Oblivious Updates in Storage Networks

Preetum Nakkiran, Nihar B. Shah, K. V. Rashmi

Berkeley
UNIVERSITY OF CALIFORNIA

Distributed Storage

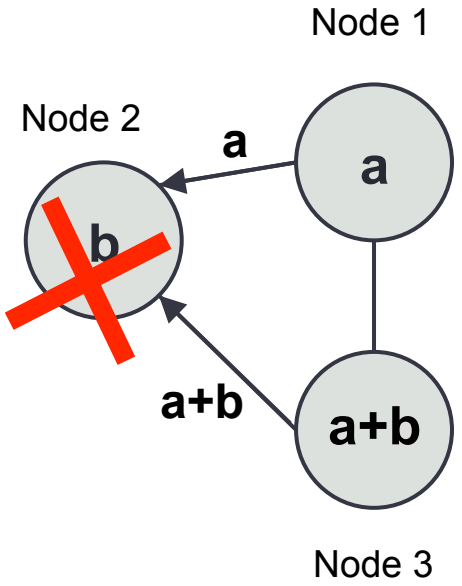
- Store data across many machines
  - Fault-tolerant (machine failures)

- Applications:
  - Data-centers
  - Peer-to-peer (P2P) storage
  - Caching networks

Prior Work

- Considerable recent work on repair of failed nodes
  - [Tamo et al. '14, Papailiopoulos et al. '14, Cadambe et al. '11, Shum et al. '11, Oggier et al. '11, Rawat et al. '12, Gopalan et al. '11 ,Sasidharan et al. '13, Rouayheb et al. '10, Prakash et al. '12]
- Optimize
  - Total storage overhead
  - Bandwidth required to 'repair' failed nodes
- Tradeoff well understood:
  - **Replication**: poor storage / best bandwidth
  - **Reed-Solomon:** best storage / poor bandwidth
  - **Regenerating codes**: arbitrary tradeoff

Reed-Solomon: Store file [a, b] on 3 nodes, tolerating 1 failure.



Mutable Data

Data changes, all nodes need to update.

- Efficient updates well-studied (“low-density” codes)
  - [Blaum, Roth ’99, Xu et al. ’99, Plank ’09, Tamo et al. ‘12]
- **Centralized** update
  - Central controller directly contacts all nodes

Oblivious Updates

Can updates be decentralized?

(Nodes update using other nodes)

Oblivious Update Model

- Some nodes have updated data
- No central controller

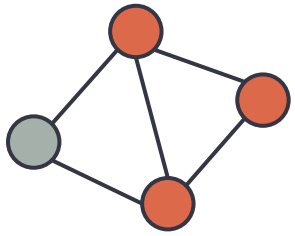
- Oblivious Update
  - *Everyone knows something changed; No one knows what*
  - No changelogs (no storage overhead; decentralized)

  - "Stale" node downloads from updated nodes
  - Uses downloaded-data + stale-data → updated-data

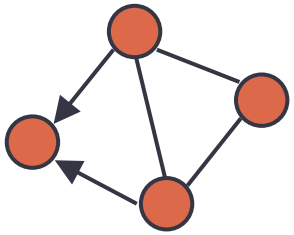
Legend

Stale data

Updated data



initial



oblivious update

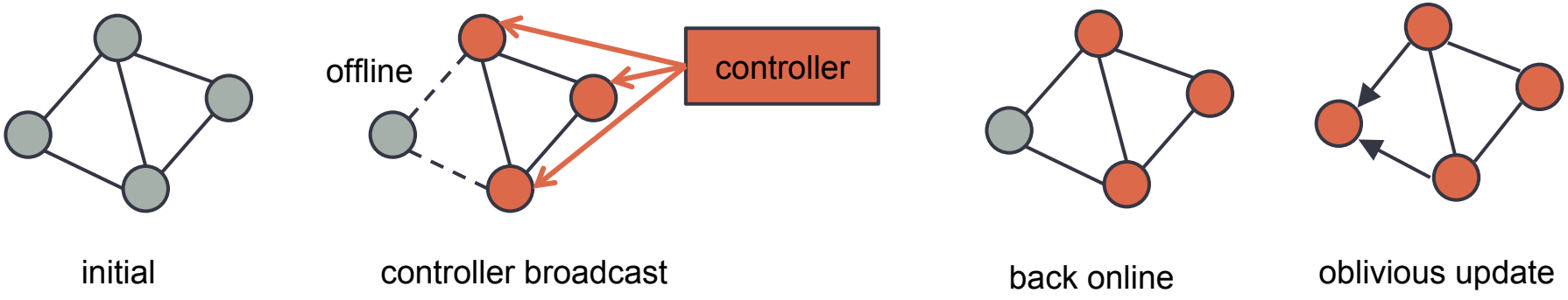
Oblivious Update Model

- Linear encoding
  - File comprises symbols from a finite field
- Update: File changes by “small amount”
- Two classes of codes:
  1. Arbitrary linear codes
  2. MDS (n, k) linear codes

Sample Scenario

Some nodes offline during controller broadcast of update

1. All nodes store initial encoded data
2. Some nodes go offline
3. Controller broadcasts an update
4. Nodes come back online
   - Missed the broadcast
   - Oblivious update



Oblivious Update vs. Node Repair

- Node repair
  - Assumes total node failure – no useful stored data
  - Requires downloading at least as much as node data
- Oblivious update
  - Stale node has stale data – potentially useful

Can we do (much) *better* than node repair?

Our Results: Lower-Bounds for Linear Codes

Theorem

For any linear code, oblivious update requires the stale node to download at least **2x** changed symbols.

- Total download ≥ **2** × (change size)
- Helpers are allowed to send non-linear functions of the data

Our Results: Lower-Bounds for Linear MDS Codes

Theorem

For any linear MDS code, oblivious update requires the stale node to download at least **2x** changed symbols from **each of k helpers**.

- Total download ≥ **2k** × (change size)
- Helpers are allowed to send non-linear functions of the data

Lower Bounds: Powerful Genie

- Proofs use powerful genie
- Linear codes
  - Gives stale node entire stale file
  - Gives helpers entire updated file
- Linear MDS codes
  - Gives stale node entire stale file, and entire data of first (k-1) helpers
  - Gives last helper entire updated file

Can any code achieve this, without a genie?

Our Results: Linear Constructions

Theorem

There exist linear codes that support single-symbol oblivious updates by connecting to **any 2 helpers** and downloading **1 symbol from each.**

- Total download = **2 symbols**
- Optimal w.r.t. storage/bandwidth tradeoff

Our Results: Linear MDS Constructions

Theorem

There exist linear MDS codes that support single-symbol oblivious updates by connecting to **any k helpers** and downloading **2 symbols from each.**

- Total download = **2k symbols**
- Optimal w.r.t. storage/bandwidth tradeoff

Some Numbers

- File stored across 16 nodes, can recover from any k=8 nodes
- Update: single-symbol changed

MDS	Storage / node	Node Repair download (MSR)	Oblivious Update download (our construction)
File size = 1024 symbols	128 symbols	**240 symbols**	**16 symbols**
File size = 65536 symbols	8192 symbols	**15360 symbols**	**16 symbols**

MBR	Storage / node	Node Repair download (MBR)	Oblivious Update download (our construction)
File size = 1024 symbols	228 symbols	**228 symbols**	**2 symbols**
File size = 65536 symbols	14563 symbols	**14563 symbols**	**2 symbols**

Toy Example: Oblivious Update Algorithm

initial configuration

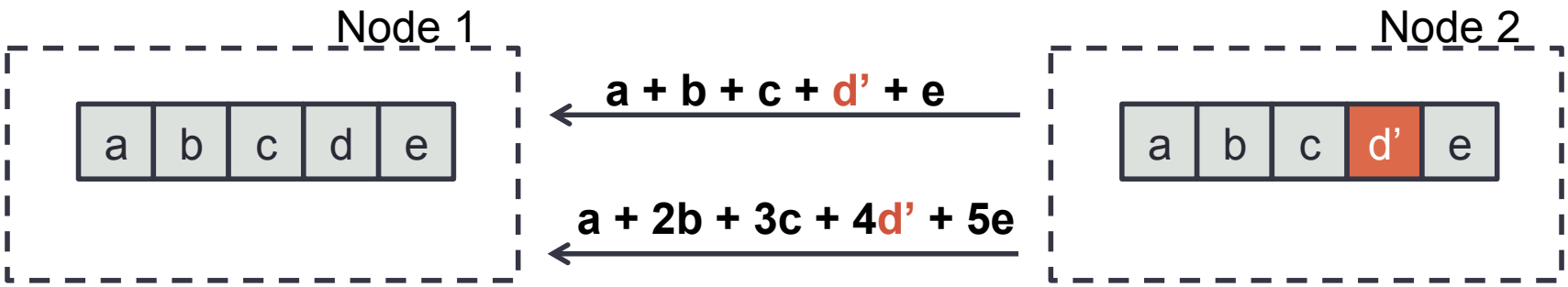
What does an oblivious update look like for Replication?

Toy Example: Oblivious Update Algorithm

update

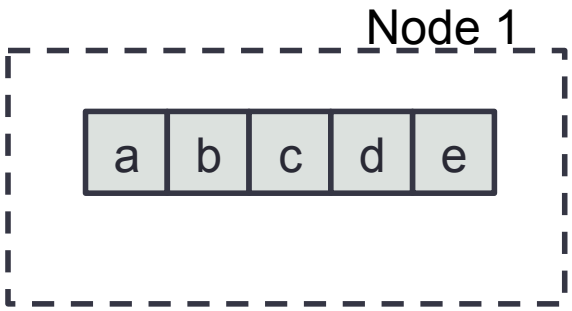
Toy Example: Oblivious Update Algorithm

oblivious update



- Node 2 does not remember that ‘d’ changed.

Toy Example: Oblivious Update Algorithm



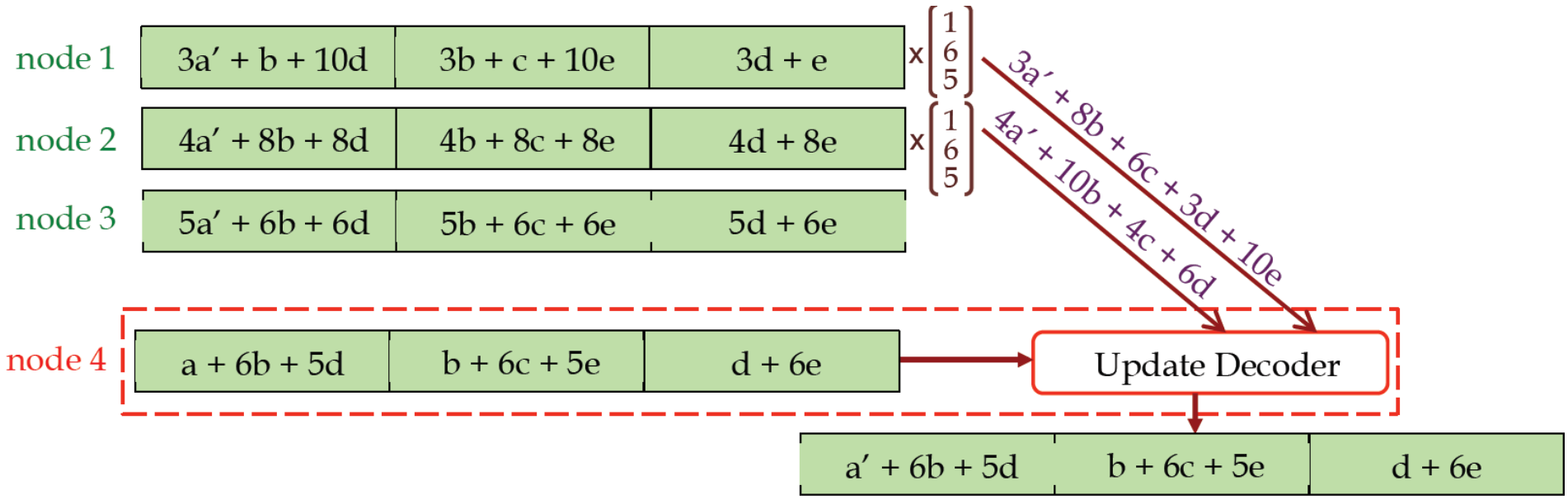
Update algorithm of Stale node:

- Receives symbols:
  - **a + b + c + d' + e**
  - **a + 2b + 3c + 4d' + 5e**
- Stores symbols [a, b, c, d, e]

- Computes symbols:
  - **a + b + c + d + e**
  - **a + 2b + 3c + 4d + 5e**
- Computes differences:
  - (d' - d)
  - 4(d' - d)

- Identifies changed symbol 'd' from ratio (4).

Example Construction



Original File: [a, b, c, d, e] (in F_{11})

- Each node stores 3 (encoded) symbols.
- File recoverable from any 2 nodes.
- Optimally handles node failure (connecting to 3 nodes).
- Based on Product-Matrix codes
  [Rashmi et al. '11]

Summary

- Proposed *oblivious updates* in storage networks
  - Decentralized, no change logs
- Derived **lower-bounds** on communication for oblivious updates
- Constructed **optimal codes** that enable oblivious updates (of one symbol)
  - Meet lower-bounds: optimal w.r.t. oblivious-update bandwidth
  - Optimal w.r.t storage/bandwidth
- Established **fundamental communication limits** for oblivious updates with linear codes.

Open Problems

- Lower-bounds
  - Non-linear codes
  - Interactive protocols
- Constructions
  - Support > 1 symbol change
- Goal: Establish *capacity* of oblivious updates

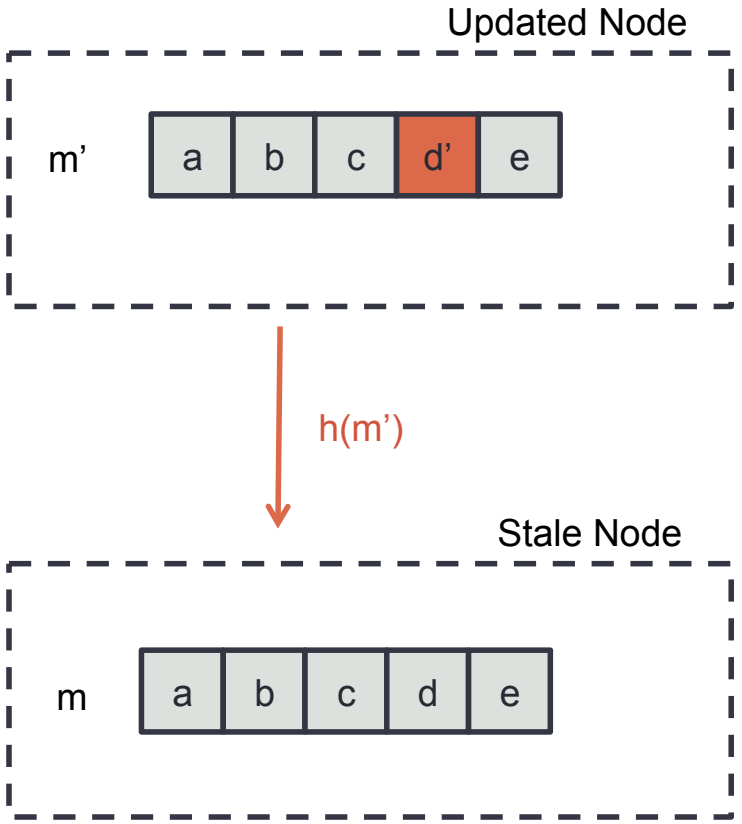
Thanks!

Questions?

{preetum,nihar,rashmikv}@berkeley.edu

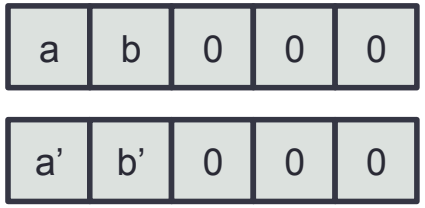
Proof Ideas: Toy Example

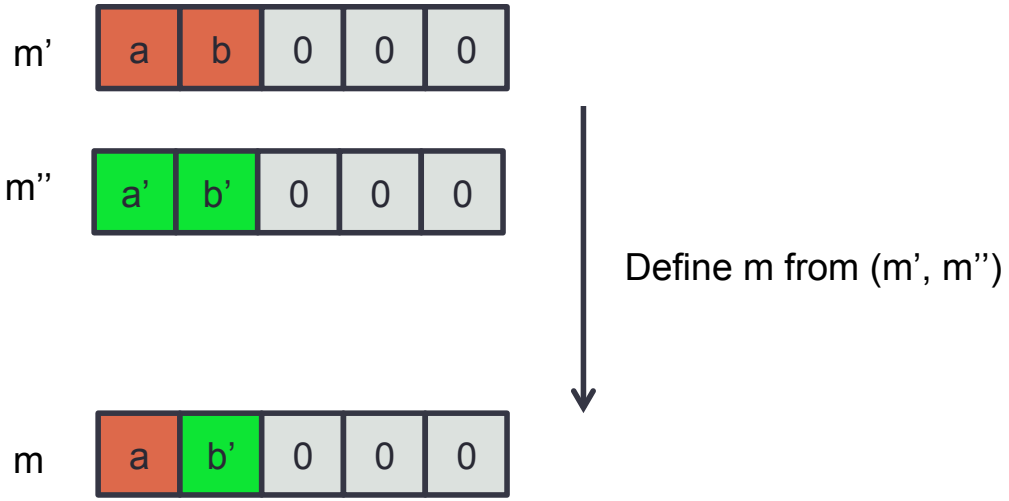
- Replication Code, 1 symbol changed.
- Will prove: Stale node must download at least 2 symbols.

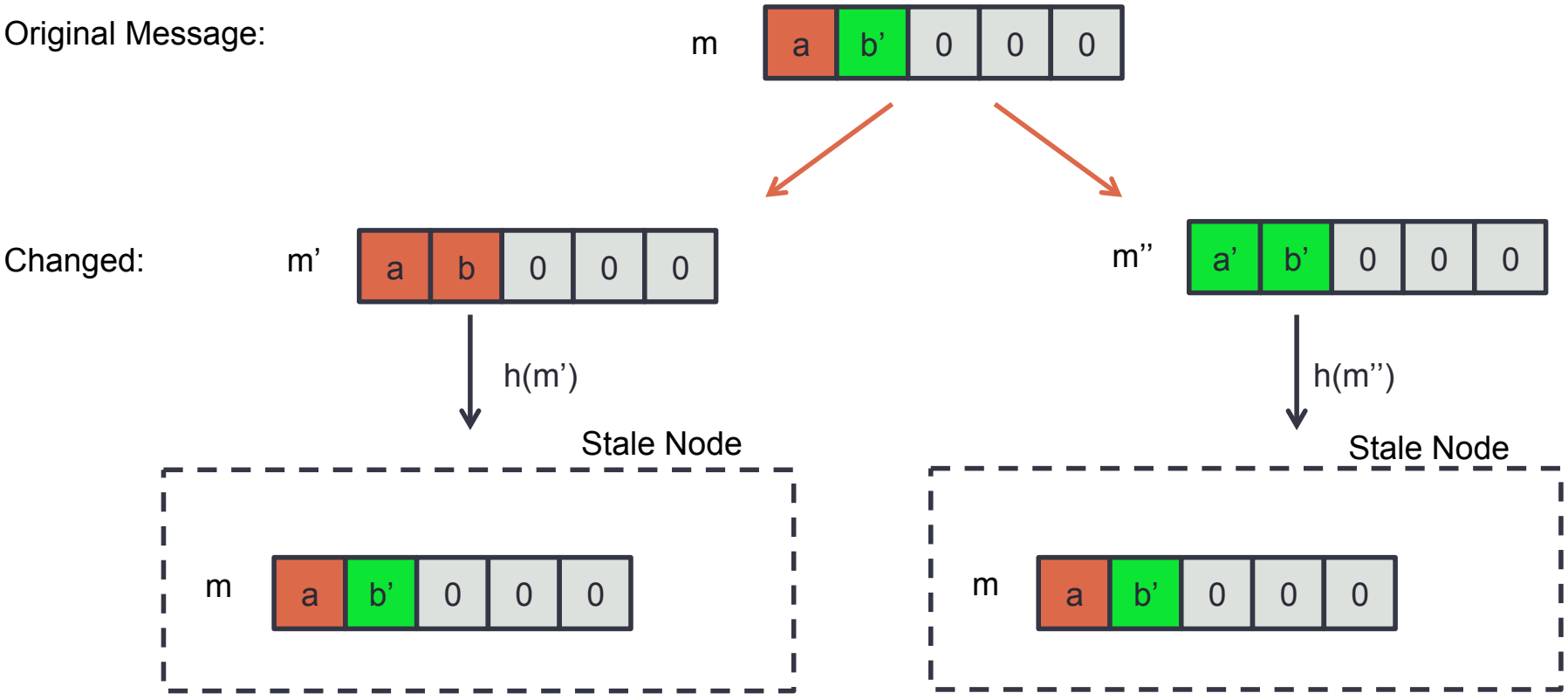


Proof Ideas: Toy Example

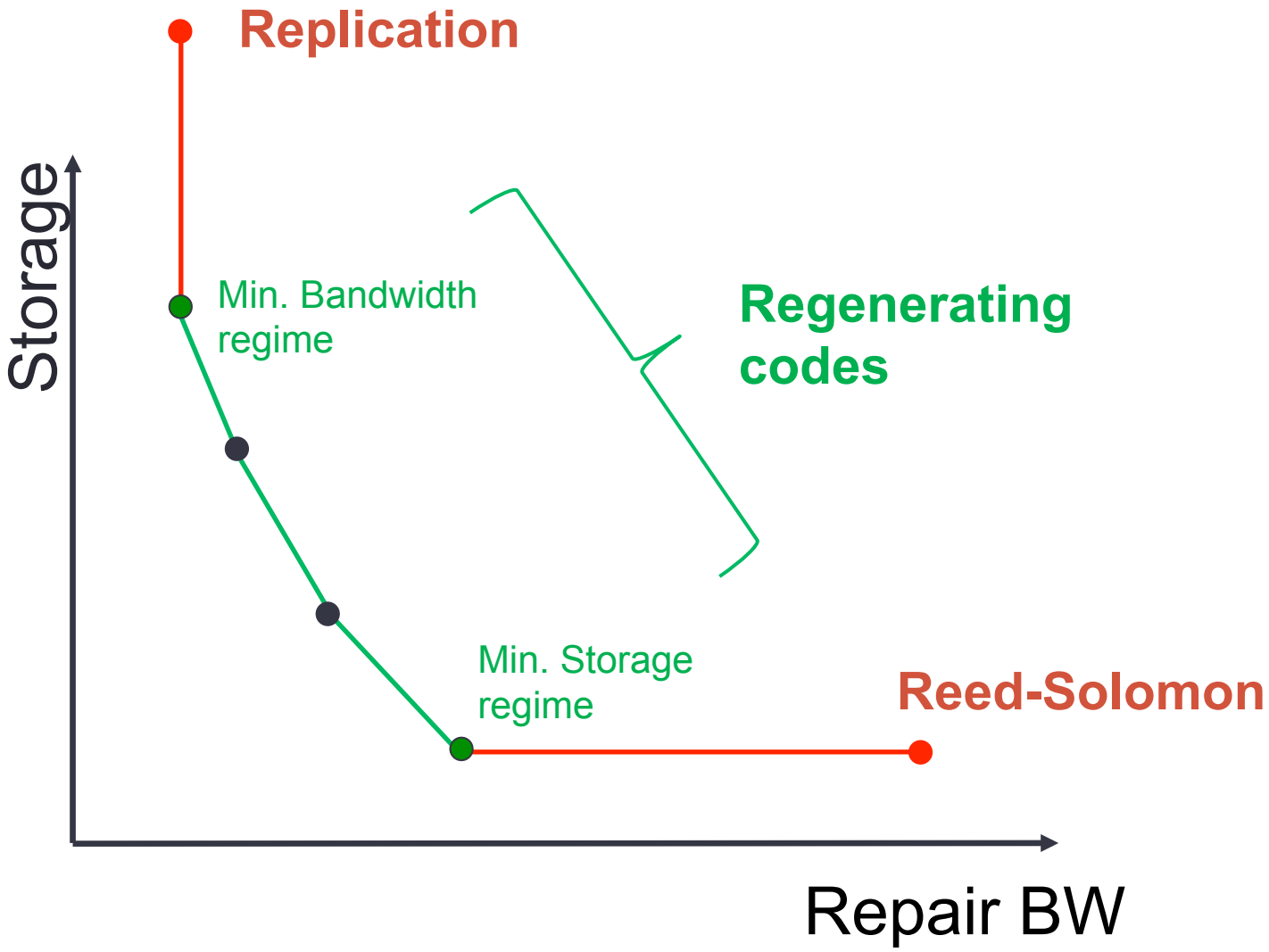
- Consider all messages using only first two symbols.
- If Range(h) < 2 symbols, h must take same value for some two messages in this set: h(m') = h(m'')







Need h(m') ≠ h(m'').
Contradiction!



[Dimakis et. al ’07]
[Rashmi et al.’09, Shah et al. ’10]