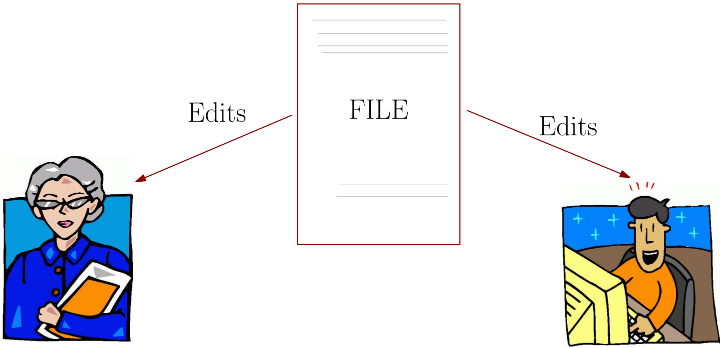
Interactive Codes for Synchronization

Ramji Venkataraman

Yale University

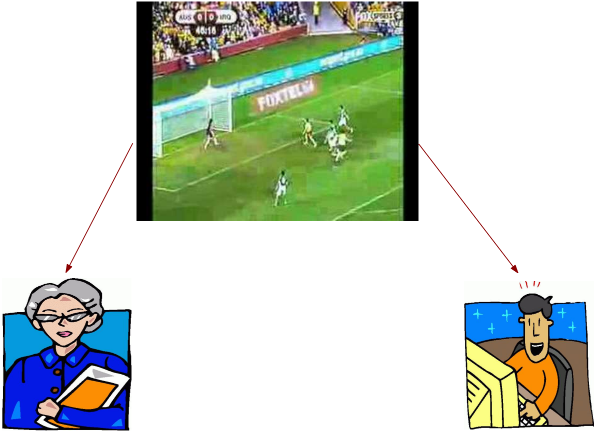
Joint work with H. Zhang, K. Ramchandran and S. Tatikonda

File Synchronization



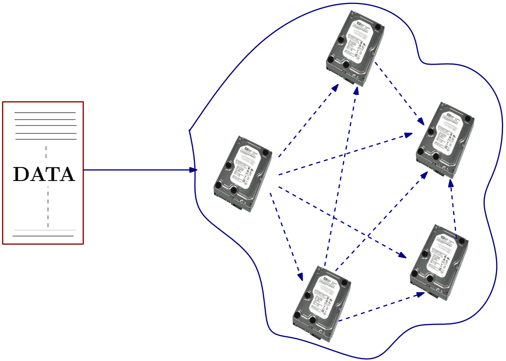
- Alice and Bob edit document separately
  - Modify some portions, Delete some portions, Insert new info
- Bob wants Alice's updated version
- RSYNC: Unix utility for exact bit-level synchronization

Video Synchronization



- Alice and Bob want their versions to *roughly* match
  - Within prescribed level of distortion and resolution
- VSync [Zhang et al '08]: Distance-aware hashing

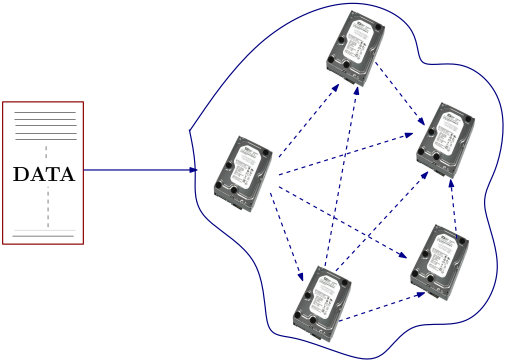
Uploading Data to the Cloud



DNA Sequencing Caught in Deluge of Data - NYT, Nov.2011

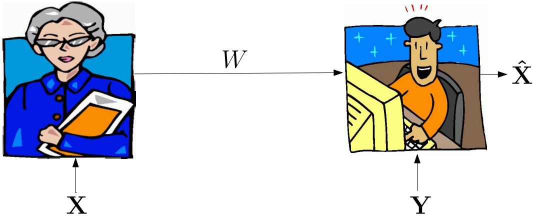
> *. . . there is now so much raw data that it is becoming not feasible to re-analyze it. In the case of human genomes, they might store even less – only the difference between a particular genome and some reference genome.*

Uploading Data to the Cloud



Incremental Upload: *Dropbox, HP JumboStore*

Model



- Binary length-n string $\mathbf{Y}$ is edited version of $\mathbf{X}$
- - Insertions and deletions of bits or small bursts of bits

  - includes block transpositions

$$\begin{cases} \mathbf{X} = \ldots abr aca\ dabradum\ dumdum\ ababab \ldots . \\ \mathbf{Y} = \ldots abacad\ dabradum\ ababab \ldots , \end{cases}$$

- Update $\mathbf{Y}$ so that it exactly or approximately equals $\mathbf{X}$

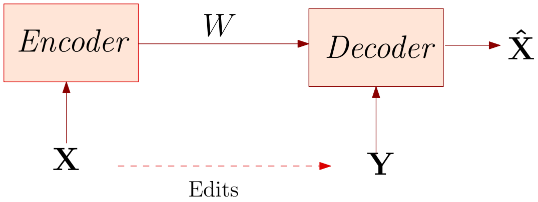
  - with minimal communication rate (information exchange)

In this talk . . .

Two Cases

- Small number of edits: $o(n)$
  - Optimal rate
  - Computationally efficient codes
- Large number of edits: $\sim \alpha n$
  - Bounds on optimal rate

Info-theoretic Model



'Source coding with side-information'

Lower bound (for zero-error code)

If **X** *knew* the locations of the s edits, rate at least $\frac{\log_2 \binom{n}{s}}{n}$

$$\approx \frac{s \log_2 n}{n} \quad \text{for} \quad s = \log n$$

$$\approx \frac{0.5 \;\, s \log_2 n}{n} \quad \text{for} \quad s = \sqrt{n}$$

Closely related problem

GOAL: Communicate over an edit channel

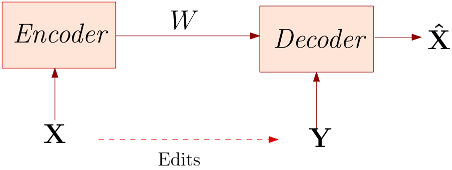
Levenshtein '65

To correct s insertions + deletions in large block of length n

$$1-\frac{2s\log_2 n/s}{n} \lesssim \text{ Rate of optimal \textit{zero-error} code } \lesssim 1-\frac{s\log_2 n/s}{n}$$

Fundamental limit

Synchronize from s insertions + deletions:



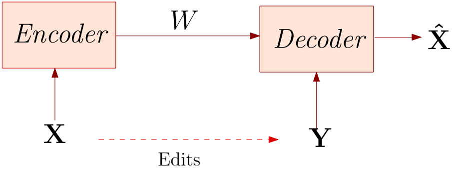
Lev's result + [Orlitsky-Viswanathan '03] $\Rightarrow$

- For large n, there exists *zero-error* code of rate $\sim \frac{1}{n} 2s \log_2 n/s$

- Near-optimal!

Fundamental limit

Synchronize from s insertions + deletions:



Lev's result + [Orlitsky-Viswanathan '03] $\Rightarrow$

- For large n, there exists *zero-error* code of rate $\sim \frac{1}{n} 2s \log_2 n/s$

- Near-optimal!

- How to find the code ? (Exhaustive search)
- Encoding and Decoding ? (Prohibitively complex)

Still open for $s > 1$

In this talk ...

Computationally efficient codes at near-optimal rate:

- Prob. error $\to 0$ as $n \to \infty$
- By allowing a small amount of *interaction*

$$\text{Rate} = \text{Rate}(\text{encoder} \to \text{decoder}) + \text{Rate}(\text{decoder} \to \text{encoder})$$

Interaction measured by:

- Rate from decoder $\to$ encoder
- Number of rounds

Related Work

- Rsync [Trigdell, Mackaras '98]
- VSync [Zhang, Ramchandran '08]
- Comm. complexity of synchronization: [Cormode et al '00]
- String Reconciliation: [Trachtenberg et al '06]

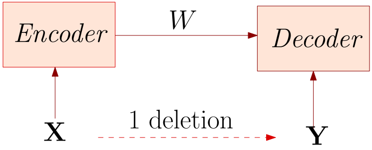
Coding and Capacity for deletion/insertion channels:

- Gallager ['61]
- Dobrushin ['67]
- Mackay et al ['01]
- Schulman & Zuckerman ['02]
- Diggavi & Grossglauser ['06]
- Mitzenmacher & others ['06-'10]

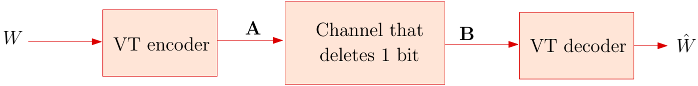
$\vdots$

Correcting 1 deletion

What we want:



Closely related problem:



Optimal channel code to correct 1- deletion: VT code

Varshamov-Tenengolts Codes

Position sum

$$\mathbf{x} = (x_1, \ldots, x_n)$$

- Define $P(\mathbf{x})$ = sum of positions of ones
- Example: $\mathbf{x} = (1, 0, 1, 1) \Rightarrow P(\mathbf{x}) = 1 + 3 + 4 = 8$

Varshamov-Tenengolts Codes

Position sum

$$\mathbf{x} = (x_1, \ldots, x_n)$$

- Define $P(\mathbf{x})$ = sum of positions of ones
- Example: $\mathbf{x} = (1, 0, 1, 1) \Rightarrow P(\mathbf{x}) = 1 + 3 + 4 = 8$

Code

- $VT(n)$ - code of block length n
  - All $\mathbf{x}$ such that $P(\mathbf{x}) \bmod (n+1) \equiv 0$
- $VT(4) = \{0000, 1001, 0110, 1111\}$ – Rate $\frac{\log_2 4}{4} = 0.5$

VT Decoder

Decoder [Levenshtein '65] computes

1. Weight W of received word
2. Position sum of received word

   Compute *deficiency* D in the position sum

VT Decoder

Decoder [Levenshtein '65] computes

1. Weight W of received word
2. Position sum of received word

   Compute *deficiency* D in the position sum

$D \leq W \Rightarrow 0$ was deleted

$$1\,0\,0\,1 \;\Rightarrow\; 1\,0\,1 \;\Rightarrow\; (W = 2, D = 1)$$

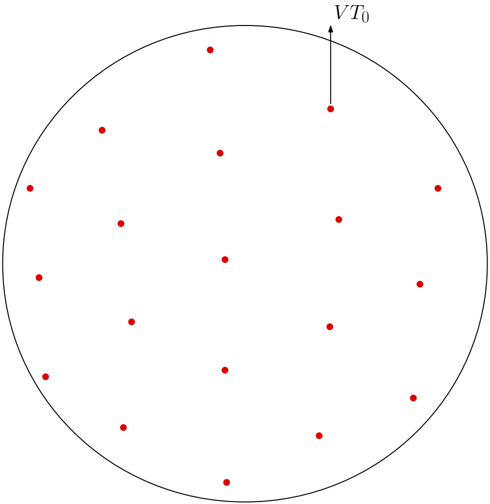
Insert 0 after D ones from the right $\Rightarrow\; 1\,0\,0\,1$

$D > W \Rightarrow 1$ was deleted

$$1\,0\,0\,1 \;\Rightarrow\; 1\,0\,0 \;\Rightarrow\; (W = 1, D = 4)$$

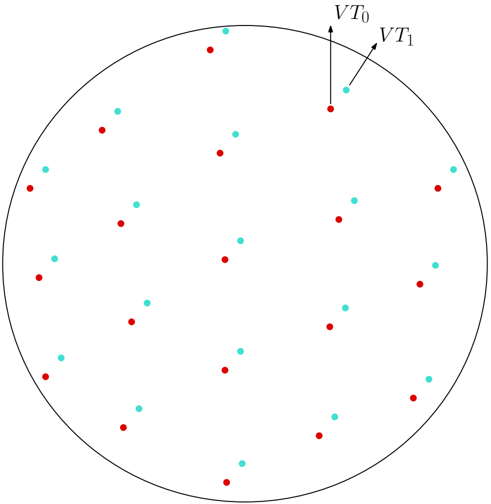
Insert 1 after $(D - W - 1)$ zeros from the left $\Rightarrow\; 1\,0\,0\,1$

VT Partition



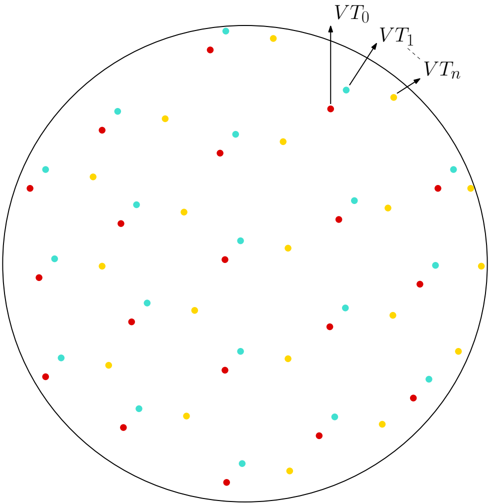
$VT_0(n) =$ All $\mathbf{x}$ such that $P(\mathbf{x}) \bmod (n+1) \equiv 0$

VT Partition



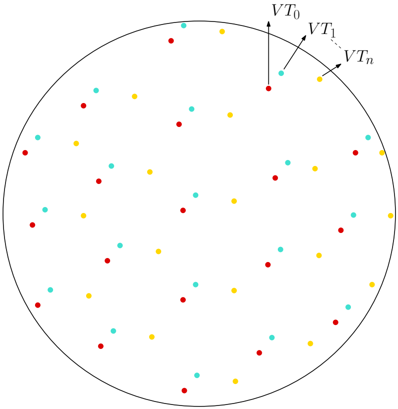
$VT_1(n) =$ All $\mathbf{x}$ such that $P(\mathbf{x}) \bmod (n+1) \equiv 1$

VT Partition



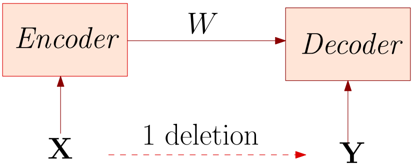
$VT_{a}(n)$ = All $\mathbf{x}$ such that $P(\mathbf{x}) \bmod (n+1) \equiv a, \quad a = 0, \ldots, n$

VT Cosets



- $\{VT_a(n)\}$ partition the space into 'cosets'
- Any $\{VT_a(n)\}$ can be used to correct 1-deletion!
- Each coset has $\sim \frac{2^n}{n}$ points $\Rightarrow$ Rate of $VT_a(n) \sim 1 - \frac{\log_2 n}{n}$
- Optimal 1-deletion correcting codes (non-linear)

Synchronizing from 1 deletion



- $P(\mathbf{X}) \bmod (n+1) \equiv a \quad \Leftrightarrow \quad \mathbf{X}$ is in VT_a
  - Encoder sends $a \in \{0, \ldots, n\}$ VT syndrome
- Decode **Y** to a codeword in VT_a

[Orlitsky '93]

- $\log_2(n+1)$ bits to correct one deletion - Optimal!
- No interaction - One-way
- Similar algorithm can correct one insertion

Burst Deletion

$$\mathbf{X} \qquad\qquad\qquad \mathbf{Y}$$

$$0\,0\,1\,0\,1\,1\,0\,1\,1\,1\,0\,1 \longrightarrow 0\,0\,1\,0\,1\,1\,1\,0\,1$$

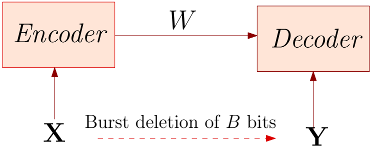
- Split X into 3 bit-planes:

$$0\;0\;1\;0\;\boxed{1\;1\;0}\;1\;1\;1\;0\;1$$

$$\mathbf{0\,0\,0\,1} \longrightarrow \mathbf{0\,0\,1}$$
$$\mathbf{0\,1\,1\,0} \longrightarrow \mathbf{0\,1\,0}$$
$$\mathbf{1\,1\,1\,1} \longrightarrow \mathbf{0\,0\,1}$$

- Exactly one deletion in each bit-plane

Correcting a burst



- Send VT-syndrome for each bit-plane:

$$\mathbf{0\,0\,0\,1} \xrightarrow{a_1} \mathbf{0\,0\,1} \text{ decoded to } VT_{a_1}$$

$$\mathbf{0\,1\,1\,0} \xrightarrow{a_2} \mathbf{0\,1\,0} \text{ decoded to } VT_{a_2}$$

$$\mathbf{1\,1\,1\,1} \xrightarrow{a_3} \mathbf{0\,0\,1} \text{ decoded to } VT_{a_3}$$

- Decoder reconstructs bit-planes & reassembles to recover $\mathbf{X}$

Performance

Number of bits $= B \log_2(1 + \frac{n}{B})$

Genie lower bound $= B + \log_2 n$

Multiple Deletions

- VT syndrome to synchronize single deletion
- 2 or more deletions?

$$\mathbf{X} \qquad\qquad\qquad \mathbf{Y}$$

$$0\textcolor{red}{0}100101011\textcolor{red}{1}1110\textcolor{red}{1} \longrightarrow 0100101010110$$

length n length $n-3$

- Channel codes for even *two* deletions known only for small n

What if we allow some interaction ?

Synchronization Algorithm

1. Encoder sends a few bits around center of **X**

$$1\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ \boxed{1\ 0\ 1\ 0\ 1}\ 1\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 1$$

Position $\frac{n}{2}$

Decoder matches these bits as close as possible to center of **Y**

$$1\ 1\ 0\ 0\ 1\ 0\ 0\ \boxed{1\ 0\ 1\ 0\ 1}\ 1\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 1$$

Position $\frac{n}{2} - 1$

Synchronization Algorithm

1. Encoder sends a few bits around center of **X**

$$1\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ \boxed{1\ 0\ 1\ 0\ 1}\ 1\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 1$$

$$\text{Position } \tfrac{n}{2}$$

Decoder matches these bits as close as possible to center of **Y**

$$1\ 1\ 0\ 0\ 1\ 0\ 0\ \boxed{1\ 0\ 1\ 0\ 1}\ 1\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 1$$

$$\text{Position } \tfrac{n}{2} - 1$$

2. Offset $\Rightarrow$ decoder knows one deletion in left half, two in right

3. Encoder sends VT syndrome a of left half of **X**

   - Decoder synchronizes left half of **Y** by decoding to $VT_a(\frac{n}{2})$

Next stage

- Send a few bits around its center of right piece of **X**

$$\ldots\ldots|\,0\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 1$$

Position $\frac{3n}{4}$

Decoder tries to match these bits ...

$$\ldots\ldots|\,0\ 1\ 1\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 1$$

Position $\frac{3n}{4} - 1$

Next stage

- Send a few bits around its center of right piece of **X**

$$\ldots\ldots|\,0\ 1\ 1\ \color{red}{0}\ 1\ \colorbox{yellow}{1 0 0}\ 1\ 1\ 0\ \color{red}{1}$$

Position $\frac{3n}{4}$

Decoder tries to match these bits ...

$$\ldots\ldots|\,0\ 1\ 1\ 1\ \colorbox{yellow}{1 0 0}\ 1\ 1\ 0\ 1$$

Position $\frac{3n}{4} - 1$

- One deletion in left half, and one in right half
- Send VT syndromes a_1, a_2 of the two halves
  - Decode left half to $VT_{a_1}(n/4)$, right half to $VT_{a_2}(n/4)$

Multiple Deletions & Insertions

0

X: 1 1 0 0 0 1 0 0 1 0 1 0 1 1 0 1 1 0 0 1 1 0 ~~1~~

Position $\frac{n}{2}$

Y: 1 1 0 0 0 1 0 0 1 0 1 0 1 1 0 1 0 1 0 0 1 1 0

Position $\frac{n}{2}$

No offset $\Rightarrow$ cannot detect insertion + del in same half

Guess-and-Check

0

X: 1 1 0 0 0 1 0 0 1 0 1 0 1 1 0 1 1 0 0 1 1 0 ~~1~~

Position $\frac{n}{2}$

Y: 1 1 0 0 0 1 0 0 1 0 1 0 1 1 0 1 0 1 0 0 1 1 0

Position $\frac{n}{2}$

- Hash left halves of **X** and **Y** - hashes agree
- Hash right halves of **X** and **Y** - hashes disagree

Synchronization Algorithm

Work on right half:

X: . . . | 1 1 0 1 1 0 0 1 1 0 1

Position $\frac{3n}{4}$

Y: . . . | 1 1 0 1 0 1 0 0 1 1 0

Position $\frac{3n}{4} + 1$

- Detect an offset of 1 to the right
  - One *net* insertion on the left, one *net* deletion on the right
- Exchange VT syndromes for each of these parts
- Check hash to confirm match

Algorithm for Insertions + Deletions

In each round . . .

- *Encoder*: Send center-bits for unsynchronized pieces
- *Decoder*: For each unsynchronized piece, Align centers
  - If offset is 0, request hashes
  - If offset is 1, request VT syndrome + hashes
  - If offset for either half is > 1, request center-bits
- Continue until all pieces are synchronized

Performance

Theorem (RV-Zhang-Ramchandran, Allerton '10)

- *s insertions + deletions in random locations ($s \sim o(n)$)*
- *Number of center-bits = number of hash-bits = $c \log_2 n$*

(a) *The probability of error* $< \frac{s \log n}{n^c}$

(b)

$$ER_{1\to 2}(s) < (4c+1)\,\frac{s \log_2 n}{n},$$

$$ER_{2\to 1}(s) < 10\,\frac{(s-1)}{n}$$

(c) *The expected number of rounds is* $< 4 + 2\log_2 s$

Experiments

Config.		Rate $\mathbf{X} \to \mathbf{Y}$		Rate $\mathbf{Y} \to \mathbf{X}$		No. of rounds	
n	$d = i$	Sim.	Theo.	Sim.	Theo.	Sim.	Theo.
10^7	100	$2.1e^{-3}$	$2.1e^{-3}$	$1.6e^{-4}$	$2.0e^{-4}$	14.9	19.3
10^7	1000	$2.0e^{-2}$	$2.1e^{-2}$	$1.6e^{-3}$	$2.0e^{-3}$	21.3	25.9

No. of center bits = No. of hash bits = 20

(1000 random simulations for each case)

Burst edits?

1 1 0 0 0 1 0 0 1 0 1 0 1 1 0 1 1 0 0 1 1 0 1

Worst-case scenario for the algorithm:

- Tries to separate the deletions in the burst
- Number of rounds $\sim \log_2 n$ (avg. case $\log_2 s$)
- Rate $R_{1\to 2} \sim \frac{s(\log_2 n)^2}{n}$ (avg. case $\frac{s \log_2 n}{n}$)

  Rate $R_{2\to 1} \sim \frac{s \log_2 n}{n}$ (avg. case $\frac{s}{n}$)

But burst edits are common!

Adapting to bursts

$$1\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 1$$

If offset does not change after T rounds, hypothesize a burst:

- Send syndromes for bit-planes , correct *as if* it were a burst
- Use hashes to check
- If not continue splitting

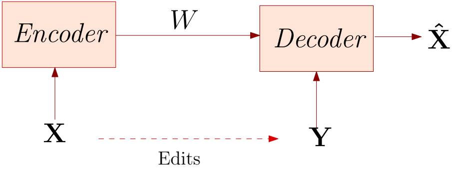
Summary

When the number of edits is small ...

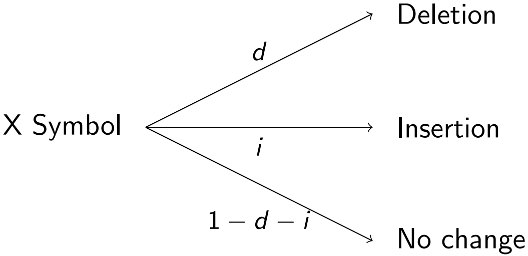
A small amount of interaction can help:

- Primitive- single deletion/insertion or single burst
- Isolate primitive pieces
- Use VT syndromes for primitive, check with hashes

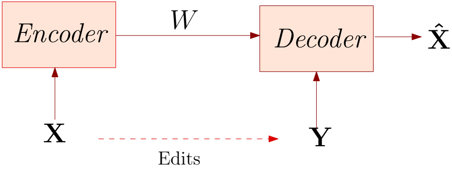
Large number of edits



IID edit model relating **X** and **Y**:



Large number of edits



Optimal Synchronization Rate R^*

Minimum rate of W such that $P(\hat{\mathbf{X}} \neq \mathbf{X}) \to 0$ as $n \to \infty$

- $R^* = \lim_{n\to\infty} \frac{1}{n} H(\mathbf{X}|\mathbf{Y}) \;\Rightarrow\;$ hard to compute.

$$\mathbf{X} = 0\ {\color{red}0\ 0\ 1\ 1\ 1}\ 0 \overset{\color{blue}1}{\color{blue}\downarrow}\ 0 \qquad \mathbf{Y} = 0\ 0\ {\color{blue}1}\ 0$$

- Bounds on R^* [ITA '11]
- Bounds on capacity of channels with deletions and insertions [ISIT '11]

Ideas for the future

- Hard restriction on number of rounds
  - Higher rate algorithm that works with 1 round of interaction
- Synchronize from a few large bursts of length $\sim \alpha n$
- Use for video synchronization
  - Sync within targeted distortion: distance-aware hashing
  - Combine with outer error-correcting code
- To correct a large number of insertions + deletions
  - LDPC-like codes using VT-primitive?