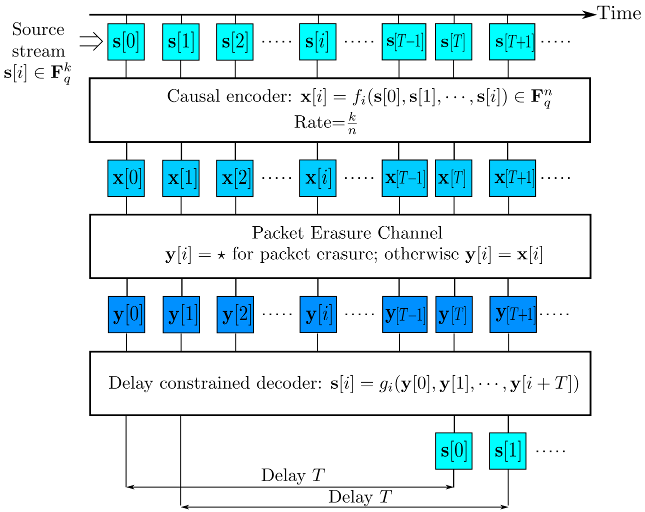
Basic delay-sensitive streaming setup (matched scenario)



Basic delay-sensitive streaming setup (matched scenario)



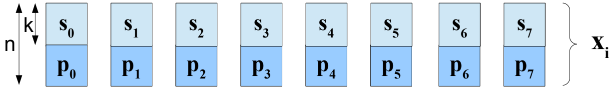
Basic delay-sensitive streaming setup (matched scenario)



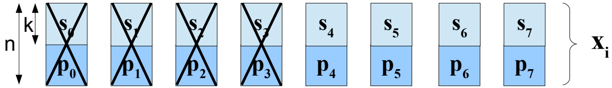
What codes are optimal for such a delay-constrained setup?

- How about random linear codes?

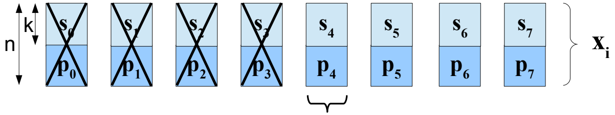
- How about random linear codes?



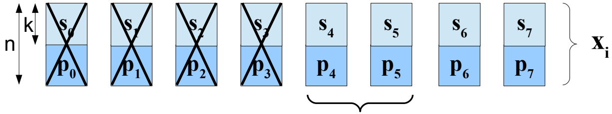
- How about random linear codes?



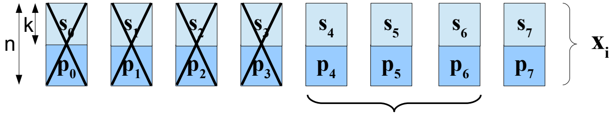
- How about random linear codes?



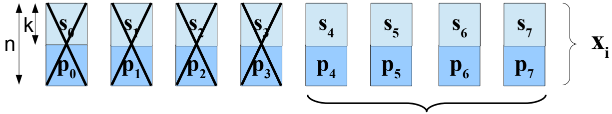
- How about random linear codes?



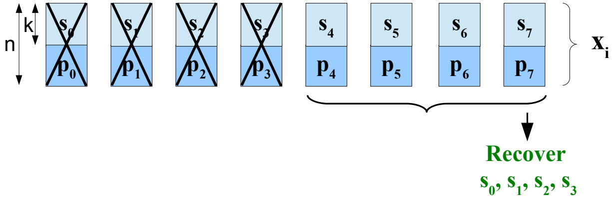
- How about random linear codes?



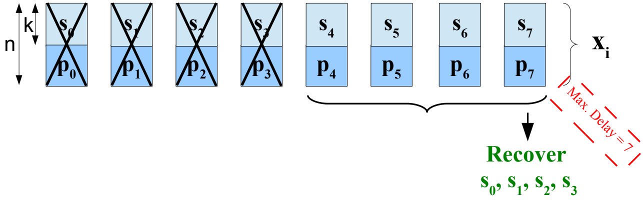
- How about random linear codes?



- How about random linear codes?

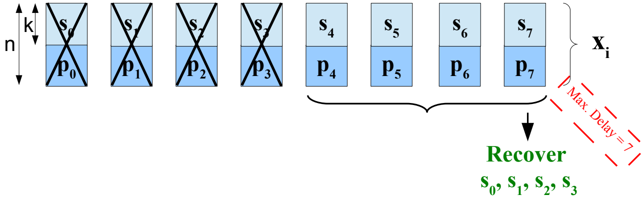


- How about random linear codes?



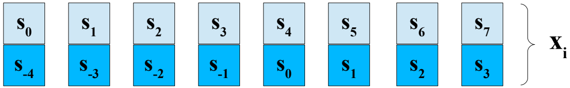
Lost symbols recovered simultaneously once sufficient parities are available!

- How about random linear codes?

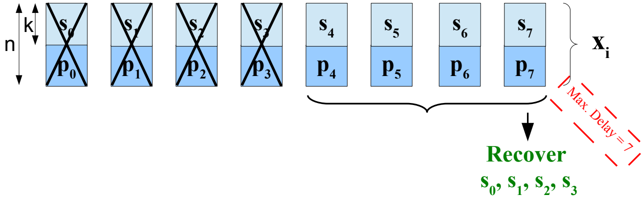


Lost symbols recovered simultaneously once sufficient parities are available!

- What about just repetition?

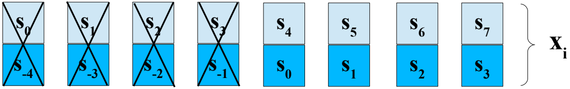


- How about random linear codes?

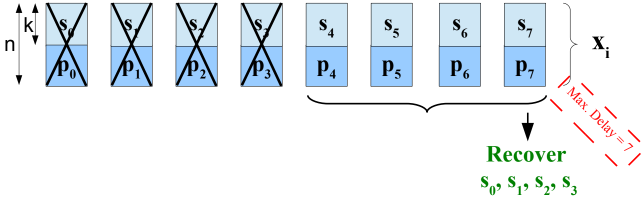


Lost symbols recovered simultaneously once sufficient parities are available!

- What about just repetition?

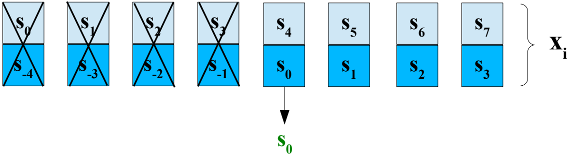


- How about random linear codes?

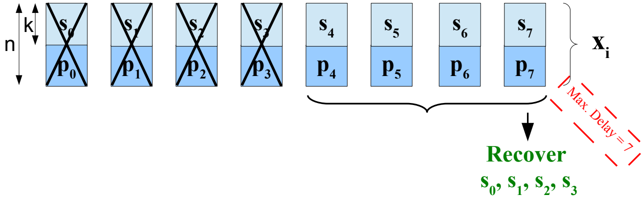


Lost symbols recovered simultaneously once sufficient parities are available!

- What about just repetition?

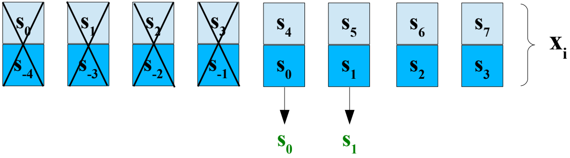


- How about random linear codes?

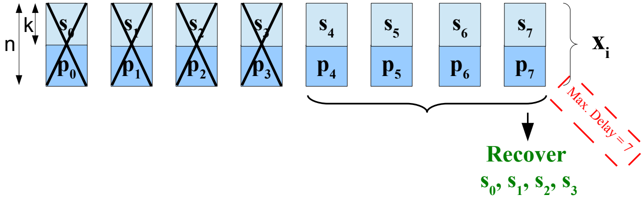


Lost symbols recovered simultaneously once sufficient parities are available!

- What about just repetition?

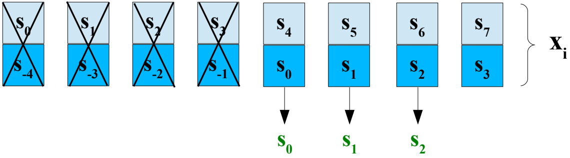


- How about random linear codes?

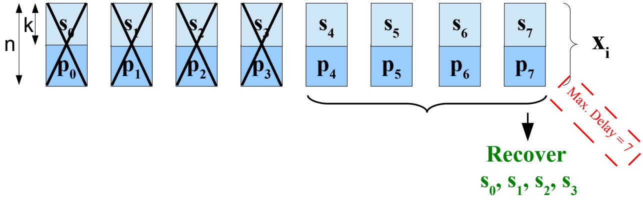


Lost symbols recovered simultaneously once sufficient parities are available!

- What about just repetition?

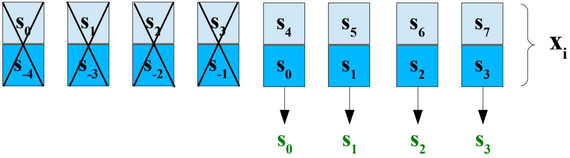


- How about random linear codes?

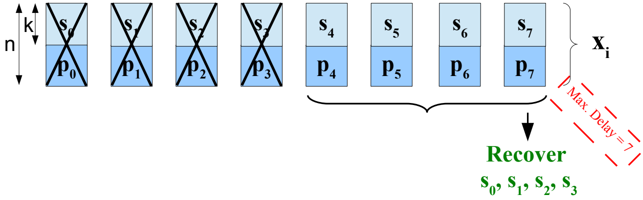


Lost symbols recovered simultaneously once sufficient parities are available!

- What about just repetition?

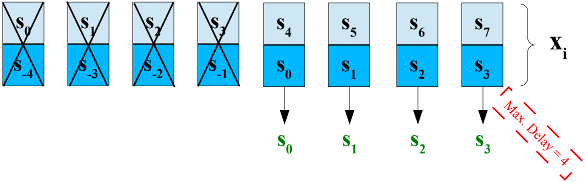


- How about random linear codes?



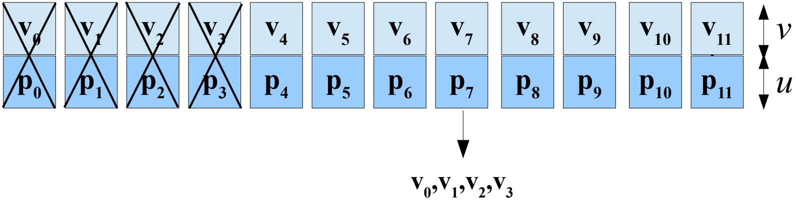
Lost symbols recovered simultaneously once sufficient parities are available!

- What about just repetition?

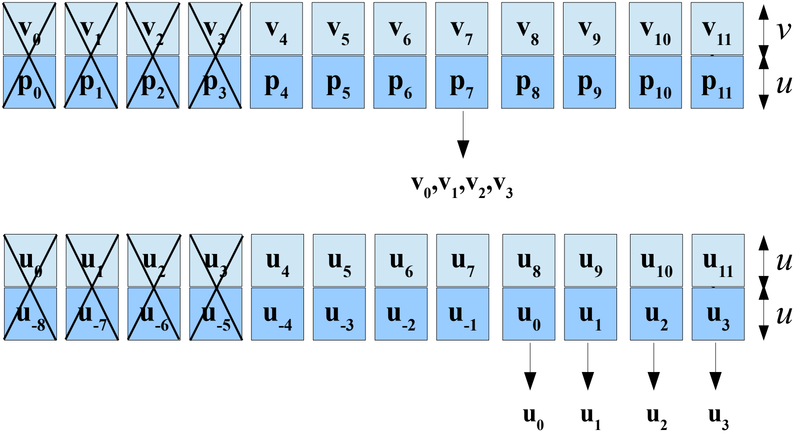


Rate is only half!

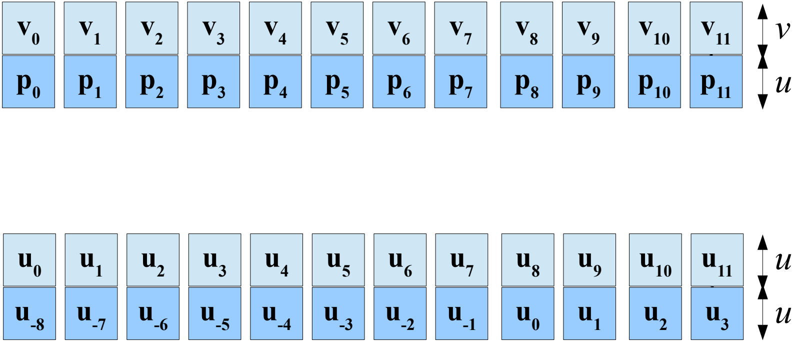
Layered Architecture



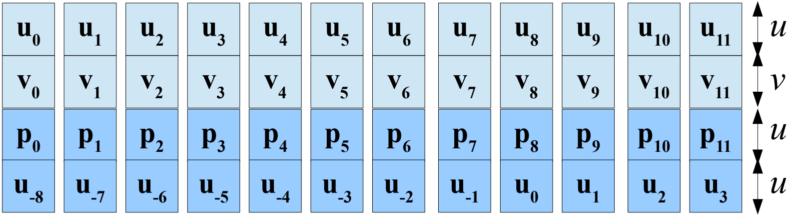
Layered Architecture



Layered Architecture

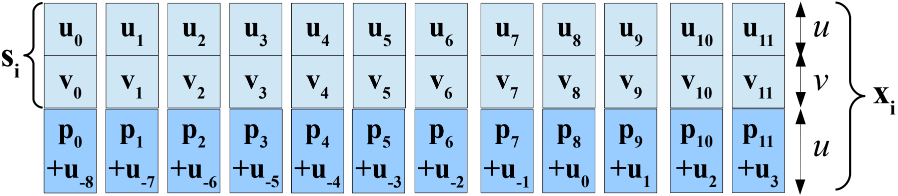


Layered Architecture



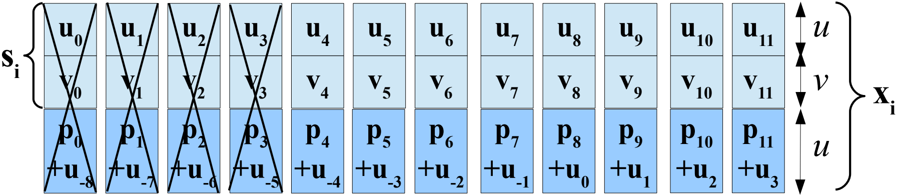
$$R = \frac{u+v}{3u+v}$$

Layered Architecture



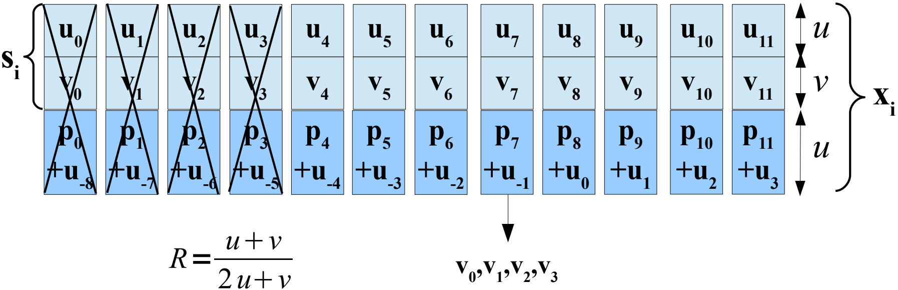
$$R = \frac{u+v}{2u+v}$$

Layered Architecture



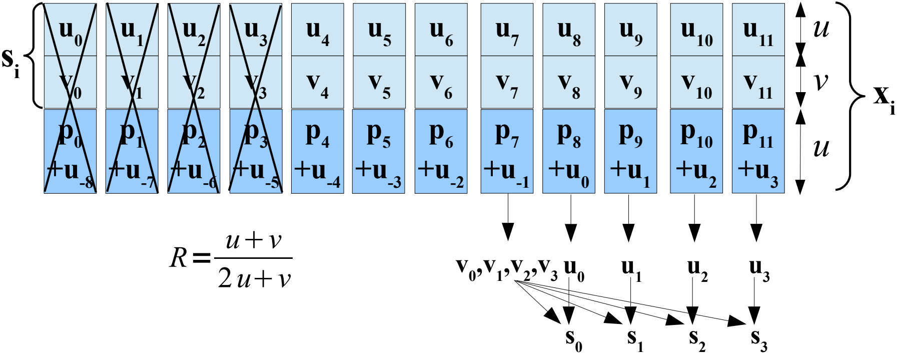
$$R = \frac{u+v}{2u+v}$$

Layered Architecture



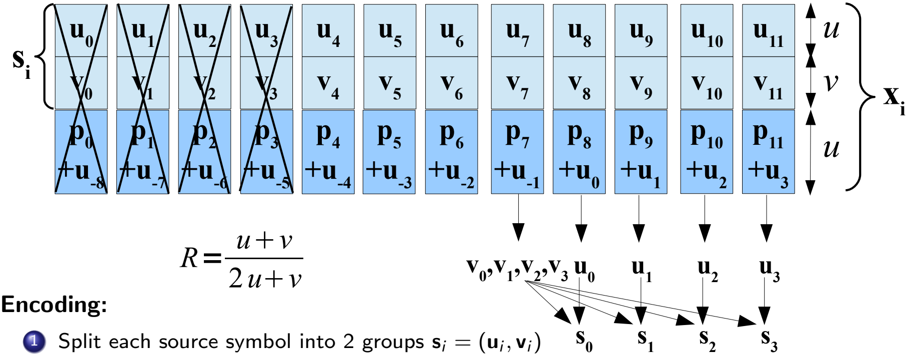
$$R = \frac{u+v}{2u+v}$$

Layered Architecture



$$R = \frac{u+v}{2u+v}$$

Layered Architecture

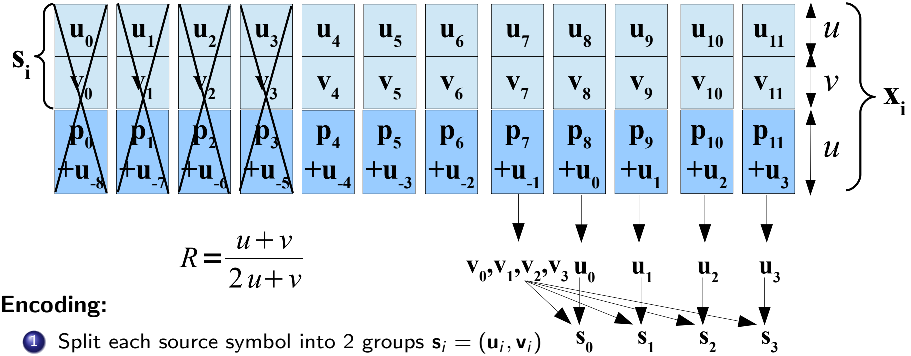


$$R = \frac{u+v}{2u+v}$$

Encoding:

1. Split each source symbol into 2 groups $\mathbf{s}_i = (\mathbf{u}_i, \mathbf{v}_i)$
2. Apply random linear code to the $\mathbf{v}_i$ stream generating $\mathbf{p}_i$ parities
3. Repeat the $\mathbf{u}_i$ symbols with a shift of T
4. Combine the repeated $\mathbf{u}_i$'s with the $\mathbf{p}_i$'s

Layered Architecture



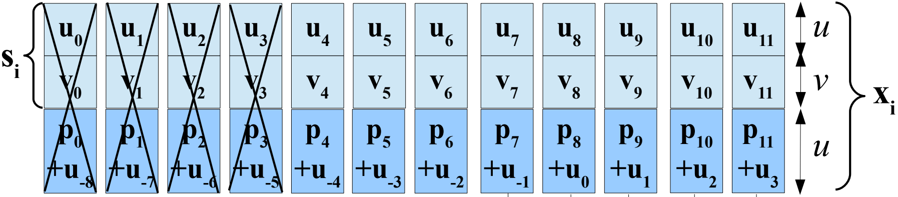
$$R = \frac{u+v}{2u+v}$$

Encoding:

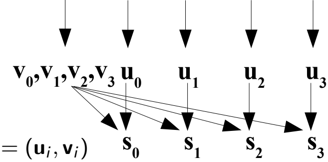
1. Split each source symbol into 2 groups $\mathbf{s}_i = (\mathbf{u}_i, \mathbf{v}_i)$
2. Apply random linear code to the $\mathbf{v}_i$ stream generating $\mathbf{p}_i$ parities
3. Repeat the $\mathbf{u}_i$ symbols with a shift of T
4. Combine the repeated $\mathbf{u}_i$'s with the $\mathbf{p}_i$'s

- Choosing $u = B$ and $v = T - B$, $R = \frac{T}{T+B}$ (Optimal) [Badr, Khisti-Infocom '13]

Layered Architecture



$$R = \frac{u+v}{2u+v}$$



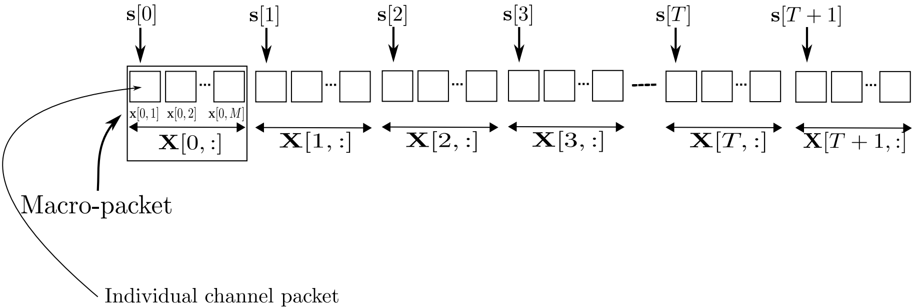
Encoding:

1. Split each source symbol into 2 groups $\mathbf{s}_i = (\mathbf{u}_i, \mathbf{v}_i)$
2. Apply random linear code to the $\mathbf{v}_i$ stream generating $\mathbf{p}_i$ parities
3. Repeat the $\mathbf{u}_i$ symbols with a shift of T
4. Combine the repeated $\mathbf{u}_i$'s with the $\mathbf{p}_i$'s

- Choosing $u = B$ and $v = T - B$, $R = \frac{T}{T+B}$ (Optimal) [Badr, Khisti-Infocom '13]
- Capacity first analyzed by Martinian and Sundberg (IT-2004) (alternative construction)

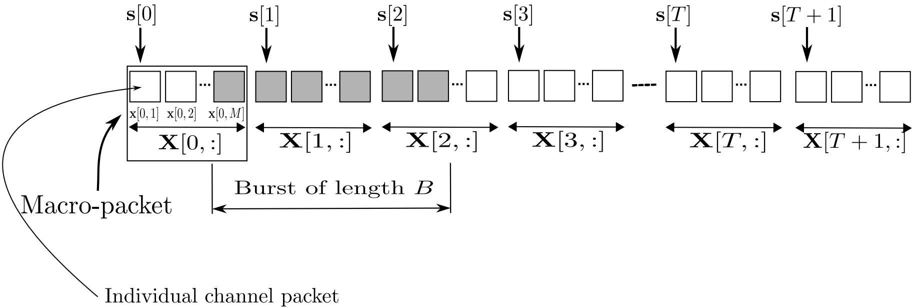
General streaming setup (mismatched scenario)

General streaming setup (mismatched scenario)



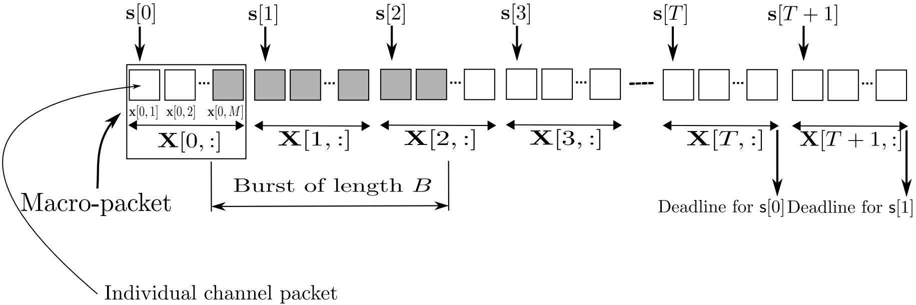
- Source model: i.i.d. process with $\mathbf{s}[i] \sim$ uniform over $\mathbf{F}_q^k$
- Streaming encoder: $\mathbf{x}[i,j] = f_{i,j}(\mathbf{s}[0], \mathbf{s}[1], \cdots, \mathbf{s}[i]) \in \mathbf{F}_q^n$
- Macro-packet: $\mathbf{X}[i,:] = [\mathbf{x}[i,1] \mid \ldots \mid \mathbf{x}[i,M]]$
- Rate: $R = \frac{\mathsf{H}(\mathbf{s})}{n \times M} = \frac{k}{n \times M}$

General streaming setup (mismatched scenario)



- Source model: i.i.d. process with $\mathbf{s}[i] \sim$ uniform over $\mathbf{F}_q^k$
- Streaming encoder: $\mathbf{x}[i,j] = f_{i,j}(\mathbf{s}[0], \mathbf{s}[1], \cdots, \mathbf{s}[i]) \in \mathbf{F}_q^n$
- Macro-packet: $\mathbf{X}[i,:] = [\mathbf{x}[i,1] \mid \ \ldots \ \mid \mathbf{x}[i,M]]$
- Rate: $R = \frac{\mathsf{H}(\mathbf{s})}{n \times M} = \frac{k}{n \times M}$
- Packet erasure channel: erasure burst of maximum B channel packets

General streaming setup (mismatched scenario)



- Source model: i.i.d. process with $\mathbf{s}[i] \sim$ uniform over $\mathbf{F}_q^k$
- Streaming encoder: $\mathbf{x}[i,j] = f_{i,j}(\mathbf{s}[0], \mathbf{s}[1], \cdots, \mathbf{s}[i]) \in \mathbf{F}_q^n$
- Macro-packet: $\mathbf{X}[i,:] = [\mathbf{x}[i,1] \mid \ldots \mid \mathbf{x}[i,M]]$
- Rate: $R = \frac{\mathsf{H}(\mathbf{s})}{n \times M} = \frac{k}{n \times M}$
- Packet erasure channel: erasure burst of maximum B channel packets
- Delay-constrained decoder: $\mathbf{s}[i]$ needs to be recovered by macro-packet $i + T$

Main result

Theorem

For the streaming setup considered, with any M, T and B, the streaming capacity C is given by the following expression:

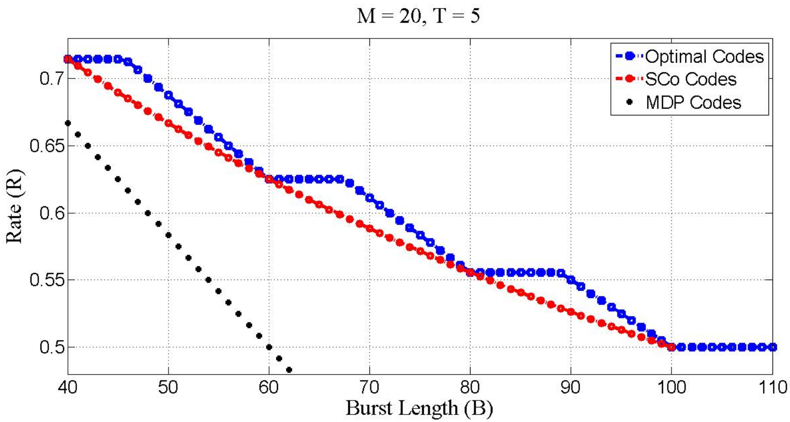
$$C = \begin{cases} \frac{T}{T+b}, & B' \leq \frac{b}{T+b} M,\ \ T \geq b, \\ \frac{M(T+b+1)-B}{M(T+b+1)}, & B' > \frac{b}{T+b} M,\ \ T > b, \\ \frac{M-B'}{M}, & B' > \frac{M}{2},\ \ T = b, \\ 0, & T < b. \end{cases}$$

where the constants b and B' are defined via

$$B = bM + B', \quad B' \in \{0, 1, \ldots, M-1\}, \quad b \in \mathbb{N}^0.$$

□

Numerical Comparison

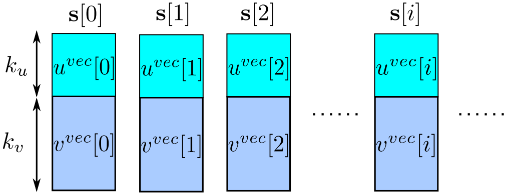


Code construction

Encoding

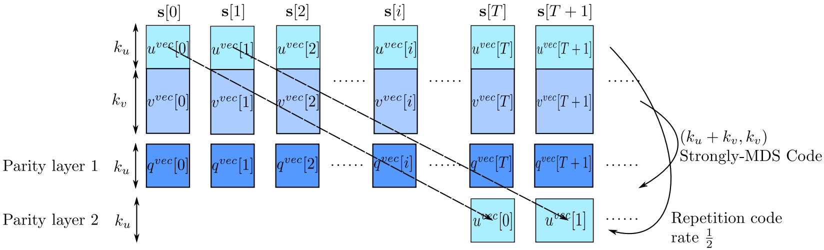
1. Source splitting
   - split $\mathbf{s}[i]$ into k symbols and divide them into two groups, urgent symbols and non-urgent symbols

$$\mathbf{s}[i] = (s_1[i], \ldots, s_k[i]) = (\underbrace{u_1[i], \ldots, u_{k_u}[i]}_{\mathbf{u}^{\mathrm{vec}}[i]}, \underbrace{v_1[i], \ldots, v_{k_v}[i]}_{\mathbf{v}^{\mathrm{vec}}[i]})$$



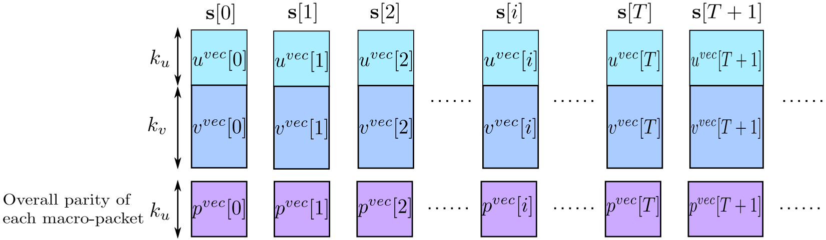
Code construction (cont.)

2. Parity generation
   - layer 1: $(k_v + k_u, k_v, T)$ Strongly-MDS code applied to $\mathbf{v}^{\text{vec}}[\cdot]$ generating $\mathbf{q}^{\text{vec}}[i]$
   - layer 2: repetition code on urgent symbols with a shift of T



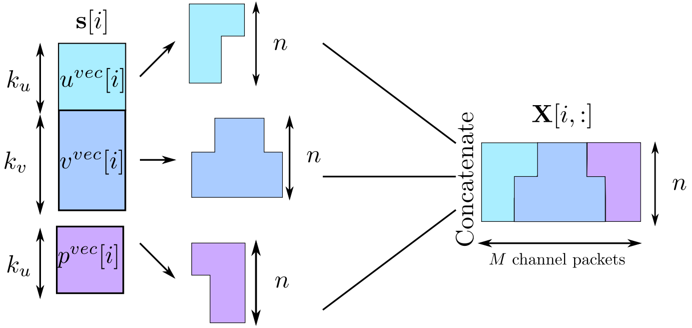
Code construction (cont.)

- Overall combined parity: $\mathbf{p}^{\text{vec}}[i] = \mathbf{q}^{\text{vec}}[i] + \mathbf{u}^{\text{vec}}[i - T]$



Code construction (cont.)

3. Reshaping and macro-packet generation
   - reshape $\mathbf{u}^{\mathrm{vec}}[i]$, $\mathbf{v}^{\mathrm{vec}}[i]$ and $\mathbf{p}^{\mathrm{vec}}[i]$ into groups each of n symbols (recall-each individual packet has n symbols)
   - concatenate groups generated in the last step to form macro-packet $bX[i,:]$ with M channel packets of n symbols each as required

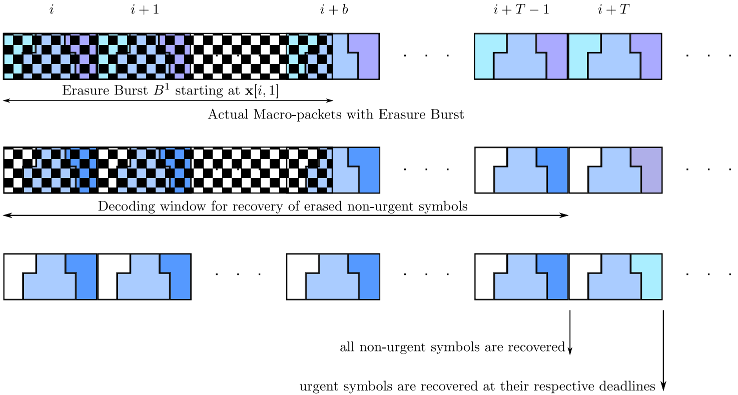


Rate of the code$=\frac{k_u+k_v}{2k_u+k_v}$

Code construction (cont.)

Decoding

1. Step 1: All non-urgent symbol recovered before the first deadline
2. Step 2: Urgent symbols recovered at their respective deadlines



Simulation results

- Two state Gilbert channel (good state, bad state)
- Pr.{good state to bad state}=α, Pr.{bad state to good state}: β

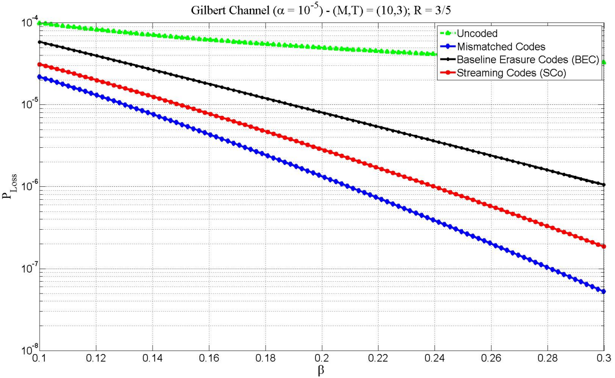


Figure: $(M, T, R) = (10, 3, 3/5)$

Conclusions

1. Extension to previously studied streaming setup ($M = 1$) for the mismatched scenario (general M)
2. Complete characterization of the associated capacity
3. New layered code construction
4. Improvements in packet-loss rate over statistical Gilbert channel
5. What about both burst and isolated erasures?

Thank you for listening!

Any questions/comments/thoughts?