Balancing forward and feedback error correction

Anant Sahai
based in part on joint work with:
Stark Draper, Tunc Simsek, and Paul Liu

sahai@eecs.berkeley.edu
Wireless Foundations
Department of Electrical Engineering and Computer Sciences
University of California at Berkeley

Major Support from NSF ITR

November 6th 2006
Information theory studies:

- **Goals:** high rate, low delay, and low probability of error.

![Diagram of a general communication system](image)

*Fig. 1—Schematic diagram of a general communication system.*
Information theory studies:

- Goals: high rate, low delay, and low probability of error.
- What if there is feedback?
Information theory studies:

- Feedback
  - Goals: high rate, low delay, and low probability of error.
  - What if there is feedback?
    - If noise is memoryless, capacity is unchanged.
    - What about delay and probability of error?
Information theory studies:

- Goals: high rate, low delay, and low probability of error.
- 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 studies:

Goals: high rate, low delay, and low probability of error.

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?
“Feedback communications was an area of intense activity in 1968... A number of authors had shown constructive, even simple, schemes using noiseless feedback to achieve Shannon-like behavior... The situation in 1973 is dramatically different... The subject itself seems to be a burned out case...

In extending the simple noiseless feedback model to allow for more realistic situations, such as noisy feedback channels, bandlimited channels, and peak power constraints, theorists discovered a certain “brittleness” or sensitivity in their previous results...”
“Feedback communications was an area of intense activity in 1968... A number of authors had shown constructive, even simple, schemes using noiseless feedback to achieve Shannon-like behavior... The situation in 1973 is dramatically different... The subject itself seems to be a burned out case...

In extending the simple noiseless feedback model to allow for more realistic situations, such as noisy feedback channels, bandlimited channels, and peak power constraints, theorists discovered a certain “brittleness” or sensitivity in their previous results...”

Our goal: show “robustness” of gains due to feedback.
1. Problem and Background
2. Erasure channels: packet-wise hard deadlines with limited feedback
3. BSC: bitwise soft deadlines with truly noisy feedback
Communication with a latency requirement

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

- Bits/packets arrive steadily at rate $R$ 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 $R$ per channel use.

End-to-end latency requirement of $d$:

- **Hard**: Declared or undetected errors are equally bad.
- **Soft**: Undetected errors are very bad, but declared errors should be infrequent.
Review: Fixed block-codes for hard deadlines

- Buffer-up $nR$ bits and use a block-code
- Must decode after a further $n$ channel uses
Review: Fixed block-codes for hard deadlines

- Buffer-up $nR$ bits and use a block-code
- Must decode after a further $n$ 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 block-codes for hard deadlines

- Buffer-up $nR$ bits and use a block-code
- Must decode after a further $n$ 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-77)

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

- Holds with or without feedback
Review: Fixed block-codes for hard deadlines

- Buffer-up $nR$ bits and use a block-code
- Must decode after a further $n$ 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:

\[
E_{sp}(R) = \max_{\bar{q}} \min_{G: I(\bar{q},G) \leq R} D(G||P|\bar{q}) \\
= \sup_{\rho \geq 0} E_0(\rho) - \rho R
\]

\[
E_0(\rho) = \max_{\bar{q}} - \ln \sum_z \left[ \sum_y q_y p^{\frac{1}{1+\rho}}_{z|y} \right]^{(1+\rho)}
\]

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

Hard decision regions cover space
Hard decisions regions but check hash signatures
Review: Fixed blocks, Soft deadlines

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

- 1 bit feedback can request retransmissions
- Can interpret as expected block-length

- Hard decisions regions but check hash signatures

Anant Sahai (UC Berkeley)
Limited Feedback
Nov 6, 2006
Review: Fixed blocks, Soft deadlines

- Hard decisions regions but check hash signatures
- 1 bit feedback can request retransmissions
- Can interpret as expected block-length
- Run close to capacity
- Use \( \approx n(C - R) \) bits for signatures

\[ Anant \ Sahai \ (UC \ Berkeley) \]

Limited Feedback

Nov 6, 2006 8/44
Review: Fixed blocks, Soft deadlines

- Hard decisions regions but check hash signatures

- 1 bit feedback can request retransmissions
- Can interpret as expected block-length
- Run close to capacity
- Use \( \approx n(C - R) \) bits for signatures
- Linear slope for error exponent
Refuse to decide when ambiguous

- 1 bit feedback can request retransmissions
- Can interpret as expected block-length
Refuse to decide when ambiguous

- 1 bit feedback can request retransmissions
- Can interpret as expected block-length
- Decision regions catch the typical sets only
Refuse to decide when ambiguous

- 1 bit feedback can request retransmissions
- Can interpret as expected block-length
- Decision regions catch the typical sets only
- Better error exponents at lower rates
Review: Fixed blocks, Soft deadlines: Burnashev-76

- Is more feedback helpful?
- Burnashev said yes:
Is more feedback helpful?
Burnashev said yes:
  - Considered expected stopping time and used Martingale arguments.
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 $C_1 = \max_{i,j} D(p_i \| p_j)$
Is more feedback helpful?

Burnashev said yes:

- Considered expected stopping time and used Martingale arguments.
- Showed $C_1 \left(1 - \frac{R}{C}\right)$ was a bound where $C_1 = \max_{i,j} D(p_i || p_j)$.
**Data Transmission:** $\lambda n$ channel uses for block code at $R \sim C$
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 $R \approx C$
- **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.
- $\Pr[\text{err}] = \Pr[\text{NAK} \rightarrow \text{ACK}] = 2^{-(1-\lambda)nC_1} \approx 2^{-nC_1(1-\frac{R}{C})}$
Review: Yamamoto-Itoh-79 strategy attains the Burnashev bound

Data Transmission: $\lambda n$ channel uses for block code at $R \approx C$

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.

$$\Pr[\text{err}] = \Pr[\text{NAK} \rightarrow \text{ACK}] = 2^{-(1-\lambda)nC_1} \approx 2^{-nC_1(1-\frac{R}{C})}$$

Moral: Collective reward/punishment is good for reliability
Outline

1. Problem and Background
2. Erasure channels: packet-wise hard deadlines with limited feedback
   - Mildly delayed, but high-rate and reliable, feedback
   - Low-rate, but reliable, feedback
   - Unreliable feedback
3. BSC: bitwise soft deadlines with truly noisy feedback
Nonblock coding for BEC with perfect feedback

- Simple capacity $1 - \delta$ bits per channel use
- With perfect non-delayed feedback, simple to achieve: retransmit until it gets through
Nonblock coding for BEC 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 deadlines 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*
First approach: treat as \( k \) parallel unit-delay channels

Serious penalty to waiting \( k \) steps between retransmissions
The traditional approach: block-coding

- Group bits into packets with length $nR'$
  - Transmit using a rate $R' > R$ random block-code of length $n \gg k$
The traditional approach: block-coding

- 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
The traditional approach: block-coding

- 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 $n + k \approx n$
The traditional approach: block-coding

- 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
The traditional approach: block-coding

- 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
Low-rate ("bandlimited") feedback picture

Forward BEC uses

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

$S_1$  $S_2$  $S_3$  $S_4$  $S_5$  $S_6$

$c$ is part of the problem, not tied to the block-length.
Low-rate (“bandlimited”) 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 $k \ll c$
Incremental Redundancy Hybrid ARQ

- Gentler retransmission:
Gentler retransmission:

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

1. Group bits into blocks of size $nR$. ($c \ll n \ll d$)
2. Hold blocks in a FIFO queue
Gentler 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)
Gentler 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.
Incremental Redundancy Hybrid ARQ

- Gentler 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:
Gentler 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 $d$
Gentler 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
Gentler 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.

Need to do list-decoding at low rates.
How to pick $\rho$ for bounding

Pick $R < R_\rho < C$ and aim for $E^+_a(R_\rho) = E_0(\rho)$ exponent.
Consider the queue at the level of blocks:

- Arrive every $n$ channel uses
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 $\delta_{\rho} = \exp(-E_0(\rho))$
Queueing delay: the point message view

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

- Delay $> d$ implies $\frac{d}{n}$ messages waiting in the queue
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 $\delta_\rho = \exp(-E_0(\rho))$

Delay $> d$ implies $\frac{d}{n}$ messages waiting in the queue

Suppose the queue last renewed $r$ messages ago
- $rn$ channel uses since then
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 $\delta_\rho = \exp(-E_0(\rho))$

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 \( \delta_\rho = \exp(-E_0(\rho)) \)

- 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 $\delta_\rho = \exp(-E_0(\rho))$

- 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.
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 $\delta_\rho = \exp(-E_0(\rho))$

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 bit-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.

If $n$ large, effective point-message rate $(n(1 - \frac{R}{R_\rho}))^{-1}$ is small.
Pick $R < R_\rho < C$ and aim for $E_\rho^+(R_\rho) = E_0(\rho)$ exponent.

If $n$ large, effective point-message rate $(n(1 - \frac{R}{R_\rho}))^{-1}$ is small. Erasure focusing bound is approximately flat at low rates so queuing delay exponent $\approx -\ln \delta_\rho = E_0(\rho)$. 
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 back an $O((L - 1) \log (nR))$ mini-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 back an $O((L - 1) \log(nR))$ mini-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 back an $O((L - 1) \log(nR))$ mini-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 back an $O((L - 1) \log(nR))$ mini-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 $O(n)$ 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 back an $O((L - 1) \log(nR))$ mini-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 $O(n)$ slack.

- Approaches the focusing bound.
Forward packet-erasure channels $\delta$

Rate $\frac{1}{c}$ packet-erasure feedback channels $\delta_f$

- Both channels drop packets randomly
Unreliable feedback picture

Forward packet-erasure channels $\delta$

Rate $\frac{1}{c}$ packet-erasure feedback channels $\delta_f$

- Both channels drop packets randomly
- Assume packets of moderate size that can have “header bits”
Forward packet-erasure channels $\delta$

Rate $\frac{1}{c}$ packet-erasure feedback channels $\delta_f$

- Both channels drop packets randomly
- Assume packets of moderate size that can have “header bits”
- Assume target latency $d \gg c, k$
How to deal with unreliable feedback

- Basic 4-Part Strategy:
  - Incremental transmission of message block
  - Feedback a request for disambiguation message
  - Repetition-code the disambiguation
  - Feedback a final ACK
How to deal with unreliable feedback

- Basic 4-Part Strategy:
  - Incremental transmission of message block
  - Feedback a request for disambiguation message
  - Repetition-code the disambiguation
  - Feedback a final ACK

- Add one header bit to forward packets
  - 0: main message packet
  - 1: first disambiguation message packet
  - 0: second disambiguation message packet
  - :
How to deal with unreliable feedback

- Basic 4-Part Strategy:
  - Incremental transmission of message block
  - Feedback a request for disambiguation message
  - Repetition-code the disambiguation
  - Feedback a final ACK

- Add one header bit to forward packets
  - 0: main message packet
  - 1: first disambiguation message packet
  - 0: second disambiguation message packet

- Repetition-code the feedback messages
How to deal with unreliable feedback

- Basic 4-Part Strategy:
  - Incremental transmission of message block
  - **Feedback** a request for disambiguation message
  - Repetition-code the disambiguation
  - **Feedback** a final ACK

- Add one header bit to forward packets
  - 0: main message packet
  - 1: first disambiguation message packet
  - 0: second disambiguation message packet
  - ...

- Repetition-code the **feedback messages**
  - Geometric service time as \((\delta_f^\frac{1}{c})^d\)
How to deal with unreliable feedback

- Basic 4-Part Strategy:
  - Incremental transmission of message block
  - Feedback a request for disambiguation message
  - Repetition-code the disambiguation
  - Feedback a final ACK

- Add one header bit to forward packets
  - 0: main message packet
  - 1: first disambiguation message packet
  - 0: second disambiguation message packet
  
- Repetition-code the feedback messages
  - Geometric service time as $\left(\delta_f^{-c}\right)^d$

- Each message now acts like $O(2 + (L - 1) \log(nR))$ point messages.
  - Low rate relative to $O(n)$ slack
  - As long as $\frac{-1}{c} \log \delta_f > E_0(\rho)$ target exponent, no loss in achieved end-to-end delay exponent!
Performance with unreliable feedback channel

- Balanced forward+feedback strategy
- Forward only strategy
- Naive feedback strategy with free feedback
- Rate limit with round-trip quality
- Capacity Limit due to forward quality
Performance with unreliable feedback channel

![Graph](image)

- Balanced forward+feedback strategy
- Forward only strategy
- Limited with free 1/3 rate feedback
$\frac{c-1}{c}$ forward packet-erasure channels $\delta$

Rate $\frac{1}{c}$ packet-erasure feedback channels $\delta$

- Both users share a single physical channel
- Half-duplex constraint: only one can use at a time
- Assume we must schedule them in advance
- Same erasure probability in both directions
Shared unreliable channel used for feedback

If feedback were free

Delay Error Exponent (base e)

Rate (in packets per slot)

Forward only strategy

Pure feedback strategy

1:1 split
Shared unreliable channel used for feedback

- Pure feedback strategy
- Forward only strategy
- Dividing time slots between feedback and forward
- If feedback were free

Rate (in packets per slot) vs. Delay Error Exponent (base e)
Shared unreliable channel used for feedback

If feedback were free

Forward only strategy

Dividing time slots between feedback and forward

Pure feedback strategy

Delay Error Exponent (base e)

Rate (in packets per slot)
Shared unreliable channel used for feedback

Anant Sahai (UC Berkeley)
Final comments on packet erasure channels

- Perfect feedback performance as long as $\frac{-1}{c} \log \delta_f$ is high enough.
Final comments on packet erasure channels

- Perfect feedback performance as long as $\frac{-1}{c} \log \delta_f$ is high enough.
- It is worth allocating slots for feedback even if this means taking them away from the forward channel.
Final comments on packet erasure channels

- Perfect feedback performance as long as $\frac{-1}{c} \log \delta_f$ is high enough.
- It is worth allocating slots for feedback even if this means taking them away from the forward channel.
- The code is “anytime” in that it is delay universal — application can pick what latency is needed.
- The code is universal over erasure probabilities.
Final comments on packet erasure channels

- Perfect feedback performance as long as $\frac{-1}{c} \log \delta_f$ is high enough.
- It is worth allocating slots for feedback even if this means taking them away from the forward channel.
- The code is “anytime” in that it is delay universal — application can pick what latency is needed.
- The code is universal over erasure probabilities.
- The code is mildly “broadcast-friendly.”
Perfect feedback performance as long as $-\frac{1}{c} \log \delta_f$ is high enough.

It is worth allocating slots for feedback even if this means taking them away from the forward channel.

The code is “anytime” in that it is delay universal — application can pick what latency is needed.

The code is universal over erasure probabilities.

The code is mildly “broadcast-friendly.”

Computation depends only on $n$, not on delay.
Problem and Background

2 Erasure channels: packet-wise hard deadlines with limited feedback

3 BSC: bitwise soft deadlines with truly noisy feedback
   - The opportunity
   - Noiseless feedback
   - Optimality since matches “Hallucination bound”
   - Noisy feedback
A streaming data perspective on soft deadlines
Queue is optional. Needed if retransmissions are required.
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.
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

Typical noise sphere

Hypothesis testing threshold:
Is it a Nak or is it data?

Unequal Error Protection
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

- Delay exponent
- Burnashev exponent
- Forney exponent
- Sphere packing exponent

Graph showing the relationship between average communication rate (bits/channel use) and error exponent for different exponent types.
Positive error exponent $D(\frac{1}{2}||p)$ even near capacity!
Positive error exponent $D(\frac{1}{2}||p)$ even near capacity!

Also 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?
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.

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.
- With feedback, you can choose the channel input to minimize this probability.
- Matches achievability.

- No converse for Forney, unlike Burnashev and Sphere-packing.
- What about here?
Noisy feedback: two distinct issues
Noisy feedback: two distinct issues

Retransmission control: maintaining synchronization
Noisy feedback: two distinct issues

Retransmission control: maintaining synchronization
- Can model the state of the receiver as a random walk with drift.
Noisy feedback: 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 \textit{anytime} code.
Noisy feedback: 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.
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 \textit{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.
Noisy feedback: 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.

Data Blocks

![Diagram of Data Blocks and Forwarding]

- 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(hash collision) with Pr(missed 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 $\Pr($hash collision$)$ with $\Pr($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 general hard deadline case?
Summary of reliability for *symmetric* channels

<table>
<thead>
<tr>
<th></th>
<th>Hard Deadlines</th>
<th>Soft Deadlines (undetected errors only)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Bound</td>
<td>Achievable</td>
</tr>
<tr>
<td>Block</td>
<td>Sphere-packing</td>
<td>Yes</td>
</tr>
<tr>
<td>No FB</td>
<td>Sphere-packing</td>
<td>Yes</td>
</tr>
<tr>
<td>Delay</td>
<td>Focusing*</td>
<td>Partial*</td>
</tr>
<tr>
<td>No FB</td>
<td>Sphere-packing</td>
<td>Yes</td>
</tr>
</tbody>
</table>

* entries are our contributions with **bold** for those in this talk.

* Trivial robustness is when feedback is not used at all.

* Partial achievability of the focusing bound is:
  - Tight for erasure channels and all DMCs with $C_{0,f} > 0$
  - Robust for packet erasure channels with erasure feedback
  - Better than sphere-packing bound for essentially all channels
### Summary of reliability for symmetric channels

<table>
<thead>
<tr>
<th></th>
<th>Hard Deadlines</th>
<th>Soft Deadlines (undetected errors only)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Bound</td>
<td>Achievable</td>
</tr>
<tr>
<td><strong>Block</strong></td>
<td>Sphere-packing</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>No FB</strong></td>
<td>Sphere-packing</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Delay</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>No FB</strong></td>
<td>Sphere-packing</td>
<td>Yes</td>
</tr>
</tbody>
</table>

- * entries are our contributions with **bold** for those in this talk.
- Trivial robustness is when feedback is not used at all.
- Partial achievability of the focusing bound is:
  - Tight for erasure channels and all DMCs with $C_{0,f} > 0$
  - Robust for packet erasure channels with erasure feedback
  - Better than sphere-packing bound for essentially all channels

For asymmetric channels, Haroutunian vs sphere-packing style gaps exist between bounds and achievable codes everywhere except **Block, No-FB**.
**Summary of reliability for symmetric channels**

<table>
<thead>
<tr>
<th></th>
<th><strong>Hard Deadlines</strong></th>
<th></th>
<th><strong>Soft Deadlines</strong> (undetected errors only)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Boundary</td>
<td>Achievable</td>
<td>Robust</td>
<td>Boundary</td>
</tr>
<tr>
<td><strong>Block</strong></td>
<td>Sphere-packing</td>
<td>Yes</td>
<td>trivial</td>
<td>Burnashev</td>
</tr>
<tr>
<td><strong>No FB</strong></td>
<td>Sphere-packing</td>
<td>Yes</td>
<td>trivial</td>
<td>Telatar</td>
</tr>
<tr>
<td><strong>Delay</strong></td>
<td>Focusing*</td>
<td>Partial*</td>
<td>Partial*</td>
<td>Hallucination*</td>
</tr>
<tr>
<td><strong>No FB</strong></td>
<td>Sphere-packing</td>
<td>Yes</td>
<td>trivial</td>
<td>Unknown</td>
</tr>
</tbody>
</table>

- * entries are our contributions with **bold** for those in this talk.
- Trivial robustness is when feedback is not used at all.
- Partial achievability of the focusing bound is:
  - Tight for erasure channels and all DMCs with $C_{0,f} > 0$
  - Robust for packet erasure channels with erasure feedback
  - Better than sphere-packing bound for essentially all channels
- For asymmetric channels, Haroutunian vs sphere-packing style gaps exist between bounds and achievable codes everywhere except **Block, No-FB**.
- Ultimately, there should be a continuum between perfect feedback and no feedback, as well as between hard and soft deadlines.