

# How valuable is noisy or limited feedback?

Anant Sahai

based in part on joint work with:

Stark Draper    Tunc Simsek

Wireless Foundations

Department of Electrical Engineering and Computer Sciences

University of California at Berkeley

**Major Support from NSF ITR**

EPFL Summer Research Institute: Jul 19th, 2006

# Information theory tells us:

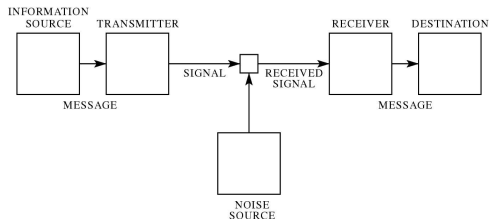


Fig. 1— Schematic diagram of a general communication system.

- Delay is the most basic price of reliability

# Information theory tells us:

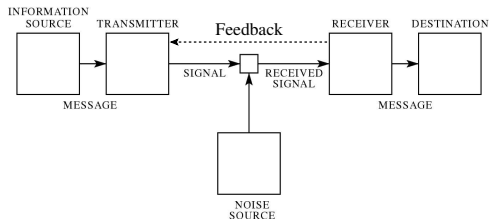


Fig. 1— Schematic diagram of a general communication system.

- Delay is the most basic price of reliability
- What if there is feedback?

# Information theory tells us:

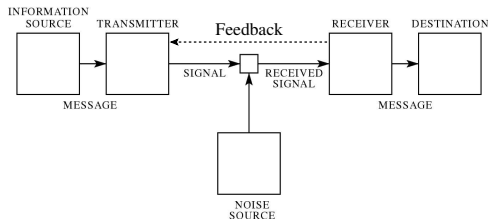


Fig. 1— Schematic diagram of a general communication system.

- Delay is the most basic price of reliability
- What if there is feedback?
  - ▶ If noise is memoryless, capacity is unchanged.

# Information theory tells us:

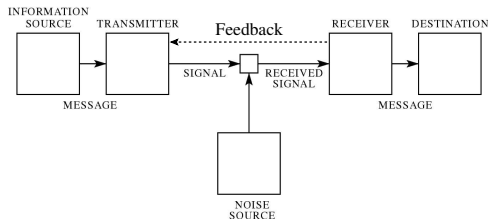


Fig. 1— Schematic diagram of a general communication system.

- Delay is the most basic price of reliability
- What if there is feedback?
  - ▶ If noise is memoryless, capacity is unchanged.
  - ▶ What about delay and probability of error?

# Information theory tells us:

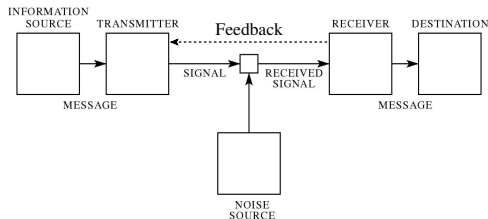


Fig. 1— Schematic diagram of a general communication system.

- Delay is the most basic price of reliability
- What if there is feedback?
  - ▶ If noise is memoryless, capacity is unchanged.
  - ▶ What about delay and probability of error?
  - ▶ How should we use it?

# Information theory tells us:

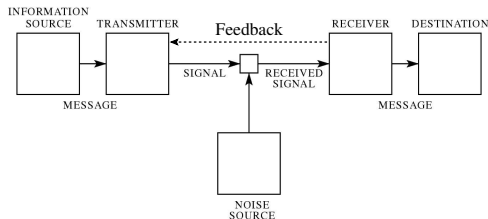


Fig. 1— Schematic diagram of a general communication system.

- Delay is the most basic price of reliability
- What if there is feedback?
  - ▶ If noise is memoryless, capacity is unchanged.
  - ▶ What about delay and probability of error?
  - ▶ How should we use it?
  - ▶ How much feedback do we need?

# Outline

- 1 **Background review**
- 2 Problem 1: BEC: bitwise hard deadlines with limited feedback
- 3 Problem 2: BSC: bitwise soft deadlines with noisy feedback

## Review: Fixed blocks, Hard deadlines

- One of  $2^{nR}$  equally likely messages to send
- Must decode after  $n$  channel uses

## Review: Fixed blocks, Hard deadlines

- One of  $2^{nR}$  equally likely messages to send
- Must decode after  $n$  channel uses
- Study the probability of block error in the limit of large  $n$
- Block error exponents:  $P_e \propto \exp(-nE(R))$

## Review: Fixed blocks, Hard deadlines

- One of  $2^{nR}$  equally likely messages to send
- Must decode after  $n$  channel uses
- Study the probability of block error in the limit of large  $n$
- Block error exponents:  $P_e \propto \exp(-nE(R))$ 
  - ▶ Generic channels: (Haroutunian)

$$E^+(R) = \inf_{G:C(G)<R} \max_{\vec{q}} D(G||P|\vec{q})$$

- ▶ Holds with or without feedback

# Review: Fixed blocks, Hard deadlines

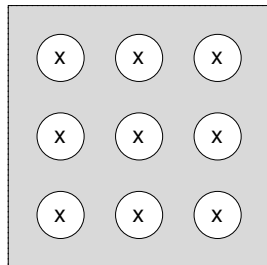
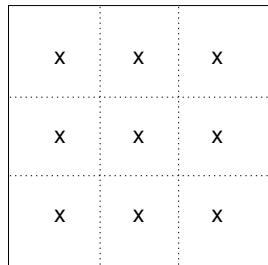
- One of  $2^{nR}$  equally likely messages to send
- Must decode after  $n$  channel uses
- Study the probability of block error in the limit of large  $n$
- Block error exponents:  $P_e \propto \exp(-nE(R))$ 
  - ▶ “Sphere-packing” bound:

$$\begin{aligned} E_{sp}(R) &= \max_{\vec{q}} \min_{G: I(\vec{q}, G) \leq R} D(G \| P | \vec{q}) \\ &= \sup_{\rho \geq 0} E_0(\rho) - \rho R \\ E_0(\rho) &= \max_{\vec{q}} -\ln \sum_z \left[ \sum_y q_y p_{y,z}^{\frac{1}{1+\rho}} \right]^{(1+\rho)} \end{aligned}$$

- ▶ Holds without feedback
- ▶ Same as  $E^+(R)$  for symmetric channels! (Feedback useless)

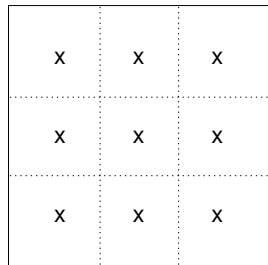
# Review: Fixed blocks, Soft deadlines: Forney-68

## Review: Fixed blocks, Soft deadlines: Forney-68

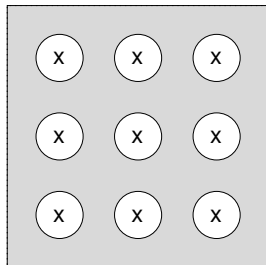


- Hard deadline  
regions cover space
- Feedback is  
pointless

## Review: Fixed blocks, Soft deadlines: Forney-68

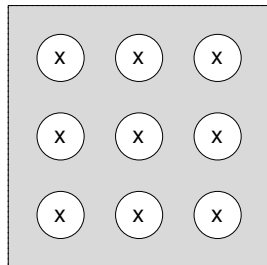
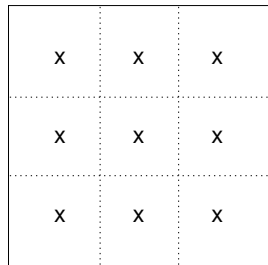


- Hard deadline regions cover space
- Feedback is pointless



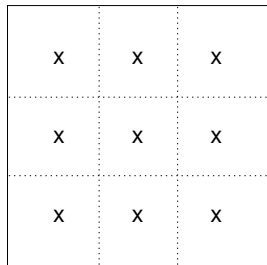
- Refuse to decode when ambiguous

## Review: Fixed blocks, Soft deadlines: Forney-68

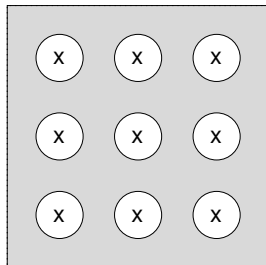


- Hard deadline regions cover space
- Feedback is pointless
- Refuse to decode when ambiguous
- Keep erasures rare
- Minimize undetected errors

## Review: Fixed blocks, Soft deadlines: Forney-68



- Hard deadline regions cover space
- Feedback is pointless



- Refuse to decode when ambiguous
- Keep erasures rare
- Minimize undetected errors
- 1 bit feedback can request retransmissions

## Review: Fixed blocks, Soft deadlines: Burnashev-76

- Is more feedback helpful?
- Burnashev said yes:

# Review: Fixed blocks, Soft deadlines: Burnashev-76

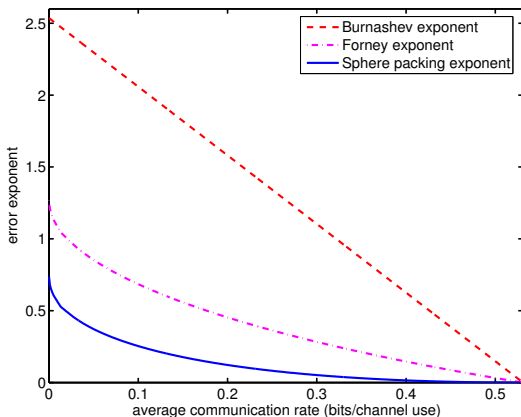
- Is more feedback helpful?
- Burnashev said yes:
  - ▶ Considered expected stopping time and used Martingale arguments.

# Review: Fixed blocks, Soft deadlines: Burnashev-76

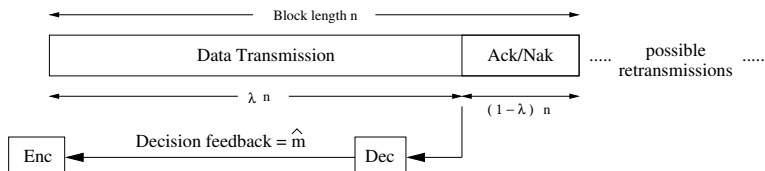
- Is more feedback helpful?
- Burnashev said yes:
  - ▶ Considered expected stopping time and used Martingale arguments.
  - ▶ Showed  $C_1(1 - \frac{R}{C})$  was a bound where  $C_1 = \max_{i,j} D(p_i || p_j)$

# Review: Fixed blocks, Soft deadlines: Burnashev-76

- Is more feedback helpful?
- Burnashev said yes:
  - ▶ Considered expected stopping time and used Martingale arguments.
  - ▶ Showed  $C_1(1 - \frac{R}{C})$  was a bound where  $C_1 = \max_{i,j} D(p_i || p_j)$

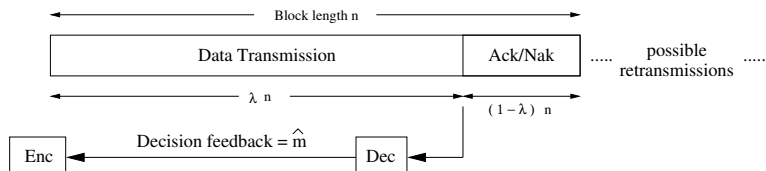


# Review: Yamamoto-Itoh-79 strategy attains the Burnashev bound



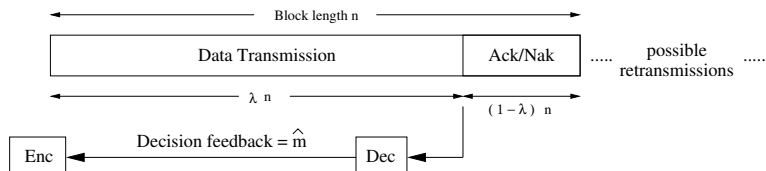
- **Data Transmission:**  $\lambda n$  channel uses for block code at  $R \simeq C$

# Review: Yamamoto-Itoh-79 strategy attains the Burnashev bound



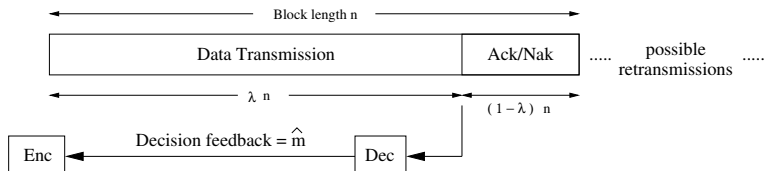
- **Data Transmission:**  $\lambda n$  channel uses for block code at  $R \simeq C$
- **Decision Feedback:**  $\hat{m}$  sent back

# Review: Yamamoto-Itoh-79 strategy attains the Burnashev bound



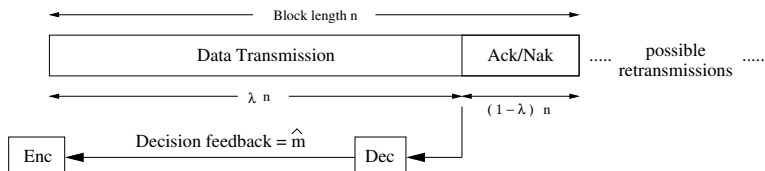
- **Data Transmission:**  $\lambda n$  channel uses for block code at  $R \simeq C$
- **Decision Feedback:**  $\hat{m}$  sent back
- **Confirm/Deny:**  $(1 - \lambda)n$  channel uses to ACK or NAK

# Review: Yamamoto-Itoh-79 strategy attains the Burnashev bound



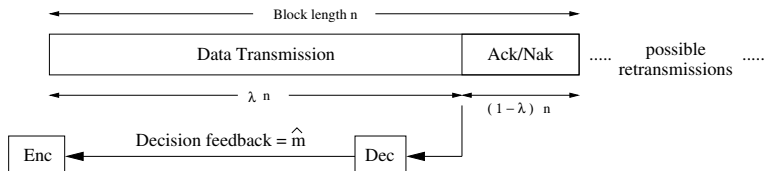
- **Data Transmission:**  $\lambda n$  channel uses for block code at  $R \simeq C$
- **Decision Feedback:**  $\hat{m}$  sent back
- **Confirm/Deny:**  $(1 - \lambda)n$  channel uses to ACK or NAK
- If confirmed, decode to  $\hat{m}$  otherwise erase.

# Review: Yamamoto-Itoh-79 strategy attains the Burnashev bound



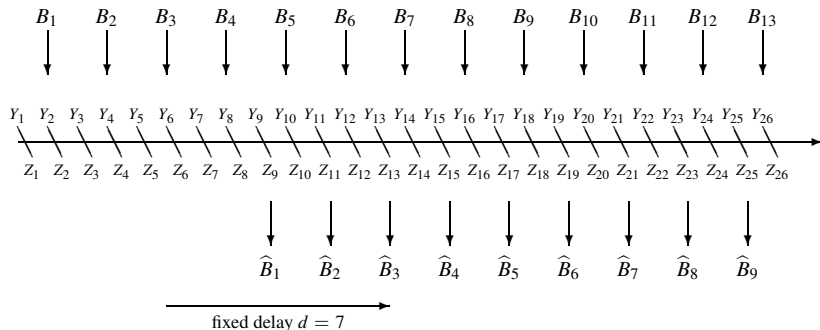
- **Data Transmission:**  $\lambda n$  channel uses for block code at  $R \simeq C$
- **Decision Feedback:**  $\hat{m}$  sent back
- **Confirm/Deny:**  $(1 - \lambda)n$  channel uses to ACK or NAK
- If confirmed, decode to  $\hat{m}$  otherwise erase.
- $\Pr[\text{err}] = \Pr[\text{NAK} \rightarrow \text{ACK}] = 2^{-(1-\lambda)nC_1} \simeq 2^{-nC_1(1-\frac{R}{C})}$

# Review: Yamamoto-Itoh-79 strategy attains the Burnashev bound



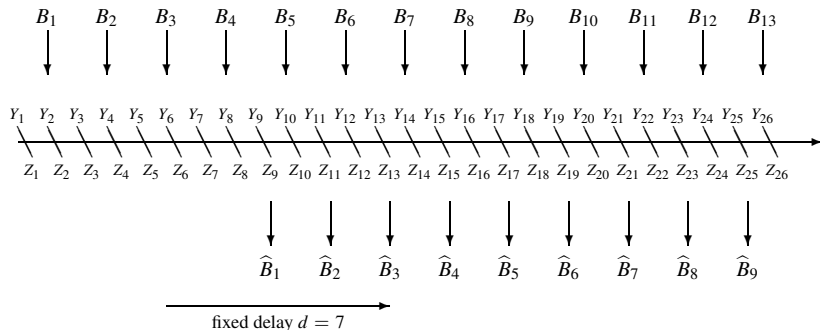
- **Data Transmission:**  $\lambda n$  channel uses for block code at  $R \simeq C$
- **Decision Feedback:**  $\hat{m}$  sent back
- **Confirm/Deny:**  $(1-\lambda)n$  channel uses to ACK or NAK
- If confirmed, decode to  $\hat{m}$  otherwise erase.
- $\Pr[\text{err}] = \Pr[\text{NAK} \rightarrow \text{ACK}] = 2^{-(1-\lambda)nC_1} \simeq 2^{-nC_1(1-\frac{R}{C})}$
- **Moral: Collective reward/punishment is good for reliability**

# Communication with a latency requirement



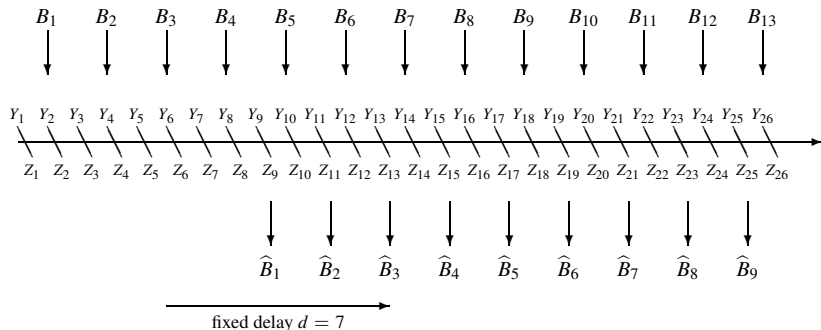
- Bits/packets arrive steadily at rate  $R$  bits per channel use.
- End-to-end latency requirement of  $d$

# Communication with a latency requirement



- Bits/packets arrive steadily at rate  $R$  bits per channel use.
- End-to-end latency requirement of  $d$ 
  - ▶ Hard: Declared or undetected errors are equally bad.

# Communication with a latency requirement

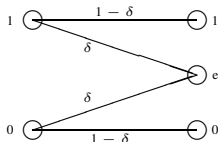


- Bits/packets arrive steadily at rate  $R$  bits per channel use.
- End-to-end latency requirement of  $d$ 
  - ▶ Hard: Declared or undetected errors are equally bad.
  - ▶ Soft: Undeclared errors are very bad, but declared errors should be infrequent.

# Outline

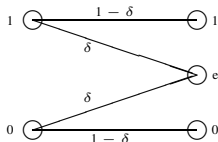
- 1 Background review
- 2 **Problem 1: BEC: bitwise hard deadlines with limited feedback**
  - ▶ Review of perfect feedback
  - ▶ Round-trip delay
  - ▶ Packetization and retransmission
  - ▶ Limited feedback and Hybrid ARQ
  - ▶ Low-rate: lists and disambiguation
- 3 Problem 2: BSC: bitwise soft deadlines with noisy feedback

# Erasure channels with perfect feedback

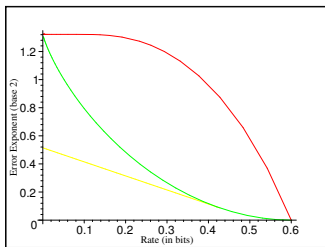


- Simple capacity  $1 - \delta$  bits per channel use
- With perfect non-delayed feedback, simple to achieve: retransmit until it gets through

# Erasure channels with perfect feedback



- Simple capacity  $1 - \delta$  bits per channel use
- With perfect non-delayed feedback, simple to achieve: retransmit until it gets through



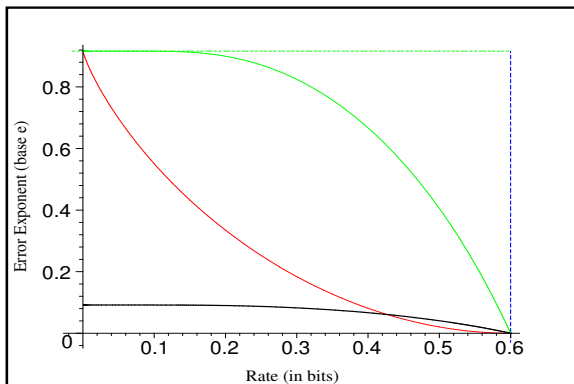
- Hard Deadline bounds
  - ▶ Without Feedback: Sphere-packing bound  $D(1 - R || \delta)$
  - ▶ With Feedback: Focusing bound  $(\frac{E_0(\rho)}{\rho}, E_0(\rho))$

## $k$ -delayed feedback packet erasure case

*First approach: treat as  $k$  parallel unit-delay channels*

## $k$ -delayed feedback packet erasure case

*First approach: treat as  $k$  parallel unit-delay channels*



Serious penalty to waiting  $k$  steps between retransmissions

# Packetization: first try

- Use block-code
  - ▶ Group bits into packets with length  $nR'$
  - ▶ Transmit using a rate  $R' > R$  random block-code of length  $n \gg k$

# Packetization: first try

- Use block-code
  - ▶ Group bits into packets with length  $nR'$
  - ▶ Transmit using a rate  $R' > R$  random block-code of length  $n \gg k$
  - ▶ Use feedback to ACK/NAK only (low rate)
  - ▶ Retransmit block if unsuccessful

# Packetization: first try

- Use block-code
  - ▶ Group bits into packets with length  $nR'$
  - ▶ Transmit using a rate  $R' > R$  random block-code of length  $n \gg k$
  - ▶ Use feedback to ACK/NAK only (low rate)
  - ▶ Retransmit block if unsuccessful
- Effective block-length  $n + k \approx n$

# Packetization: first try

- Use block-code
  - ▶ Group bits into packets with length  $nR'$
  - ▶ Transmit using a rate  $R' > R$  random block-code of length  $n \gg k$
  - ▶ Use feedback to ACK/NAK only (low rate)
  - ▶ Retransmit block if unsuccessful
- Effective block-length  $n + k \approx n$
- Performance
  - ▶  $E_r(R) < E_0(\rho)$  governs probability of block retransmission

# Packetization: first try

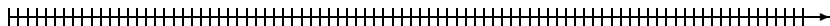
- Use block-code
  - ▶ Group bits into packets with length  $nR'$
  - ▶ Transmit using a rate  $R' > R$  random block-code of length  $n \gg k$
  - ▶ Use feedback to ACK/NAK only (low rate)
  - ▶ Retransmit block if unsuccessful
- Effective block-length  $n + k \approx n$
- Performance
  - ▶  $E_r(R) < E_0(\rho)$  governs probability of block retransmission
  - ▶ **Bad, even without accounting for queuing delay**

# Outline

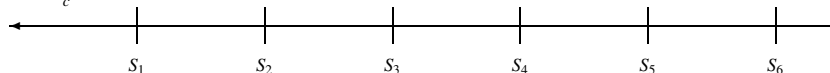
- 1 Background review
- 2 Problem 1: BEC: bitwise hard deadlines with limited feedback
  - ▶ Review of perfect feedback
  - ▶ Round-trip delay
  - ▶ Packetization and retransmission
  - ▶ **Limited feedback and Hybrid ARQ**
  - ▶ Low-rate: lists and disambiguation
- 3 Problem 2: BSC: bitwise soft deadlines with noisy feedback

# Low-rate feedback picture

Forward BEC uses



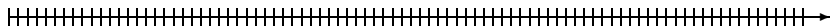
Rate  $\frac{1}{c}$  noiseless feedback channel uses



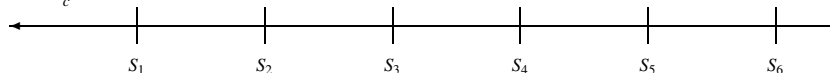
- $c$  is part of the problem, not tied to the block-length.

# Low-rate feedback picture

Forward BEC uses



Rate  $\frac{1}{c}$  noiseless feedback channel uses



- $c$  is part of the problem, not tied to the block-length.
- Assume target latency  $d \gg c$
- Assume round-trip time  $k \ll c$

# Hybrid ARQ

- Softer retransmission:

# Hybrid ARQ

- Softer retransmission:
  - ① Group bits into blocks of size  $nR$ . ( $c \ll n \ll d$ )

# Hybrid ARQ

- Softer retransmission:
  - 1 Group bits into blocks of size  $nR$ . ( $c \ll n \ll d$ )
  - 2 Hold blocks in a FIFO queue

# Hybrid ARQ

- Softer retransmission:
  - 1 Group bits into blocks of size  $nR$ . ( $c \ll n \ll d$ )
  - 2 Hold blocks in a FIFO queue
  - 3 Service using an  $\infty$ -length random codebook. (rateless code)

# Hybrid ARQ

- Softer retransmission:
  - 1 Group bits into blocks of size  $nR$ . ( $c \ll n \ll d$ )
  - 2 Hold blocks in a FIFO queue
  - 3 Service using an  $\infty$ -length random codebook. (rateless code)
  - 4 Feedback “ACK” when we can decode.

# Hybrid ARQ

- Softer retransmission:
  - 1 Group bits into blocks of size  $nR$ . ( $c \ll n \ll d$ )
  - 2 Hold blocks in a FIFO queue
  - 3 Service using an  $\infty$ -length random codebook. (rateless code)
  - 4 Feedback “ACK” when we can decode.
- Three delays:

# Hybrid ARQ

- Softer retransmission:
  - 1 Group bits into blocks of size  $nR$ . ( $c \ll n \ll d$ )
  - 2 Hold blocks in a FIFO queue
  - 3 Service using an  $\infty$ -length random codebook. (rateless code)
  - 4 Feedback “ACK” when we can decode.
- Three delays:
  - ▶ Assembly:  $n$  insignificant relative to  $d$

# Hybrid ARQ

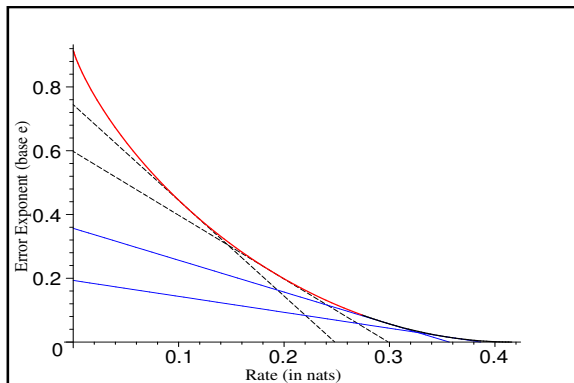
- Softer retransmission:
  - 1 Group bits into blocks of size  $nR$ . ( $c \ll n \ll d$ )
  - 2 Hold blocks in a FIFO queue
  - 3 Service using an  $\infty$ -length random codebook. (rateless code)
  - 4 Feedback “ACK” when we can decode.
- Three delays:
  - ▶ Assembly:  $n$  insignificant relative to  $d$
  - ▶ Queuing: Wait before servicing starts

# Hybrid ARQ

- Softer retransmission:
  - 1 Group bits into blocks of size  $nR$ . ( $c \ll n \ll d$ )
  - 2 Hold blocks in a FIFO queue
  - 3 Service using an  $\infty$ -length random codebook. (rateless code)
  - 4 Feedback “ACK” when we can decode.
- Three delays:
  - ▶ Assembly:  $n$  insignificant relative to  $d$
  - ▶ Queuing: Wait before servicing starts
  - ▶ Transmission: Service-time distribution

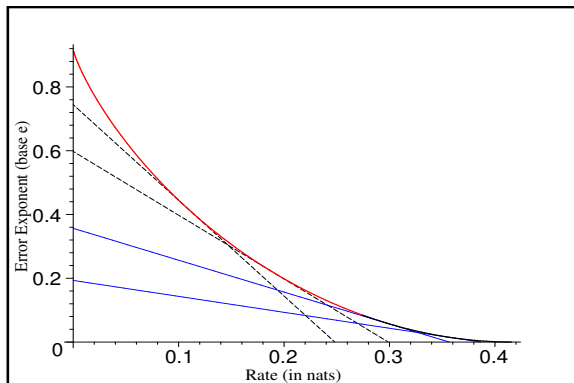
# Transmission-delay: operational interpretation of $E_0(\rho)$

*Block transmission time  $T$  can be bounded by a constant plus a geometric random variable.*



# Transmission-delay: operational interpretation of $E_0(\rho)$

*Block transmission time  $T$  can be bounded by a constant plus a geometric random variable.*



Need to do list-decoding at low rates.

# Queuing delay: the point message view

- Consider the queue at the level of blocks:
  - ▶ Arrive every  $n$  channel uses

# Queuing delay: the point message view

- Consider the queue at the level of blocks:
  - ▶ Arrive every  $n$  channel uses
  - ▶ When serviced, immediately consume a constant  $n \frac{R}{R_\rho}$  channel uses

# Queuing delay: the point message view

- Consider the queue at the level of blocks:
  - ▶ Arrive every  $n$  channel uses
  - ▶ When serviced, immediately consume a constant  $n \frac{R}{R_\rho}$  channel uses
  - ▶ After that, geometric service-time with  $1 - \delta_\rho = 1 - \exp(-E_0(\rho))$

# Queuing delay: the point message view

- Consider the queue at the level of blocks:
  - ▶ Arrive every  $n$  channel uses
  - ▶ When serviced, immediately consume a constant  $n \frac{R}{R_\rho}$  channel uses
  - ▶ After that, geometric service-time with  $1 - \delta_\rho = 1 - \exp(-E_0(\rho))$
- Delay  $> d$  implies  $\frac{d}{n}$  messages waiting in the queue

## Queuing delay: the point message view

- Consider the queue at the level of blocks:
  - ▶ Arrive every  $n$  channel uses
  - ▶ When serviced, immediately consume a constant  $n \frac{R}{R_\rho}$  channel uses
  - ▶ After that, geometric service-time with  $1 - \delta_\rho = 1 - \exp(-E_0(\rho))$
- Delay  $> d$  implies  $\frac{d}{n}$  messages waiting in the queue
- Suppose the queue last renewed  $r$  messages ago
  - ▶  $rn$  channel uses since then

# Queuing delay: the point message view

- Consider the queue at the level of blocks:
  - ▶ Arrive every  $n$  channel uses
  - ▶ When serviced, immediately consume a constant  $n \frac{R}{R_\rho}$  channel uses
  - ▶ After that, geometric service-time with  $1 - \delta_\rho = 1 - \exp(-E_0(\rho))$
- Delay  $> d$  implies  $\frac{d}{n}$  messages waiting in the queue
- Suppose the queue last renewed  $r$  messages ago
  - ▶  $rn$  channel uses since then
  - ▶ At most  $q = r - \frac{d}{n}$  blocks serviced

# Queuing delay: the point message view

- Consider the queue at the level of blocks:
  - ▶ Arrive every  $n$  channel uses
  - ▶ When serviced, immediately consume a constant  $n \frac{R}{R_\rho}$  channel uses
  - ▶ After that, geometric service-time with  $1 - \delta_\rho = 1 - \exp(-E_0(\rho))$
- Delay  $> d$  implies  $\frac{d}{n}$  messages waiting in the queue
- Suppose the queue last renewed  $r$  messages ago
  - ▶  $rn$  channel uses since then
  - ▶ At most  $q = r - \frac{d}{n}$  blocks serviced
  - ▶ So remove  $qn \frac{R}{R_\rho}$  channel uses

## Queuing delay: the point message view

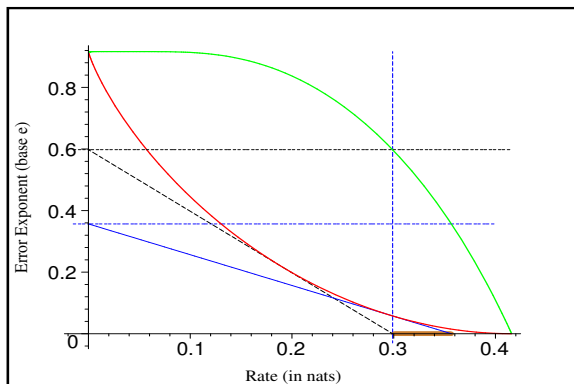
- Consider the queue at the level of blocks:
  - ▶ Arrive every  $n$  channel uses
  - ▶ When serviced, immediately consume a constant  $n \frac{R}{R_\rho}$  channel uses
  - ▶ After that, geometric service-time with  $1 - \delta_\rho = 1 - \exp(-E_0(\rho))$
- Delay  $> d$  implies  $\frac{d}{n}$  messages waiting in the queue
- Suppose the queue last renewed  $r$  messages ago
  - ▶  $rn$  channel uses since then
  - ▶ At most  $q = r - \frac{d}{n}$  blocks serviced
  - ▶ So remove  $qn \frac{R}{R_\rho}$  channel uses
  - ▶ Leaving  $d + qn(1 - \frac{R}{R_\rho})$  “slack” uses of which only  $q$  were successful.

## Queuing delay: the point message view

- Consider the queue at the level of blocks:
  - ▶ Arrive every  $n$  channel uses
  - ▶ When serviced, immediately consume a constant  $n \frac{R}{R_\rho}$  channel uses
  - ▶ After that, geometric service-time with  $1 - \delta_\rho = 1 - \exp(-E_0(\rho))$
- Delay  $> d$  implies  $\frac{d}{n}$  messages waiting in the queue
- Suppose the queue last renewed  $r$  messages ago
  - ▶  $rn$  channel uses since then
  - ▶ At most  $q = r - \frac{d}{n}$  blocks serviced
  - ▶ So remove  $qn \frac{R}{R_\rho}$  channel uses
  - ▶ Leaving  $d + qn(1 - \frac{R}{R_\rho})$  “slack” uses of which only  $q$  were successful.
- This is exactly like a rate  $\frac{1}{n(1 - \frac{R}{R_\rho})}$  code on a perfect-feedback BEC with erasure probability  $\delta_\rho$ .

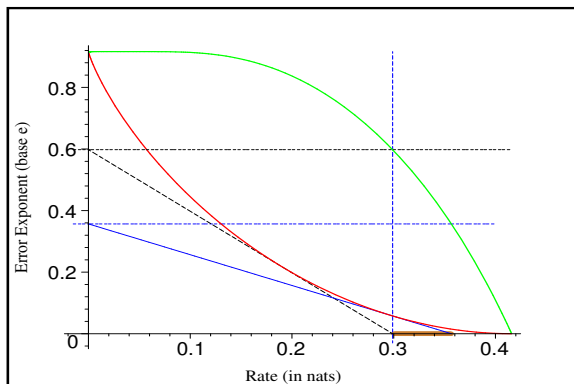
# Queuing delay: reduction to the low-rate erasure case

Pick  $R < R_\rho < C$  and aim for  $E_a^+(R_\rho) = E_0(\rho)$  exponent.



## Queuing delay: reduction to the low-rate erasure case

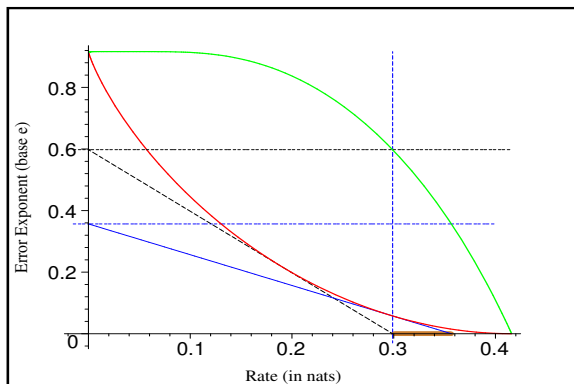
Pick  $R < R_\rho < C$  and aim for  $E_a^+(R_\rho) = E_0(\rho)$  exponent.



If  $n$  large, effective point-message rate  $(n(1 - \frac{R}{R'}))^{-1}$  is small.

## Queuing delay: reduction to the low-rate erasure case

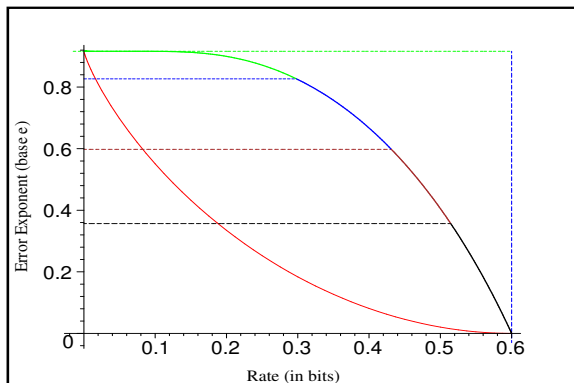
Pick  $R < R_\rho < C$  and aim for  $E_a^+(R_\rho) = E_0(\rho)$  exponent.



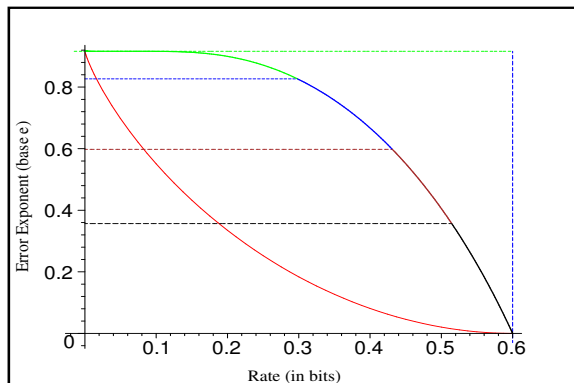
If  $n$  large, effective point-message rate  $(n(1 - \frac{R}{R'}))^{-1}$  is small.

Erasures focusing bound is approximately flat at low rates so queuing delay exponent  $\approx E_0(\rho)$ .

# Performance



# Performance



What is the price to get list-decoding gains?

# How to use list-decoding

- At low rates, forward error correction suffers from ambiguity.

# How to use list-decoding

- At low rates, forward error correction suffers from ambiguity.
  - 1 Group bits into blocks of size  $nR$ . ( $k \ll n \ll d$ )
  - 2 Hold blocks in a FIFO queue
  - 3 Transmit using an  $\infty$ -length random codebook. (rateless code)

# How to use list-decoding

- At low rates, forward error correction suffers from ambiguity.
  - 1 Group bits into blocks of size  $nR$ . ( $k \ll n \ll d$ )
  - 2 Hold blocks in a FIFO queue
  - 3 Transmit using an  $\infty$ -length random codebook. (rateless code)
  - 4 Feedback “ACK” when we can **almost** decode.

# How to use list-decoding

- At low rates, forward error correction suffers from ambiguity.
  - 1 Group bits into blocks of size  $nR$ . ( $k \ll n \ll d$ )
  - 2 Hold blocks in a FIFO queue
  - 3 Transmit using an  $\infty$ -length random codebook. (rateless code)
  - 4 Feedback “ACK” when we can **almost** decode.
  - 5 Send an  $O((L - 1) \log(nR))$  message requesting clarification of specific bits within the block

# How to use list-decoding

- At low rates, forward error correction suffers from ambiguity.
  - 1 Group bits into blocks of size  $nR$ . ( $k \ll n \ll d$ )
  - 2 Hold blocks in a FIFO queue
  - 3 Transmit using an  $\infty$ -length random codebook. (rateless code)
  - 4 Feedback “ACK” when we can **almost** decode.
  - 5 Send an  $O((L - 1) \log(nR))$  message requesting clarification of specific bits within the block
  - 6 Use repetition codes to clarify

# How to use list-decoding

- At low rates, forward error correction suffers from ambiguity.
  - 1 Group bits into blocks of size  $nR$ . ( $k \ll n \ll d$ )
  - 2 Hold blocks in a FIFO queue
  - 3 Transmit using an  $\infty$ -length random codebook. (rateless code)
  - 4 Feedback “ACK” when we can **almost** decode.
  - 5 Send an  $O((L - 1) \log(nR))$  message requesting clarification of specific bits within the block
  - 6 Use repetition codes to clarify
  - 7 ACK the clarifications

# How to use list-decoding

- At low rates, forward error correction suffers from ambiguity.
  - 1 Group bits into blocks of size  $nR$ . ( $k \ll n \ll d$ )
  - 2 Hold blocks in a FIFO queue
  - 3 Transmit using an  $\infty$ -length random codebook. (rateless code)
  - 4 Feedback “ACK” when we can **almost** decode.
  - 5 Send an  $O((L - 1) \log(nR))$  message requesting clarification of specific bits within the block
  - 6 Use repetition codes to clarify
  - 7 ACK the clarifications
- A block corresponds to  $L$  point messages — low rate relative to slack.

# How to use list-decoding

- At low rates, forward error correction suffers from ambiguity.
  - 1 Group bits into blocks of size  $nR$ . ( $k \ll n \ll d$ )
  - 2 Hold blocks in a FIFO queue
  - 3 Transmit using an  $\infty$ -length random codebook. (rateless code)
  - 4 Feedback “ACK” when we can **almost** decode.
  - 5 Send an  $O((L - 1) \log(nR))$  message requesting clarification of specific bits within the block
  - 6 Use repetition codes to clarify
  - 7 ACK the clarifications
- A block corresponds to  $L$  point messages — low rate relative to slack.
- Approaches the focusing bound.

# Final comments on BEC

- Perfect feedback performance with arbitrarily low feedback rate.

# Final comments on BEC

- Perfect feedback performance with arbitrarily low feedback rate.
- For  $L = 1$ , random linear block coding is good enough.

# Final comments on BEC

- Perfect feedback performance with arbitrarily low feedback rate.
- For  $L = 1$ , random linear block coding is good enough.
- For  $L > 1$ , random polynomial block coding suffices.

# Final comments on BEC

- Perfect feedback performance with arbitrarily low feedback rate.
- For  $L = 1$ , random linear block coding is good enough.
- For  $L > 1$ , random polynomial block coding suffices.
- The code is “anytime” in that it is delay universal — application can pick what latency is needed.

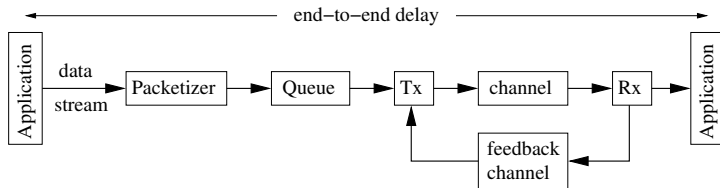
# Final comments on BEC

- Perfect feedback performance with arbitrarily low feedback rate.
- For  $L = 1$ , random linear block coding is good enough.
- For  $L > 1$ , random polynomial block coding suffices.
- The code is “anytime” in that it is delay universal — application can pick what latency is needed.
- Computation depends only on  $n$ , not on delay.

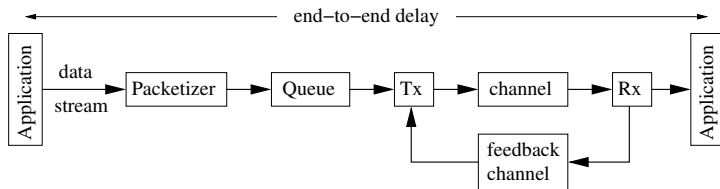
# Outline

- 1 Background review
- 2 Problem 1: BEC: bitwise hard deadlines with limited feedback
- 3 **Problem 2: BSC: bitwise soft deadlines with noisy feedback**
  - ▶ Perfect feedback: the opportunity
  - ▶ Within-stream control messages
  - ▶ The “Hallucination Bound”
  - ▶ Noisy feedback

# A streaming data perspective

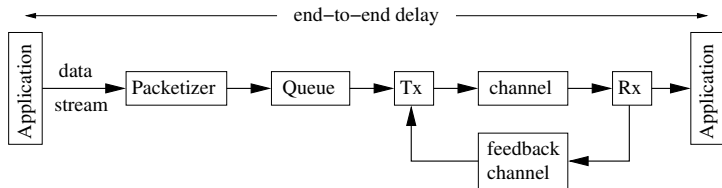


# A streaming data perspective



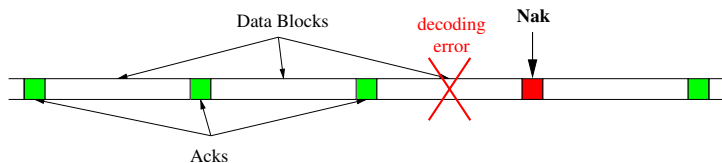
- Queue is optional. Needed if retransmissions are required

# A streaming data perspective



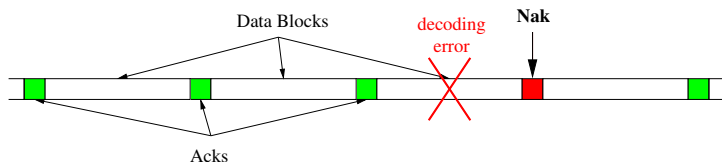
- Queue is optional. Needed if retransmissions are required
- If retransmissions are rare, then **expected** end-to-end delay is dominated by the Tx to Rx delay.

# An opportunity presents itself



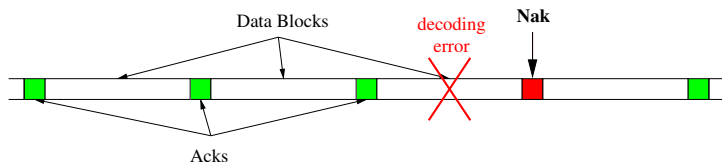
- Erasures are rare so most messages are confirmed.

# An opportunity presents itself



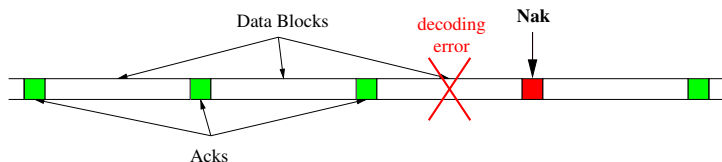
- Erasures are rare so most messages are confirmed.
- We are wasting channel uses.

# An opportunity presents itself



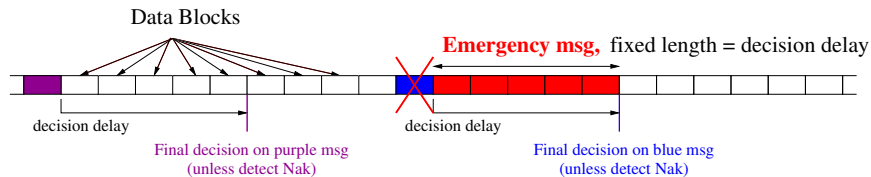
- Erasures are rare so most messages are confirmed.
- We are wasting channel uses.
- What if we only sent NAKs when needed?

# An opportunity presents itself



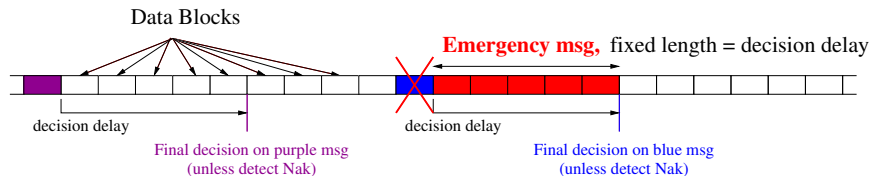
- Erasures are rare so most messages are confirmed.
- We are wasting channel uses.
- What if we only sent NAKs when needed?
- Have a special message for this purpose.

# Sliding blocks with collective punishment only (Kudryashov-79)



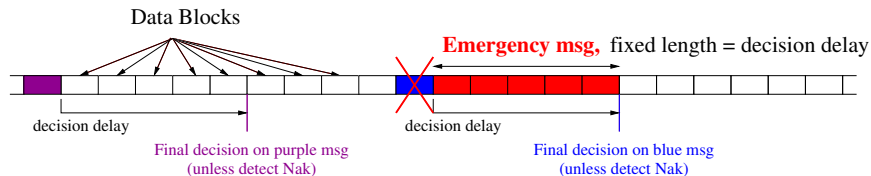
- Make packet size  $n$  much smaller than soft deadline  $d$ .

# Sliding blocks with collective punishment only (Kudryashov-79)



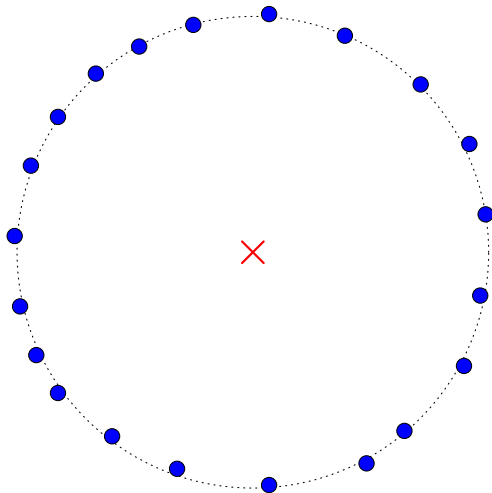
- Make packet size  $n$  much smaller than soft deadline  $d$ .
- A NAK collectively denies the past  $\frac{d}{n} - 1$  packets

# Sliding blocks with collective punishment only (Kudryashov-79)

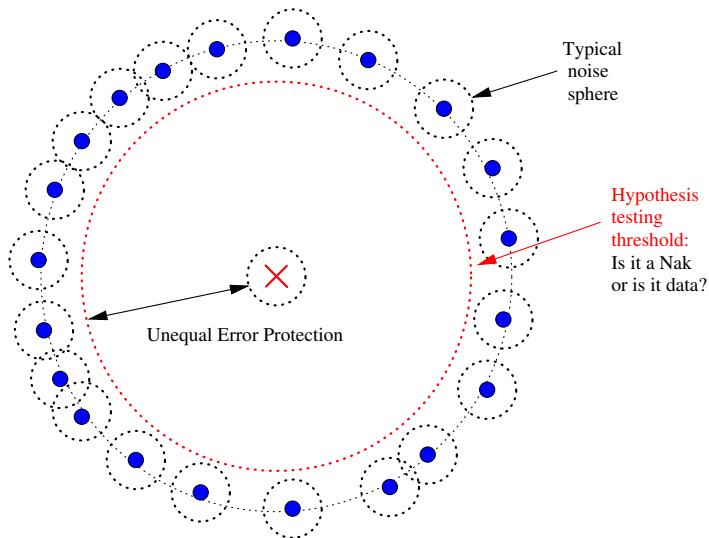


- Make packet size  $n$  much smaller than soft deadline  $d$ .
- A NAK collectively denies the past  $\frac{d}{n} - 1$  packets
- Error only if  $\frac{d}{n} - 1$  NAKs are all missed

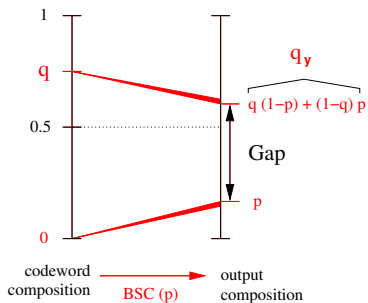
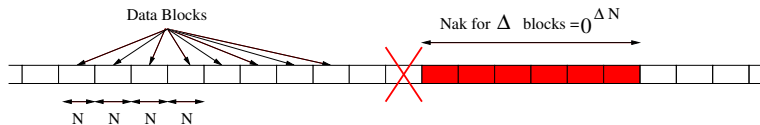
# Unequal error protection required in codebook



# Unequal error protection required in codebook

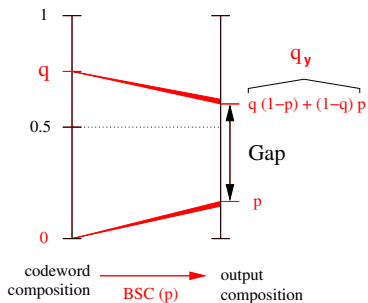
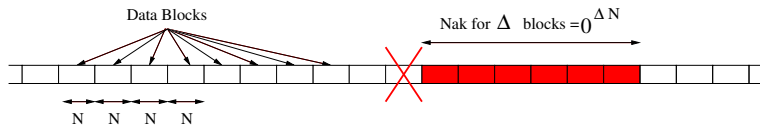


# Specialize to BSC case



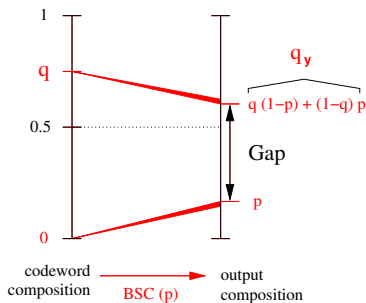
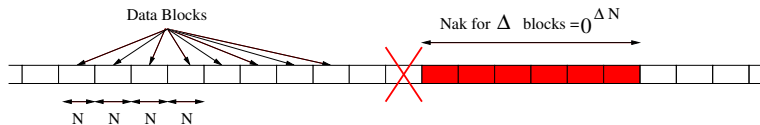
- Use all zero for NAK

# Specialize to BSC case



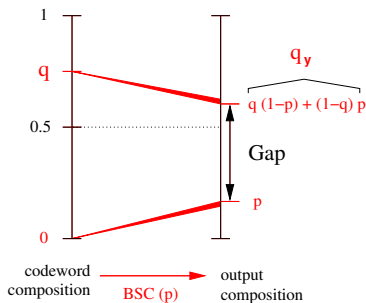
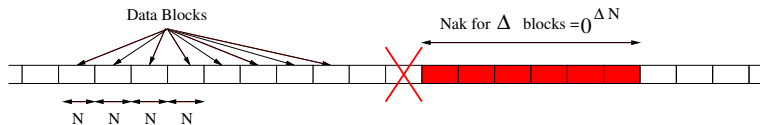
- Use all zero for NAK
- Use composition  $q$  code for data:  $R < H(q) - H(p)$

# Specialize to BSC case



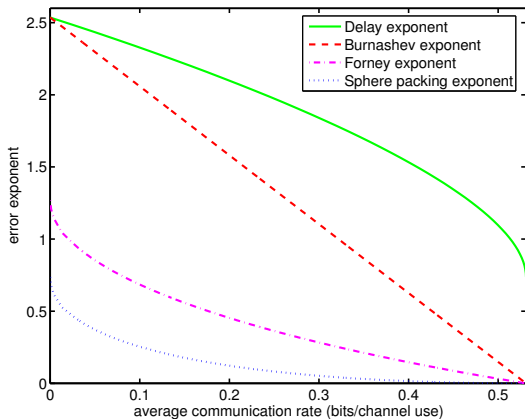
- Use all zero for NAK
- Use composition  $q$  code for data:  $R < H(q) - H(p)$
- Probability of missed NAK is  $2^{-ND(q_y||p)}$

# Specialize to BSC case

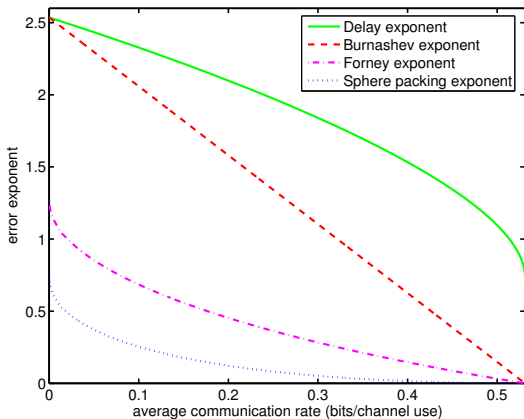


- Use all zero for NAK
- Use composition  $q$  code for data:  $R < H(q) - H(p)$
- Probability of missed NAK is  $2^{-ND(q_y||p)}$
- Get  $\Delta$  chances:  $2^{-\Delta ND(q_y||p)}$

# Resulting exponents

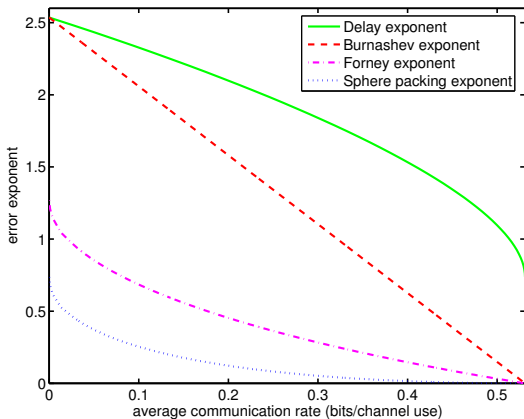


# Resulting exponents



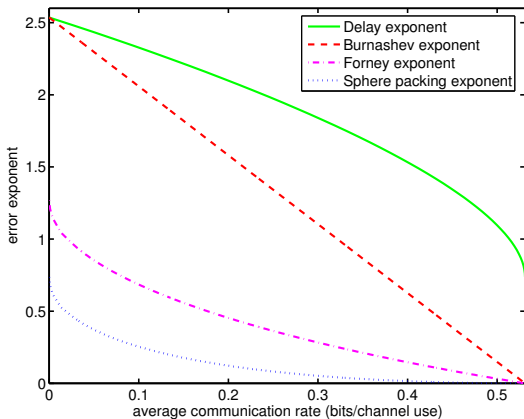
- Positive error exponent  $D(\frac{1}{2}||p)$  even at capacity!

# Resulting exponents



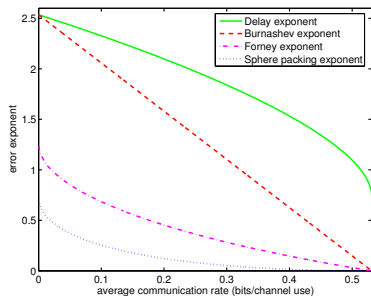
- Positive error exponent  $D(\frac{1}{2}||p)$  even at capacity!
- Matches Horstein's exponent.

# Resulting exponents



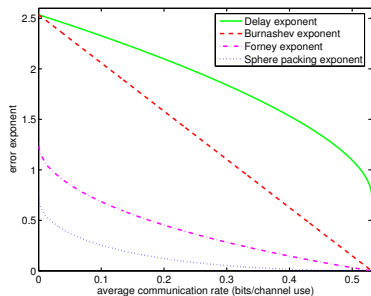
- Positive error exponent  $D(\frac{1}{2}||p)$  even at capacity!
- Matches Horstein's exponent.
- Far better than the focusing bound.

# Matches the “Hallucination Bound”



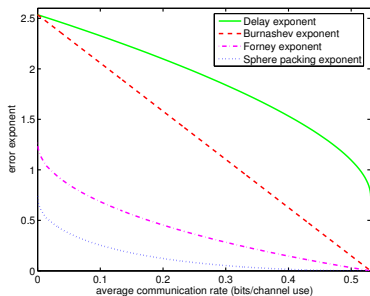
- No converse for Forney, unlike Burnashev and Sphere-packing.

# Matches the “Hallucination Bound”



- No converse for Forney, unlike Burnashev and Sphere-packing.
- What about here?

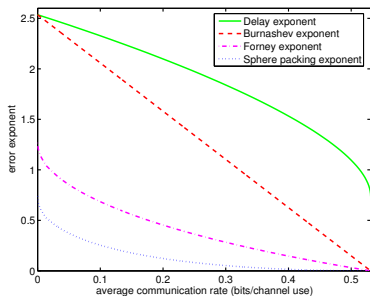
# Matches the “Hallucination Bound”



- Decoding correctly requires a certain “typical” volume of output sequences

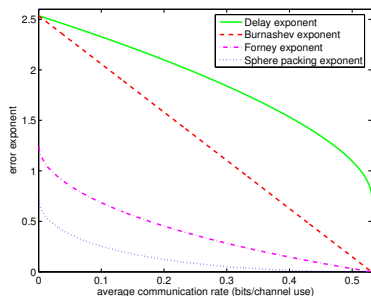
- No converse for Forney, unlike Burnashev and Sphere-packing.
- What about here?

# Matches the “Hallucination Bound”



- Decoding correctly requires a certain “typical” volume of output sequences
  - If the channel forces you to hallucinate for  $d$  time-steps, you are doomed.
- 
- No converse for Forney, unlike Burnashev and Sphere-packing.
  - What about here?

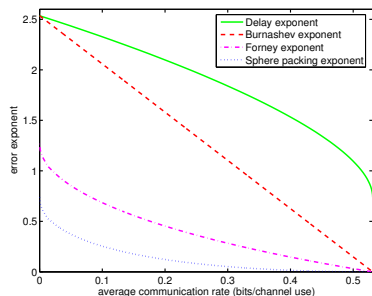
# Matches the “Hallucination Bound”



- No converse for Forney, unlike Burnashev and Sphere-packing.
- What about here?

- Decoding correctly requires a certain “typical” volume of output sequences
- If the channel forces you to hallucinate for  $d$  time-steps, you are doomed.
- With feedback, you can choose the channel input to minimize this probability.

# Matches the “Hallucination Bound”



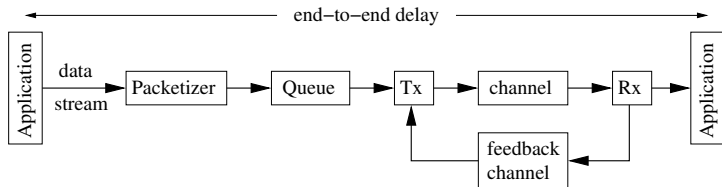
- No converse for Forney, unlike Burnashev and Sphere-packing.
- What about here?

- Decoding correctly requires a certain “typical” volume of output sequences
- If the channel forces you to hallucinate for  $d$  time-steps, you are doomed.
- With feedback, you can choose the channel input to minimize this probability.
- Matches achievability perfectly.

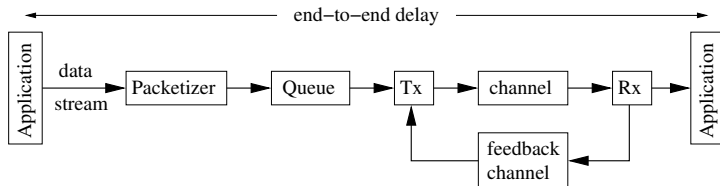
# Outline

- 1 Background review
- 2 Problem 1: BEC: bitwise hard deadlines with limited feedback
- 3 Problem 2: BSC: bitwise soft deadlines with noisy feedback
  - ▶ Perfect feedback: the opportunity
  - ▶ Within-stream control messages
  - ▶ The “Hallucination Bound”
  - ▶ **Noisy feedback**

# Two distinct issues

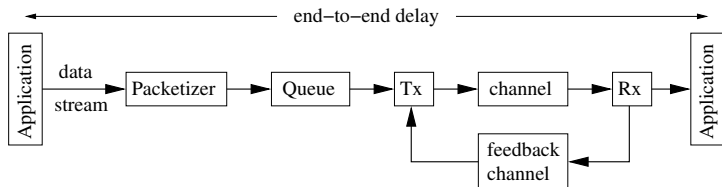


# Two distinct issues



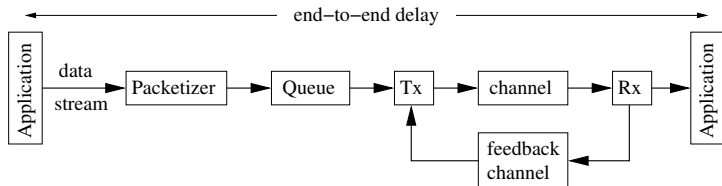
- Retransmission control: maintaining synchronization

# Two distinct issues



- Retransmission control: maintaining synchronization
  - ▶ Can model the state of the receiver as a random walk with drift.

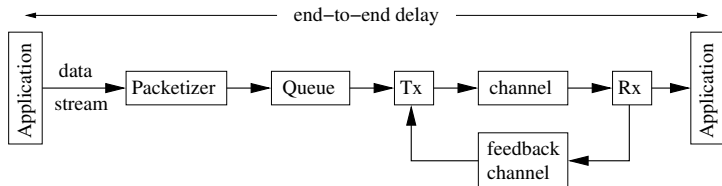
# Two distinct issues



- Retransmission control: maintaining synchronization

- ▶ Can model the state of the receiver as a random walk with drift.
- ▶ Unstable process, but it can be tracked using an *anytime* code.

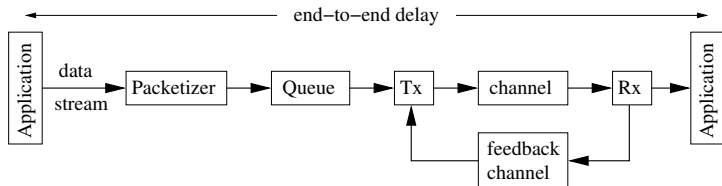
# Two distinct issues



## ● Retransmission control: maintaining synchronization

- ▶ Can model the state of the receiver as a random walk with drift.
- ▶ Unstable process, but it can be tracked using an *anytime* code.
- ▶ Since retransmissions are rare, can afford extra latency to allow state estimates to converge.

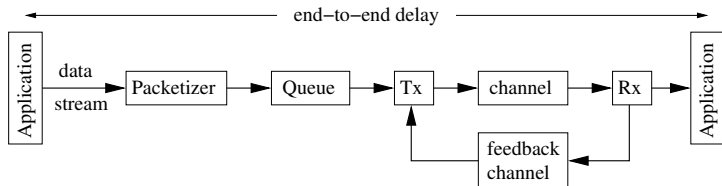
# Two distinct issues



## ● Retransmission control: maintaining synchronization

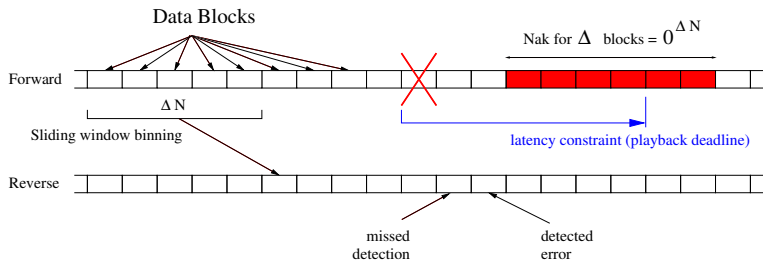
- ▶ Can model the state of the receiver as a random walk with drift.
- ▶ Unstable process, but it can be tracked using an *anytime* code.
- ▶ Since retransmissions are rare, can afford extra latency to allow state estimates to converge.
- ▶ Synchronization rate is less than 1 bit per packet.

# Two distinct issues



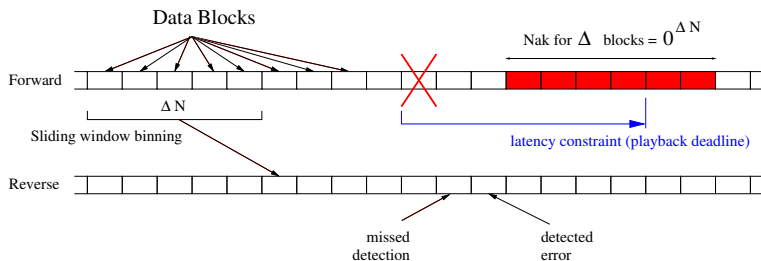
- Retransmission control: maintaining synchronization
  - ▶ Can model the state of the receiver as a random walk with drift.
  - ▶ Unstable process, but it can be tracked using an *anytime* code.
  - ▶ Since retransmissions are rare, can afford extra latency to allow state estimates to converge.
  - ▶ Synchronization rate is less than 1 bit per packet.
- How to NAK?

# What is needed to NAK?



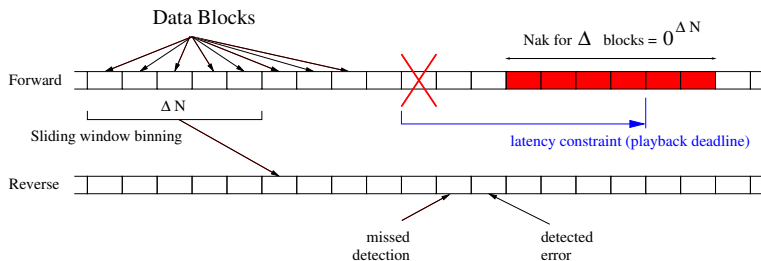
- Must know if any errors in the sliding window.

# What is needed to NAK?



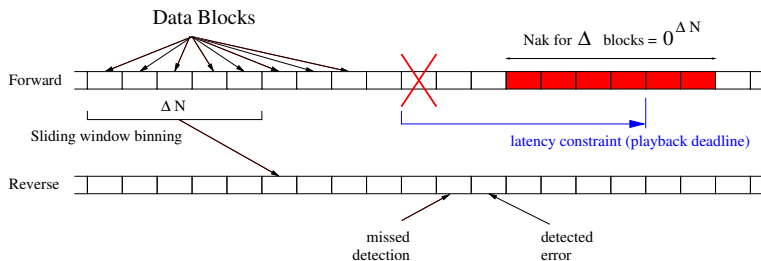
- Must know if any errors in the sliding window.
- Identification problem since encoder knows what it wants to hear.

# What is needed to NAK?



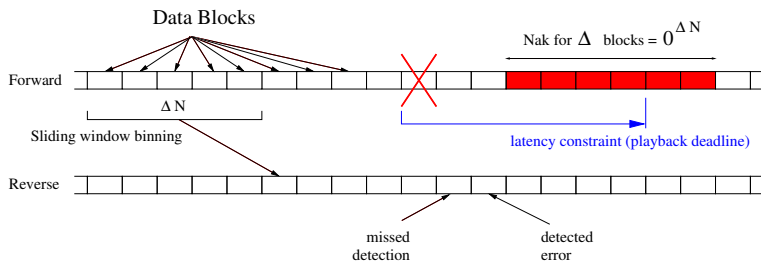
- Must know if any errors in the sliding window.
- Identification problem since encoder knows what it wants to hear.
- Use a random hash of entire sliding window.

# What is needed to NAK?



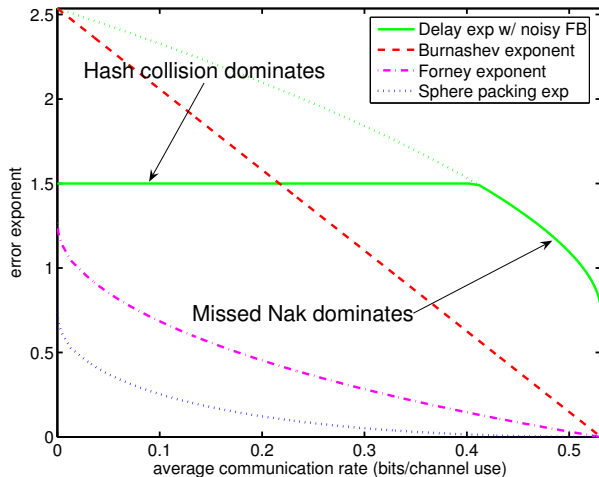
- Must know if any errors in the sliding window.
- Identification problem since encoder knows what it wants to hear.
- Use a random hash of entire sliding window.
- Compare  $\Pr(\text{hash collision})$  with  $\Pr(\text{missed NAK})$

# What is needed to NAK?



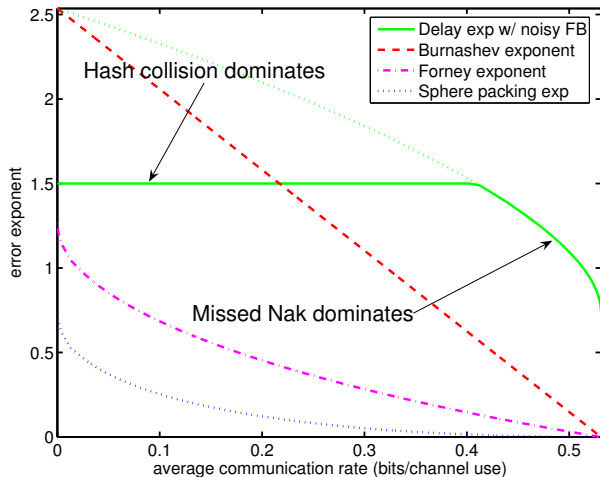
- Must know if any errors in the sliding window.
- Identification problem since encoder knows what it wants to hear.
- Use a random hash of entire sliding window.
- Compare  $\Pr(\text{hash collision})$  with  $\Pr(\text{missed NAK})$
- If  $C_{fb} > D(q_y||p)$ , no loss in net exponent!

# Feedback capacity acts as a ceiling to reliability



- **Feedback reliability gains are robust to noisy feedback.**

# Feedback capacity acts as a ceiling to reliability



- **Feedback reliability gains are robust to noisy feedback.**
- Open problem: does this also hold in the hard deadline case?