Efficient Synchronization of Files in Distributed Storage Systems

Han Mao Kiah
joint work with Salim El Rouayheb (IIT), Sreechakra Goparaju (UCSD), and Olgica Milenkovic (UIUC)

Coordinated Science Lab,
University of Illinois at Urbana-Champaign

Nov 5, 2014

Problem Description: A $[3, 2]$-MDS Code

Files (Data chunks)

x_1	x_2	x_3	x_4	x_5

y_1	y_2	y_3	y_4	y_5

Storage Nodes

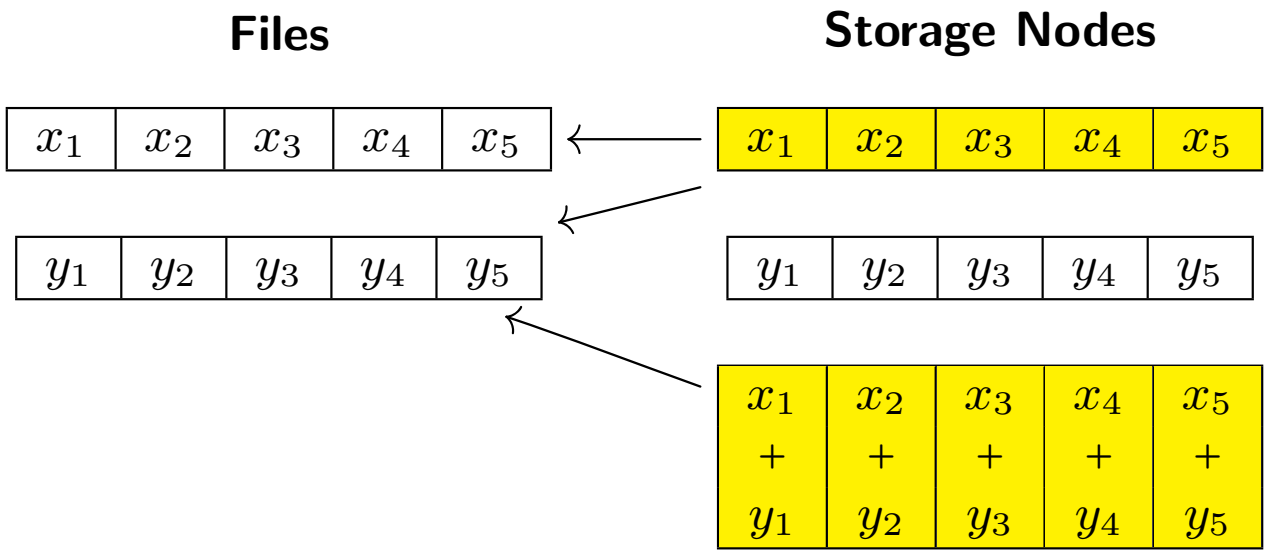
x_1	x_2	x_3	x_4	x_5

x_1	y_2	y_3	y_4	y_5

x_1	x_2	x_3	x_4	x_5
$+$	$+$	$+$	$+$	$+$
y_1	y_2	y_3	y_4	y_5

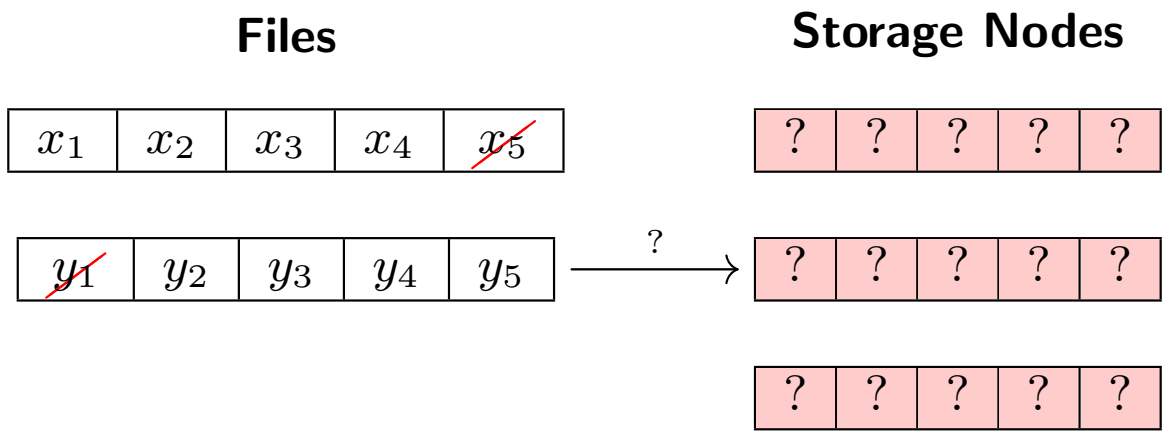
Consider the two users with files $\mathbf{x}$ and $\mathbf{y}$ of length five over $\mathbb{F}_q$.
Here, q is chosen based on the smallest number of consecutive, editable bits.

Problem Description: A $[3, 2]$-MDS Code



Consider three storage nodes where nodes 1, 2 and 3 store user information $\mathbf{x}$ and $\mathbf{y}$, and parity information $\mathbf{x} + \mathbf{y}$, respectively.
Then the system is able to reconstruct both data files by accessing any two nodes.

Problem Description: File Edits



Suppose both files are subjected to a single symbol deletion.
What protocol should the users employ and what information do they have to communicate to the three storage nodes so as to retain reconstruction functionality with minimal data transmission cost?

Problem Description: Retaining Reconstructability

To retain reconstruction functionality, one way is for the nodes to update their respective contents to the following.

Files

x_1	x_2	x_3	x_4

y_2	y_3	y_4	y_5

Storage Nodes (after update)

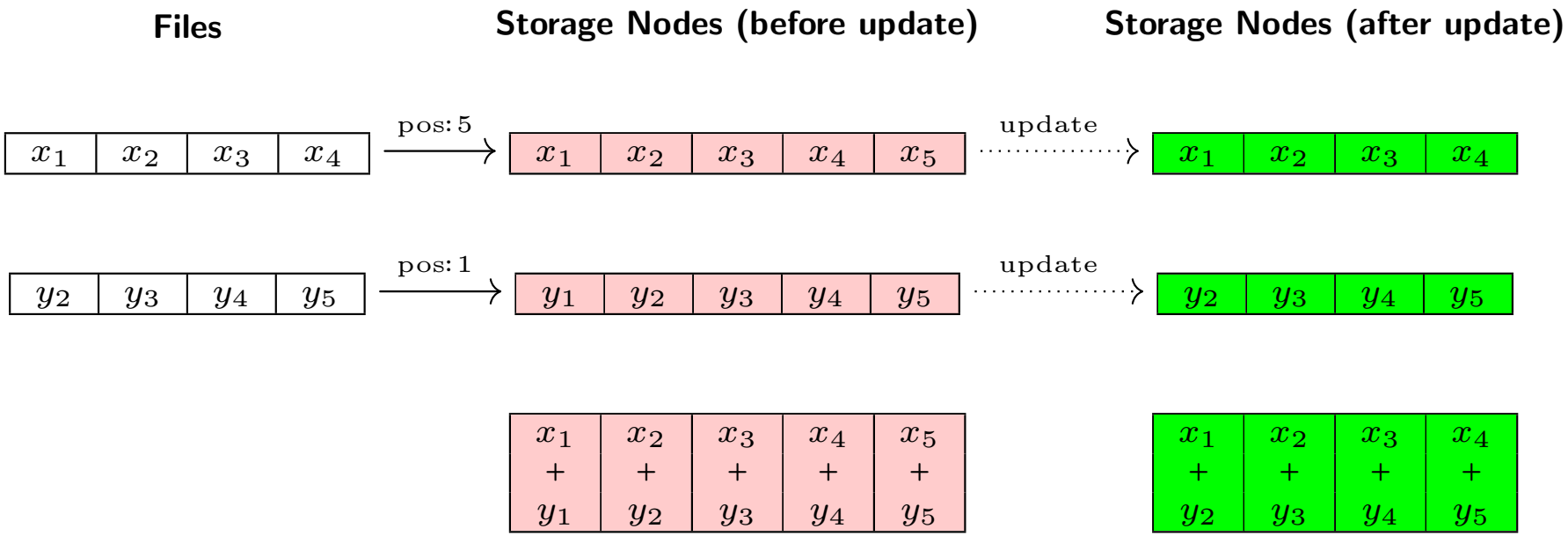
x_1	x_2	x_3	x_4

y_2	y_3	y_4	y_5

x_1 + y_2	x_2 + y_3	x_3 + y_4	x_4 + y_5

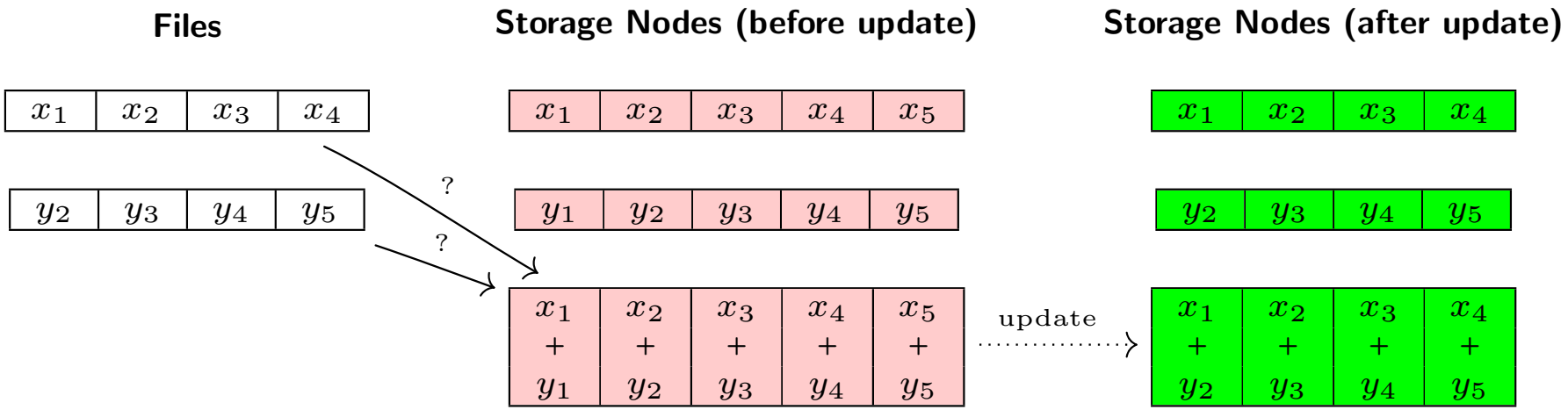
Problem Description: Updating Nodes 1 and 2

To update the nodes 1 and 2, we do the following.



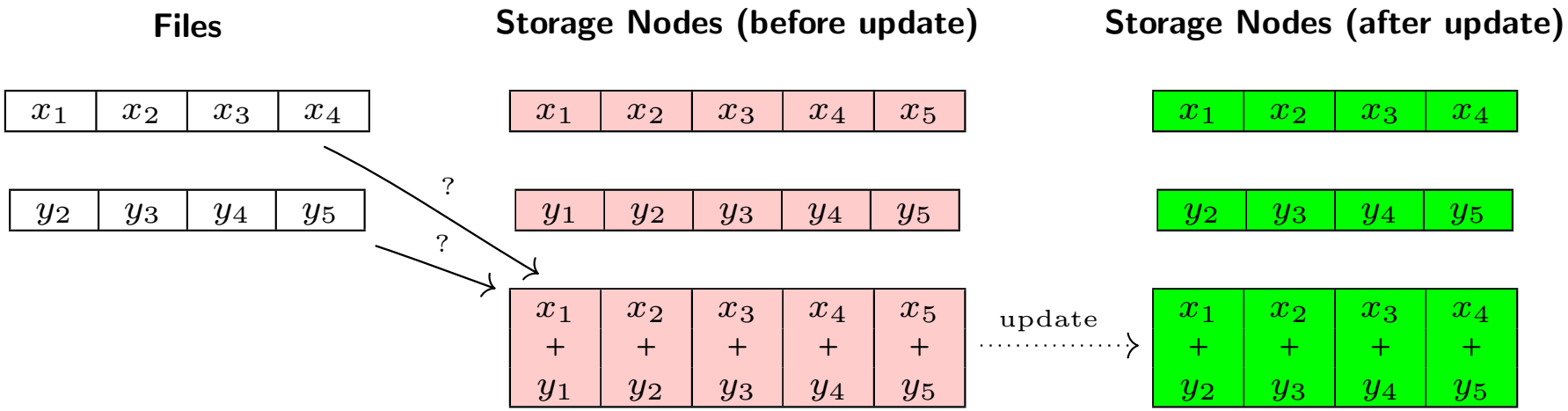
Problem Description: Updating Node 3

What is the minimum communication complexity needed for node 3 to update its content?



Problem Description: Updating Node 3

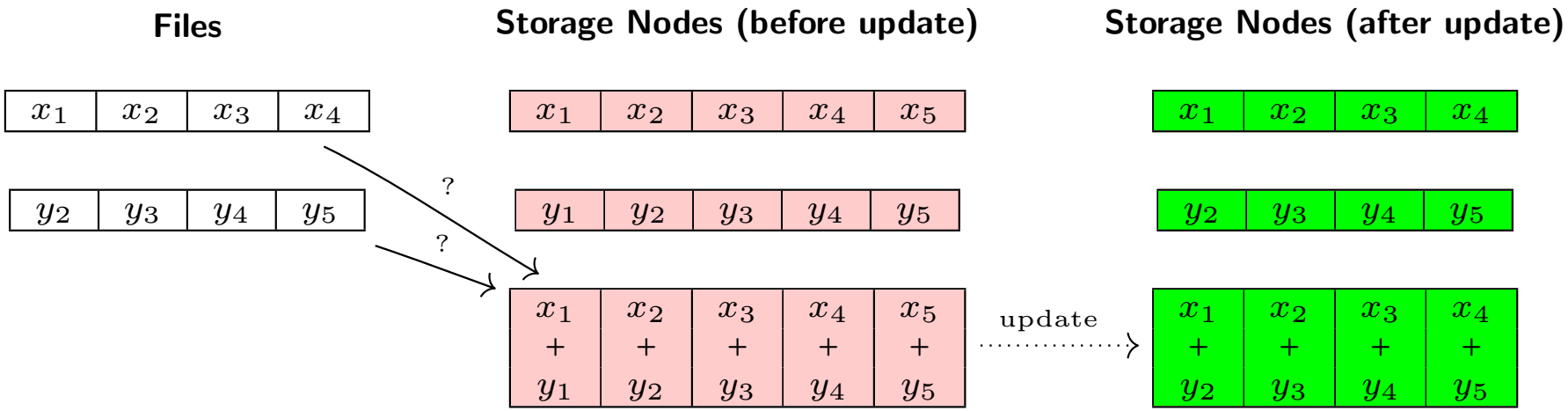
What is the minimum communication complexity needed for node 3 to update its content?



It appears that one of the files have to transmit the entire string.

Problem Description: Updating Node 3

What is the minimum communication complexity needed for node 3 to update its content?



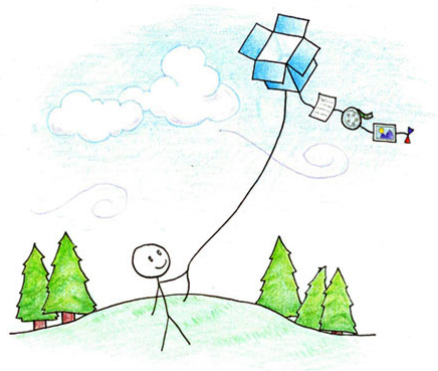
It appears that one of the files have to transmit the entire string.

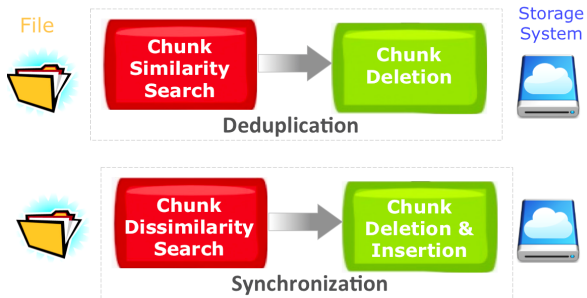
Main Contribution

Significant savings in communication complexity can be achieved via a change in code structure.

Motivation

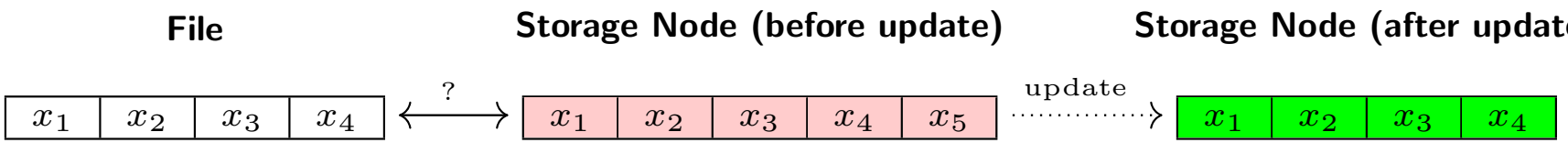
Applicable to distributed storage systems that store redundant coded copies of files over a set of servers, disks or nodes connected through a communication network.
Two key functionalities: reconstructablity and repairability of the system need to be retained when the content of the files undergoes edits.
Examples: Dropbox, Sugarsync and dual deduplication protocols.





Related Work: Uncoded Domain

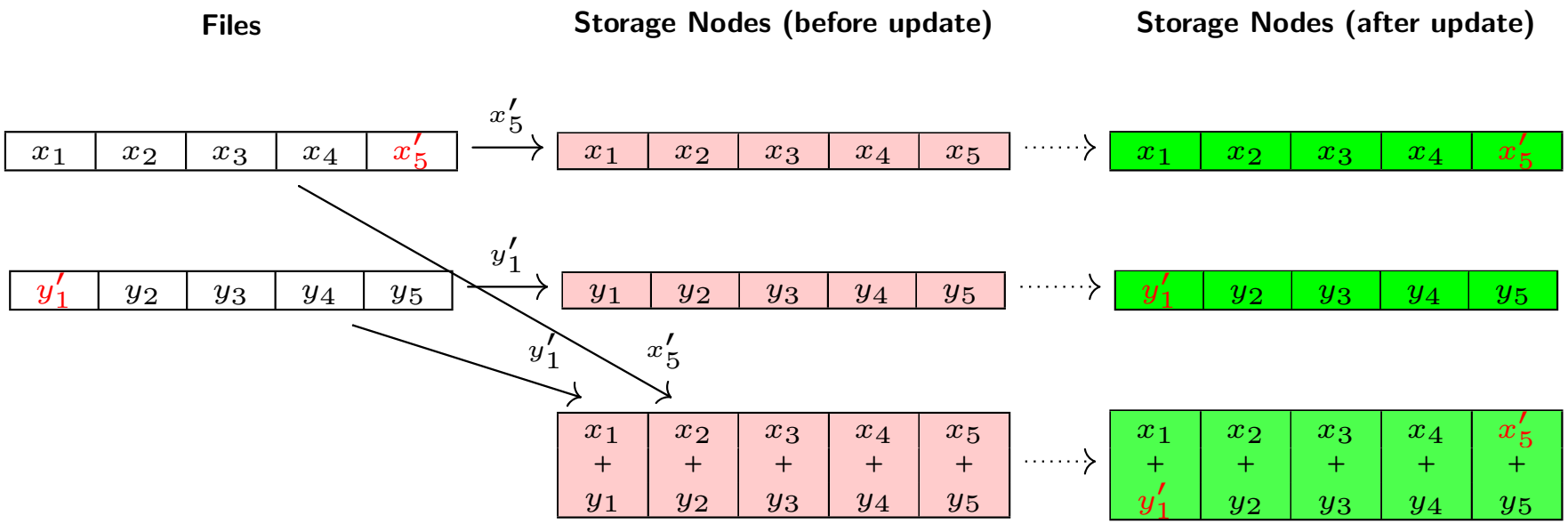
Single user and single node storing a replicate (uncoded) of the user's file.
Assumes no knowledge of edits.



- A. Tridgell, P. Mackerras (1996) "The rsync algorithm"
- T. Knauth, C. Fetzer (2013) "dsync: Efficient block-wise synchronization of multi-gigabyte binary data"
- A. Orlitsky, K. Viswanathan (2003) "One-way communication and error-correcting codes"'
- R. Venkataramanan, H. Zhang, K. Ramchandran (2010) "Interactive low-complexity codes for synchronization from deletions and insertions"
- See also next talk.

Related Work: Update Efficient Codes

Minimize number of nodes that need to be updated.
Edits vaguely viewed as substitutions.

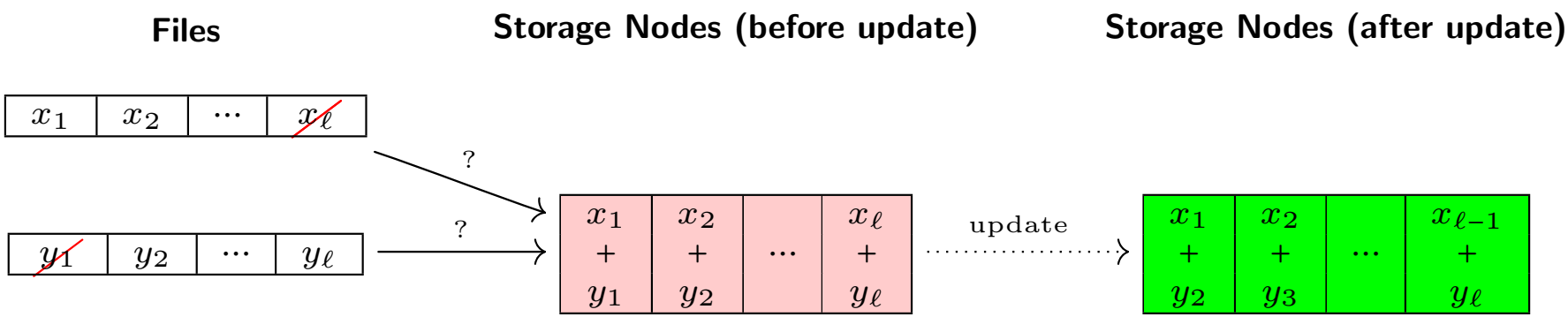


- A. S. Rawat, S. Vishwanath, A. Bhowmick, E. Soljanin (2011) "Update efficient codes for distributed storage"
- A. Mazumdar, V. Chandar, G. W. Wornell (2014) "Update-efficiency and local repairability limits for capacity approaching codes"

Problem Revisited

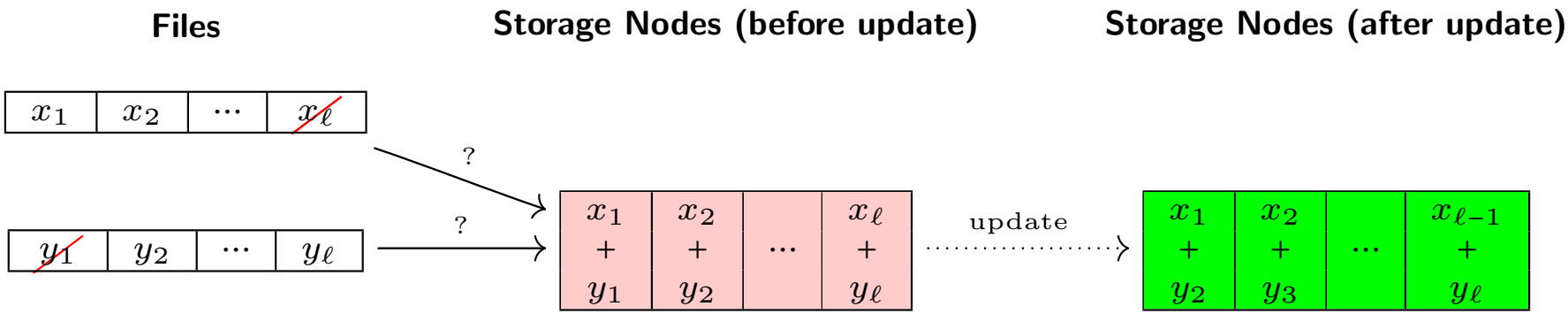
This Work

- Focuses on updating coded copies of information with minimal data transmission rates.
- Assume full knowledge of edits.
- Edits in the form of deletions and insertions.

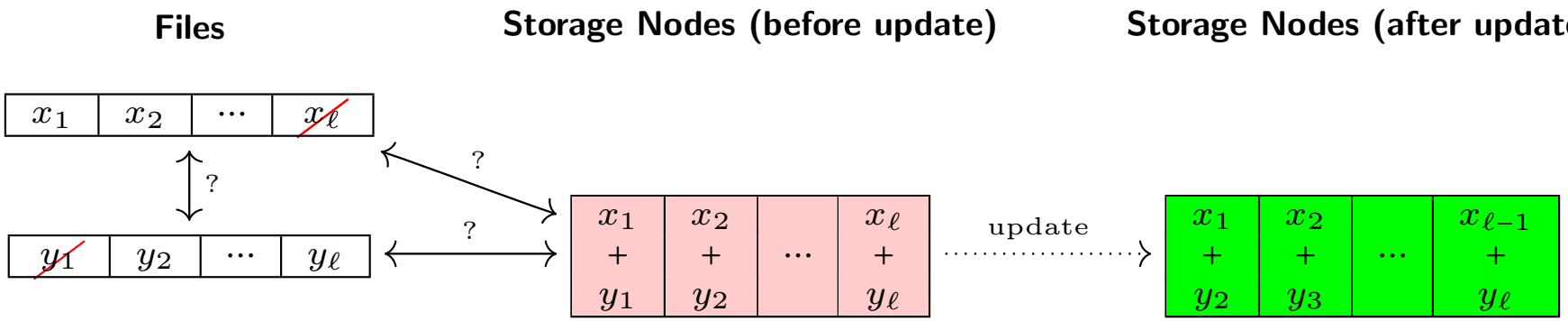


Problem Revisited: Updating Node 3

What is the minimum communication complexity needed for node 3 to update its content?



Problem Revisited: Updating Node 3



Proposition

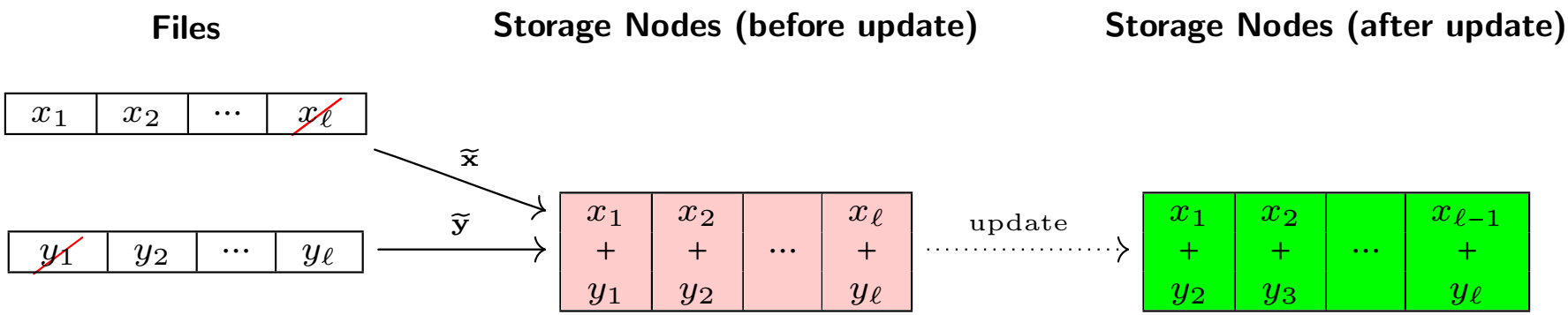
The total communication cost is at least $(\ell - 1)$ symbols, or $(\ell - 1)\log q$ bits, independent of the network topology between users and nodes.

Proof.

Fooling set method from communication complexity. □

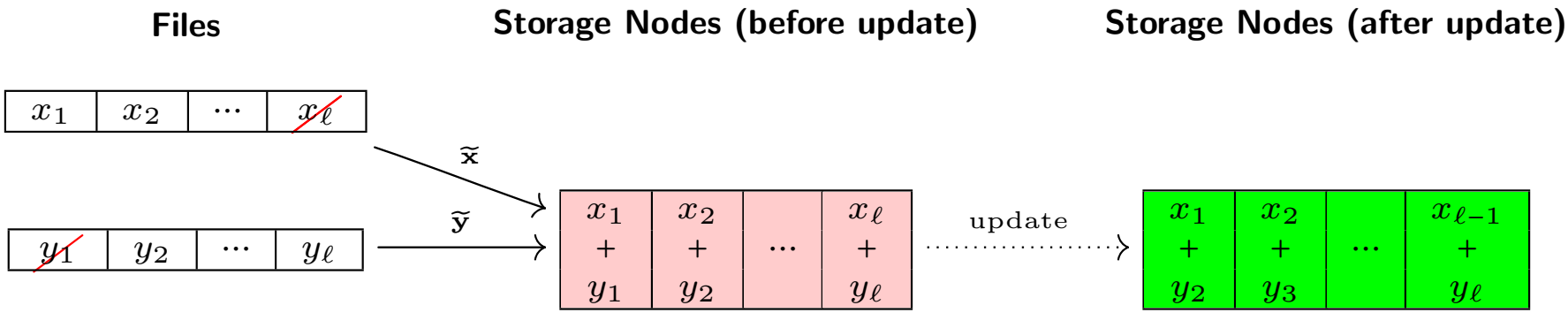
Problem Revisited: Updating Node 3

Simple scheme that achieves $\Omega(\ell \log q)$ communication cost.



Problem Revisited: Updating Node 3

Simple scheme that achieves $\Omega(\ell \log q)$ communication cost.

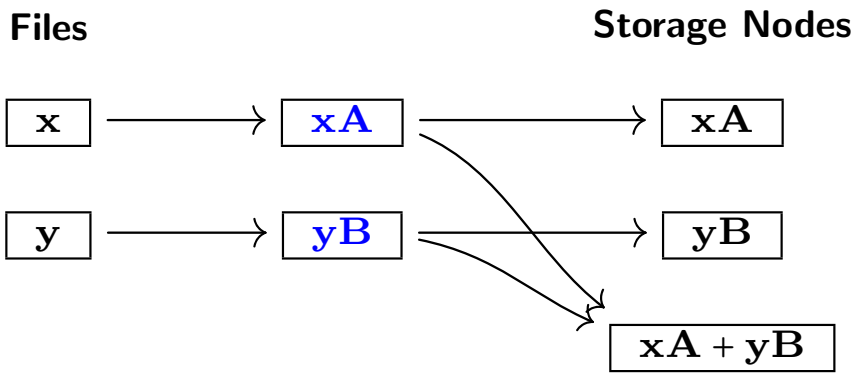


Unsatisfactory solution

- We need only $\log \ell$ bits to encode each deletion, but transmit $\ell \log q$ bits.
- Problem arose due to inflexibility in code structure.

Intermediary Encoding

Consider a new intermediary encoding scheme, where we code "modified versions of the files". Here, $\mathbf{A}$ and $\mathbf{B}$ are $\ell \times \ell$-matrices.

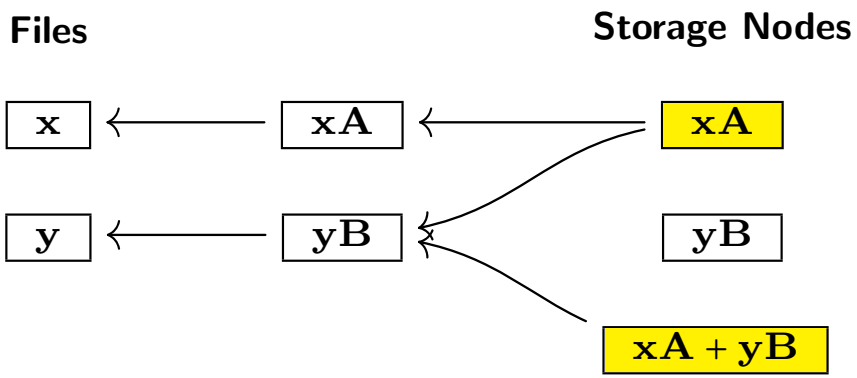


Intermediary Encoding

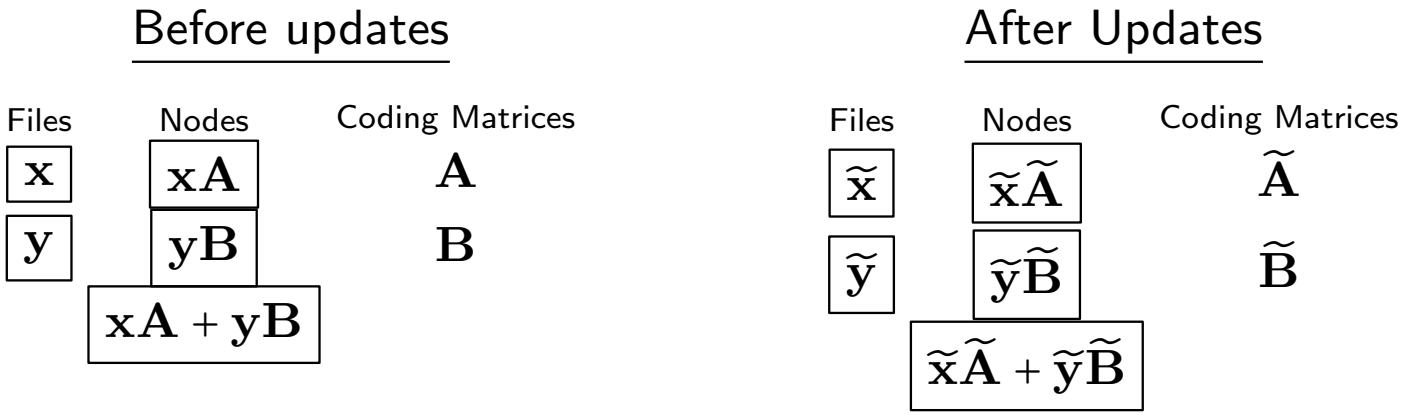
To retain reconstruction properties, we require the following.

Proposition

If $\mathbf{A}$ and $\mathbf{B}$ are invertible $\ell \times \ell$-matrices, then we have an $[3, 2]$-MDS code.



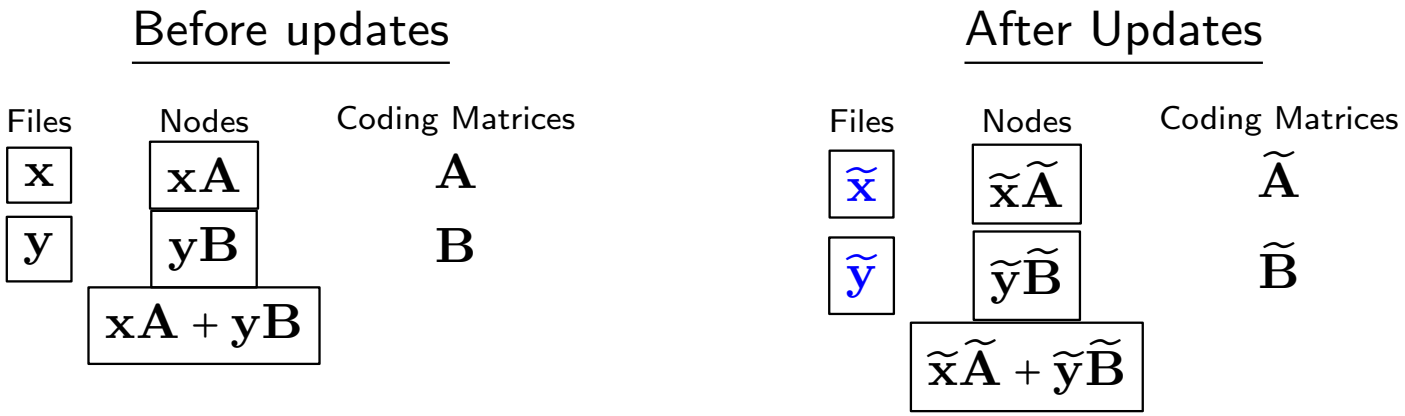
Synchronization Scheme



Idea behind approach

- Suppose $\mathbf{x}$ and $\mathbf{y}$ are edited and resulting files $\widetilde{\mathbf{x}}$, $\widetilde{\mathbf{y}}$ are of length ℓ'.
- Users $\mathbf{x}$ and $\mathbf{y}$ modify respective matrices $\mathbf{A}$ and $\mathbf{B}$ to invertible $\ell' \times \ell'$ matrices $\widetilde{\mathbf{A}}$ and $\widetilde{\mathbf{B}}$ according to edits made.
- Users transmit only locations and values of their edits.
- Storage nodes update respective information so that encoding with respect to $\widetilde{\mathbf{A}}$ and $\widetilde{\mathbf{B}}$ holds.

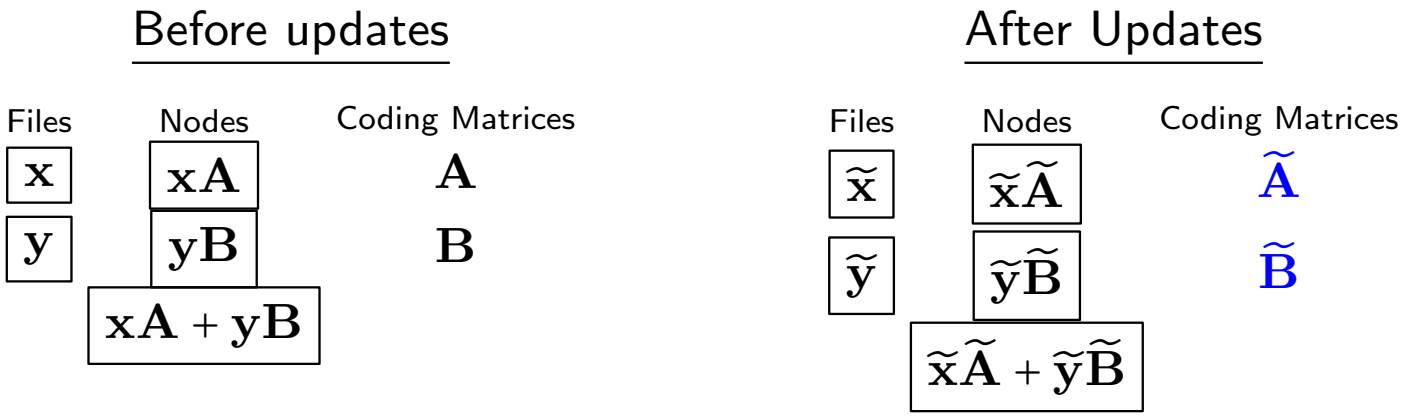
Synchronization Scheme



Idea behind approach

- Suppose $\mathbf{x}$ and $\mathbf{y}$ are edited and resulting files $\widetilde{\mathbf{x}}$, $\widetilde{\mathbf{y}}$ are of length ℓ'.
- Users $\mathbf{x}$ and $\mathbf{y}$ modify respective matrices $\mathbf{A}$ and $\mathbf{B}$ to invertible $\ell' \times \ell'$ matrices $\widetilde{\mathbf{A}}$ and $\widetilde{\mathbf{B}}$ according to edits made.
- Users transmit only locations and values of their edits.
- Storage nodes update respective information so that encoding with respect to $\widetilde{\mathbf{A}}$ and $\widetilde{\mathbf{B}}$ holds.

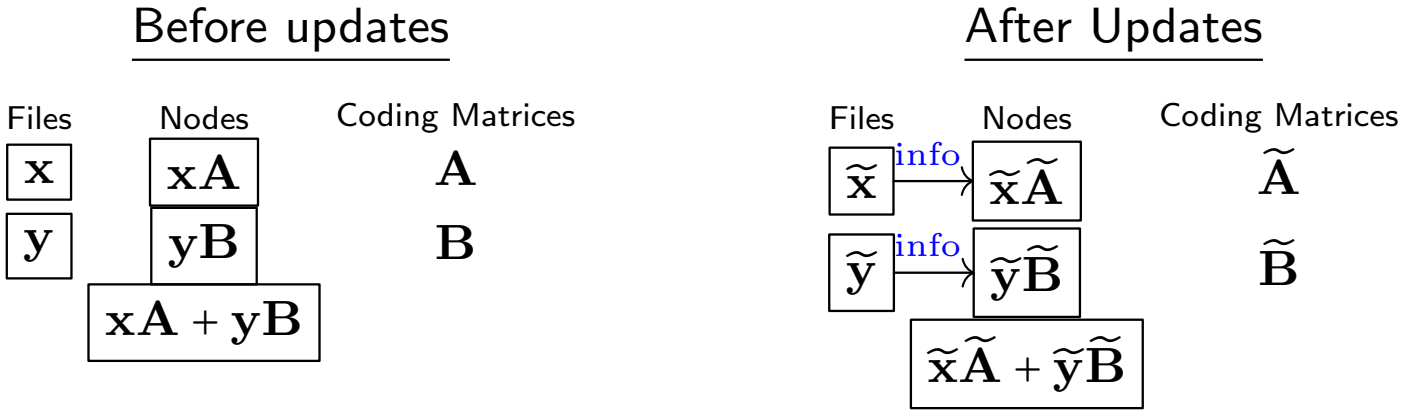
Synchronization Scheme



Idea behind approach

- Suppose $\mathbf{x}$ and $\mathbf{y}$ are edited and resulting files $\widetilde{\mathbf{x}}$, $\widetilde{\mathbf{y}}$ are of length ℓ'.
- Users $\mathbf{x}$ and $\mathbf{y}$ modify respective matrices $\mathbf{A}$ and $\mathbf{B}$ to invertible $\ell' \times \ell'$ matrices $\widetilde{\mathbf{A}}$ and $\widetilde{\mathbf{B}}$ according to edits made.
- Users transmit only locations and values of their edits.
- Storage nodes update respective information so that encoding with respect to $\widetilde{\mathbf{A}}$ and $\widetilde{\mathbf{B}}$ holds.

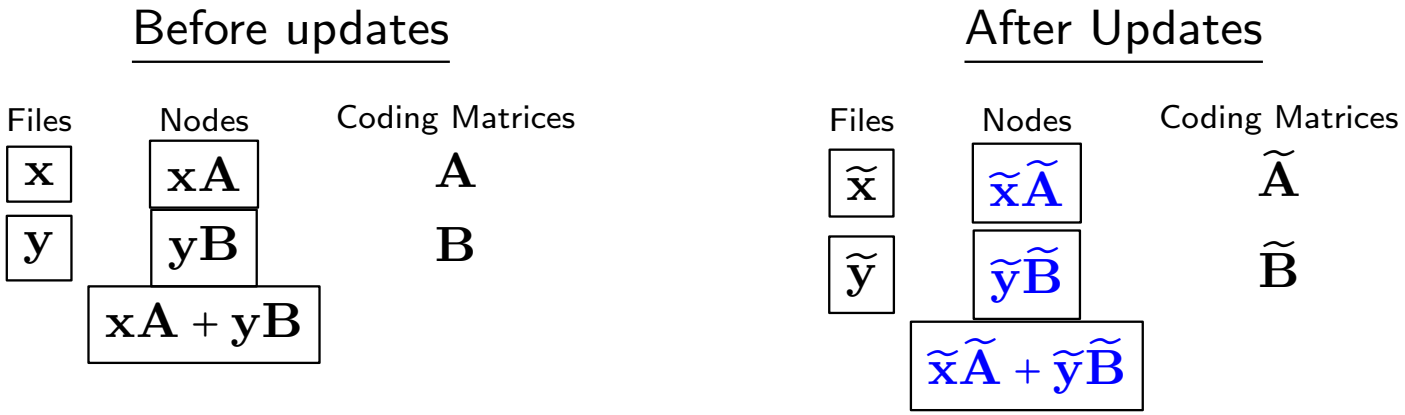
Synchronization Scheme



Idea behind approach

- Suppose $\mathbf{x}$ and $\mathbf{y}$ are edited and resulting files $\widetilde{\mathbf{x}}$, $\widetilde{\mathbf{y}}$ are of length ℓ'.
- Users $\mathbf{x}$ and $\mathbf{y}$ modify respective matrices $\mathbf{A}$ and $\mathbf{B}$ to invertible $\ell' \times \ell'$ matrices $\widetilde{\mathbf{A}}$ and $\widetilde{\mathbf{B}}$ according to edits made.
- Users transmit only locations and values of their edits.
- Storage nodes update respective information so that encoding with respect to $\widetilde{\mathbf{A}}$ and $\widetilde{\mathbf{B}}$ holds.

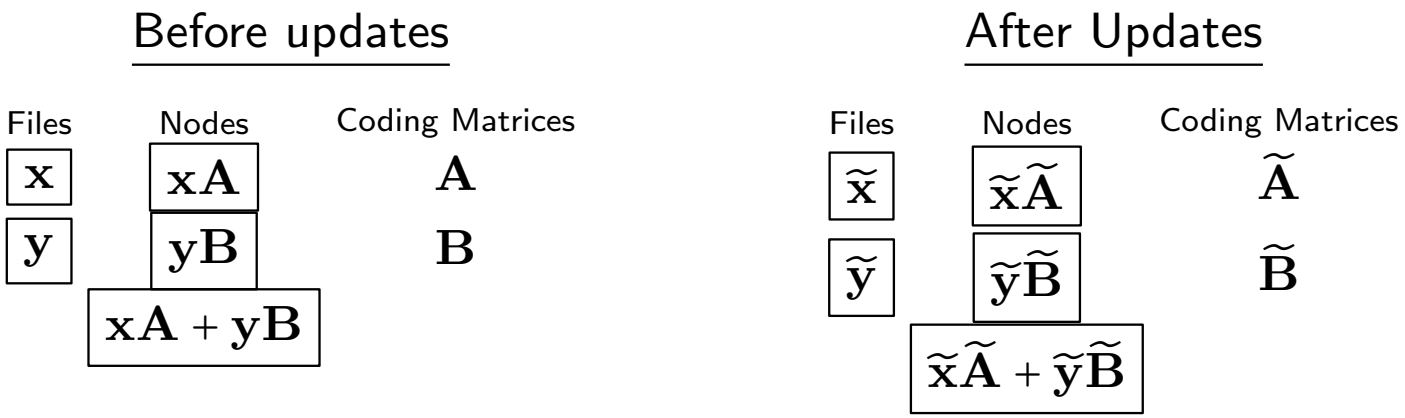
Synchronization Scheme



Idea behind approach

- Suppose $\mathbf{x}$ and $\mathbf{y}$ are edited and resulting files $\widetilde{\mathbf{x}}$, $\widetilde{\mathbf{y}}$ are of length ℓ'.
- Users $\mathbf{x}$ and $\mathbf{y}$ modify respective matrices $\mathbf{A}$ and $\mathbf{B}$ to invertible $\ell' \times \ell'$ matrices $\widetilde{\mathbf{A}}$ and $\widetilde{\mathbf{B}}$ according to edits made.
- Users transmit only locations and values of their edits.
- Storage nodes update respective information so that encoding with respect to $\widetilde{\mathbf{A}}$ and $\widetilde{\mathbf{B}}$ holds.

Synchronization Scheme



Idea behind approach

- Suppose $\mathbf{x}$ and $\mathbf{y}$ are edited and resulting files $\widetilde{\mathbf{x}}$, $\widetilde{\mathbf{y}}$ are of length ℓ'.
- Users $\mathbf{x}$ and $\mathbf{y}$ modify respective matrices $\mathbf{A}$ and $\mathbf{B}$ to invertible $\ell' \times \ell'$ matrices $\widetilde{\mathbf{A}}$ and $\widetilde{\mathbf{B}}$ according to edits made.
- Users transmit only locations and values of edits.
- Storage nodes update respective information so that encoding with respect to $\widetilde{\mathbf{A}}$ and $\widetilde{\mathbf{B}}$ holds.

Choose $\mathbf{A}$ and $\mathbf{B}$ to be Vandermonde matrices.

Synchronization Scheme

Consider a $[3, 2]$-MDS code over $\mathbb{F}_7$.

Files | **Storage Nodes** | **Intermediary coding matrices**

$\boxed{\mathbf{x} = (1, 0, 1, 0, 1)}$ $\quad \boxed{\mathbf{xA} = (3, 2, 0, 6, 0)}$ $\quad \mathbf{A} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 1 & 2 \\ 1 & 3 & 2 & 6 & 4 \\ 1 & 4 & 2 & 1 & 4 \\ 1 & 5 & 4 & 6 & 2 \end{pmatrix}$

$\boxed{\mathbf{y} = (0, 1, 0, 1, 0)}$ $\quad \boxed{\mathbf{yB} = (2, 6, 6, 2, 6)}$ $\quad \mathbf{B} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 1 & 2 \\ 1 & 3 & 2 & 6 & 4 \\ 1 & 4 & 2 & 1 & 4 \\ 1 & 5 & 4 & 6 & 2 \end{pmatrix}$

$\boxed{\mathbf{xA} + \mathbf{yB} = (5, 1, 6, 1, 6)}$

Synchronization Scheme

Consider the following edits.

Files | **Storage Nodes** | **Intermediary coding matrices**

$\widetilde{\mathbf{x}} = (1, 0, 1, 0, 1)$

$\mathbf{xA} = (3, 2, 0, 6, 0)$

$$\mathbf{A} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 1 & 2 \\ 1 & 3 & 2 & 6 & 4 \\ 1 & 4 & 2 & 1 & 4 \\ 1 & 5 & 4 & 6 & 2 \end{pmatrix}$$

$\widetilde{\mathbf{y}} = (0, 1, 0, 1, 0)$

$\mathbf{yB} = (2, 6, 6, 2, 6)$

$$\mathbf{B} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 1 & 2 \\ 1 & 3 & 2 & 6 & 4 \\ 1 & 4 & 2 & 1 & 4 \\ 1 & 5 & 4 & 6 & 2 \end{pmatrix}$$

$\mathbf{xA} + \mathbf{yB} = (5, 1, 6, 1, 6)$

Synchronization Scheme

Modify the $\mathbf{A}$ and $\mathbf{B}$.

- Remove the row corresponding to the deleted position.
- Remove the last column.

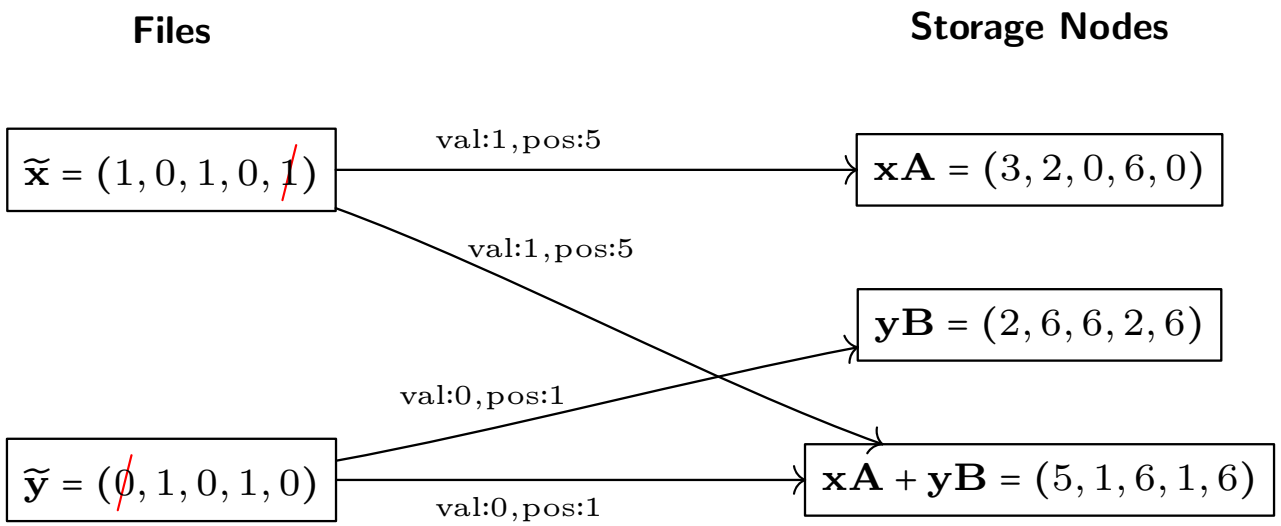
Files **Intermediary coding matrices**

$$\widetilde{\mathbf{x}} = (1, 0, 1, 0, \cancel{1}) \qquad \widetilde{\mathbf{A}} = \begin{pmatrix} 1 & 1 & 1 & 1 & \cancel{1} \\ 1 & 2 & 4 & 1 & \cancel{2} \\ 1 & 3 & 2 & 6 & \cancel{4} \\ 1 & 4 & 2 & 1 & \cancel{4} \\ \cancel{1} & \cancel{5} & \cancel{4} & \cancel{6} & \cancel{2} \end{pmatrix}$$

$$\widetilde{\mathbf{y}} = (\cancel{0}, 1, 0, 1, 0) \qquad \widetilde{\mathbf{B}} = \begin{pmatrix} \cancel{1} & \cancel{1} & \cancel{1} & \cancel{1} & \cancel{1} \\ 1 & 2 & 4 & 1 & \cancel{2} \\ 1 & 3 & 2 & 6 & \cancel{4} \\ 1 & 4 & 2 & 1 & \cancel{4} \\ 1 & 5 & 4 & 6 & \cancel{2} \end{pmatrix}$$

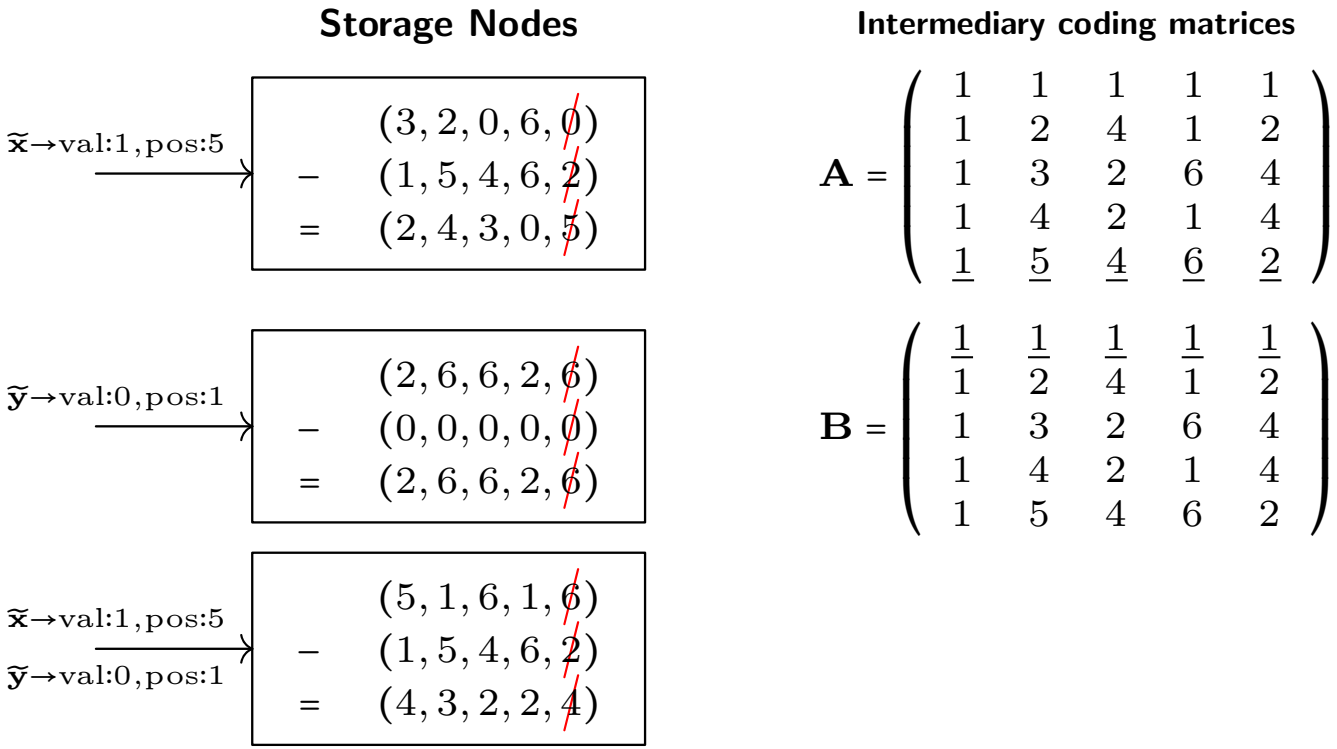
Synchronization Scheme

Users transmit deleted values ($\log q$ bits) and positions ($\log \ell$ bits).



Synchronization Scheme

Nodes update their values.



Intermediary coding matrices

$$\mathbf{A} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 1 & 2 \\ 1 & 3 & 2 & 6 & 4 \\ 1 & 4 & 2 & 1 & 4 \\ \underline{1} & \underline{5} & \underline{4} & \underline{6} & \underline{2} \end{pmatrix}$$

$$\mathbf{B} = \begin{pmatrix} \frac{1}{1} & \frac{1}{2} & \frac{1}{4} & \frac{1}{1} & \frac{1}{2} \\ 1 & 3 & 2 & 6 & 4 \\ 1 & 4 & 2 & 1 & 4 \\ 1 & 5 & 4 & 6 & 2 \end{pmatrix}$$

Synchronization Scheme

Edited files and storage nodes updated as desired.

Files

$$\boxed{\widetilde{\mathbf{x}} = (1, 0, 1, 0)}$$

$$\boxed{\widetilde{\mathbf{y}} = (1, 0, 1, 0)}$$

Storage Nodes

$$\boxed{\widetilde{\mathbf{x}}\widetilde{\mathbf{A}} = (2, 4, 3, 0)}$$

$$\boxed{\widetilde{\mathbf{y}}\widetilde{\mathbf{B}} = (2, 6, 6, 2)}$$

$$\boxed{\widetilde{\mathbf{x}}\widetilde{\mathbf{A}} + \widetilde{\mathbf{y}}\widetilde{\mathbf{B}} = (4, 3, 2, 2)}$$

Intermediary coding matrices

$$\widetilde{\mathbf{A}} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 1 \\ 1 & 3 & 2 & 6 \\ 1 & 4 & 2 & 1 \end{pmatrix}$$

$$\widetilde{\mathbf{B}} = \begin{pmatrix} 1 & 2 & 4 & 1 \\ 1 & 3 & 2 & 6 \\ 1 & 4 & 2 & 1 \\ 1 & 5 & 4 & 6 \end{pmatrix}$$

Synchronization Scheme

Edited files and storage nodes updated as desired.

Files	Storage Nodes	Intermediary coding matrices
$\widetilde{\mathbf{x}} = (1,0,1,0)$	$\widetilde{\mathbf{x}}\widetilde{\mathbf{A}} = (2,4,3,0)$	$\widetilde{\mathbf{A}} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 1 \\ 1 & 3 & 2 & 6 \\ 1 & 4 & 2 & 1 \end{pmatrix}$
$\widetilde{\mathbf{y}} = (1,0,1,0)$	$\widetilde{\mathbf{y}}\widetilde{\mathbf{B}} = (2,6,6,2)$	$\widetilde{\mathbf{B}} = \begin{pmatrix} 1 & 2 & 4 & 1 \\ 1 & 3 & 2 & 6 \\ 1 & 4 & 2 & 1 \\ 1 & 5 & 4 & 6 \end{pmatrix}$

$$\widetilde{\mathbf{x}}\widetilde{\mathbf{A}} + \widetilde{\mathbf{y}}\widetilde{\mathbf{B}} = (4,3,2,2)$$

- $\widetilde{\mathbf{A}}$ and $\widetilde{\mathbf{B}}$ remain Vandermonde and hence, invertible.
- Communication cost between each user and his connected node: $\log q + \log \ell$ bits.

Extensions and Generalizations

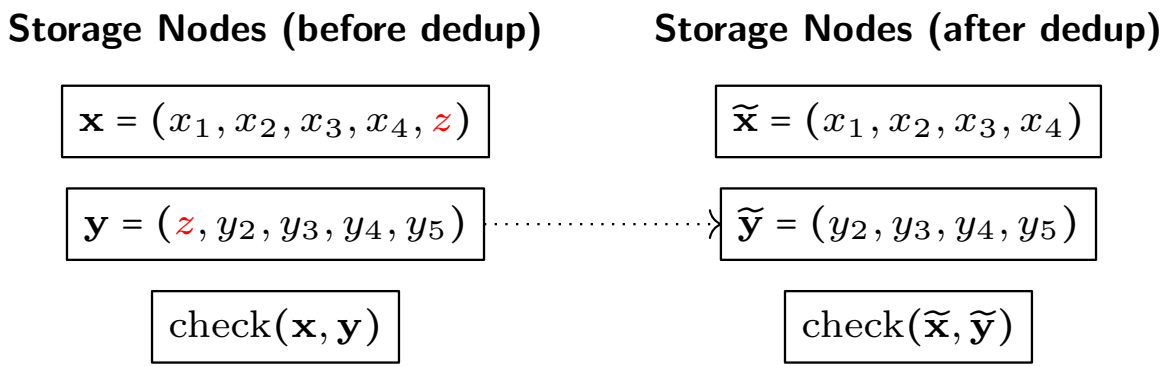
- We can use permutation or Cauchy matrices, instead of Vandermonde matrices, and achieve approximately the same communication complexity.

Extensions and Generalizations

- We can use permutation or Cauchy matrices, instead of Vandermonde matrices, and achieve approximately the same communication complexity.
- Scheme easily extend to any $[n, k]$-MDS codes and more generally, any linear regenerating and locally repairable codes.

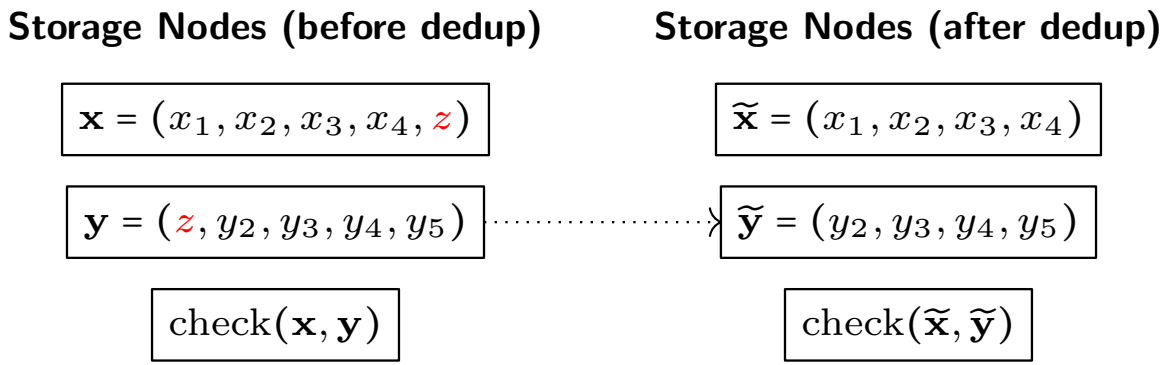
Extensions and Generalizations

- We can use permutation or Cauchy matrices, instead of Vandermonde matrices, and achieve approximately the same communication complexity.
- Scheme easily extend to any $[n, k]$-MDS codes and more generally, any linear regenerating and locally repairable codes.
- Has applications to data deduplication.



Extensions and Generalizations

- We can use permutation or Cauchy matrices, instead of Vandermonde matrices, and achieve approximately the same communication complexity.
- Scheme easily extend to any $[n, k]$-MDS codes and more generally, any linear regenerating and locally repairable codes.
- Has applications to data deduplication.



- More extensions: systematic encoding, encoding files of variable lengths, non-uniform deletions, . . .

Storage Overhead

- In intermediary coding, a user needs to store the associated matrix $\mathbf{A}$. Hence, a user *a priori* requires $\ell^2 \log q$ bits to store this matrix.

$$\mathbf{x} = (1, 0, 1, 0, 1) \quad \mathbf{A} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 1 & 2 \\ 1 & \mathbf{3} & 2 & 6 & 4 \\ 1 & 4 & 2 & 1 & 4 \\ 1 & 5 & 4 & 6 & 2 \end{pmatrix}$$

Storage Overhead

- In intermediary coding, a user needs to store the associated matrix $\mathbf{A}$. Hence, a user *a priori* requires $\ell^2 \log q$ bits to store this matrix.

$$\mathbf{x} = (1,0,1,0,1) \quad \mathbf{A} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 1 & 2 \\ 1 & \mathbf{3} & 2 & 6 & 4 \\ 1 & 4 & 2 & 1 & 4 \\ 1 & 5 & 4 & 6 & 2 \end{pmatrix}$$

- This stringent storage requirement may be easily relaxed. We first store the description of the initial matrices. Subsequently, to generate the matrices $\mathbf{A}^{(s)}$, it suffices for the users to store the modifications to the initial matrices.

$$\mathbf{x} = (1,0,1,0,1) \qquad \mathbf{A} \leftarrow 5 \times 5 \text{ Vandermonde matrix over } \mathbb{F}_7$$
$$\widetilde{\mathbf{x}} = (1,0,1,0) \qquad \widetilde{\mathbf{A}} \leftarrow \mathbf{A} \text{ with fifth row removed}$$

- Term this information the storage overhead per edit.

Storage Overhead

- In intermediary coding, a user needs to store the associated matrix $\mathbf{A}$. Hence, a user *a priori* requires $\ell^2 \log q$ bits to store this matrix.

$$\mathbf{x} = (1,0,1,0,1) \quad \mathbf{A} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 1 & 2 \\ 1 & \mathbf{3} & 2 & 6 & 4 \\ 1 & 4 & 2 & 1 & 4 \\ 1 & 5 & 4 & 6 & 2 \end{pmatrix}$$

- This stringent storage requirement may be easily relaxed. We first store the description of the initial matrices. Subsequently, to generate the matrices $\mathbf{A}^{(s)}$, it suffices for the users to store the modifications to the initial matrices.

$$\mathbf{x} = (1,0,1,0,1) \qquad \mathbf{A} \leftarrow 5 \times 5 \text{ Vandermonde matrix over } \mathbb{F}_7$$
$$\widetilde{\mathbf{x}} = (1,0,1,0) \qquad \widetilde{\mathbf{A}} \leftarrow \mathbf{A} \text{ with fifth row removed}$$

- Term this information the storage overhead per edit.
- Scheme V: storage overhead is $\log \ell$.

Scheme V: Synchronization scheme with intermediary encoding using Vandermonde matrices.

Storage Overhead and Communication Complexity Tradeoff

	Scheme T	Scheme V
Communication cost	$(\ell - 1) \log q$ (worst case)	$\log \ell + \log q$
Storage overhead	0	$\log \ell$

Scheme T: Synchronization scheme without intermediary encoding.

Scheme V: Synchronization scheme with intermediary encoding using Vandermonde matrices.

Storage Overhead and Communication Complexity Tradeoff

	Scheme T	Scheme V
Communication cost	$(\ell - 1)\log q$ (worst case)	$\log \ell + \log q$
Storage overhead	0	$\log \ell$

Scheme T: Synchronization scheme without intermediary encoding.

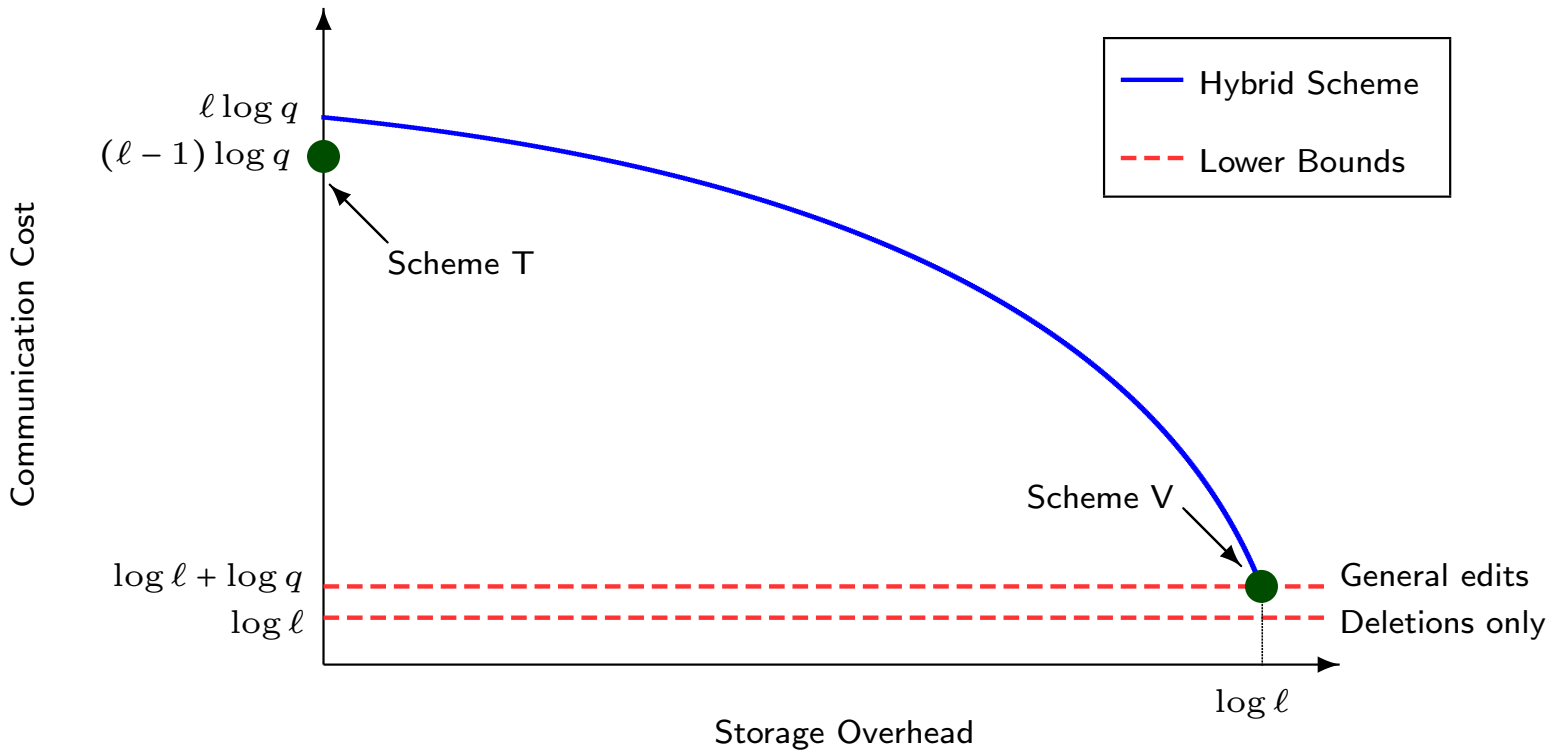
Scheme V: Synchronization scheme with intermediary encoding using Vandermonde matrices.

Storage overhead versus information storage

- Suppose d edits.
- Scheme V incur a total storage overhead of $d \log \ell$ bits.
- File itself is of size $\ell \log q$ bits.
- For desirable storage allocation properties, one would want $d \log \ell = o(\ell)$ or the number of edits to be $o(\ell / \log \ell)$.
- Scheme V should be used only in the small/moderate edit regime.

Storage Overhead and Communication Complexity Tradeoff

Suppose that we are given a constraint on the storage overhead.
What is the minimum communication cost possible?



Conclusion

- Examined problem of synchronizing edits in a distributed storage environment.
- Proposed synchronization schemes with low communication complexity that made use of intermediary coding.
- Accommodates a broad family of coding schemes.
- More extensions of the edit models and an average case analysis is presented in full paper.
- Full paper available at arxiv:

Synchronizing Edits in Distributed Storage Networks

Questions?