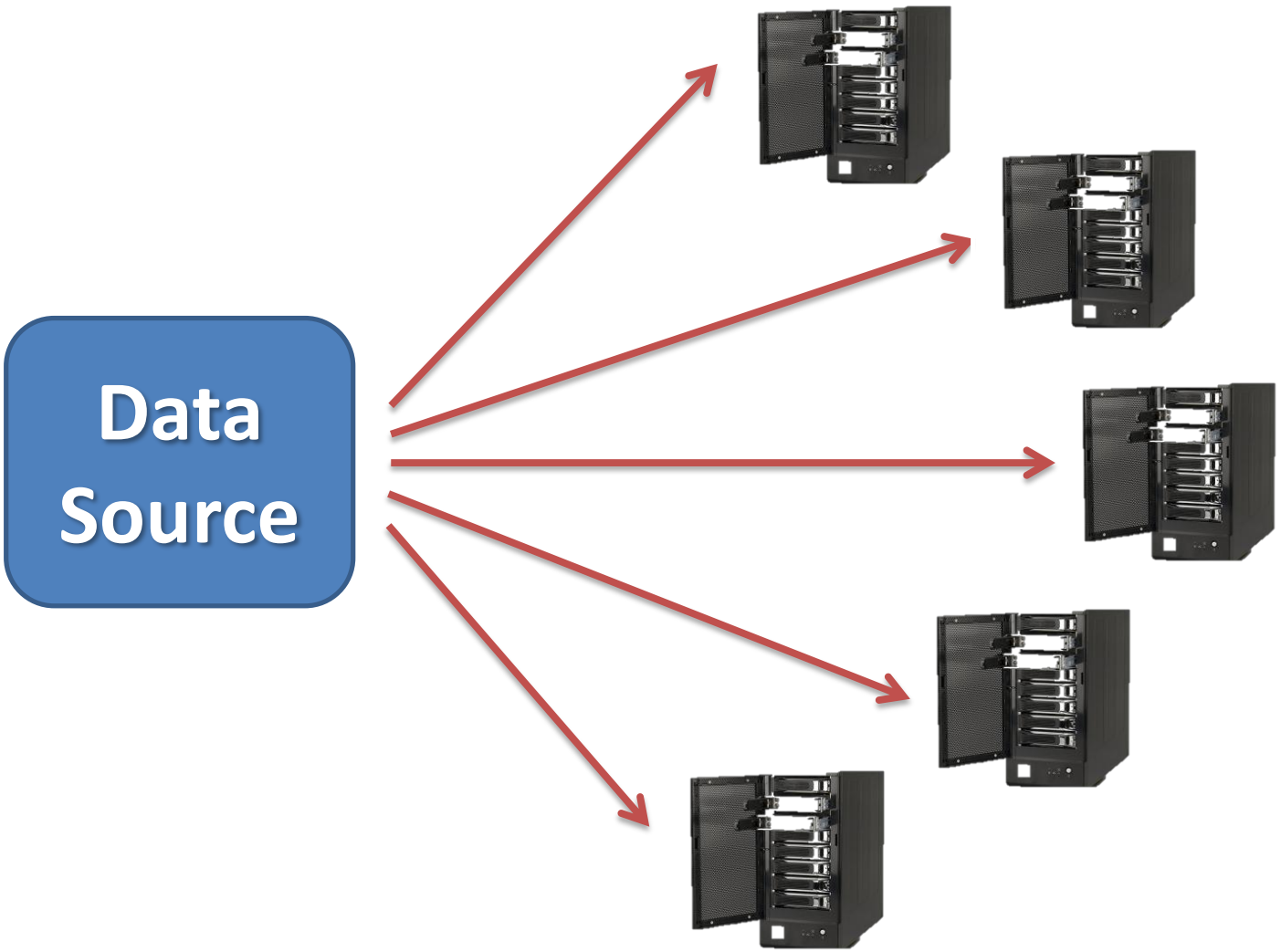
Update and Repair Efficient Codes for Distributed Storage

Ankit Singh Rawat

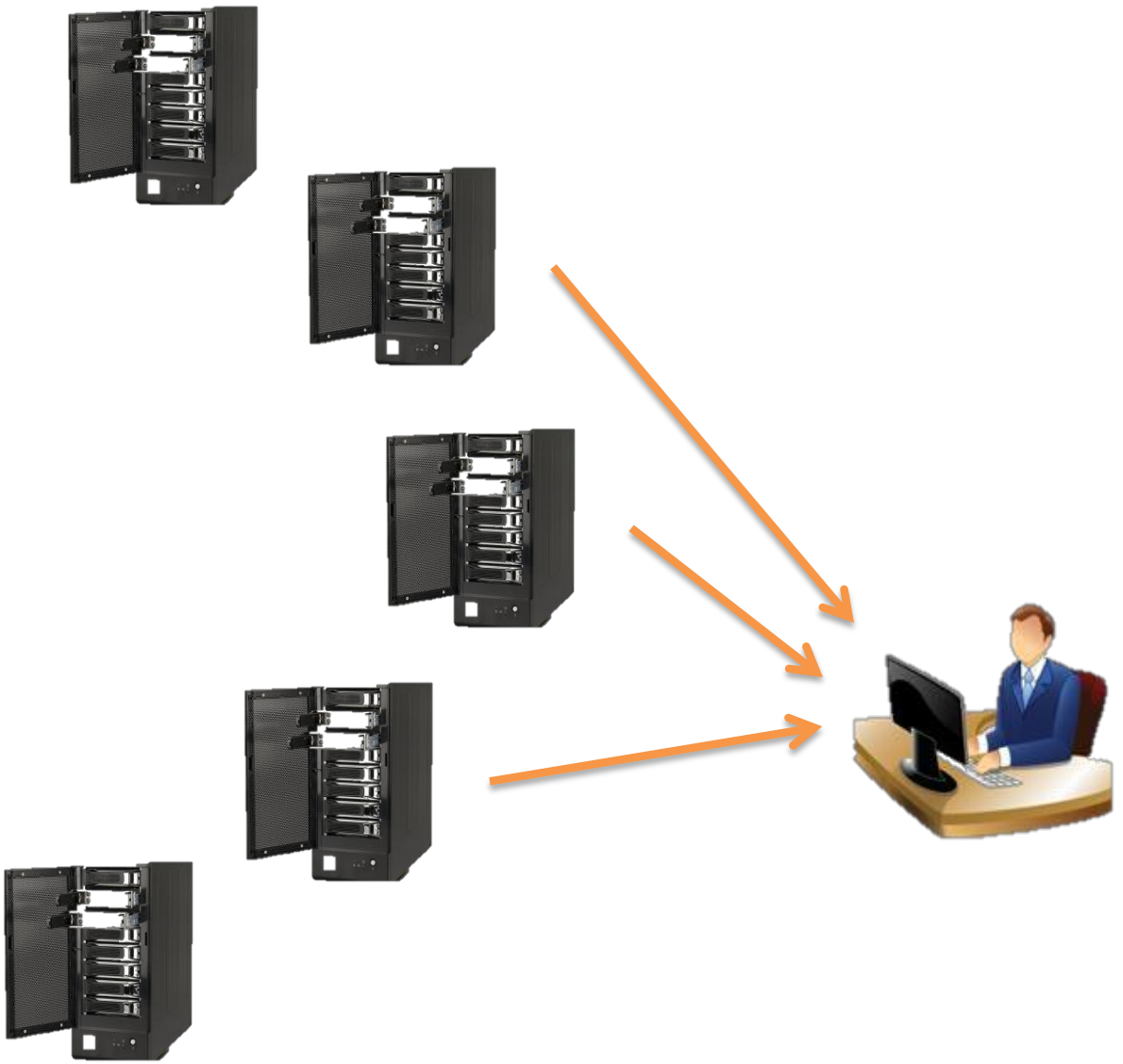
Wireless Networking and Communications Group (WNCG)
The University of Texas at Austin

December 18, 2013

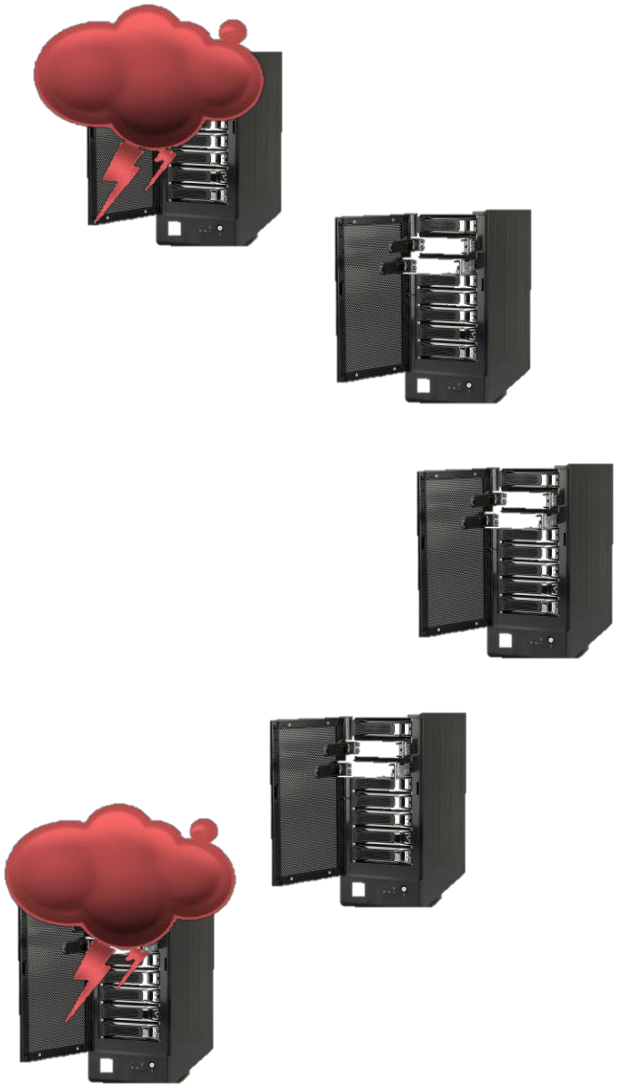
Distributed Storage System (DSS)



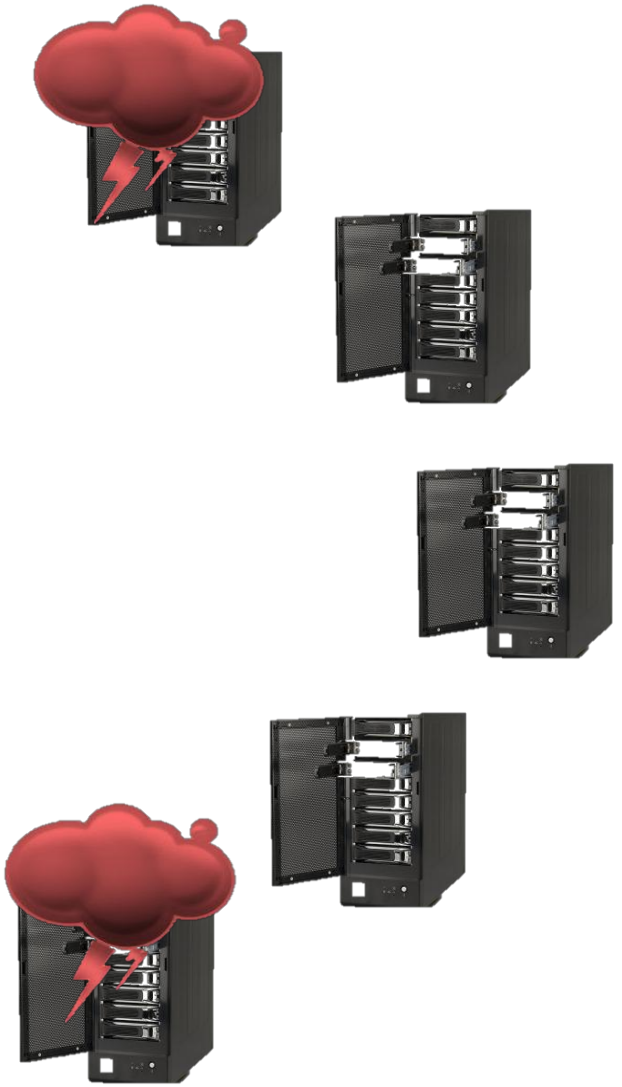
Data reconstruction



Node failure

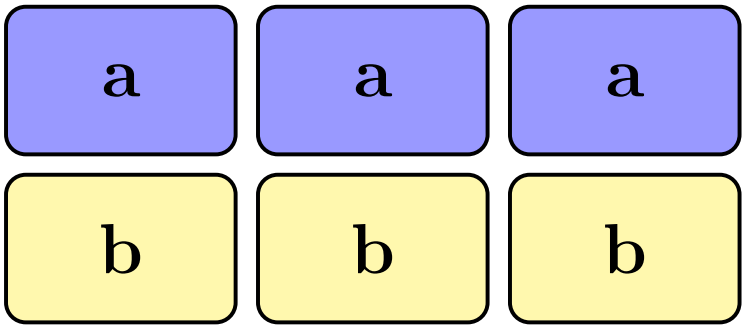


Node failure

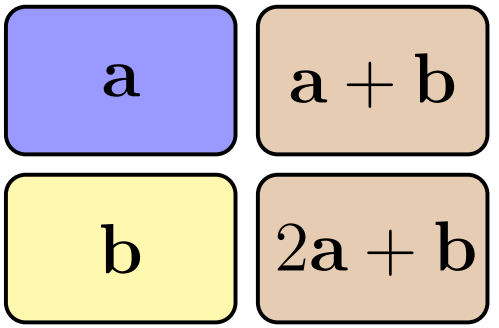


Need to introduce redundancy into the system.

Replication vs. coding

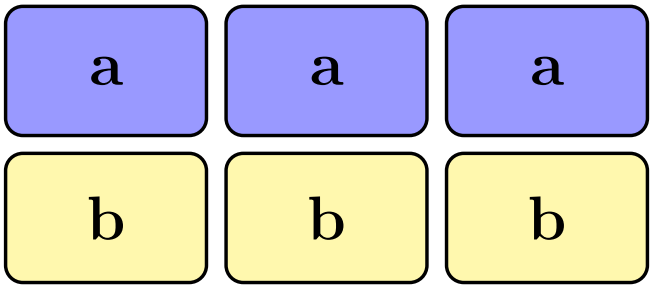


3-replication

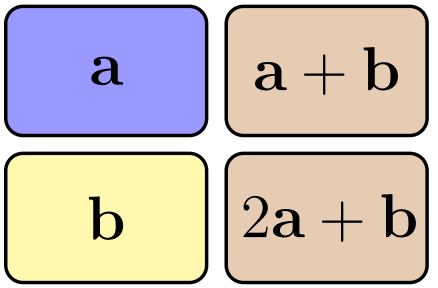


$(4, 2)$ Erasure coding

Replication vs. coding



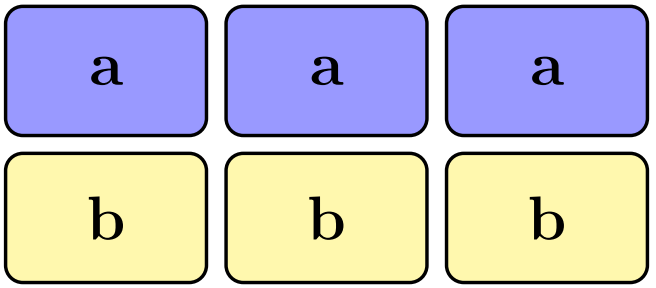
3-replication



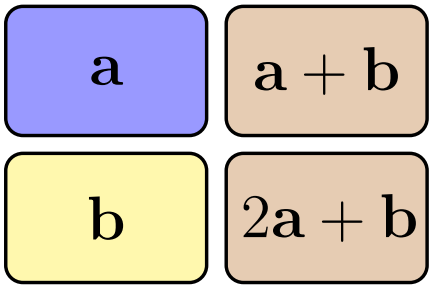
$(4, 2)$ Erasure coding

- Traditional MDS codes are optimal for *storage vs. reliability* trade-off.

Replication vs. coding



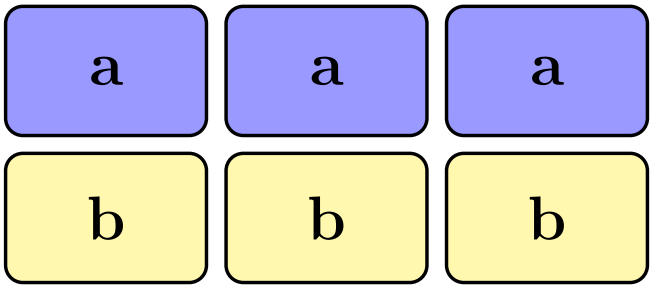
3-replication



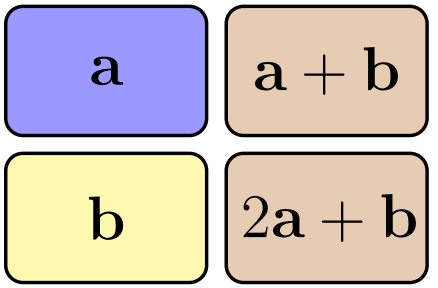
$(4, 2)$ Erasure coding

- Traditional MDS codes are optimal for *storage vs. reliability* trade-off.
- Reliability is not the only metric of interest:
  - Overhead of updating data.
  - Node repair overhead.
  - Availability.
  - .....

Replication vs. coding



3-replication



$(4, 2)$ Erasure coding

- Traditional MDS codes are optimal for *storage vs. reliability* trade-off.
- Reliability is not the only metric of interest:
  - Overhead of updating data.
  - Node repair overhead.
  - Availability.
  - .....

May need to move away from MDS codes.

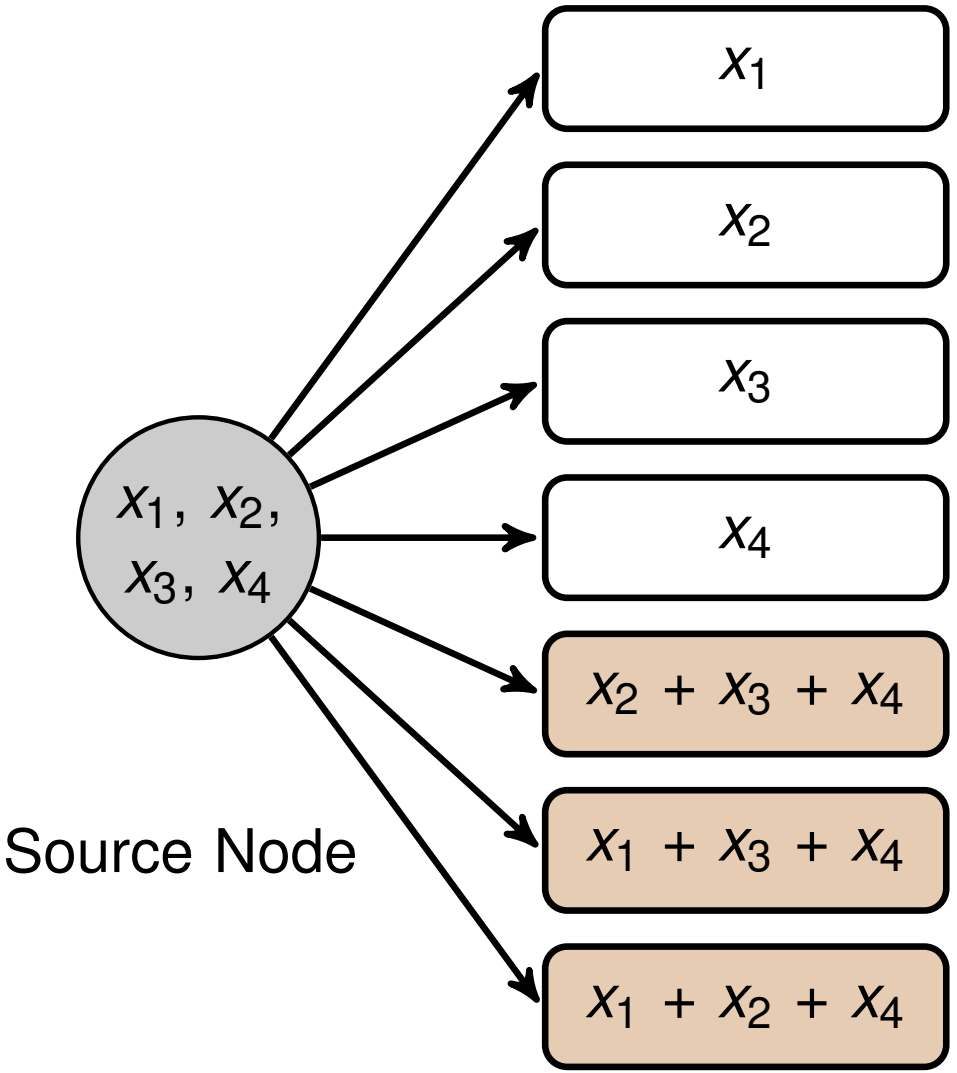
Overhead of updating data (Update Complexity)

- Information to be stored is almost never static.

Overhead of updating data (Update Complexity)

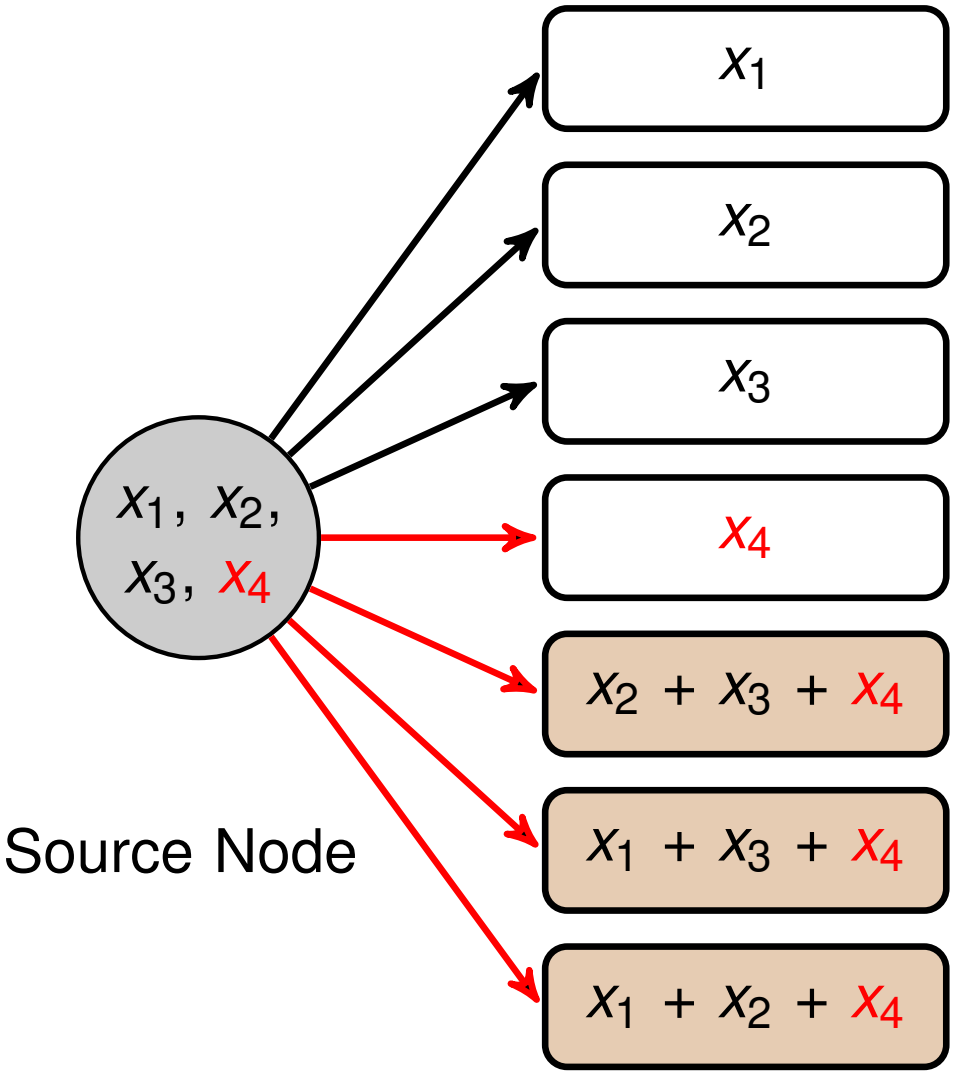
- Information to be stored is almost never static.
- **Update complexity:** maximum number of encoded symbols that must be updated when any single information symbol is changed.

Example: $(7, 4)$ Hamming code



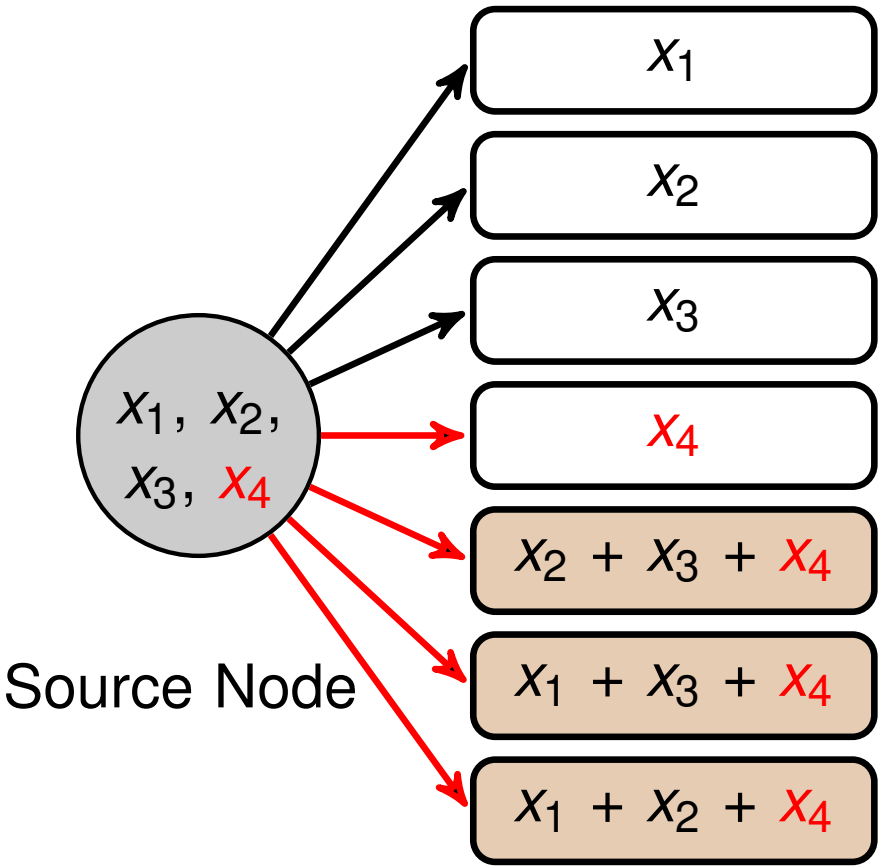
$$\mathbf{y} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

Example: $(7, 4)$ Hamming code



$$\mathbf{y} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

Example: $(7, 4)$ Hamming code



$$\mathbf{y} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

Update complexity $\geq$ minimum distance.

Update Complexity

- Information to be stored is almost never static.
- **Update complexity:** maximum number of encoded symbols that must be updated when any single information symbol is changed.
- Updating data consumes bandwidth and energy.

Design codes with *small update complexity*.

Update Efficiency

- Constant update complexity.

Update Efficiency

- Constant update complexity.
- Linear update complexity.

Update Efficiency

- Constant update complexity.
- Linear update complexity.
  - Low update complexity MDS codes for storage:
    - ★ e.g. X-code **[XuBruck]**, EVENODD code **[BlaumBradyBruckMenon]**.

Update Efficiency

- Constant update complexity.
- Linear update complexity.
  - Low update complexity MDS codes for storage:
    - ★ e.g. X-code **[XuBruck]**, EVENODD code **[BlaumBradyBruckMenon]**.
- Coding schemes with sub-linear update complexity.

Kolchin generator (KG) codes

- Random ensemble of codes with logarithmic update complexity.

P. Anthapadmanabhan, E. Soljanin and S. Vishwanath, in Allerton 2010.

Kolchin generator (KG) codes

- Random ensemble of codes with logarithmic update complexity.
- Generator matrix:

$$G = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 1 & 0 & \cdots & 0 \end{bmatrix}_{n \times k_n}$$

$$\mathbb{P}(G_{i,j} = 1) = \mathcal{O}\left(\frac{\log n}{n}\right)$$

P. Anthapadmanabhan, E. Soljanin and S. Vishwanath, in Allerton 2010.

Kolchin generator (KG) codes

- Random ensemble of codes with logarithmic update complexity.
- Generator matrix:

$$G = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 1 & 0 & \cdots & 0 \end{bmatrix}_{n \times k_n}$$

$$\mathbb{P}(G_{i,j} = 1) = \mathcal{O}\left(\frac{\log n}{n}\right)$$

- **With high probability**, every column has $\mathcal{O}(\log n)$ non-zero entries.

P. Anthapadmanabhan, E. Soljanin and S. Vishwanath, in Allerton 2010.

Failure tolerance of KG codes

- Minimum distance of KG codes is at most $\mathcal{O}(\log n)$.

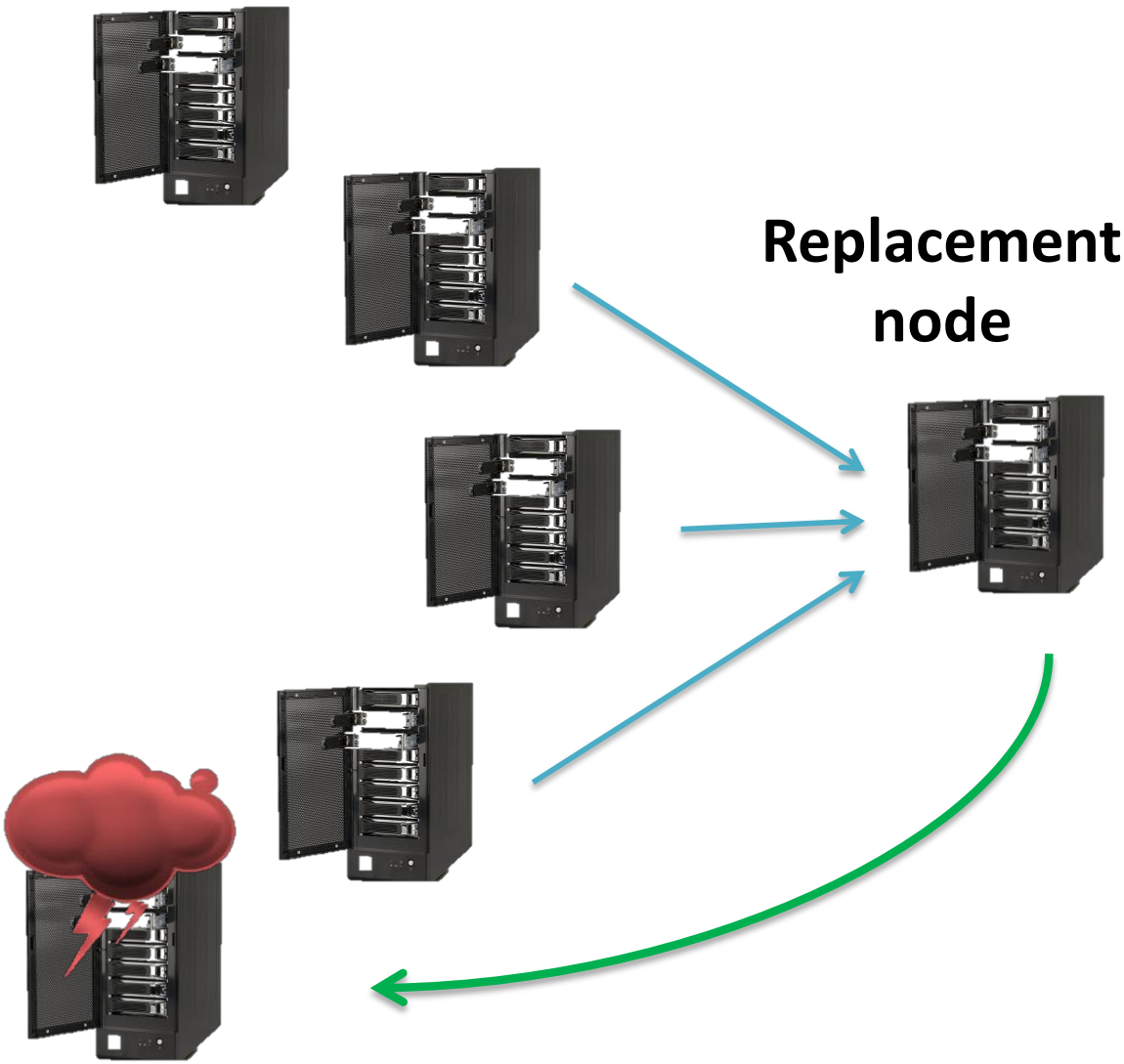
Failure tolerance of KG codes

- Minimum distance of KG codes is at most $\mathcal{O}(\log n)$.
- Random erasure model: each node answer with probability p.

Failure tolerance of KG codes

- Minimum distance of KG codes is at most $\mathcal{O}(\log n)$.
- Random erasure model: each node answer with probability p.
- **W.h.p.**, a random set of $k_n(1+\epsilon)$ encoded symbols is sufficient to reconstruct data.
  - Can tolerate erasure probability $p < \frac{n-k_n}{n}$.

Node repair



Efficient node repair

- Important to perform efficient node repairs.
- Multiple metrics to measure repair efficiency.
- **Repair bandwidth:** amount of data downloaded during a node repair.
- **Locality:** number of remaining nodes contacted during a node repair.

Efficient node repair

- Important to perform efficient node repairs.
- Multiple metrics to measure repair efficiency.
- **Repair bandwidth:** amount of data downloaded during a node repair.
  - **A. S. Rawat, A. Bhowmick, S. Vishwanath and E. Soljanin**, in ISIT 2011.
- **Locality:** number of remaining nodes contacted during a node repair.
  - **M. Asteris and A. Dimakis**, in ISIT 2012.
  - Modification of KG codes appraoch.
  - **A. Mazumdar, V. Chandar and G. Wornell**, in ITA 2013.

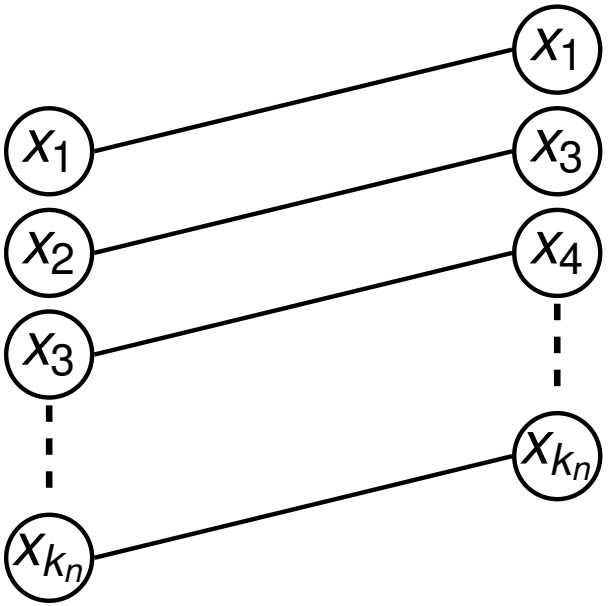
Repairable fountain codes

M. Asteris and A. Dimakis, in ISIT 2012.

Repairable fountain codes

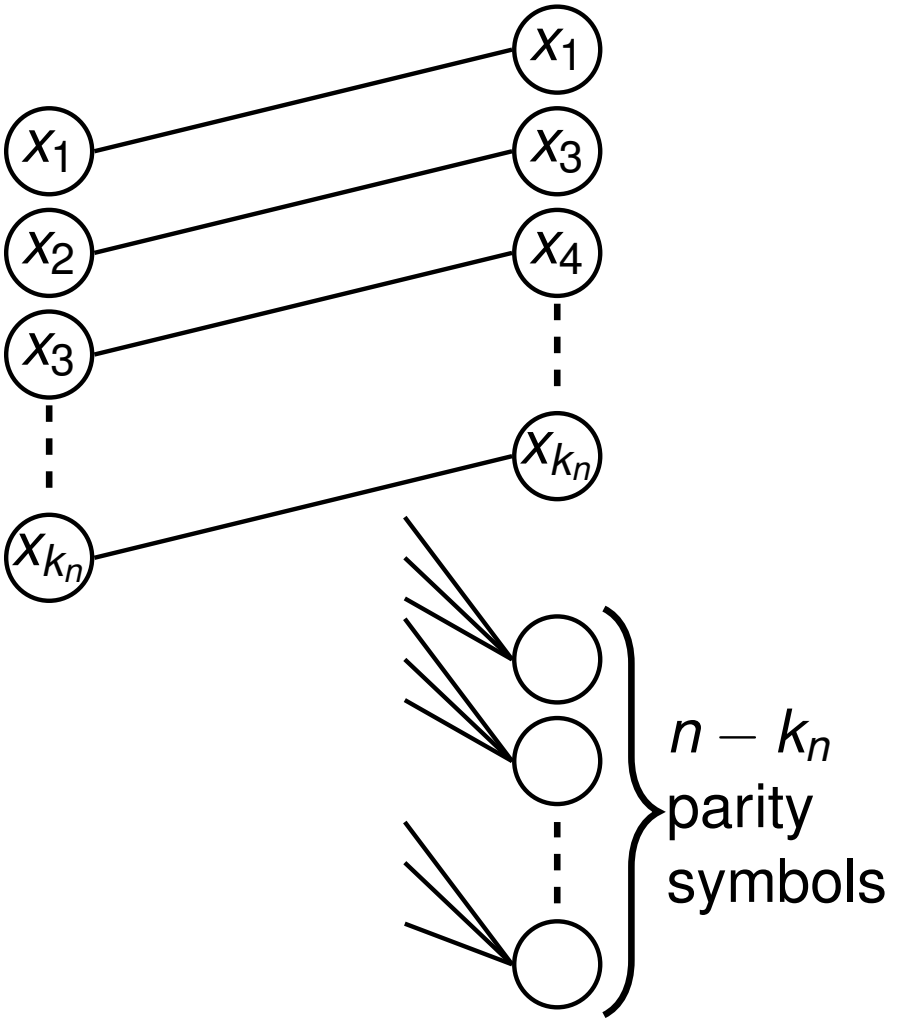
M. Asteris and A. Dimakis, in ISIT 2012.

Repairable fountain codes



M. Asteris and A. Dimakis, in ISIT 2012.

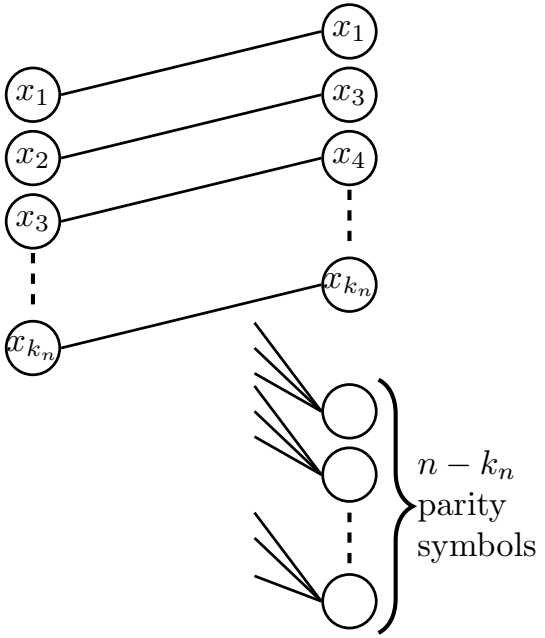
Repairable fountain codes



- Each parity node is obtained by randomly throwing $\Theta(\log n)$ edges.

M. Asteris and A. Dimakis, in ISIT 2012.

Repairable fountain codes.



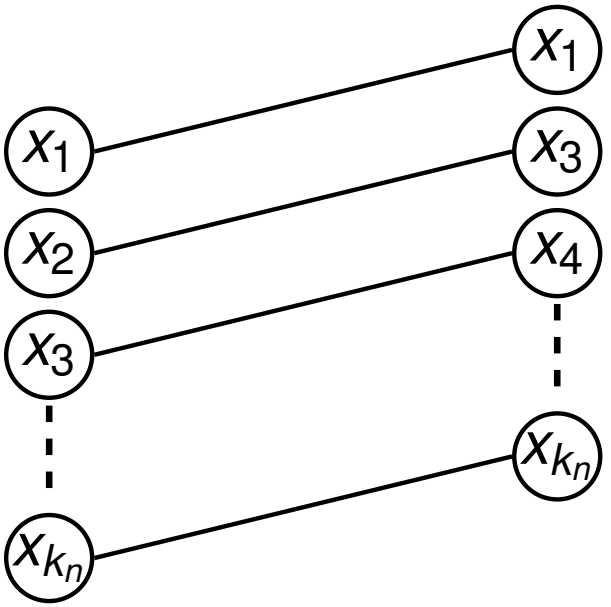
- Each node has locality $\Theta(\log n)$.
- **W.h.p.**, update complexity: $\mathcal{O}(\log n)$.
  - Maximum load of throwing $\mathcal{O}(k_n \log k_n)$ balls into k_n bins.
- Failure tolerance:
  - Any random set of $k_n(1+\epsilon)$ encoded symbols are sufficient for data reconstruction.

M. Asteris and A. Dimakis, in ISIT 2012.

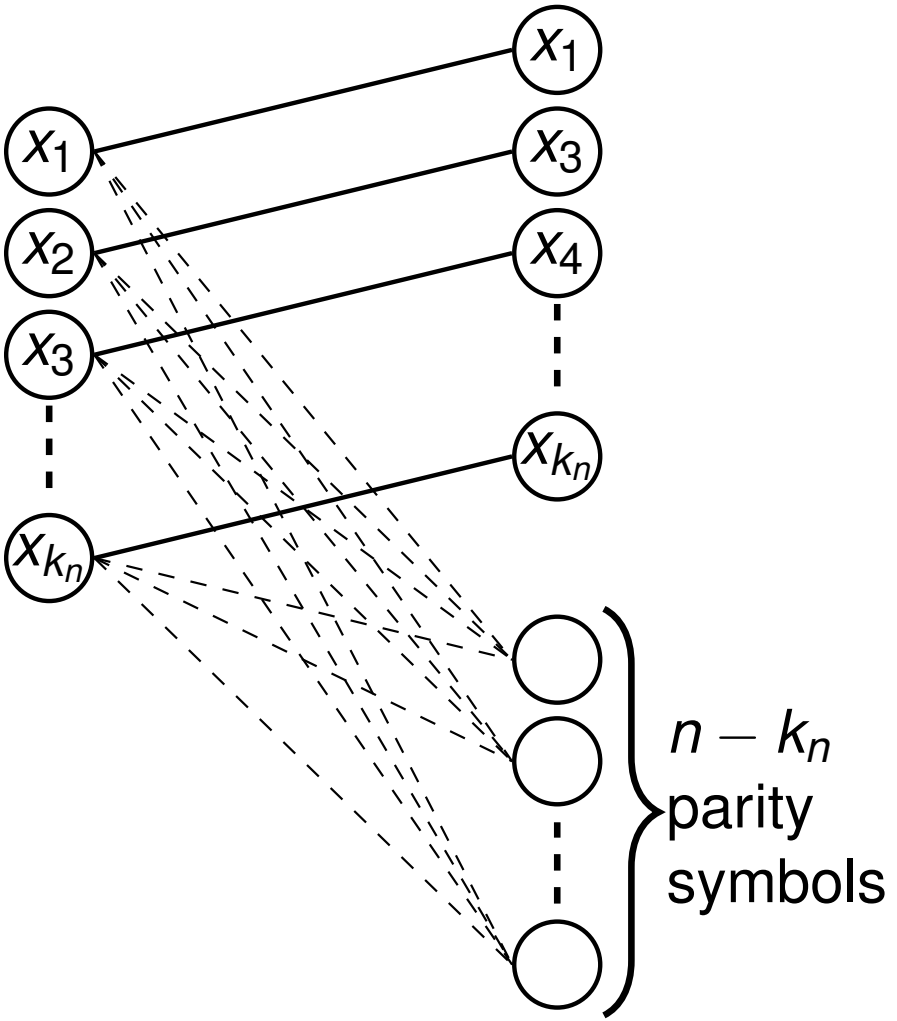
Modification of KG codes approach

Modification of KG codes approach

Modification of KG codes approach

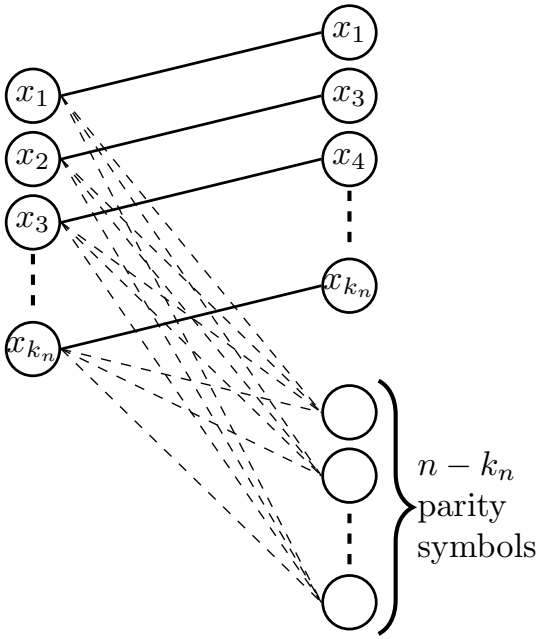


Modification of KG codes approach



- For parity nodes, each edge is present w. p. $\Theta(\frac{\log n}{n})$.

Modification KG codes approach.



- **W.h.p.**, each node has locality $\Theta(\log n)$.
- **W.h.p**, update complexity: $\mathcal{O}(\log n)$.
- Failure tolerance:
  - Any random set of $k_n(1+\epsilon)$ encoded symbols are sufficient for data reconstruction.

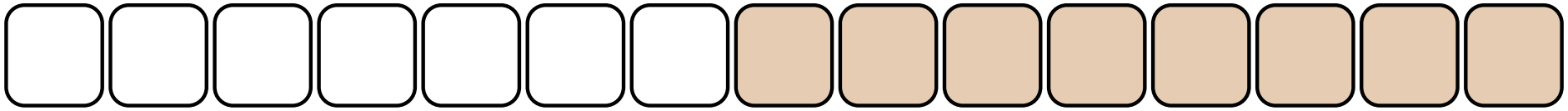
Discussion and open questions

- Explicit constructions for update efficient codes.

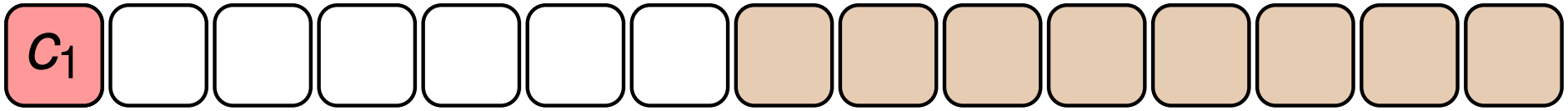
Discussion and open questions

- Explicit constructions for update efficient codes.
- Study of update efficient codes under general failure model.
  - **B. Kanukurthi, N. Chandran and R. Ostrovsky**, in TCC 2014
  - Construction of update efficient *locally decodable* codes for prefix hamming metrics.

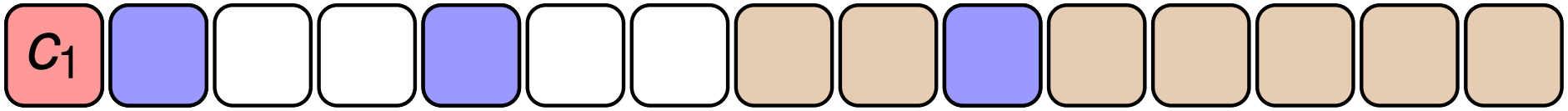
Locality and availability



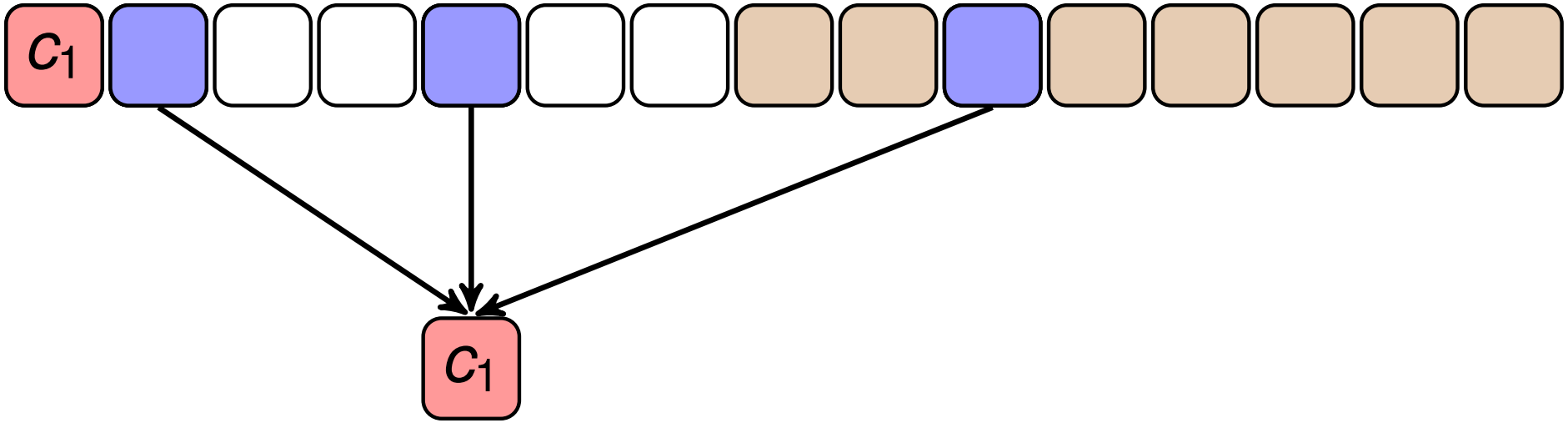
Locality and availability



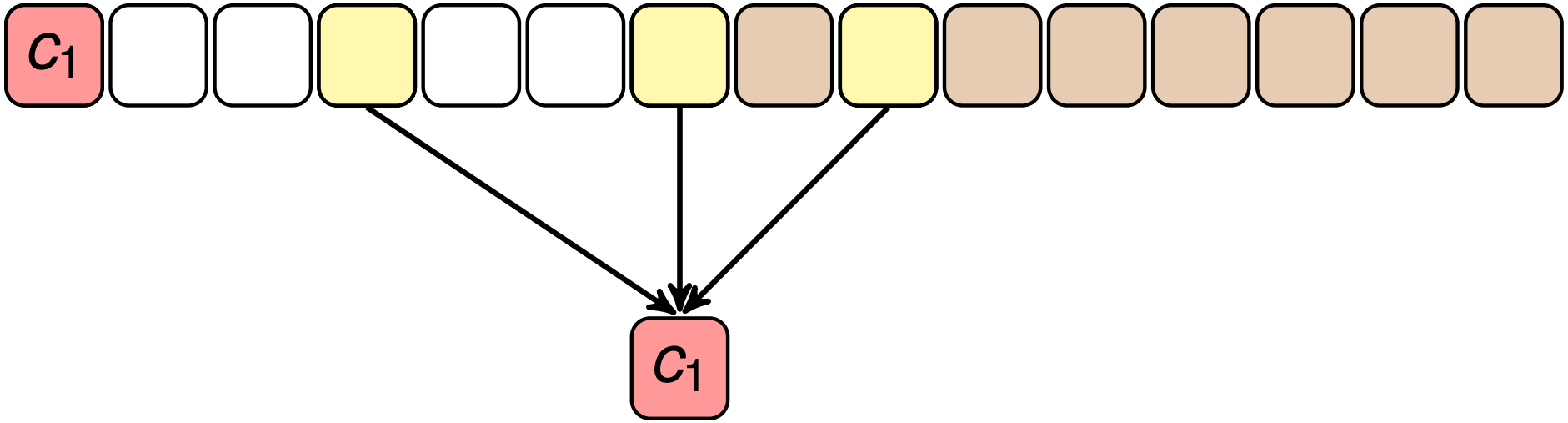
Locality and availability



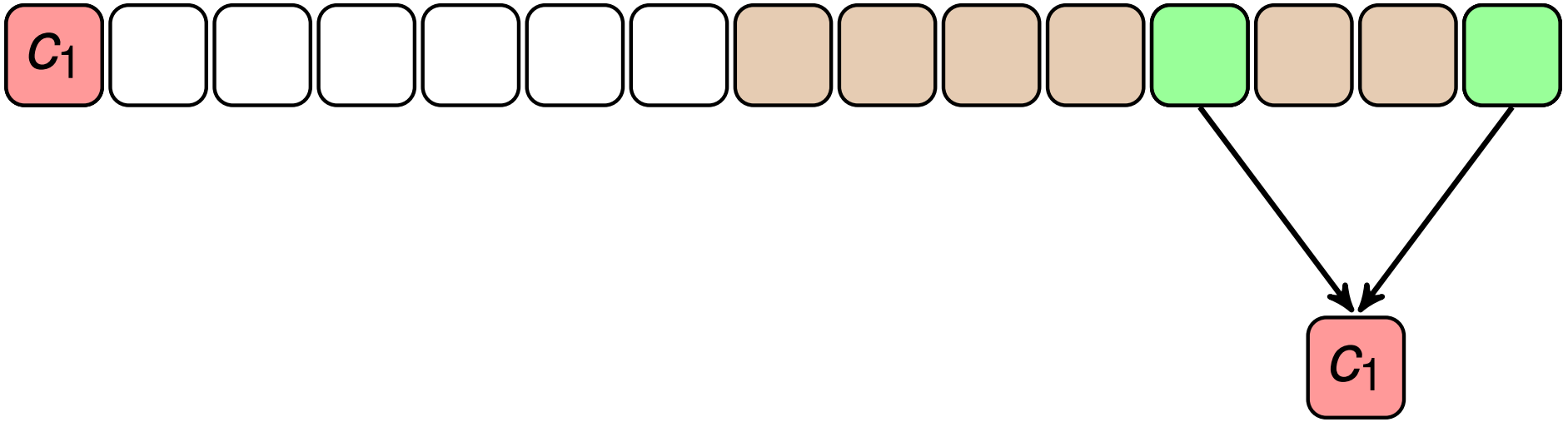
Locality and availability



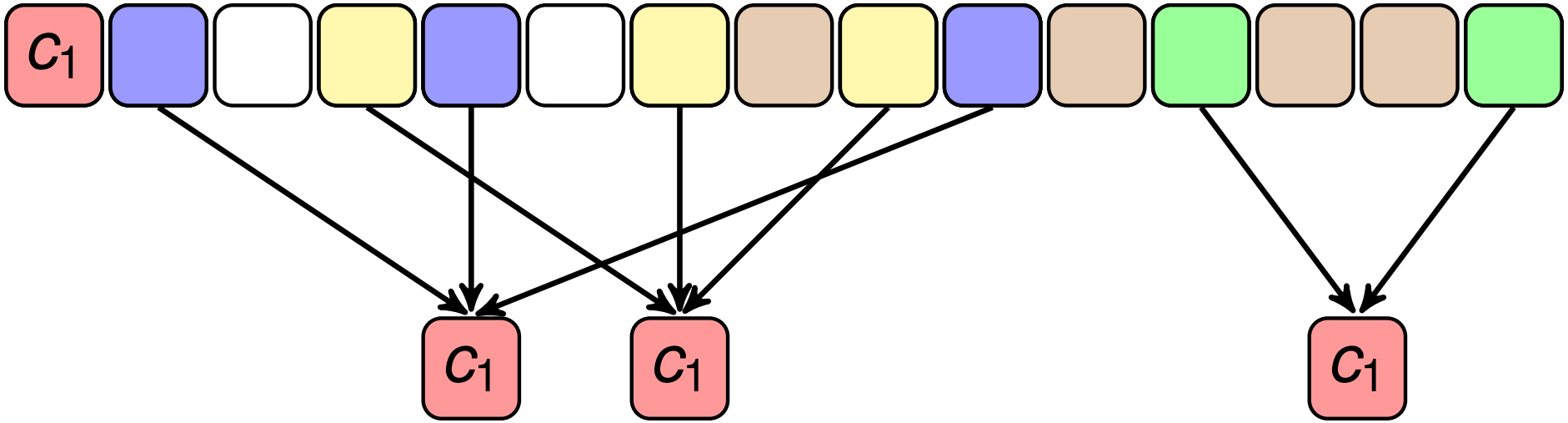
Locality and availability



Locality and availability



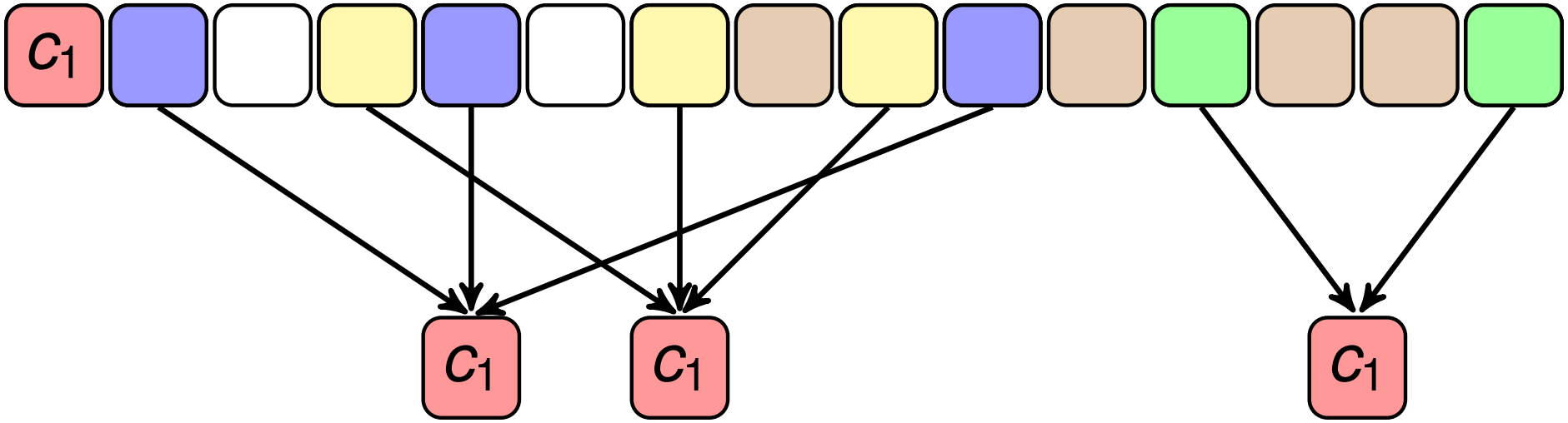
Locality and availability



3 disjoint repair groups $\Rightarrow$ 3-availability.

A. Rawat, D. Papailiopoulos, A. Dimakis, and S. Vishwanath, in Allerton 2013.

Locality and availability

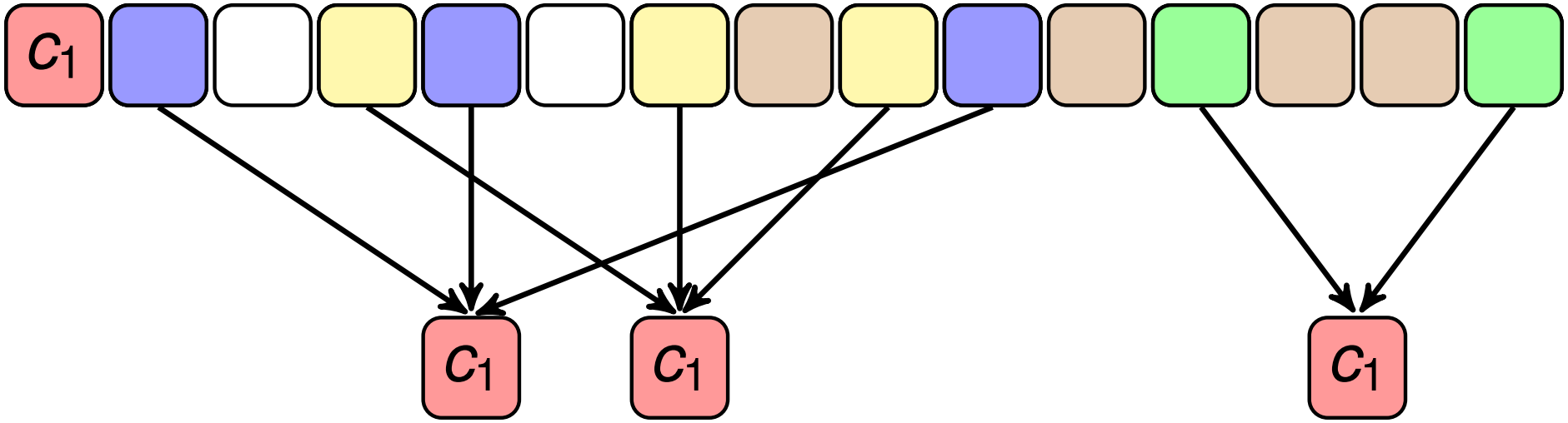


3 disjoint repair groups $\Rightarrow$ 3-availability.

- In general, t availability with r locality.

A. Rawat, D. Papailiopoulos, A. Dimakis, and S. Vishwanath, in Allerton 2013.

Locality and availability



3 disjoint repair groups $\Rightarrow$ 3-availability.

- In general, t availability with r locality.
- **Open question:** how to address general t request patterns.
  - t requests for c_1.
  - $t/2$ requests for c_1, and $t/2$ requests for c_2.
  - $t/3$ requests for each of c_1, c_2 and c_3.

A. Rawat, D. Papailiopoulos, A. Dimakis, and S. Vishwanath, in Allerton 2013.

Thank you!