CCSW ’10

Remote Data Checking for Network Coding-based Distributed Storage Systems

Bo Chen, Reza Curtmola, Giuseppe Ateniese, Randal Burns

New Jersey Institute of Technology

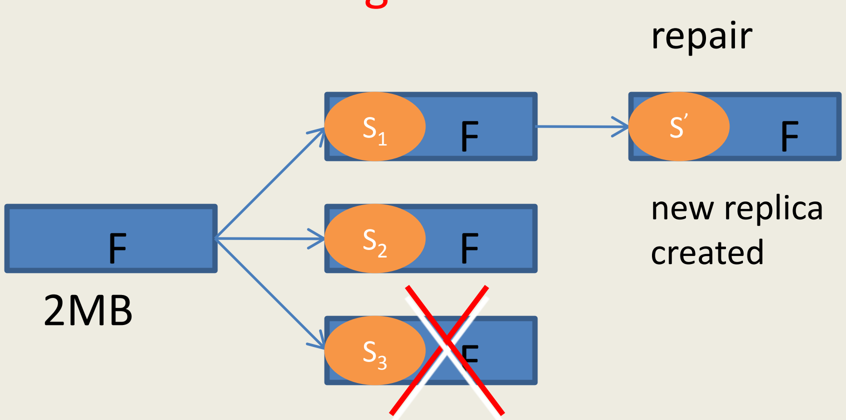
Johns Hopkins University

Motivation

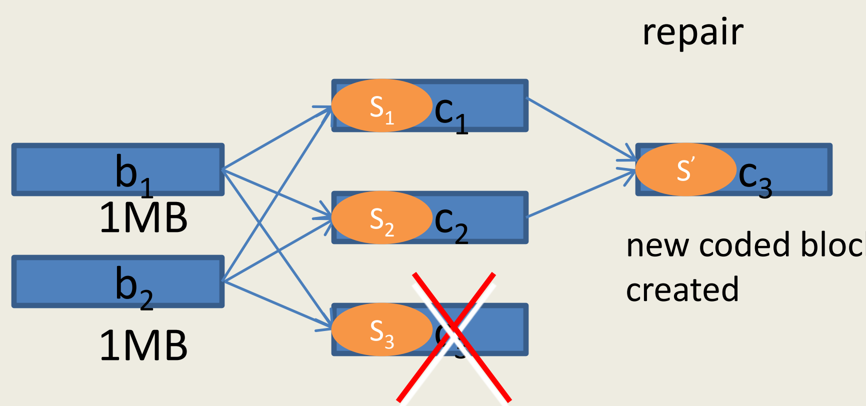
- Cloud storage can release people from the burden of hardware management
- Reduce the cost (storage as a service, pay as you use)
- Increased reliability

Reliability in Distributed Storage Systems

- Traditional approaches to store data redundantly at multiple servers:
  - Replication
  - Erasure Coding
    - Reduced storage overhead
    - Large bandwidth overhead for repair (entire file is retrieved)



Replication

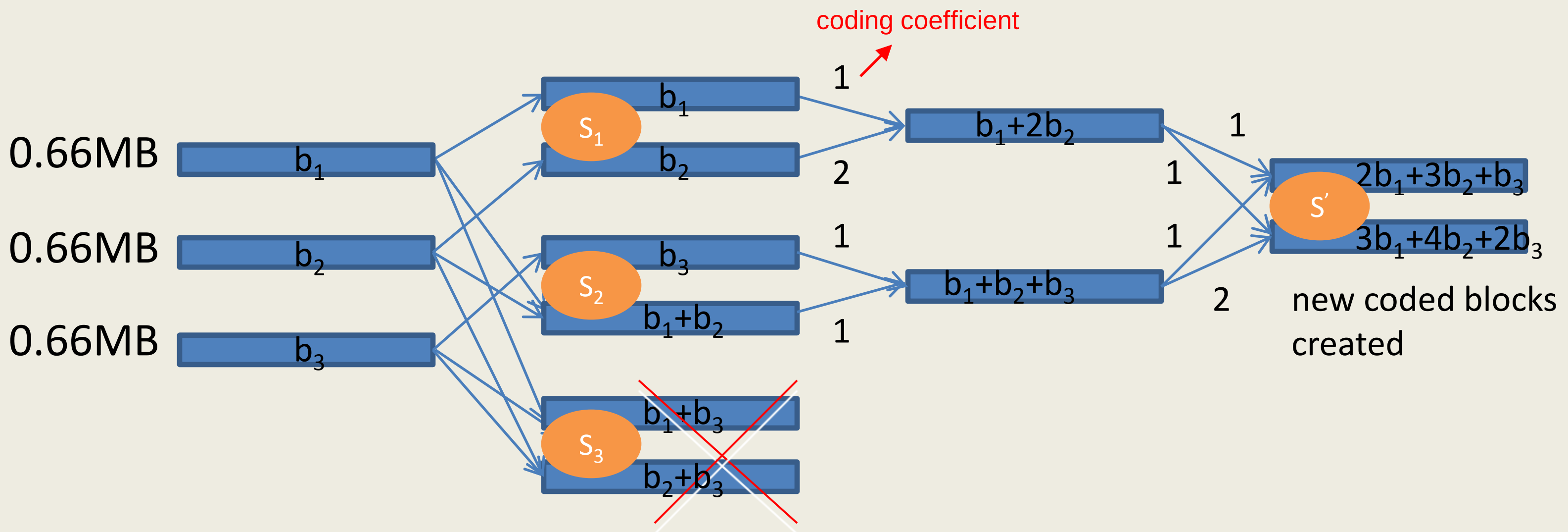


Any 2 blocks are enough to recover original file

(3,2) Erasure coding

Reliability based on Network Coding

- Network Coding (Regenerating Code): a new coding method that sacrifices some storage overhead for repair bandwidth
  - Compute coded blocks as linear combinations of original blocks
  - Repair bandwidth is optimal (retrieve x bits to repair x bits)



Network coding (n=3, k=2)

Applications that benefit from network coding

- Applications with read-rarely workloads benefit most from the low bandwidth repair overhead of network coding:
  - Regulatory storage
  - Data escrow
  - Deep archival stores
  - Preservation systems for old datasets

The Need for Remote Data Integrity Checking

- What if storage servers are not trusted?

- Client must ensure storage servers don't misbehave
- Client periodically checks integrity of outsourced data (challenge phase)
- Client takes action (repair) upon detecting corruption at one of the storage servers (repair phase)

Performance Comparison

	Replication (MR-PDP) [CKBA 08]	(n, k) Erasure Coding (HAIL) [BJO 09]	(n, k) Network Coding (RDC-NC)
Total server storage	n|F|	n|F|/k	2n|F|/(k+1)
Communication (repair phase)	|F|	|F|	2|F|/(k+1)
Network overhead factor (repair phase)	1	k	1
Server computation (repair phase)	O(1)	O(1)	O(1)

RDC-NC is built on top of network coding-based distributed storage systems

- |F| = size of the file F, which is outsourced at n servers
- Any k out of n servers have enough information to recover F (for erasure coding and network coding)
- Network overhead factor: the ratio between the amount of data that needs to be retrieved to the amount of data that is created to be stored on a new server

Adversarial Model

- Mobile adversary that can behave arbitrarily (Byzantine behavior).
- The adversary can corrupt at most n-k out of the n servers within any given time interval (an epoch).
- An epoch consists of two phases
  - Challenge phase
    - Corruption sub-phase (adversary can corrupt up to b1 servers)
    - Challenge sub-phase
  - Repair phase
    - Corruption sub-phase (adversary can corrupt up to b2 servers)
    - Repair sub-phase
- $b1+b2<=n-k$

Contributions

- Design a secure Remote Data Integrity Checking scheme for Network Coding-based distributed storage systems (Our focus in this presentation)
  - Optimize combined costs of challenge and repair phases
  - Preserve in an adversarial setting the repair bandwidth advantage of network coding over erasure coding
- Guidelines on how to apply network coding in a distributed-storage system based on untrusted server
- Experimental evaluation for our scheme

Challenges

- Localize faulty servers

- Lack of fixed file layout (makes it difficult to maintain constant storage on client)
  - Erasure coding has fixed file layout (a new, repaired block is identical to the original block)
- Additional attacks. Replay attack, pollution attack, …
  - The newly generated blocks in repair are not necessarily equal to the original blocks (replay attack)
  - The untrusted servers are responsible for generating the blocks in repair phase (pollution attack)

Maintaining Constant Client Storage

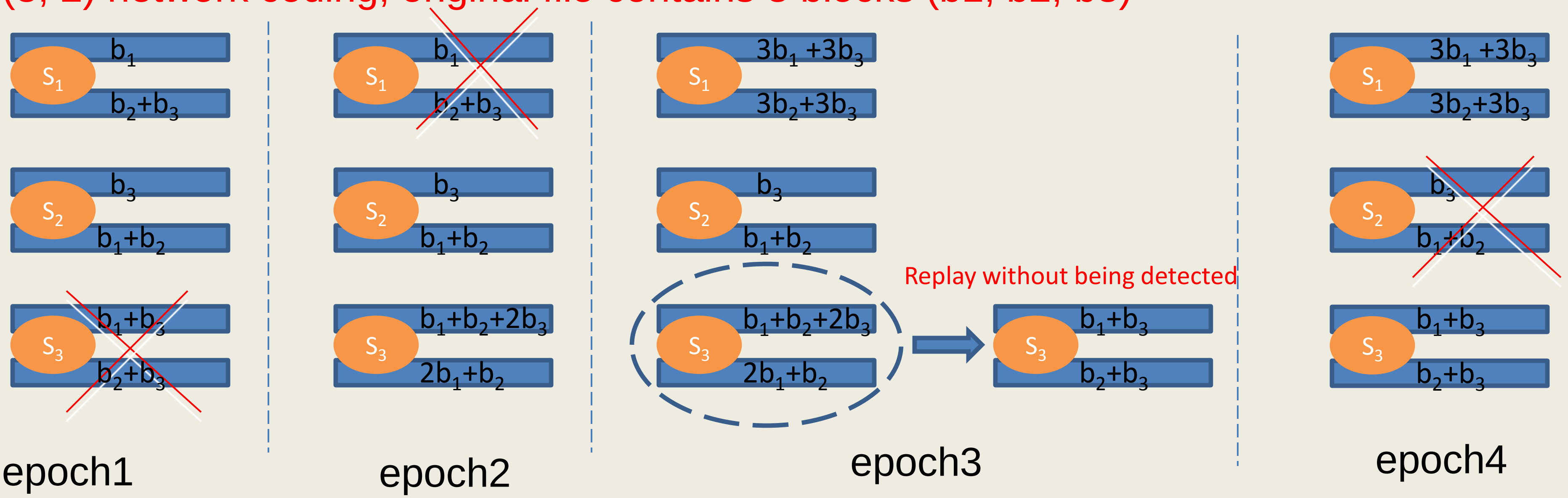
- Can single server solutions (PDP [ABCHKPS 07], PoR [JK 07, SW 08]) be adapted? No!
  - collusion of servers (server can reuse each other's data and meta-data to answer the challenge)
- Use metadata for integrity checks (allows to easily localize faulty servers)
- Meta-data is customized per server per block: assign a logical ID to coded blocks (server_index||block_index) and embed IDs and coding coefficients into meta-data
  - Tackle the problem of collusion of servers
  - Provide integrity for every block in every server

Replay Attack

- By replaying intentionally, the adversary can corrupt the whole system
  - Replay attack is specific for random network coding-based distributed storage systems (reduce the linear independency of blocks, eventually corrupt the whole system)
  - Difficult to detect and maintain constant client storage

The original data is unrecoverable

(3, 2) network coding, original file contains 3 blocks (b1, b2, b3)



Replay Attack (cont.)

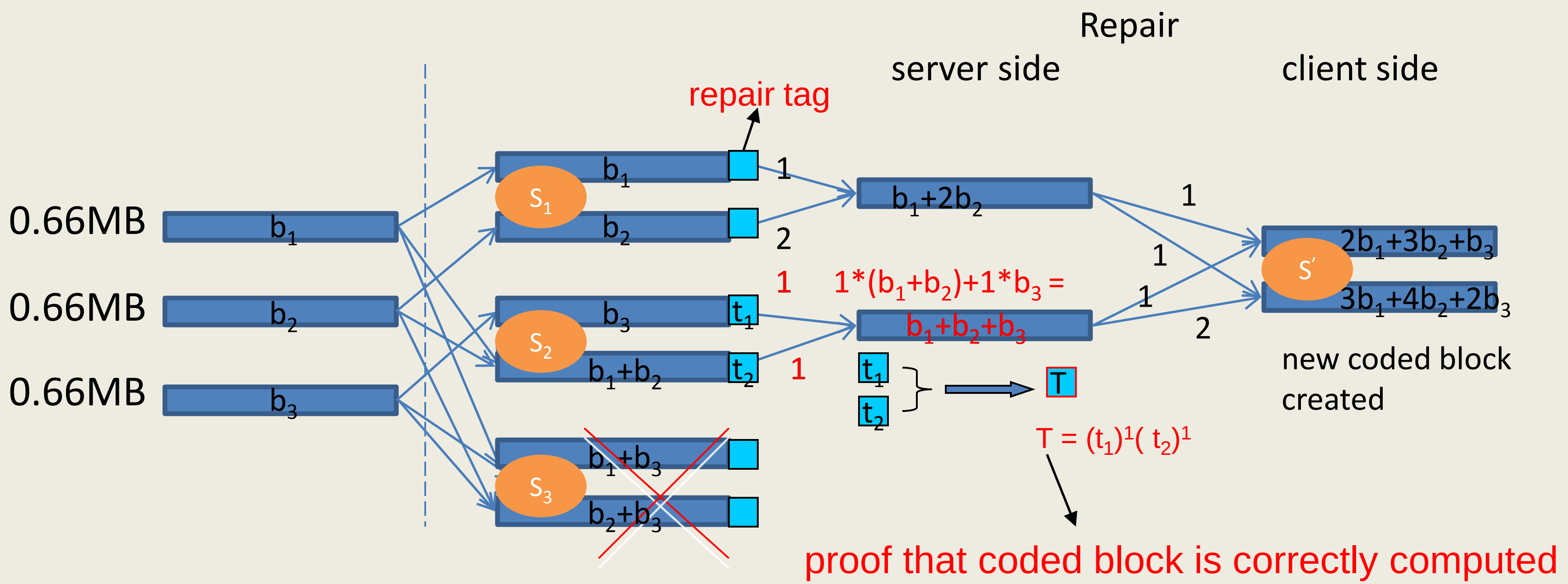
- Our solution for replay attack
  - We encrypt the coding coefficients (under the assumption that the original file should not be public)
  - We prove that by encrypting the coefficients, a malicious server's ability to execute a harmful replay attack becomes negligible
    - The server cannot do better than randomly select blocks for replay attack
    - Please refer to the paper for the detailed proof.

Inconsistency between Challenge Phase and Repair Phase

- Malicious servers can pretend to be good in challenge phase, but behave maliciously in repair phase.
  - Corrupt data (pollution attack)
  - Do not use the random coefficients to generate the new block (entropy attack)

Inconsistency between Challenge Phase and Repair Phase (cont.)

- Our solution
  - Repair tag which supports aggregation
  - Client picks the random coefficients and enforces servers to use
  - Client checks if servers use correctly coded blocks
  - Client checks if servers use coding coefficients provided by client



RDC-NC Overview

- Setup phase
  - Encode the original m-block file into nα blocks by random network coding (coefficients are generated randomly).
  - Generate challenge tags and repair tag for every block
    - Every block is a collection of segments, every segment has one challenge tag (PDP or PoR tag), used in challenge phase
    - One repair tag per block (to prevent attacks in repair phase)
  - Encrypt the coefficients (replay attack)
  - Outsource the encoded blocks (together with encrypted coding coefficients) and metadata (challenge and verification tags)
    - α blocks at each of the n servers

Scheme Overview (cont.)

- Challenge phase
  - Check every block in every server based on challenge tags
    - Optimize the communication cost by aggregating the responses of α blocks (PDP or PoR tags supports aggregation)
- Repair phase
  - Repair phase is activated after having found corrupted servers in challenge phase
  - Client will communicate with some healthy servers
    - Client send random coefficients to servers
    - Servers use the random coefficients to compute new coded blocks
    - Servers also use the random coefficients to compute a proof that the new coded blocks are correctly computed
    - Severs send back the coded blocks and the proofs
  - Client checks the proofs, and uses the correctly generated blocks to repair the corrupted servers

Conclusion

- Network coding (regenerating code) is a promising coding method for distributed storage systems (reduced repair bandwidth)
- Our RDC-NC scheme is designed for a strong adversarial model (mobile and Byzantine)
- RDC-NC is secure by tackling various attacks (data corruption, collusion of servers, replay attack, pollution attack, …)

Thank you!

Questions?