Tambur: efficient loss recovery for videoconferencing via streaming codes

Presented by **Michael Rudow** at NSDI '23

Joint work with Francis Y. Yan, Abhishek Kumar, Ganesh Ananthanarayanan, Martin Ellis, and K.V. Rashmi

Carnegie Mellon University

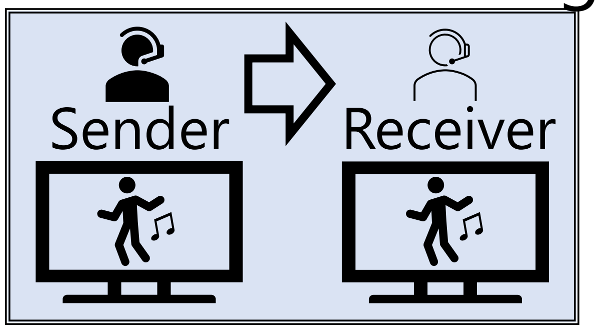
Microsoft



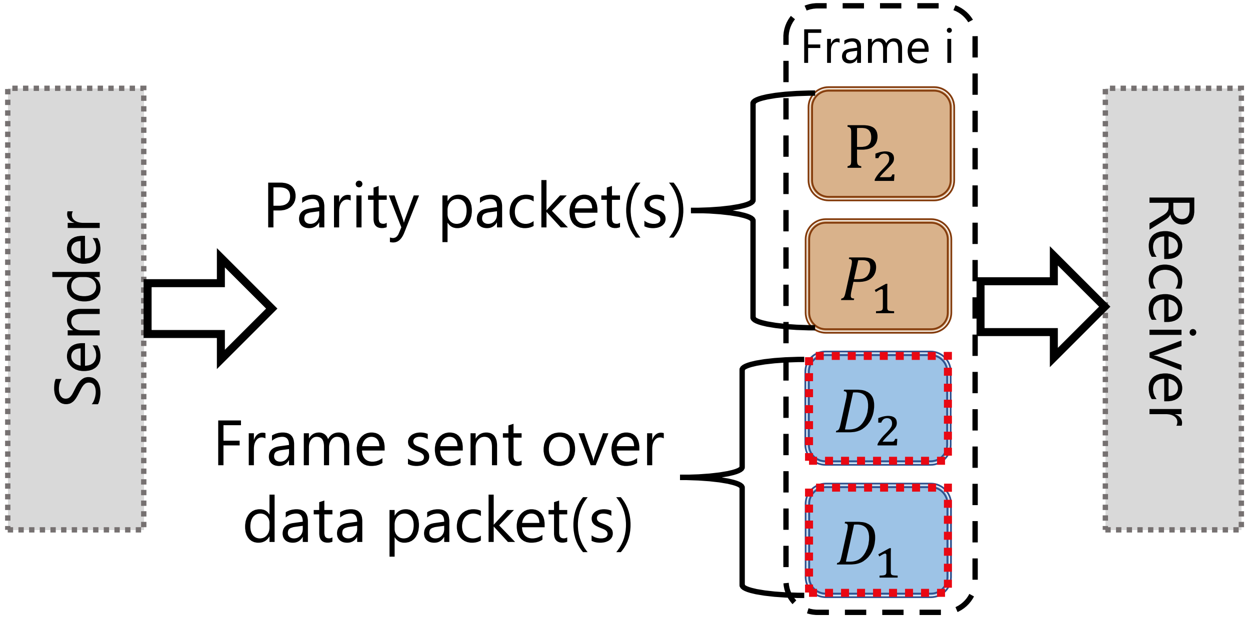



Motivation: packet loss reduces live-streaming QoE

- Streaming applications like videoconferencing (VC)



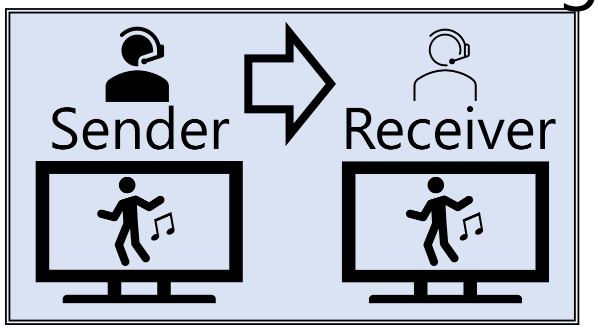
- Transmit sequence of video frames over a lossy network
  - Sending frame i



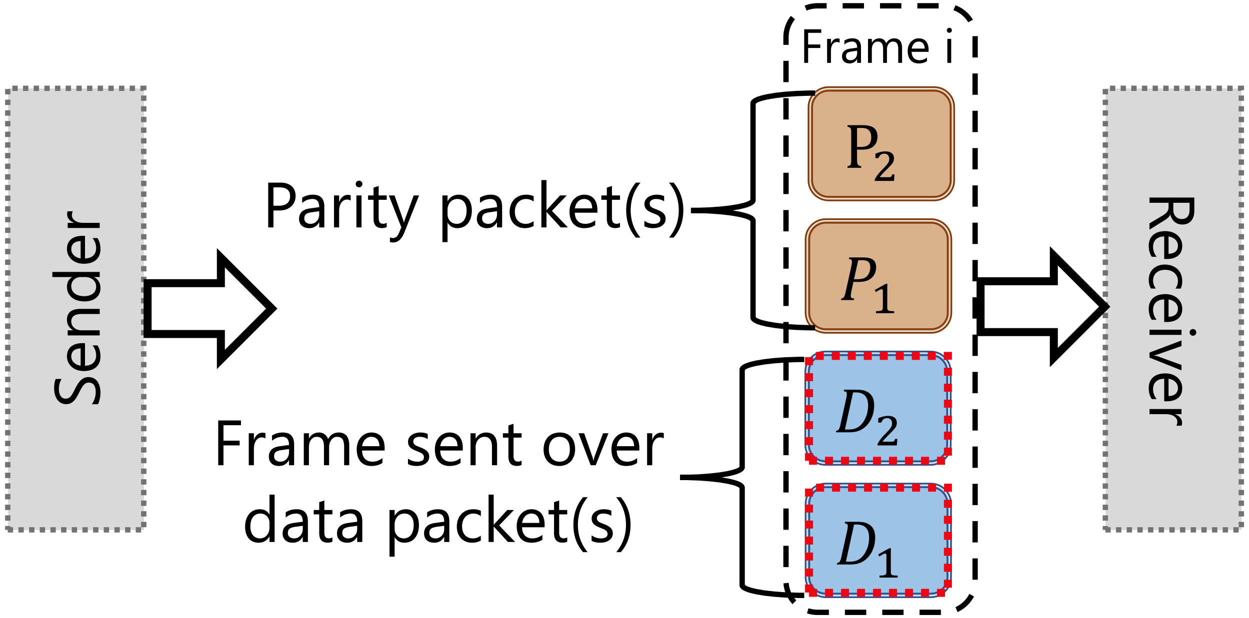
Low-latency packet loss recovery is needed!

Motivation: packet loss reduces live-streaming QoE

- Streaming applications like videoconferencing (VC)



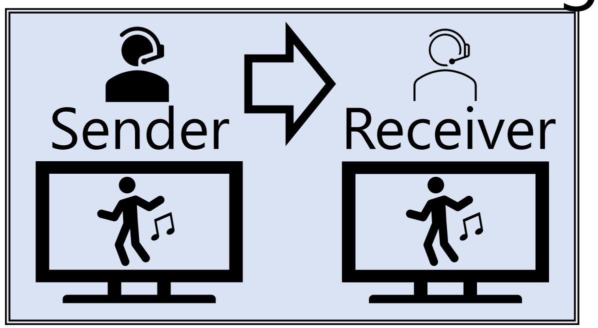
- Transmit sequence of video frames over a lossy network
  - Sending frame i



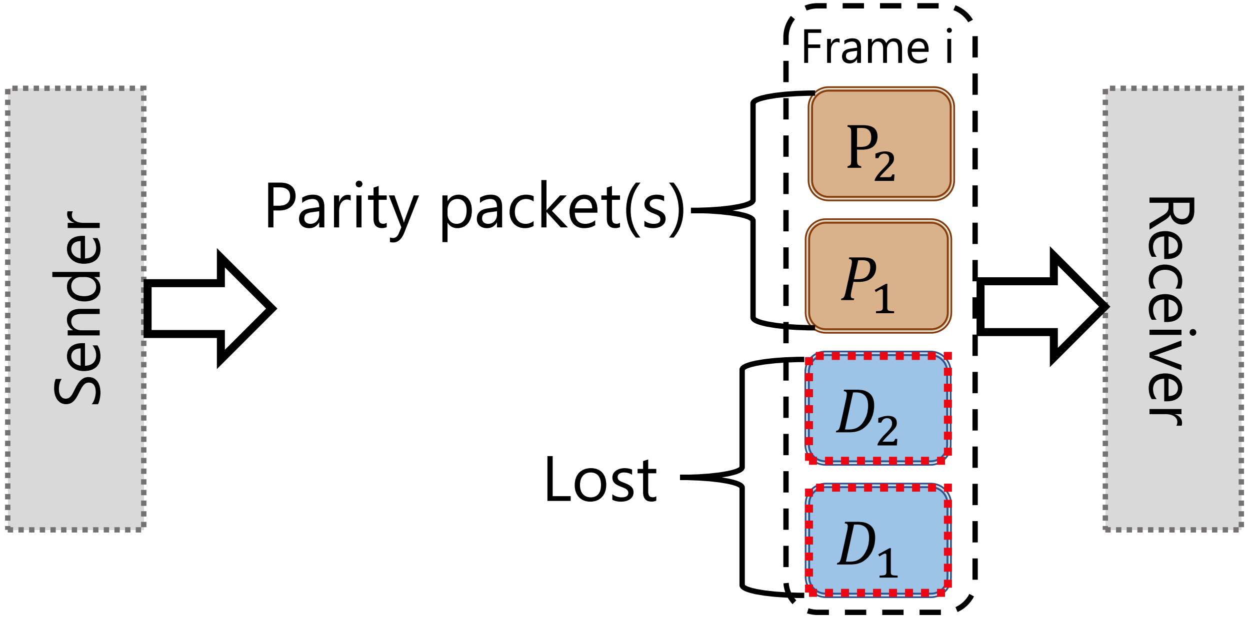
Low-latency packet loss recovery is needed!

Motivation: packet loss reduces live-streaming QoE

- Streaming applications like videoconferencing (VC)



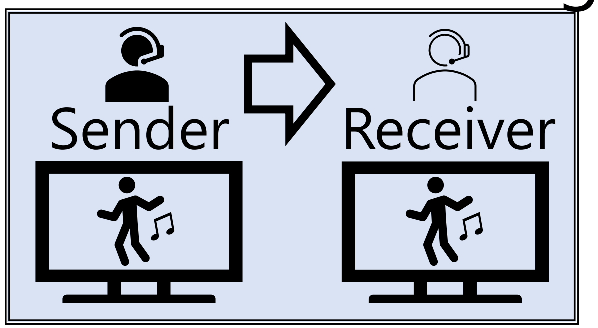
- Transmit sequence of video frames over a lossy network
  - Sending frame i



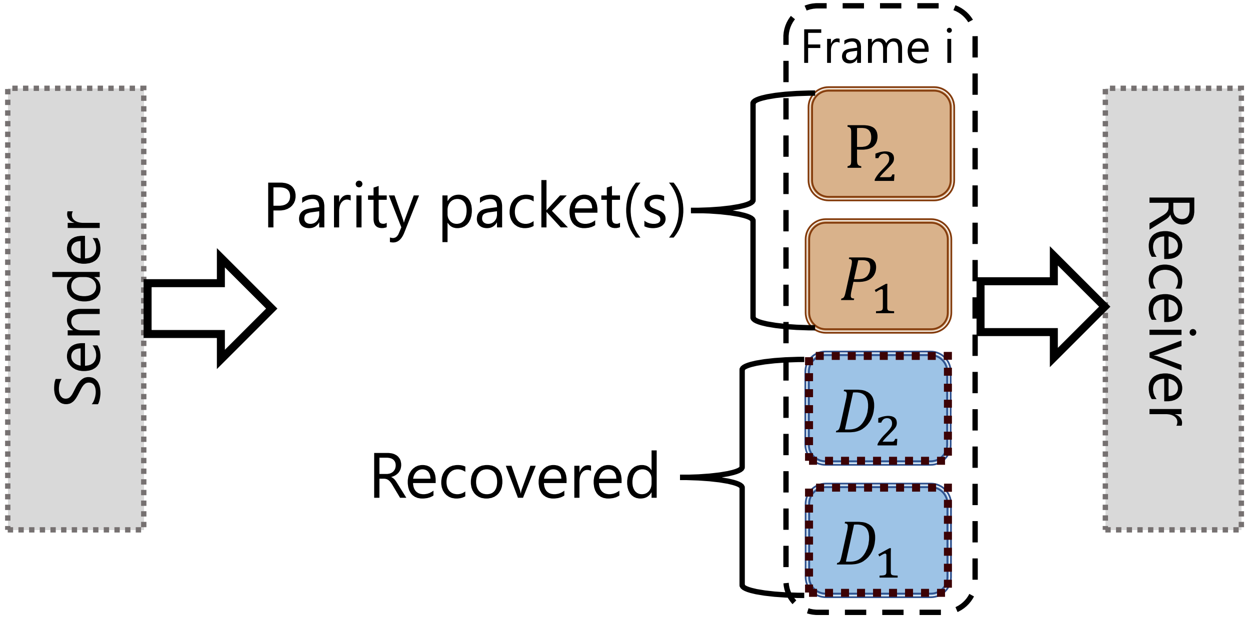
Low-latency packet loss recovery is needed!

Motivation: packet loss reduces live-streaming QoE

- Streaming applications like videoconferencing (VC)



- Transmit sequence of video frames over a lossy network
    - Sending frame i



Low-latency packet loss recovery is needed!

Outline: improve VC QoE via streaming codes

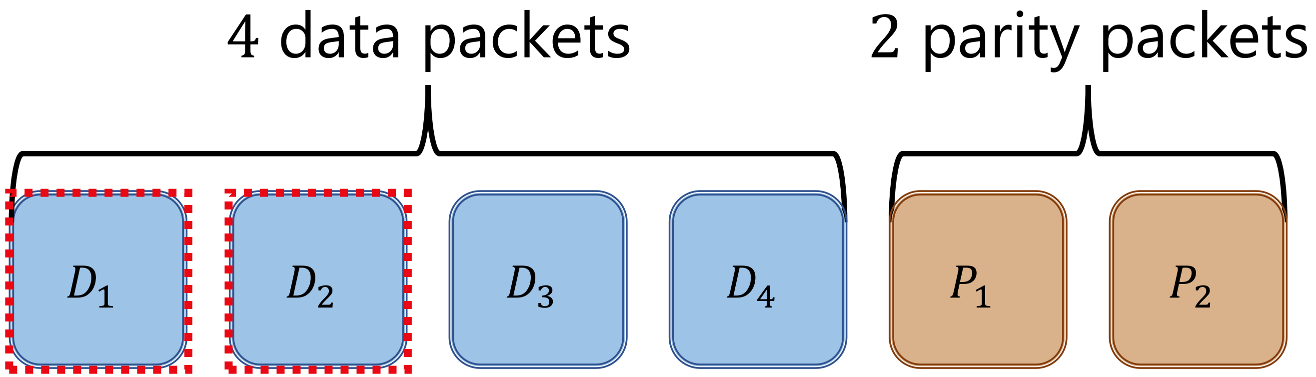
- **Problem**: conventional loss recovery sub-optimal QoE

- **Approach**: new streaming codes for low-latency loss recovery

- **Outcome:** improve key metrics of QoE like video freeze

Conventional loss-recovery is ill-suited to VC

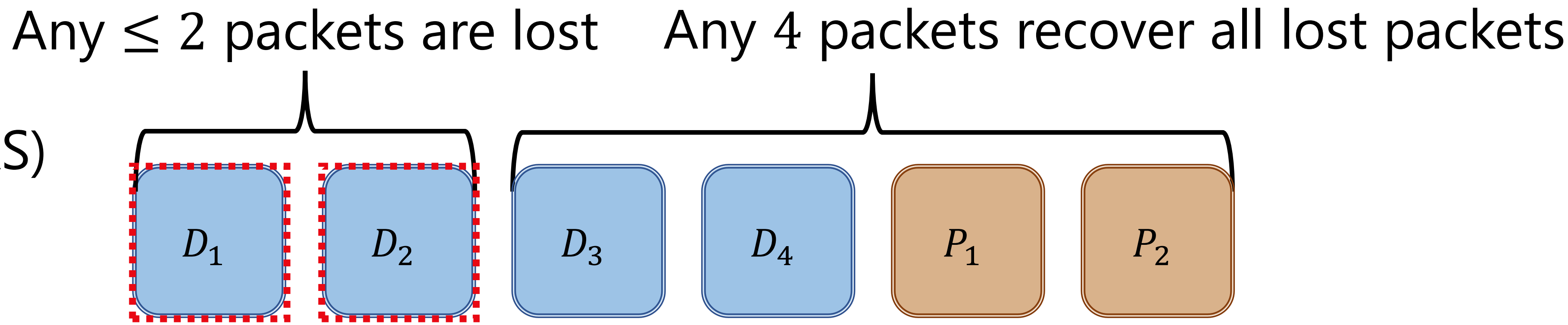
- Retransmission has too high latency if high RTT (e.g., over long-distance)
- Replication requires a 100% BW overhead
- FEC in form of block codes widely used (e.g., by Teams)
  - Reed-Solomon (RS)



  - Traditional erasure codes use **sub-optimal BW** for VC, as we see next

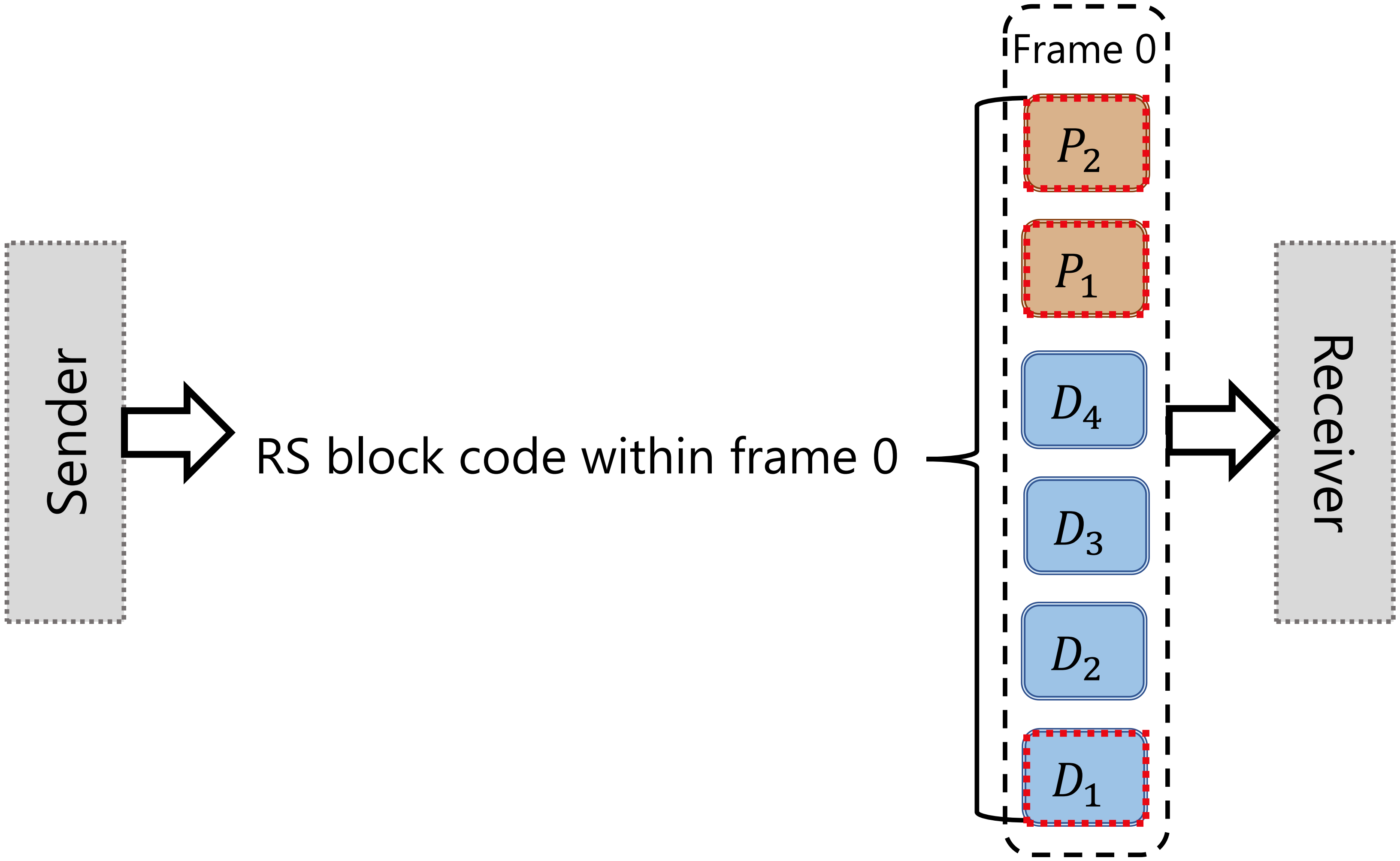
Conventional loss-recovery is ill-suited to VC

- Retransmission has too high latency if high RTT (e.g., over long-distance)
- Replication requires a 100% BW overhead
- FEC in form of block codes widely used (e.g., by Teams)
  - Reed-Solomon (RS)

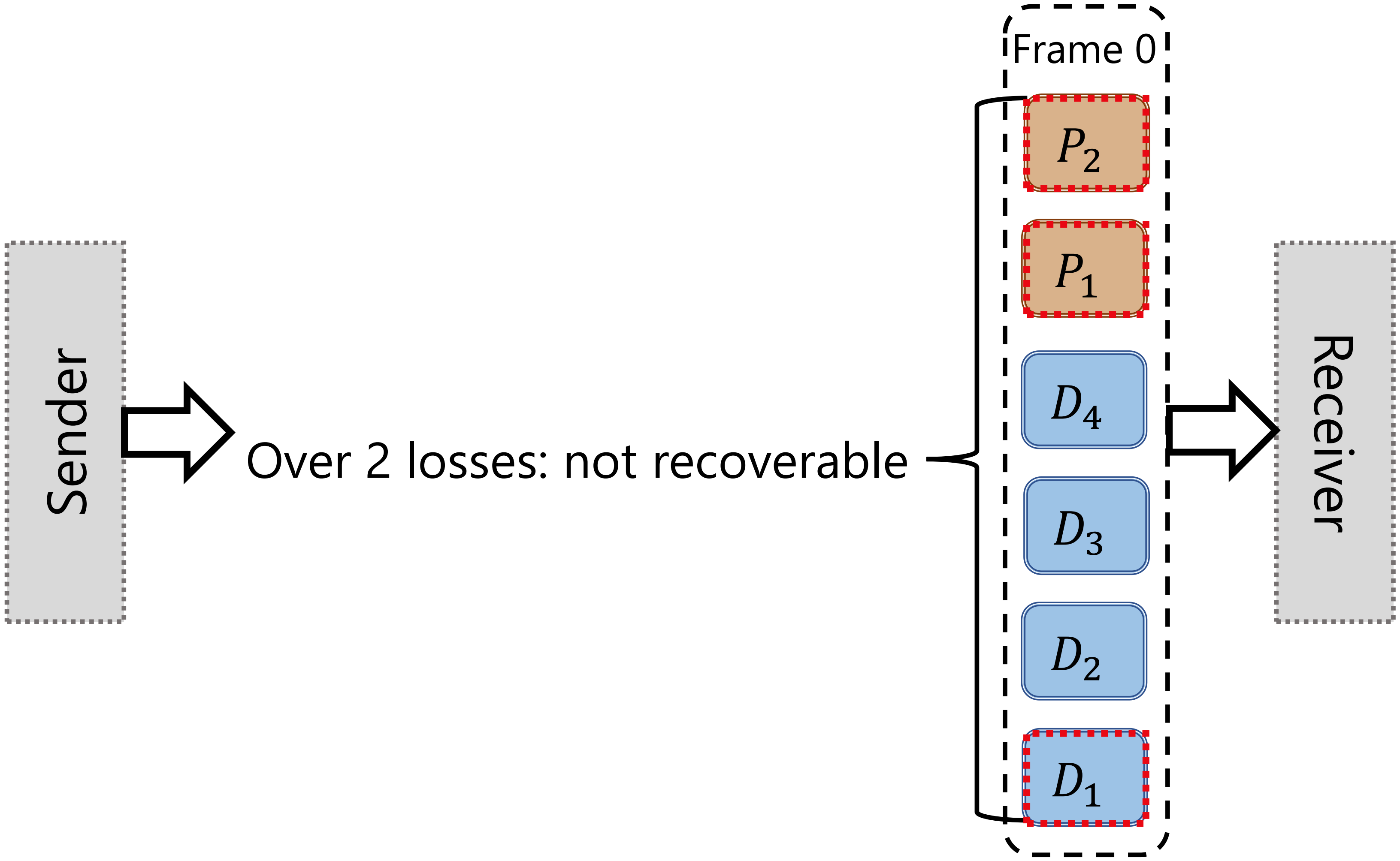


  - Traditional erasure codes use **sub-optimal BW** for VC, as we see next

RS code within each frame wastes parity



RS code within each frame wastes parity

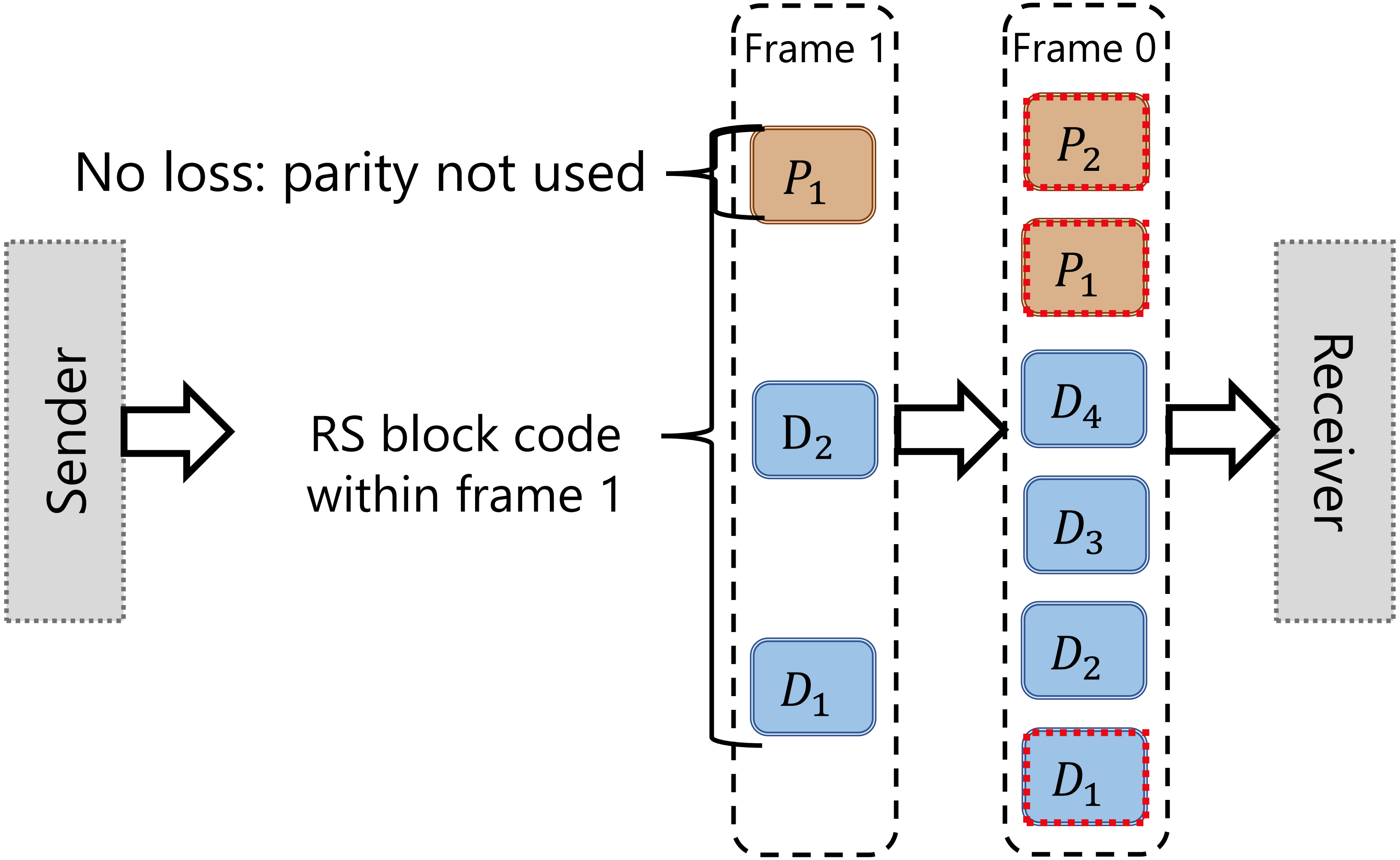


RS code within each frame wastes parity

Drawbacks:

Wasted parity for frame 1 not useful for frame 0

Freeze: frame 1 not playable without frame 0

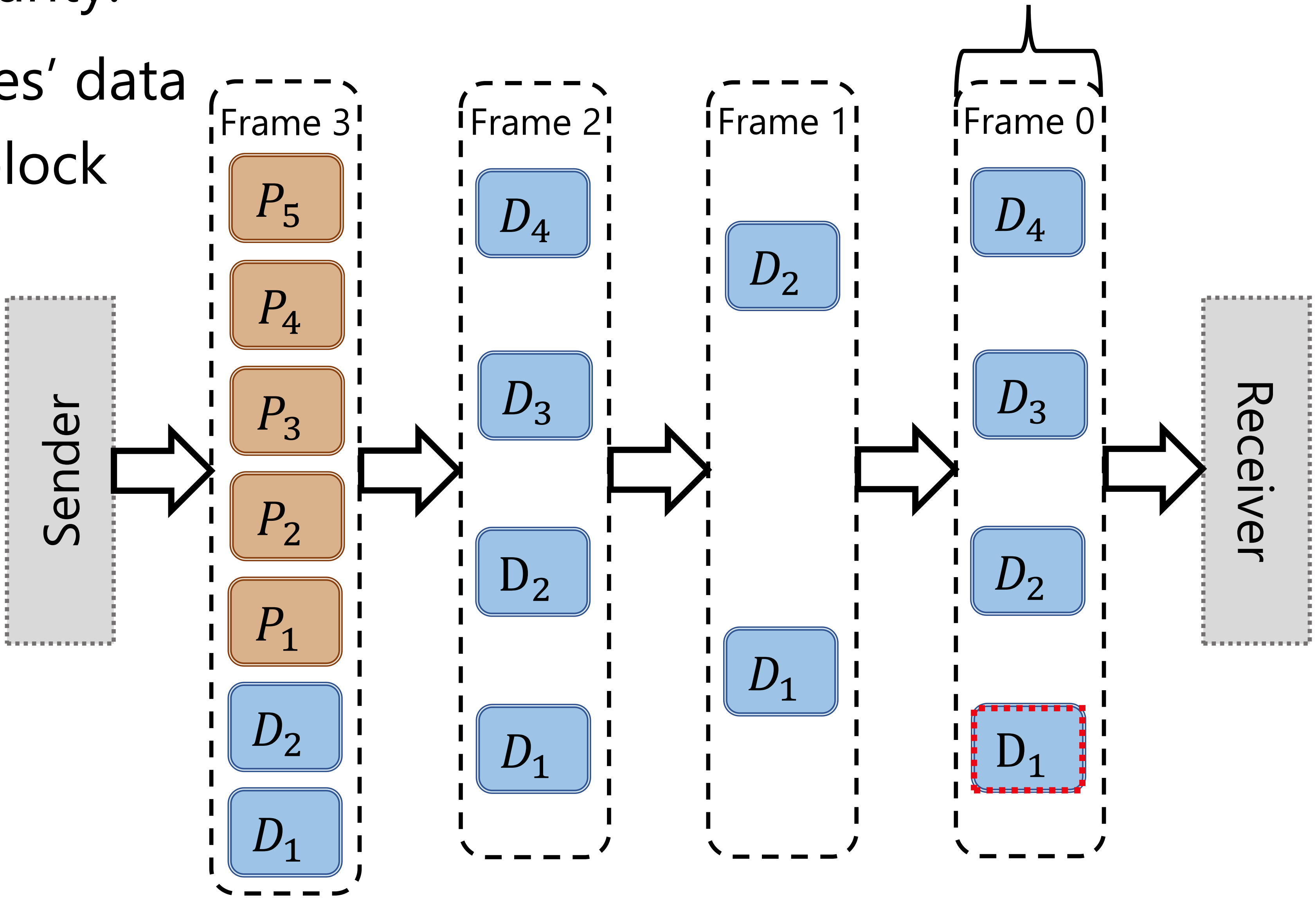


RS across frames costs latency and spikes BW

Quick fix for wasted parity:

Block code for 4 frames' data

Parity sent at end of block



RS across frames costs latency and spikes BW

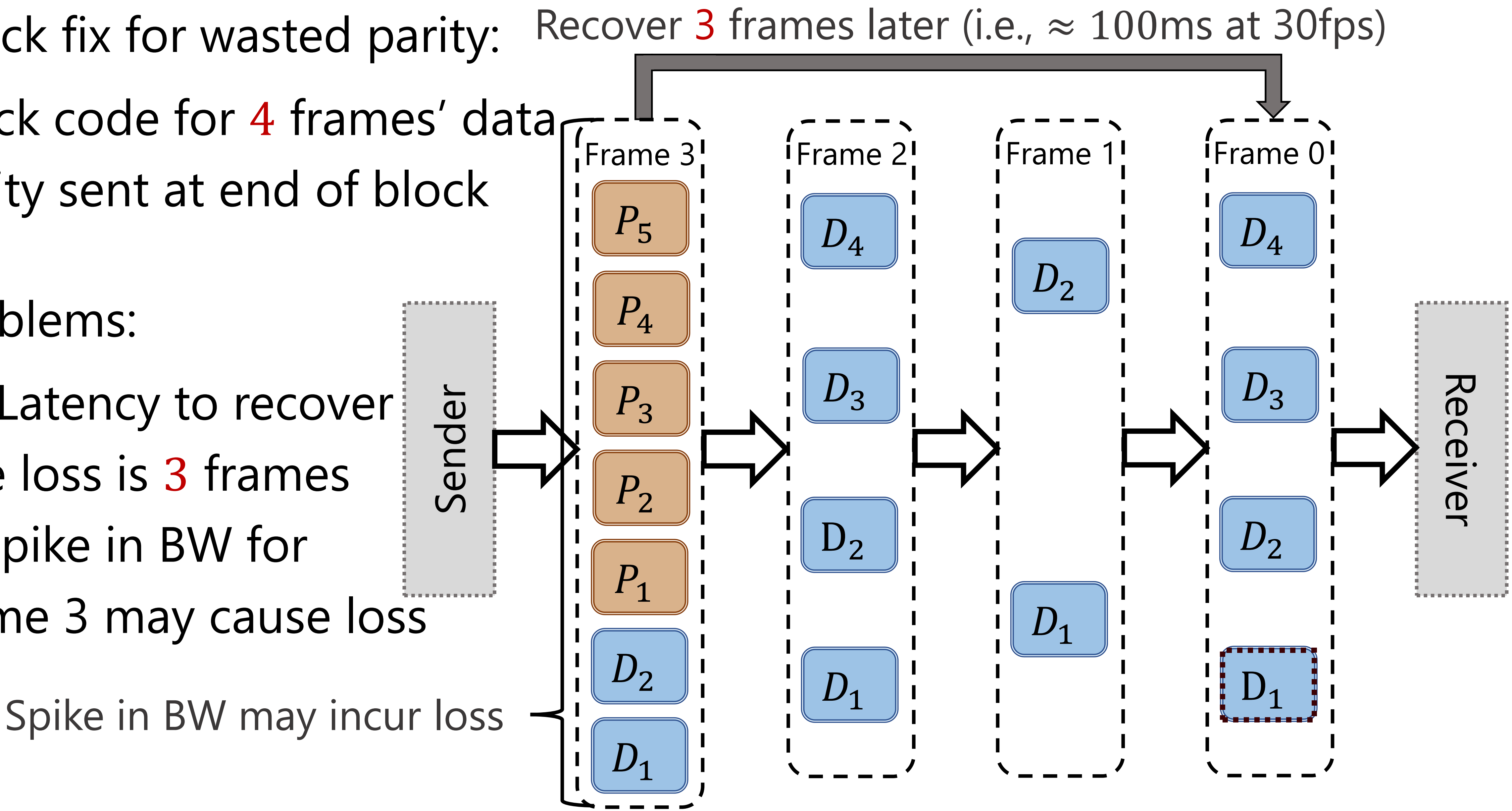
Quick fix for wasted parity:

Block code for 4 frames' data

Parity sent at end of block

Problems:

1. Latency to recover one loss is 3 frames
2. Spike in BW for frame 3 may cause loss

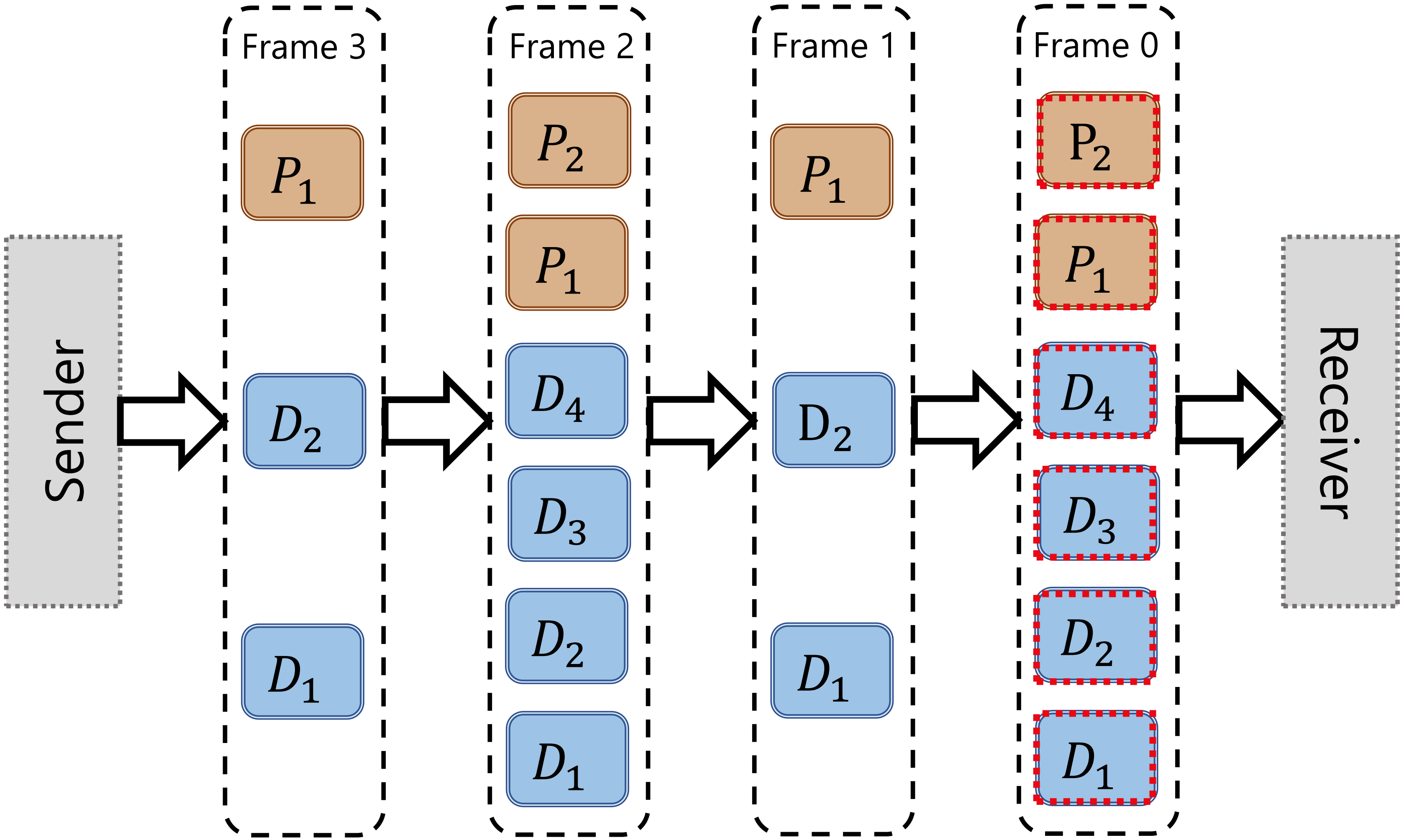


Streaming codes: bandwidth-efficient loss recovery

- Problem: RS codes sub-optimal for live communication: BW and latency
  - Block codes over 2 or 3 frames trades off these metrics
  - Our goal: fast recovery for one loss without wasting parity

- Streaming codes designed for following live-communication model
  - Latency: recover each frame within τ extra frames

Latency in # of frames to reflect end-to-end latency



Latency in # of frames to reflect end-to-end latency

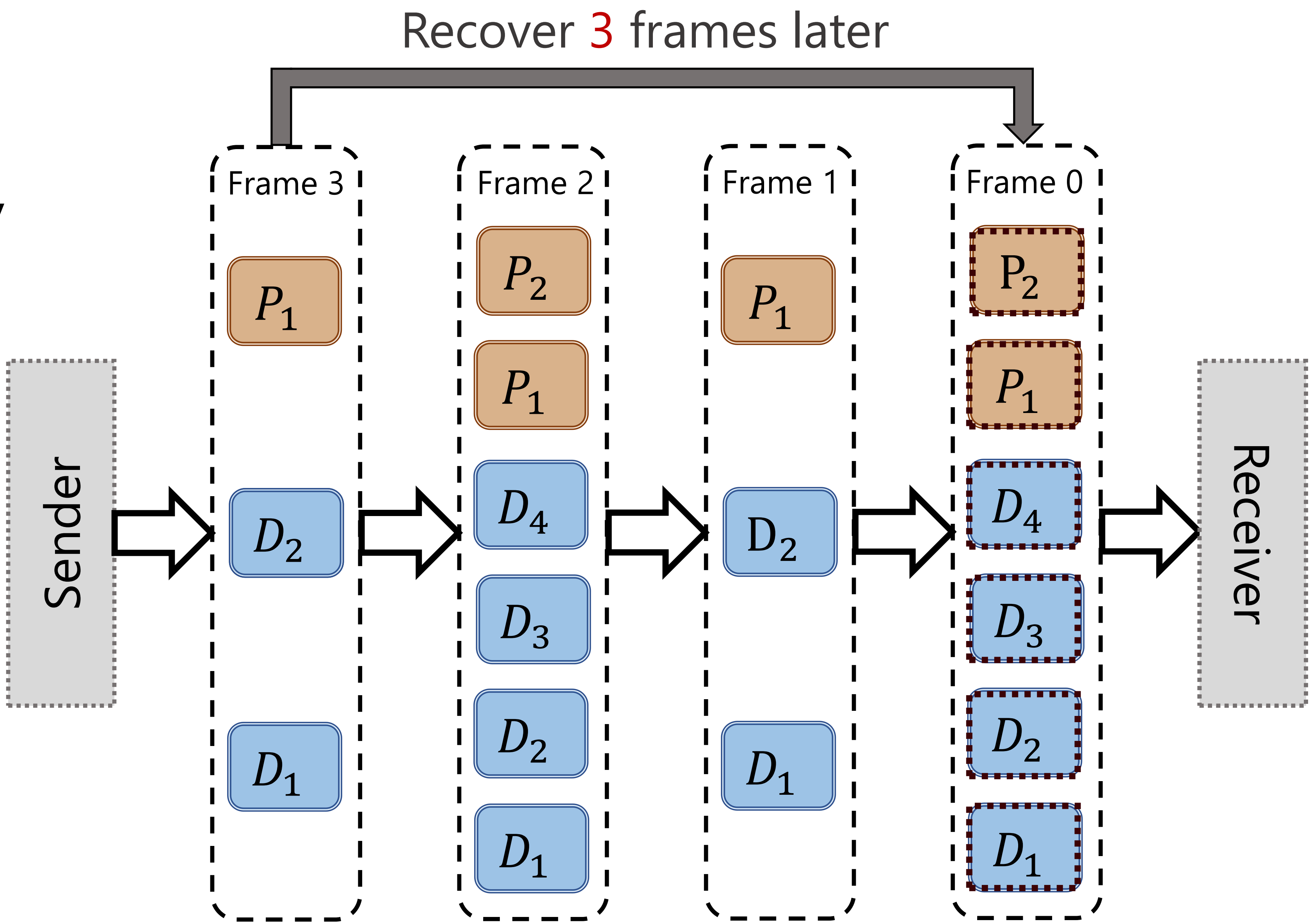
Suppose the call has

- 30 fps
- 50ms one-way delay

End-to-end latency:

$\approx 3 \cdot 33.3 + 50$

$= 150ms$

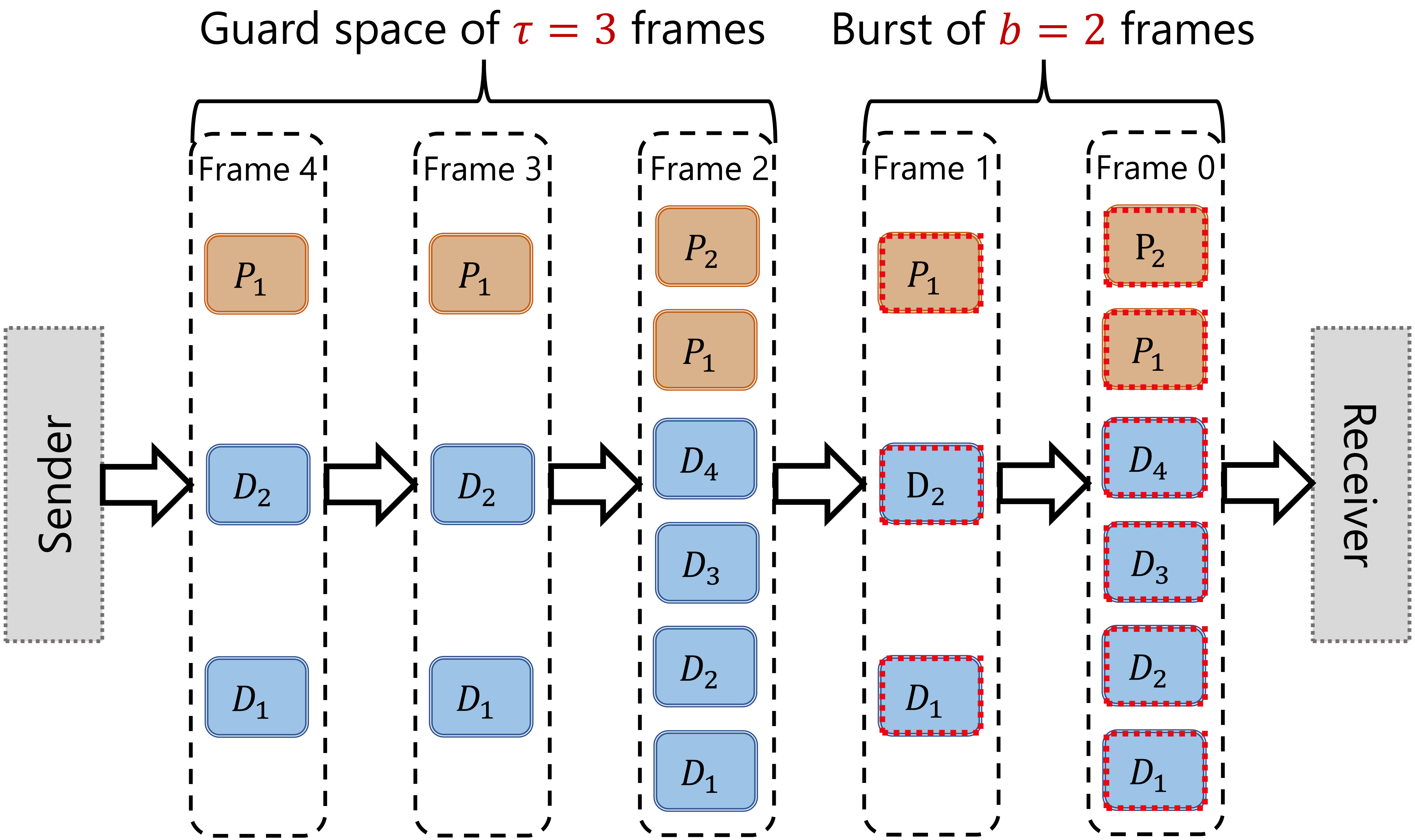


Streaming codes: bandwidth-efficient loss recovery

- Problem: RS codes sub-optimal for live communication: BW and latency
  - Block codes over 2 or 3 frames trades off these metrics
  - Our goal: fast recovery for one loss without wasting parity

- Streaming codes designed for following live-communication model
  - Latency: recover each frame within τ extra frames
  - Burst: at most b consecutive lossy frames, then
  - Guard space: at least τ consecutive frames with no losses

Loss model of bursts followed by guard spaces



Streaming codes: bandwidth-efficient loss recovery

- Problem: RS codes sub-optimal for live communication: BW and latency
  - Block codes over 2 or 3 frames trades off these metrics
  - Our goal: fast recovery for one loss without wasting parity

- Streaming codes designed for following live-communication model
  - Latency: recover each frame within τ extra frames
  - Burst: at most b consecutive lossy frames, then
  - Guard space: at least τ consecutive frames with no losses

- Streaming codes work by
  - Sending parity packets within each frame and computed over multiple frames to
  - **Sequentially recover** lost frames of burst each at their deadlines
  - As opposed to *simultaneously recovering* all lost packets (e.g., of a block)

Streaming codes: challenges

- Suitability over real-world losses unknown

- Gaps between theory and practice, including
  - Drop all packets of a frame
  - Never loss in guard space

- Not yet assessed for impact on the QoE

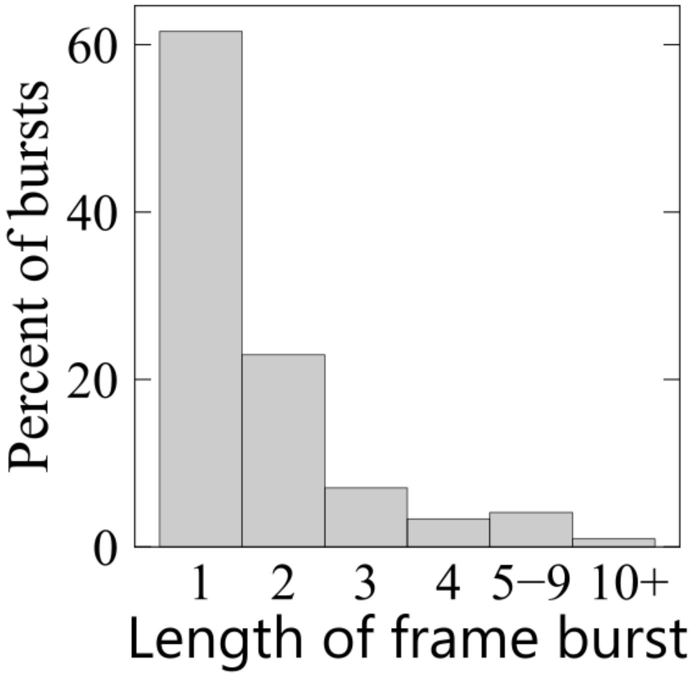
Analysis of traces from Teams video calls

- ≈9700 traces from two-week random sample Microsoft Teams 1:1 calls

- Burst losses are characterized by
  - Number of consecutive frames with at least one lost packet
  - Fraction of packets lost in a burst over multiple frames

- Guard spaces need only exceed τ to enable loss recovery
  - Set $\tau = 3$ to cap the latency at ≈ 150 ms at 30 fps with a 50 ms one-way delay

Losses suited to streaming codes… if address gaps



- Many burst losses of 2 – 4 frames determine parity needed
- No clear worst-case value, b

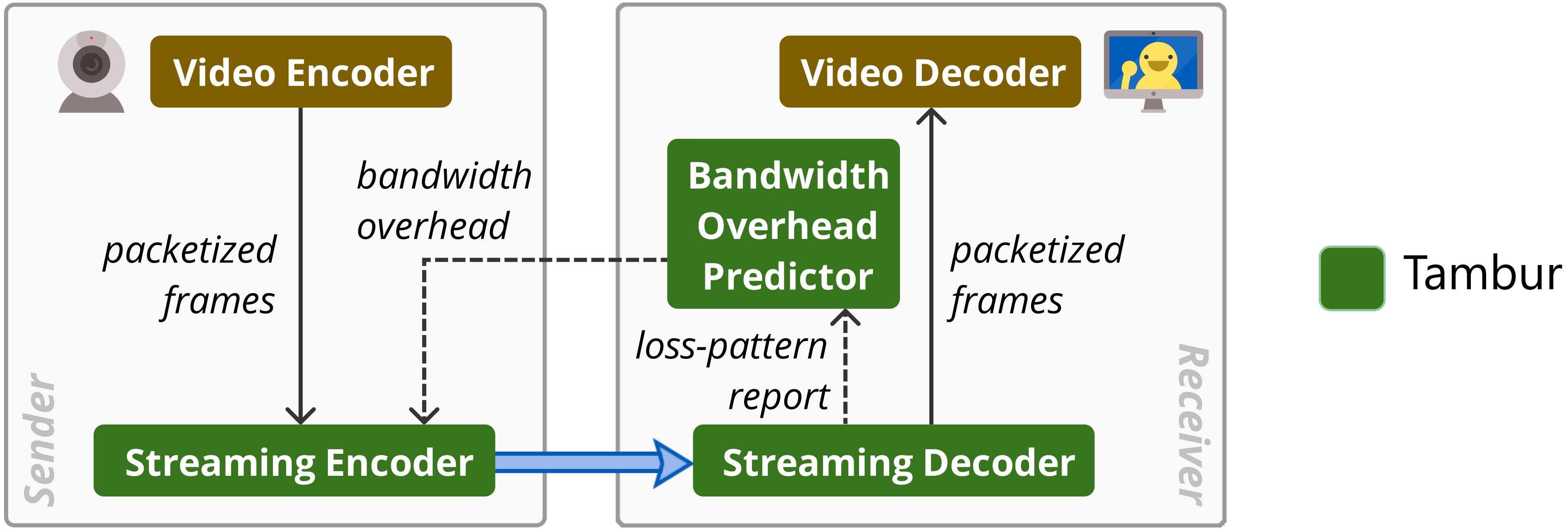
Fraction of packets lost in multi-frame burst

- Varies from just over 0 to 1
- Model of all packets lost is pessimistic

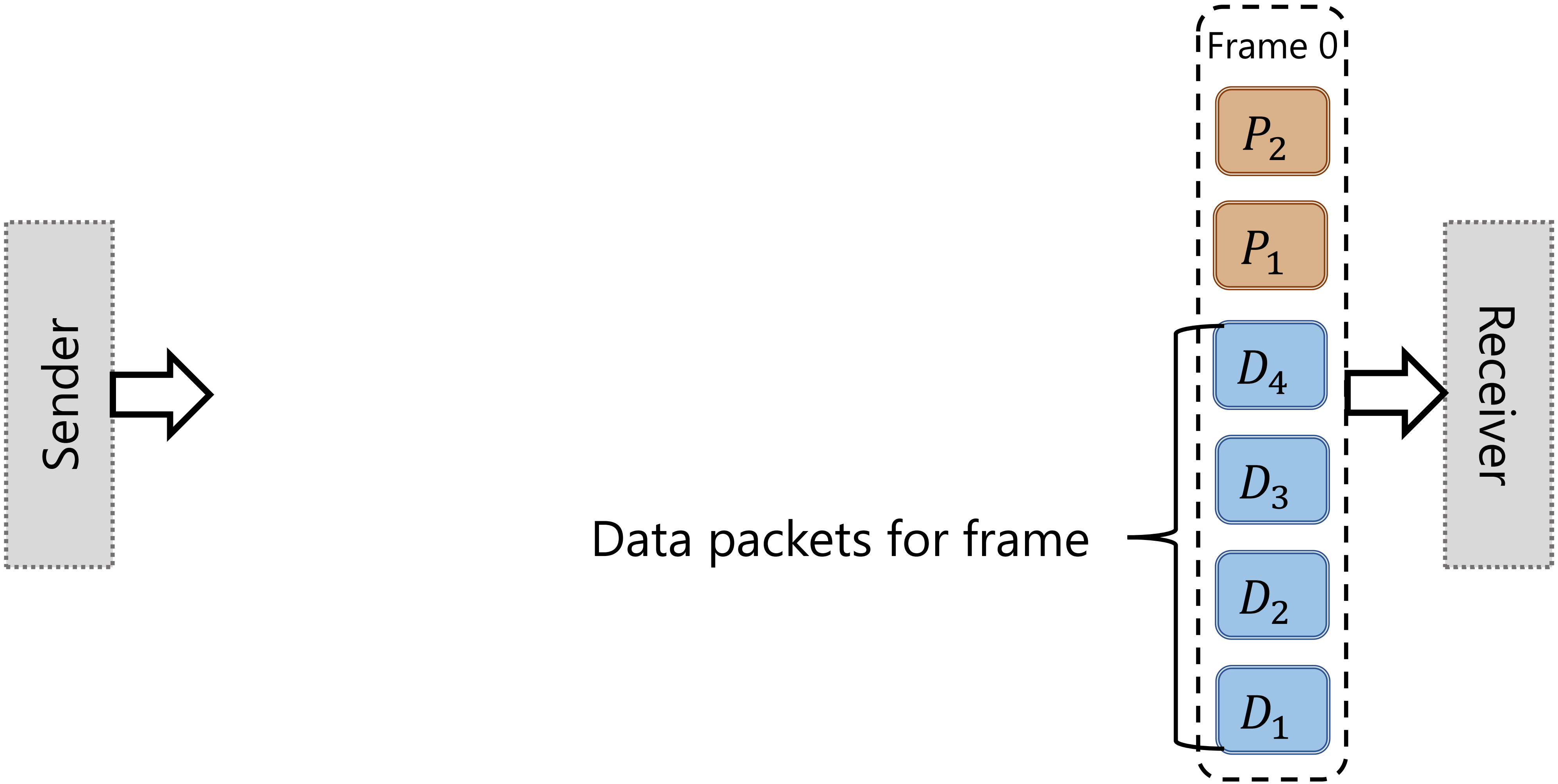
Guard spaces are common, but sometimes losses occur in guard space

Tambur: a new communication paradigm for VC

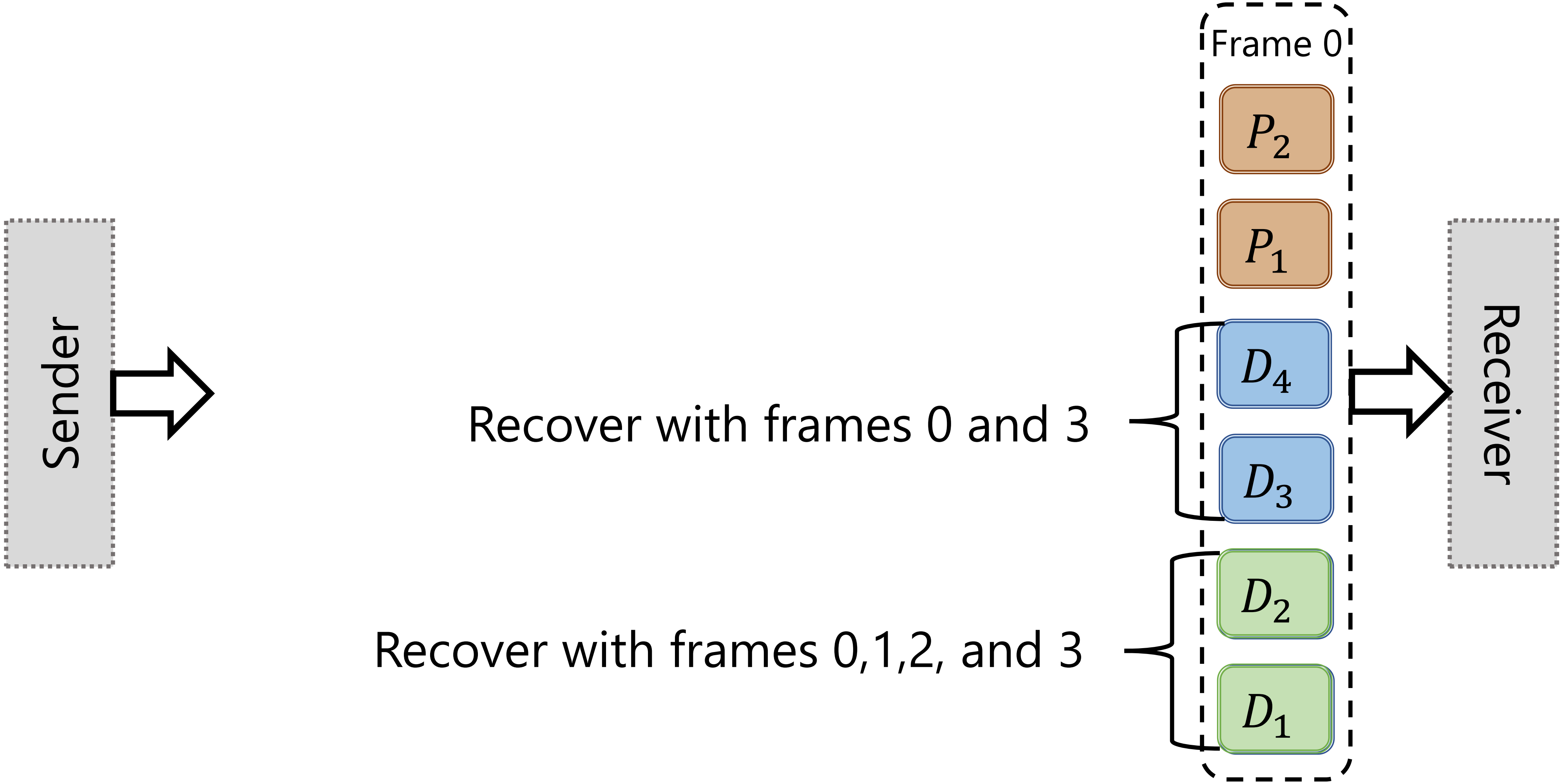
- Design Tambur by combining
  - New streaming codes (shown shortly)
  - Lightweight binary classifier instead of b and τ set parity size (see paper)
    - Match existing system's parity size or reduce it by 50%



Tambur recovers with bounded latency



Tambur recovers with bounded latency

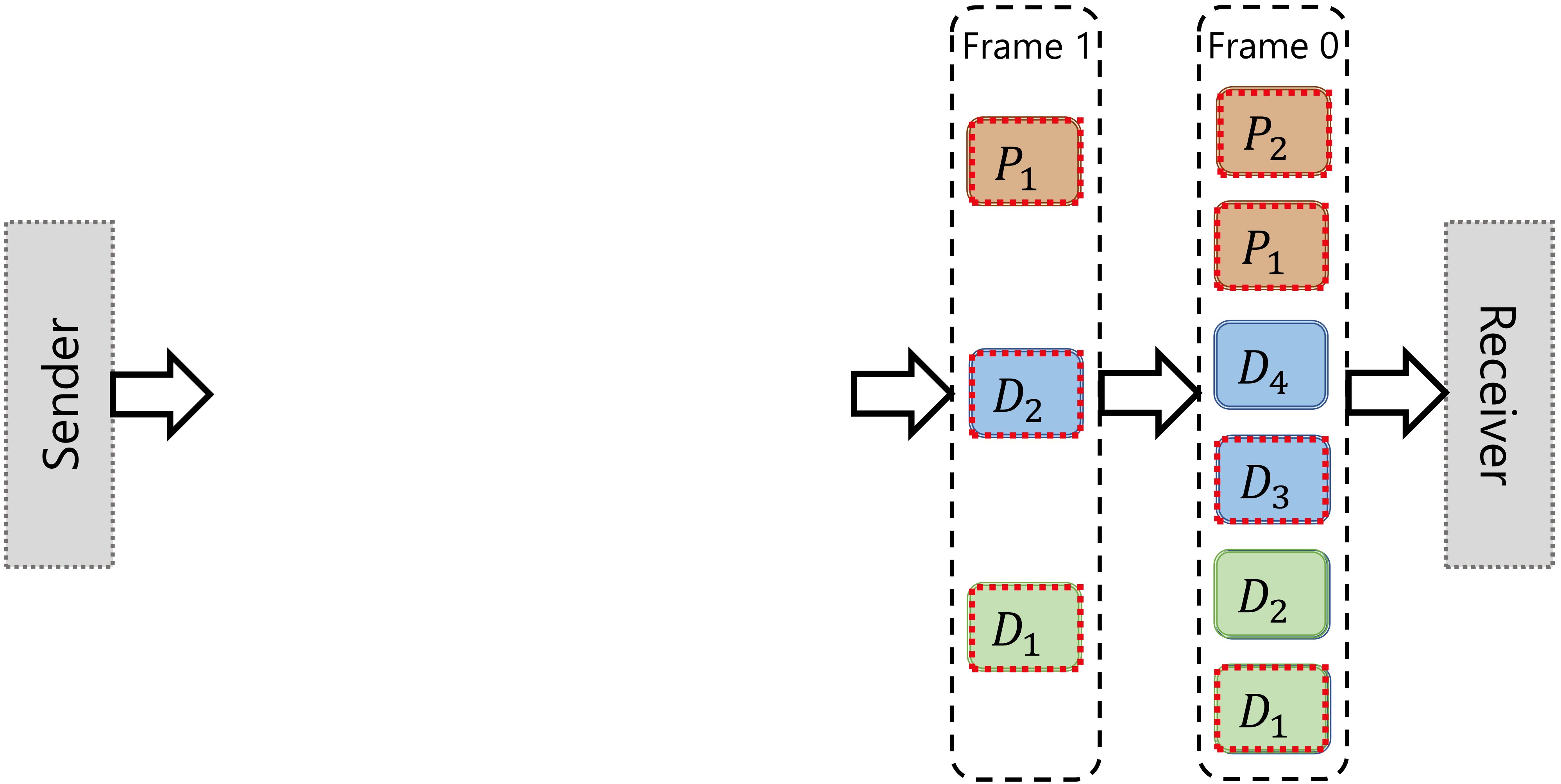


Recover with frames 0 and 3

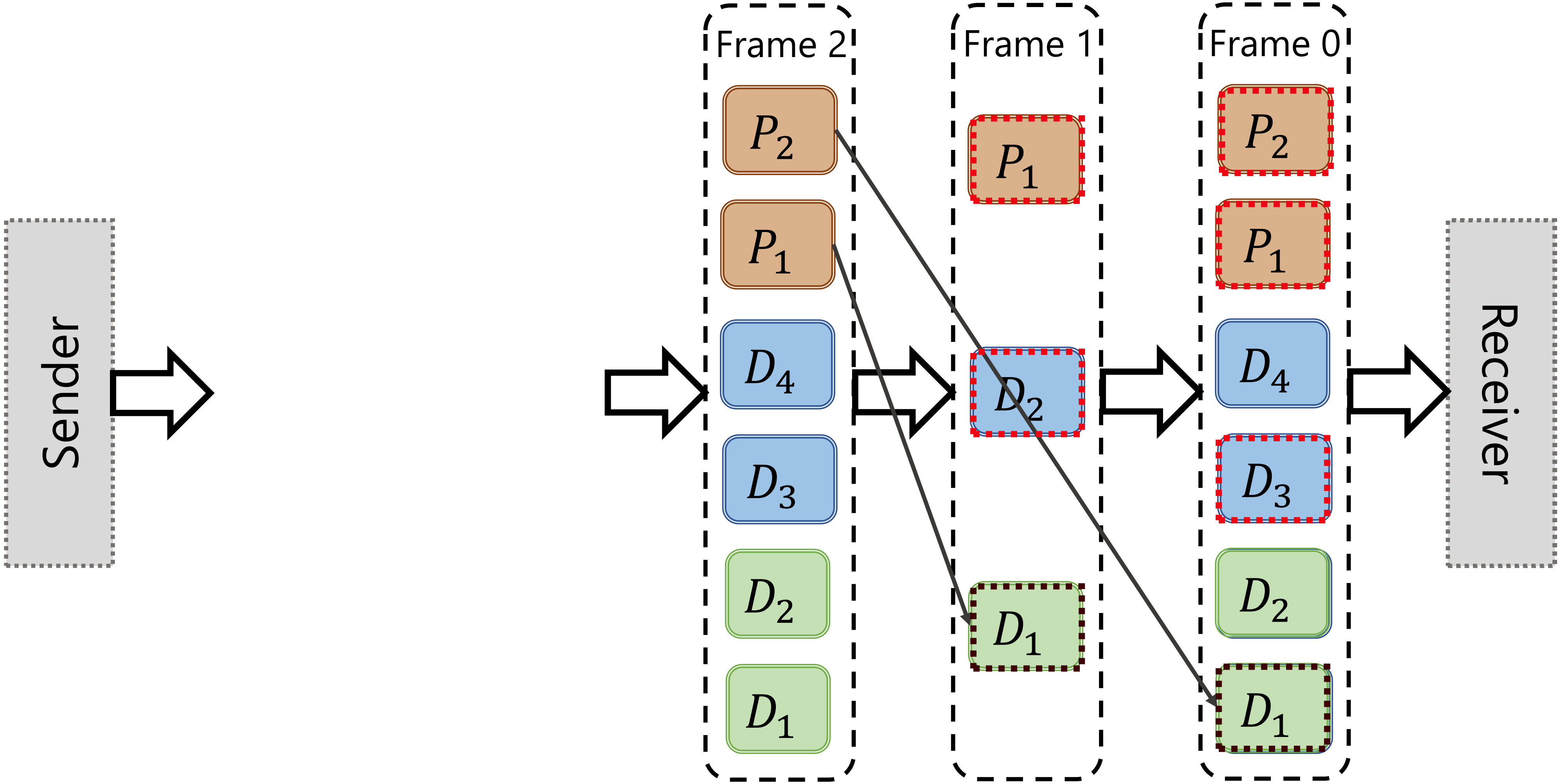
Recover with frames 0,1,2, and 3

Tambur recovers with bounded latency

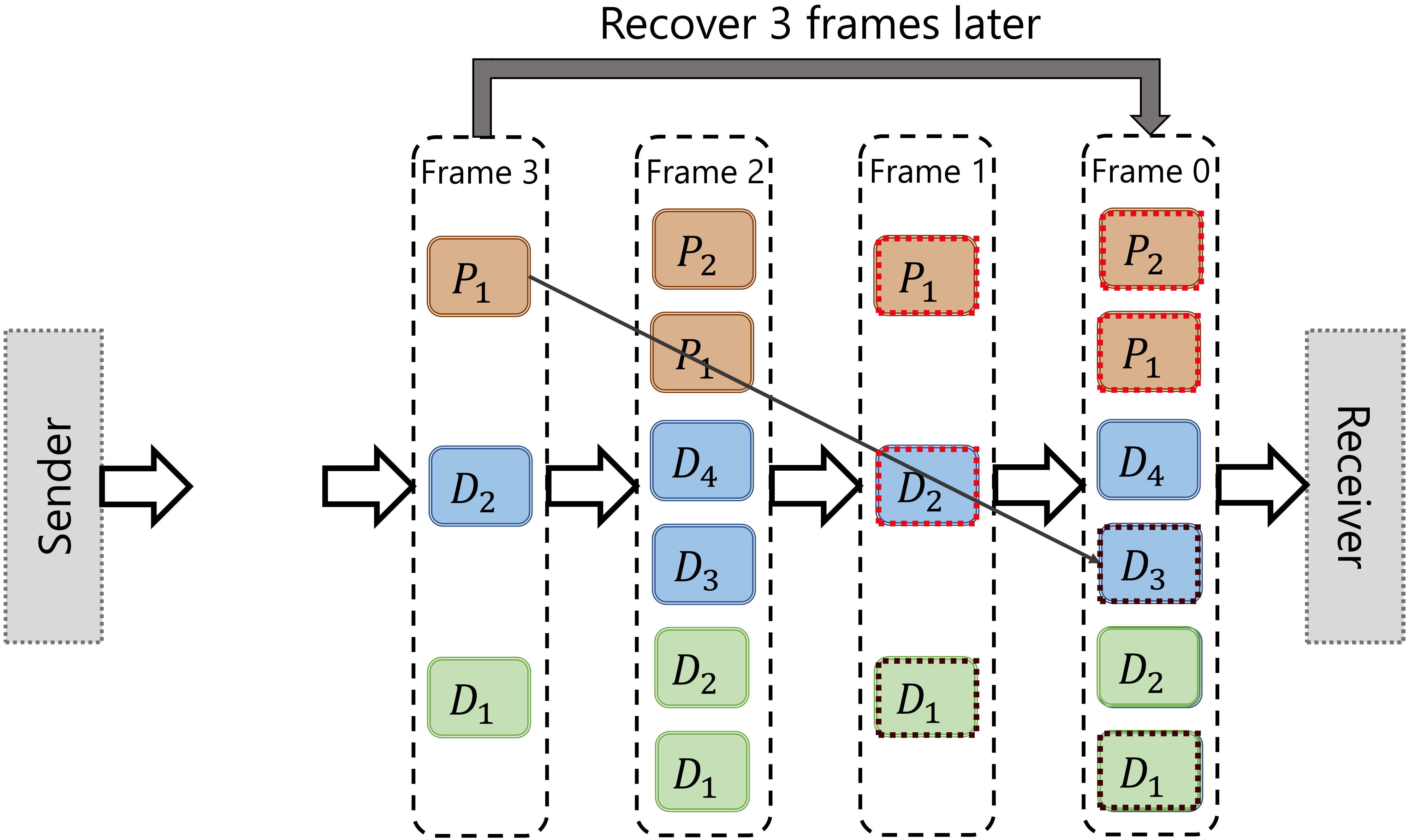
Tambur recovers with bounded latency



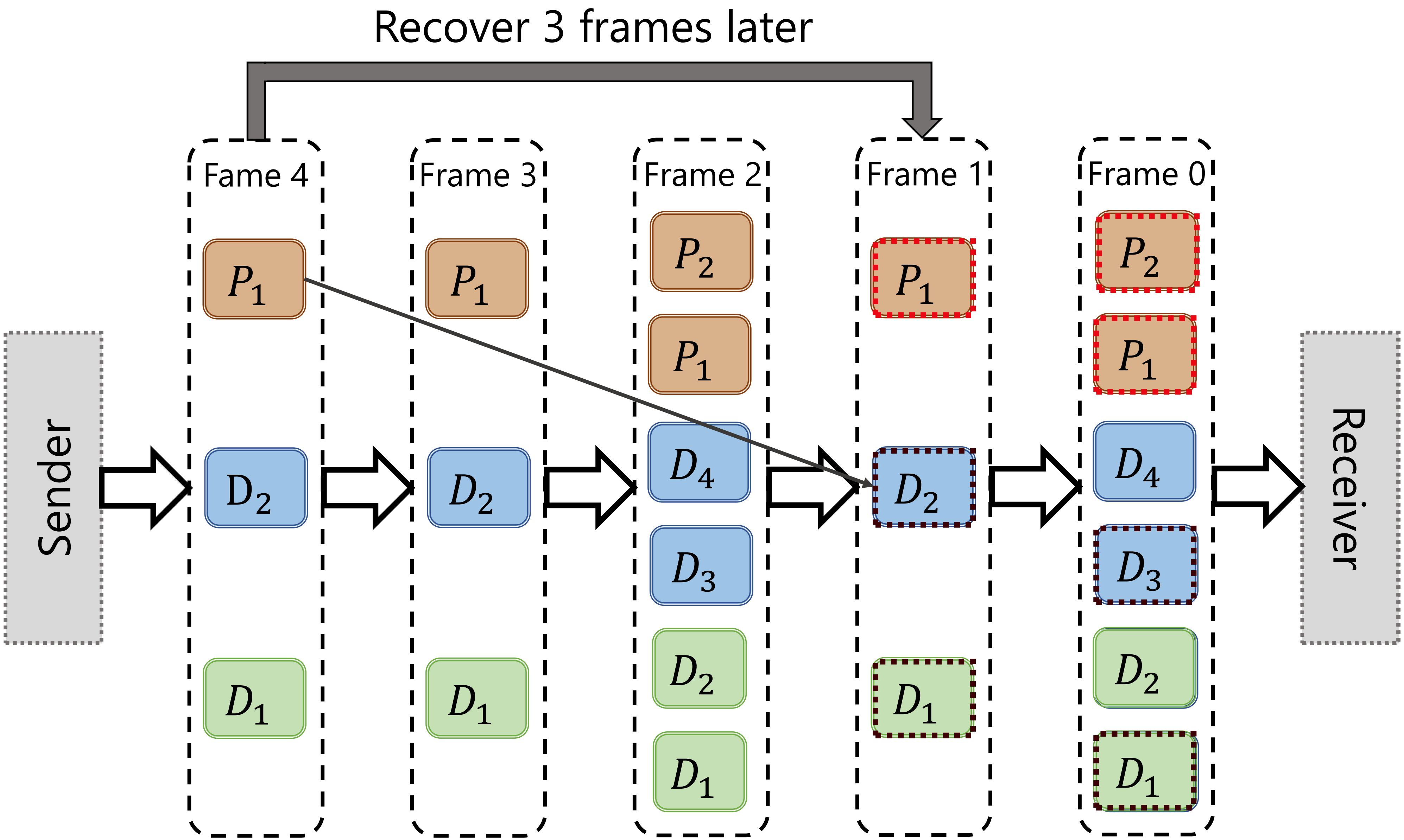
Tambur recovers with bounded latency



Tambur recovers with bounded latency

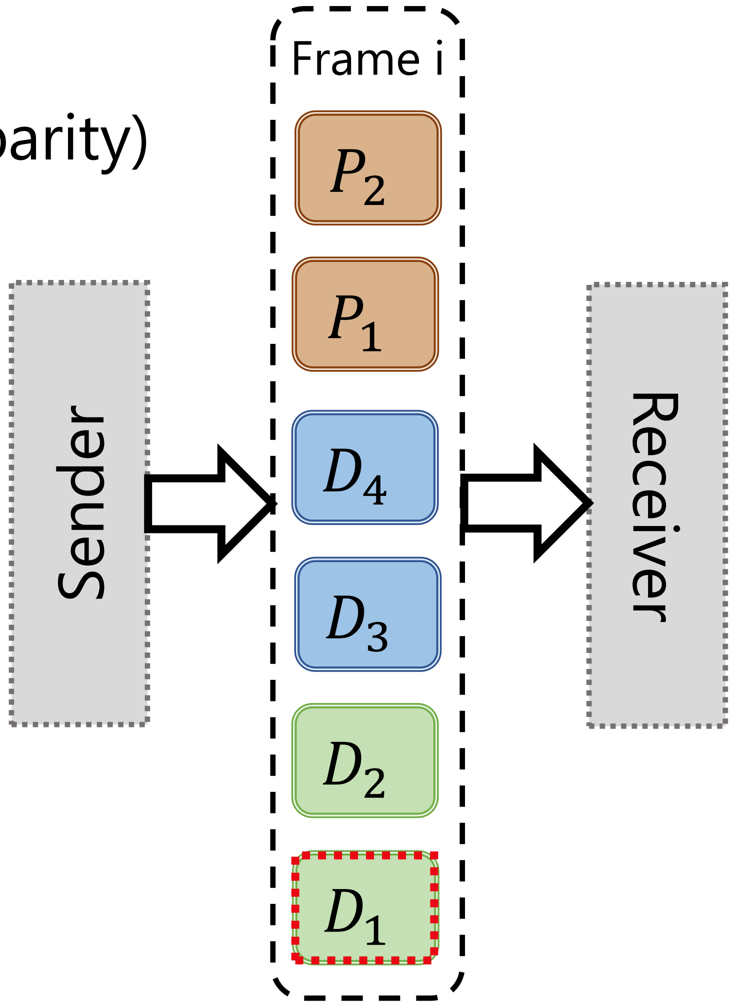


Tambur recovers with bounded latency



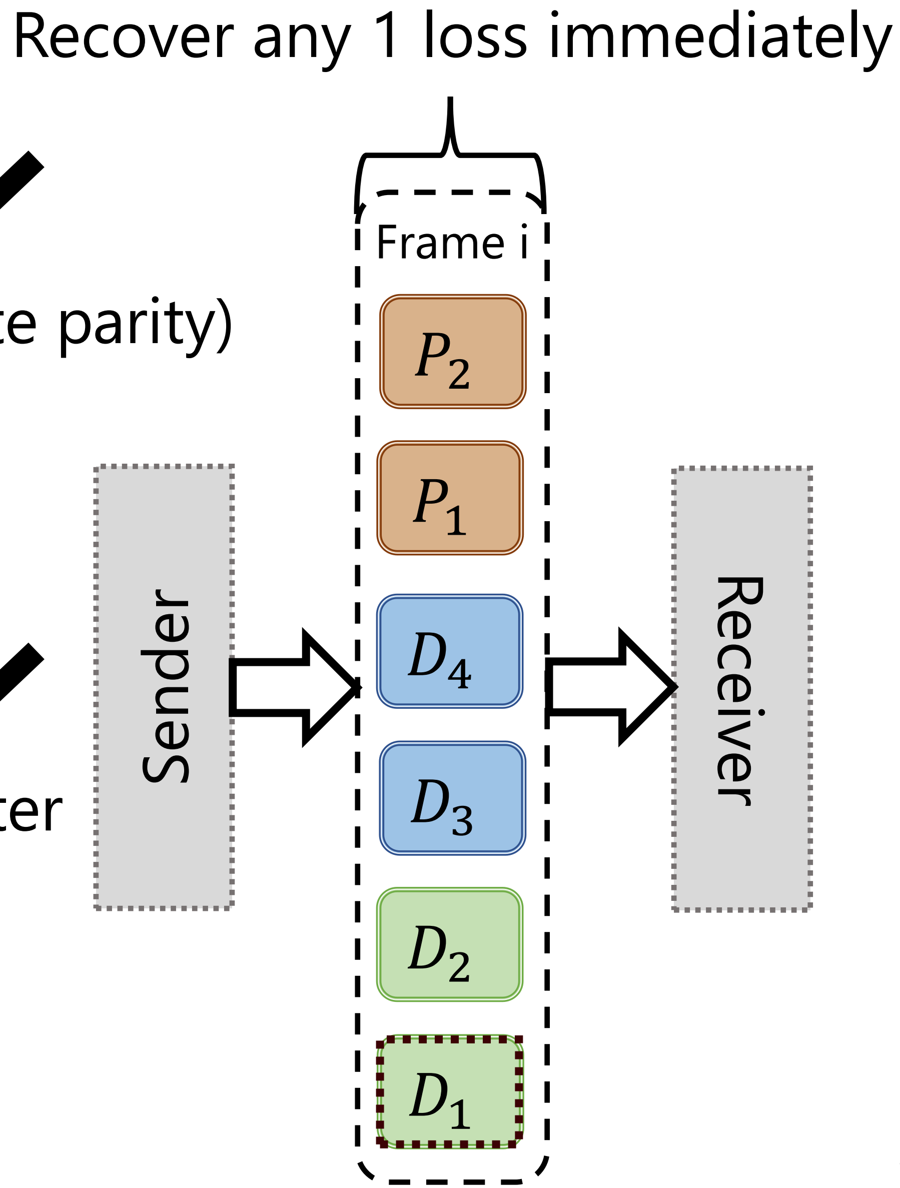
Tambur has minimal latency to recover rare losses

- Before: worst-case loss recovery
  - Leverage parity in guard space for recovery ✓
  - Unlike RS within each frame not recovering (waste parity)
- Now: address occasional losses
  - Loss recovery should have minimal latency
  - Unlike RS across 4 frames recovering 3 frames later



Tambur has minimal latency to recover rare losses

- Before: worst-case loss recovery
  - Leverage parity in guard space for recovery ✓
  - Unlike RS within each frame not recovering (waste parity)
- Now: address occasional losses
  - Loss recovery should have minimal latency ✓
  - Unlike RS across 4 frames recovering 3 frames later

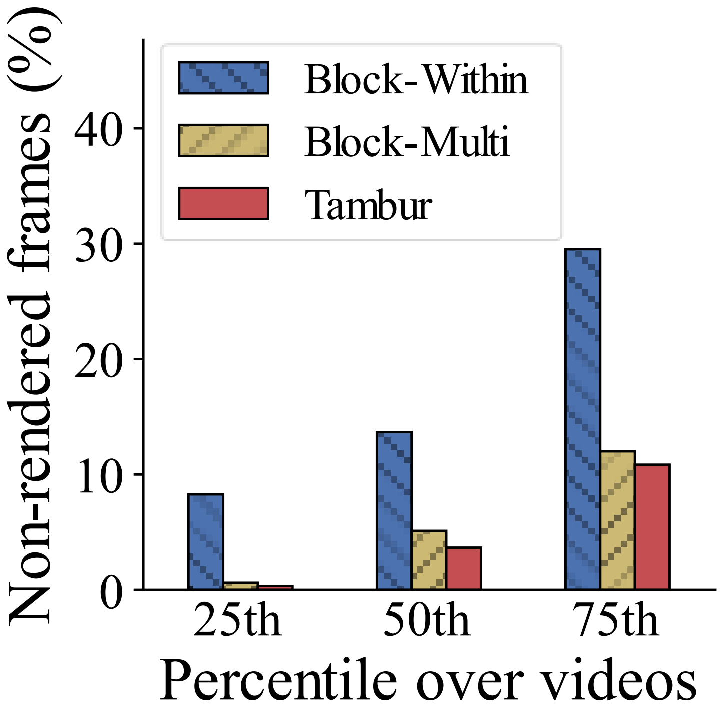


Online evaluation methodology

- Implement Tambur in C++ (https://github.com/Thesys-lab/tambur/)
- Integrate with Ringmaster (https://github.com/microsoft/ringmaster/)
  - Ringmaster is a VC platform for emulating 1:1 calls
- Compare to two standard baselines with **slightly extra parity**
  - Block-within—RS within each frame
  - Block-multi—RS across 4 frames
- Evaluate over 80 10-minute videos of varying bitrates
- Over Mahimahi and emulated networks (details in paper)

Tambur renders more frames at lower latency

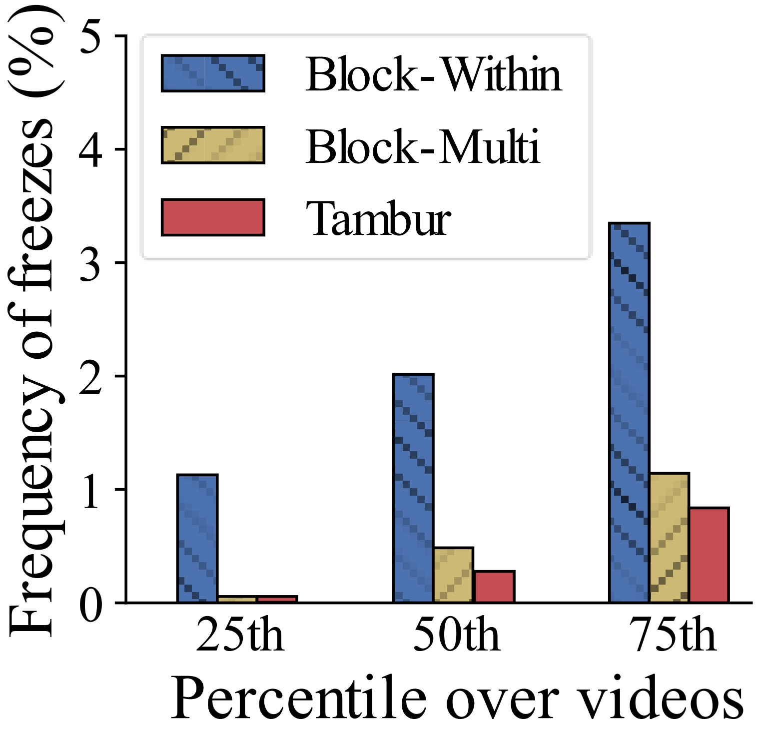
- Reasons for degrading QoE: not rendering frames or latency



- Fails to render **73% fewer frames** than Block-Within at median
- Fails to render **28% fewer frames** than Block-Multi at median
- 6.5 ms higher median latency than Block-within
- 18.9 ms lower median latency than Block-Multi

Tambur mitigates freeze frequency

- Freeze frequency crucial to mean opinion score (i.e., QoE)



- Freeze frequency **reduced by 78%** over Block-Within at median
- Freeze frequency **reduced by 26%** over Block-Multi at median

Takeaway: Tambur improves several key metrics of the QoE

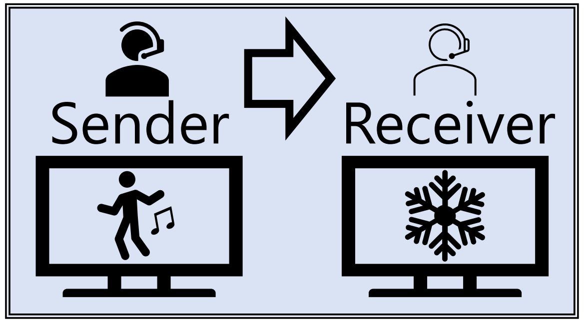
New interdisciplinary loss recovery VC

- **Challenge**: conventional loss-recovery sub-optimal videoconferencing

- **Approach:** build Tambur by designing new streaming codes + using ML

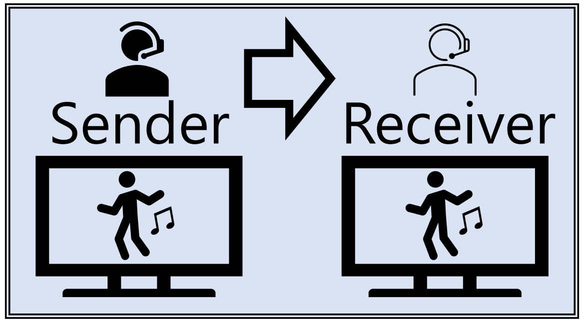
- **Outcome:**

Before



Eliminate 26% of freezes and 28% of rendering failures

After



This work was funded in part by an NSF grant (CCF1910813).