Applications of Secure Coding in Distributed Storage and Wireless Networking

Reza Curtmola

New Jersey Institute of Technology

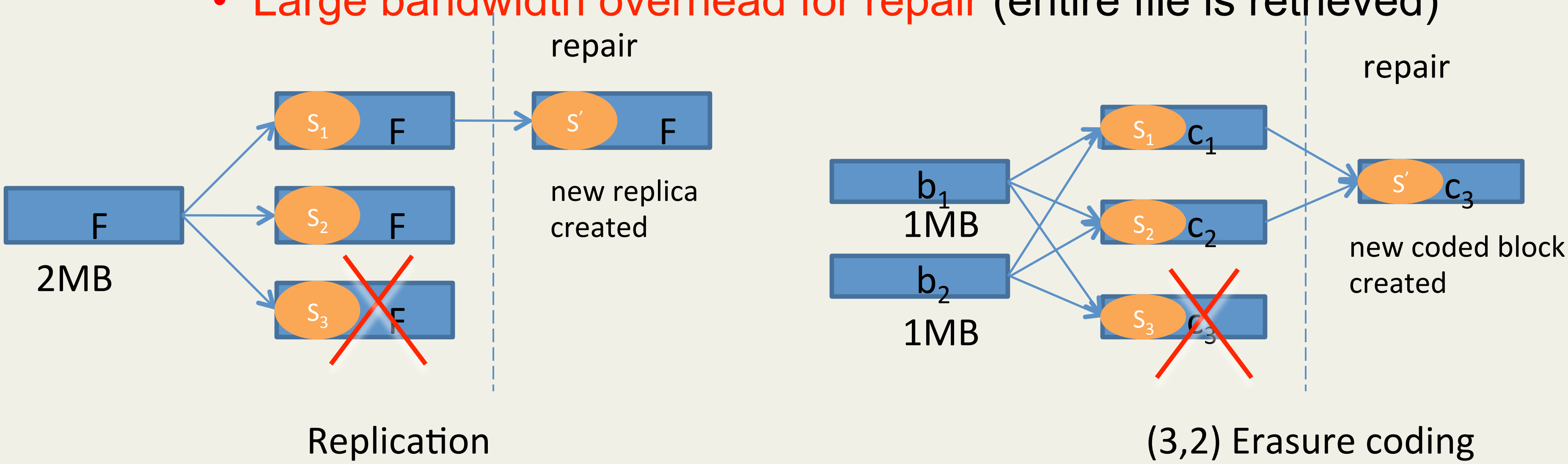
Parts of this presentation are based on joint work with Bo Chen, Randal Burns, Giuseppe Ateniese, Andrew Newell, and Cristina Nita-Rotaru

Motivation

- Remote storage is ubiquitous:
  - Web-based email (Yahoo Mail, GMail)
  - Online data backup/recovery/archival:
    - Enterprise (iron mountain, evault)
    - Consumer (mozy, carbonite, dropbox, google drive)
  - Cloud Storage (Amazon, Microsoft, Google, IBM, etc.)
- Cloud storage can release people from the burden of hardware management
- Reduce the cost (Storage AS Service, pay as you use)
- More reliable (S3 99.999999999% durability, with 99.99% availability)

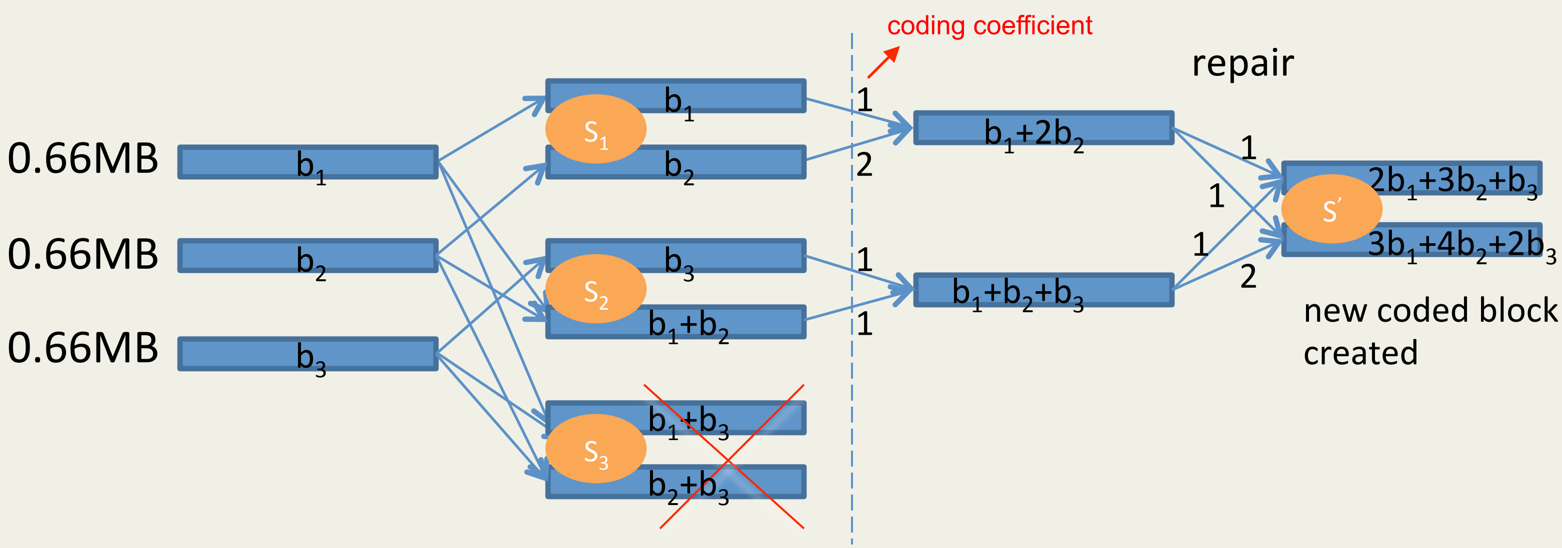
Reliability in Distributed Storage Systems

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



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 overhead of network coding:
  - Regulatory storage
  - Data escrow
  - Deep archival stores
  - Preservation systems for old datasets

Online Backup/Archival Systems

- Users can check data authenticity upon retrieval
  - Insufficient to verify data on read
- An important feature is missing:

  the ability to prove data possession

  (a way to periodically check that the server still has the data)
- The risk of outsourcing storage cannot be assessed
  - Data owners lose control over the faith of their data
  - Cloud storage providers must be trusted unconditionally
  - Numerous reports of data loss incidents
  - This makes cloud storage unsuitable for applications that require strong security and long-term reliability guarantees

Archival Storage

- Storage servers:
  - Retain tremendous amounts of data
  - Only small parts of the data are retrieved
  - Hold data for long periods of time (forever)

- Unique performance demands:
  - Accessing the entire data is expensive in I/O costs for the server
  - Sending all the data across a network is expensive

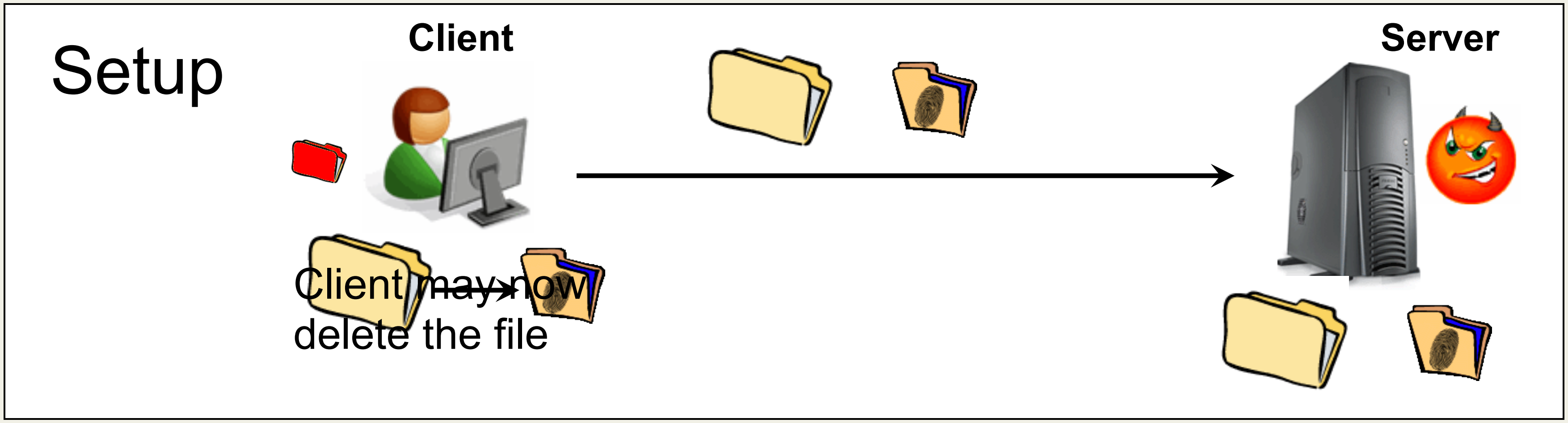
Remote Data Integrity Checking

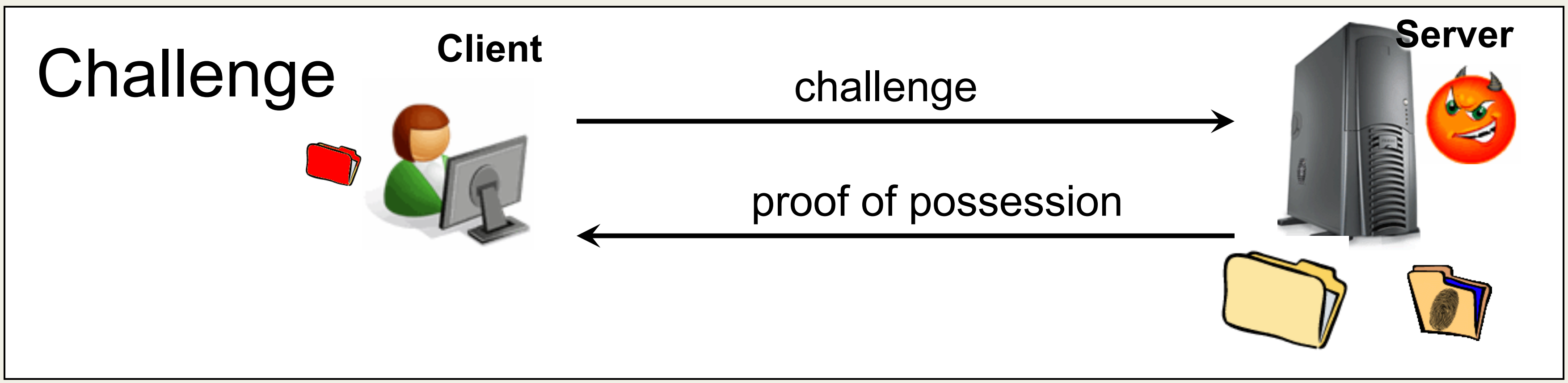
- Remote Data Checking (RDIC) is a mechanism used by the data owner to check the integrity of data stored at an untrusted server
  - without having the server access all the data
  - without retrieving the data from the server

Why Not Trust Service Providers?

- Financial motivations to cheat
  - Charge for terabytes and store gigabytes
  - Discard un-accessed data (based on statistical analysis)
  - Keep fewer replicas than promised
- Reputation
  - Hide data loss incidents
- Latent errors
  - Of which service providers are unaware

Remote Data Integrity Checking (RDIC)





Security requirement: Detect server misbehavior when the file (or parts of it) cannot be retrieved

Single-server Remote Data Integrity Checking

- Tag-based:
  - Provable Data Possession (PDP)

    [Ateniese, Burns, Curtmola, Herring, Khan, Kissner, Peterson, Song, "Remote Data Checking Using Provable Data Possession"]
  - Compact Proofs of Retrievability (CPOR)

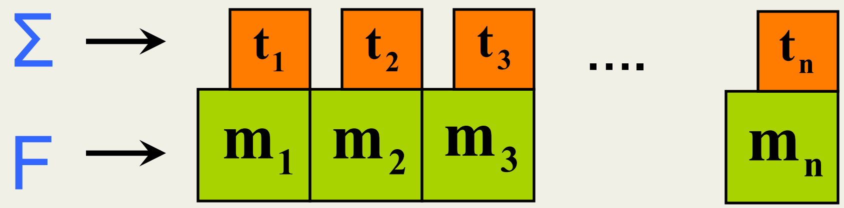
    [Shacham, Waters, "Compact Proofs of Retrievability"]

- Sentinel-based
  - Proofs of Retrievability (POR)

    [Juels, Kaliski, "PORs: Proofs of Retrievability for Large Files"]

Tag-based RDIC

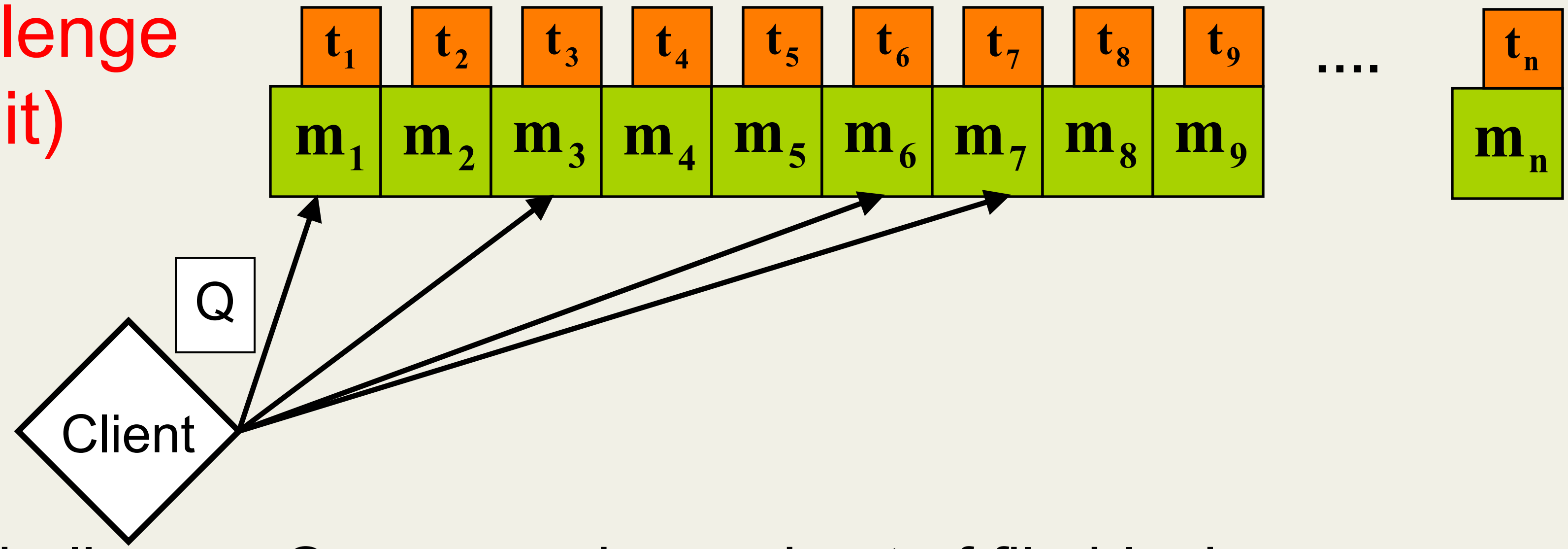
Setup



- C generates tags for each file block
  - Tags have special properties, can be aggregated
- C sends F and Σ to S
- C keeps some cryptographic key material and deletes F, Σ

Tag-based RDIC

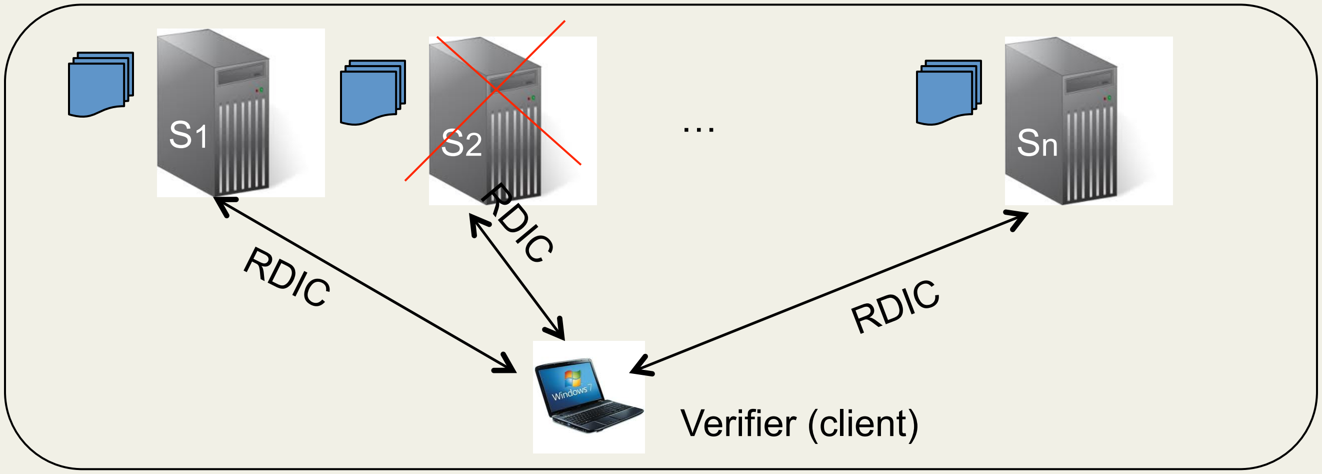
Challenge (Audit)



- C challenges S on a random subset of file blocks
  - query Q is different per challenge
  - checked subset of blocks is different per challenge
- S responds with a proof of possession: V = (T, M)
  - T is a function of tags t_1, t_3, t_6, t_7 (aggregation)
  - M is a function of the challenged blocks m_1, m_3, m_6, m_7
- C checks if a certain relationship holds between T and M

Beyond Single-server RDIC

- Single-server RDIC is only one facet of maintaining the health of data (prevention)
- We really want to ensure long-term data reliability
  - Remote data checking for distributed storage systems



- Phases: Store, Audit, Repair
- Additional challenges: server collusion, keep costs sub-linear in n, new attacks

Summary of Remote Data Integrity Checking

- 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)

RDIC for Distributed Storage Systems

- Data is stored redundantly at multiple servers
  - Replication
    - Simplicity, requires more storage
    - [Curtmola, Khan, Burns, Ateniese, “MR-PDP: Multiple-Replica Provable Data Possession”]
  - Erasure coding
    - Optimal storage to achieve desired reliability level, expensive repair phase
    - [Bowers, Juels, Oprea, “HAIL: A High-Availability and Integrity Layer for Cloud Storage”]
  - Network Coding
    - Minimal communication overhead to repair damaged data
    - [Chen, Curtmola, Ateniese, Burns, “Remote data checking for network coding-based distributed storage systems”]

Performance Comparison

	Replication (MR-PDP)	(n, k) Erasure Coding (HAIL)	(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
- $b1+b2 \leq n-k$

Our Focus

- Design a secure Remote Data Integrity Checking scheme for Network Coding-based distributed storage systems
  - Optimize combined costs of challenge and repair phases
  - Preserve in an adversarial setting the repair bandwidth advantage of network coding over erasure coding

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)

Maintain Low Storage Cost (client)

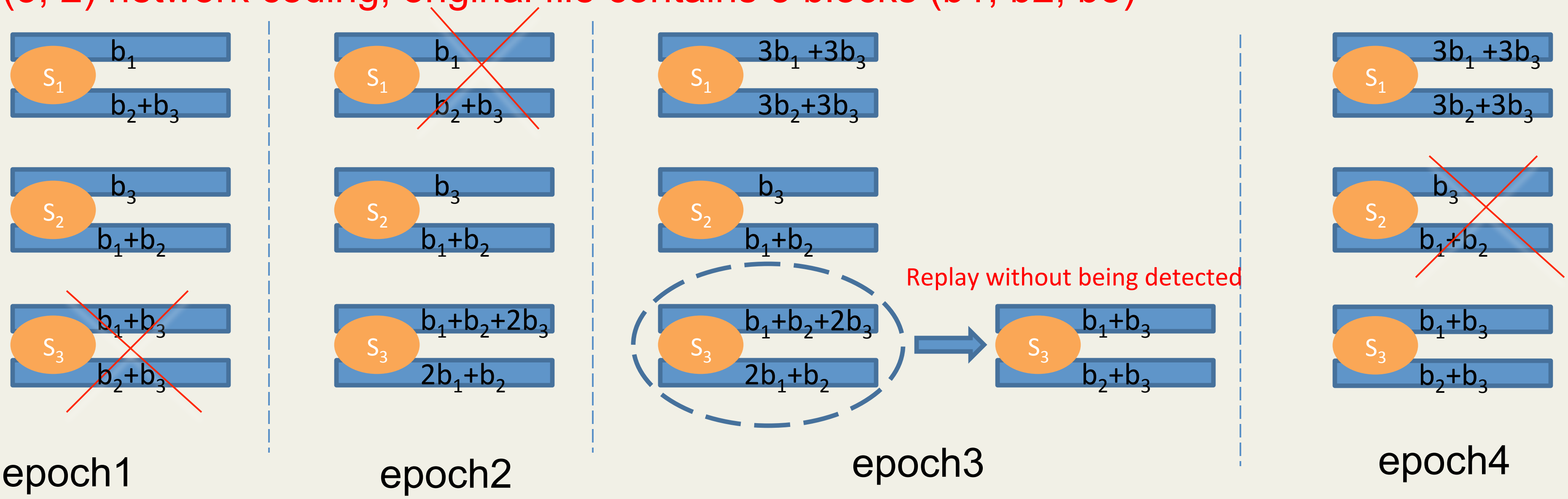
- Can single server solutions (PDP, PoR) be adapted? No!
  - collusion of servers (server can reuse each other's data and meta-data to answer the challenge)
- Use metadata for integrity checks (allows to easily localize faulty servers)
- Meta-data is customized per server per block: assign a logical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

Replay Attack

- By replaying intentionally, the adversary can corrupt the whole system
  - Replay attack is specific to random network coding-based distributed storage systems (reduce the linear independency of blocks, eventually corrupting 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.)

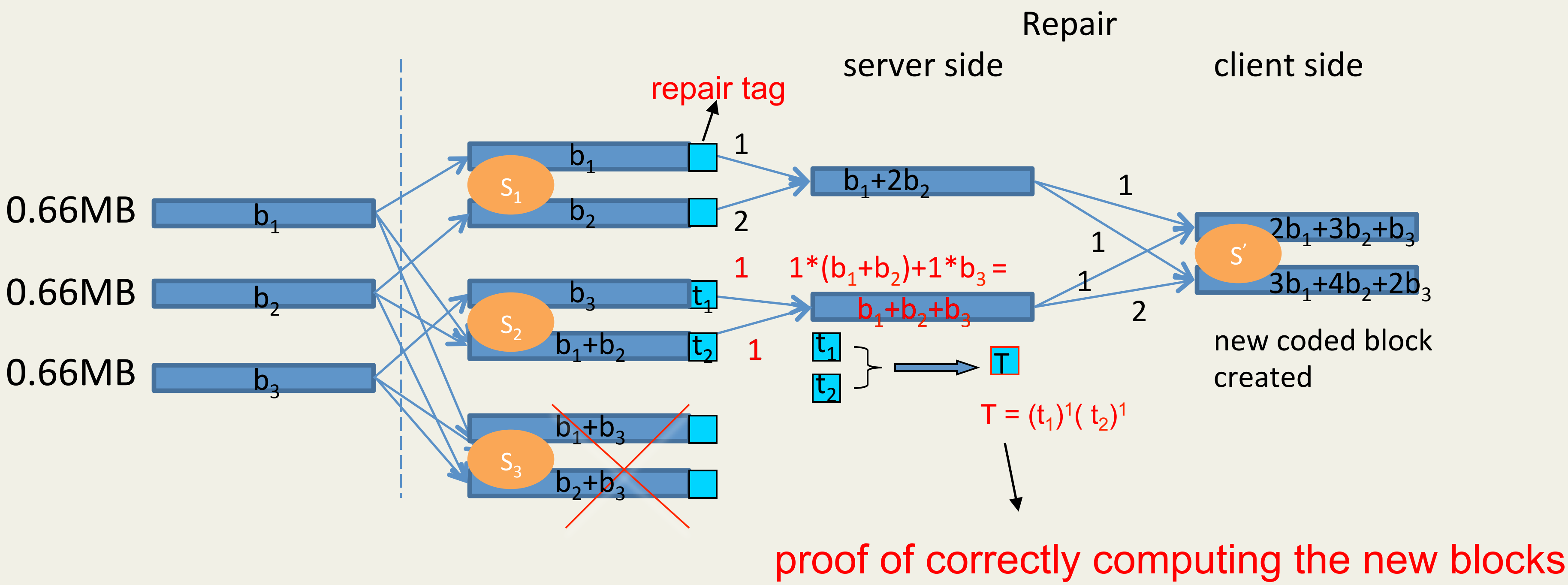
- Our solution
  - We encrypt the coding coefficients
  - We prove that by encrypting the coefficients, a malicious server's ability to execute a harmful replay attack becomes negligible (the server cannot do better than randomly select blocks for replay attack)

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 based on random network coding
  - 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 the challenge tags
    - Reduce the communication cost by aggregating the responses of α blocks (PDP or PoR tags support aggregation)

Scheme Overview (cont.)

- Repair phase
  - Repair phase is activated after finding corrupted servers in the Challenge phase
  - Client works with a set of healthy servers
    - Client sends 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, then uses the correctly generated blocks to repair the corrupted servers

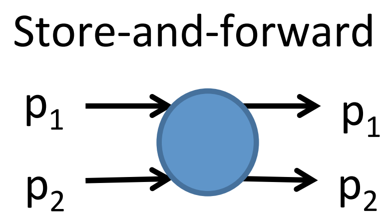
Conclusion

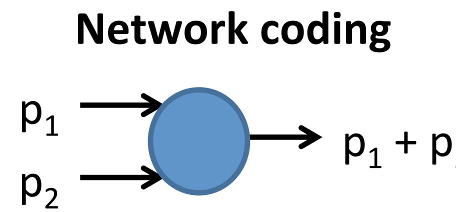
- Network coding is a promising coding method for distributed storage systems (minimize repair bandwidth)
- Our RDC-NC scheme is designed to withstand a strong adversarial model (mobile and Byzantine)
- RDC-NC ensures data integrity and long-term reliability by mitigating various attacks (data corruption, collusion of servers, replay attack, pollution attack, …)

Entropy Attacks and Countermeasures in Wireless Network Coding

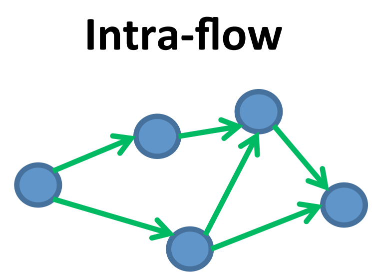
Network Coding

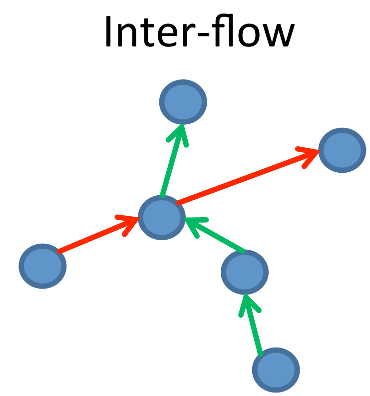
- Network coding fundamentally changes routing





- Different strategies for coding within a flow or among multiple flows





Application
Transport
Network
MAC
Physical

- Coding at various levels of networking stack

Wireless Network Coding Routing

- Random linear network coding
  - Random coding offers completely decentralized coding and shown to be sufficient [Ho et al., 03]
  - Linear coding has been shown to be optimal [Li et al., 03]
- Opportunistic routing
  - Forwarders can leverage any packet reception
  - Natural multipath routing with little coordination
  - Throughput and reliability improvements

Security in Network Coding

- Malicious store-and-forward routers
  - ➢Routers should not modify packets
  - ➢Any modification can be labeled malicious
- Malicious network coding routers
  - ➢Routers are supposed to modify packets
  - ➢Much harder to tell whether a modification was malicious

This Talk

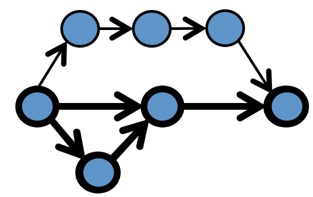
- Overview of network coding
  - Show benign operation
  - Demonstrate attacks
- Entropy attacks
  - Local and global versions
  - Demonstrate impact on network coding system
- Present detection techniques
  - Show how to defend against local entropy attacks
  - Demonstrate limitations of defending against global entropy attacks
  - Describe techniques to mitigate global entropy attacks

Network Coding Background

- Packets are vectors of elements of a small finite field (256)
- Packets (32) grouped into *generations*
- Source selects forwarders to send data
- Nodes periodically broadcasts random linear combination their *coding buffer*
- Destination can decode upon obtaining full coding buffer

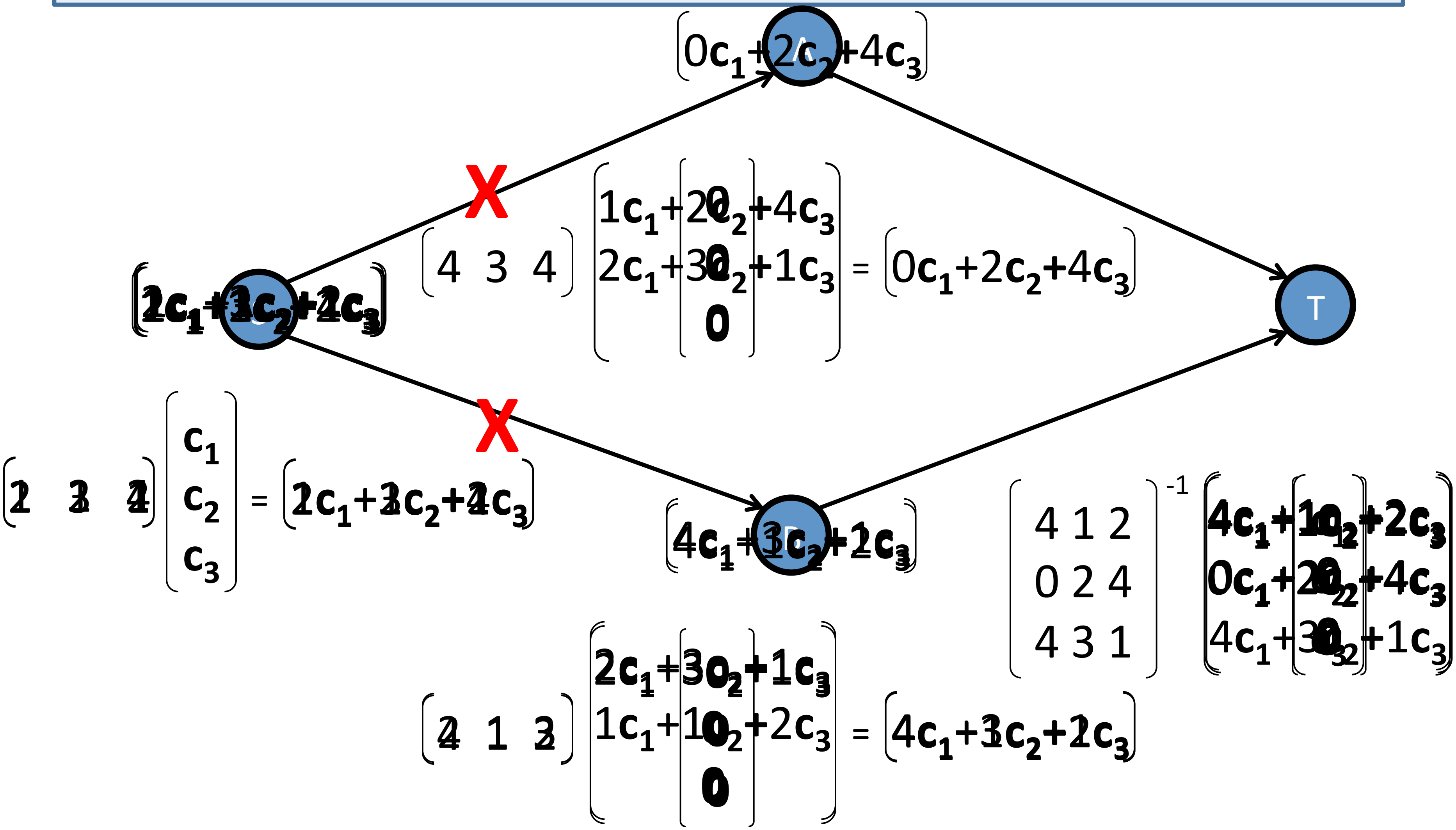
0xA318B309… → < 163, 24, 179, 9, … >

$$\begin{pmatrix} < 163, 24, 179, 9, \ldots > \\ < 3, 124, 19, 3, \ldots > \\ \vdots \\ < 123, 11, 34, 12, \ldots > \end{pmatrix}$$



$$\begin{pmatrix} r_1, r_2, \ldots\ r_{32} \end{pmatrix} \begin{pmatrix} < 163, 24, 179, 9, \ldots > \\ < 3, 124, 19, 3, \ldots > \\ \vdots \\ < 123, 11, 34, 12, \ldots > \end{pmatrix}$$

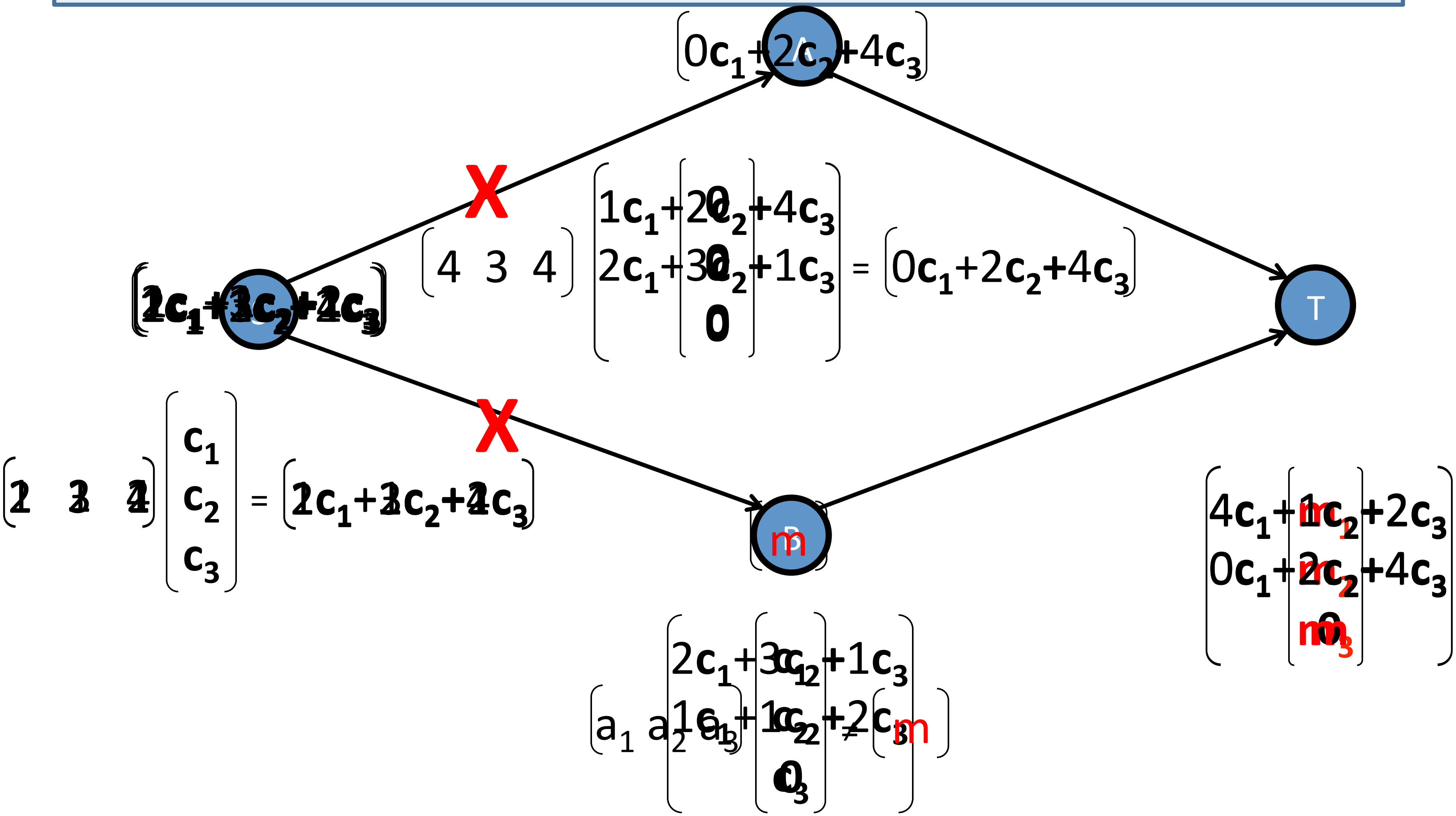
Network Coding Example



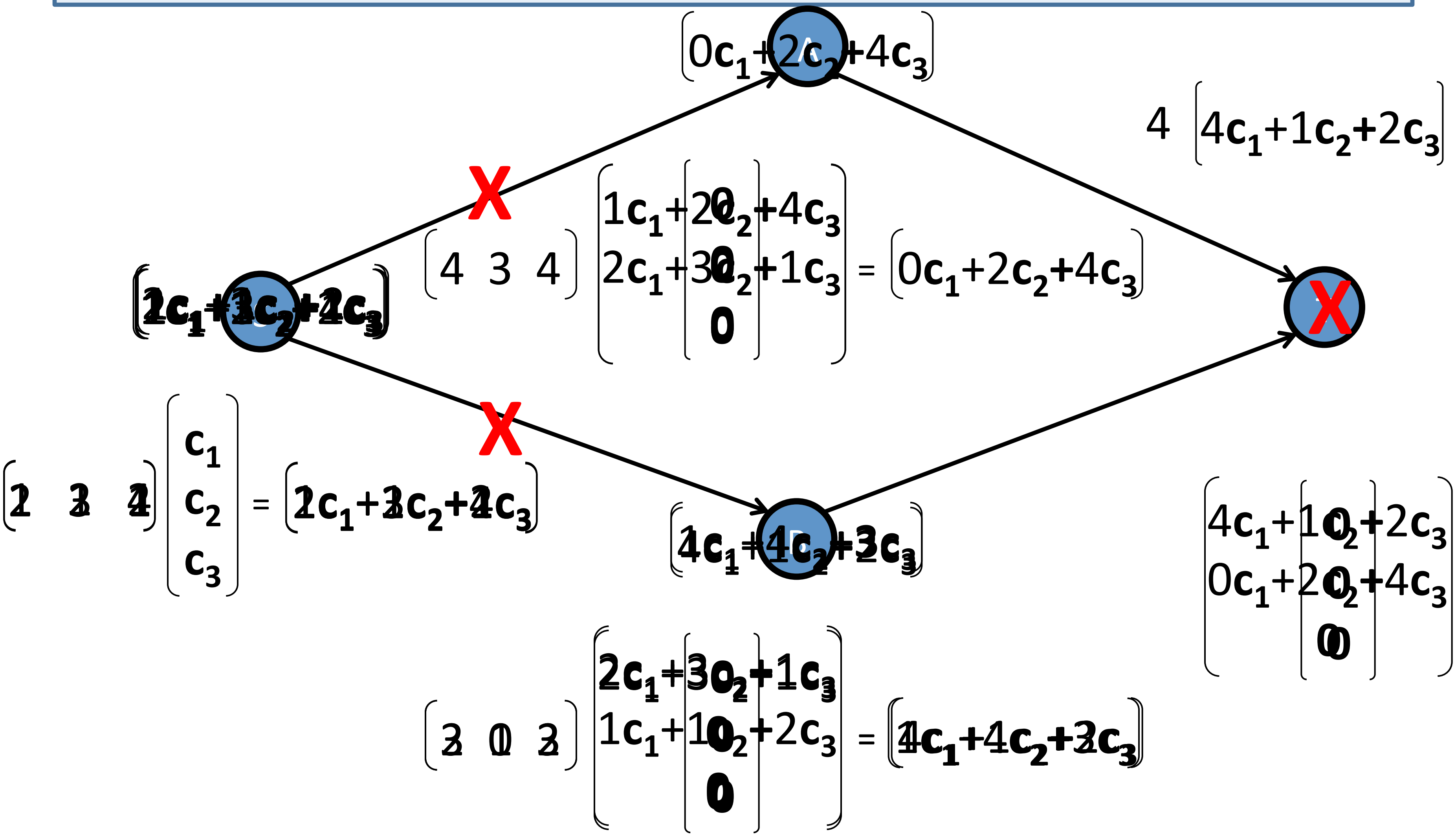
Network Coding Attacks

- Pollution: creating bogus packets
  - ➢ Results in invalid decoding at destination
  - ➢ Well-studied, many defenses
- Entropy: creating non-innovative packets
  - ➢ Waste network resources and blocks data flow
  - ➢ Less-studied
    - Locally share coefficients in P2P [Gkantsidis et al., 06]
    - Quickly perform independence check [Jiang et al., 09]

Pollution Attack Example



Entropy Attack Example

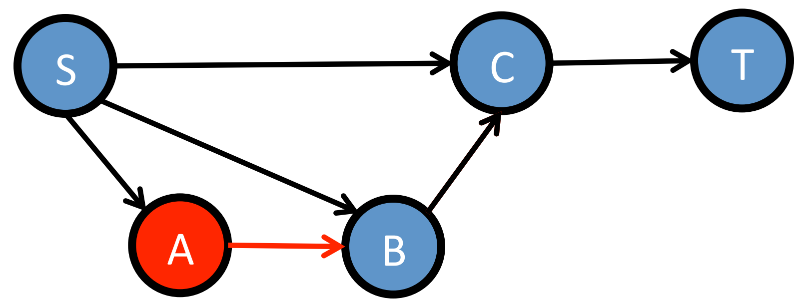


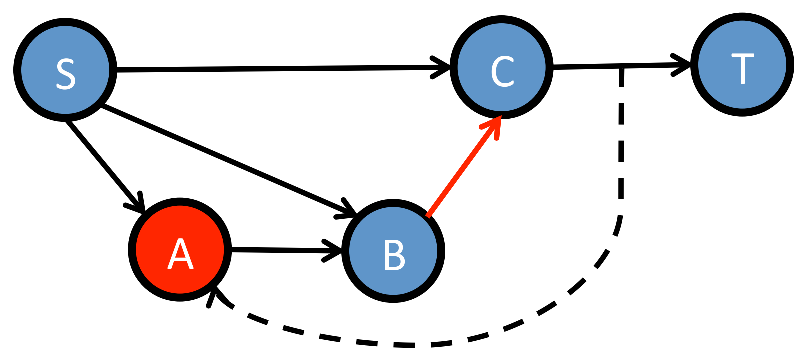
Pollution vs Entropy

- Pollution attacks
  - ➢ Huge impact with little effort
  - ➢ Blatant deviation from normal coding
- Entropy attacks
  - ➢ Less impact
  - ➢ Subtle deviation from normal coding
- Orthogonal defenses
  - ➢ Pollution defense detects whether packet **is not** a linear combination of source packets
  - ➢ Entropy attackers create packets that **are** linear combinations of source packets

Local vs Global Entropy Attacks

- Local
  - ➢ Easy to perform
  - ➢ Easy to detect
- Global
  - ➢ Requires out-of-band channel to perform
  - ➢ Difficult to detect



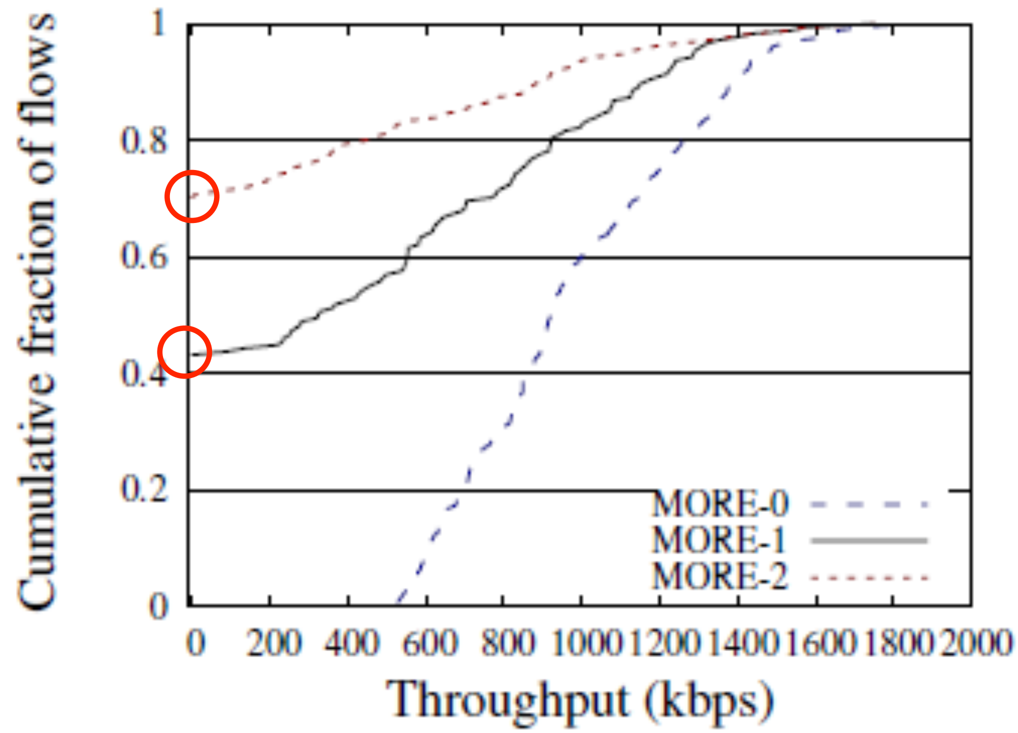


Experimental Setup

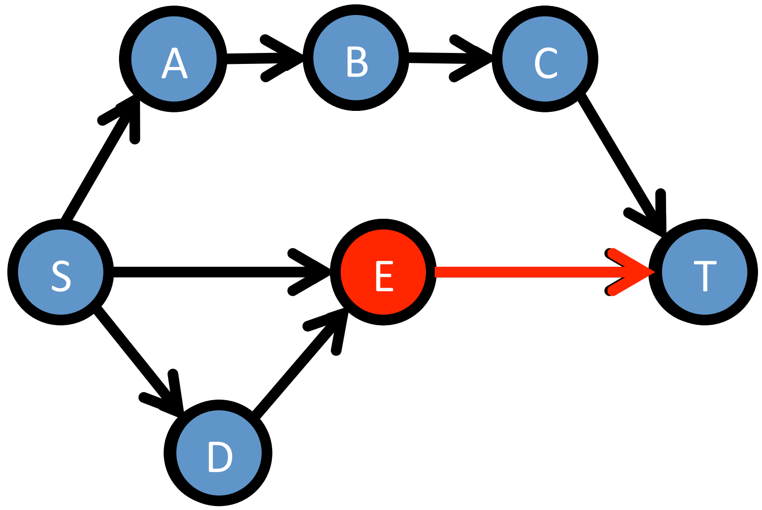
- Topology from measurements of 38-node wireless mesh network (Roofnet)
- Simulate (GlomoSim) MORE network coding protocol
- 200 simulations
  - ➢ Select a random source/destination pair
  - ➢ Simulate data transfer for 400 seconds
  - ➢ Measure throughput
- Select random entropy attackers as forwarders
  - ➢ 32 packets per generation
  - ➢ Perform coding normally on first 16 packets received
  - ➢ Apply zero coefficients to remaining 16 packets

Impact of Local Entropy Attacks

- MORE-(# attackers)
- Zero throughput cases
- Routing logic selects paths assuming all paths will deliver data



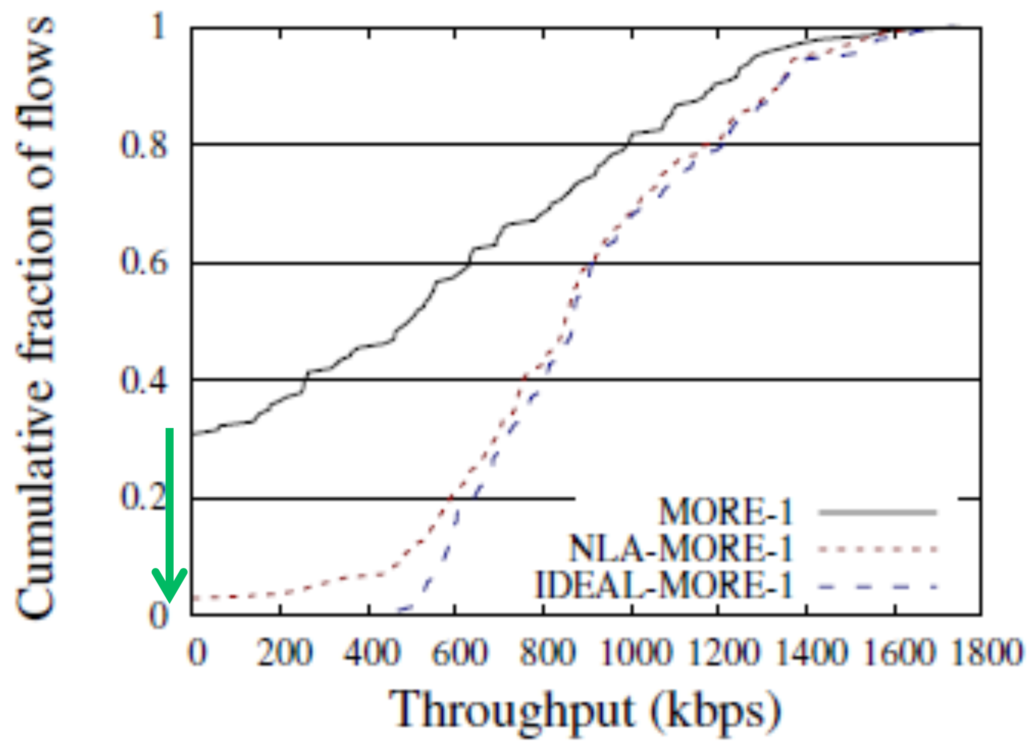
Non-innovative Link Adjustment (NLA)



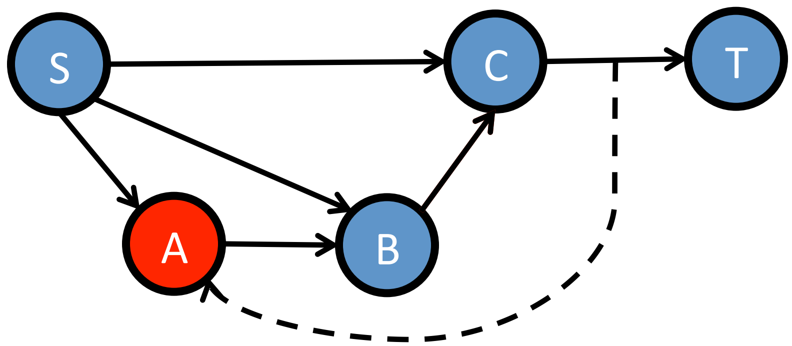
- Standard MORE: source chooses subset of nodes to forward based on link quality
- Source assumes high link quality is a high delivery rate of innovative packets
- Non-innovative Link Adjustment: each node adjusts its links based on proportion of received innovative packets

NLA Performance

- IDEAL, defense that automatically removes attacker
- Performs close to the ideal case
- With adjusted links, data routed around attacker
- Link adjustments stabilize after few rounds of updates (~3)



NLA is Insufficient against Global Entropy Attack

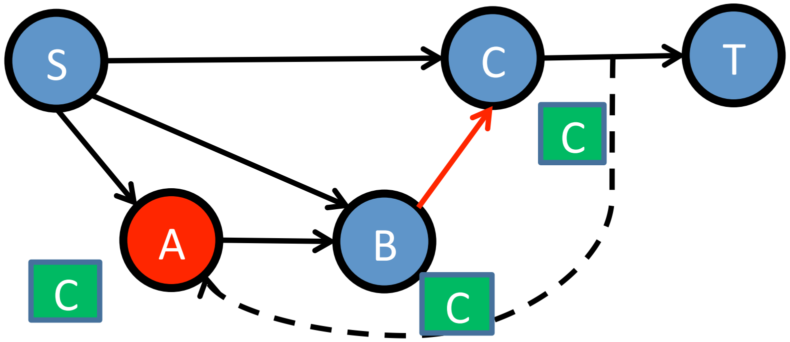


- Attacker encodes with packets from downstream link
- With no defense, 233 kbps
- With NLA defense, 214 kbps
- IDEAL, 345 kbps

Detection for Global Entropy Attacks

- Upstream Buffer Propagation
  - ➢Share information so global entropy attacker neighbors know a packet is globally non-innovative
  - ➢Low overhead, reactive detection
- Buffer Monitoring
  - ➢Watch coefficients of all traffic in and out of a suspect nodes
  - ➢High overhead, proactive detection, creates topology constraints

Upstream Buffer Propagation

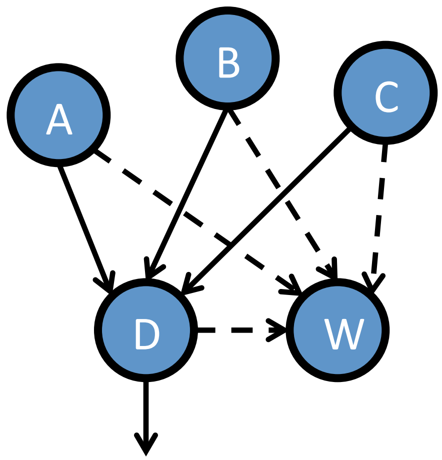


Protocol description

1. Downstream node receives a non-innovative packet
2. Buffer information propagated upstream along path
3. Accusation if innovative packets do not propagate downstream

- Key optimizations
  - Reduce buffer information size
  - Find single path
- Analysis
  - Reactive detection can be exploited
  - Hybrid scheme can be used to add an exoneration period

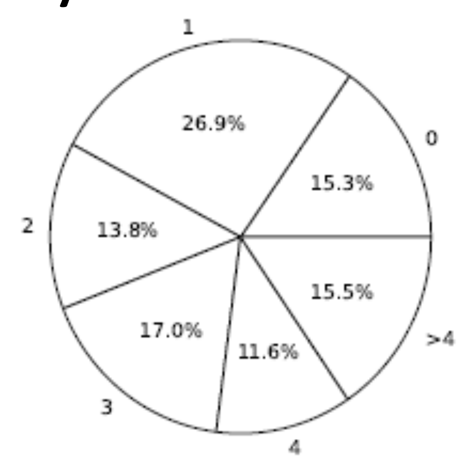
Buffer Monitoring



Protocol description

1. Watchdog buffers coefficients of packets entering suspect node
2. Watchdog ensures all coefficients of packets leaving suspect node are correct

- Key optimizations
  - Publicly known randomness for coefficients
  - Efficient wireless single-hop multicast
- Analysis



Conclusion

- Random linear network coding is inherently vulnerable to entropy attacks
- An entropy attacker can do much more than occupy some network resources, it can block data flow that routing assumes is open
- Detection becomes complicated when packets are locally innovative but globally non-innovative
- Sophisticated defenses necessary to combat this problem

Questions

?