Regenerating Codes for Errers and Erasures in Distributed Storage

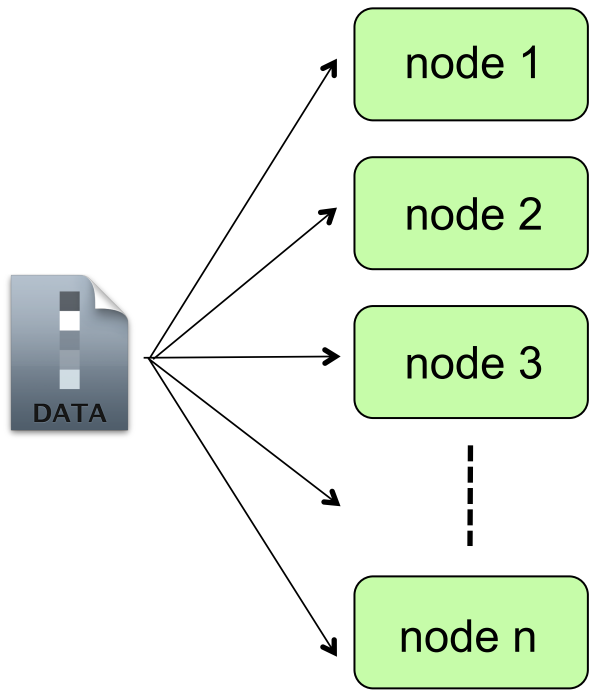
Nihar Shah

joint work with

K V Rashmi , Kannan Ramchandran , P Vijay Kumar

Regenerating Codes

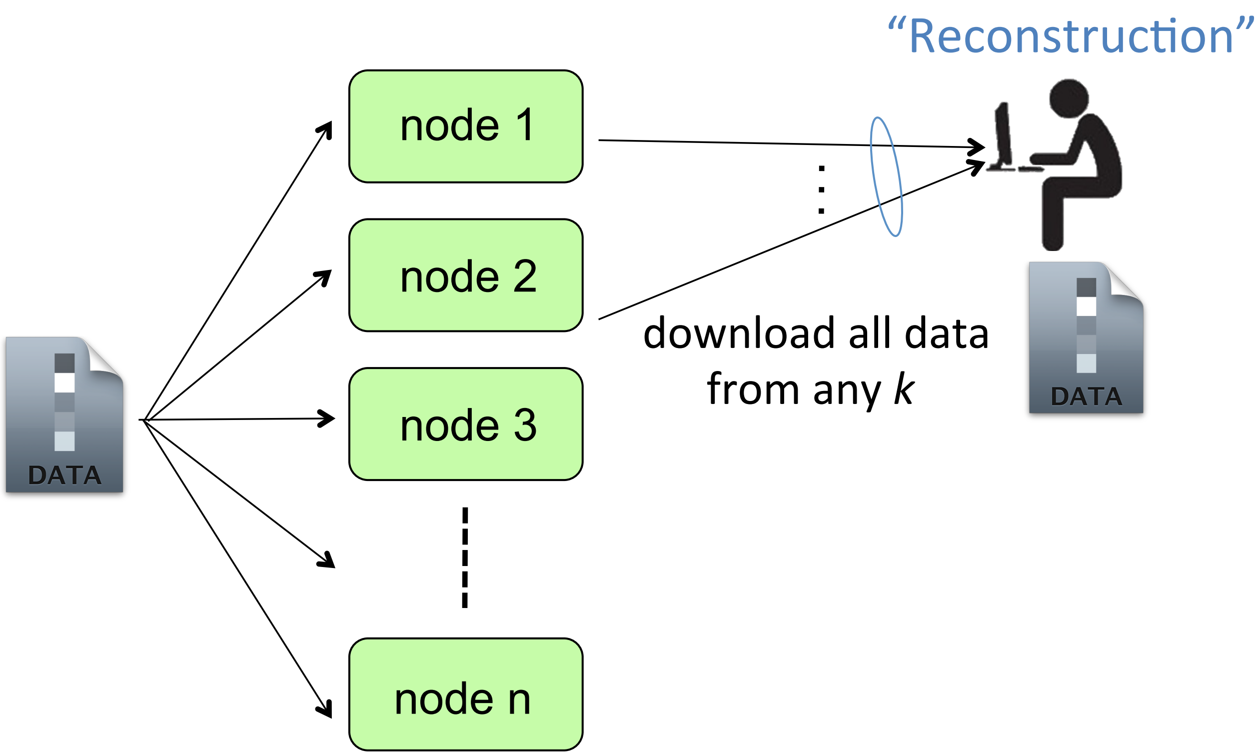
- Provide reliability
- Efficient repair



Introduced by Dimakis et al. (Infocom 07, IT-Transactions 10)

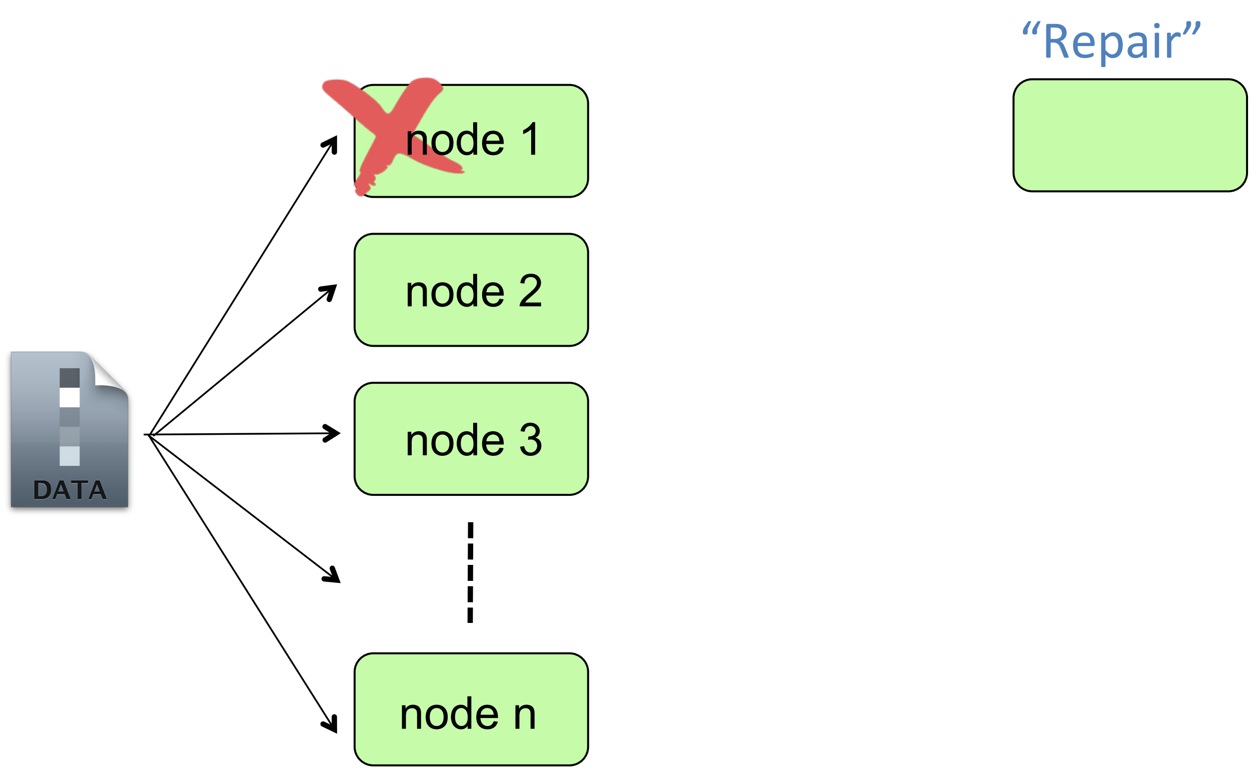
Regenerating Codes

- Provide reliability: recover data from any k nodes
- Efficient repair



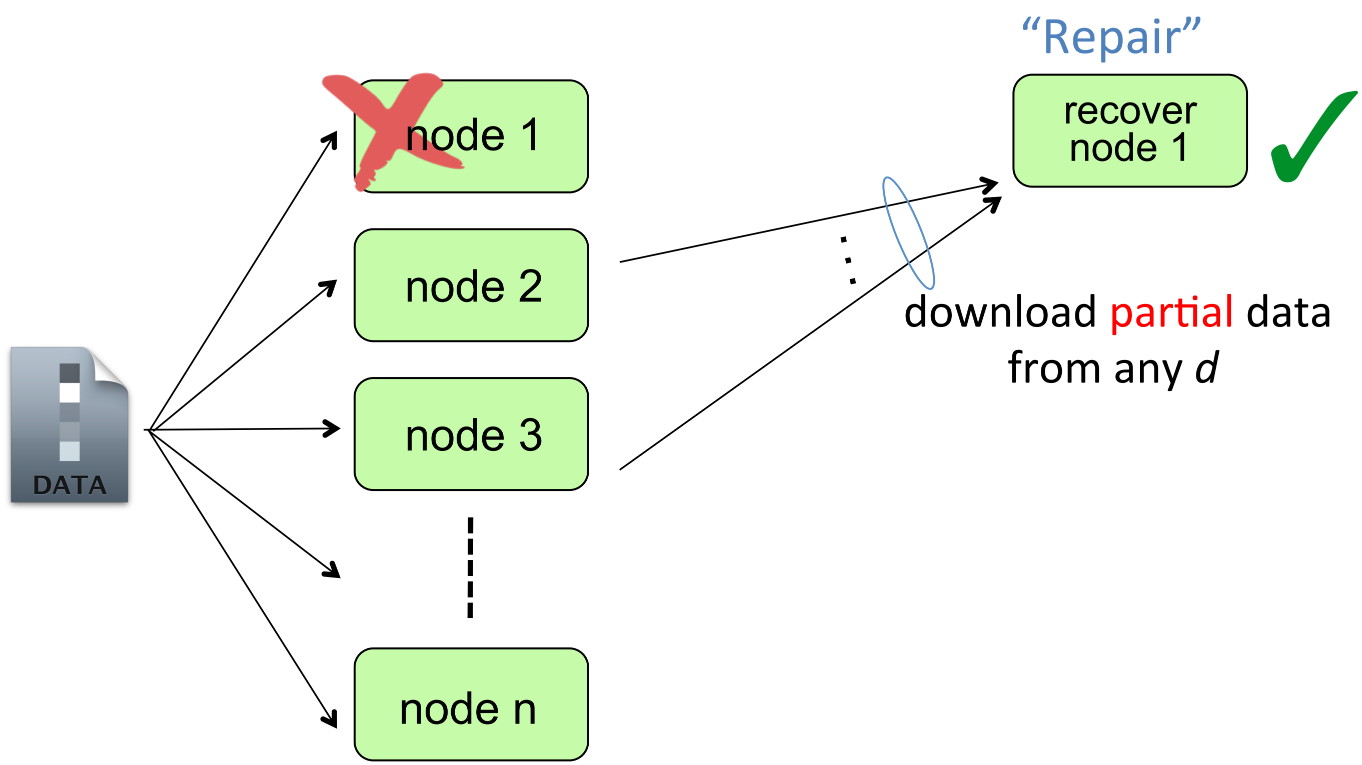
Regenerating Codes

- Provide reliability: recover data from any k nodes
- Efficient repair: small network bandwidth



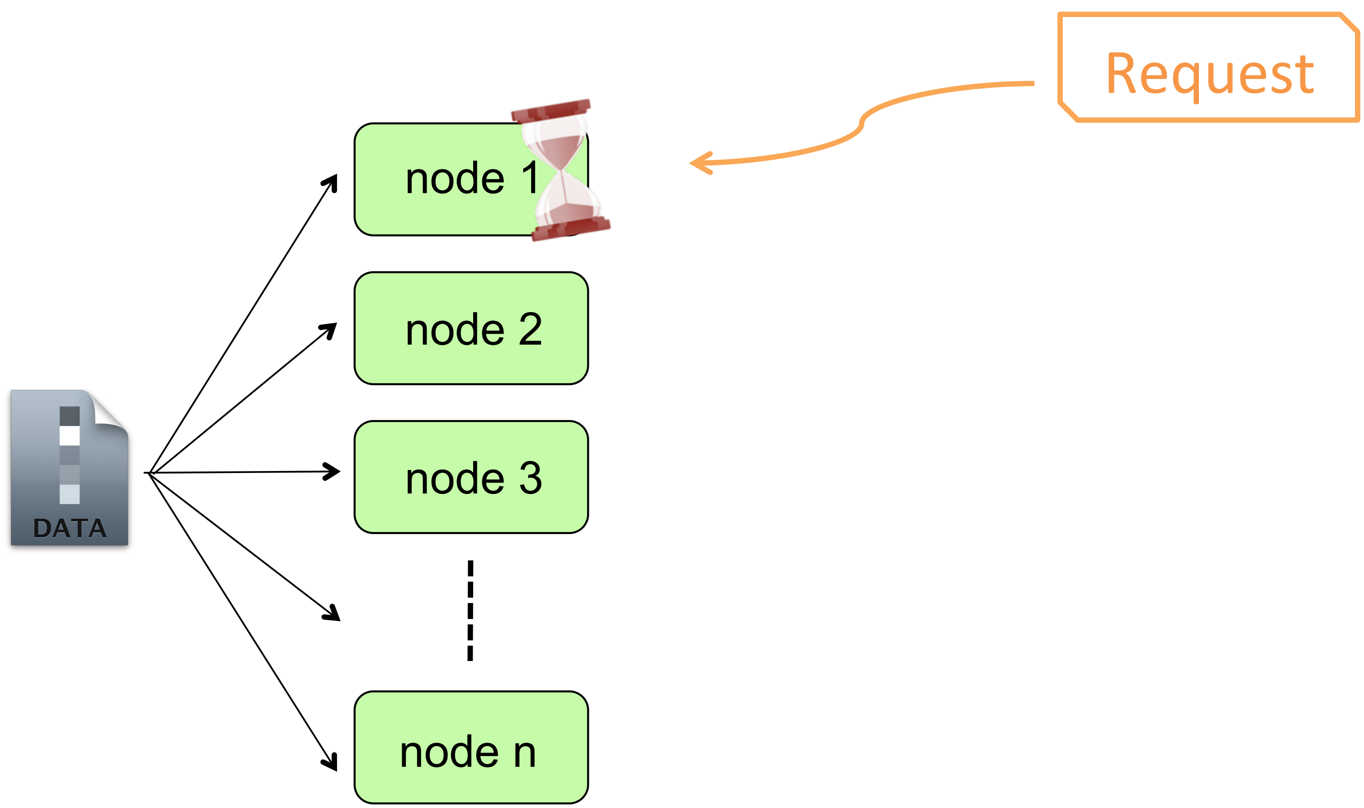
Regenerating Codes

- Provide reliability: recover data from any k nodes
- Efficient repair: small network bandwidth



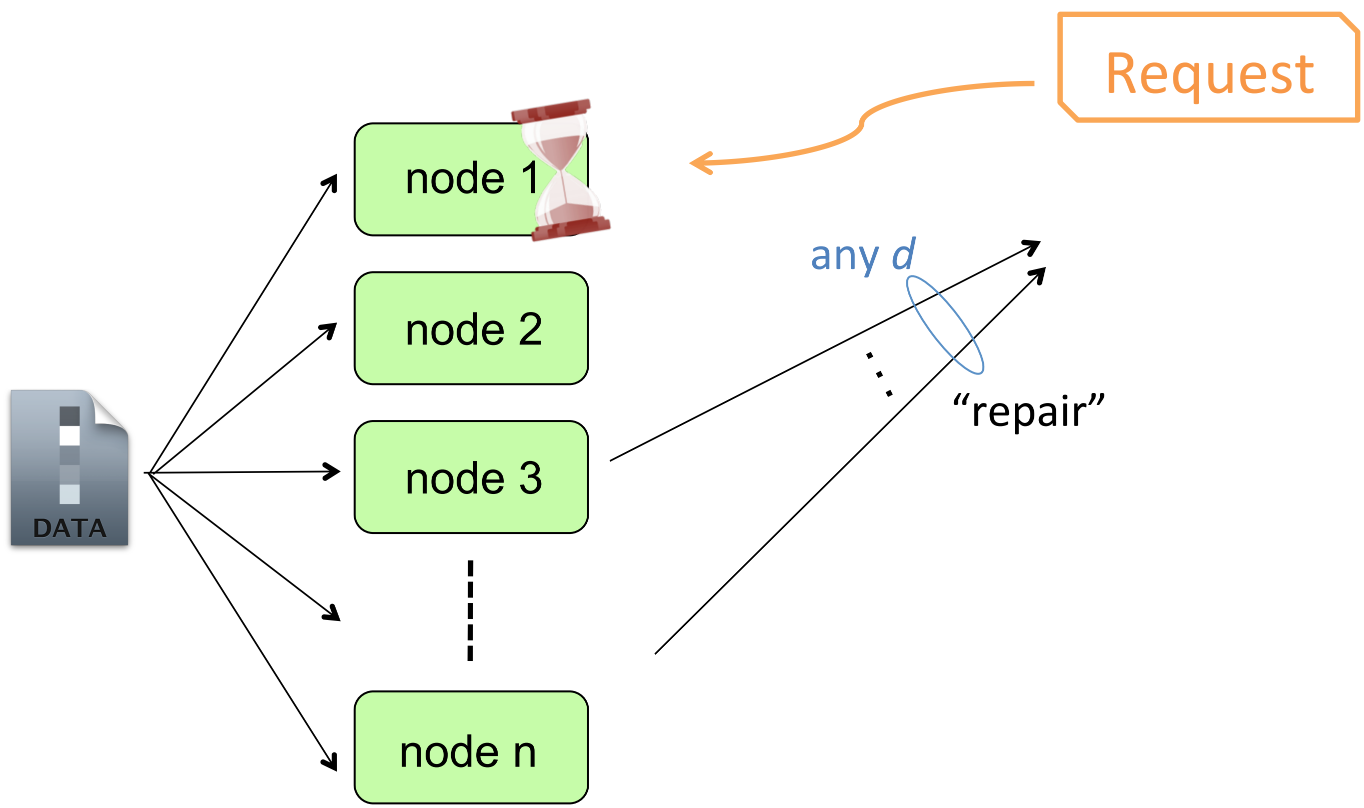
Regenerating Codes

- Provide reliability: recover data from any k nodes
- Efficient repair: small network bandwidth
  - (Fast) degraded reads



Regenerating Codes

- Provide reliability: recover data from any k nodes
- Efficient repair: small network bandwidth
  - (Fast) degraded reads



Explicit Regenerating Code Constructions

Minimum Bandwidth Point (MBR):

Rashmi et al.'09, Rashmi et al.'11

Minimum Storage Point (MSR):

Rashmi et al.'09, Shah et al. '09,
Suh et al.'10, Rashmi et al.'11, Cadambe et al.'11,
Tamo et al.'11, Wang et al.'11, Papailiopoulos et al.'11

….

Cooperative repair, adaptive repair, flexible repair…

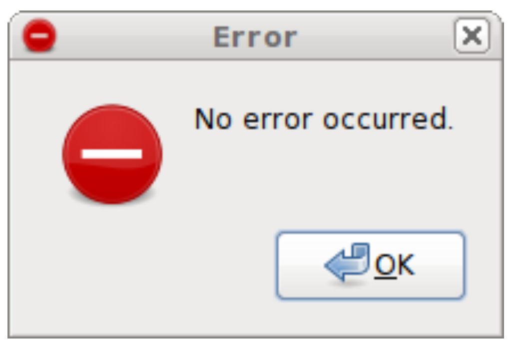
Explicit Regenerating Code Constructions

Minimum Bandwidth Point (MBR):
Rashmi et al.'09, Rashmi et al.'11

Minimum Storage Point (MSR):
Rashmi et al.'09, Shah et al. '09,
Suh et al.'10, Rashmi et al.'11, Cadambe et al.'11,
Tamo et al.'11, Wang et al.'11, Papailiopoulos et al.'11

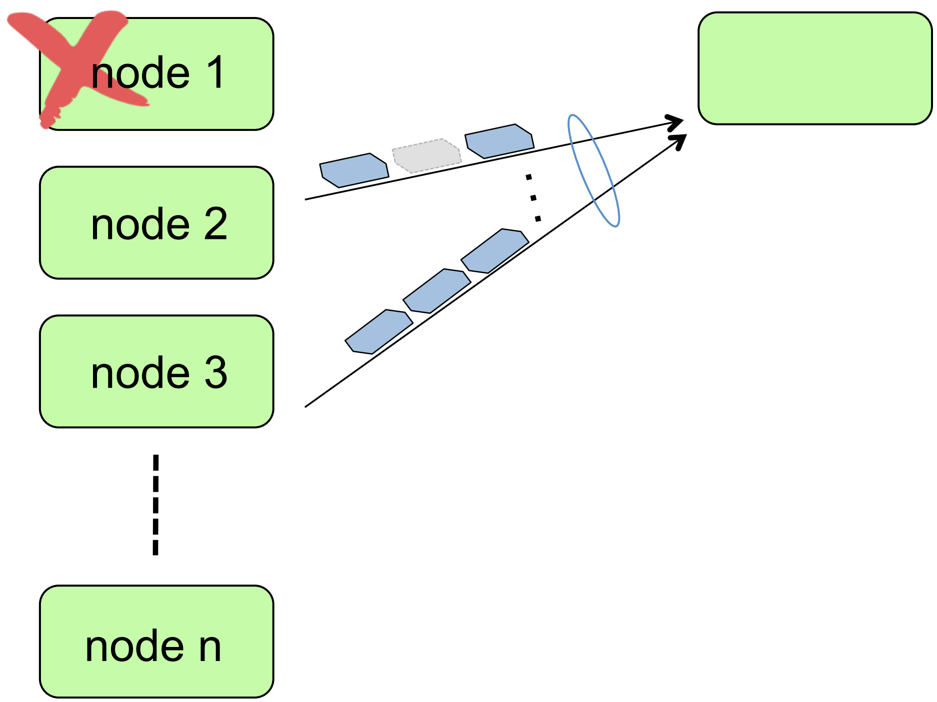
Cooperative repair, adaptive repair, flexible repair…

Assume an error/erasure-free setting

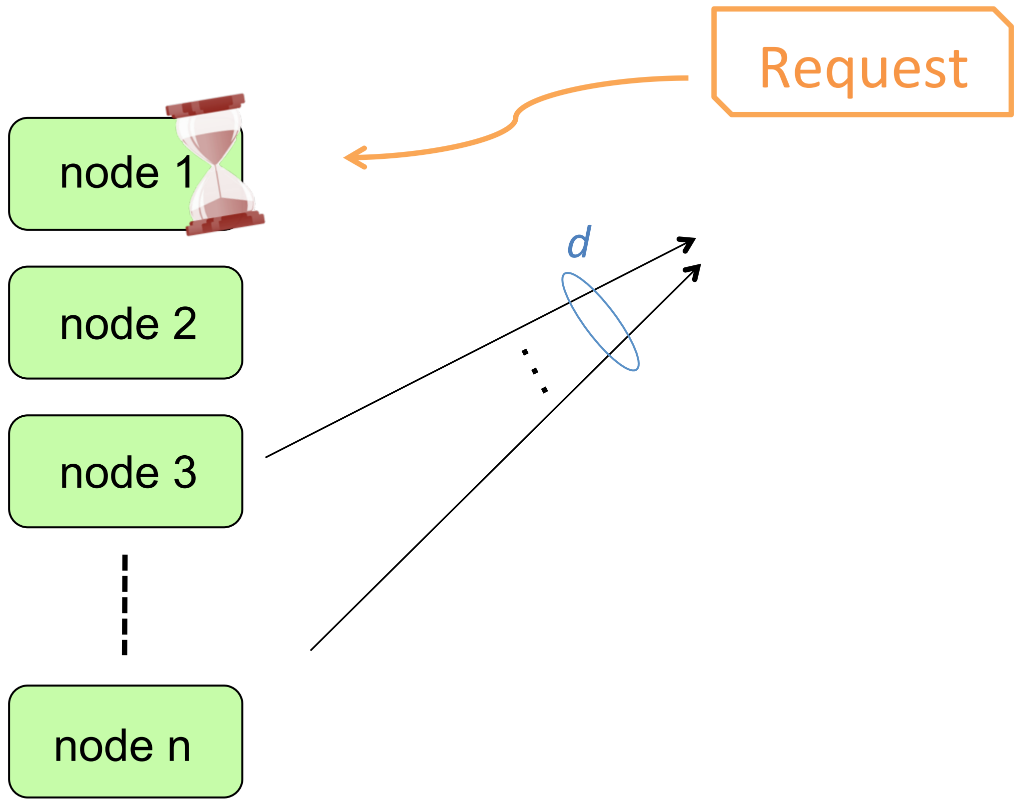


In this talk: errors and erasures…

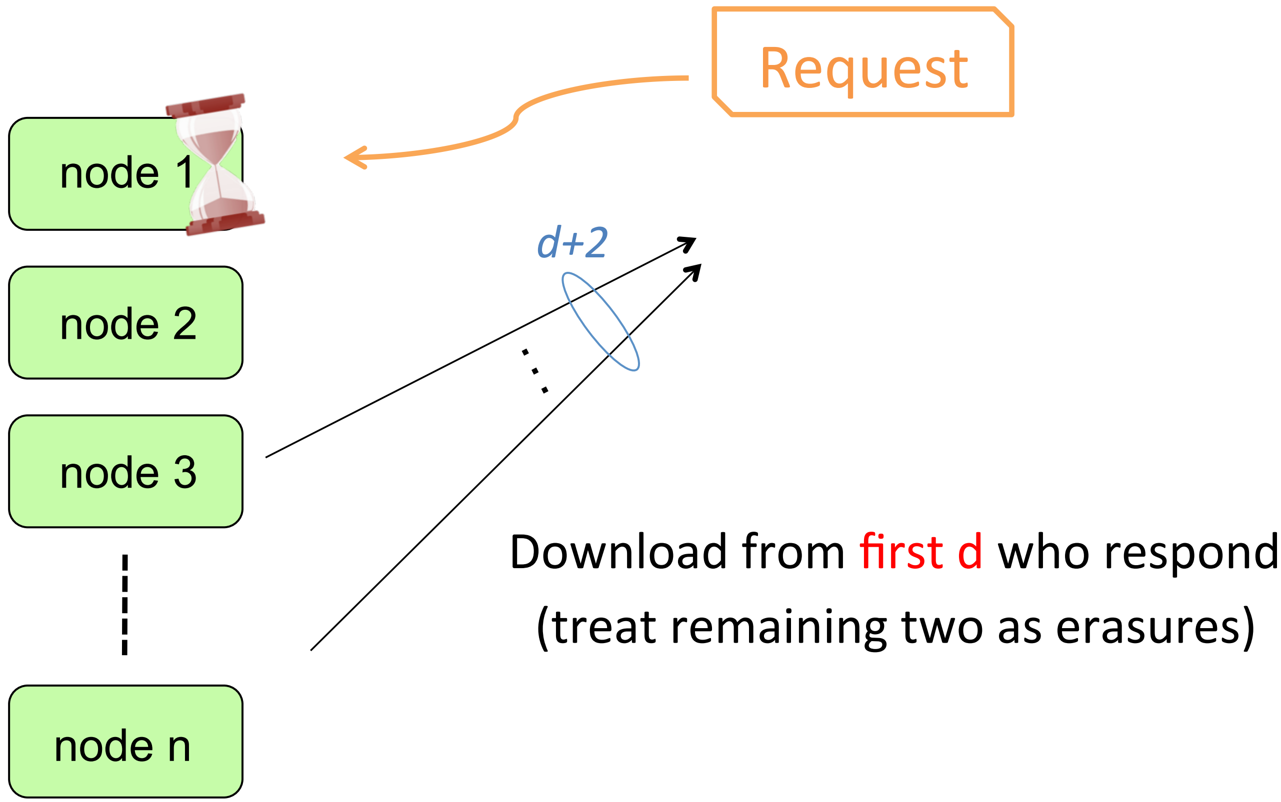
Motivation I: FEC for Network Errors



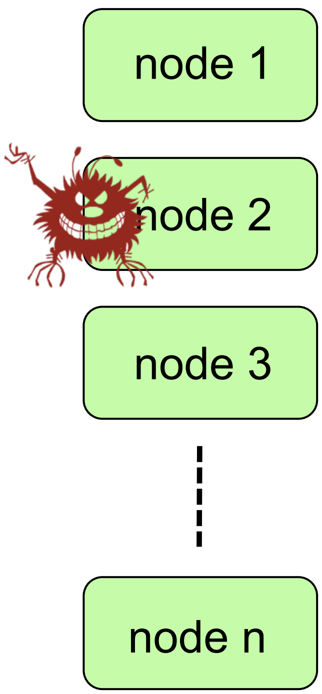
Motivation II: Improve Latency



Motivation II: Improve Latency

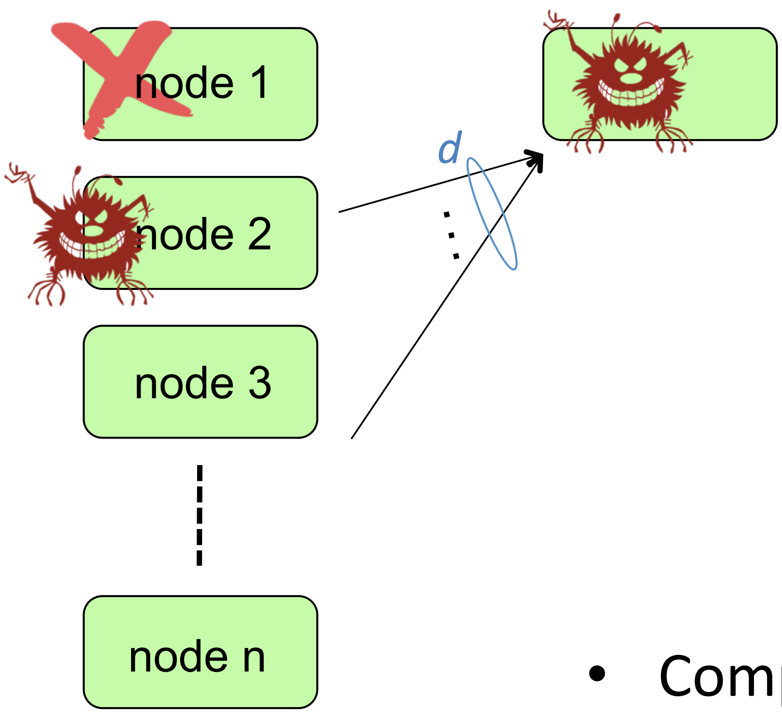


Motivation III: Security



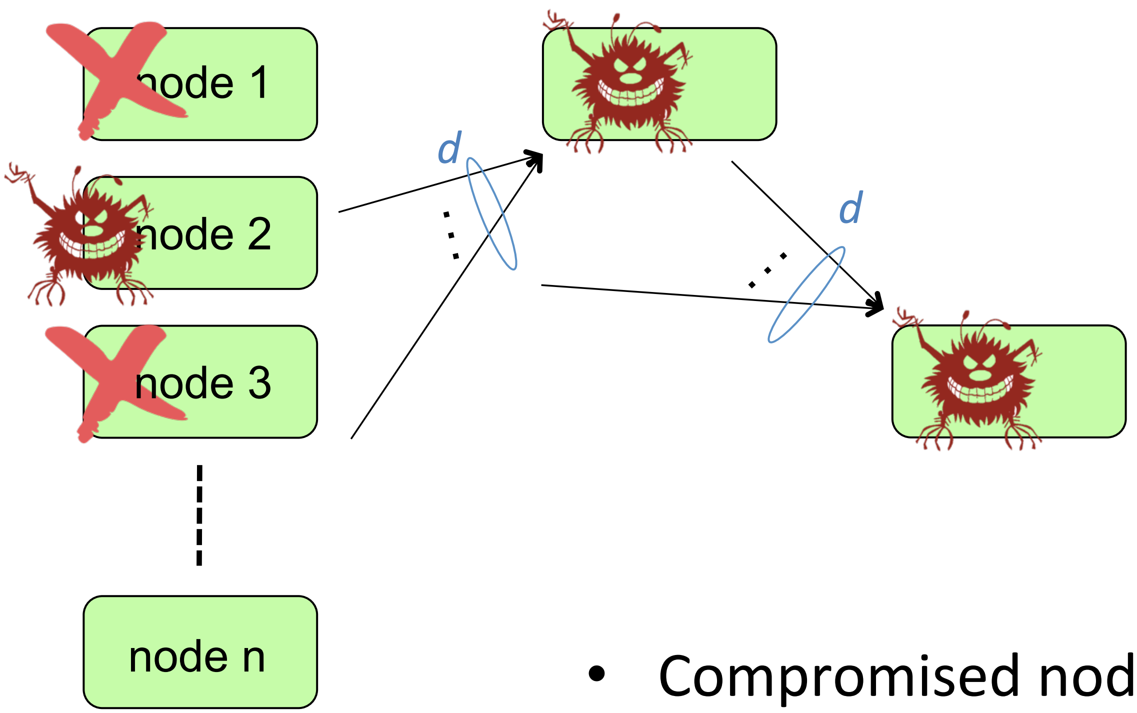
- Compromised nodes transmit erroneous data

Motivation III: Security



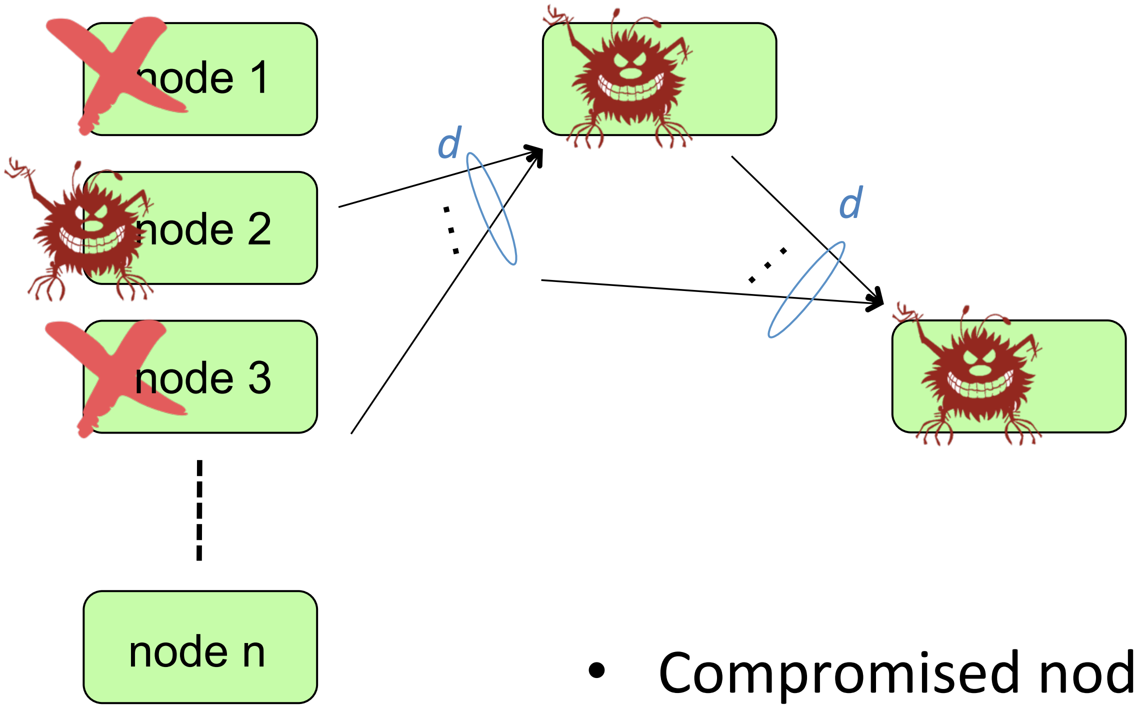
- Compromised nodes transmit erroneous data
- Errors may propagate during repair operations

Motivation III: Security



- Compromised nodes transmit erroneous data
- Errors may propagate during repair operations

Motivation III: Security



- Compromised nodes transmit erroneous data
- Errors may propagate during repair operations
- Outer Bounds: Pawar et al. '11

Motivation IV: Theoretical Interest

- Establish capacity of such systems

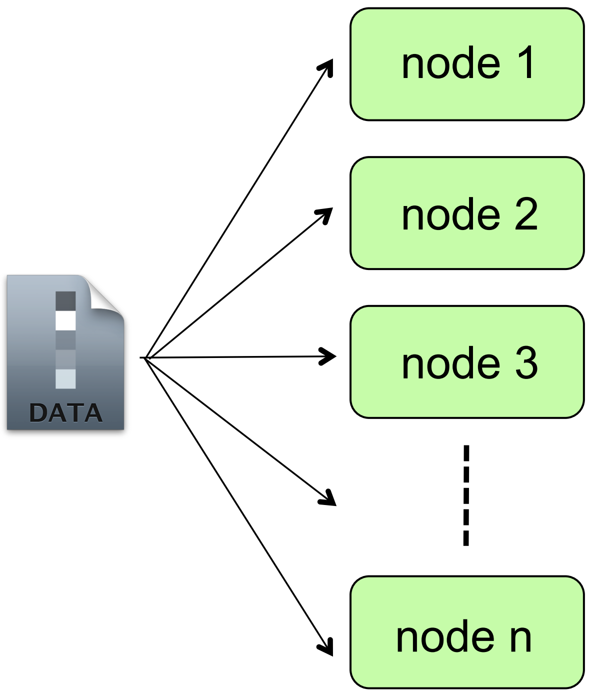
Code Constructions

Explicit codes performing error and erasure correction for

- Minimum Bandwidth (MBR): all parameters
- Minimum Storage (MSR): all $[n, k, d \geq 2k-2]$

These codes achieve (and establish) capacity.

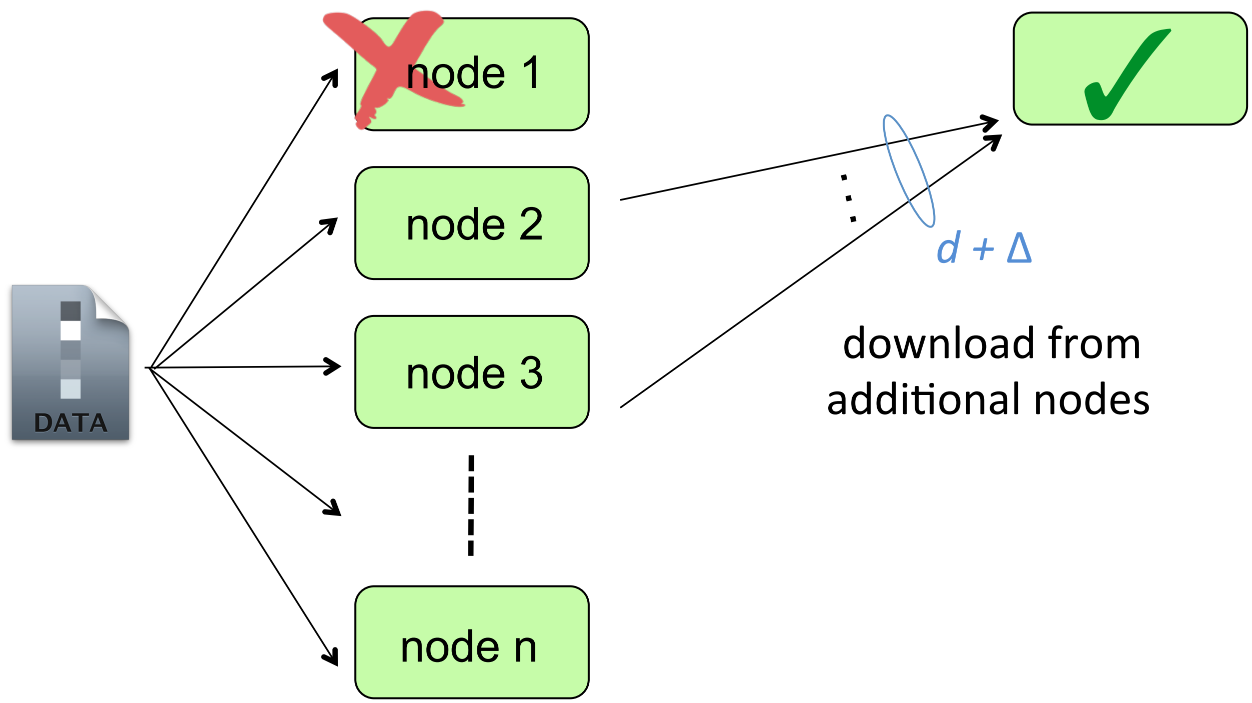
Recipe



Encode using (error-free)
Product-Matrix code[1]

[1] Rashmi, Shah, Kumar, IEEE Transactions on Information Theory, 2011

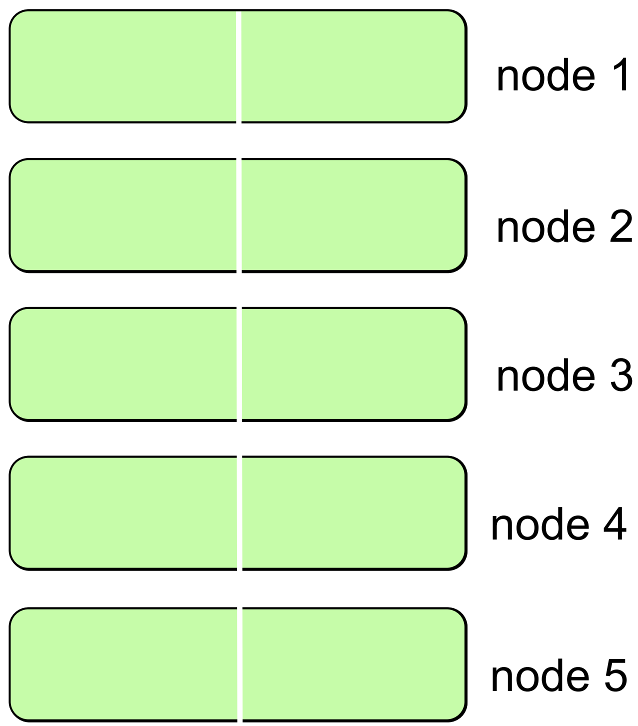
Recipe



[1] Rashmi, Shah, Kumar, IEEE Transactions on Information Theory, 2011

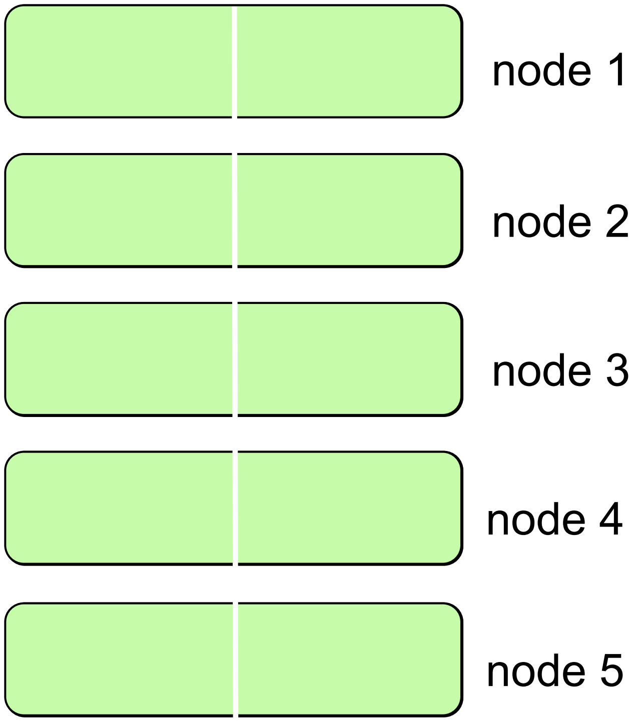
Toy Example: Minimum Bandwidth Code

- Tolerate any 3 node failures



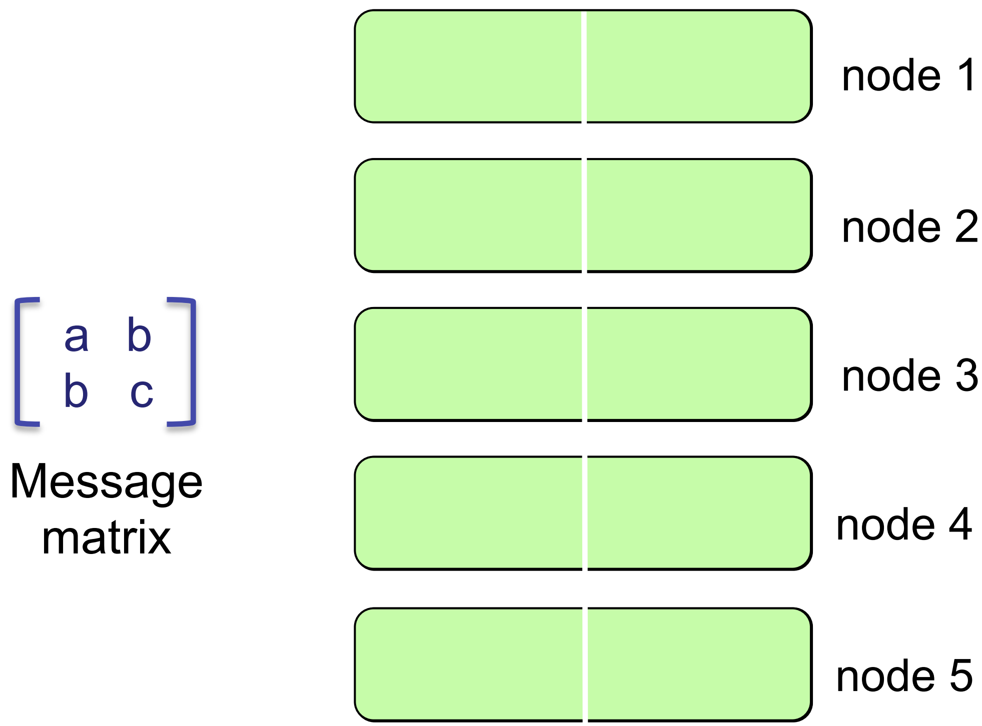
Toy Example: Minimum Bandwidth Code

- Data = {a, b, c}



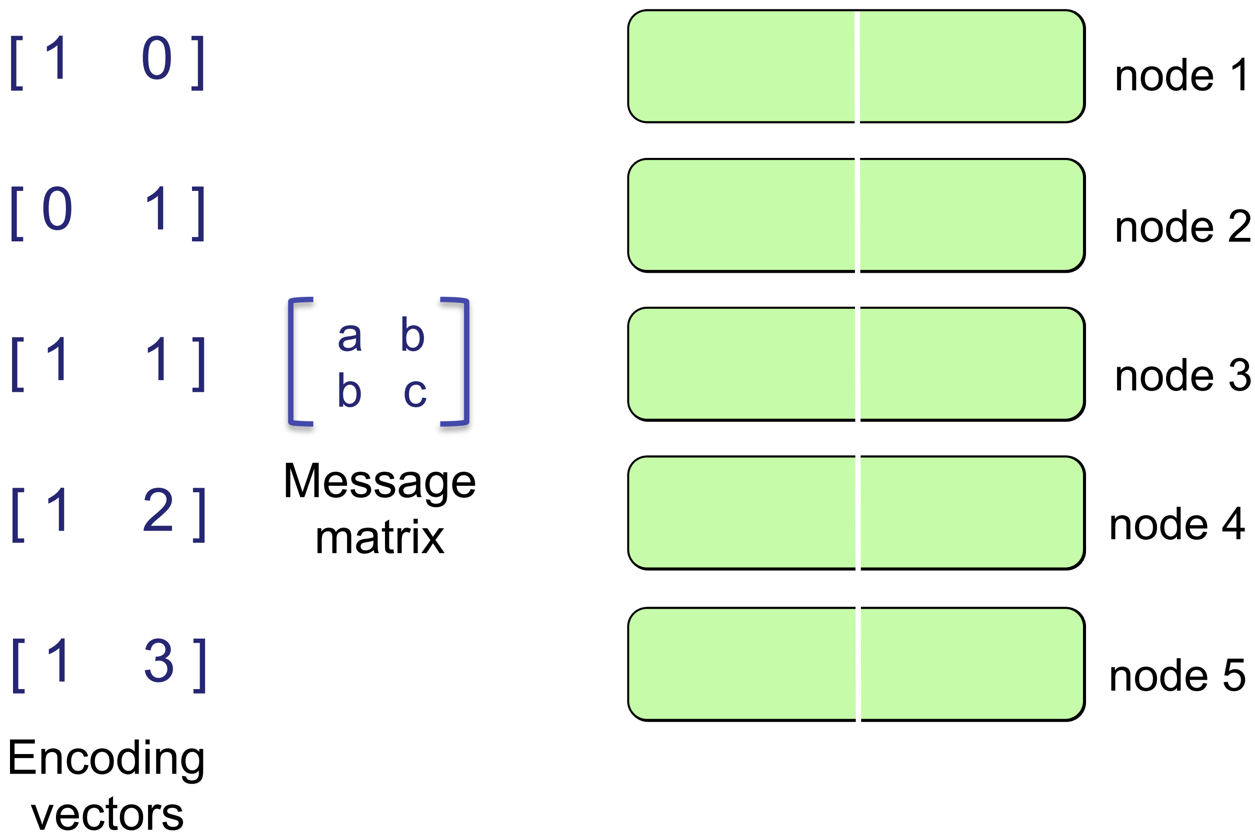
Toy Example: Minimum Bandwidth Code

- Data = {a, b, c}



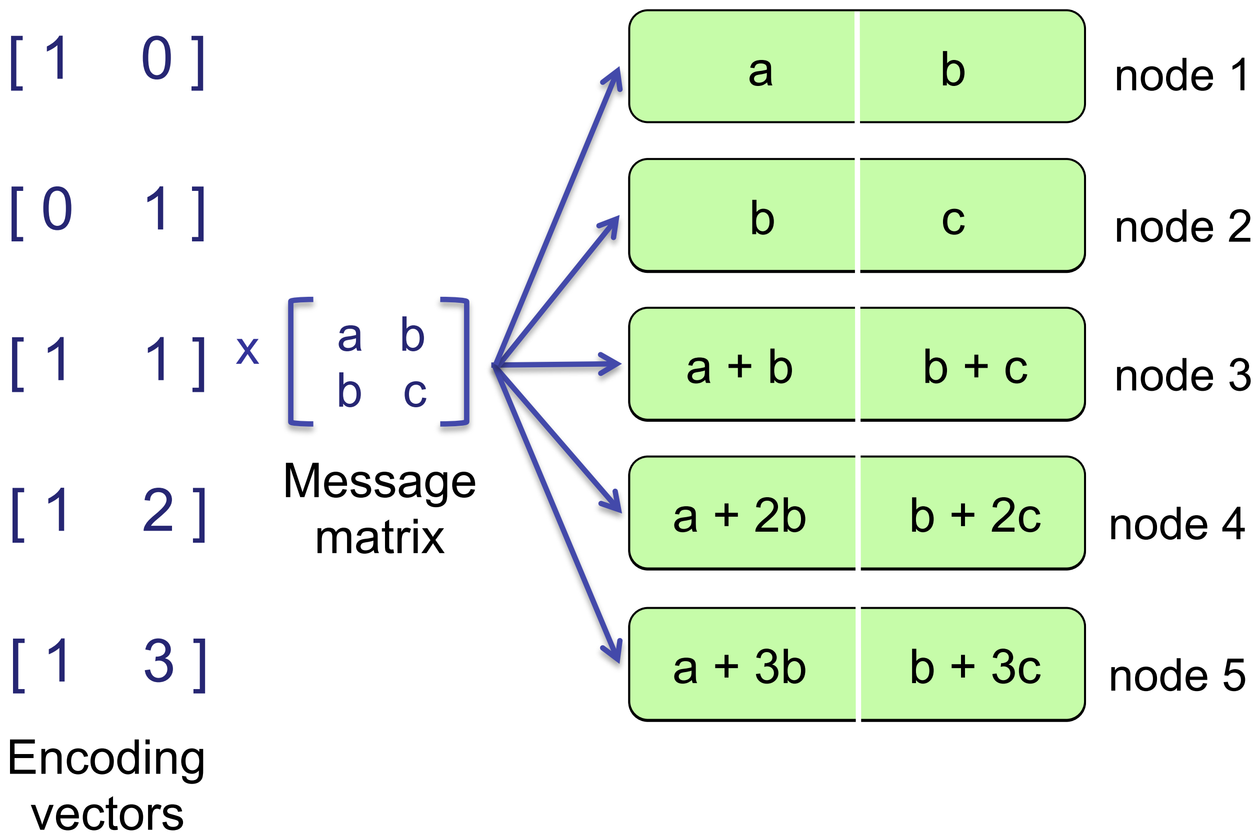
Toy Example: Minimum Bandwidth Code

- Data = {a, b, c}



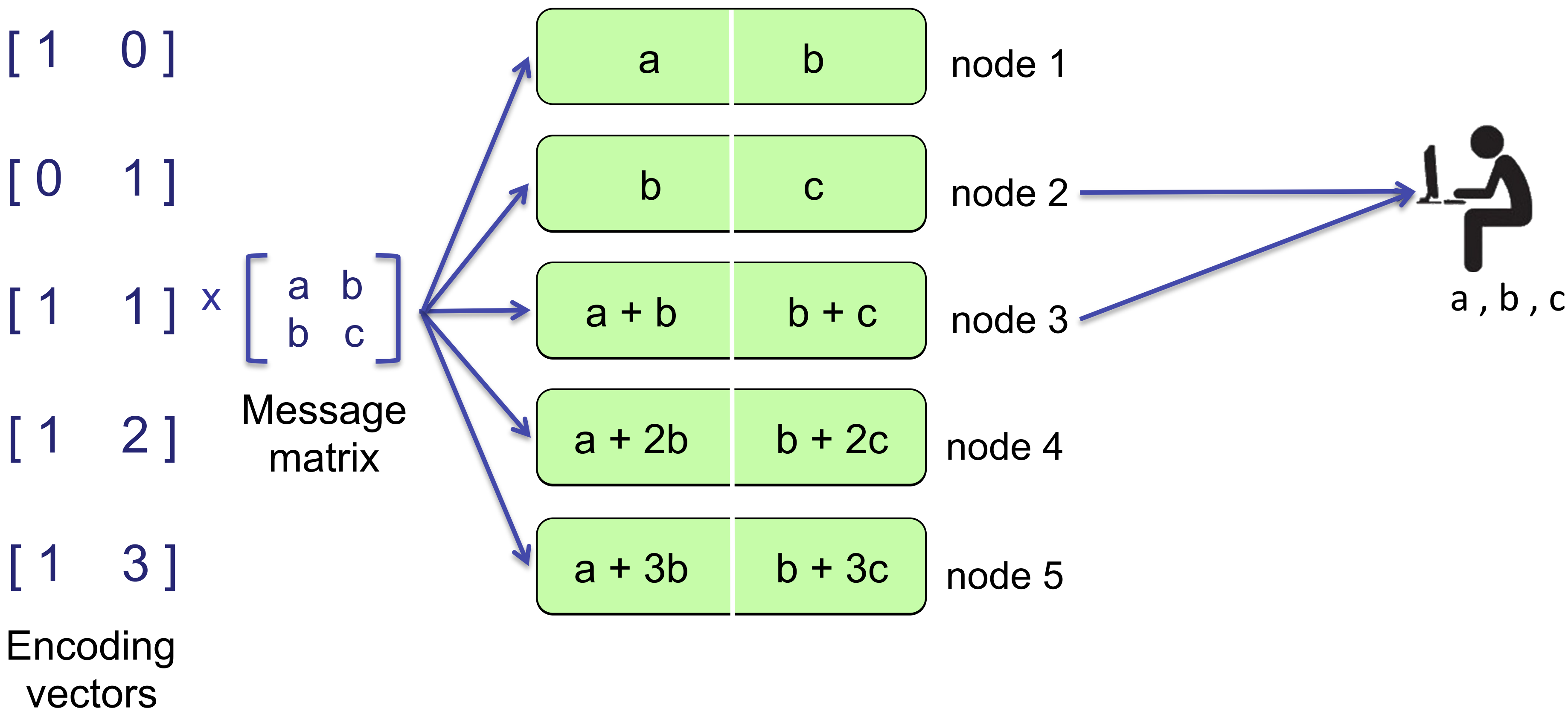
Toy Example: Minimum Bandwidth Code

- Data = {a, b, c}



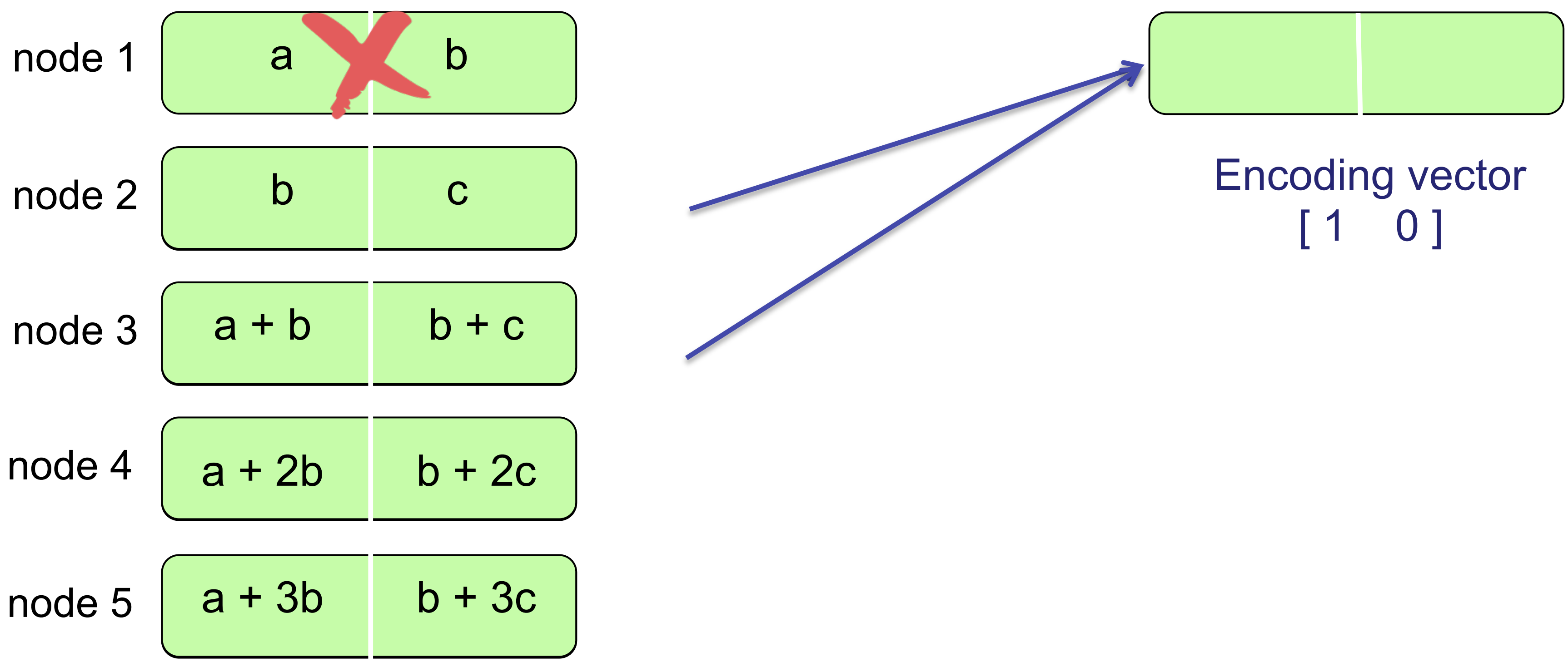
Toy Example: Minimum Bandwidth Code

- Can recover data from any 2 nodes (in absence of errors/erasures)



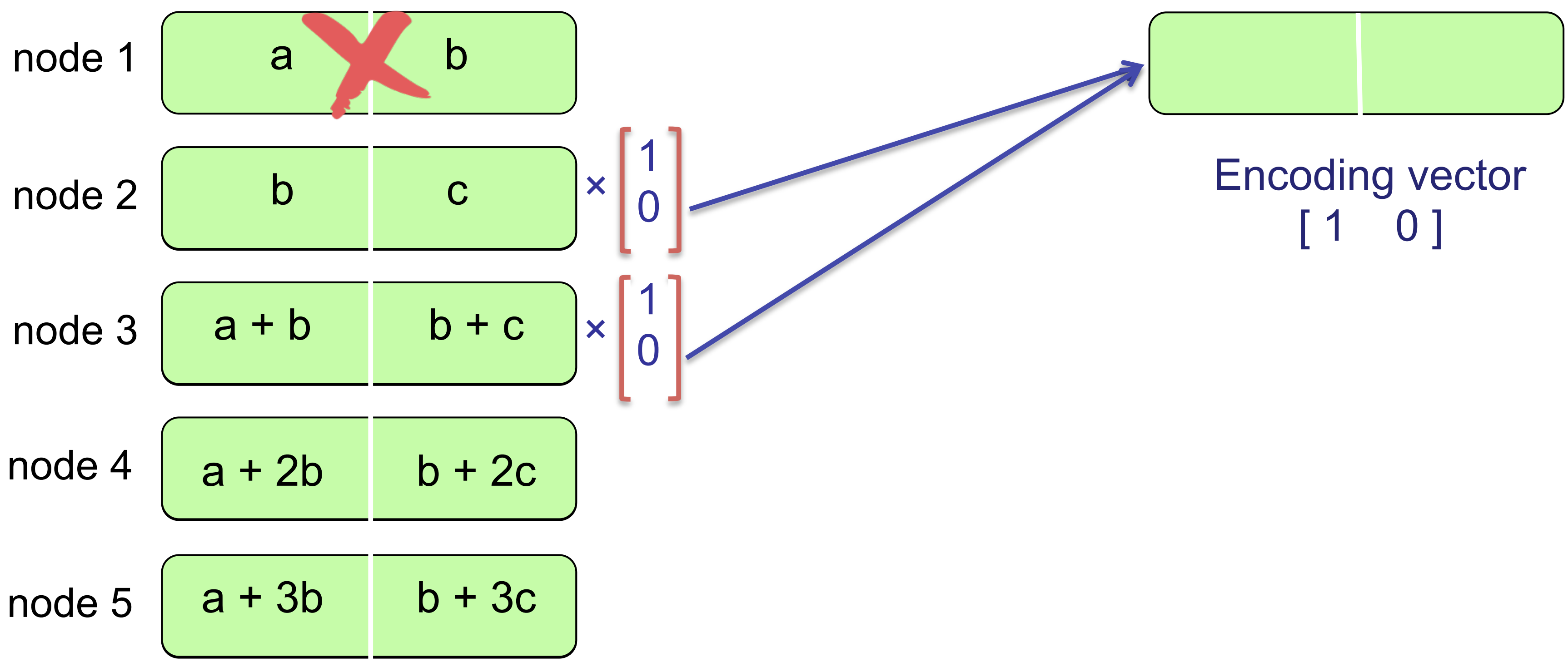
Toy Example: Minimum Bandwidth Code

- Optimal Repair (in absence of errors/erasures)



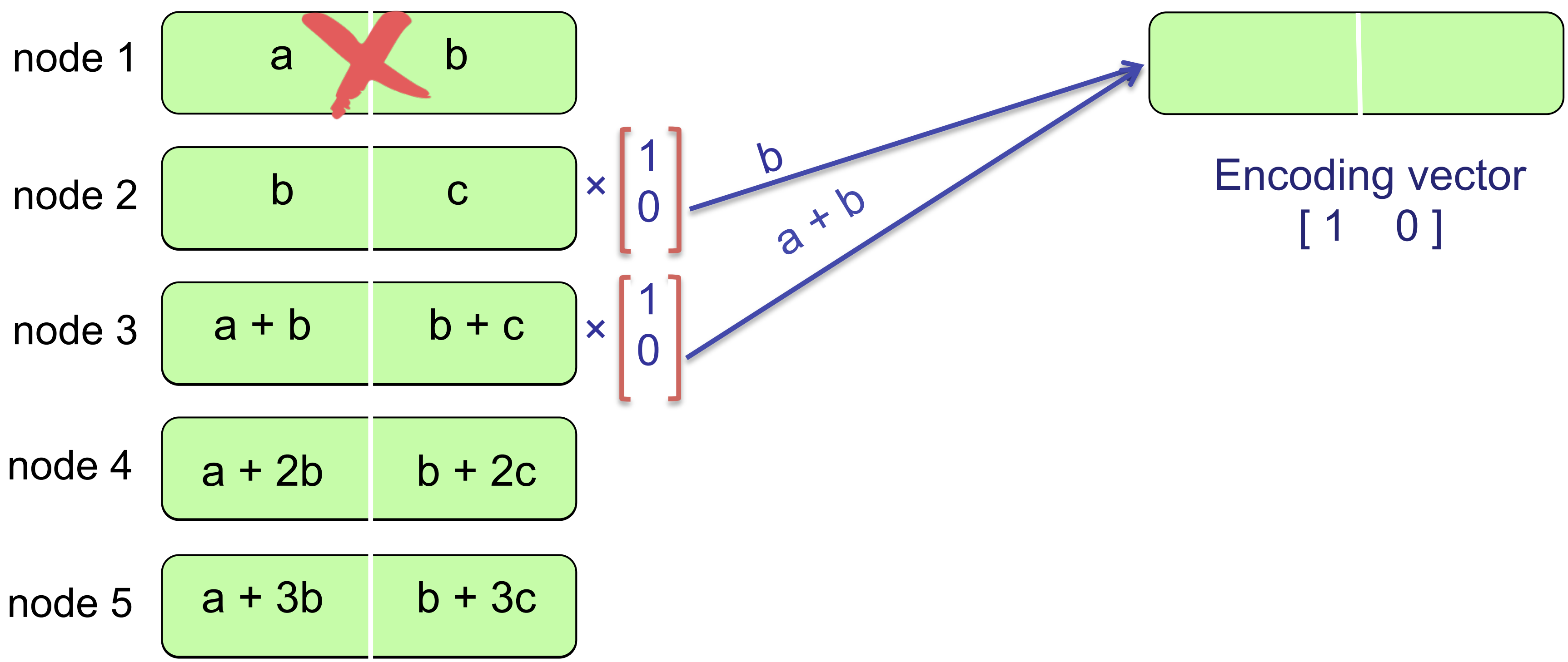
Toy Example: Minimum Bandwidth Code

- Optimal Repair (in absence of errors/erasures)



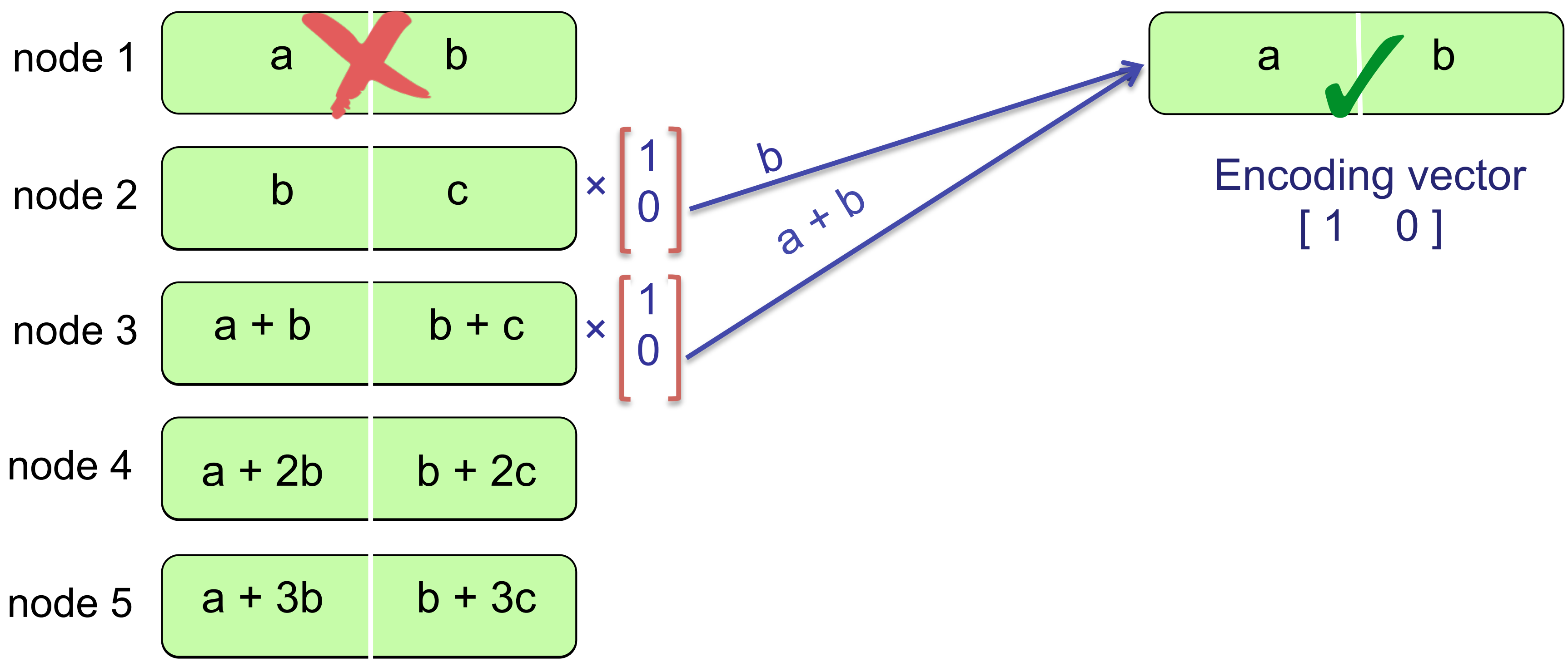
Toy Example: Minimum Bandwidth Code

- Optimal Repair (in absence of errors/erasures)



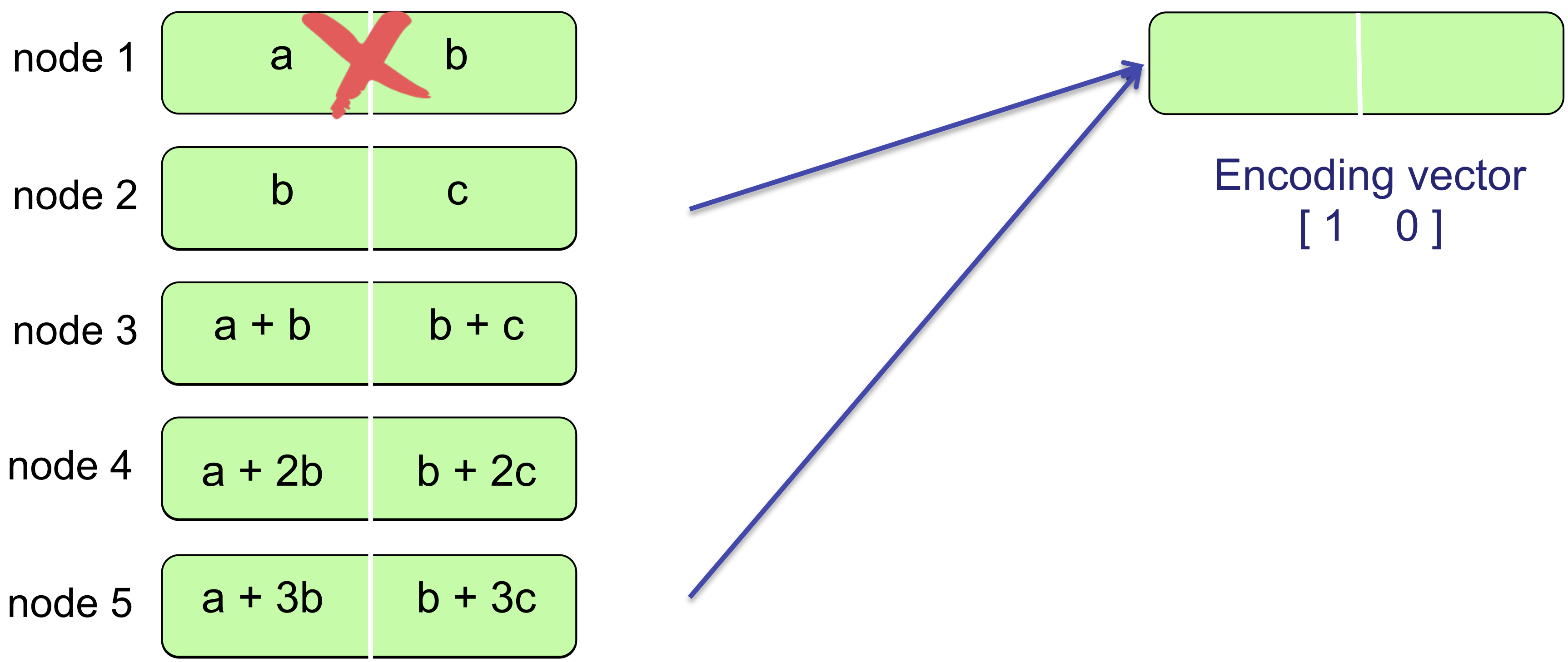
Toy Example: Minimum Bandwidth Code

- Optimal Repair (in absence of errors/erasures)



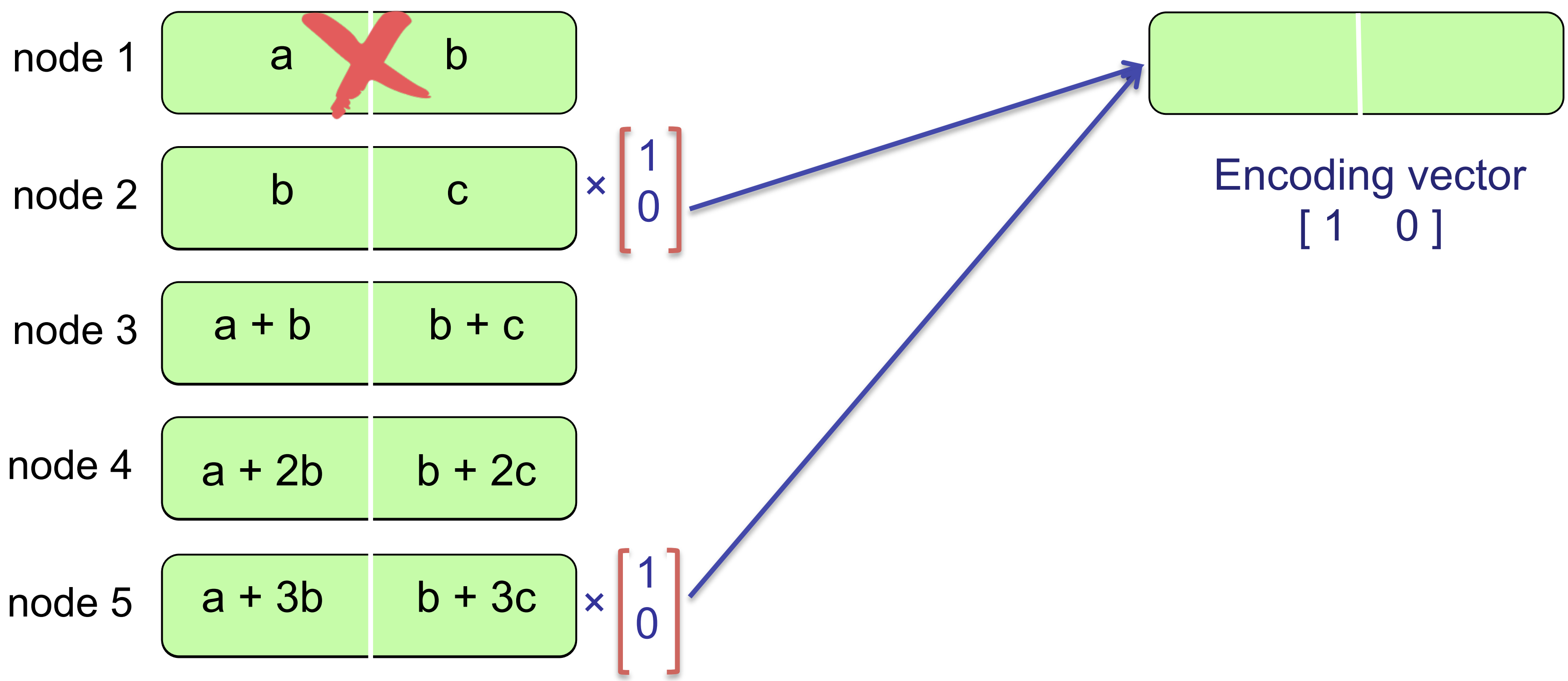
Toy Example: Minimum Bandwidth Code

- Optimal Repair (in absence of errors/erasures)



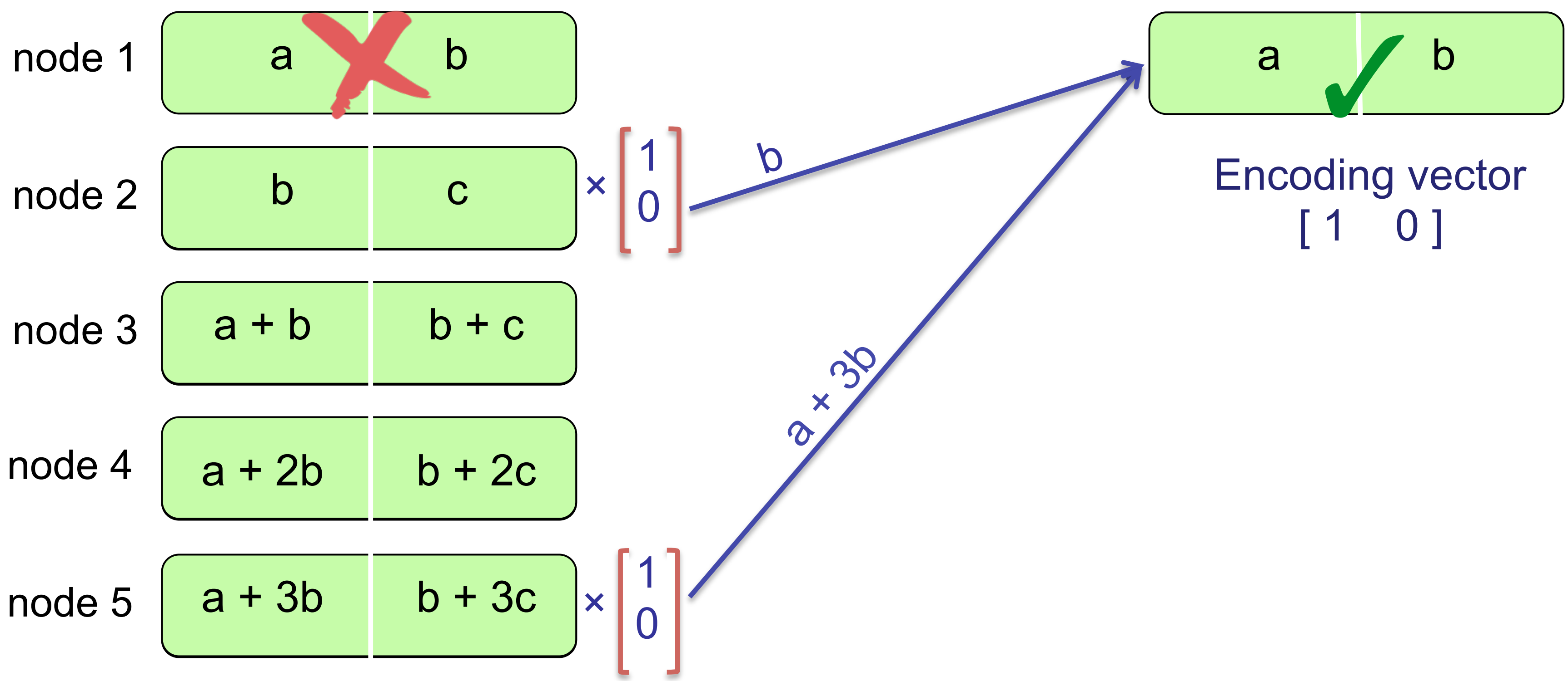
Toy Example: Minimum Bandwidth Code

- Optimal Repair (in absence of errors/erasures)



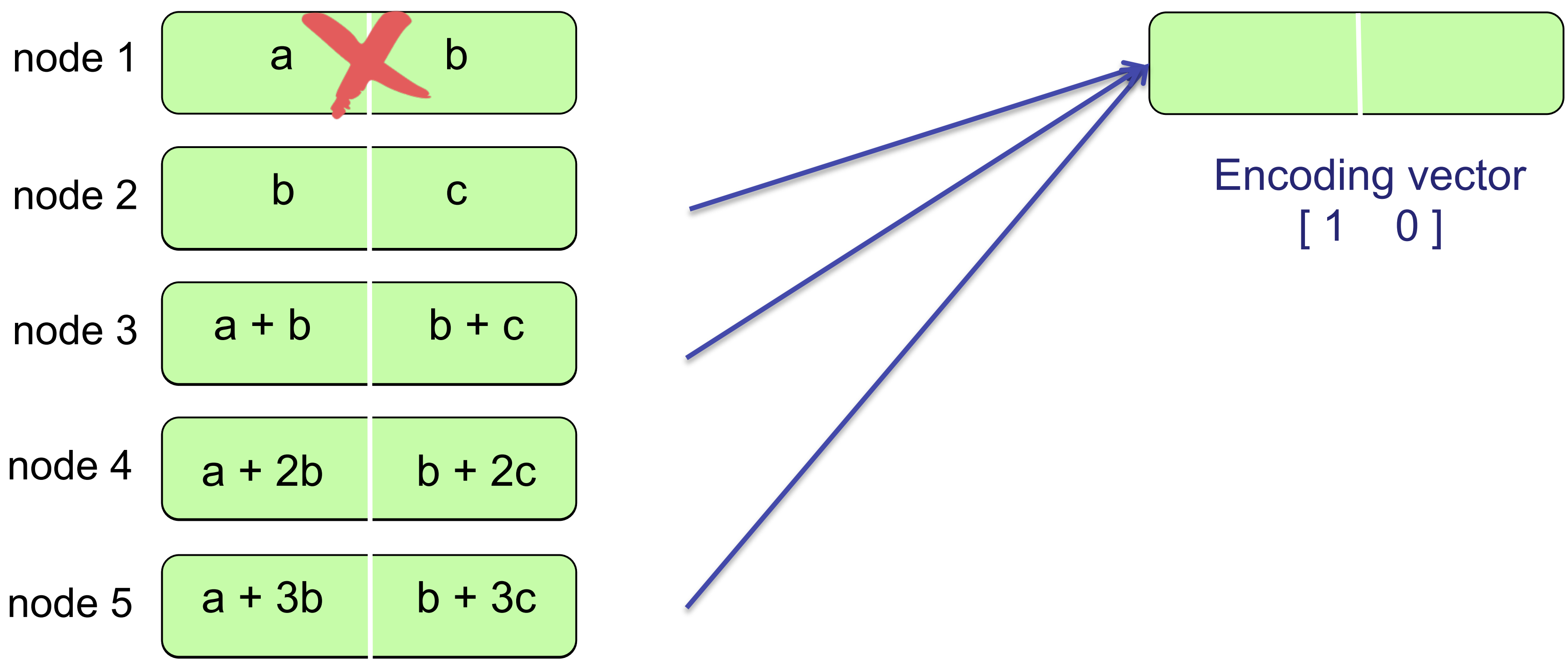
Toy Example: Minimum Bandwidth Code

- Optimal Repair (in absence of errors/erasures)

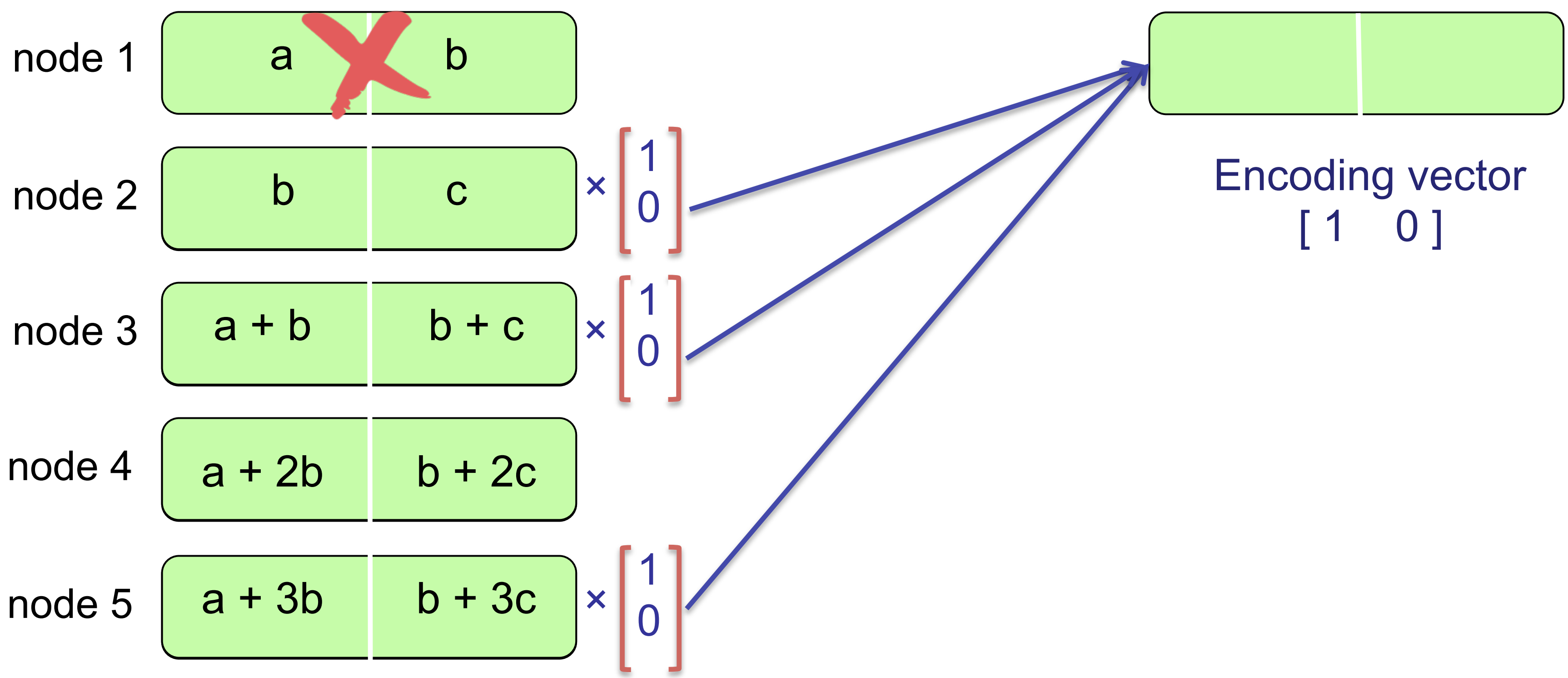


Toy Example: Minimum Bandwidth Code

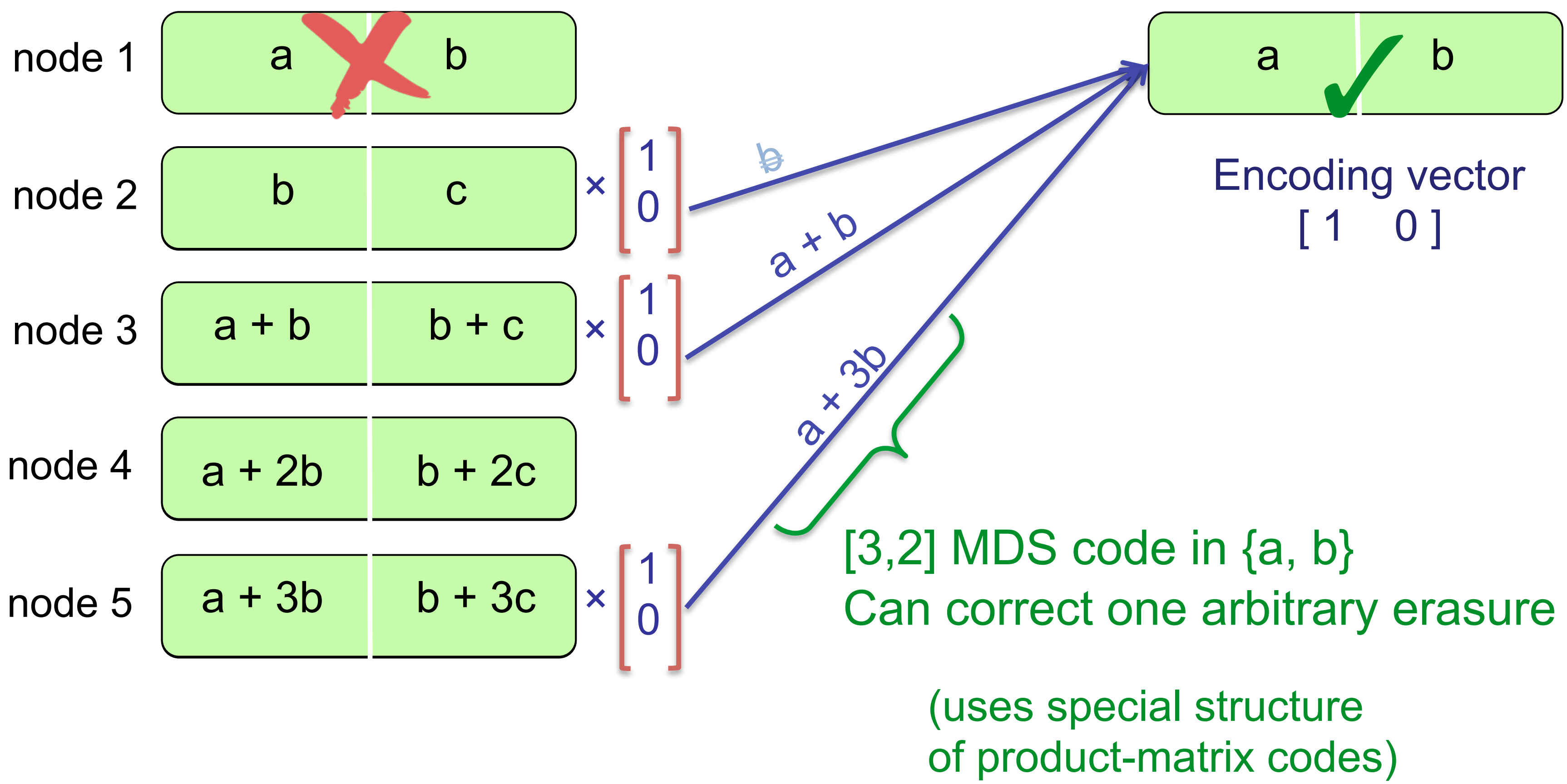
- For repair resilient to one erasure: connect to one additional node



Toy Example: Minimum Bandwidth Code

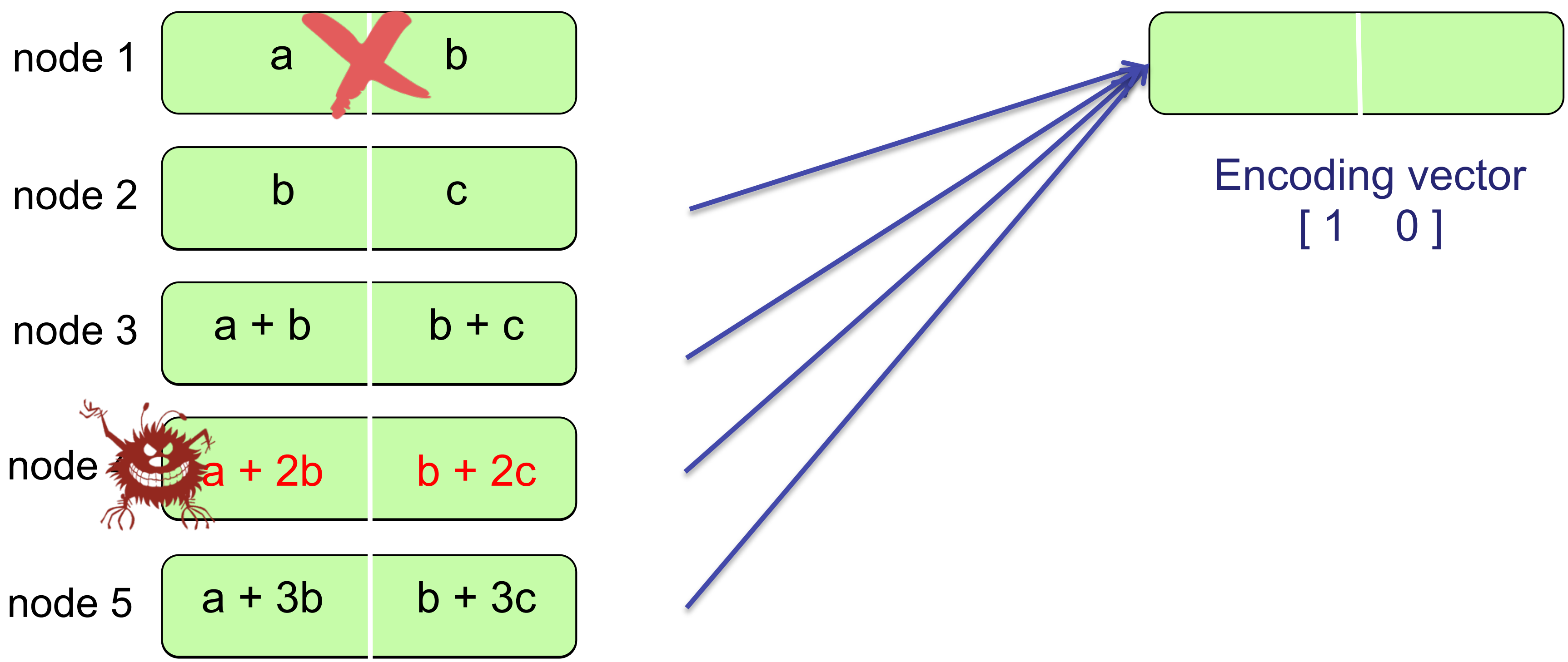


Toy Example: Minimum Bandwidth Code

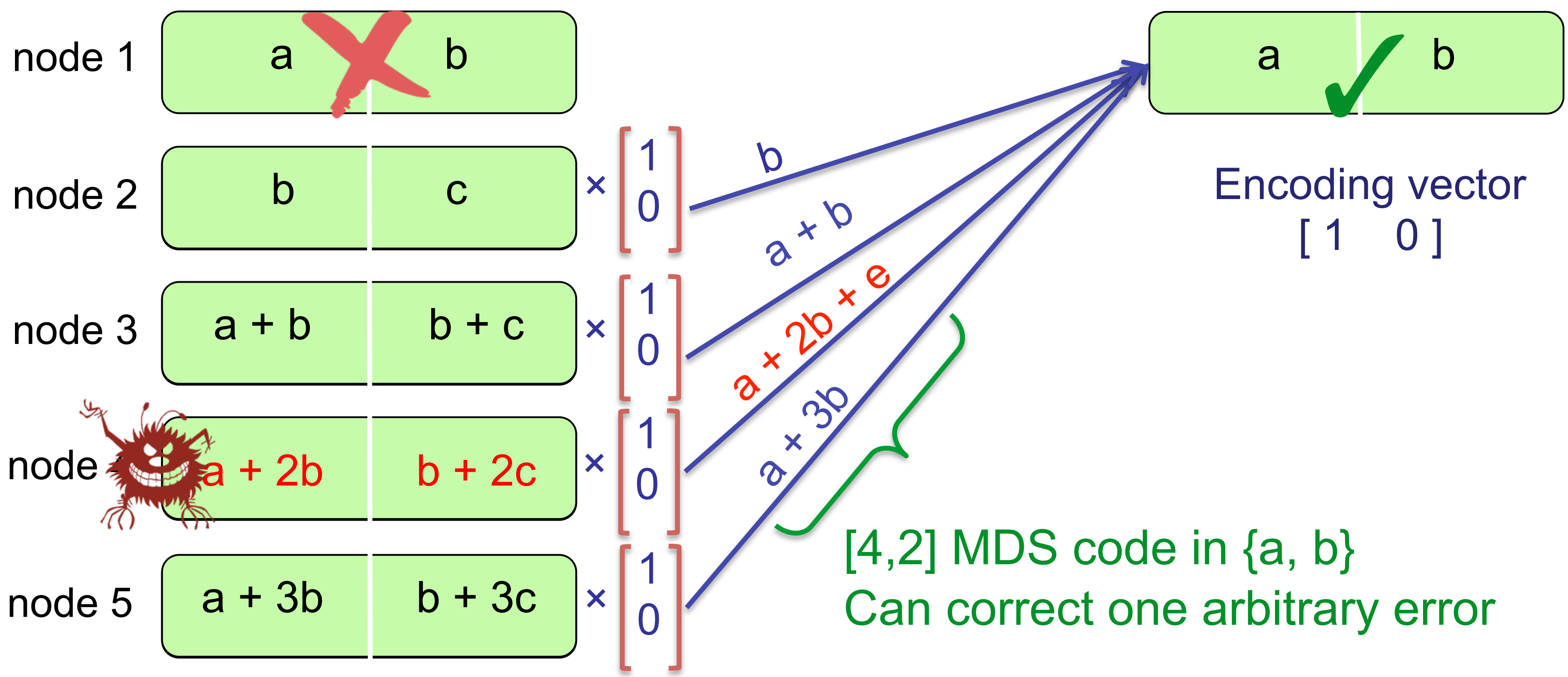


Toy Example: Minimum Bandwidth Code

- For repair resilient to one error: connect to two additional nodes

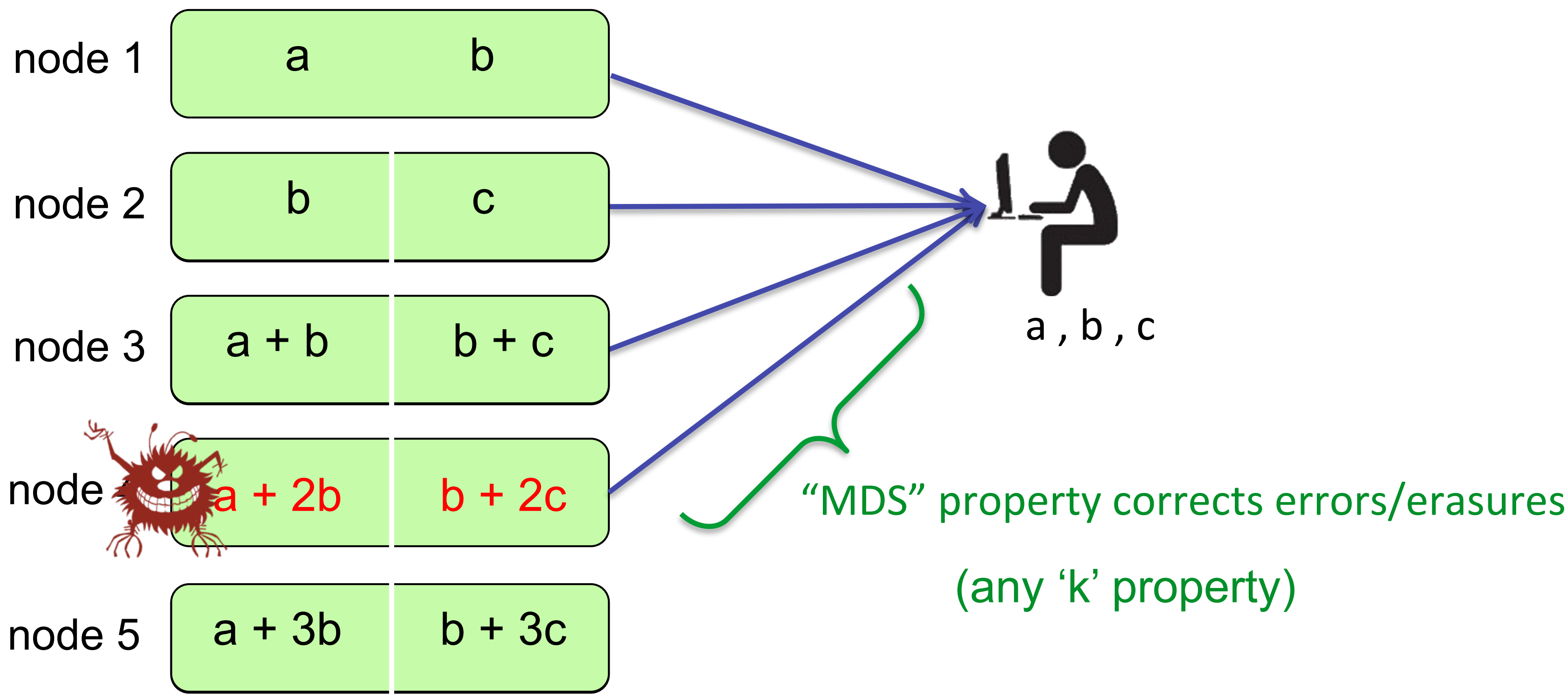


Toy Example: Minimum Bandwidth Code



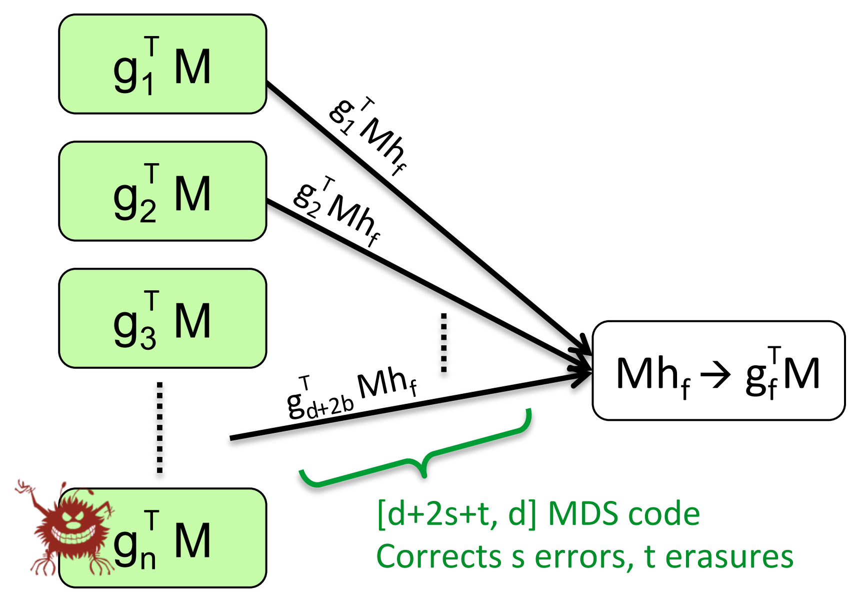
Toy Example: Minimum Bandwidth Code

- Data-reconstruction in presence of errors/erasures



General Algorithm

- Encode and store using Product-Matrix code for the error/erasure-free case
- For resilience from s errors and t erasures, connect to
  - d+2s+t for repair
  - k+2s+t for reconstruction
- Helping nodes oblivious to resilience requirements
- Resulting data passed is MDS



Analysis I: Secure-capacity

- Meets outer bound (Pawar et al. '11)

$$B \leq \sum_{i=2s+t}^{k-1} \min(\alpha, (d-i)\beta)$$

Establishes the capacity of these systems

- Related to node compromise in network coding
  - not very well understood in general
  - here: practical, explicit algorithm achieving capacity

Analysis II: Encoding Independent of Resiliency Requirements

- Meets outer bound

$$B \leq \sum_{i=2s+t}^{k-1} \min(\alpha, (d-i)\beta)$$

simultaneously for all s (#errors) and t (#erasures)

"Universal Resilience"

Analysis II: Encoding Independent of Resiliency Requirements

- Meets outer bound

$$B \leq \sum_{i=2s+t}^{k-1} \min(\alpha, (d-i)\beta)$$

simultaneously for all s (#errors) and t (#erasures)

"Universal Resilience"

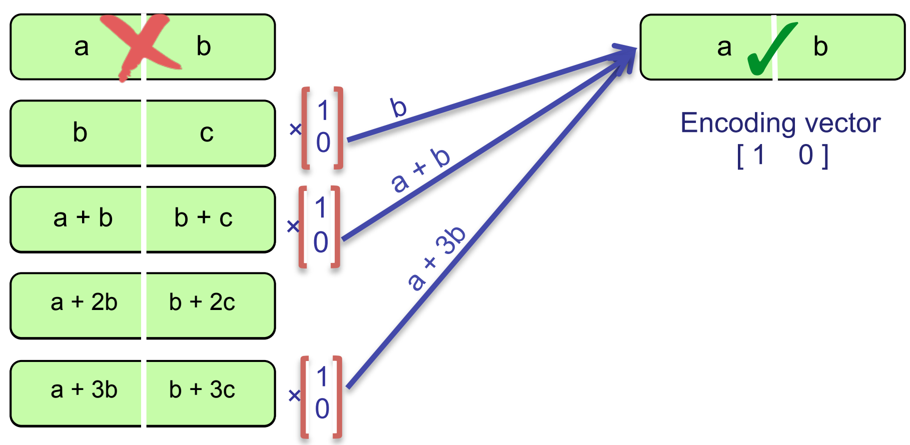
- Can choose different protection level for each instance of repair or reconstruction
  - handle time-varying channels/requirements
  - while always remaining optimal !
- Need not design for worst case
  - saves resources

Analysis III: What about other regenerating codes ?

- Our codes exploit structure of the Product-Matrix framework

Why Product-Matrix framework ?

- Higher connectivity available (d<n-1)
- Data passed by a node independent of other helpers



Analysis III: What about other regenerating codes ?

Necessary and sufficient:

- Higher connectivity available (d<n-1)
- Data passed by a node independent of other helpers

- Other existing codes[1] restrict number of nodes to n=d+1
  - thus, not applicable in this setting

[1] The high-rate MSR (d=k+1) explicit codes by Rashmi et al. (Allerton '09) do not impose this restriction, however, they perform only approximately-exact repair

Summary

- Explicit codes for correcting errors and erasures
  - Employing product-matrix framework
- Achieve and establish capacity
  - Related to network-coding with compromised nodes
- Universal resilience
  - Encoding independent of error protection requirements
- Necessary & sufficient conditions for any regenerating code
- Open: capacity in presence of errors/erasures for
  - MSR when redundancy < $\left(2 - \frac{1}{k}\right)$
  - Interior points

Thanks!

Ads – Why these ads?

Secret Share Dissemination Across a Network

Recent-results poster session

Wednesday 9.50am, Stratton Student Center,
Private Dining Rooms 1 & 2 on Third Floor

Backup Slides

Product-Matrix Codes

- Completely solves
  - MBR for all parameters
  - MSR for redundancy ≥ $\left(2 - \frac{1}{k}\right)$
- Scalable
  - n independent of all other parameters
  - Only construction supporting arbitrary #nodes
- Decentralized
  - Can connect to *any* subset of 'd' nodes for repair
- Ready to go
  - Can use (already existing) Reed-Solomon encoders/decoders for implementation
- Optimal

Product-Matrix Framework

Explicit MBR codes for all n, k, d
Explicit MSR codes for all n, k, $d \in [2k-2, n-1]$

$$\underbrace{C}_{n\times\alpha} = \underbrace{\Psi}_{n\times d}\underbrace{M}_{d\times\alpha}$$

- C : Code matrix
  - Every row represents one node
  - α symbols stored in i^{th} node are $\underline{\psi}_i^t M$
- Ψ : Encoding matrix
  - Fixed apriori
- M : Message matrix
  - Contains the B source symbols, with some symbols possibly repeated

Product-matrix MBR code

- $\alpha = d\beta$ and $B = \frac{k(k+1)}{2} + k(d-k)$

- Code $\underbrace{C}_{n\times\alpha} = \underbrace{\Psi}_{n\times d}\underbrace{M}_{d\times\alpha}$

- Message matrix $\underbrace{M}_{d\times d} = \begin{bmatrix} \underbrace{S}_{k\times k} & \underbrace{T}_{k\times(d-k)} \\ \underbrace{T^t}_{(d-k)\times k} & \underbrace{0}_{k\times(d-k)} \end{bmatrix}$

  S symmetric ($\Rightarrow M$ symmetric)

- Encoding matrix $\underbrace{\Psi}_{n\times d} = \begin{bmatrix} \underbrace{\Phi}_{n\times k} & \underbrace{\Delta}_{n\times(d-k)} \end{bmatrix}$

  Φ: any k rows linearly independent
  Ψ: any d rows linearly independent
  e.g., Ψ is a Vandermonde or Cauchy matrix

Product-matrix MBR code : Node repair

Replacement node f needs: $\underline{\psi}_f^t M$

Helper node i stores: $\underline{\psi}_i^t M$

Helper node i passes: $\underline{\psi}_i^t M \underline{\psi}_f$

From d nodes $\downarrow$

$\Psi_{\text{rep}} M \underline{\psi}_f$

(Ψ_{rep} is $d \times d$, invertible)

$\downarrow$

$M \underline{\psi}_f$

(M is symmetric)

$\downarrow$

$\underline{\psi}_f^t M$

$$\underbrace{C}_{n\times\alpha} = \underbrace{\Psi}_{n\times d} \underbrace{M}_{d\times alpha}$$

$$\underbrace{M}_{d\times d} = \begin{bmatrix} \underbrace{S}_{k\times k} & \underbrace{T}_{k\times(d-k)} \\ \underbrace{T^t}_{(d-k)\times k} & \underbrace{0}_{k\times(d-k)} \end{bmatrix}$$

S symmetric $\Rightarrow M$ symmetric

$$\underbrace{\Psi}_{n\times d} = \begin{bmatrix} \underbrace{\Phi}_{n\times k} & \underbrace{\Delta}_{n\times(d-k)} \end{bmatrix}$$

Φ: any k rows LI

Ψ: any d rows LI

Product-matrix MBR code : Data Reconstruction

Node i passes: $\underline{\psi}_i^t M$

From k nodes $\downarrow$

$$\Psi_{\text{DC}} M$$

$(\Psi_{\text{DC}} = [\Phi_{\text{DC}} \quad \Delta_{\text{DC}}]$ is $(k \times d))$

$$\downarrow$$

$$\left[\ \Phi_{\text{DC}} S + \Delta_{\text{DC}} T^t \quad \Phi_{\text{DC}} T \ \right]$$

$$\downarrow$$

Φ_{DC} is $k \times k$, invertible

Decode T

$$\downarrow$$

Subtract $\Delta_{\text{DC}} T^t$, Decode S

$$\underbrace{C}_{n \times \alpha} = \underbrace{\Psi}_{n \times d} \underbrace{M}_{d \times alpha}$$

$$\underbrace{M}_{d \times d} = \begin{bmatrix} \underbrace{S}_{k \times k} & \underbrace{T}_{k \times (d-k)} \\ \underbrace{T^t}_{(d-k) \times k} & \underbrace{0}_{k \times (d-k)} \end{bmatrix}$$

S symmetric$\Rightarrow M$ symmetric

$$\underbrace{\Psi}_{n \times d} = \begin{bmatrix} \underbrace{\Phi}_{n \times k} & \underbrace{\Delta}_{n \times (d-k)} \end{bmatrix}$$

Φ: any k rows LI

Ψ: any d rows LI

Product-Matrix MSR Codes

- $\alpha = \frac{B}{k}$
- Hence, necessarily MDS (over alphabet $\mathbb{F}_q^{\alpha}$)
- $d = (k-1) + \alpha$ (striping: $\beta = 1$)

- We design codes for all n, k, $d \in [2k-2, n-1]$
- $C = \Psi M$, $\Psi = [\Phi \ \Lambda\Phi]$, $M = \begin{bmatrix} S_1 \\ S_2 \end{bmatrix}$

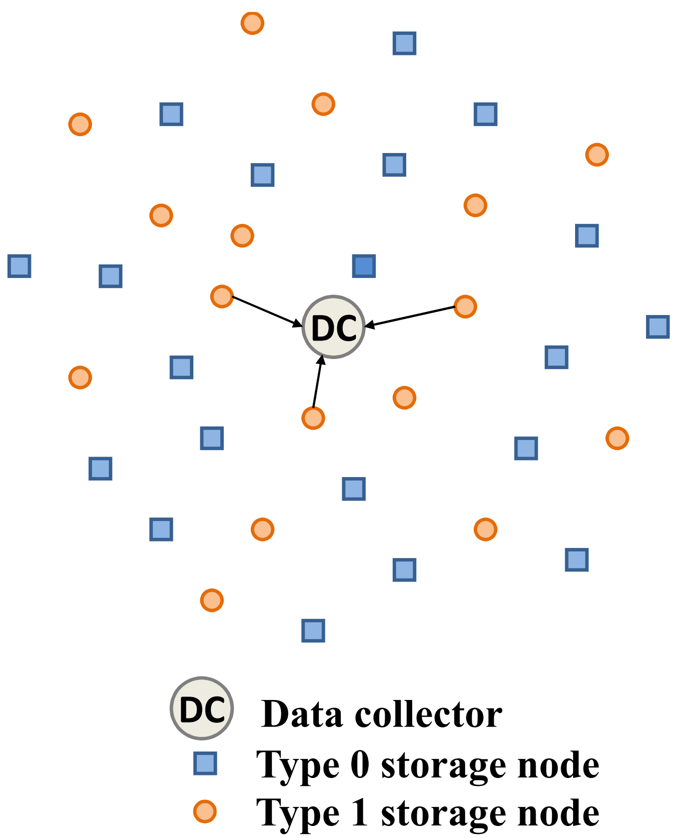
- Involves solving multiple simultaneous interference-alignment constraints

Twin Codes

- New framework to allow use of *any erasure code* and still have efficient repair

- Properties of these constituent codes used during data reconstruction/repair
  - low complexity decoding
  - error detection/correction
  - ratelessness
  - etc.

Twin Codes

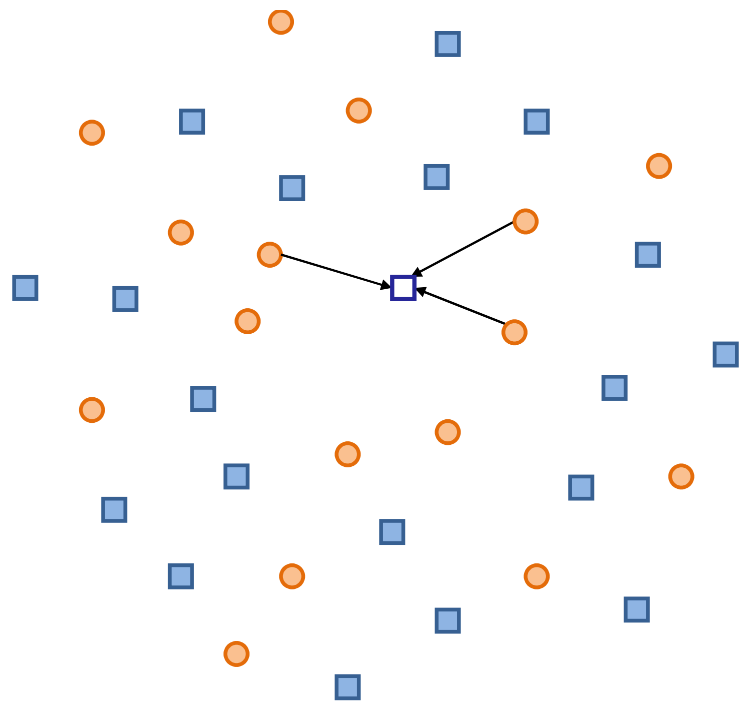


Two types of nodes

- encoded using two different codes

Data reconstruction by connecting to nodes of the *same* type

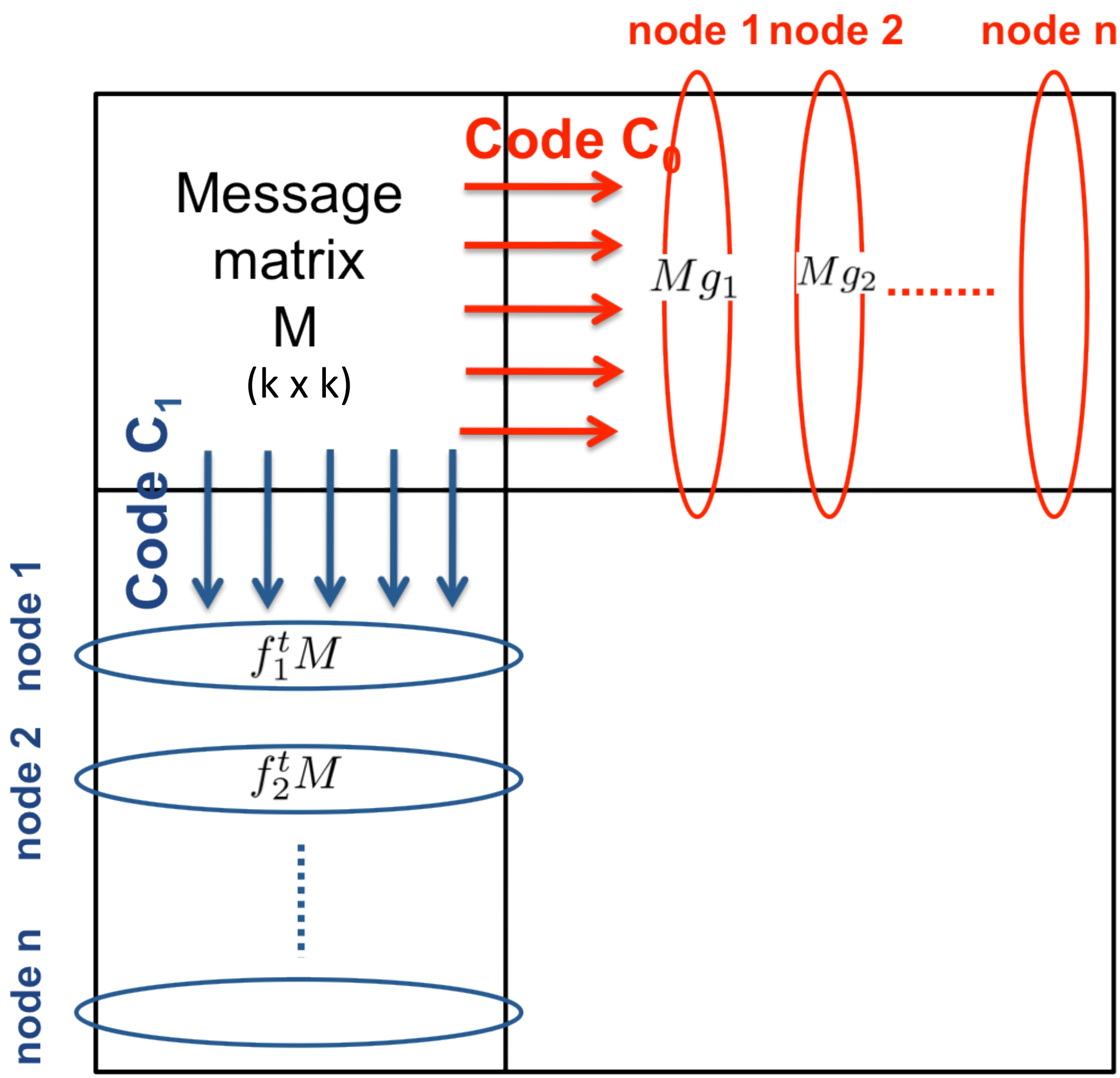
Twin Codes



Replacement node of Type 0

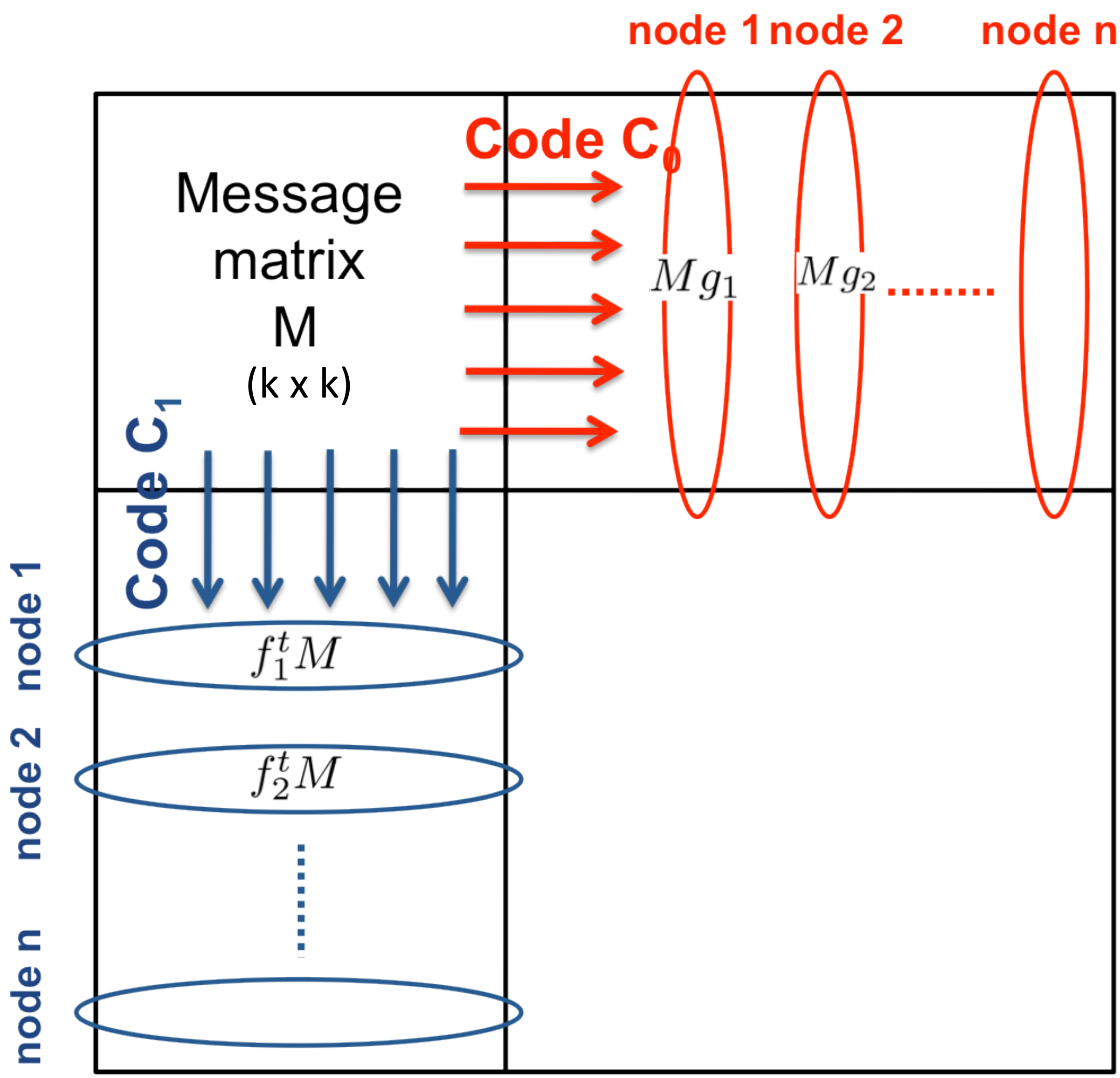
Node repair by connecting to nodes of the *other* type

Twin Codes Construction



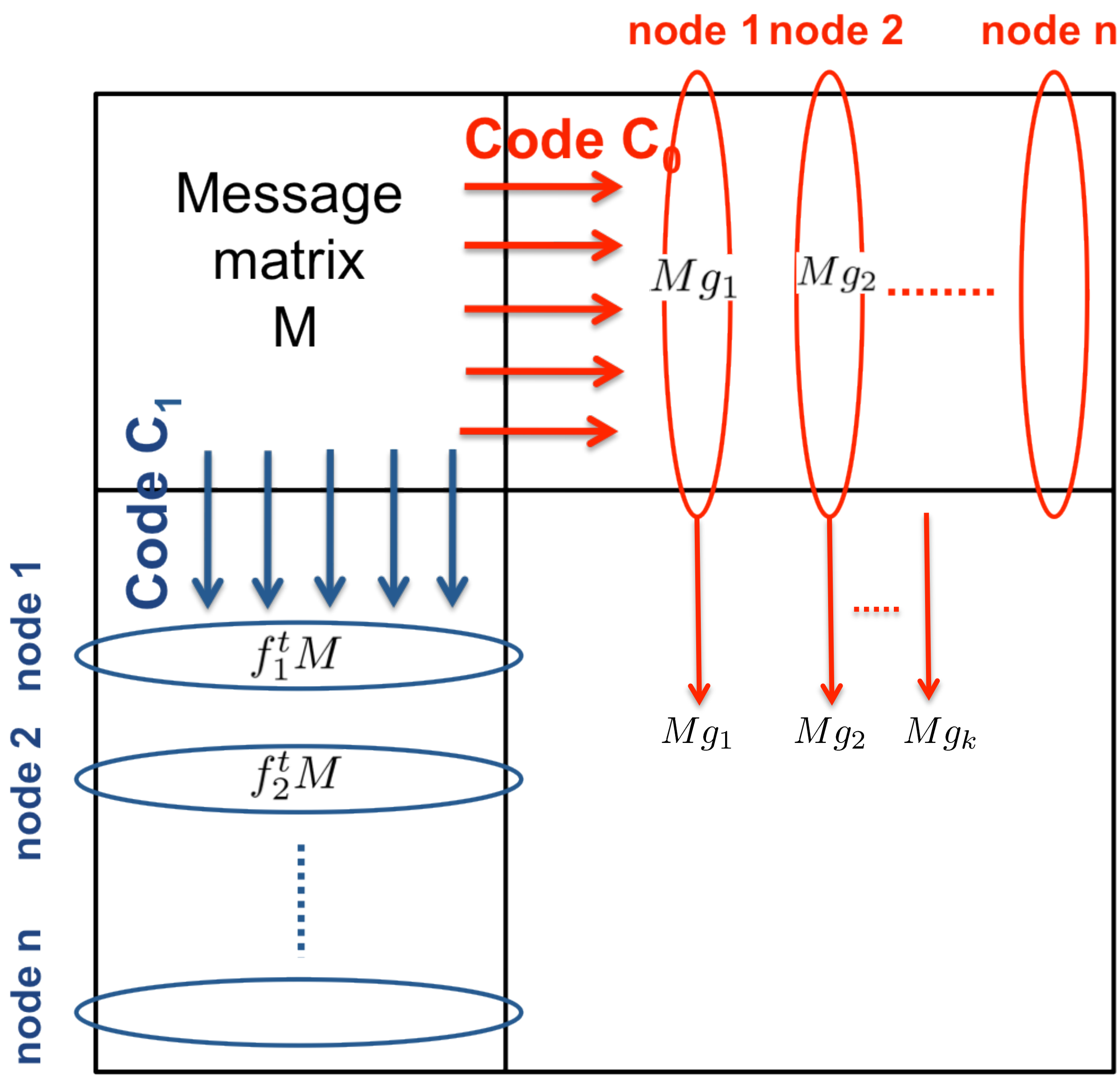
f_i

Twin Codes Construction



- any k of the g_i are linearly independent
- any k of the f_i are linearly independent

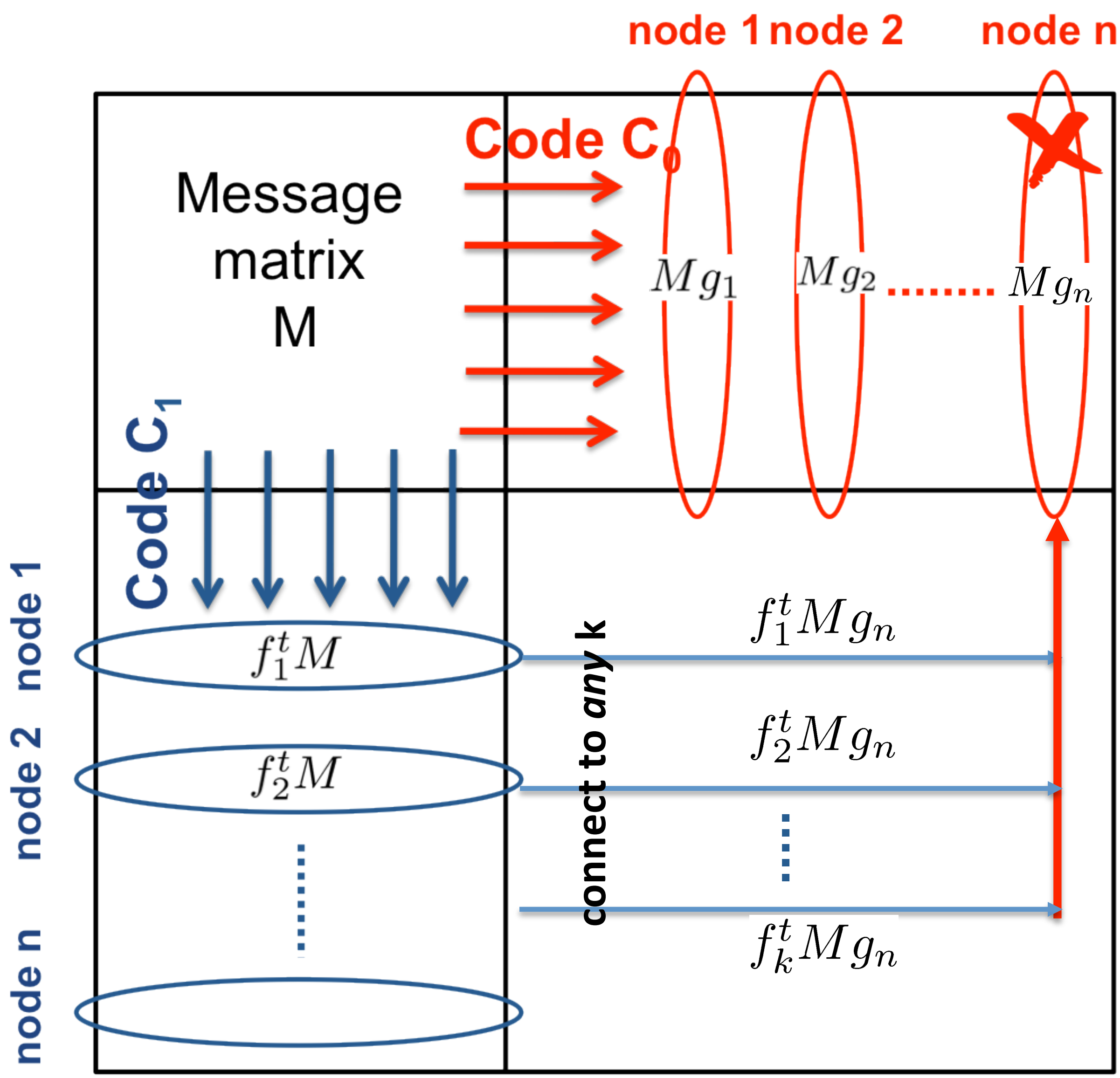
Twin Code: Data reconstruction



- Connect any k nodes of *one* type
- row-by-row decoding of data

- any k of the g_i are linearly independent
- any k of the f_i are linearly independent

Twin Code: Repair



- any k of the g_i are linearly independent
- any k of the f_i are linearly independent