

Deutsche Telekom Chair of Communication Networks
Technische Universität Dresden

Practical Implementations of Network Coding

Frank Fitzek // Summer Semester 2018

Network Coding for Storage

Network Coded Storage

Introduction and Motivation

Cloud storage

- Fast growth in the past years (both enterprise and home users)
- Free storage space: Dropbox, Google Drive, SkyDrive, Box, iCloud, and countless more
- Public APIs, usually REST-based

Issues:

- Reliability issues
- Privacy and security issues
- Limited free storage space

Distributed storage on clouds – in use today on the provider side

Our approach:

- Distribute data to multiple cloud providers on the user side
- Use network coding to create a robust, fast and secure distributed storage solution

Distributed Storage

Data Centre 1G

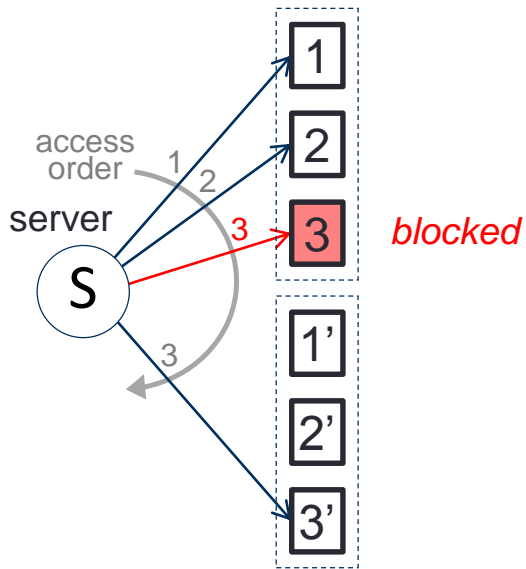


User Cloud Storage 2G/3G



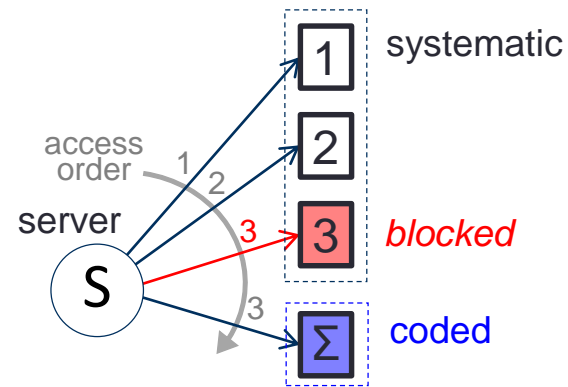
Distributed Storage

Conventional SAN



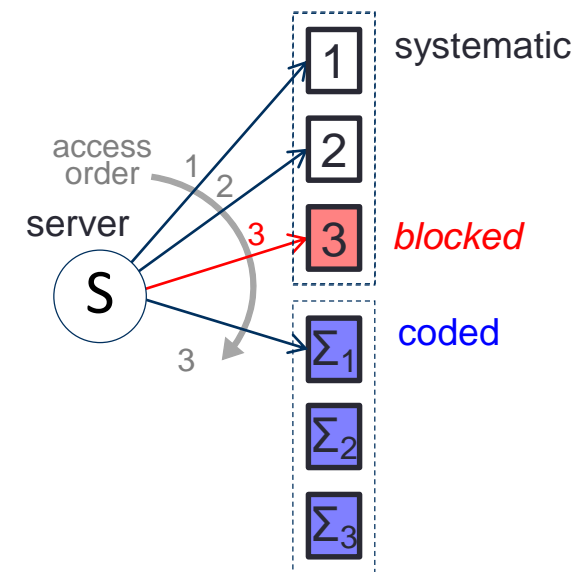
- 2 file copies
- 3 file chunks
- One chunk per drive
- Round-robin server access
- Chunk "3" blocked

RLNC-encoded SAN (two options)

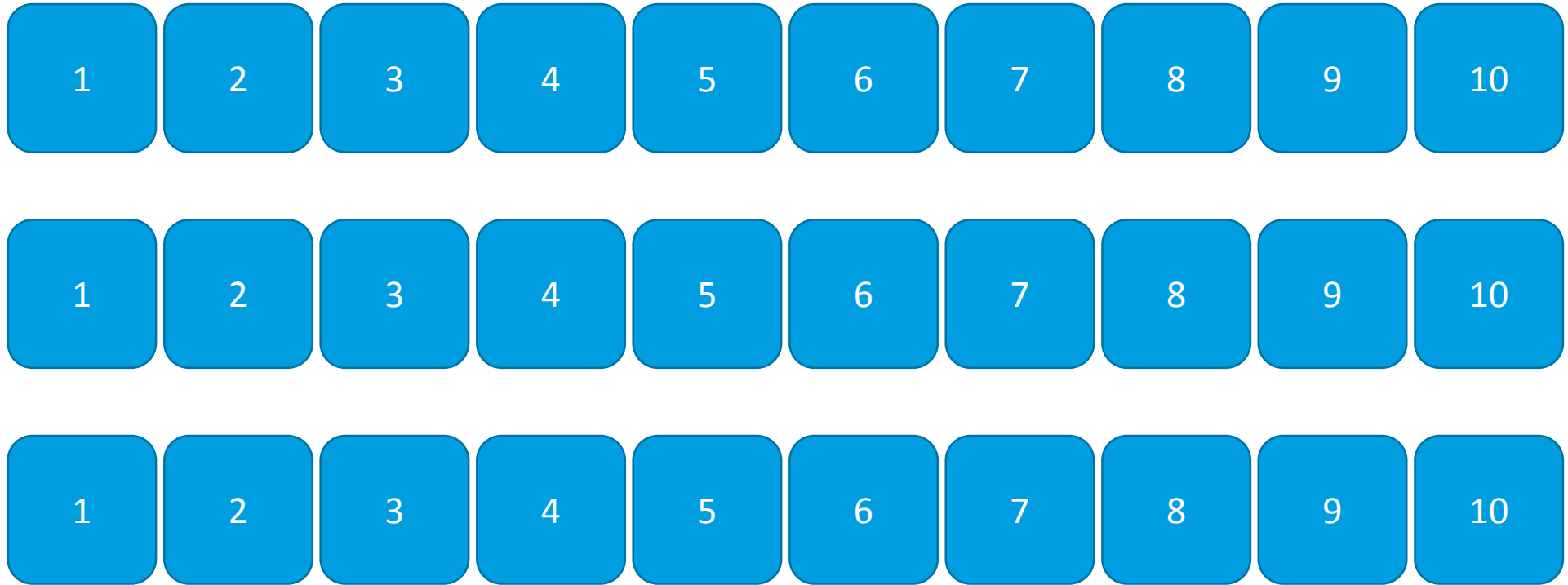


Decoding

$$3 = \Sigma - 1 - 2$$

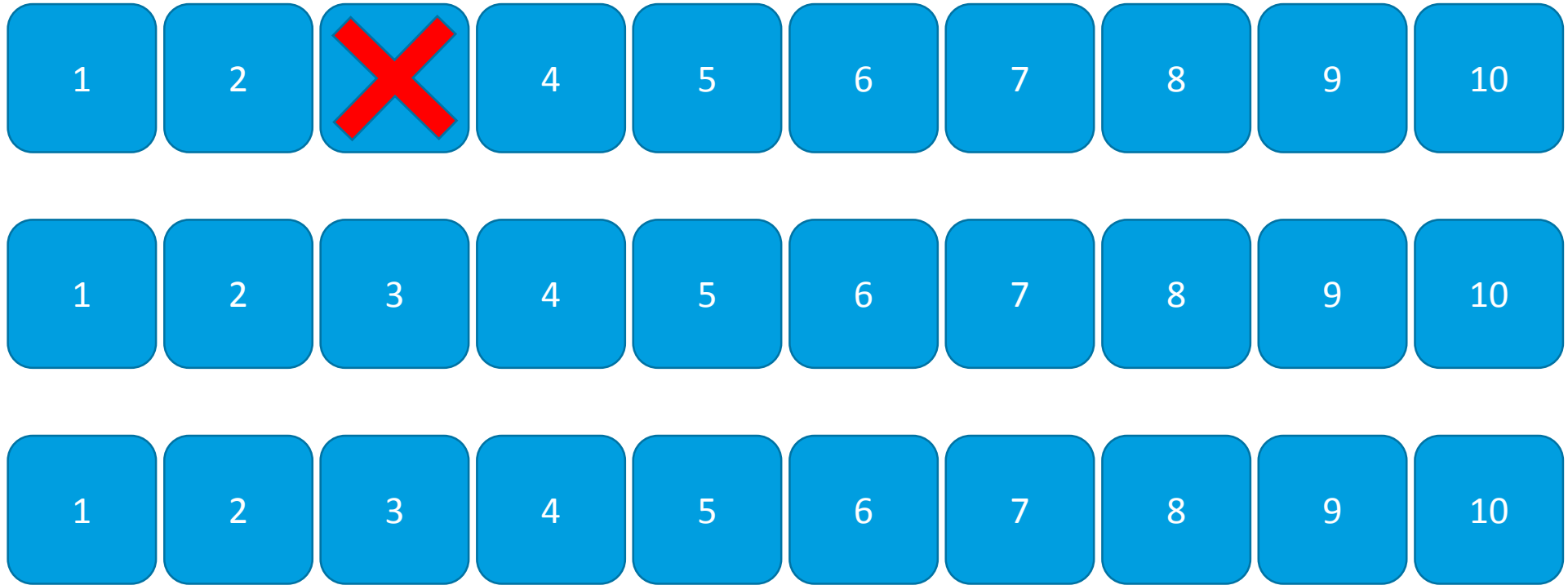


State of the Art



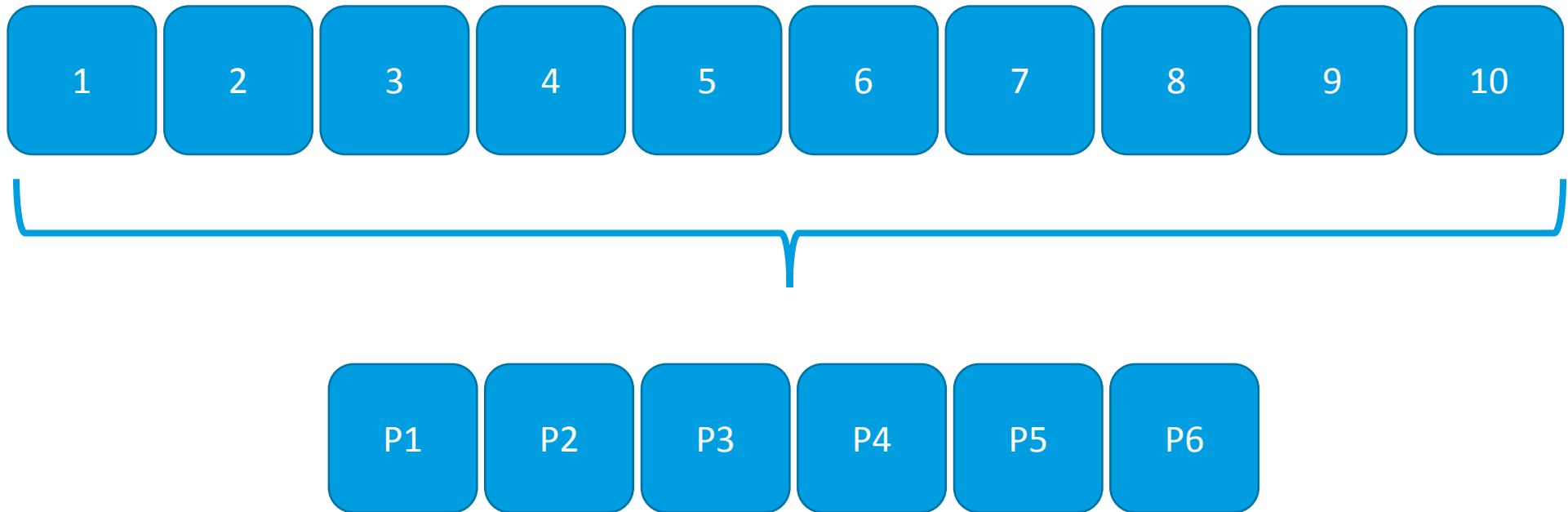
Overhead: 200%

State of the Art



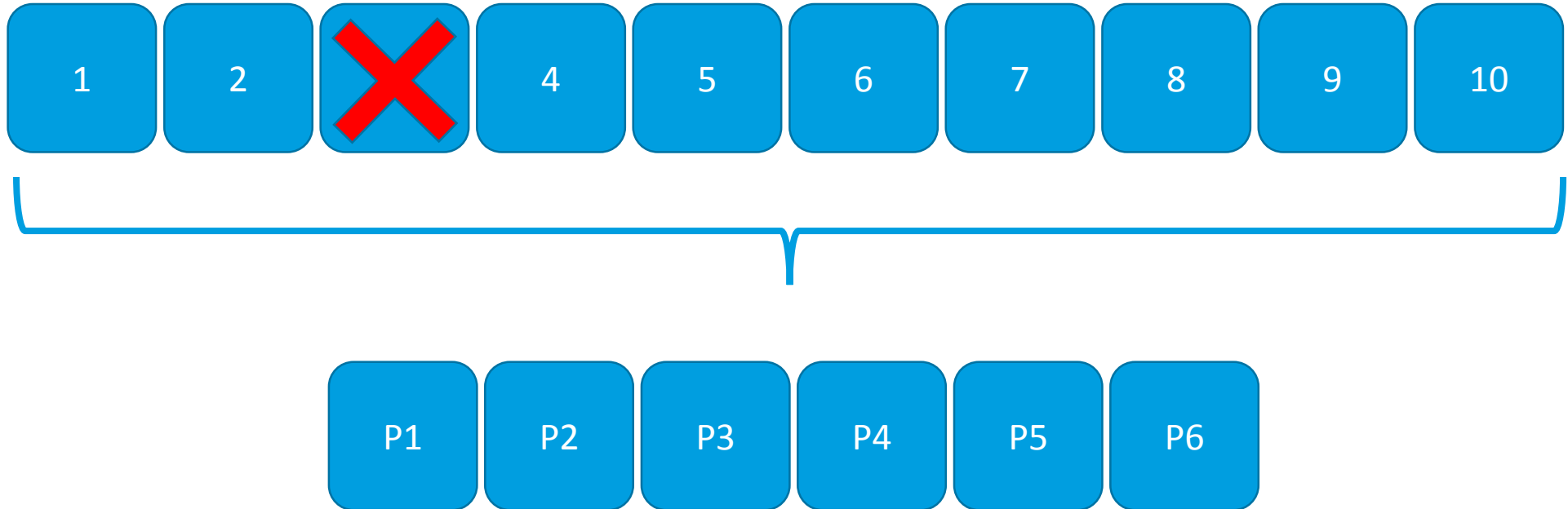
One failure results in one traffic unit

10:6 Code (Facebook)



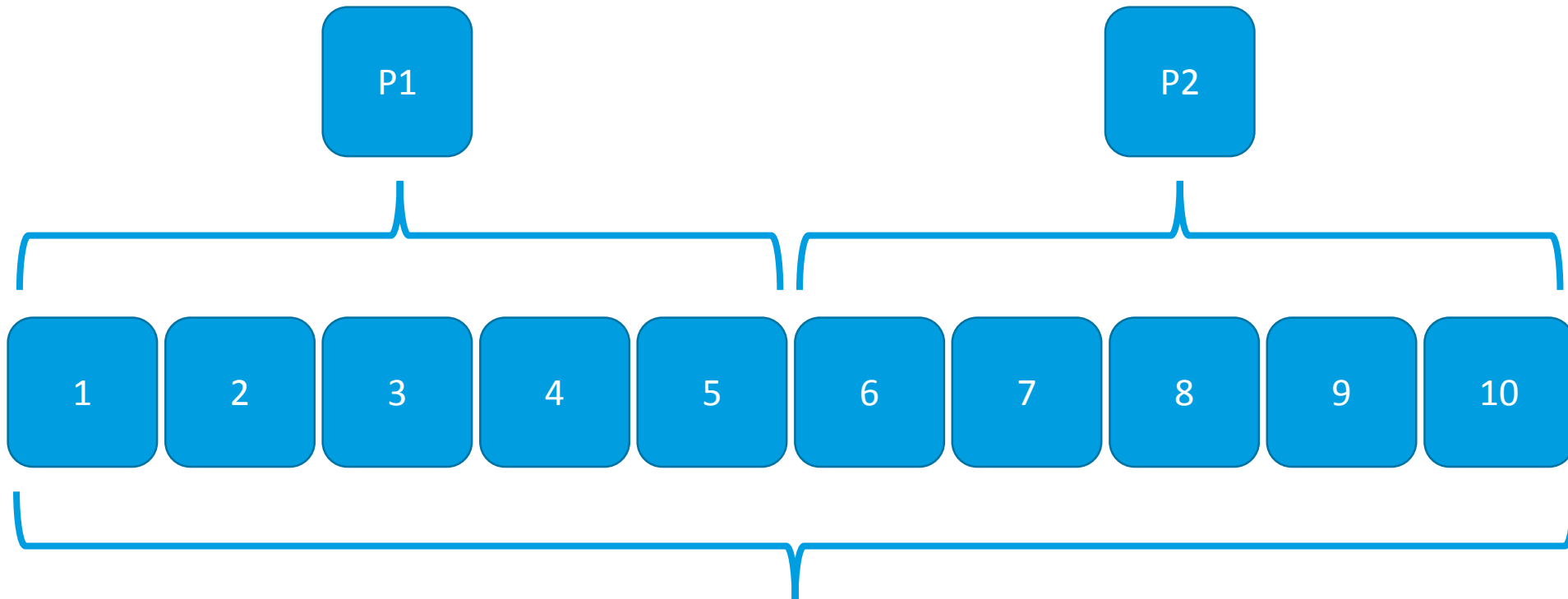
Overhead: 60%

10:6 Code (Facebook)



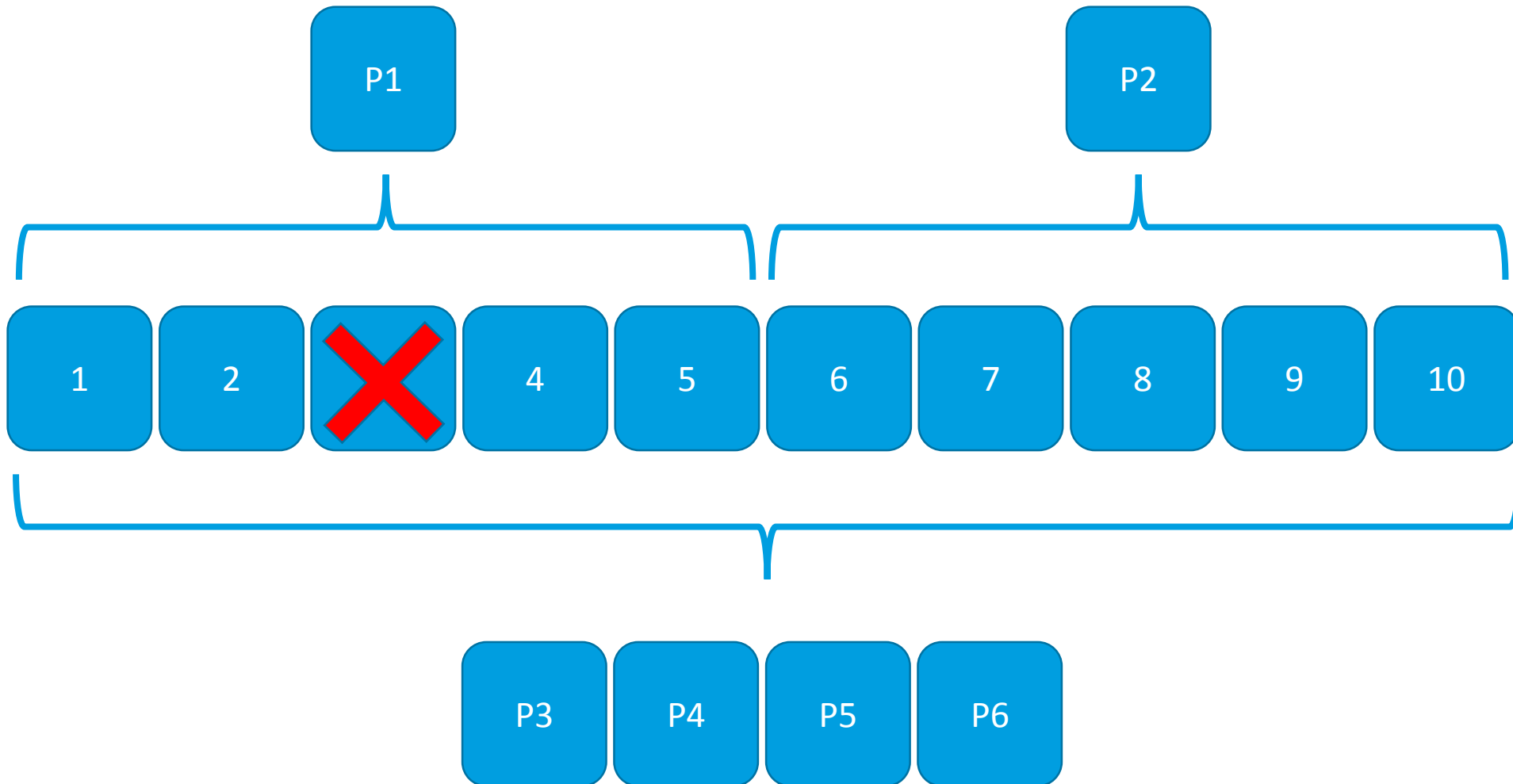
One failure results in 10 traffic units

Xorbas



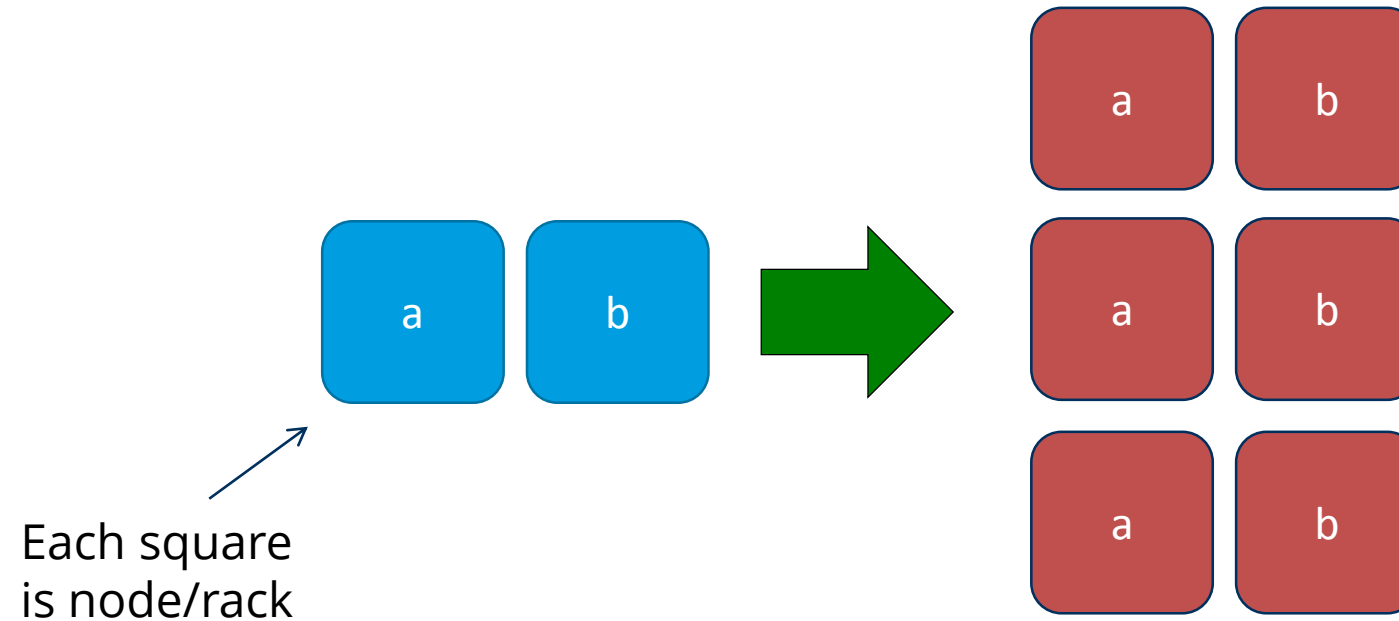
Overhead: 60%

Xorbas



One failure results in 5 traffic units

State of the Art: Replication



Overhead:

200%

State of the Art: Replication

Overhead: 200%

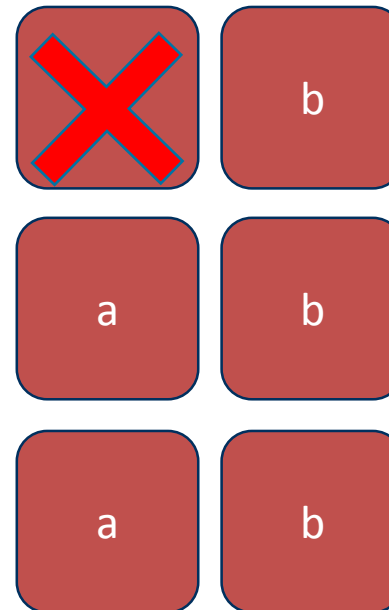
- High storage cost

Traffic to repair loss:

- 1 unit

Protection:

- 4 losses (best case)
- 2 losses (worst case)



State of the Art: Replication

Overhead: 200%

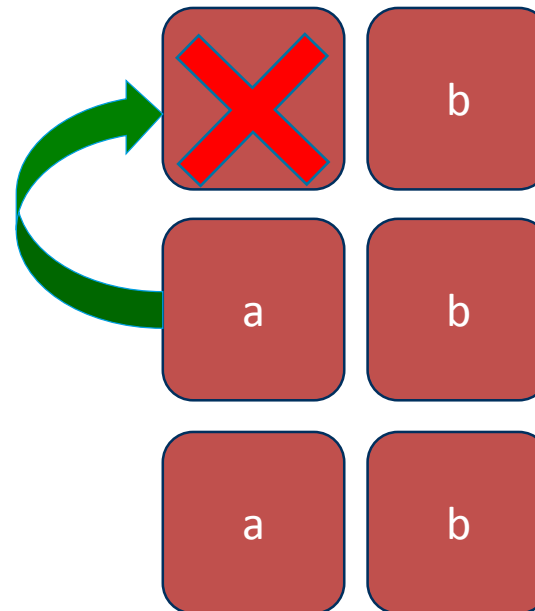
- High storage cost

Traffic to repair loss:

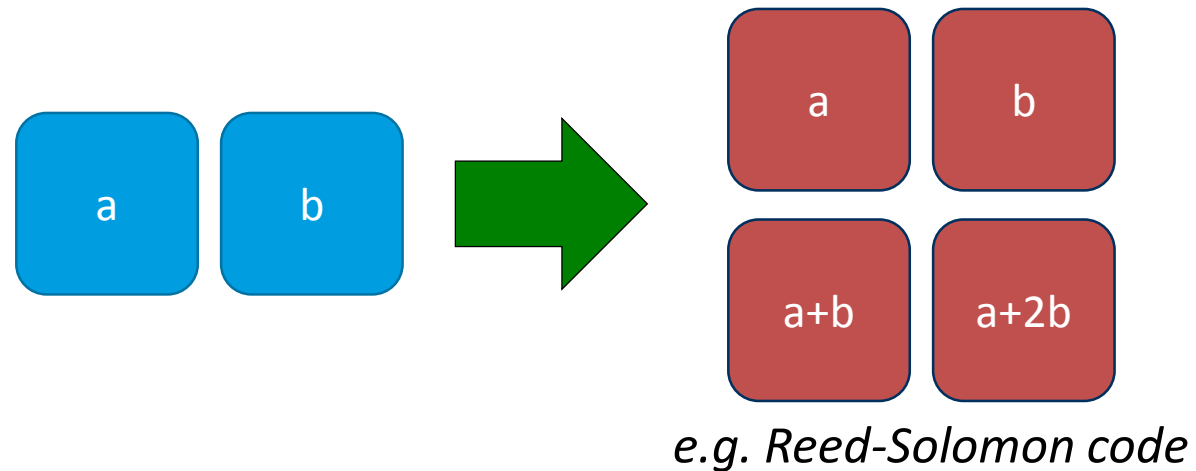
- 1 unit

Protection:

- 4 losses (best case)
- 2 losses (worst case)



State of the Art: RAID6



Overhead:

100%

State of the Art: RAID6

Overhead: 100%

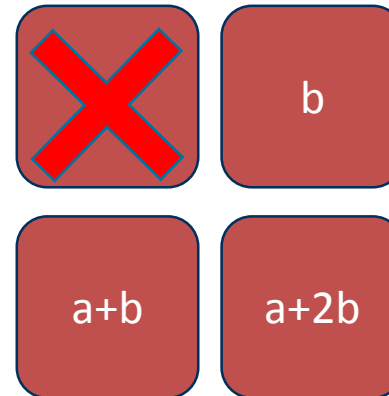
- Medium storage cost

Traffic to repair loss:

- 2 units

Protection:

- 2 losses (any case)



e.g. Reed-Solomon code

State of the Art: RAID6

Overhead: 100%

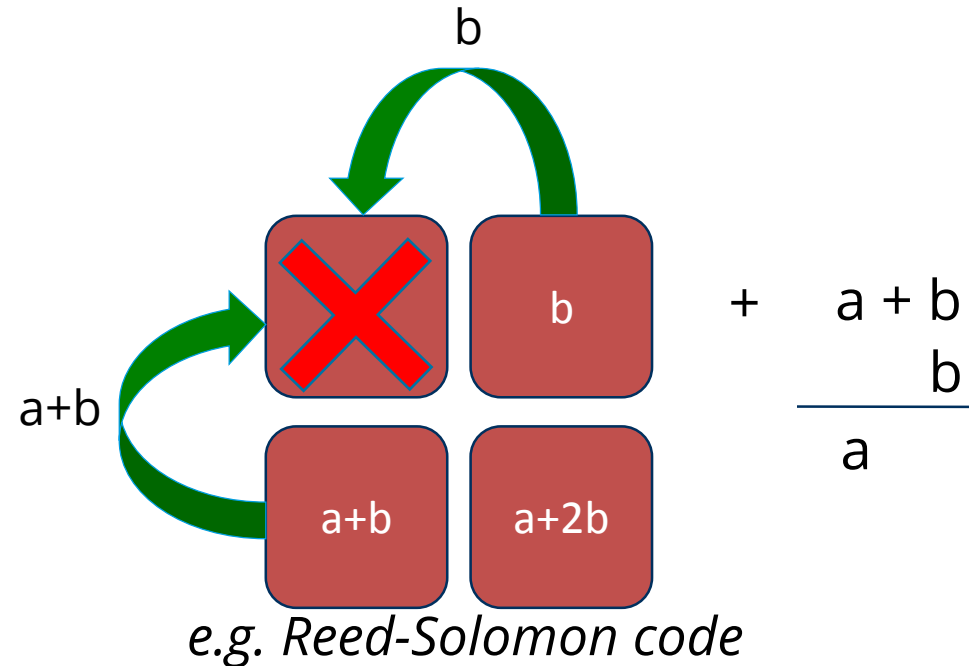
- Medium storage cost

Traffic to repair loss:

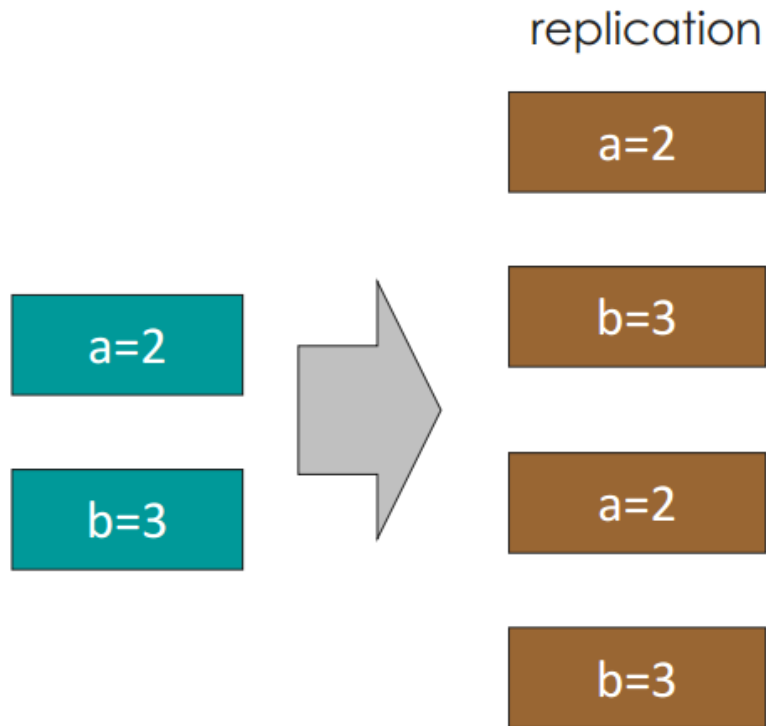
- 2 units

Protection:

- 2 losses (any case)

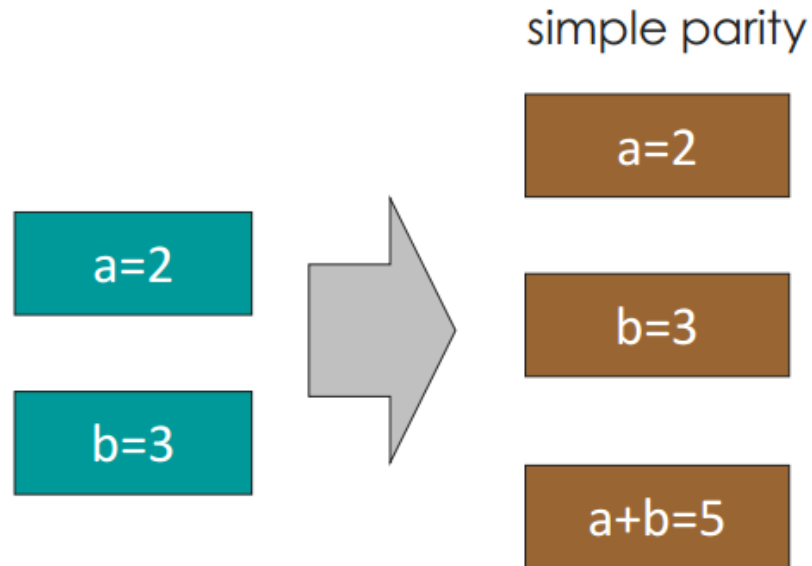


Simple Erasure Coding – Replication



- # of data units $k = 2$
- # of code units $n = 4$
- # of tolerable unit losses
 - worst case $r_w = 1$
 - best case $r_b = 2$
- overhead: $n/k = 2x$

Trivial Erasure Coding – Simple Parity



- # of data units $k = 2$
- # of code units $n = 3$
- # of tolerable unit losses
 - worst case $r_w = 1$
 - best case $r_b = 1$
- overhead: $n/k = 1.5x$

MDS Codes and Systematic Codes

MDS codes

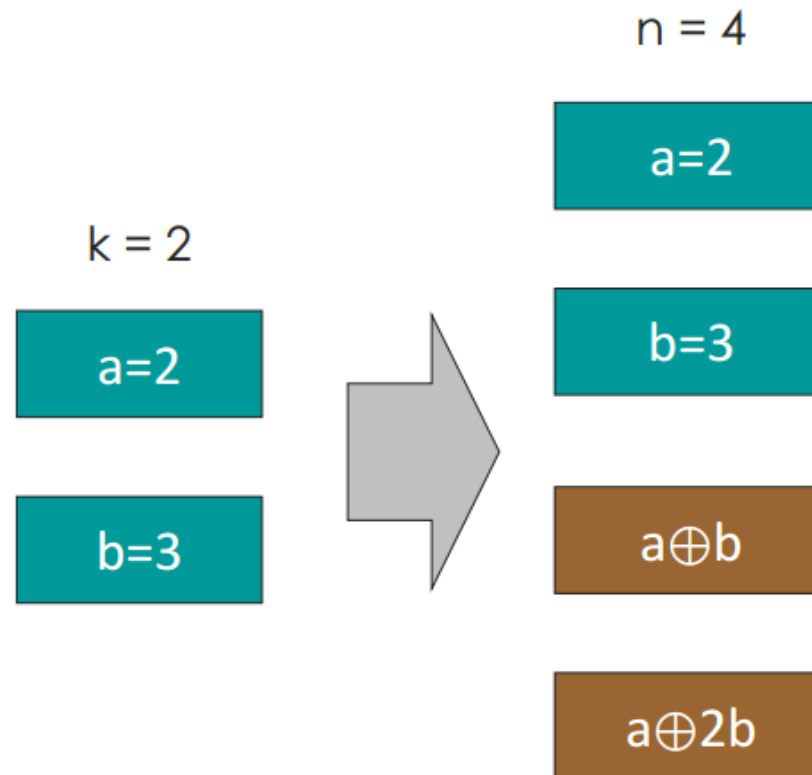
- Same number of tolerable losses (best vs. worst)
- Optimal overhead
- Extra computational complexity to perform coding

Systematic codes

- Direct access to the content without the need for coding if no losses
- Coded information only needed with losses and then also increased traffic

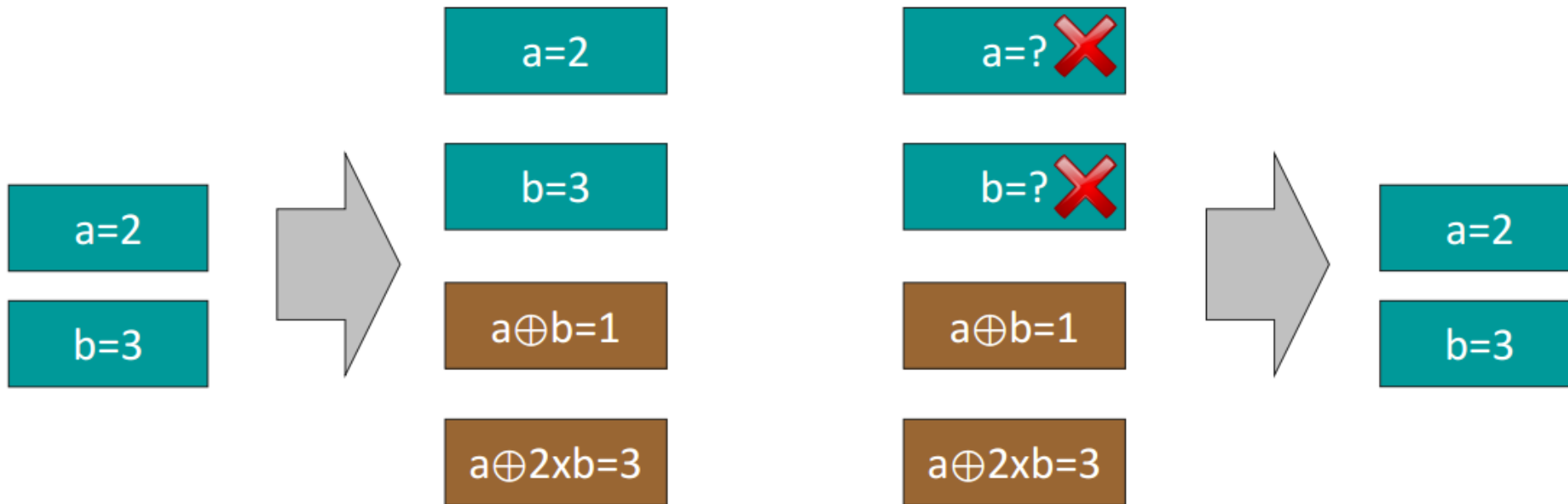
Reed-Solomon Example

- The most important class of MDS and systematic codes
- Example

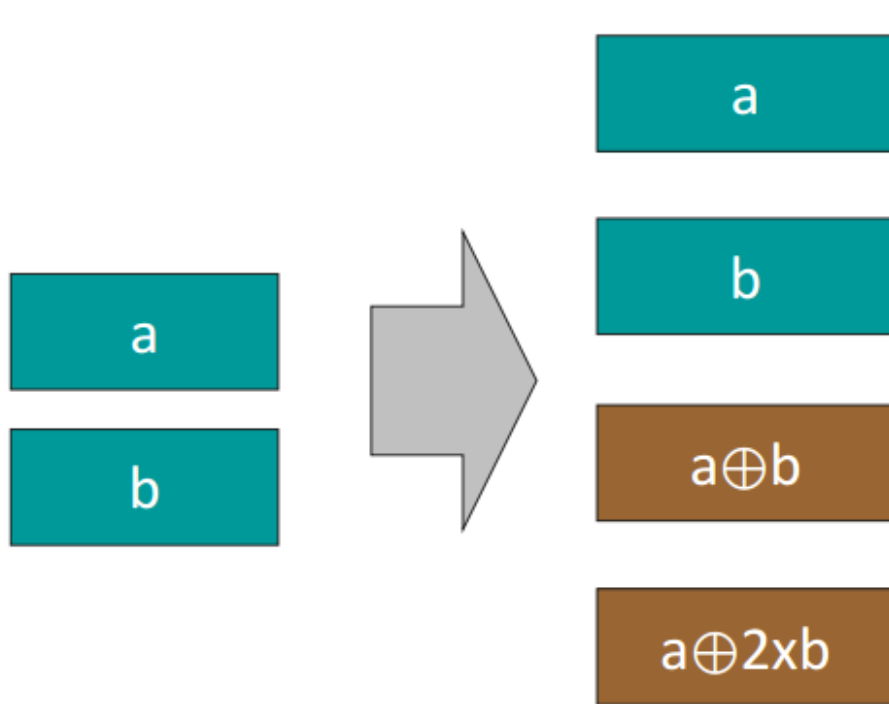


Reed-Solomon Example: Operations in FF

\oplus	0	1	2	3	\times	0	1	2	3
0	0	1	2	3	0	0	0	0	0
1	1	0	3	2	1	0	1	2	3
2	2	3	0	1	2	0	2	3	1
3	3	2	1	0	3	0	3	1	2



Reed-Solomon Example: Operations in FF



- **a** & **b** can be of any size
 - Network packet (1KB)
 - Storage disk (1TB)
- \oplus is very fast on CPU
- What about multiplication?

Repair Problem

More General Reed-Solomon

Can split data into k fragments and generate n fragments

Any k fragments would be enough to recover

Overhead: $(n-k)/k$

Traffic for repair: k

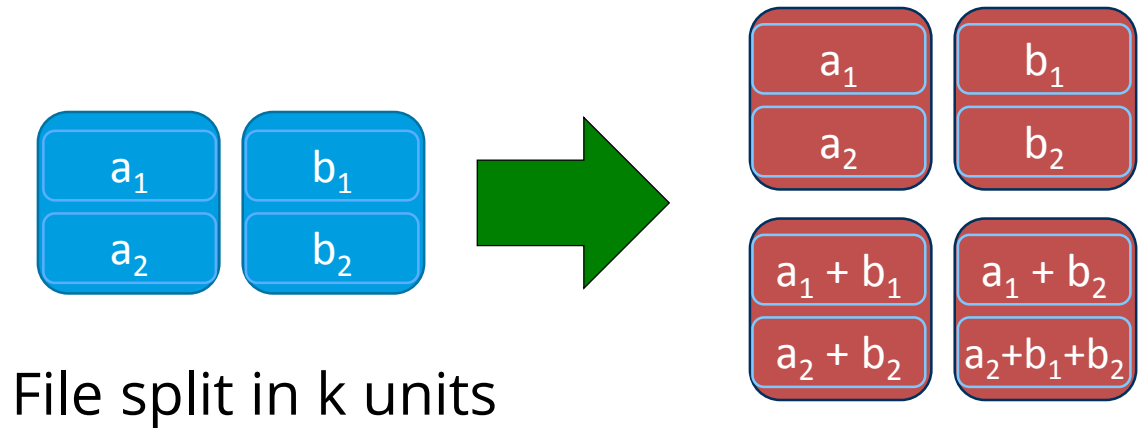
Protection: can recover from $n - k$ losses

Regenerating Codes

Goal: Minimize repair traffic with same fault tolerance as Reed-Solomon

- Faster repair
- Inherent trade-off of storage and traffic cost
- Network coding is key: recoding needed at the nodes/racks before sending

Regenerating Codes



Overhead:

100%

Regenerating Codes

Overhead:

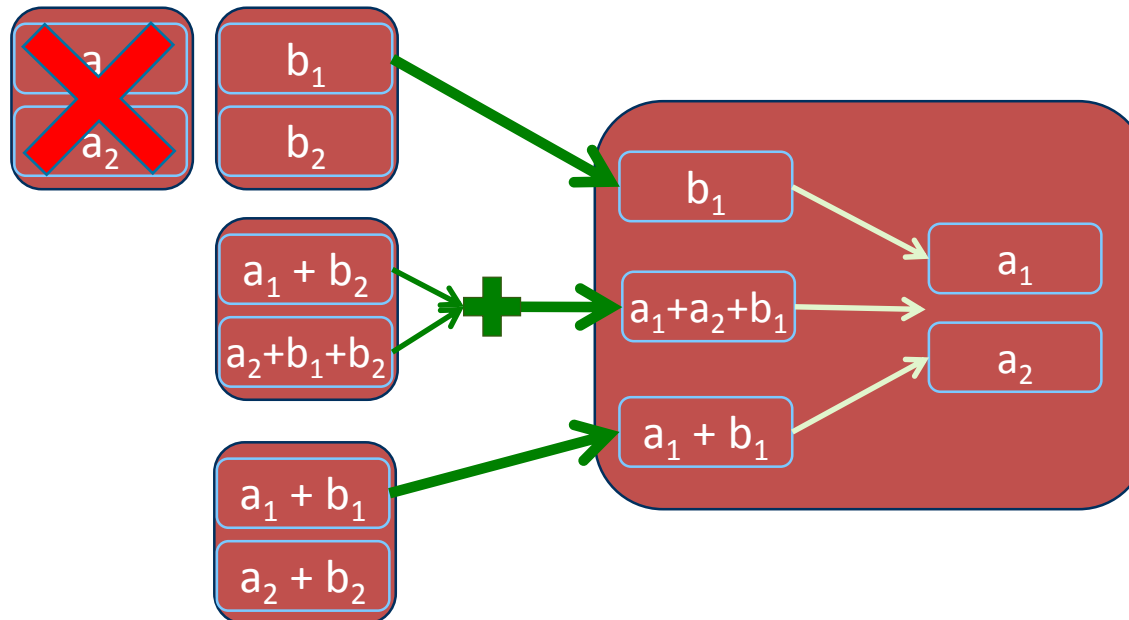
100%

Protection:

2 losses of nodes/racks

Repair traffic:

1.5 units \rightarrow $\frac{3}{4}$ of file size



Regenerating Codes

For a file of size B bits, split into k pieces and encoded into n pieces

Each piece is stored in a disk with B/k bits per disk

Theorem: it is possible to **functionally** repair a code by communicating:

$$\frac{n - 1}{n - k} \frac{B}{k} \text{ Bits}$$

As opposed to transmitting B bits (naive cost)

Previous Example: For $n = 4$, $k = 2$, then at least $\frac{3}{4} B$ bits are needed (and we used exactly that)

New Challenges in Storing

So far the structure has been static

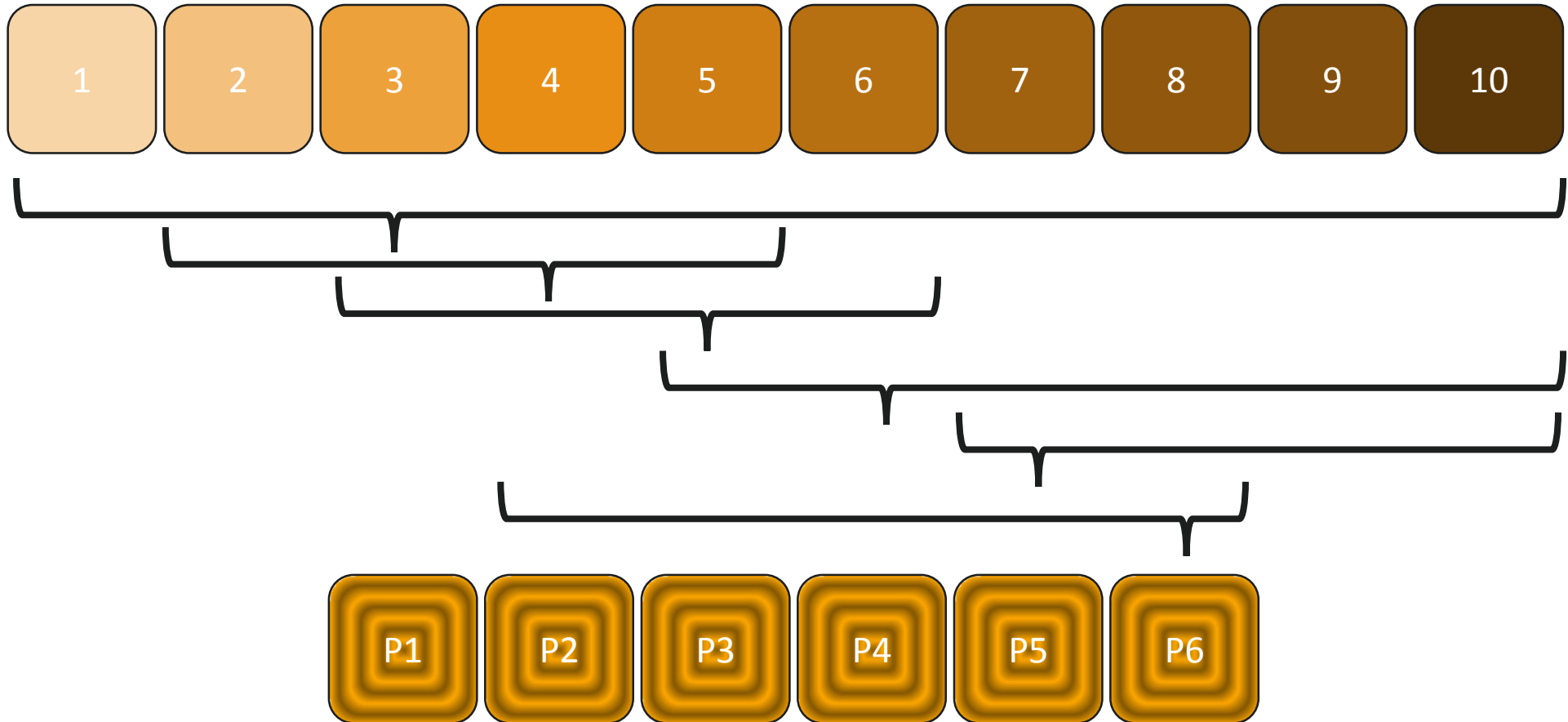
But new use cases require dynamic / versatile codes

- Edge caching
- Storage in mobile nodes

What RLNC can do

- New cloud storages can be filled with available information
- Some information might become lost or not available
- Filling storage on arrival without storing data in memory before decoding

RLNC



Overhead: scalable

What we looked at

- Reliability
- Privacy and securityStorage space
- Costs
- Download speed
- Amount of transferred data
- Computational overhead of network coding

Reliability

Failures are not uncommon in the world of cloud computing:

- Google (partial): 16th of August 2013 , 18th-19th of March 2013, 11th of August 2008
- Microsoft Azure: 22nd of February 2013, 13th of March 2009
- Amazon S3: 31st of January 2013, 20th of July 2008
- Dropbox: 10 th of January 2013

Permanent data loss is possible too:

- Microsoft & T-Mobile Sidekick: 11th of October 2009

However, the chance that more than one provider is down at a given time is very low

Solution: distribute the data to multiple providers with some added redundancy

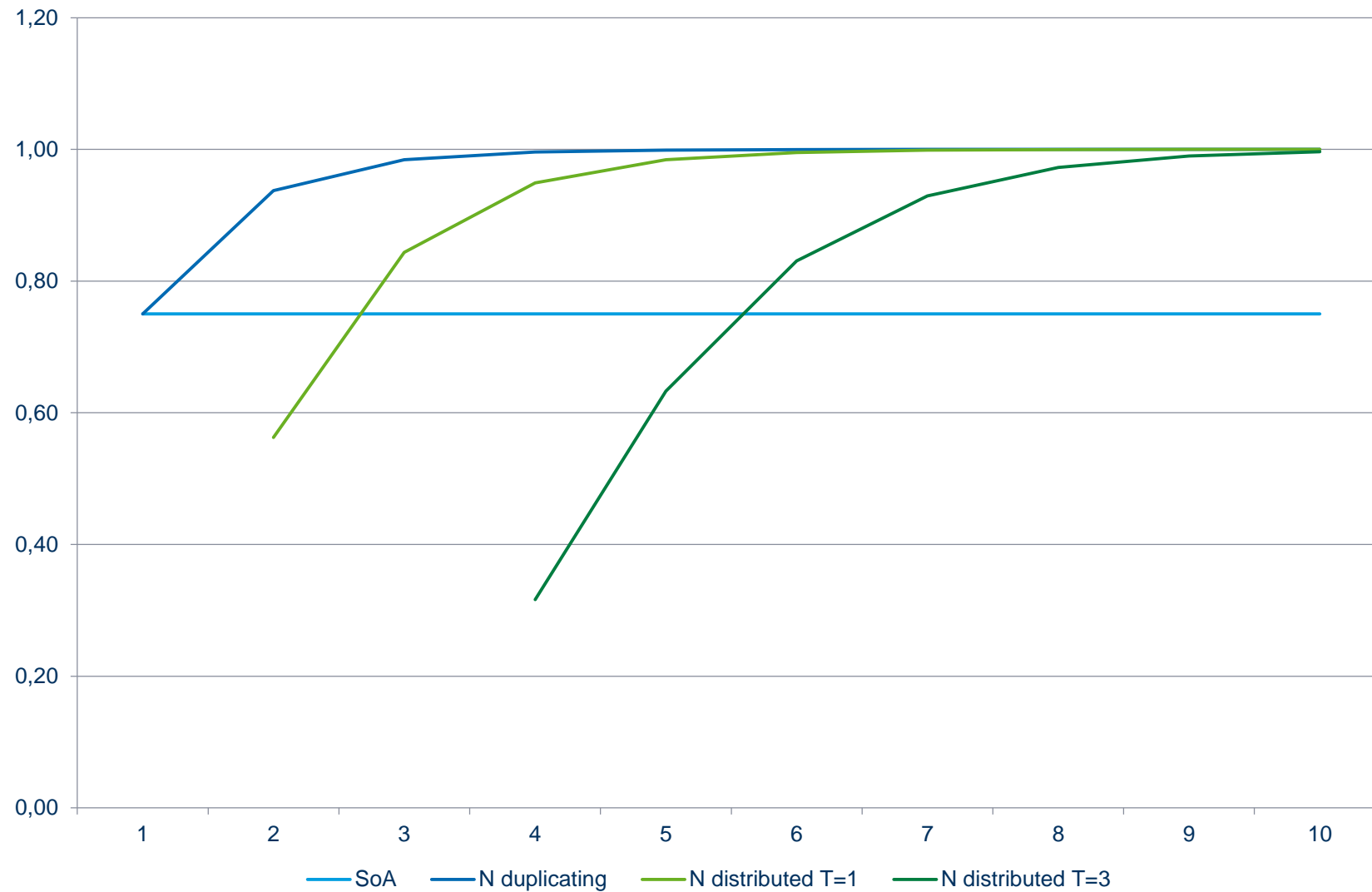
Reliability

It is relatively simple to calculate the probability that the data is not available at a given time:

$$\mathcal{P}(T, N, p) = 1 - \sum_{t=0}^{t=T} \binom{N}{t} \cdot (1 - p)^{N-t} \cdot p^t$$

- N – number of clouds
- T – number of simultaneous cloud failure to be able to sustain
- p – the probability that a provider is unavailable

Reliability



Privacy and Security

One of the biggest hurdles holding back cloud computing is lack of trust in the providers

Security breaches:

- Sony PlayStation Network: 17th and 19th of April 2011 – 77 million accounts stolen
- Smaller breaches have occurred with most cloud computing providers

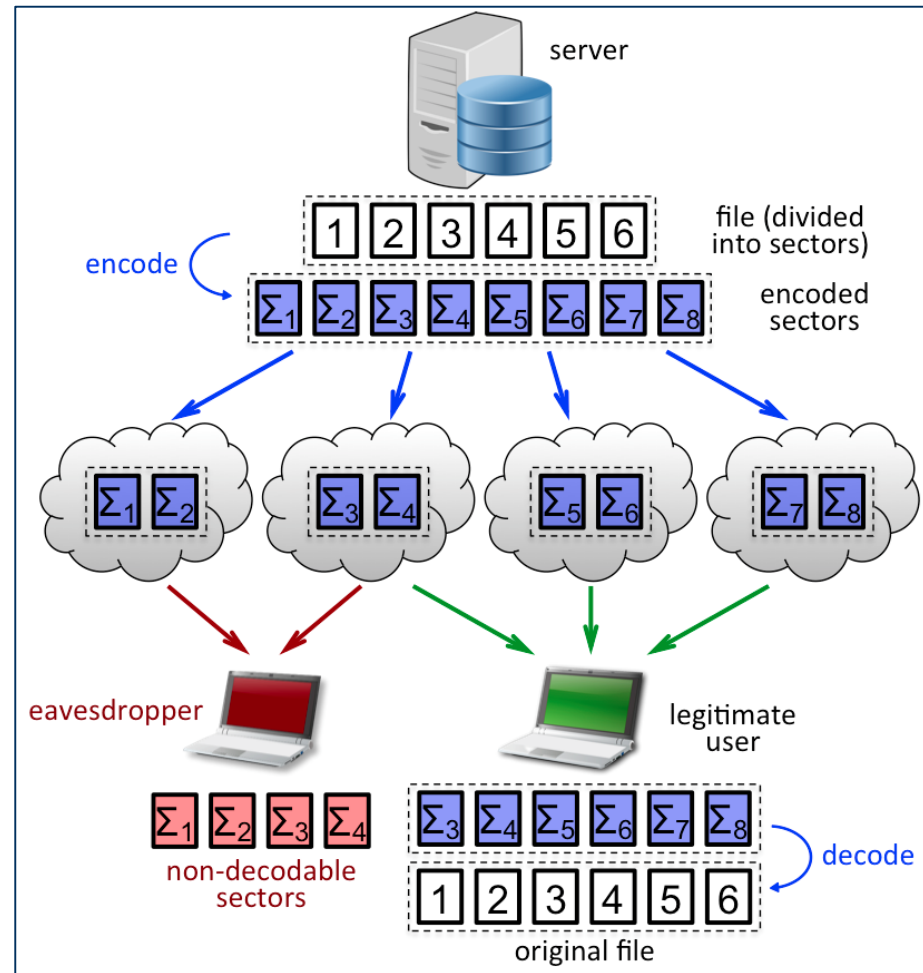
Government agencies have access to data stored by cloud services in some countries:

- PRISM

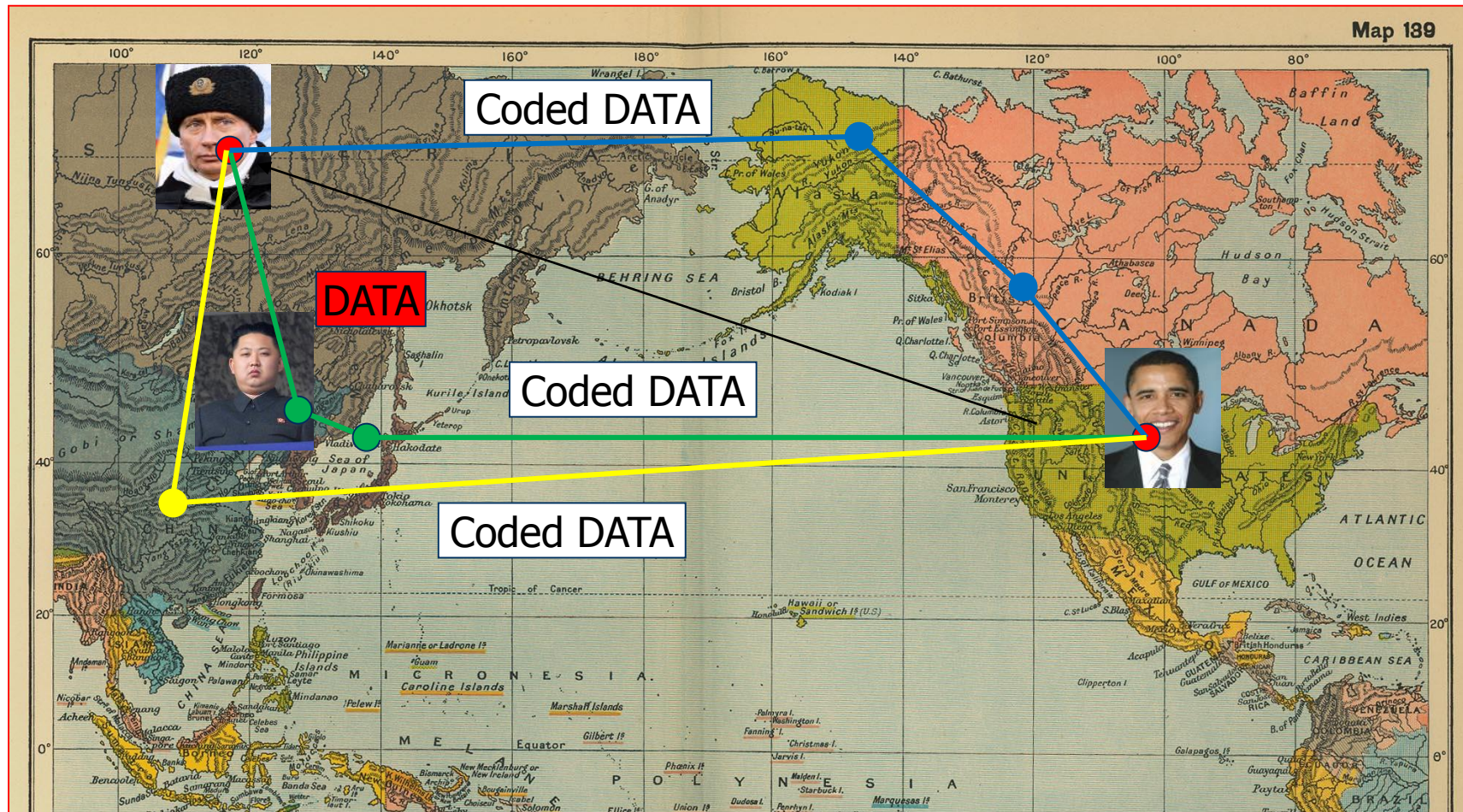
By using network coding, it is possible to force the attacker to need to compromise several of the providers to access any of the data

- The number of providers that need to be hacked can be set arbitrarily.

Coding as a Measure Content of Protection



Some insights for security



Erik Kline, USC ISI, kline@isi.edu

Storage costs

Each provider grants some free storage space. With our approach you can add up the capacities

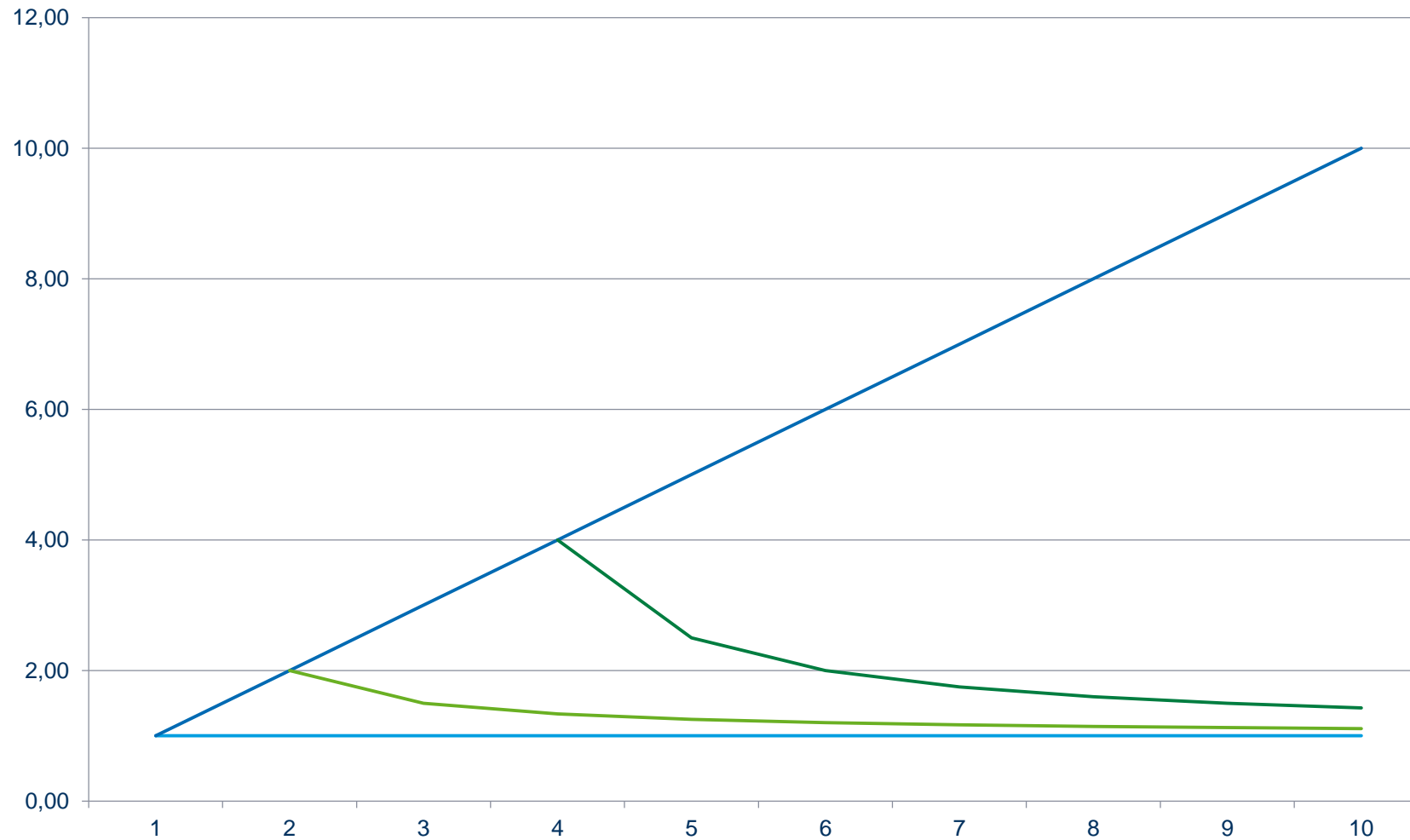
Reliability can be easily quantified in terms of costs. The required storage space is:

$$H(N, T, F, P) = \frac{N}{N - T} \cdot F = F + F \cdot \frac{T}{N - T}$$

- F – file size in bytes

Clearly if $N \gg T$, the storage cost becomes negligible

Storage/Cost



— SoA — N duplicating — N distributed T=1 — N distributed T=3

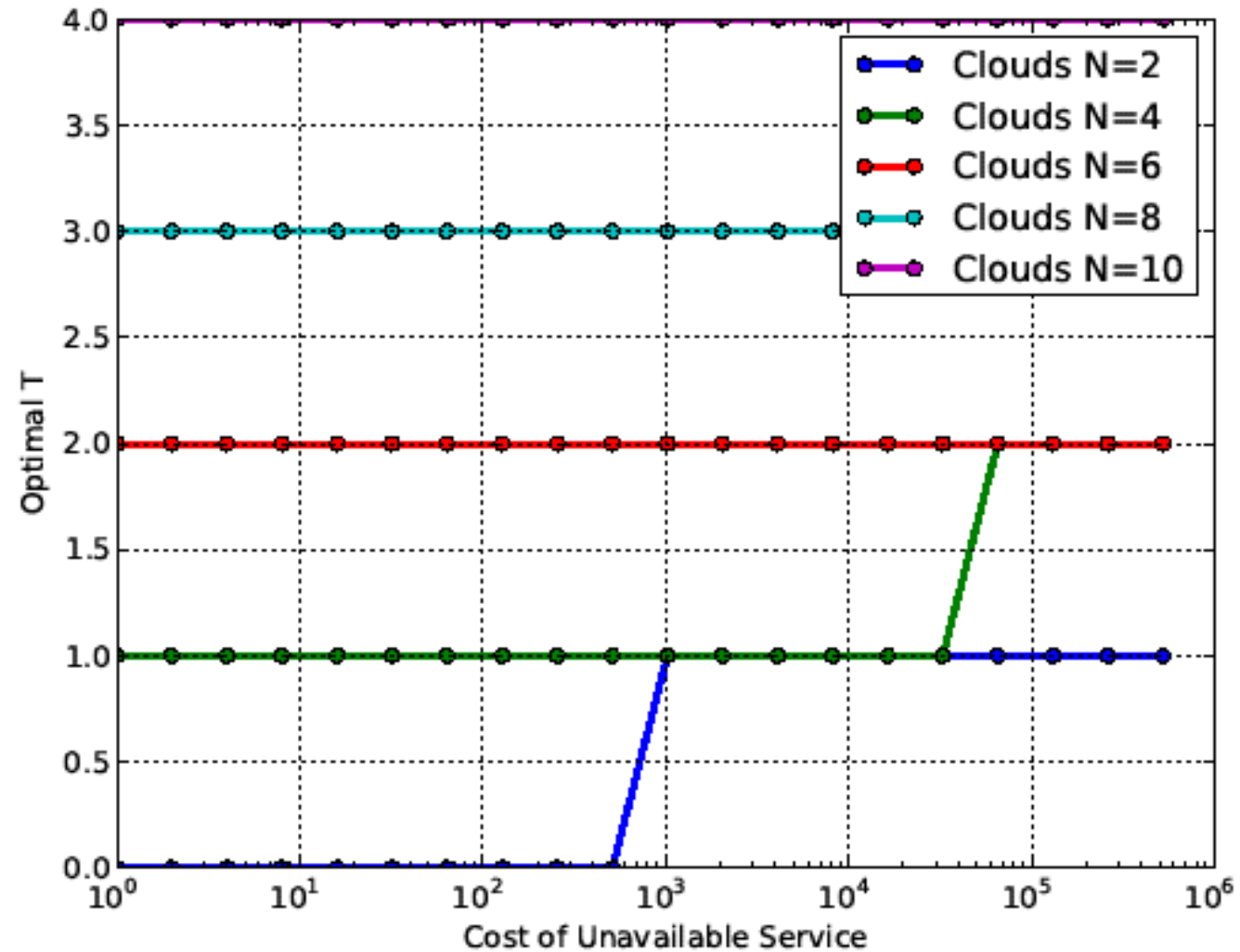
Total costs

The goal is to find the number of redundant clouds (T) for which the total cost is minimal:

$$\min_T H(N, T, F, P) + U_c \cdot \mathcal{P}(T, N, \{p_i\})$$

- U_c – cost of data being unavailable
- $H(N, T, F, P)$ – storage costs

Finding the optimal T



Download speed

Considering clouds with various download rates (R), the fraction of file requested from cloud c_i is:

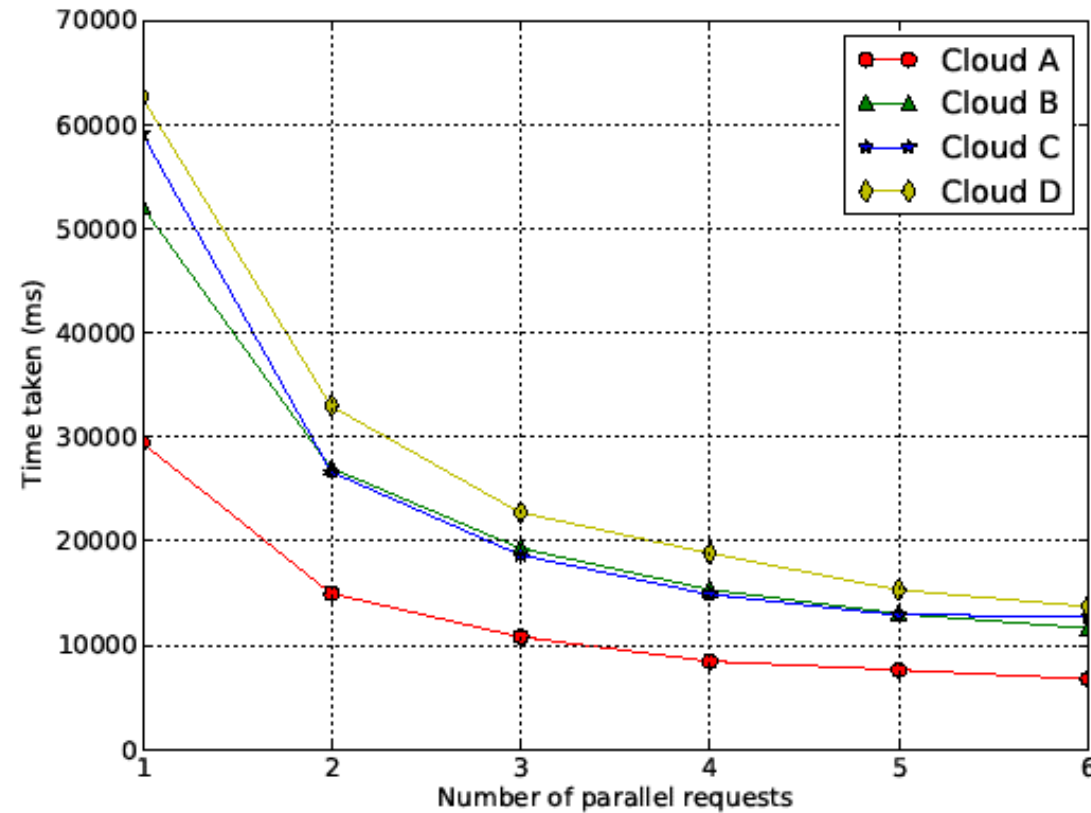
$$\alpha_i = \frac{R_{(c_i, u)}}{\sum_{i=1}^N R_{(c_i, u)}}$$

$$P_i = \max \left(\frac{1}{N - T}, \alpha_i \right) = \max \left(\frac{1}{N - T}, \frac{R_{(c_i, u)}}{\sum_{i=1}^N R_{(c_i, u)}} \right)$$

If we chose to optimize for download speed whilst still ensure a give level of reliability, the fraction of data stored in cloud should c_i be at least:

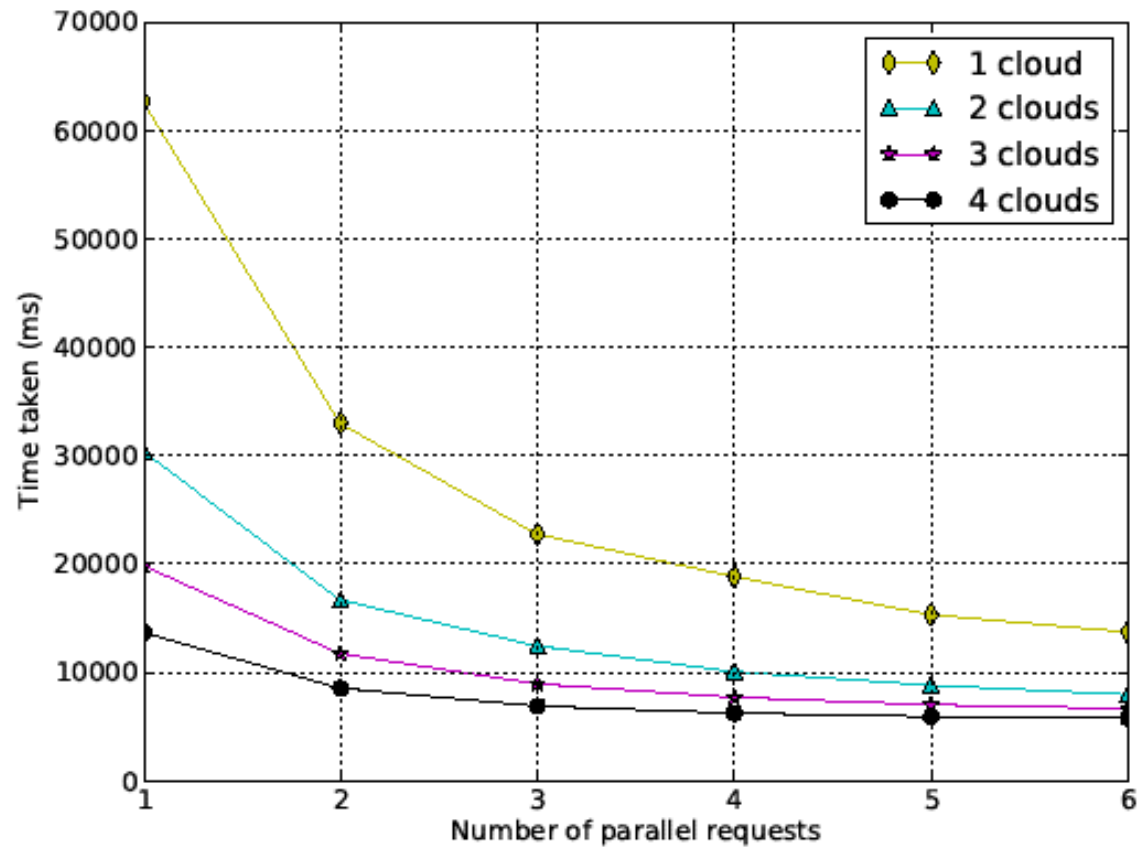
Download speed – individual clouds

We looked at how each cloud performs on its own. 16 MB file, packets of 512k



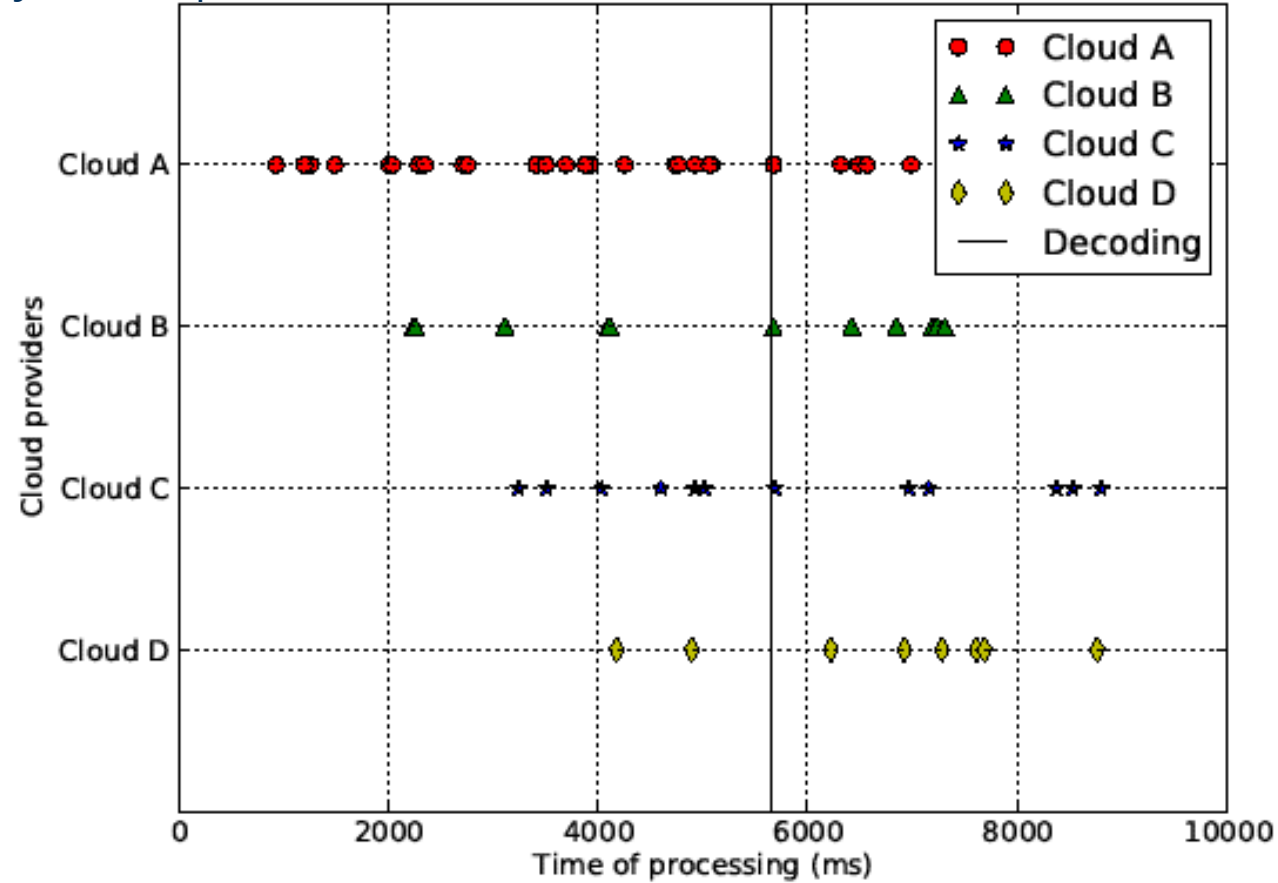
Why use multiple clouds?

Collaboration



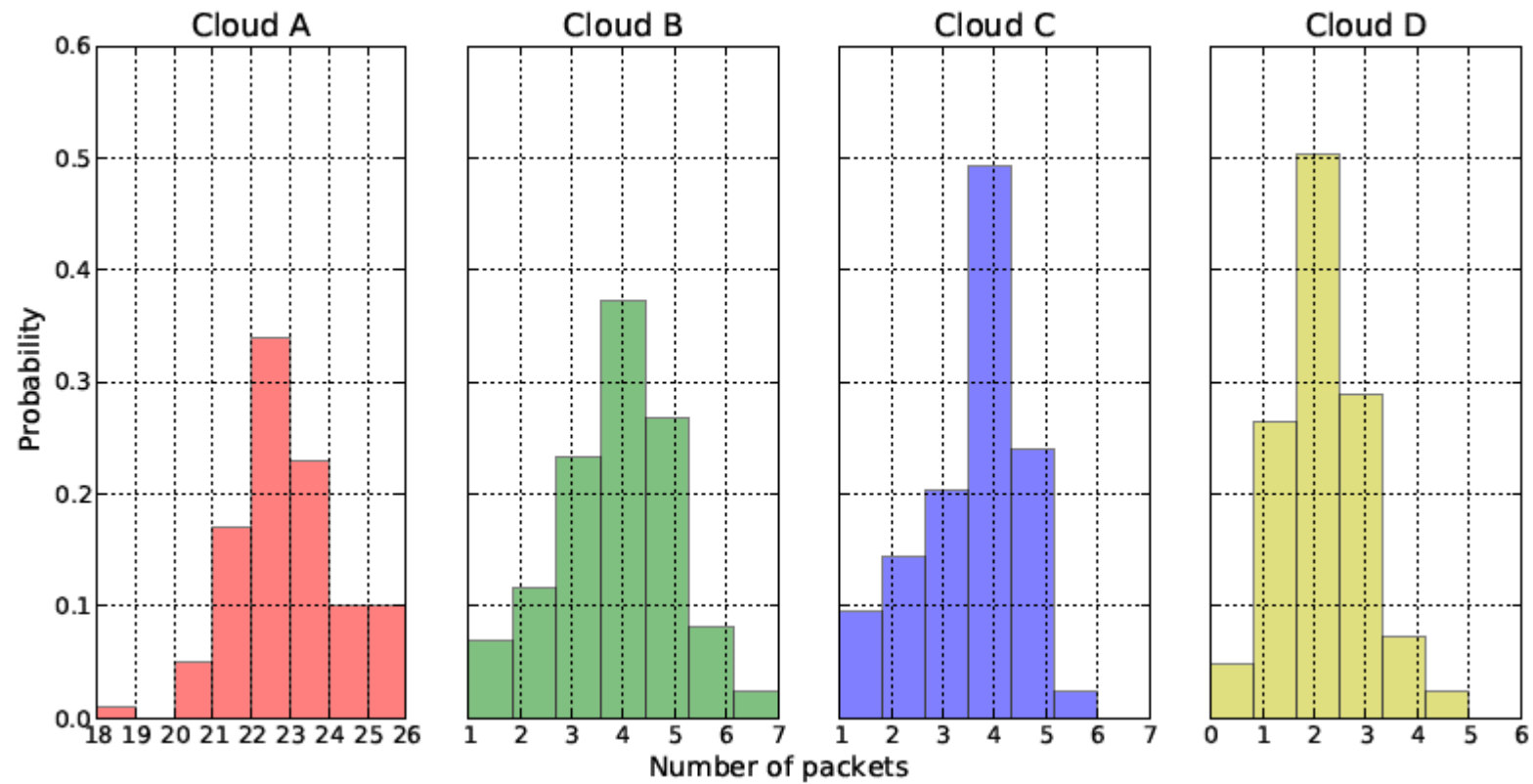
Why use network coding?

Collaboration is usually not simple

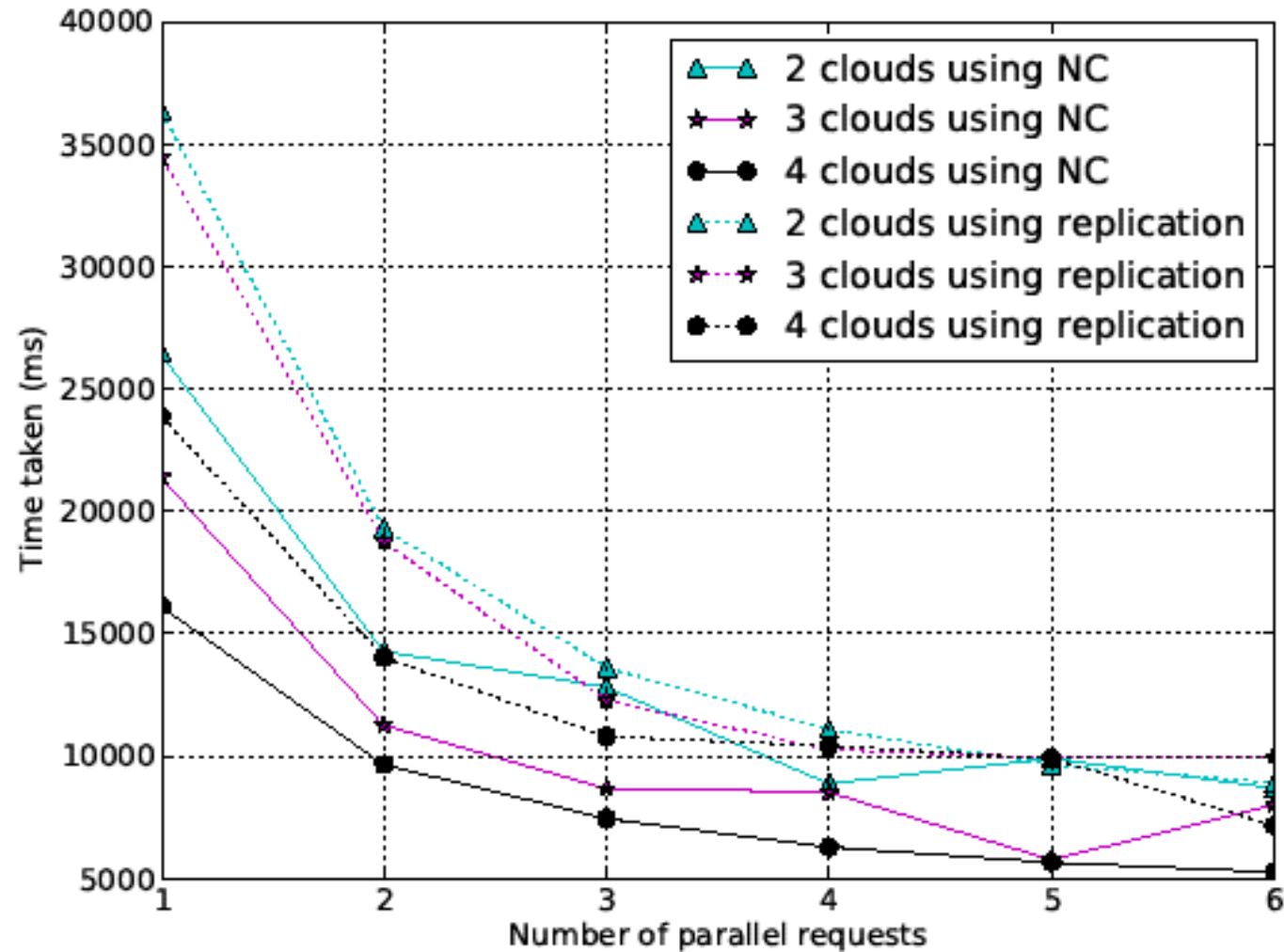


Why use network coding?

Conditions change.



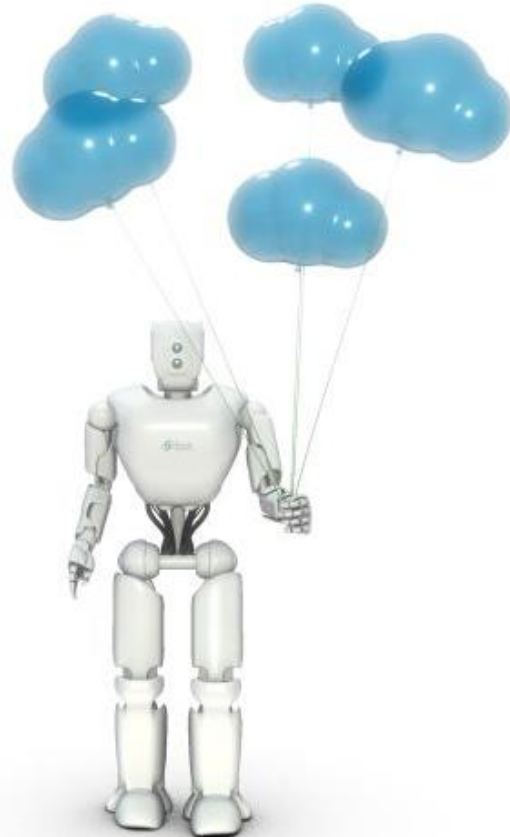
Comparing NC to replication redundancy



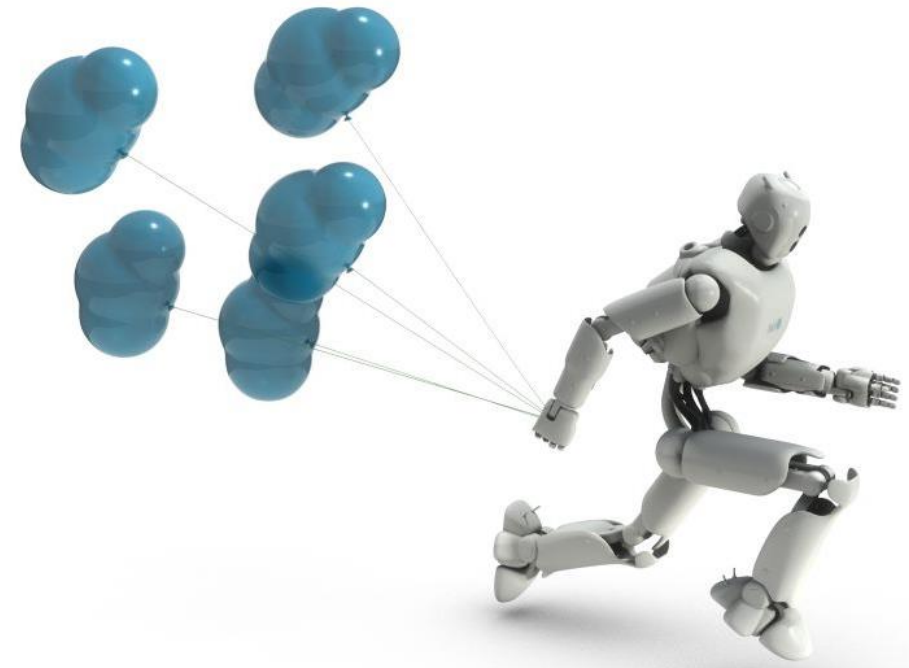
Cloud Evolution



Single/Static

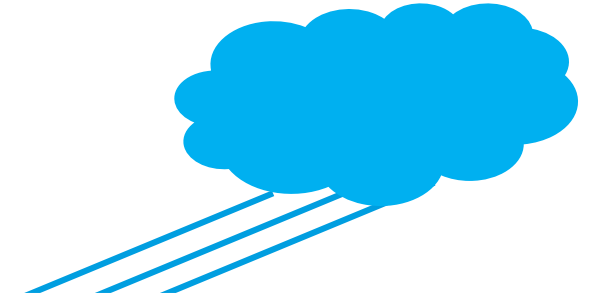
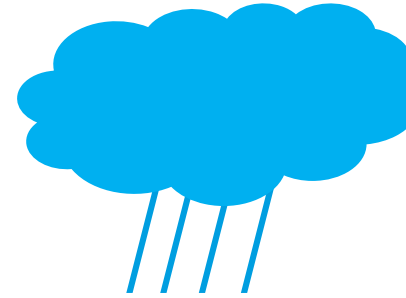
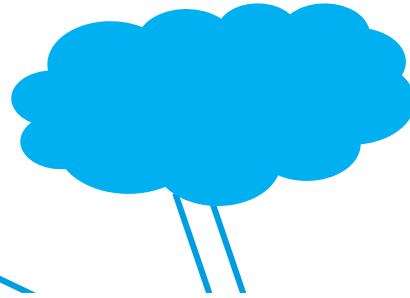
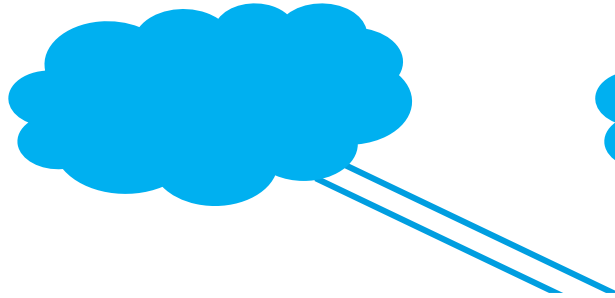


Distributed/Static



Distributed/Agile

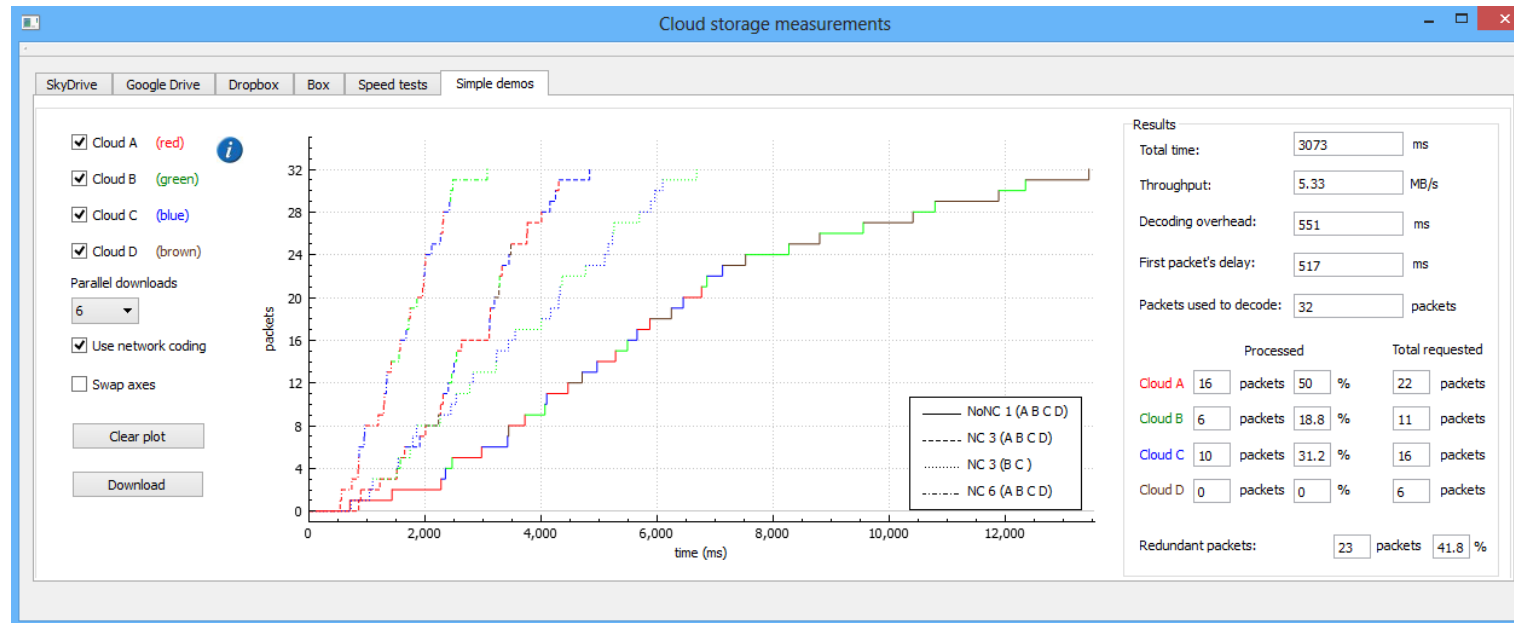
Example: Distributed Cloud



Measurement campaign

Implementation using commercial cloud solution (Dropbox, Box, Skydrive, and Google Drive) to:

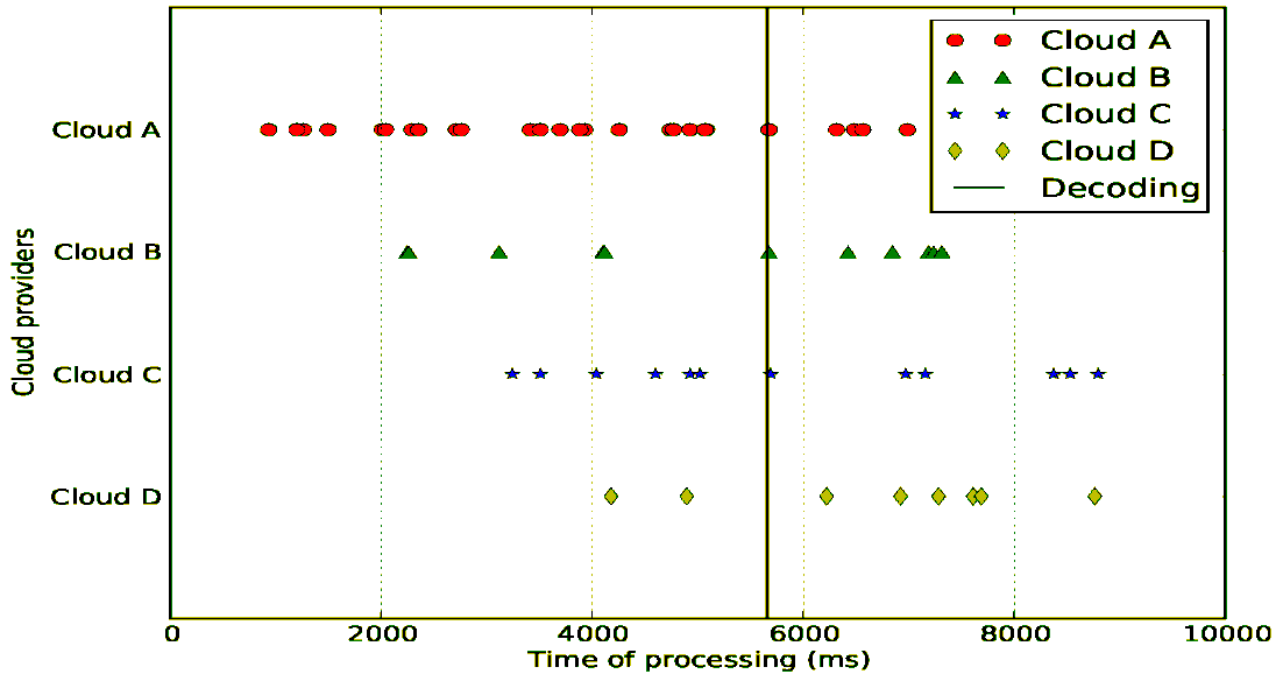
- Showcase theoretical results
- Perform practical measurements



Simple UI to be able to quickly explore different scenarios.

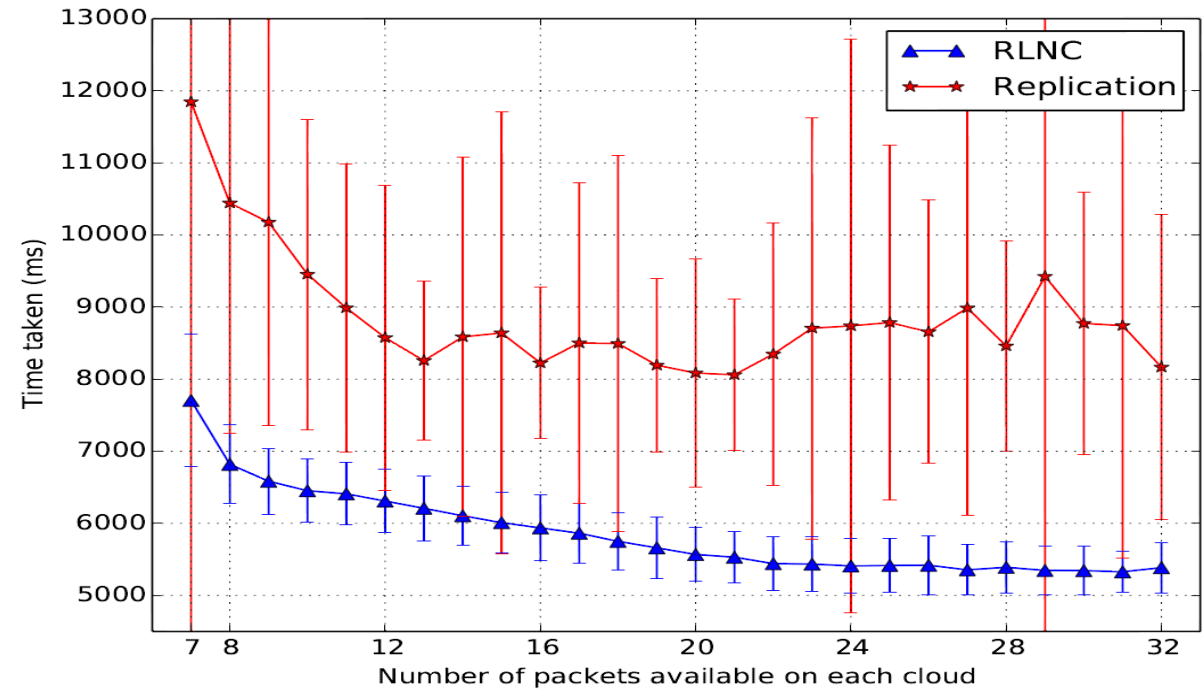
Example: Distributed Cloud

- Heterogeneity (4 clouds)



- Speed-Up (5 clouds)

RLNC does not need full degree of freedom



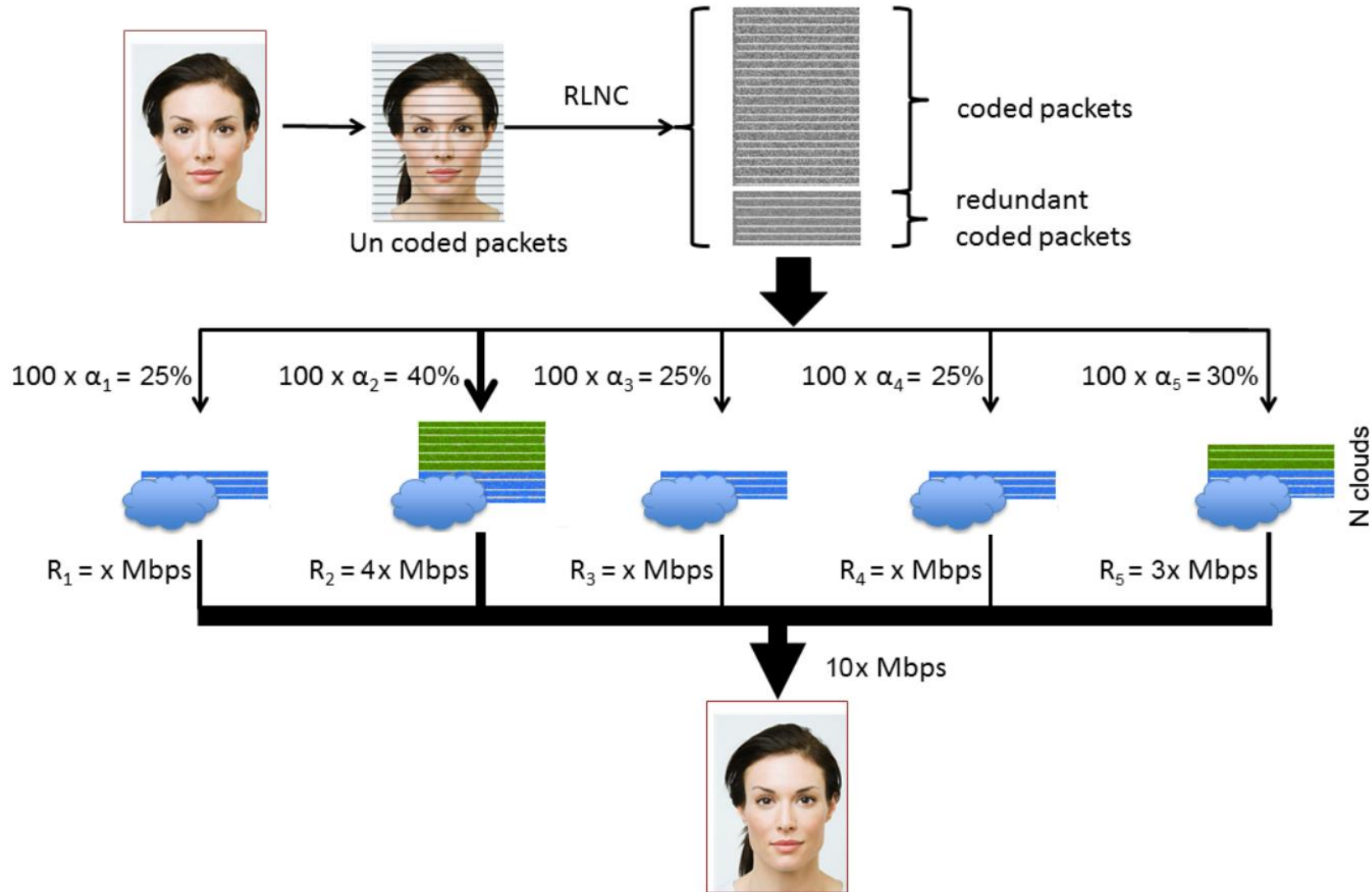
- M. Sipos, F.H.P. Fitzek, D. Lucani, and M.V. Pedersen, "Dynamic Allocation and Efficient Distribution of Data Among Multiple Clouds Using Network Coding," in IEEE International Conference on Cloud Networking (IEEE CloudNet'14), Oct. 2014.
- M. Sipos, F.H.P. Fitzek, D. Lucani, and M.V. Pedersen, "Distributed Cloud Storage Using Network Coding," in IEEE Consumer Communication and Networking Conference, Jan. 2014.

Storage Benefits in Dynamic Settings

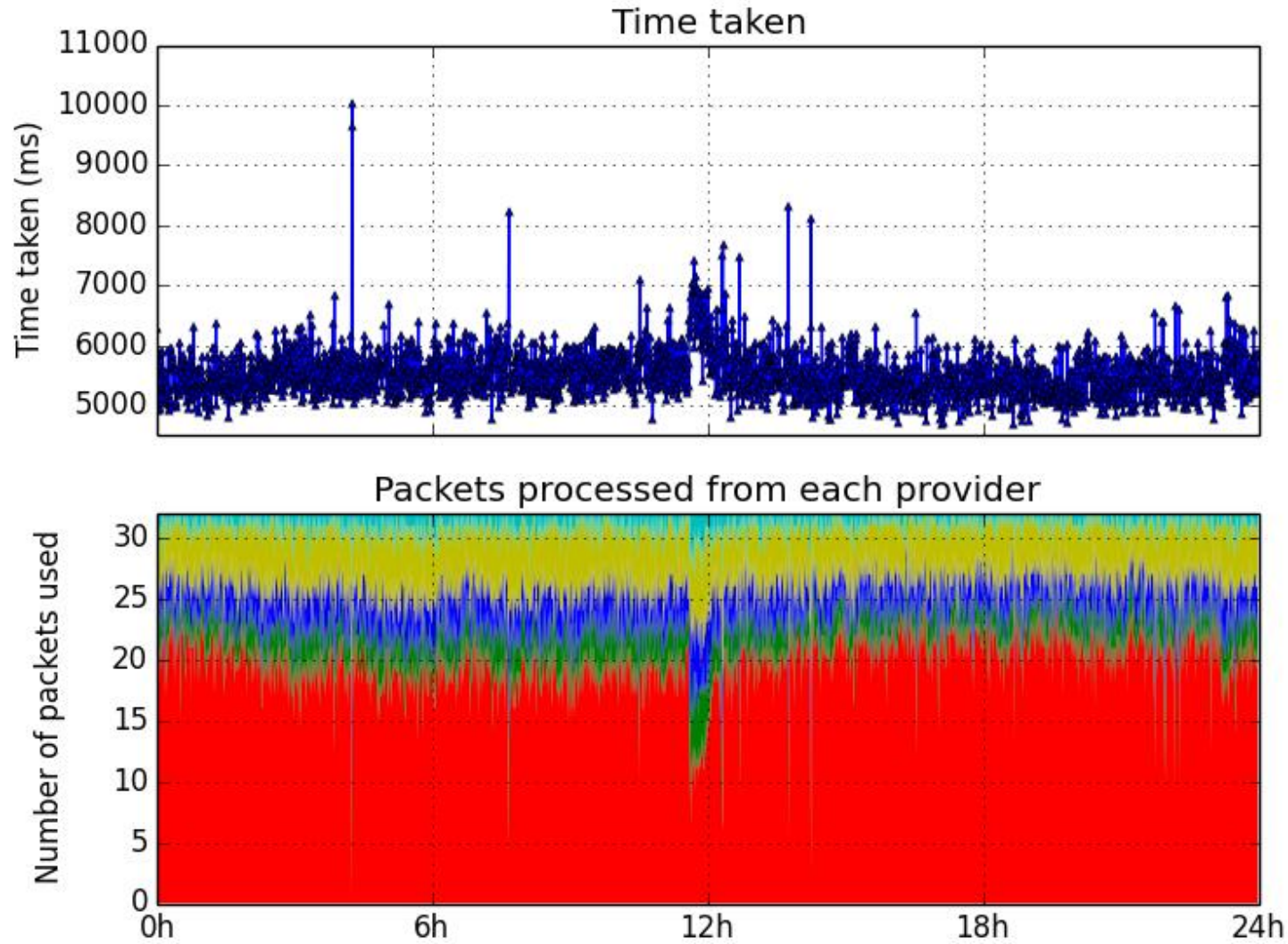
Comparing a FULLY decentralized approach (no coding, RS, RLNC) versus a centralized version (no coding, RS).

Dynamic: Not all racks are available temporarily (load, traffic, power)

Dynamics in Distributed Clouds



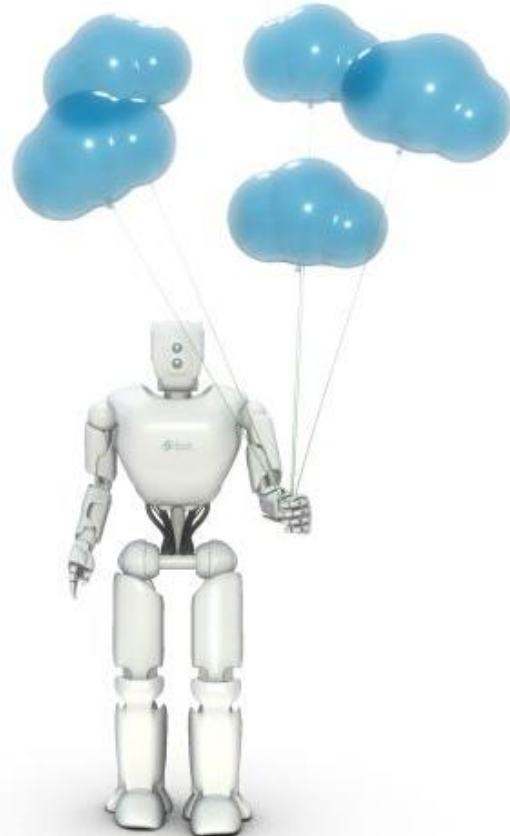
Dynamics in Distributed Clouds



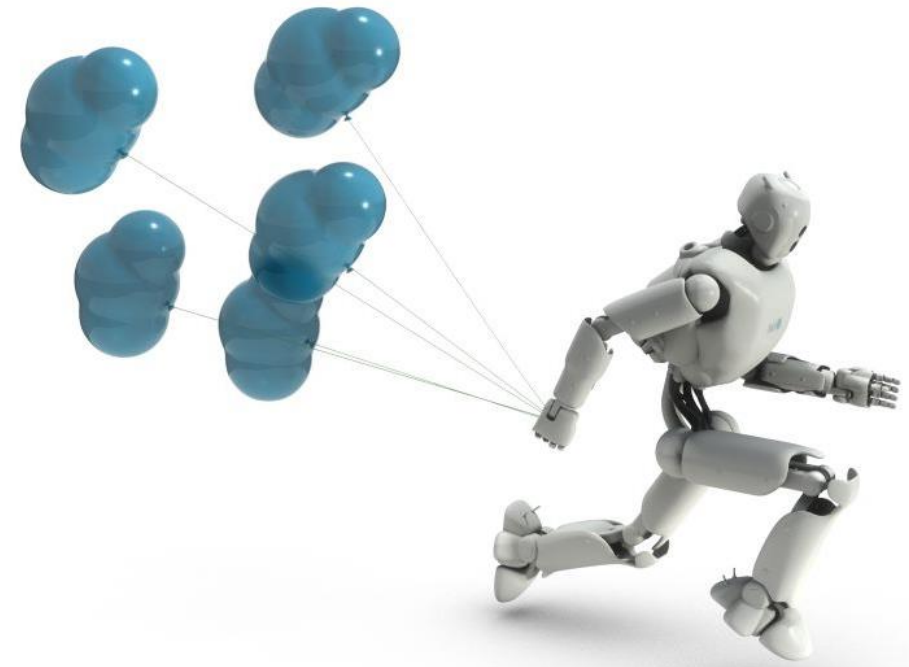
Cloud Evolution



Single/Static

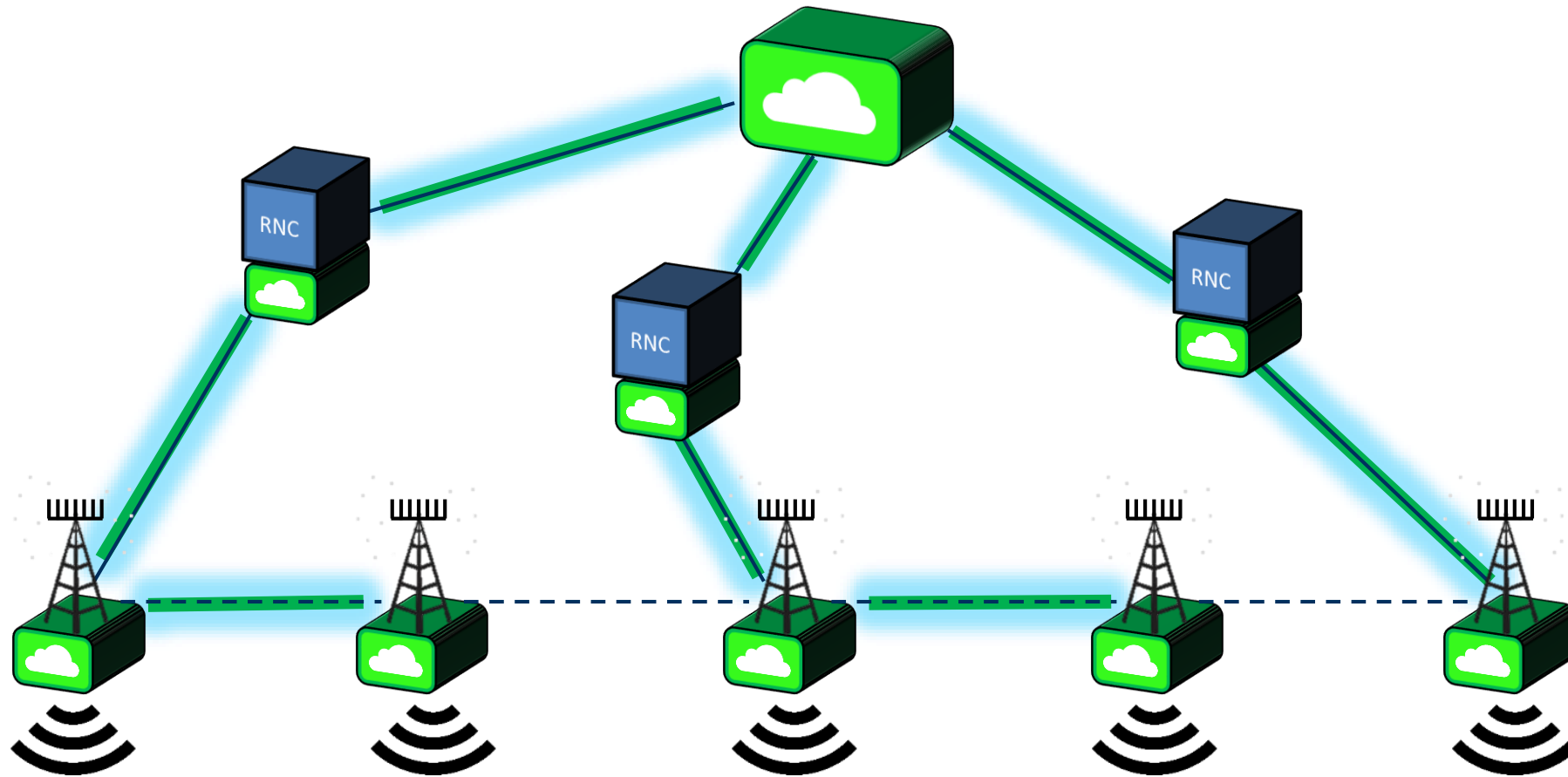


Distributed/Static

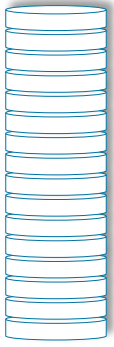


Distributed/Agile

Mobile Edge Cloud / Micro Cloud / Cloud



G=20



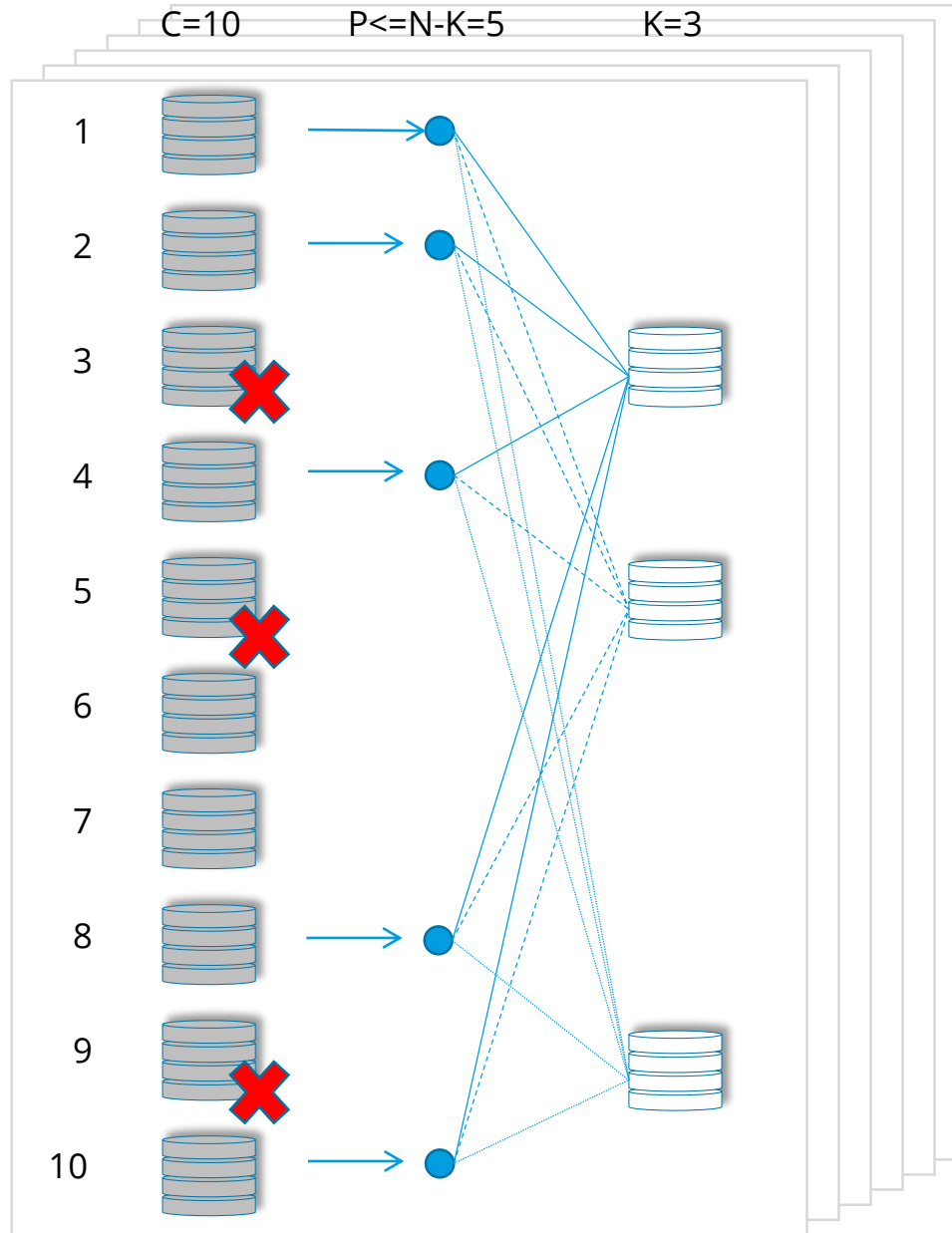
Q=4



C=10

$P \leq N - K = 5$

K=3



Rounds

Kodo Simulations 7b8dcef

Simulation Rank diagram 3D loss diagram Recoding strategies diagram

No. of Clouds: 10
No. of Storage: 4
Type of Coding: Network Coding
Symbols: 20
Galois Field: GF(2)
Symbol size: 4
Rounds: 100
Shootout number: 1
Strategy type: Post recoding
Alpha:
No. of Parents: 0 1 2 3 4 5 6 7 8 9

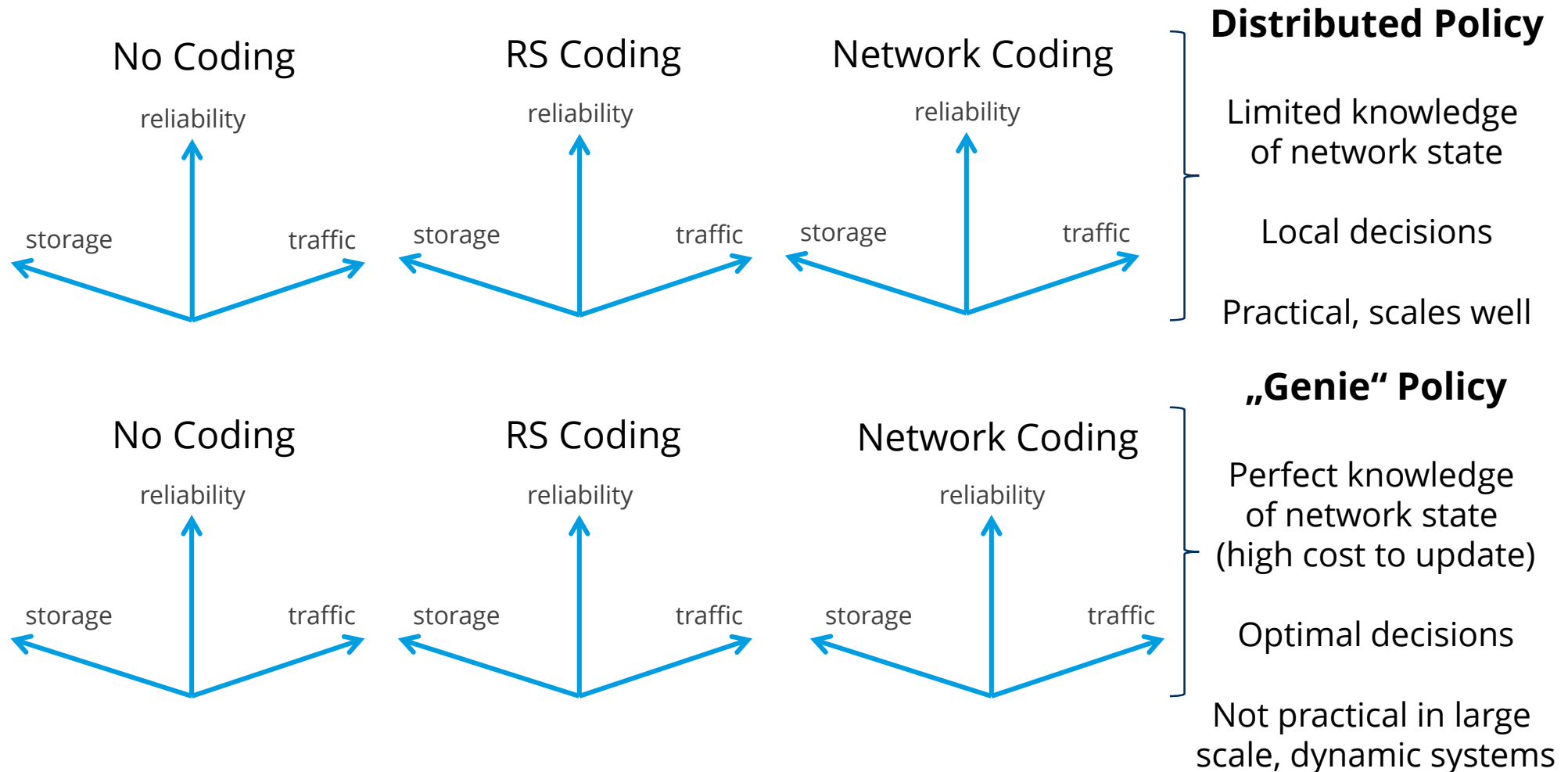
Calculated parameters

File size:
Overhead:
Header size:
Payload size:
Redundancy:
Traffic:

Operations
Start Clear

Simulation log

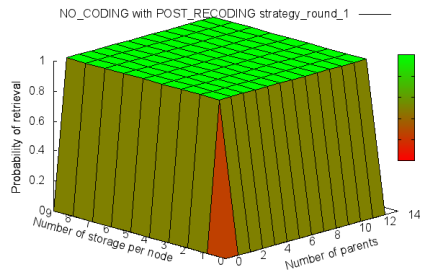
Data Survival Over Time



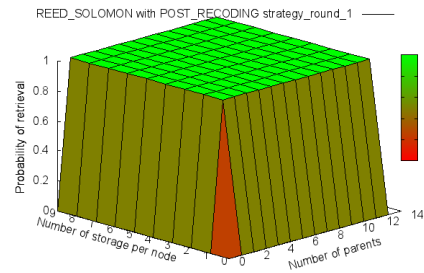
Current systems are somewhere in between these two policies

Data Survival Over Time

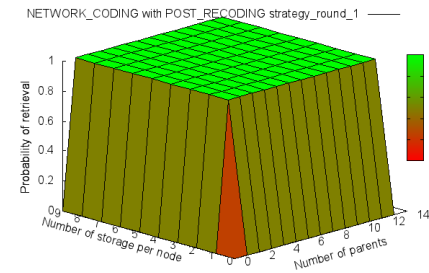
No Coding




RS Coding



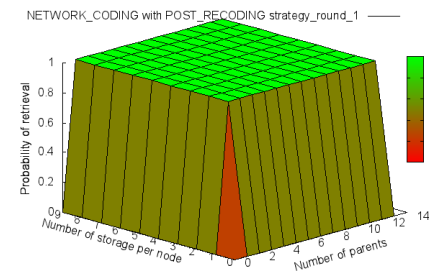
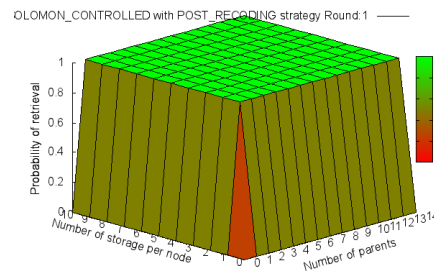
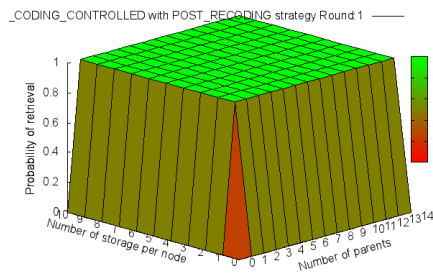
Network Coding



Distributed Policy

state-less 

„Genie“ Policy



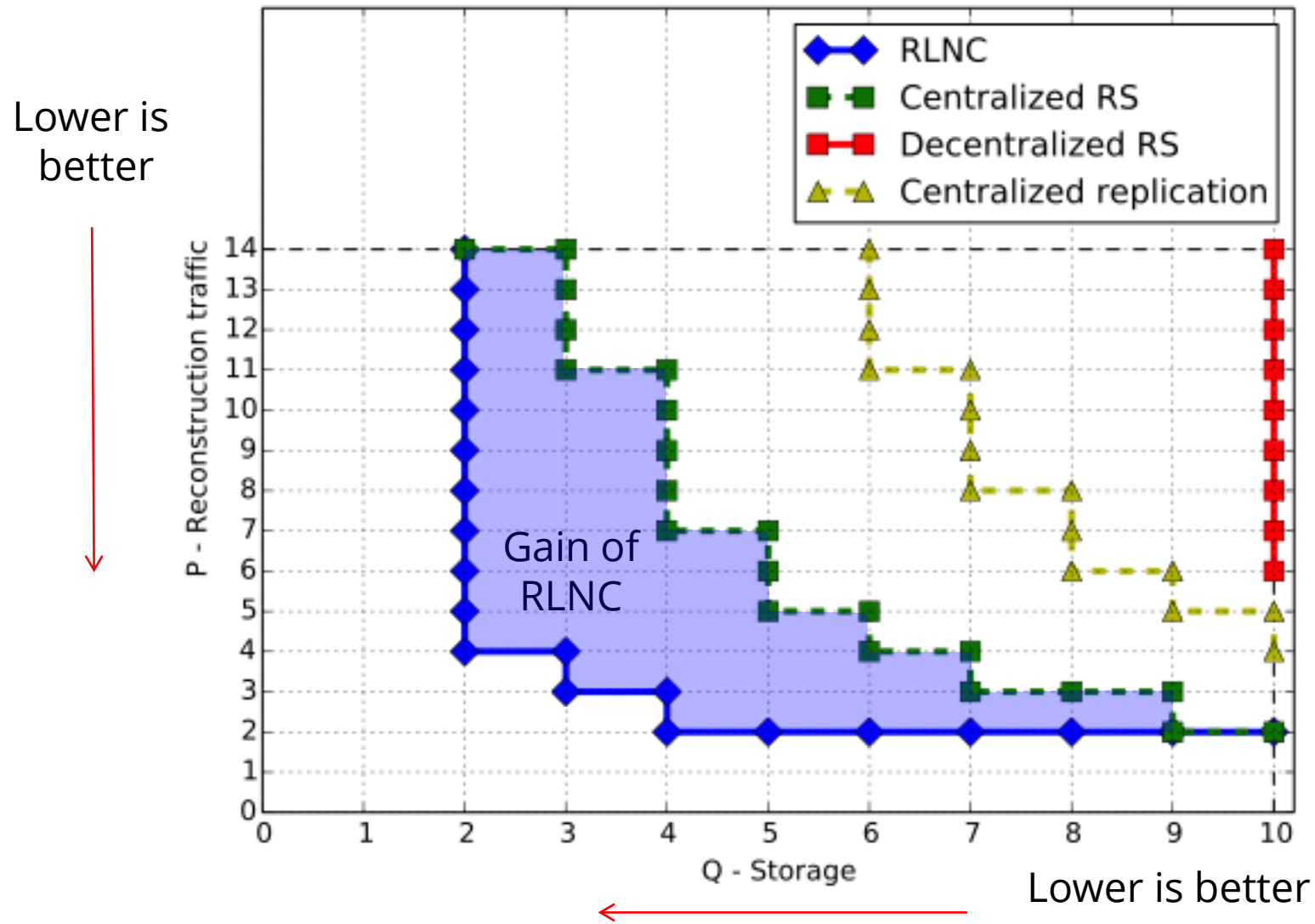
Over best alternative, NC buys you:

- Up to 65% reduction in storage
- Up to 71% reduction in network use

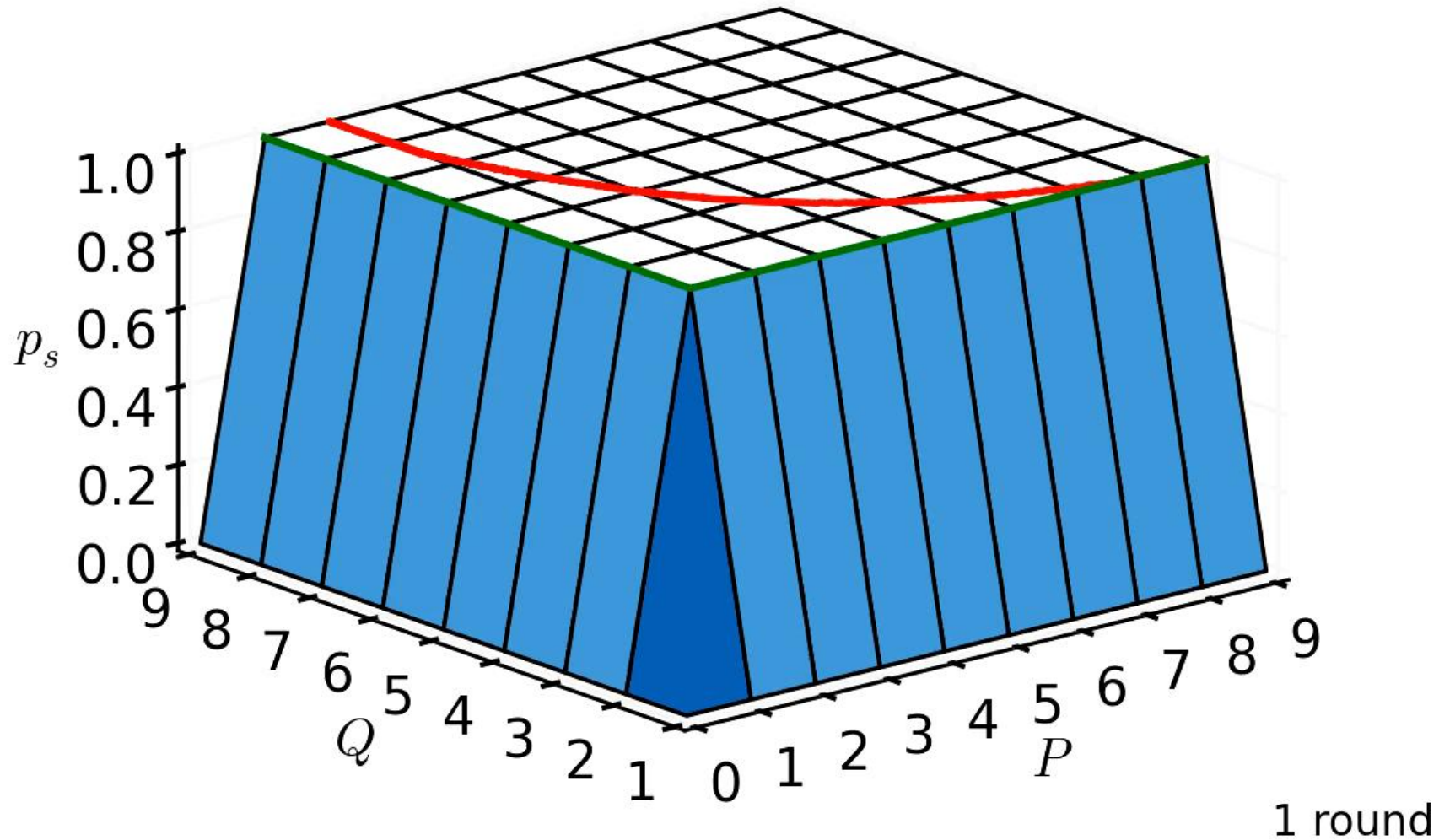


No costly methods to maintain a perfect knowledge of network state

Data Survival (large # of runs)



Ongoing Research



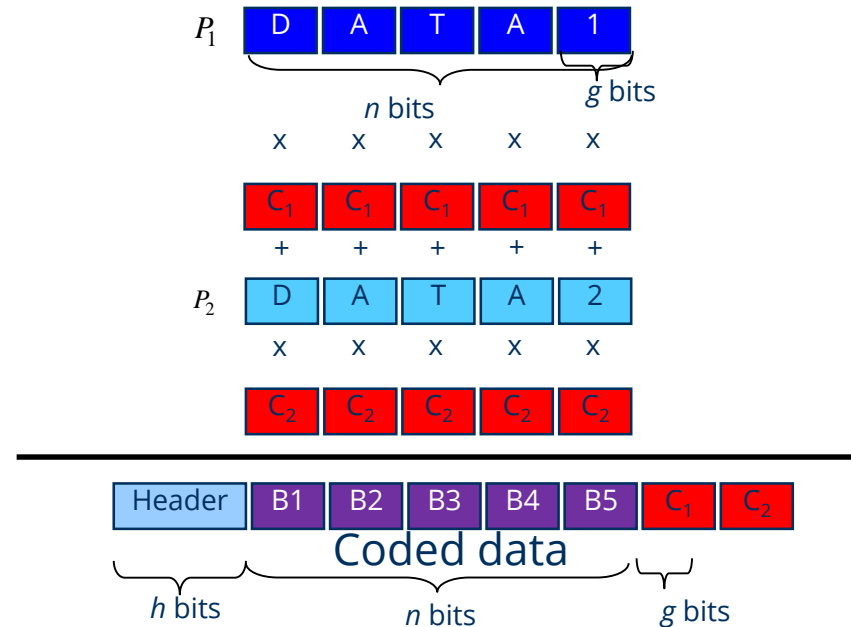
Updating Coded Stored Data

Generating a Coded Packet

Generating a Network Coded Packet
Operations over Finite Fields

$$CP_j = \sum_i C_i P_i$$

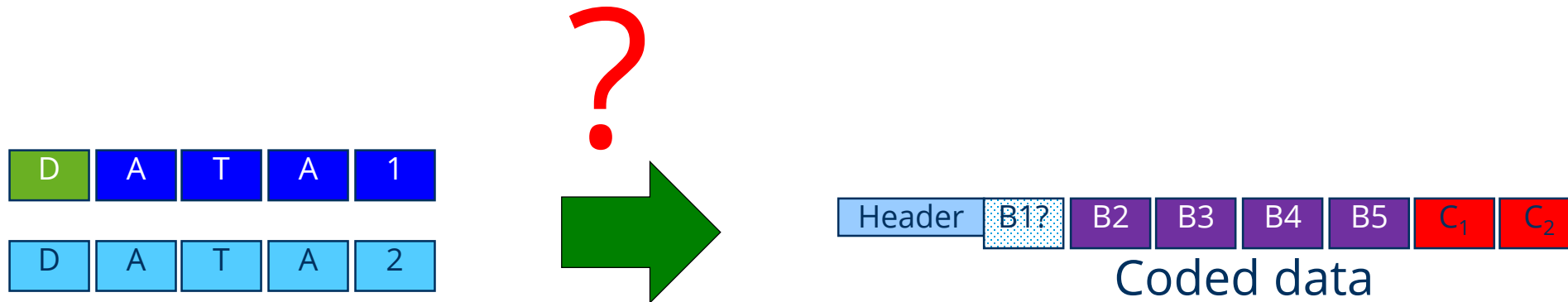
e.g. $g = 8$ bits, $q = 256$



Updating the Data

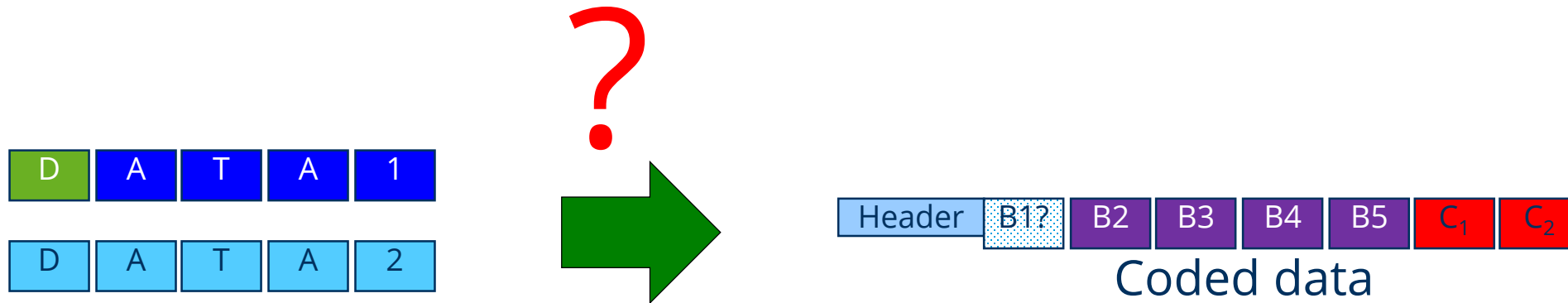
If one of the packets changed

- What do we need to transmit over the network?
- How do we update the data at the storage node?



Updating the Data

The storage node must compute the new value. How?

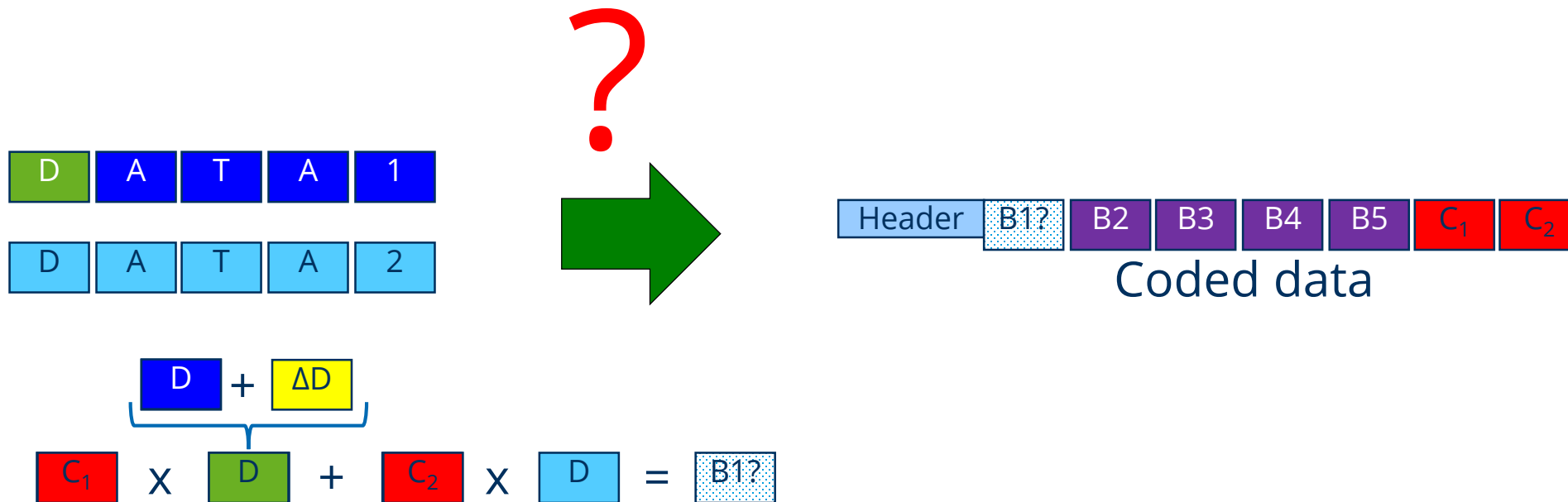


$$C_1 \times D + C_2 \times D = B1?$$

Updating the Data

The storage node must compute the new value. How?

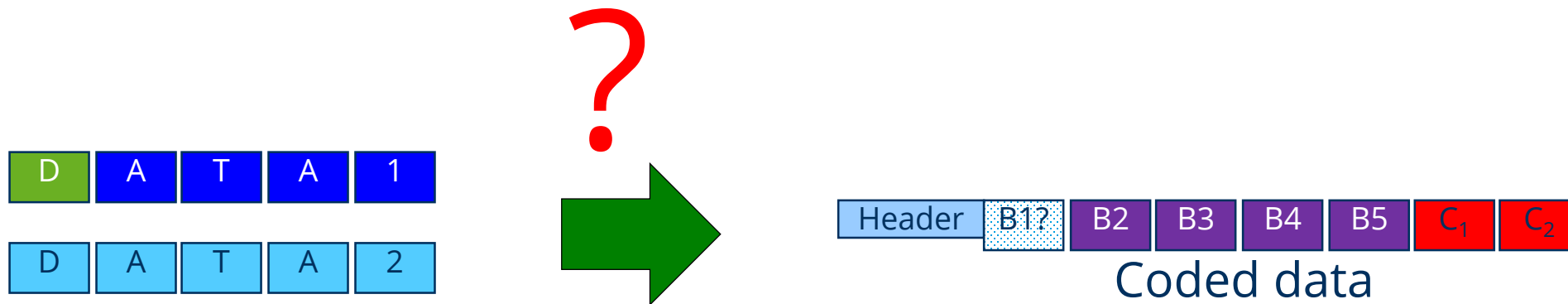
- The changed information is the old value plus a delta



Updating the Data

The storage node must compute the new value. How?

- The changed information is the old value plus a delta
- If we expand the expression, we get:

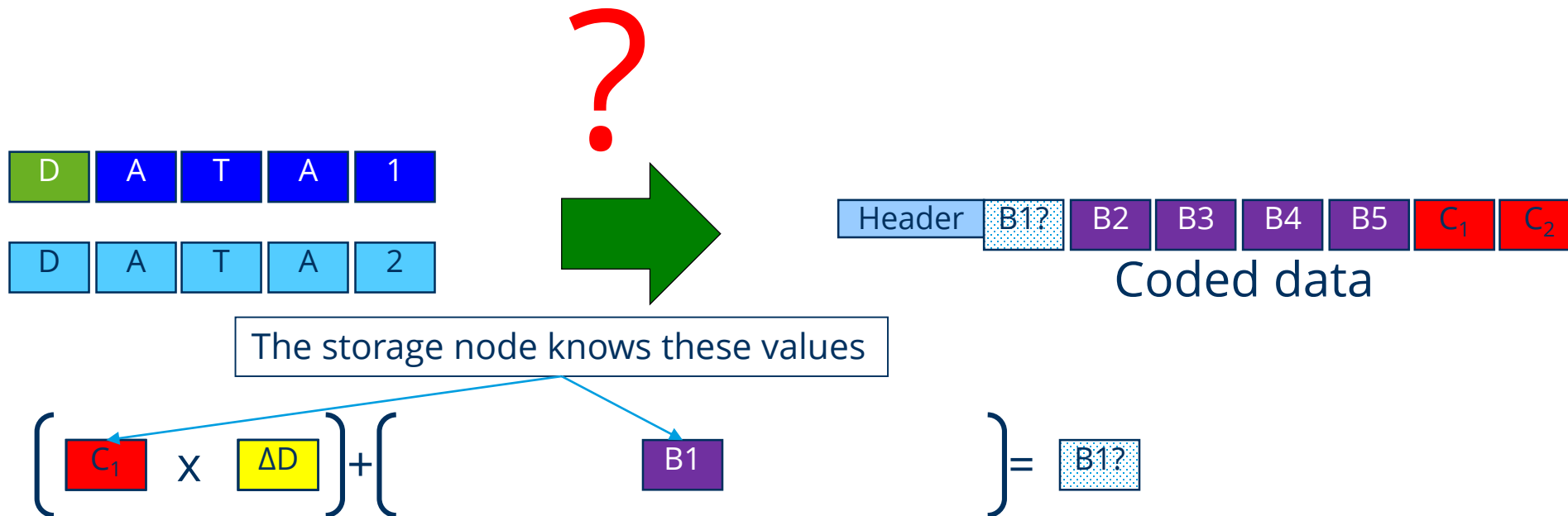


$$\left(\begin{array}{|c|} \hline C_1 \\ \hline \end{array} \times \begin{array}{|c|} \hline \Delta D \\ \hline \end{array} \right) + \left(\begin{array}{|c|} \hline C_1 \\ \hline \end{array} \times \begin{array}{|c|} \hline D \\ \hline \end{array} + \begin{array}{|c|} \hline C_2 \\ \hline \end{array} \times \begin{array}{|c|} \hline D \\ \hline \end{array} \right) = \begin{array}{|c|} \hline B1? \\ \hline \end{array}$$

Updating the Data

The storage node must compute the new value. How?

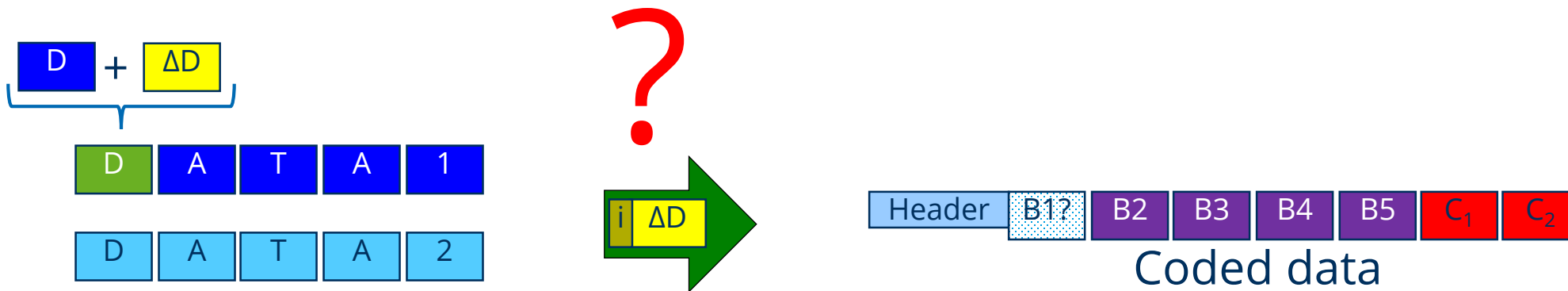
- The changed information is the old value plus a delta
- If we expand the expression, we get:
- The second part of the expression is equal to the old value



Updating the Data

The storage node must compute the new value. How?

- Therefore, we only need to broadcast over the network the deltas of the changes
- The storage nodes can compute the new values



$$\left[\begin{array}{|c|} \hline C_1 \\ \hline \end{array} \times \begin{array}{|c|} \hline \Delta D \\ \hline \end{array} \right] + \left[\begin{array}{|c|} \hline B_1 \\ \hline \end{array} \right] = \begin{array}{|c|} \hline B1? \\ \hline \end{array}$$

Coded data

On Network Traffic Considerations

Update Traffic:

- Replication:
 - The system must broadcast the new changed symbols. One byte to change one byte
- Reed-Solomon:
 - The system must encode again the information, and transmit the new packets
 - Many bytes transported to change one byte
- Network Coding:
 - The system must broadcast the new changed symbols. One byte to change one byte
 - Same as replication

On Network Traffic Considerations

	Storage Costs	Repair Costs	Update Costs
Replication	X	✓	✓
Reed-Solomon	✓	X	X
Network Coding	✓	Optimal ←→	✓