Reducing Energy Usage Through a Novel File Synchronization Algorithm

Frederic Sala
LORIS Lab, UCLA

Joint work with:
Nicolas Bitouzé (UCLA)
Clayton Schoeny (UCLA),
S. M. Sadegh Tabatabaei Yazdi (Qualcomm),
Lara Dolecek (UCLA)

Laboratory for Robust Information Systems (LORIS)
Department of Electrical Engineering, UCLA

Motivation

Combined data center electricity usage is already at 1.5% of all electricity used in the world.

J. Koomey, "Growth in data center electricity use 2005 to 2010", 2011.

Motivation

Combined data center electricity usage is already at 1.5% of all electricity used in the world.

J. Koomey, "Growth in data center electricity use 2005 to 2010", 2011.

A major contributing factor: large data storage requirements. In part, these requirements are due to the unnecessary storage of superfluous data:

Motivation

Combined data center electricity usage is already at 1.5% of all electricity used in the world.

J. Koomey, "Growth in data center electricity use 2005 to 2010", 2011.

A major contributing factor: large data storage requirements. In part, these requirements are due to the unnecessary storage of superfluous data:

- Multiple copies of the same file.
- Multiple versions of a file.

Reducing Data Storage Demand

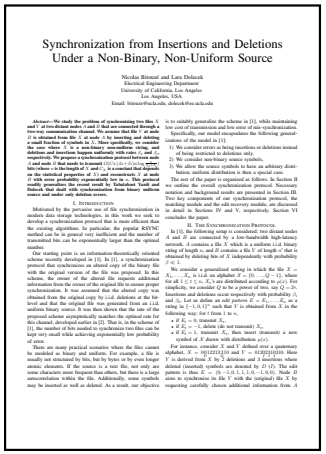
- When files are identical, we can use *deduplication* tools.

Reducing Data Storage Demand

- When files are identical, we can use *deduplication* tools.
- What if files are similar, but not identical?

Reducing Data Storage Demand

- When files are identical, we can use *deduplication* tools.
- What if files are similar, but not identical?

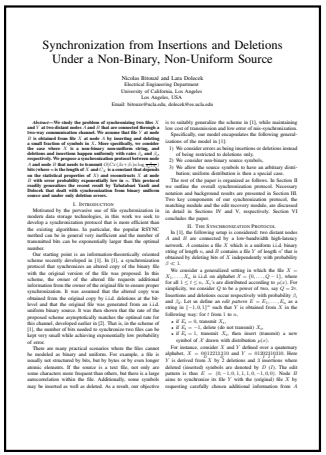


Synchronization from Insertions and Deletions
Under a Non-Binary, Non-Uniform Source

Nicolas Bitouzé and Lara Dolecek
Electrical Engineering Department
University of California, Los Angeles
Los Angeles, USA
Email: bitouze@ucla.edu, dolecek@ee.ucla.edu

Reducing Data Storage Demand

- When files are identical, we can use *deduplication* tools.
- What if files are similar, but not identical?



Synchronization from Insertions and Deletions
Under a Non-Binary, Non-Uniform Source

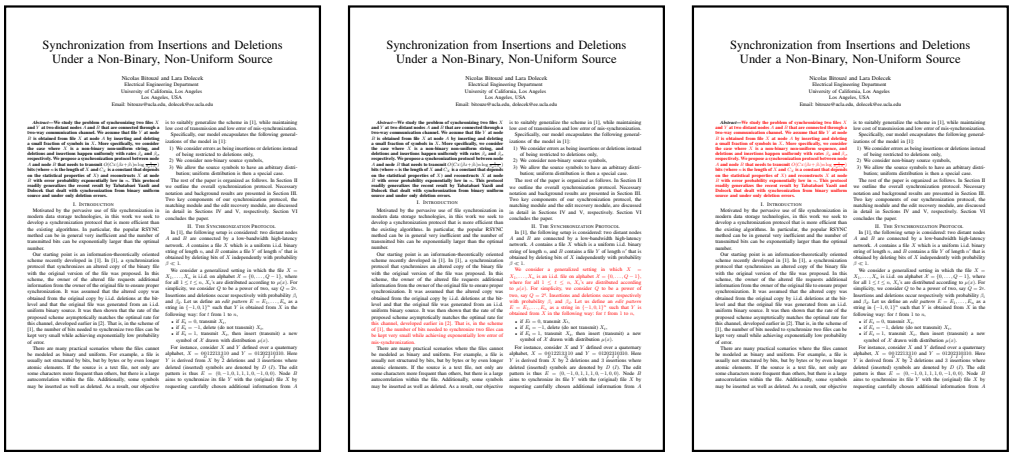
Nicolas Bitouzé and Lara Dolecek
Electrical Engineering Department
University of California, Los Angeles
Los Angeles, USA
Email: bitouze@ucla.edu, dolecek@ee.ucla.edu

Synchronization from Insertions and Deletions
Under a Non-Binary, Non-Uniform Source

Nicolas Bitouzé and Lara Dolecek
Electrical Engineering Department
University of California, Los Angeles
Los Angeles, USA
Email: bitouze@ucla.edu, dolecek@ee.ucla.edu

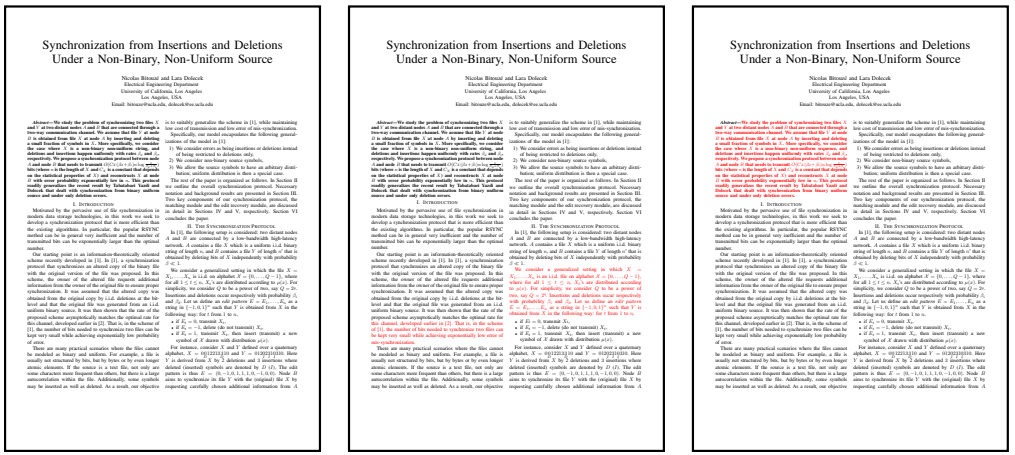
Reducing Data Storage Demand

- When files are identical, we can use *deduplication* tools.
- What if files are similar, but not identical?



Reducing Data Storage Demand

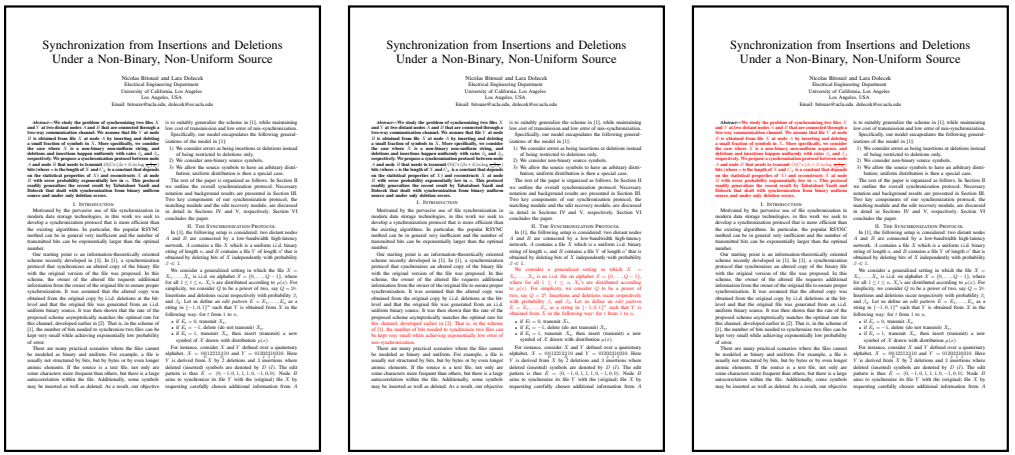
- When files are identical, we can use *deduplication* tools.
- What if files are similar, but not identical?



- We need algorithms to *synchronize* multiple versions of a file.
- Existing algorithms, such as RSYNC, suffer from high communication costs.

Reducing Data Storage Demand

- When files are identical, we can use *deduplication* tools.
- What if files are similar, but not identical?



- We need algorithms to *synchronize* multiple versions of a file.
- Existing algorithms, such as RSYNC, suffer from high communication costs.

- **Goal:** Develop a more efficient synchronization algorithm.

Synchronization Protocols

Original File

```
void ramji_sync_rec(int x_s, int x_l, int
{
  int d_l = delimiter_length;
  if (abs(x_l-y_l) <= 1) {
    stats->total_bits += control_bits;
    stats->total_bits += hash_length;
    bool sync_complete;
    if (x_l==y_l) {
      sync_complete = equal_substrings(x_
    } else {
      stats->total_bits += log2(q) + log2
      if (x_l > y_l) {
        sync_complete = exactly_one_delet
      } else {
        sync_complete = exactly_one_inser
      }
    }
    if (sync_complete) {
      stats->total_bits += control_bits;
      stats->total_rounds = max(stats->to
      return;
    } else {
      if (hashes_collide()) {
        stats->total_errors += x_l;
        stats->total_bits += control_bits
        stats->total_rounds = max(stats->
```

Synchronization Protocols

Original File

```
void ramji_sync_rec(int x_s, int x_l, int
{
  int d_l = delimiter_length;
  if (abs(x_l-y_l) <= 1) {
    stats->total_bits += control_bits;
    stats->total_bits += hash_length;
    bool sync_complete;
    if (x_l==y_l) {
      sync_complete = equal_substrings(x_
    } else {
      stats->total_bits += log2(q) + log2
      if (x_l > y_l) {
        sync_complete = exactly_one_delet
      } else {
        sync_complete = exactly_one_inser
      }
    }
    if (sync_complete) {
      stats->total_bits += control_bits;
      stats->total_rounds = max(stats->to
      return;
    } else {
      if (hashes_collide()) {
        stats->total_errors += x_l;
        stats->total_bits += control_bits
        stats->total_rounds = max(stats->
```

Edits

Alice's Version

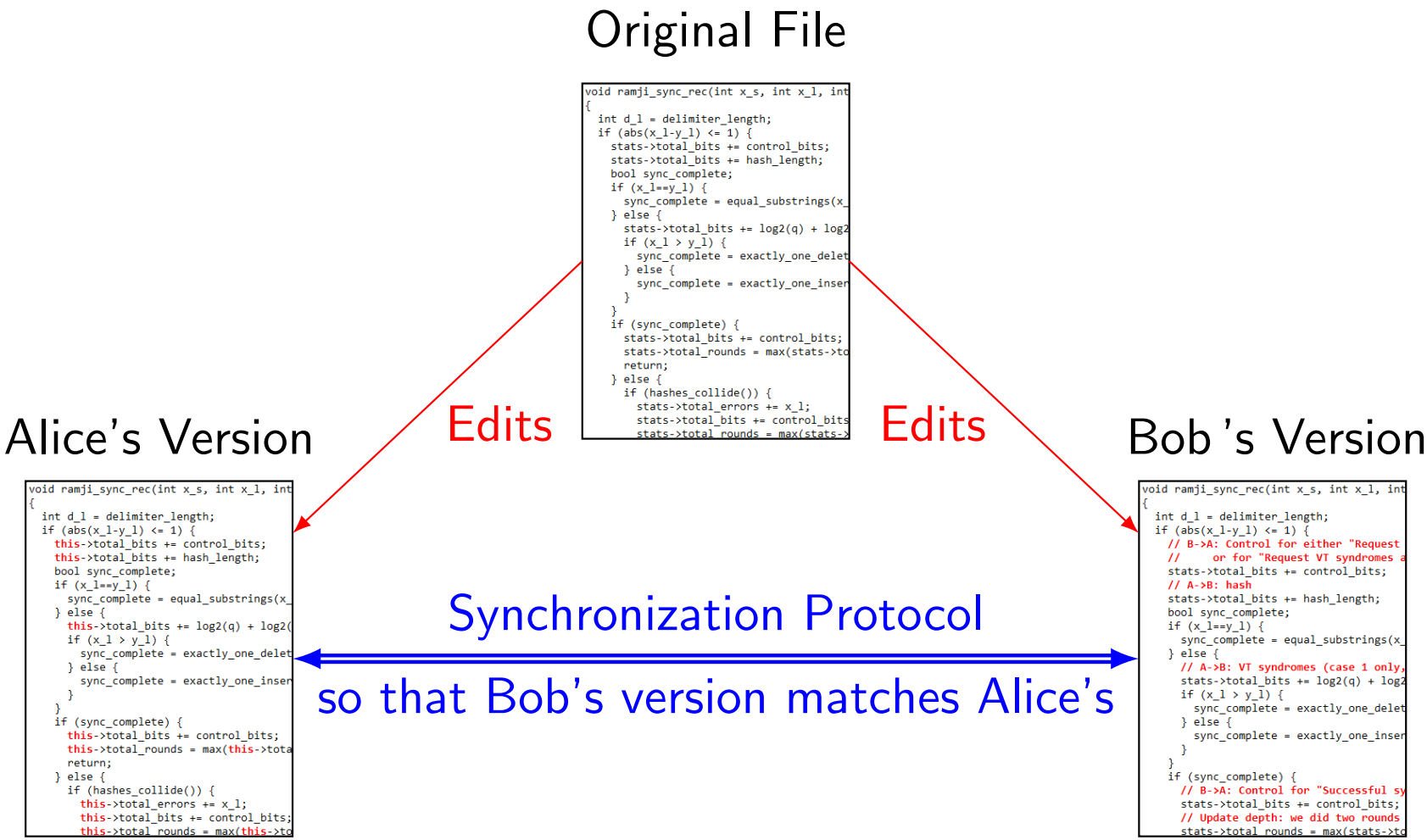
```
void ramji_sync_rec(int x_s, int x_l, int
{
  int d_l = delimiter_length;
  if (abs(x_l-y_l) <= 1) {
    this->total_bits += control_bits;
    this->total_bits += hash_length;
    bool sync_complete;
    if (x_l==y_l) {
      sync_complete = equal_substrings(x_
    } else {
      this->total_bits += log2(q) + log2(
      if (x_l > y_l) {
        sync_complete = exactly_one_delet
      } else {
        sync_complete = exactly_one_inser
      }
    }
    if (sync_complete) {
      this->total_bits += control_bits;
      this->total_rounds = max(this->tota
      return;
    } else {
      if (hashes_collide()) {
        this->total_errors += x_l;
        this->total_bits += control_bits;
        this->total_rounds = max(this->to
```

Edits

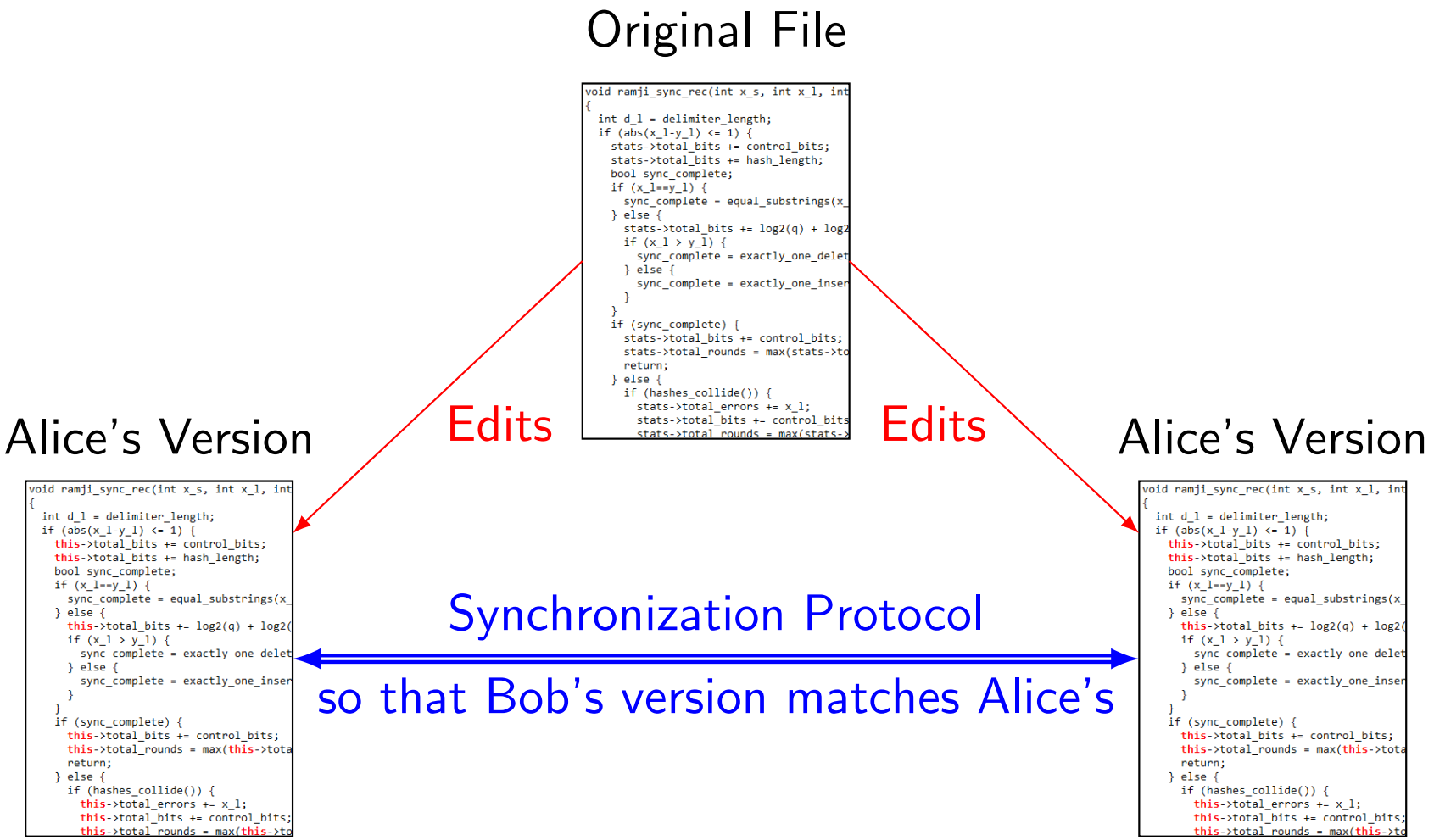
Bob's Version

```
void ramji_sync_rec(int x_s, int x_l, int
{
  int d_l = delimiter_length;
  if (abs(x_l-y_l) <= 1) {
    // B->A: Control for either "Request
    //      or for "Request VT syndromes a
    stats->total_bits += control_bits;
    // A->B: hash
    stats->total_bits += hash_length;
    bool sync_complete;
    if (x_l==y_l) {
      sync_complete = equal_substrings(x_
    } else {
      // A->B: VT syndromes (case 1 only,
      stats->total_bits += log2(q) + log2
      if (x_l > y_l) {
        sync_complete = exactly_one_delet
      } else {
        sync_complete = exactly_one_inser
      }
    }
    if (sync_complete) {
      // B->A: Control for "Successful sy
      stats->total_bits += control_bits;
      // Update depth: we did two rounds
      stats->total_rounds = max(stats->to
```

Synchronization Protocols



Synchronization Protocols



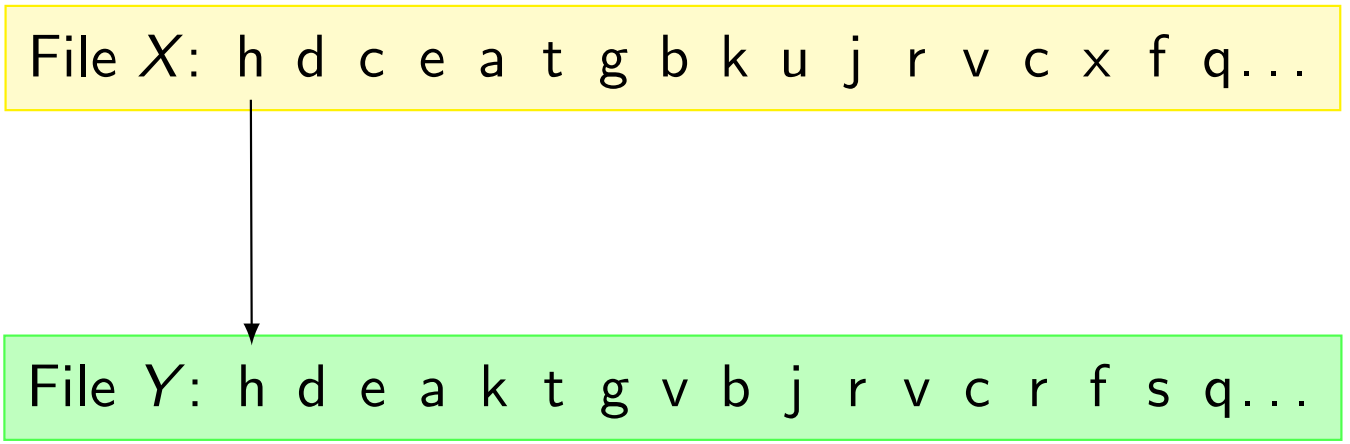
Problem Setting

File X: h d c e a t g b k u j r v c x f q...

File Y: h d e a k t g v b j r v c r f s q...

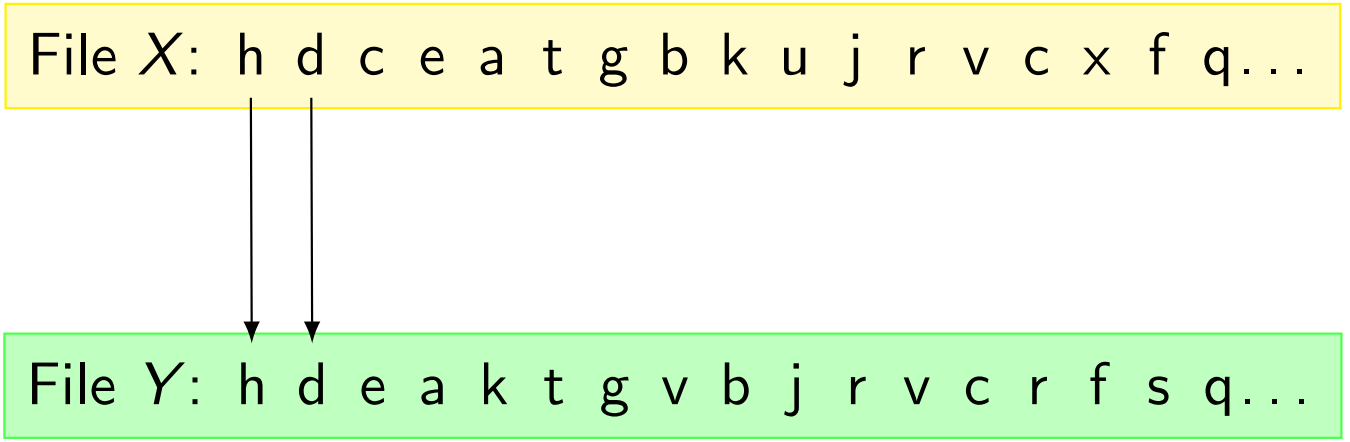
- Small rate of edits β.
- File length: $|X| = n$, $|Y| = m \approx n$.

Problem Setting



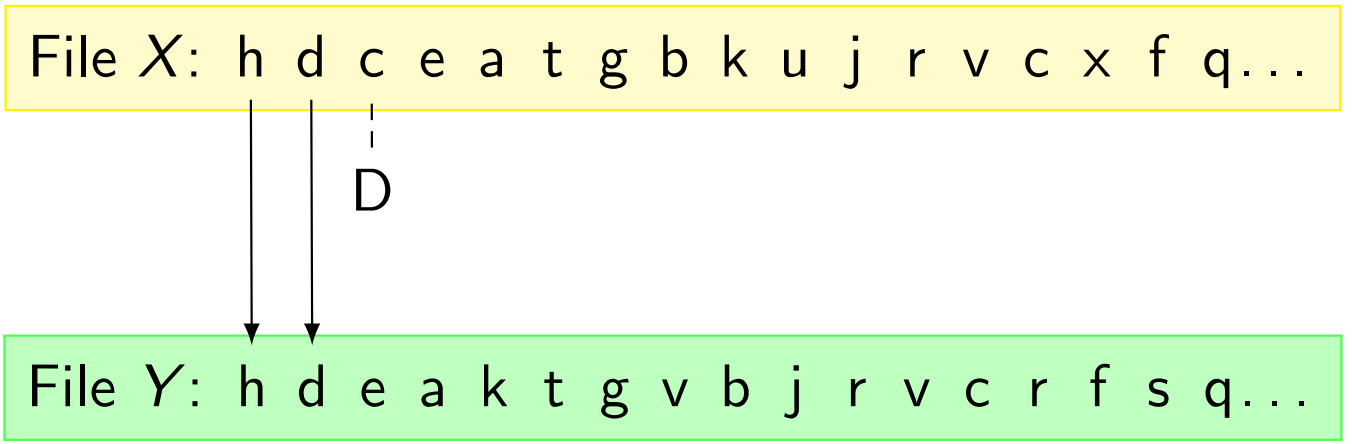
- Small rate of edits β.
- File length: $|X| = n$, $|Y| = m \approx n$.

Problem Setting



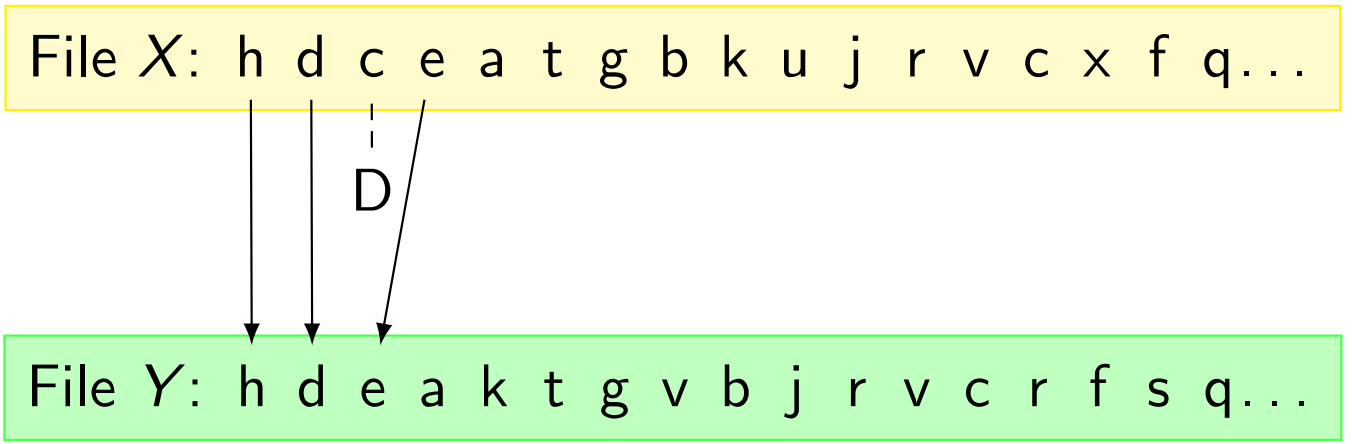
- Small rate of edits β.
- File length: $|X| = n$, $|Y| = m \approx n$.

Problem Setting



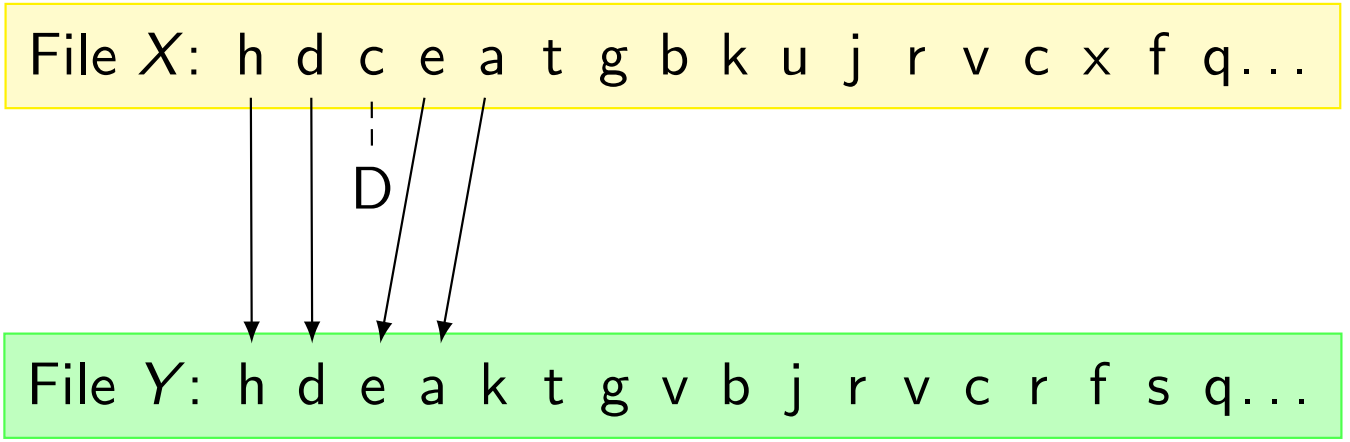
- Small rate of edits β.
- File length: $|X| = n$, $|Y| = m \approx n$.

Problem Setting



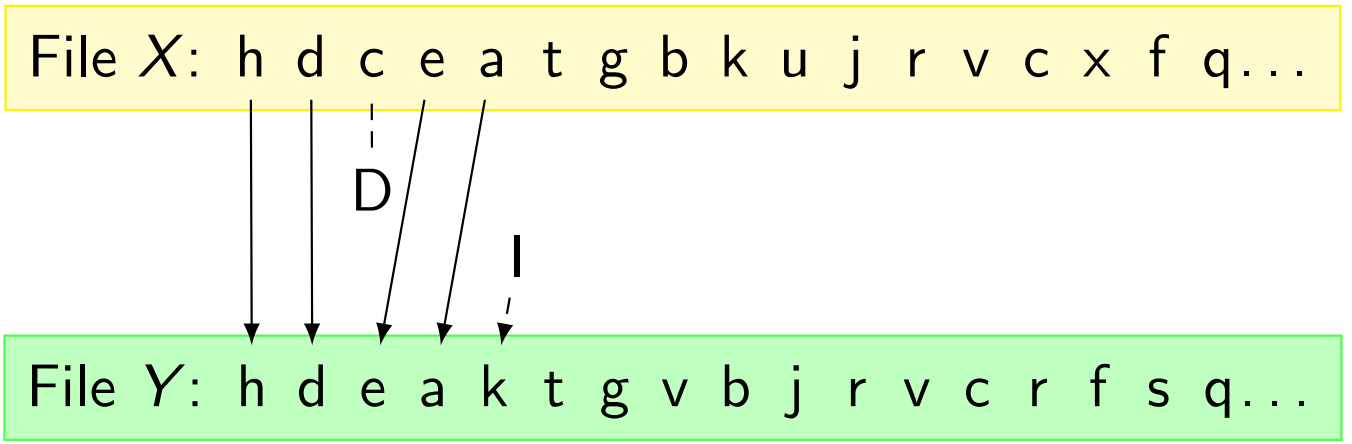
- Small rate of edits β.
- File length: $|X| = n$, $|Y| = m \approx n$.

Problem Setting



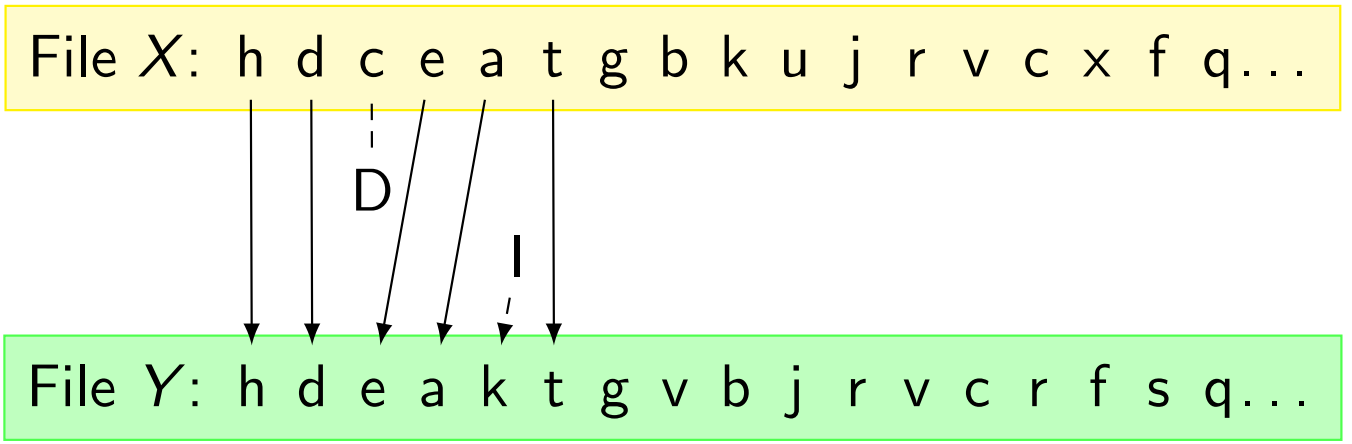
- Small rate of edits β.
- File length: $|X| = n$, $|Y| = m \approx n$.

Problem Setting



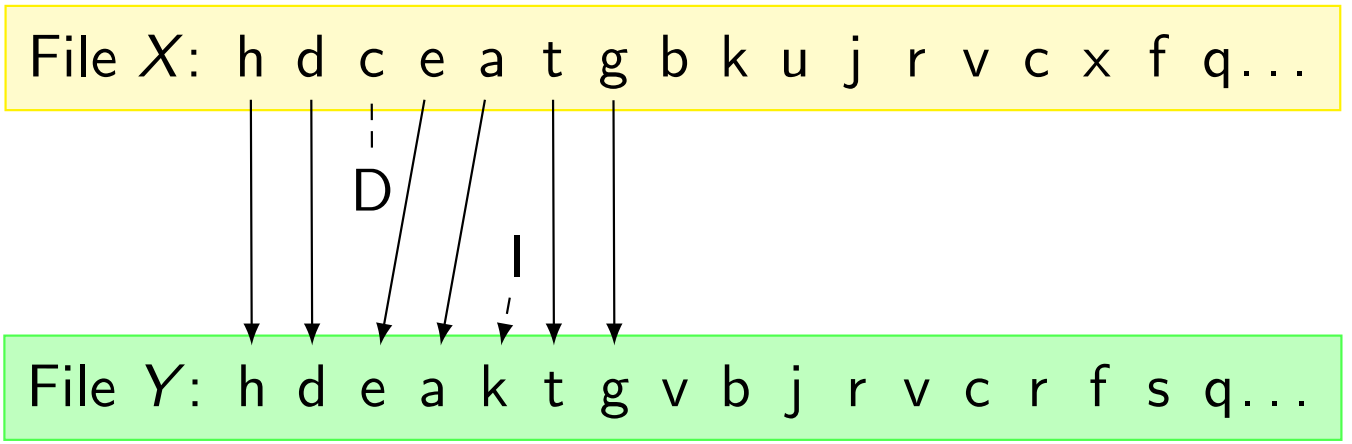
- Small rate of edits β.
- File length: $|X| = n$, $|Y| = m \approx n$.

Problem Setting



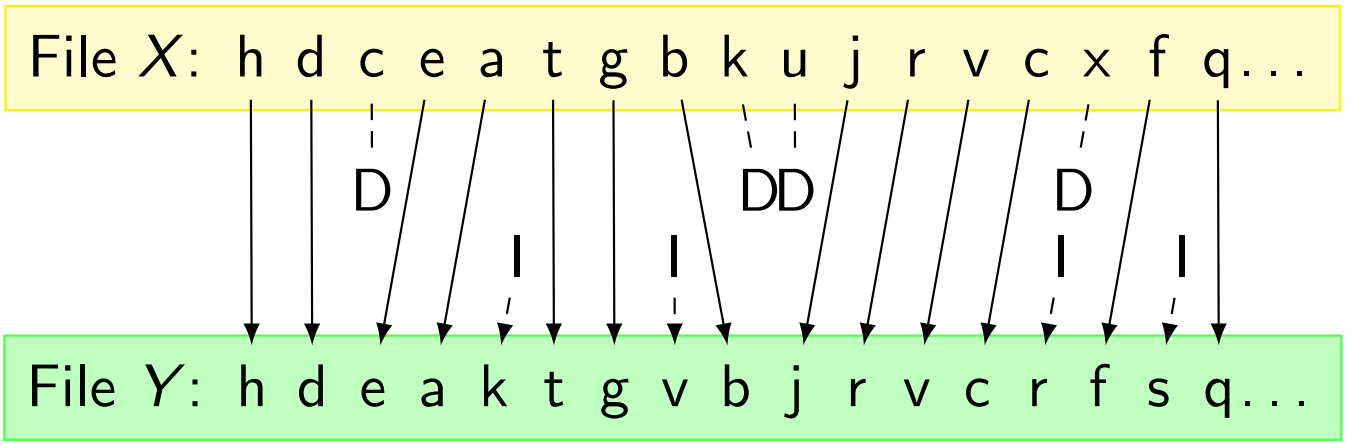
- Small rate of edits β.
- File length: $|X| = n$, $|Y| = m \approx n$.

Problem Setting



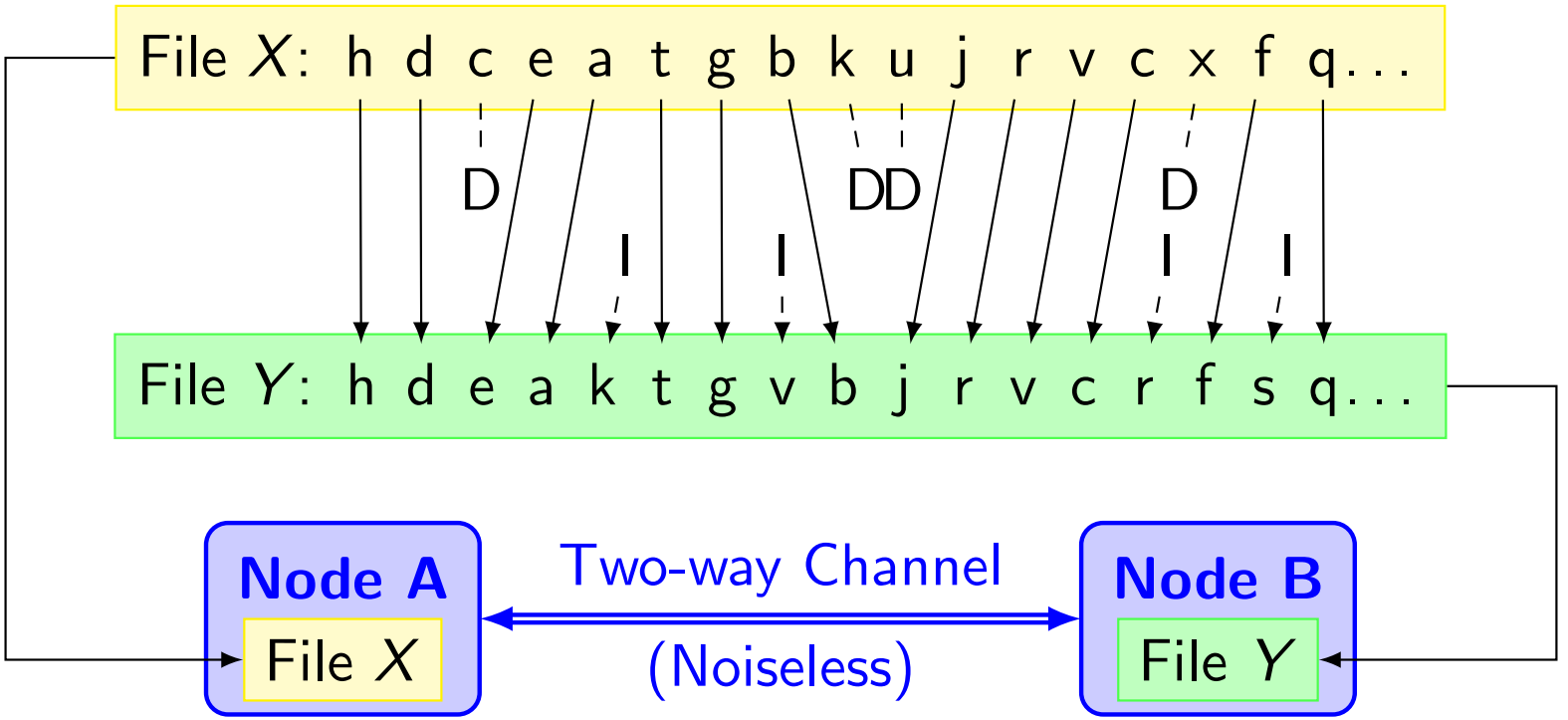
- Small rate of edits β.
- File length: $|X| = n$, $|Y| = m \approx n$.

Problem Setting



- Small rate of edits β.
- File length: $|X| = n$, $|Y| = m \approx n$.

Problem Setting



Goal: Interactive Communication Scheme

Allow Node to B recover X from Y:

- with low probability of error,
- with low communication cost.

Related Work

- Scheme that corrects a single edit (binary & non-binary):
  - V. I. Levenshtein, "Binary codes with correction of deletions, insertions and reversals", 1965.
  - G. M. Tenengolts, "Nonbinary codes, correcting single deletion or insertion", 1984.

Related Work

- Scheme that corrects a single edit (binary & non-binary):
  - V. I. Levenshtein, "Binary codes with correction of deletions, insertions and reversals", 1965.
  - G. M. Tenengolts, "Nonbinary codes, correcting single deletion or insertion", 1984.
- Scheme that corrects a fixed number of edits (binary):
  - R. Venkataramanan, H. Zhang, and K. Ramchandran, "Interactive low-complexity codes for synchronization from deletions and insertions", 2010.

Related Work

- Scheme that corrects a single edit (binary & non-binary):
  -  V. I. Levenshtein, "Binary codes with correction of deletions, insertions and reversals", 1965.
  -  G. M. Tenengolts, "Nonbinary codes, correcting single deletion or insertion", 1984.
- Scheme that corrects a fixed number of edits (binary):
  -  R. Venkataramanan, H. Zhang, and K. Ramchandran, "Interactive low-complexity codes for synchronization from deletions and insertions", 2010.
- Theoretical bound for the fixed rate of edits case:
  -  N. Ma, K. Ramchandran and D. Tse, "Efficient file synchronization: a distributed source coding approach", 2011.

Our Contributions

- Scheme for fixed rate of edits:
  - N. Bitouzé and L. Dolecek, "Synchronization from insertions and deletions under a non-binary, non-uniform source", IEEE ISIT, Jul. 2013.
  - S. M. S. Tabatabaei and L. Dolecek, "A deterministic, polynomial-time protocol for synchronizing from deletions", IEEE Trans. I.T., 2013.
  - N. Bitouzé, F. Sala, S. M. S. Tabatabaei, and L. Dolecek, "A practical framework for efficient file synchronization", Allerton, Oct. 2013.

Synchronization Scheme: Overview

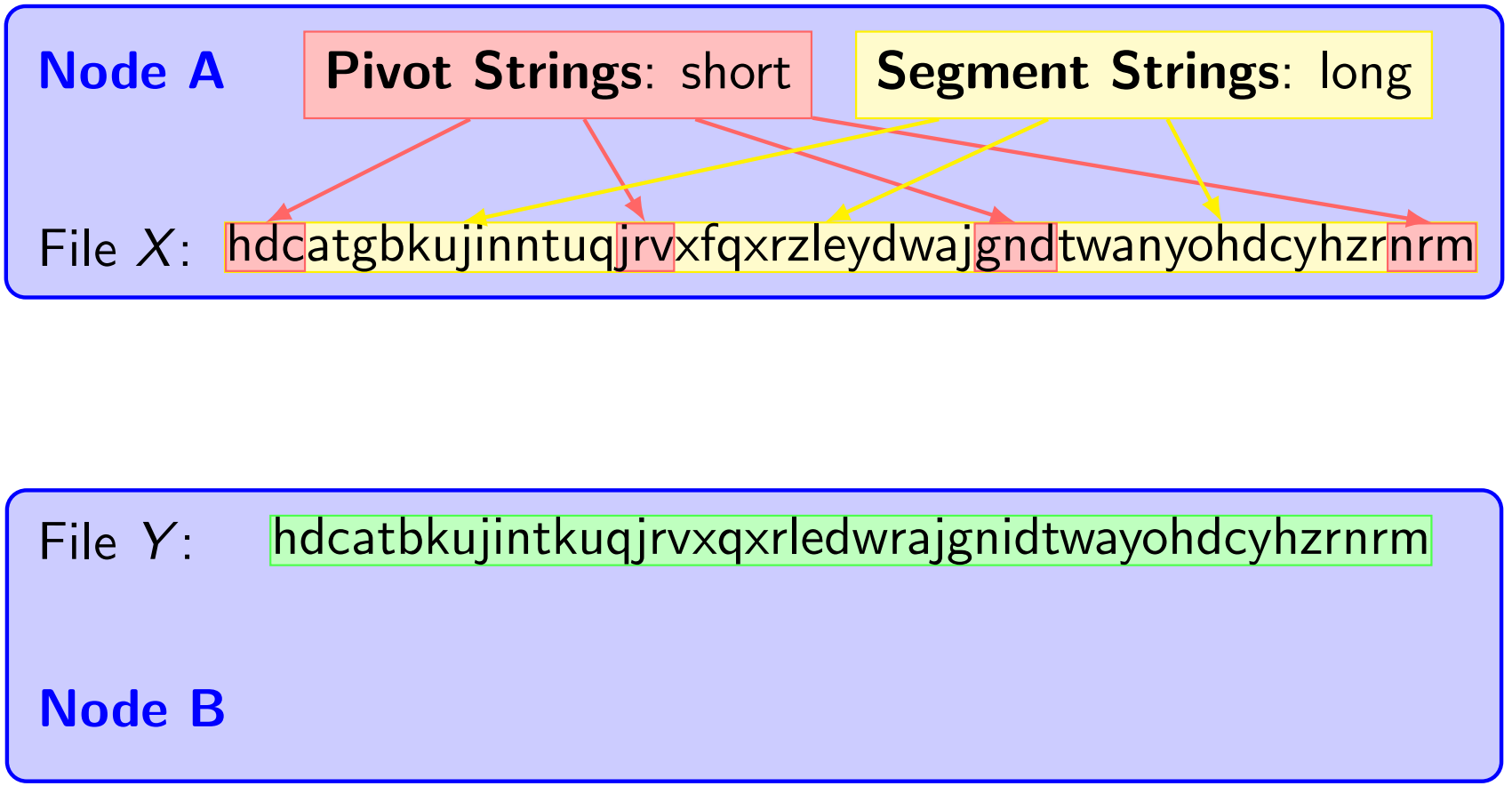
Node A

File X: hdcatgbkujinntuqjrvxfqxrzleydwajgndtwanyohdcyhzrnrm

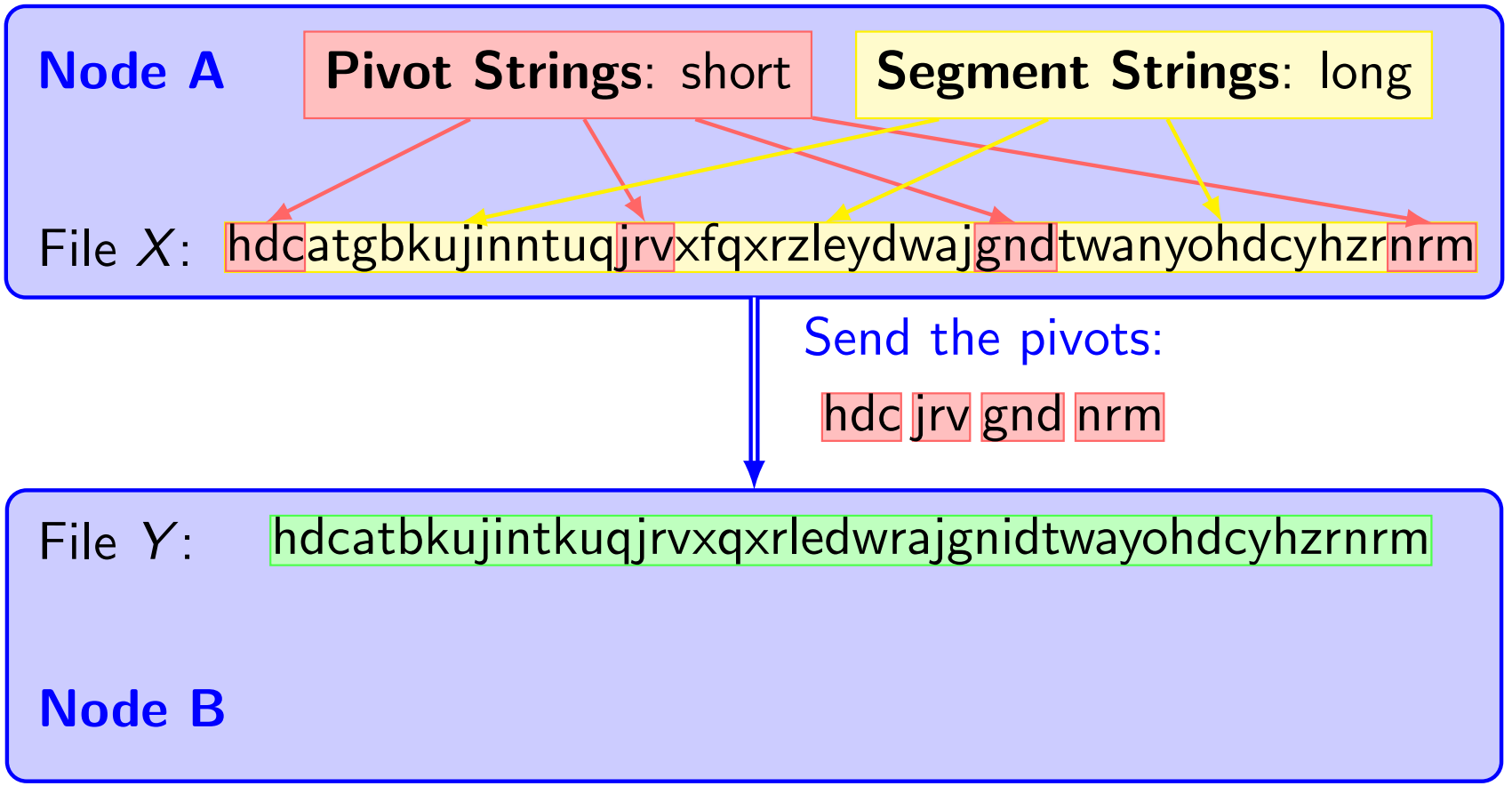
File Y: hdcatbkujintkuqjrvxqxrledwrajgnidtwayohdcyhzrnrm

Node B

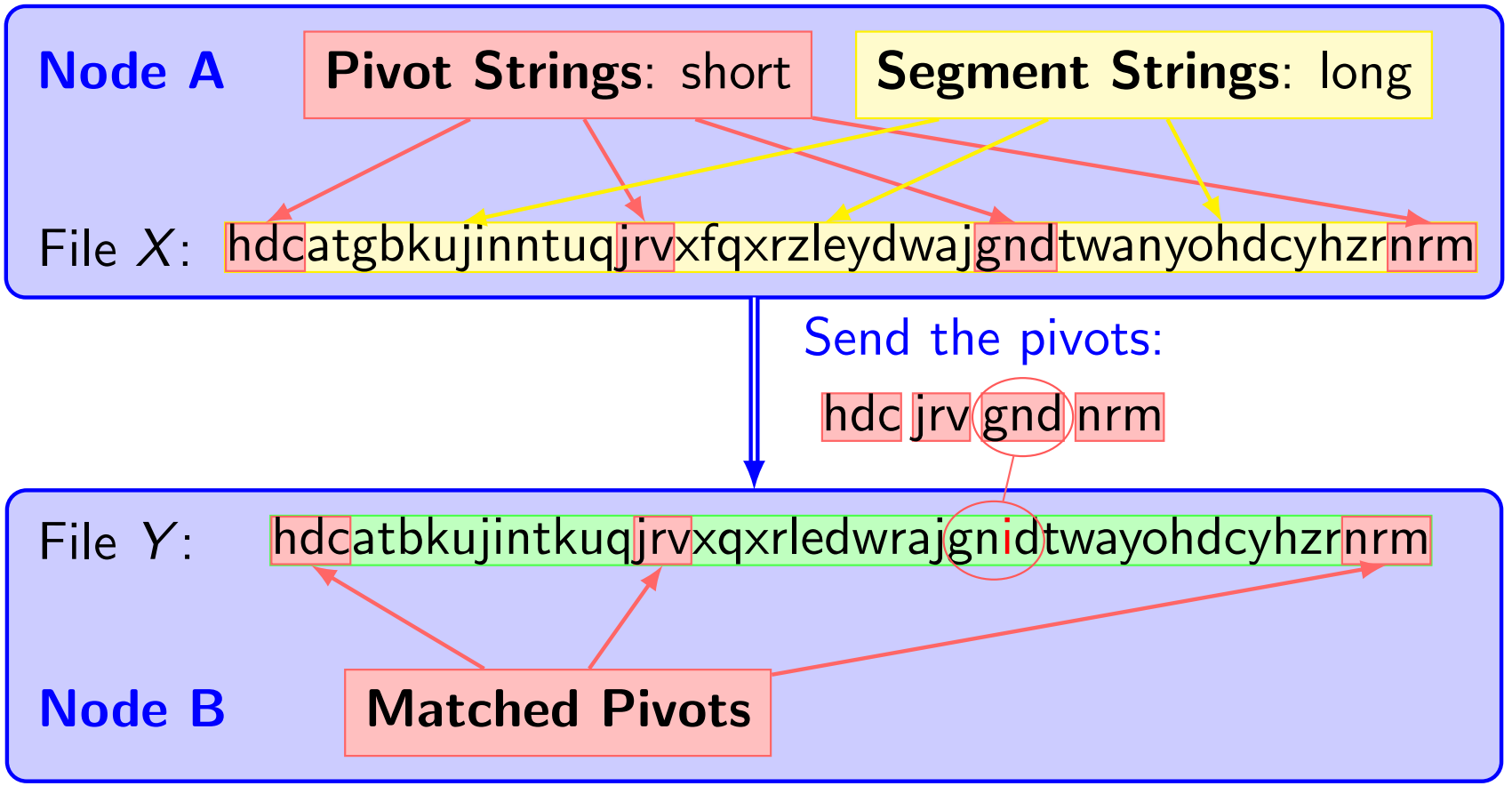
Synchronization Scheme: Overview



Synchronization Scheme: Overview

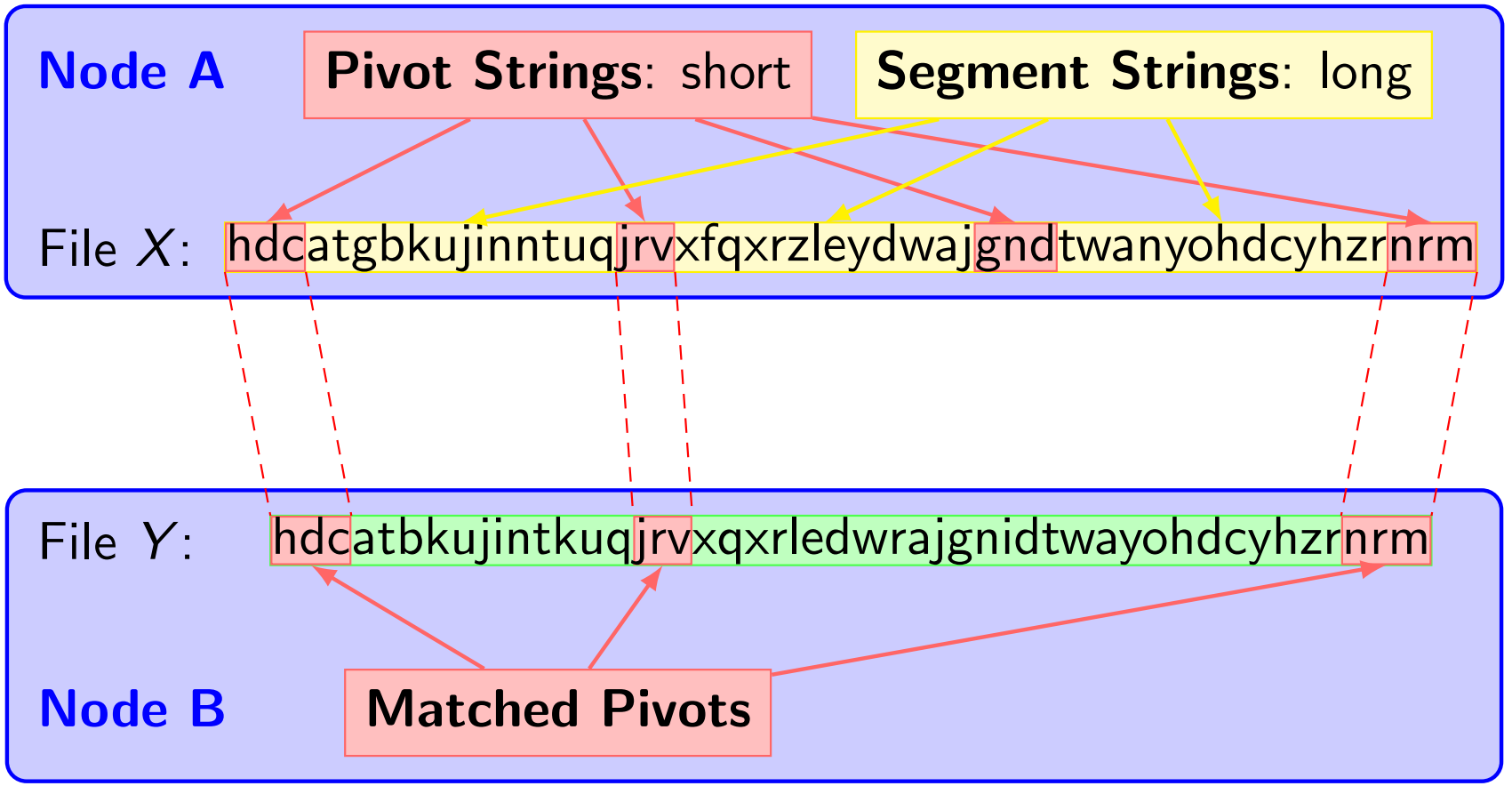


Synchronization Scheme: Overview



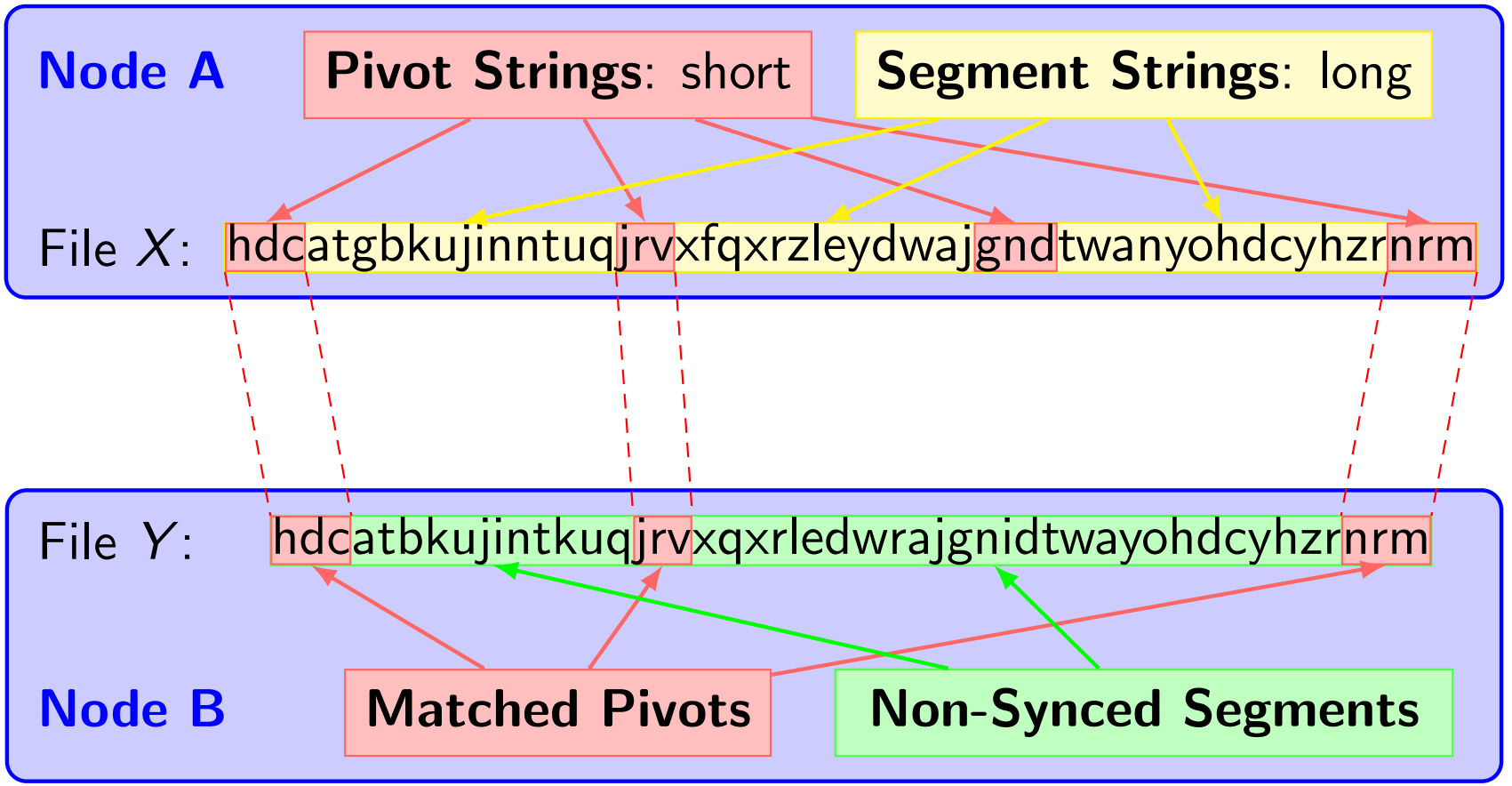
1. Matching Module: Matches the pivot strings.

Synchronization Scheme: Overview



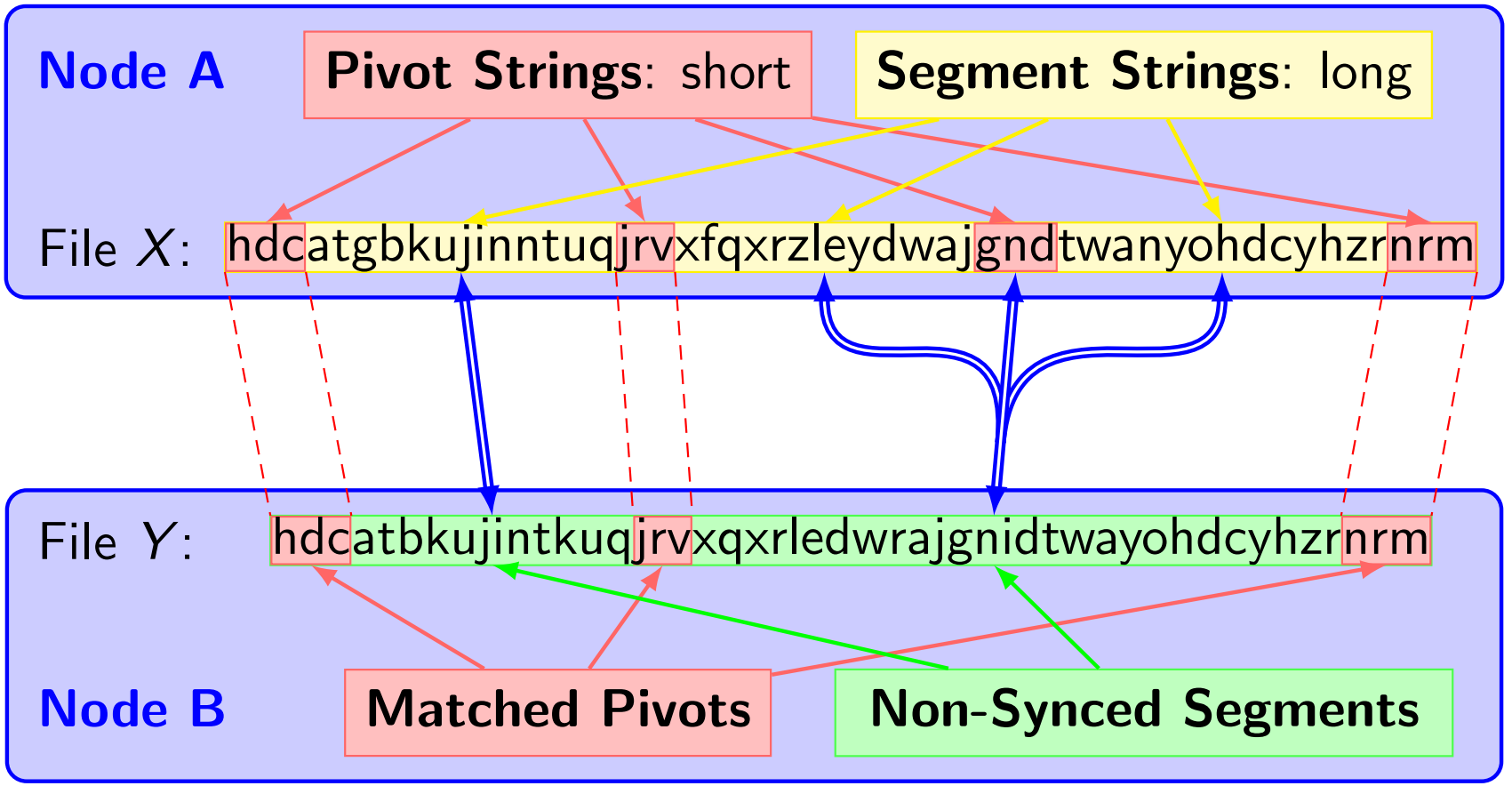
1. Matching Module: Matches the pivot strings.

Synchronization Scheme: Overview



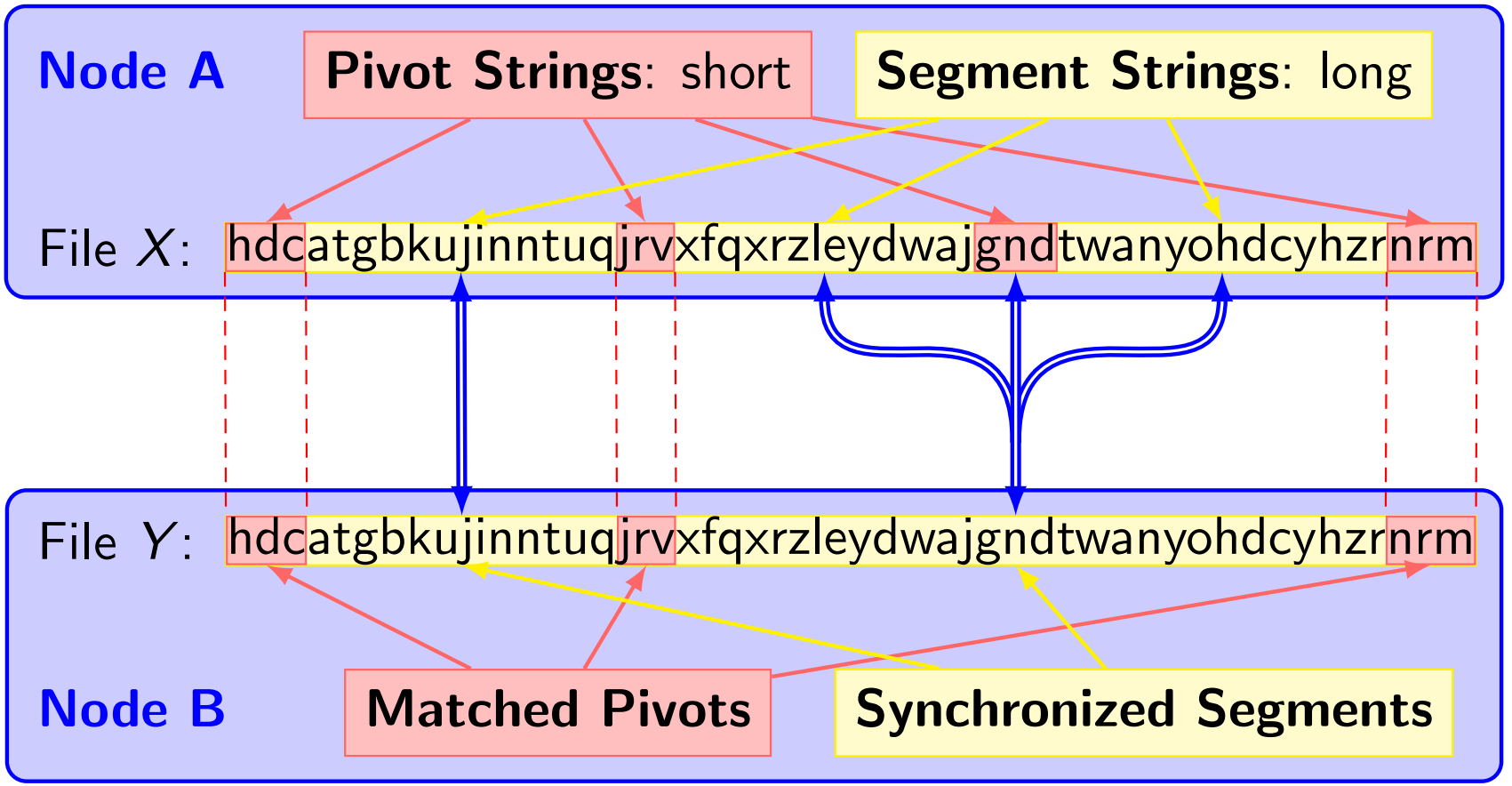
1. Matching Module: Matches the pivot strings.

Synchronization Scheme: Overview



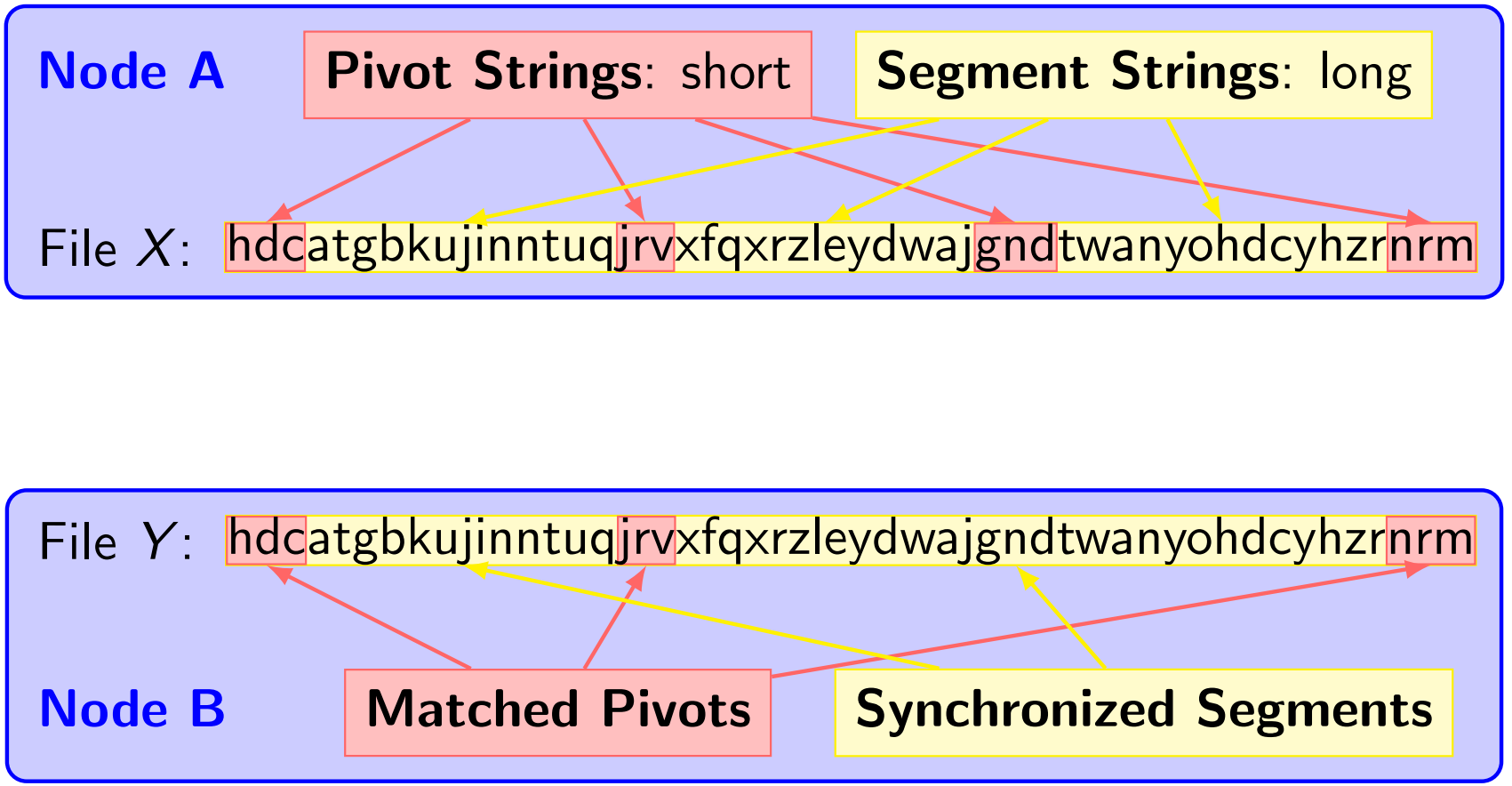
1. Matching Module: Matches the pivot strings.
2. Edit Recovery Module: Synchronizes the segment strings.

Synchronization Scheme: Overview



1. Matching Module: Matches the pivot strings.
2. Edit Recovery Module: Synchronizes the segment strings.

Synchronization Scheme: Overview



1. Matching Module: Matches the pivot strings.
2. Edit Recovery Module: Synchronizes the segment strings.
3. Channel Coding Module: Recovers from residual errors if any.

Matching Module

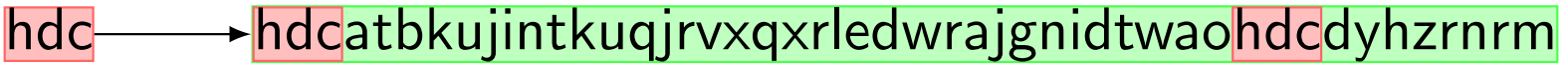
We match hdc jrv gnd nrm in

$Y =$ hdcatbkujintkuqjrvxqxrledwrajgnidtwayohdcyhzrnrm.

Matching Module

We match hdc jrv gnd nrm in

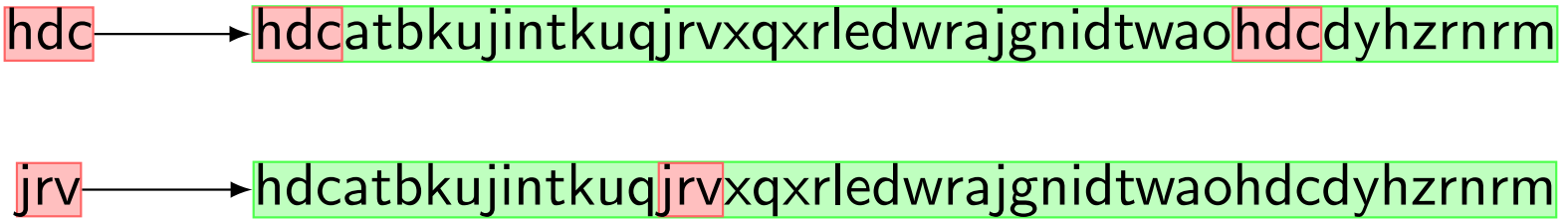
$Y =$ hdcatbkujintkuqjrvxqxrledwrajgnidtwayohdcyhzrnrm.



Matching Module

We match hdc jrv gnd nrm in

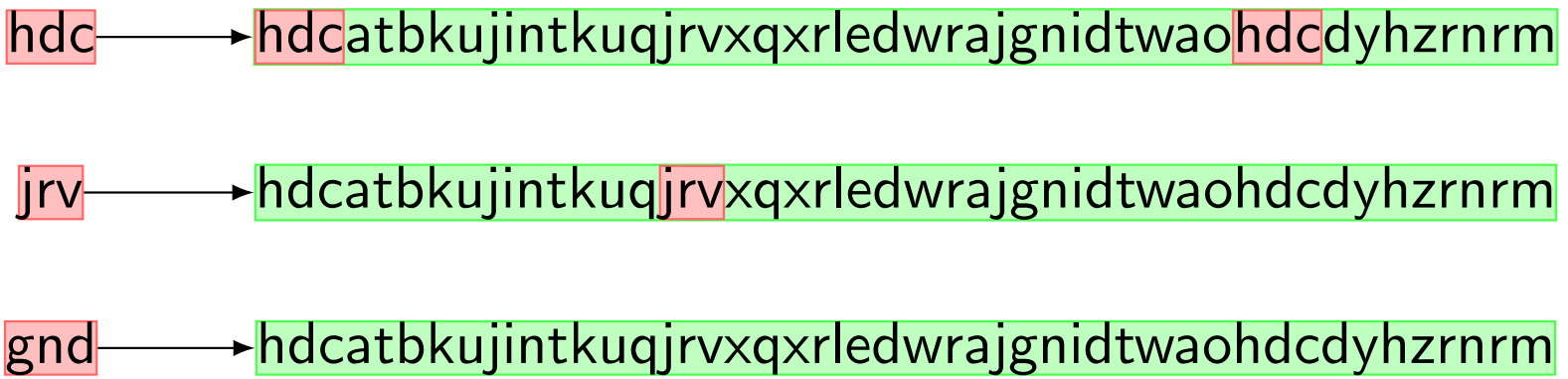
$Y =$ hdcatbkujintkuqjrvxqxrledwrajgnidtwayohdcyhzrnrm.



Matching Module

We match hdc jrv gnd nrm in

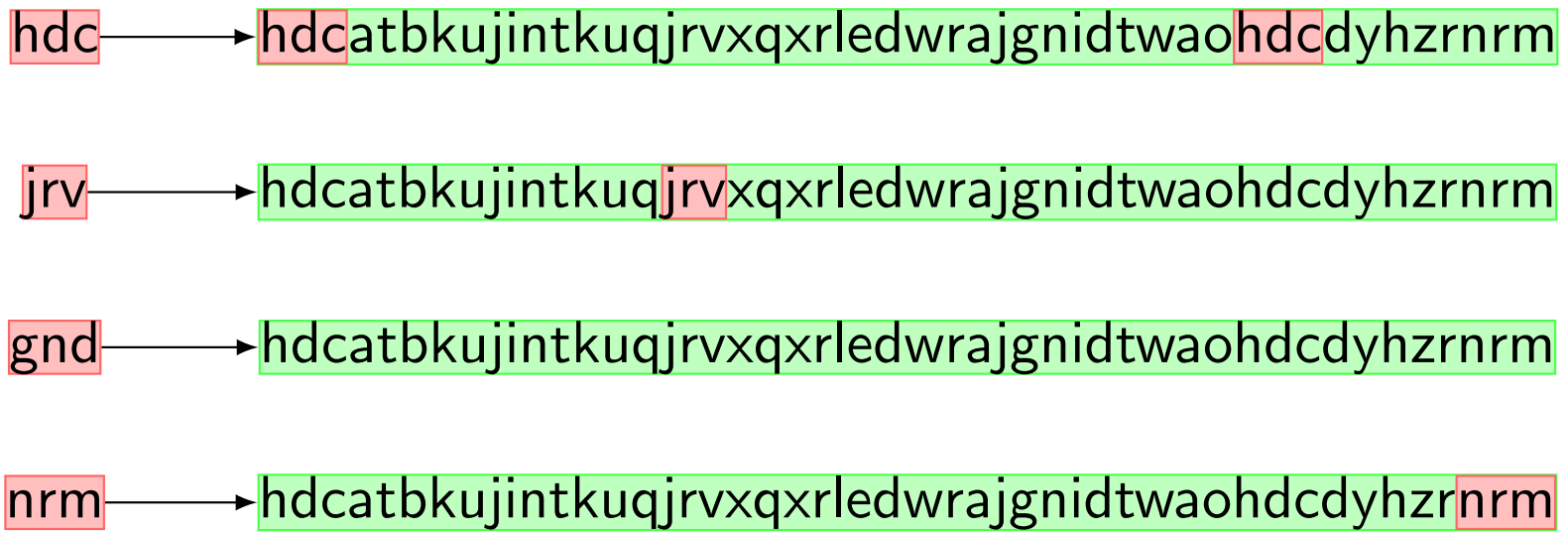
$Y =$ hdcatbkujintkuqjrvxqxrledwrajgnidtwayohdcyhzrnrm.



Matching Module

We match hdc jrv gnd nrm in

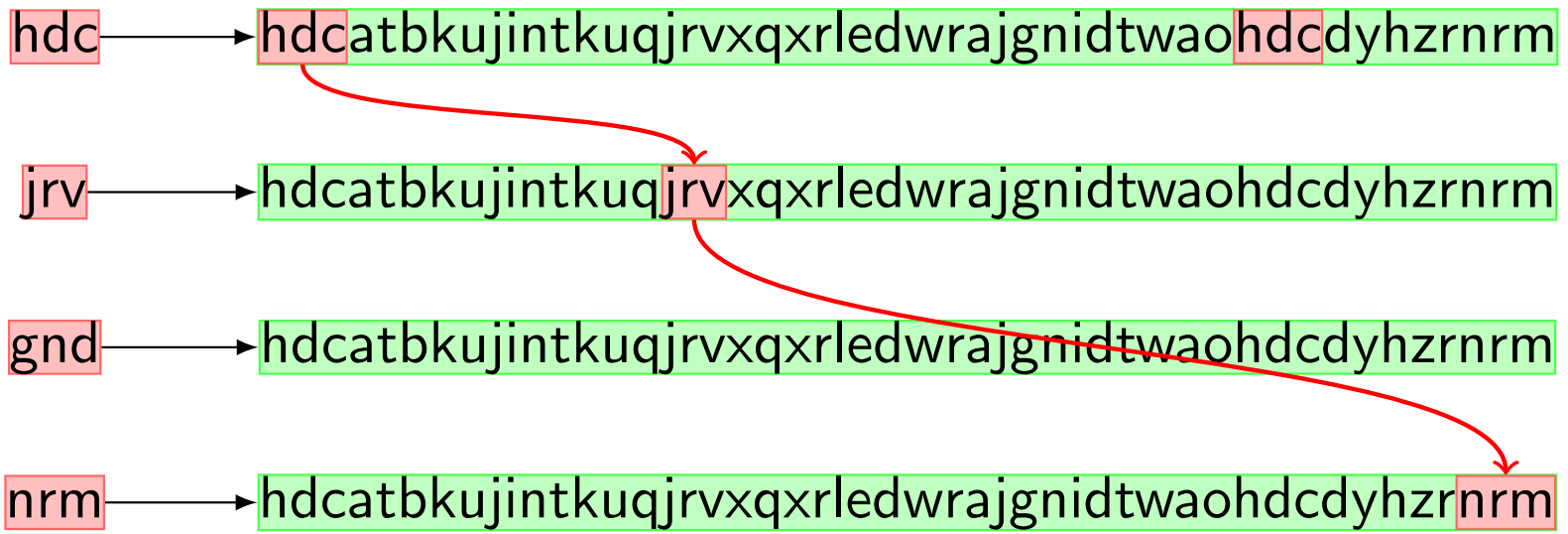
$Y =$ hdcatbkujintkuqjrvxqxrledwrajgnidtwayohdcyhzrnrm.



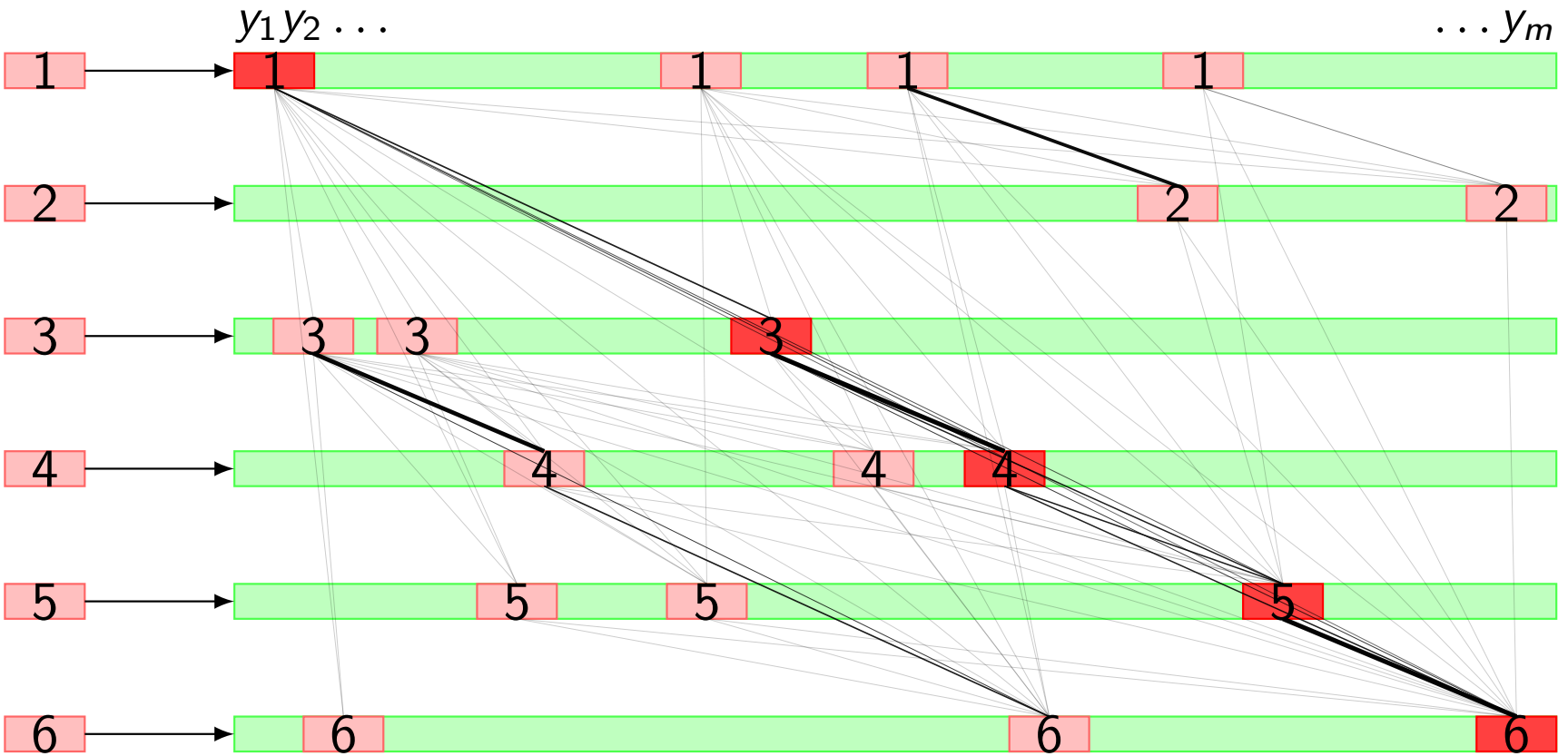
Matching Module

We match hdc jrv gnd nrm in

$Y =$ hdcatbkujintkuqjrvxqxrledwrajgnidtwayohdcyhzrnrm.

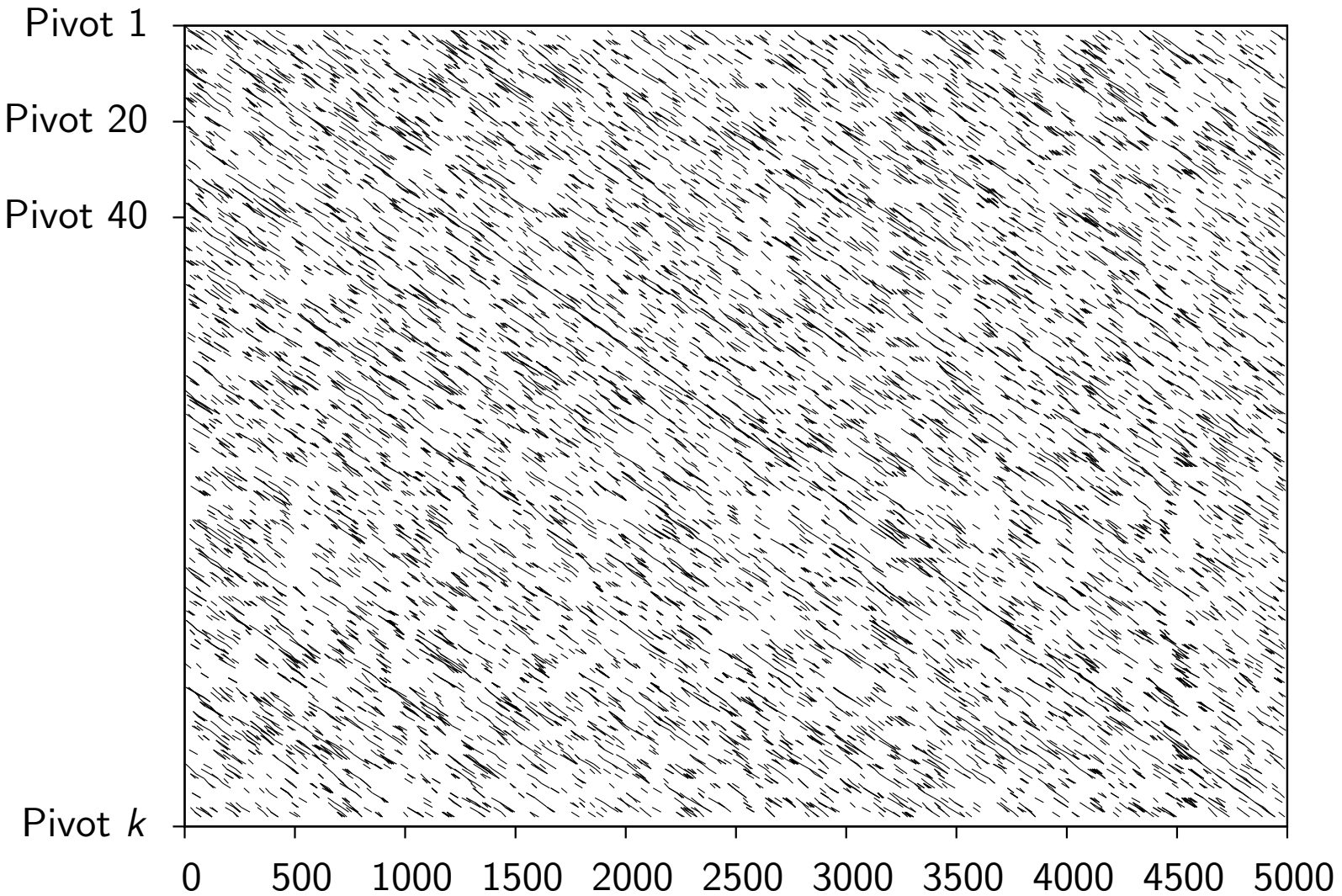


Matching Module: Larger Example



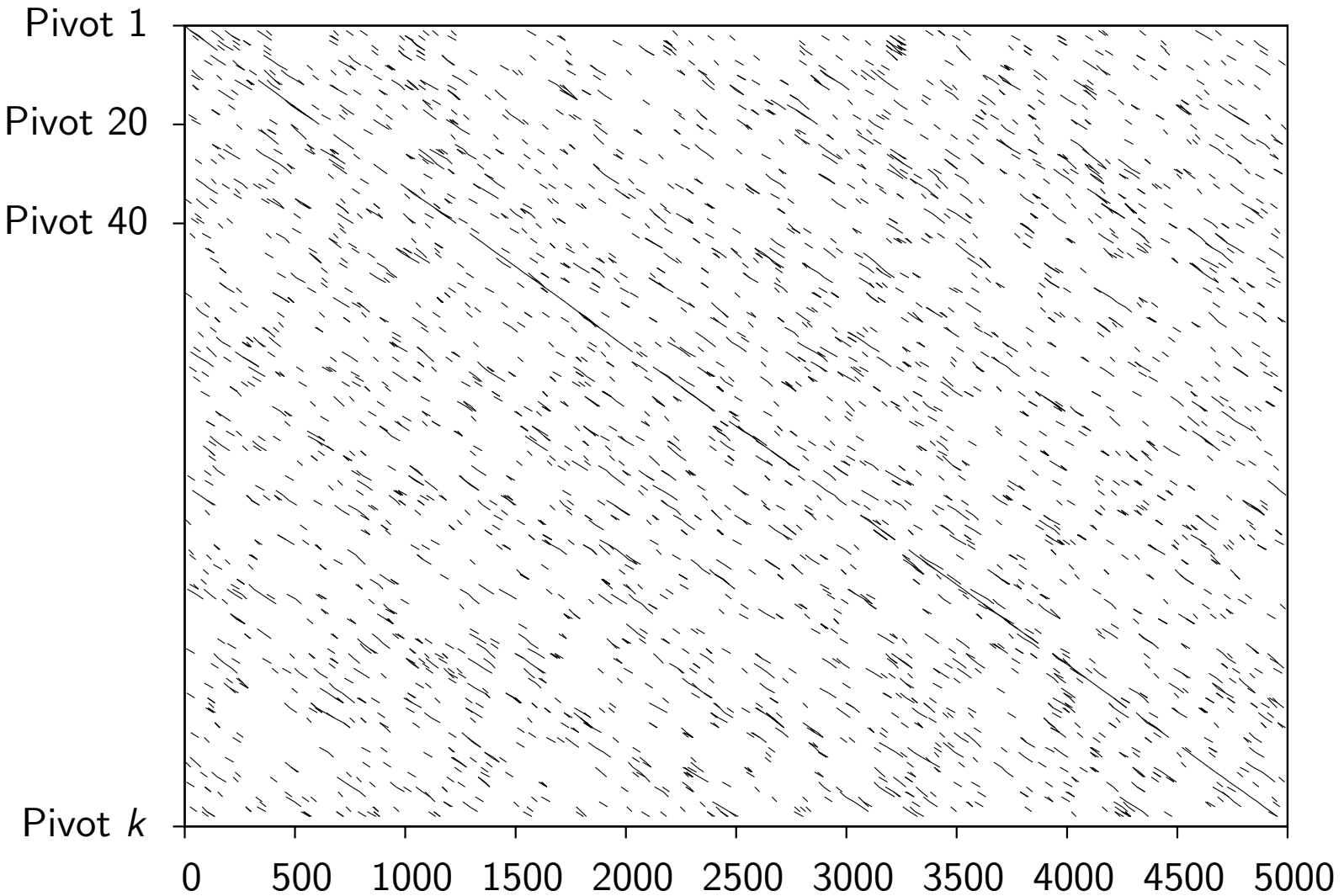
Matching Module: Even Larger Example

Edit Rates $\beta_d = \beta_i = 0.02$, $n = 5000$, Pivot Length 5



Matching Module: Even Larger Example

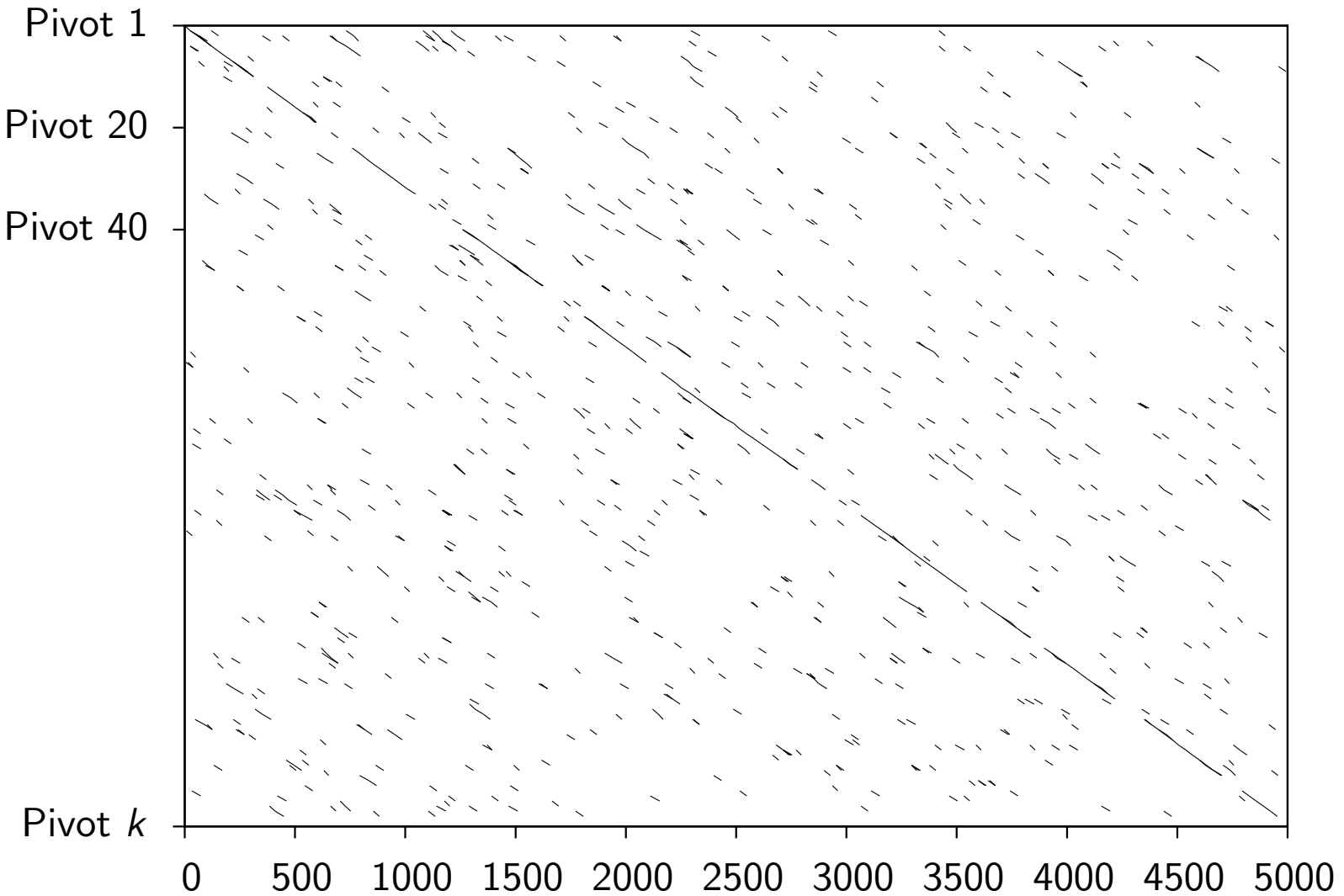
Edit Rates $\beta_d = \beta_i = 0.02$, $n = 5000$, Pivot Length 6



LOR1S

Matching Module: Even Larger Example

Edit Rates $\beta_d = \beta_i = 0.02$, $n = 5000$, Pivot Length 7



LOR1S

Matching Module: Cost and Error Probability

Errors

- Pivots are short enough to avoid edits,
- Pivots are long enough so that pivot collisions are unlikely,
- "Wrong" thick edges exist, but wrong thick paths do not.

Matching Module: Cost and Error Probability

Errors

- Pivots are short enough to avoid edits,
- Pivots are long enough so that pivot collisions are unlikely,
- "Wrong" thick edges exist, but wrong thick paths do not.

Communication

- With pivot length $O\left(\log \frac{1}{\beta}\right)$,
- With segment length $O\left(\frac{1}{\beta}\right)$,
- $O(n\beta)$ pivots are transmitted, totalling $O\left(n\beta \log \frac{1}{\beta}\right)$ bit.

Matching Module: Cost and Error Probability

Summary

- The module transmits $O\left(n\beta \log \frac{1}{\beta}\right)$ bits (in a single channel use).
- With probability $1 - O(2^{-n})$:
  - We match a fraction $1 - O\left(\beta \log \frac{1}{\beta}\right)$ of the pivots.
  - These matches are incorrect with probability $< \beta + o(\beta)$.

2 Edit Recovery Module

Edit Recovery Module: Correcting Single Edits

When two files differ by a single edit, synchronization:

- can be done in a single round of communication,
- in a perfect manner (no error),
- communicating $\sim \log L + \log q$ bits (from A to B), where L and $L \pm 1$ are the lengths of the files.

G. M. Tenengolts, "Nonbinary codes, correcting single deletion or insertion", 1984.

Edit Recovery Module

Goal: $X =$ j r v x q x r l e d w r a j g n i d t w a y o h d c y h z r n r m

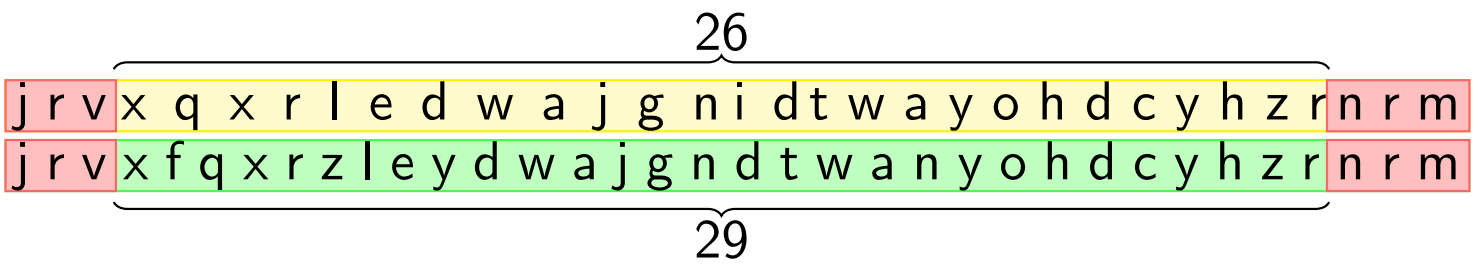
Given: $Y =$ j r v x f q x r z l e y d w a j g n d t w a n y o h d c y h z r n r m


R. Venkataramanan, H. Zhang, and K. Ramchandran, "Interactive low-complexity codes for synchronization from deletions and insertions", 2010.

Edit Recovery Module

Goal: $X =$ j r v x q x r l e d w r a j g n i d t w a y o h d c y h z r n r m

Given: $Y =$ j r v x f q x r z l e y d w a j g n d t w a n y o h d c y h z r n r m



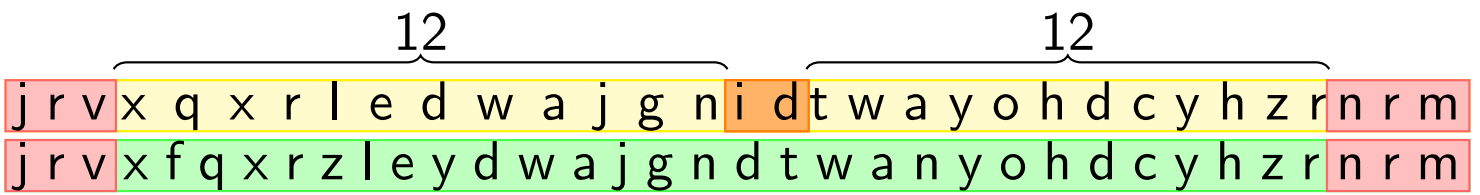
Lengths differ by more than 1:
Send a central delimiter.

R. Venkataramanan, H. Zhang, and K. Ramchandran, "Interactive low-complexity codes for synchronization from deletions and insertions", 2010.

Edit Recovery Module

Goal: $X =$ j r v x q x r l e d w r a j g n i d t w a y o h d c y h z r n r m

Given: $Y =$ j r v x f q x r z l e y d w a j g n d t w a n y o h d c y h z r n r m



Delimiter i d has no match:
Choose a different delimiter.

R. Venkataramanan, H. Zhang, and K. Ramchandran, "Interactive low-complexity codes for synchronization from deletions and insertions", 2010.

Edit Recovery Module

Goal: $X =$ j r v x q x r l e d w r a j g n i d t w a y o h d c y h z r n r m
Given: $Y =$ j r v x f q x r z l e y d w a j g n d t w a n y o h d c y h z r n r m

j r v x q x r l e d w a j g n i d t w a y o h d c y h z r n r m
j r v x f q x r z l e y d w a j g n d t w a n y o h d c y h z r n r m

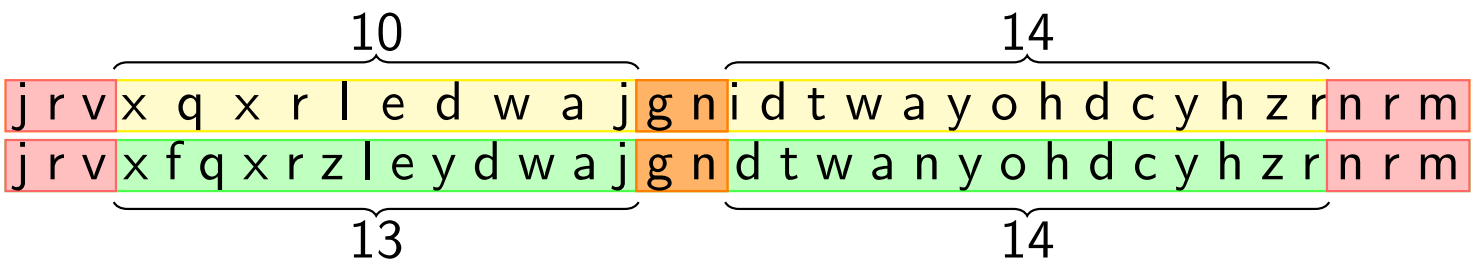
New "central" delimiter g n is matched:
Repeat on both sides.


R. Venkataramanan, H. Zhang, and K. Ramchandran, "Interactive low-complexity codes for synchronization from deletions and insertions", 2010.

Edit Recovery Module

Goal: $X =$ j r v x q x r l e d w r a j g n i d t w a y o h d c y h z r n r m

Given: $Y =$ j r v x f q x r z l e y d w a j g n d t w a n y o h d c y h z r n r m



Left side:

Lengths differ by more than 1:
Send a delimiter.

Right side:

Lengths are equal:

- Test if strings are equal (hash),
- No: Send a delimiter.

Edit Recovery Module

Goal: $X =$ j r v x q x r l e d w r a j g n i d t w a y o h d c y h z r n r m

Given: $Y =$ j r v x f q x r z l e y d w a j g n d t w a n y o h d c y h z r n r m

j r v x q x r l e d w a j g n i d t w a y o h d c y h z r n r m

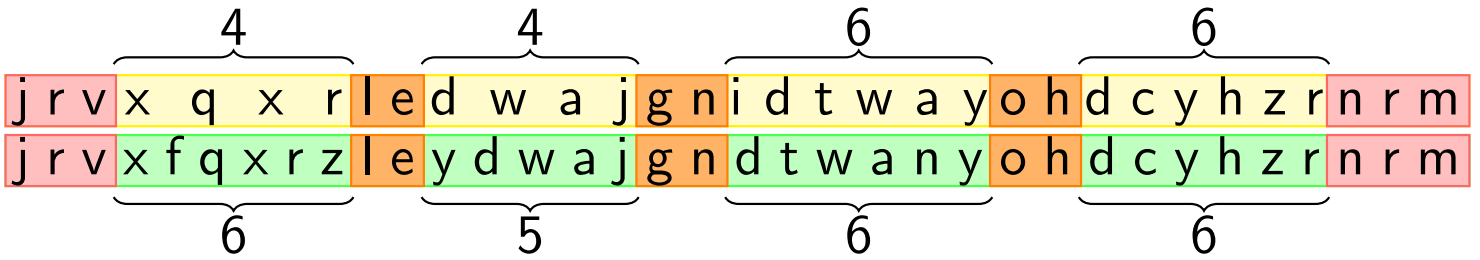
j r v x f q x r z l e y d w a j g n d t w a n y o h d c y h z r n r m

And so on...

Edit Recovery Module

Goal: $X =$ j r v x q x r l e d w r a j g n i d t w a y o h d c y h z r n r m

Given: $Y =$ j r v x f q x r z l e y d w a j g n d t w a n y o h d c y h z r n r m



And so on...

Edit Recovery Module

Goal: $X =$ j r v x q x r l e d w r a j g n i d t w a y o h d c y h z r n r m

Given: $Y =$ j r v x f q x r z l e y d w a j g n d t w a n y o h d c y h z r n r m

j r v x q x r l e d w a j g n i d t w a y o h d c y h z r n r m

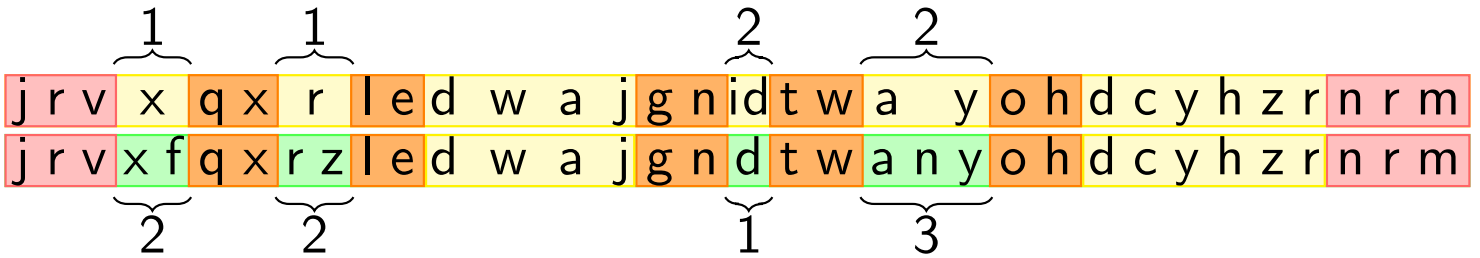
j r v x f q x r z l e d w a j g n d t w a n y o h d c y h z r n r m

And so on...

Edit Recovery Module

Goal: $X =$ j r v x q x r l e d w r a j g n i d t w a y o h d c y h z r n r m

Given: $Y =$ j r v x f q x r z l e y d w a j g n d t w a n y o h d c y h z r n r m



And so on...

Edit Recovery Module

Goal: $X =$ j r v x q x r l e d w r a j g n i d t w a y o h d c y h z r n r m

Given: $Y =$ j r v x f q x r z l e y d w a j g n d t w a n y o h d c y h z r n r m

j r v x q x r l e d w a j g n i d t w a y o h d c y h z r n r m

j r v x q x r l e d w a j g n i d t w a y o h d c y h z r n r m

And so on... until every subproblem is solved.

Edit Recovery Module: Cost and Error Probability

Errors

- Come from Hash Collisions,
- Increase hash length:
  - More communication,
  - Less collisions.

Edit Recovery Module: Cost and Error Probability

Errors

- Come from Hash Collisions,
- Increase hash length:
  - More communication,
  - Less collisions.

Communication

- Hashes (from A to B),
- Delimiters (from A to B),
- Syndromes (from A to B),
- Control (from B to A).

Edit Recovery Module: Cost and Error Probability

Summary

- The module transmits $O\left(n\beta \log \frac{1}{\beta}\right)$ bits (in a few rounds of communication).
- With probability $1 - O(2^{-n})$, only a fraction $o(\beta)$ of the segments is mis-synchronized.

3 Channel Coding Module

Channel Coding Module: Motivation

Possible Errors

- Pivots mismatched by the Matching Module.
- Delimiters mismatched by the Edit Recovery Module.
- Hash collisions.

After these two modules,
the symbol-error probability is $< 2\beta + o(\beta)$.

Channel Coding Module: Motivation

Possible Errors

- Pivots mismatched by the Matching Module.
- Delimiters mismatched by the Edit Recovery Module.
- Hash collisions.

After these two modules,
the symbol-error probability is $< 2\beta + o(\beta)$.
$\Rightarrow$ Correct these errors with the Channel Coding Module.

Channel Coding Module

Node A

X

(Same length as X)

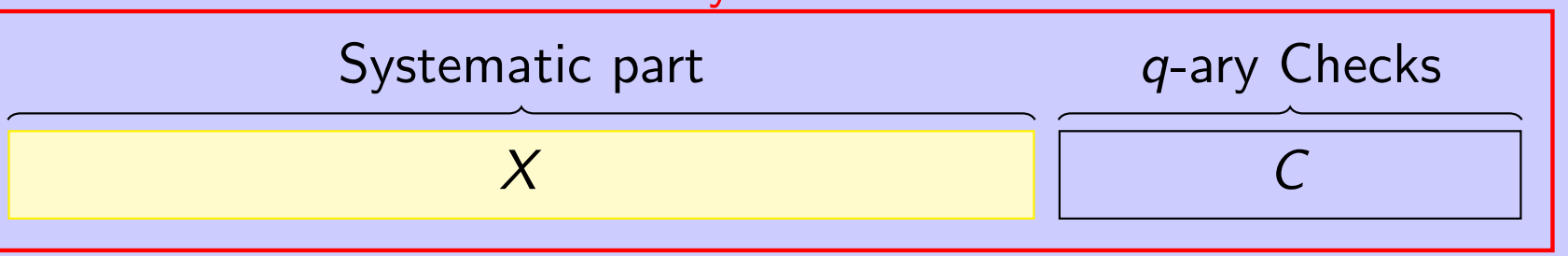
Output $\approx X$ from Edit Recov. Mod.

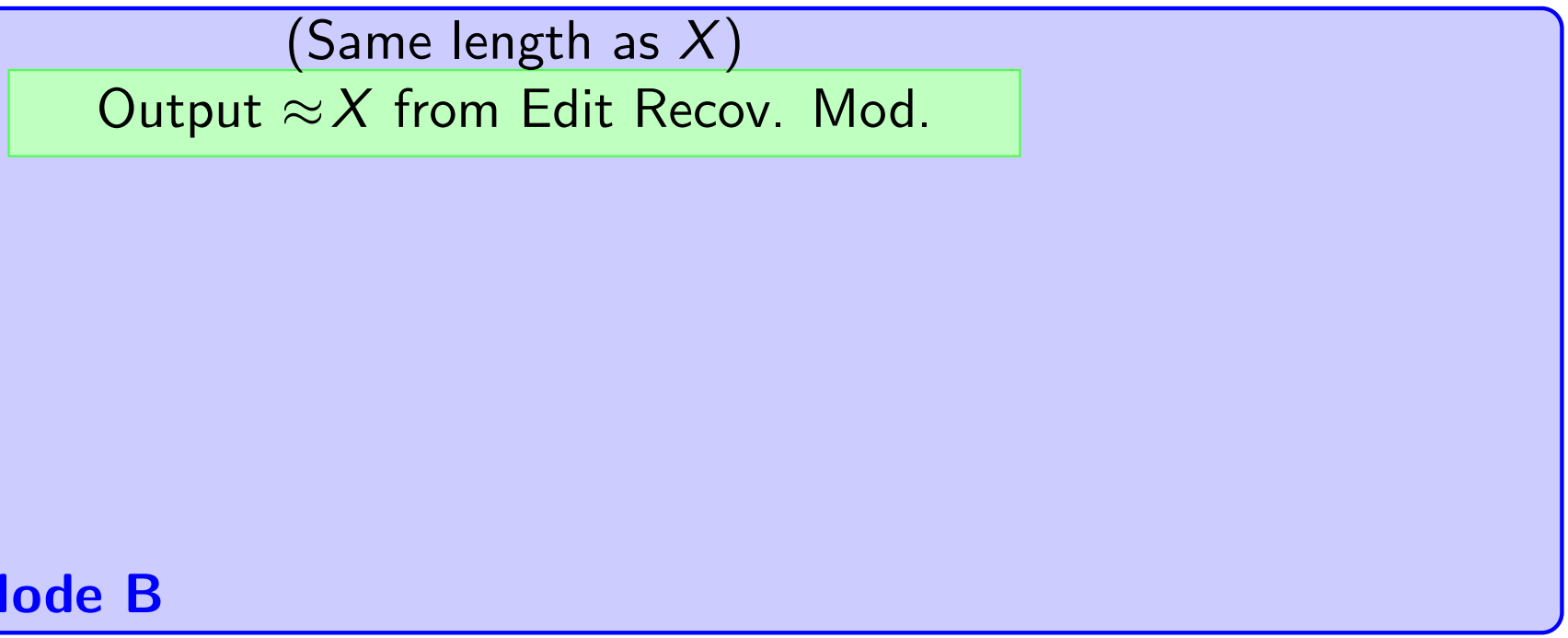
Node B

Channel Coding Module

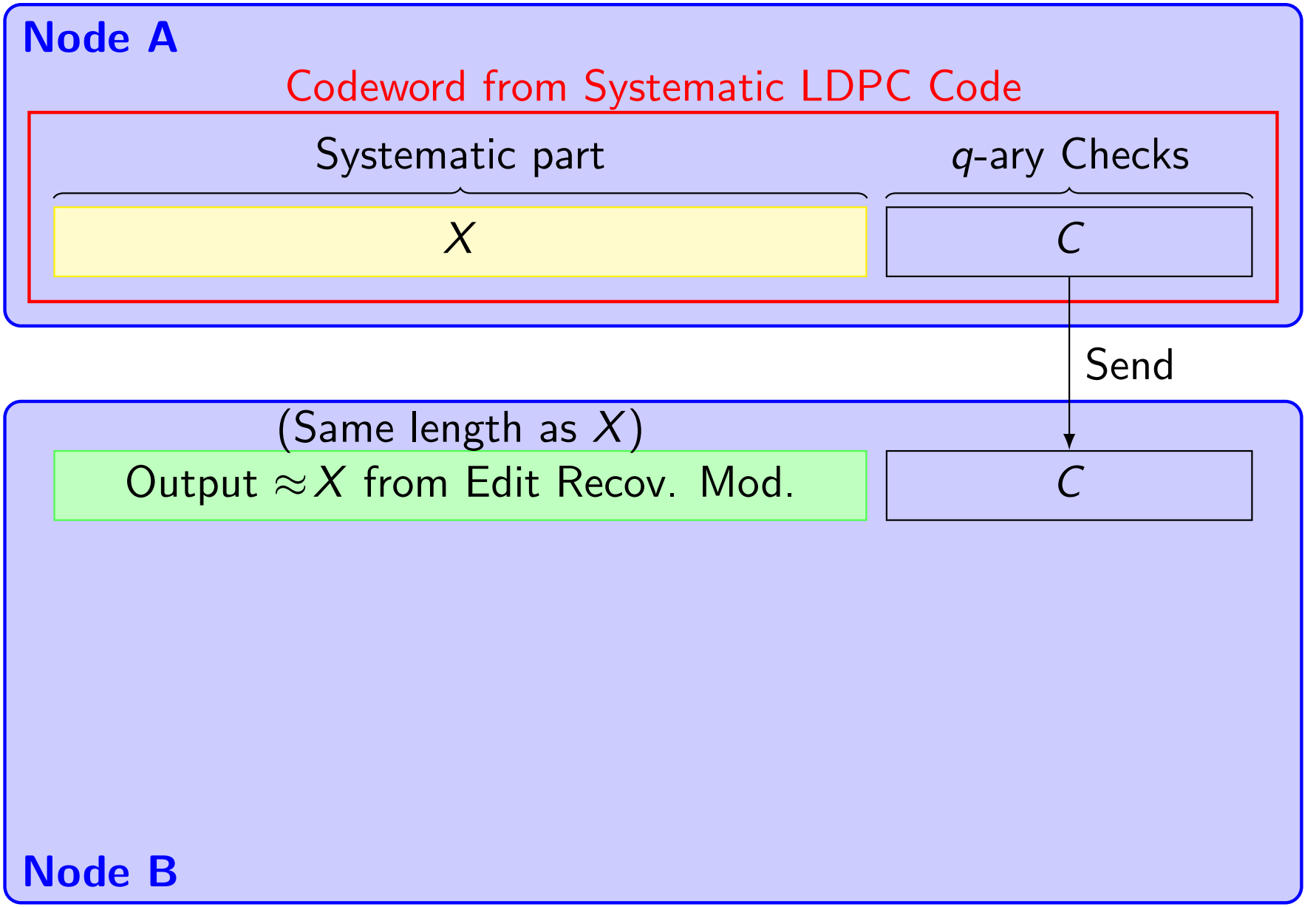
Node A

Codeword from Systematic LDPC Code

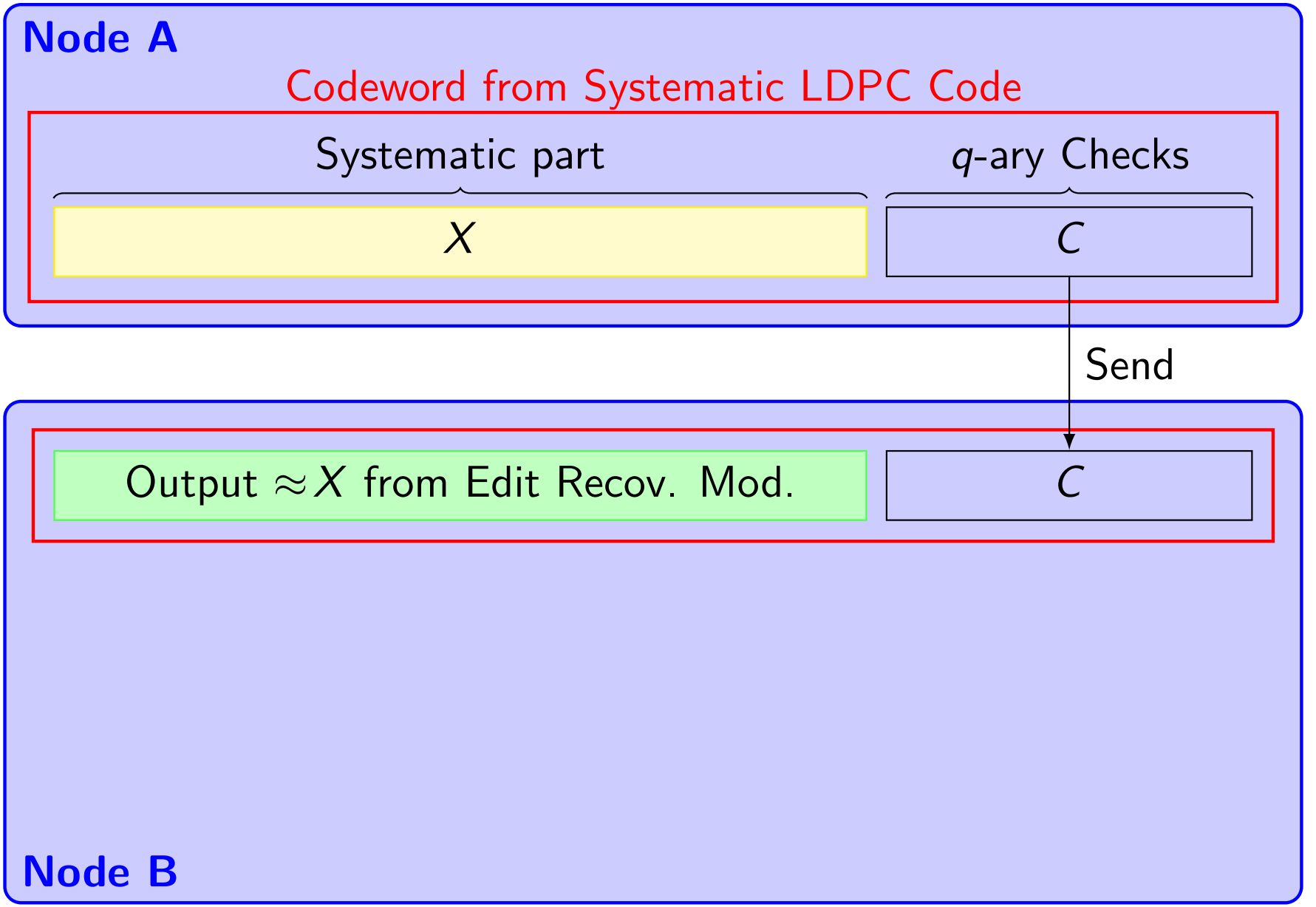




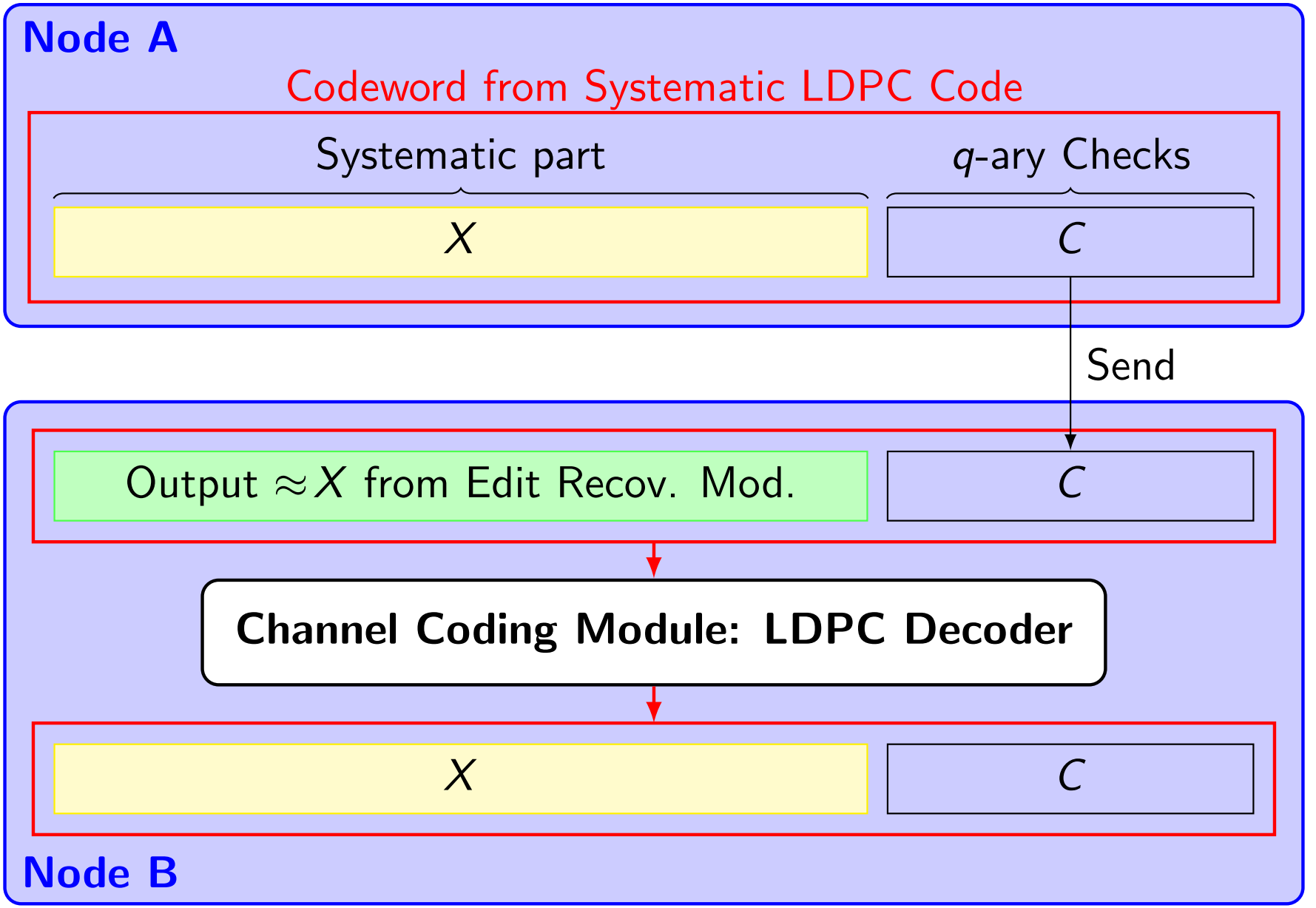
Channel Coding Module



Channel Coding Module



Channel Coding Module



Channel Coding Module: Cost and Error Probability

Communication

- Residual error probability from previous modules: $2\beta + o(\beta)$.
- Additional data required to correct these errors: $\text{Const} \cdot n \cdot H(2\beta + o(\beta)) = O\left(n\beta \log \frac{1}{\beta}\right)$ bits.

Channel Coding Module: Cost and Error Probability

Communication

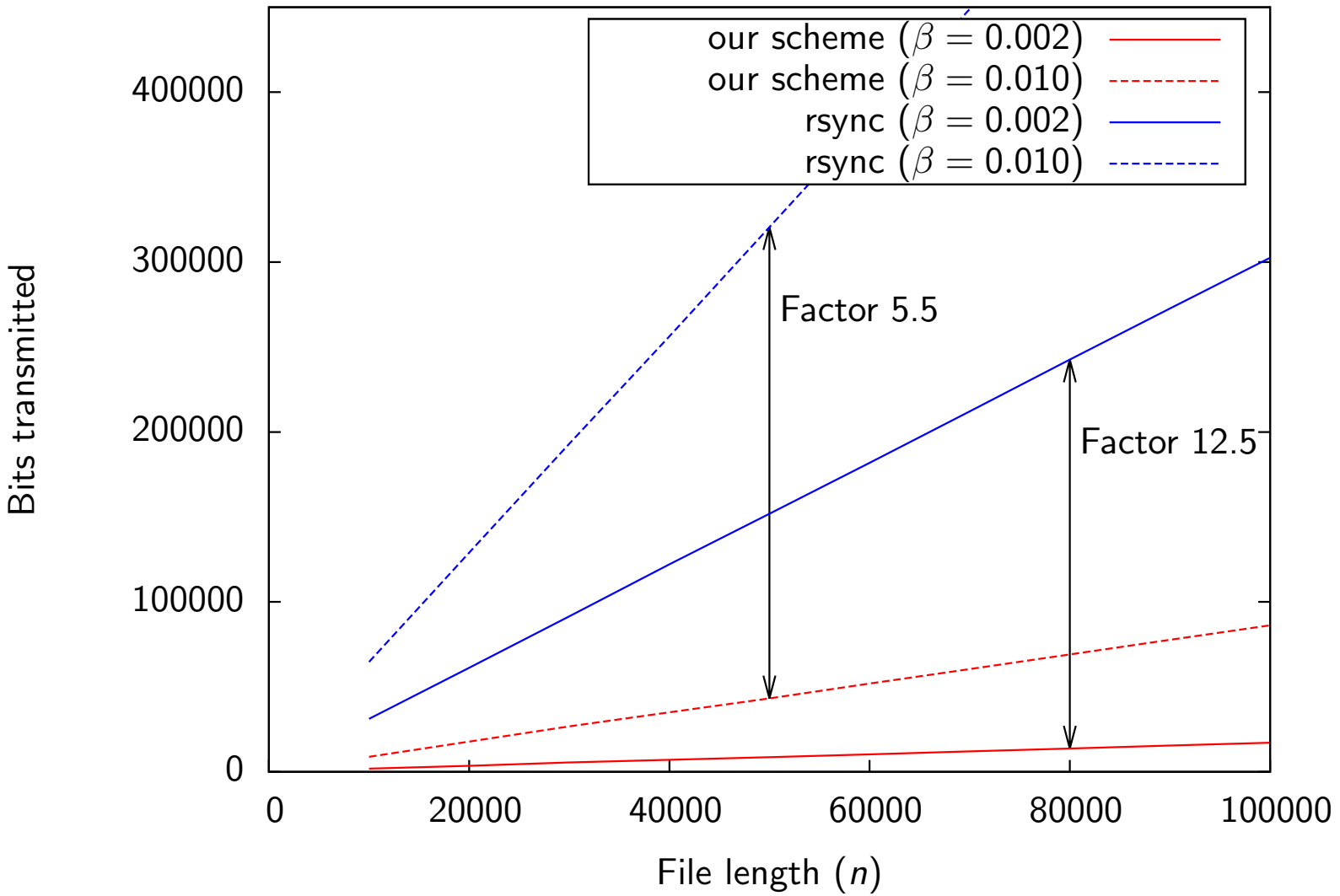
- Residual error probability from previous modules: $2\beta + o(\beta)$.
- Additional data required to correct these errors: $\mathrm{Const} \cdot n \cdot H(2\beta + o(\beta)) = O\left(n\beta \log \frac{1}{\beta}\right)$ bits.

Summary

- The module transmits $O\left(n\beta \log \frac{1}{\beta}\right)$ bits (in a single channel use).
- With probability $1 - O(2^{-n})$, Y is synchronized with no remaining error.

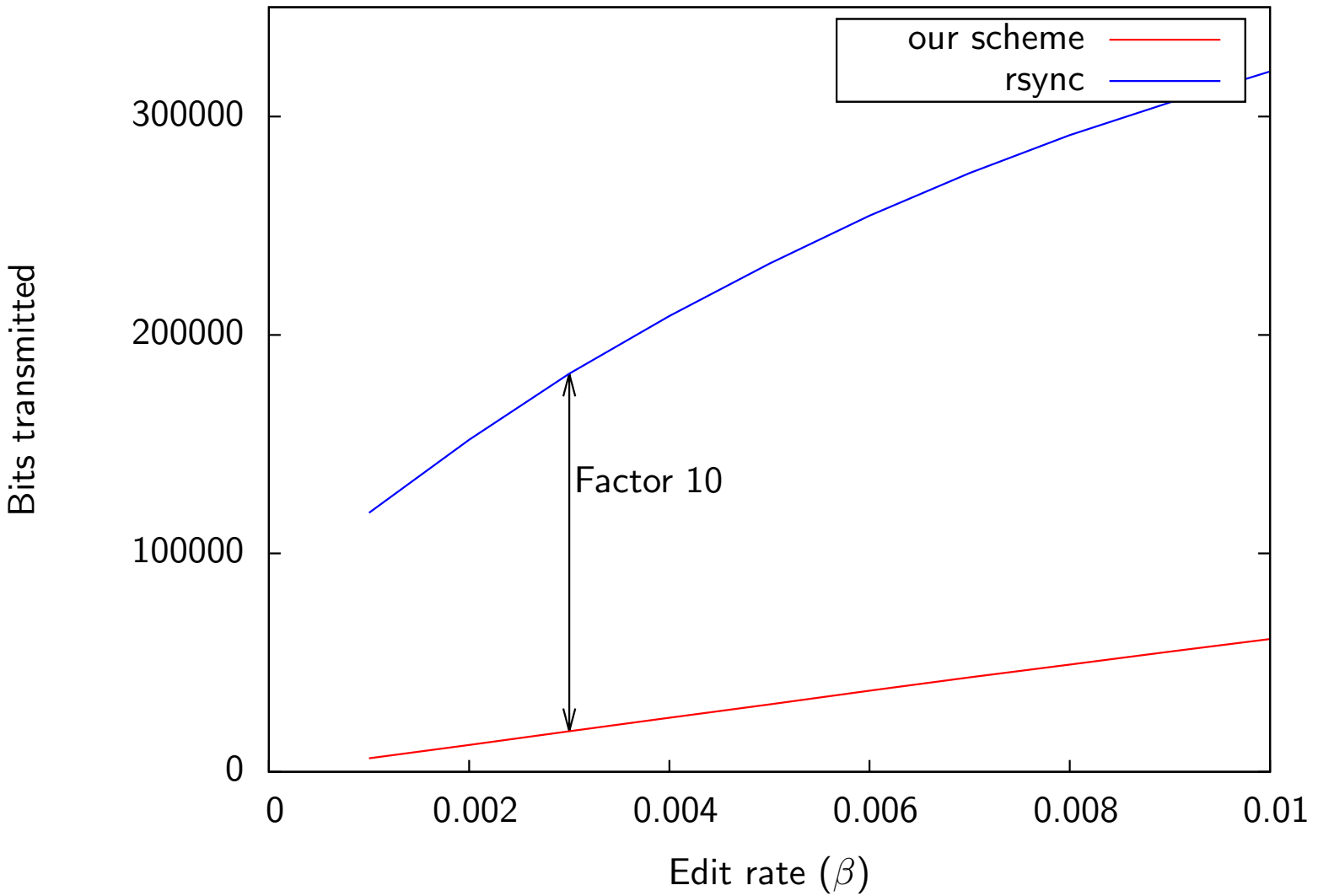
Comparison with RSYNC when n varies

- $\beta = 0.01$ and $\beta = 0.002$, i.i.d. file, i.i.d. edits, $q = 52$,
- Our pivot length: 5, segment length: $1/\beta$.



Comparison with RSYNC when β varies

- $n = 50000$, i.i.d. file, i.i.d. edits, $q = 52$,
- Our pivot length: 5, segment length: $1/\beta$.



Comparison with Venkataramanan *et al.* scheme

- $\beta = 0.01$, i.i.d. file, i.i.d. edits, $q = 52$,
- Our pivot length: 5, segment length: $1/\beta$.

Bandwidth (in bits)

	n	20k	40k	60k	100k
Median	Our scheme	18k	35k	54k	87k
	Venkataramanan	19k	41k	63k	87k
Worst-case	Our scheme	52k	96k	95k	95k
	Venkataramanan	390k	845k	1,216k	346k

Rounds of communication required

Our scheme completes in about half less rounds.

Errors prior to Channel Coding

Our error rate per symbol is also about half lower.

Applications

In addition to reducing data storage demand, our algorithm can be used for:

Applications

In addition to reducing data storage demand, our algorithm can be used for:

- Synchronization in general data storage (Dropbox),

Applications

In addition to reducing data storage demand, our algorithm can be used for:

- Synchronization in general data storage (Dropbox),
- Synchronization in particular data repositories: source control (Github, SVN, etc...), video (YouTube, Vimeo),

Applications

In addition to reducing data storage demand, our algorithm can be used for:

- Synchronization in general data storage (Dropbox),
- Synchronization in particular data repositories: source control (Github, SVN, etc...), video (YouTube, Vimeo),
- Database searches: determining whether two records are similar.

Ongoing Work

- Allow for more complex edit patterns
  (e.g., in practical scenarios, edits are often “bursty”),

Ongoing Work

- Allow for more complex edit patterns
  (e.g., in practical scenarios, edits are often "bursty"),
- Specialize our scheme to application-dependent types of files
  (e.g., if the files are source code, exploit that structure),

Ongoing Work

- Allow for more complex edit patterns
  (e.g., in practical scenarios, edits are often "bursty"),
- Specialize our scheme to application-dependent types of files
  (e.g., if the files are source code, exploit that structure),
- Optimize the implementation (e.g., in terms of both computation and network usage).