A Transpositional Redundant Data Update Algorithm for Growing Server-less Video Streaming Systems

T. K. Ho and Jack Y. B. Lee
Department of Information Engineering
The Chinese University of Hong Kong

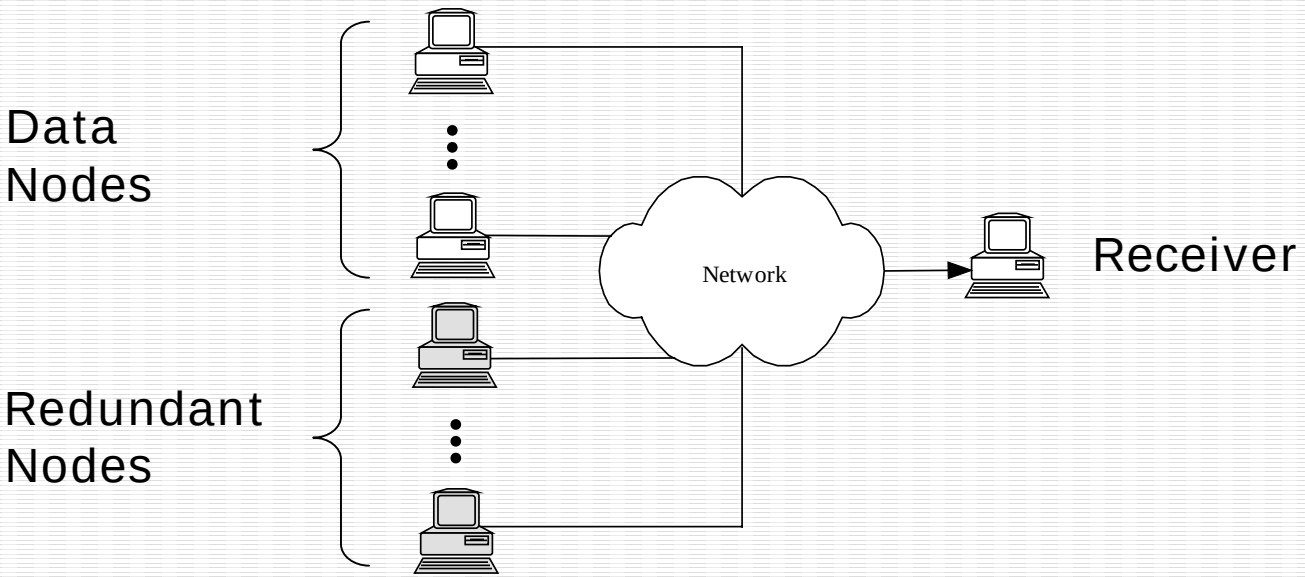
Presented by T.K. Ho
P2P 2003, 1-3 Sep, Linköping, Sweden

Outline

- Background
- Previous Works
- Transpositional Redundant Data Update (TRDU)
- Multiple Redundant Nodes
- Performance Evaluation
- Conclusion and Future Works

Background

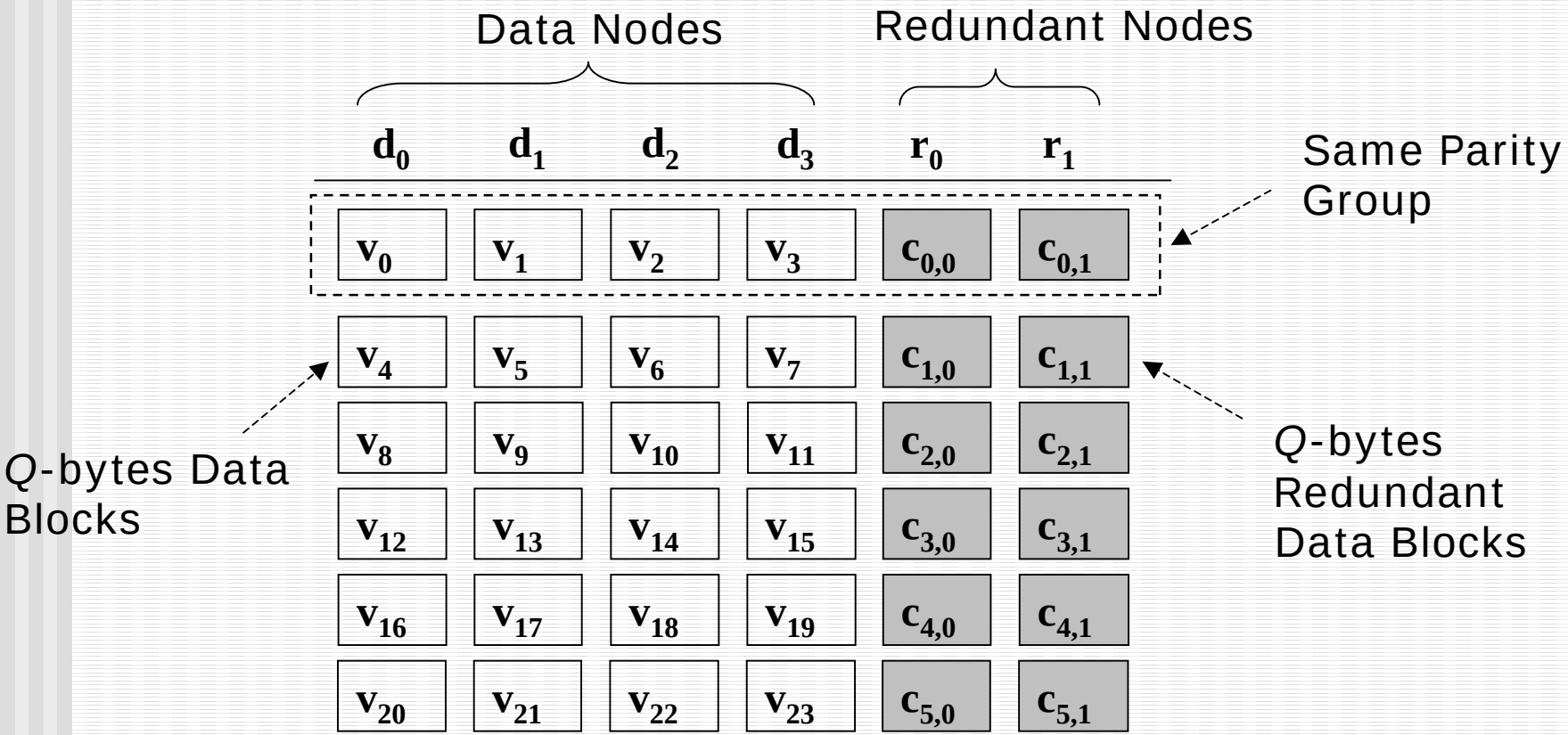
- Server-less Video Streaming Systems[1]
  - Pool of user hosts, or call nodes
  - Each node acts as both sender and receiver
  - Data redundancy to achieve reliability



[1] Jack Y. B. Lee and W. T. Leung, “Study of a Server-less Architecture for Video-on-Demand Applications,” *Proc. IEEE International Conference on Multimedia and Expo.*, August 2002.

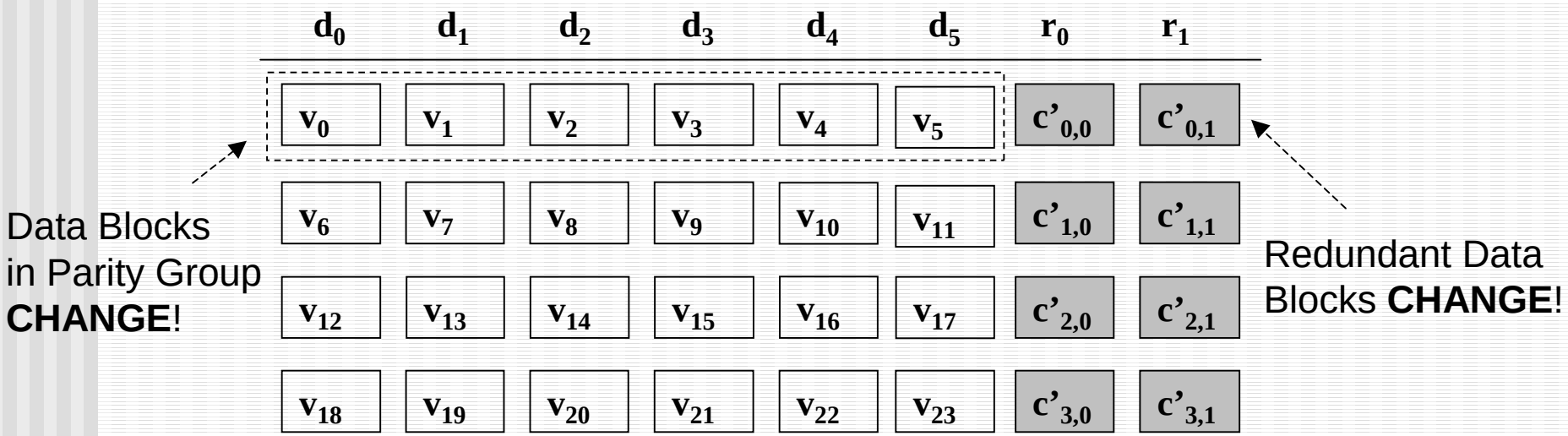
Background

- Data Placement Policy



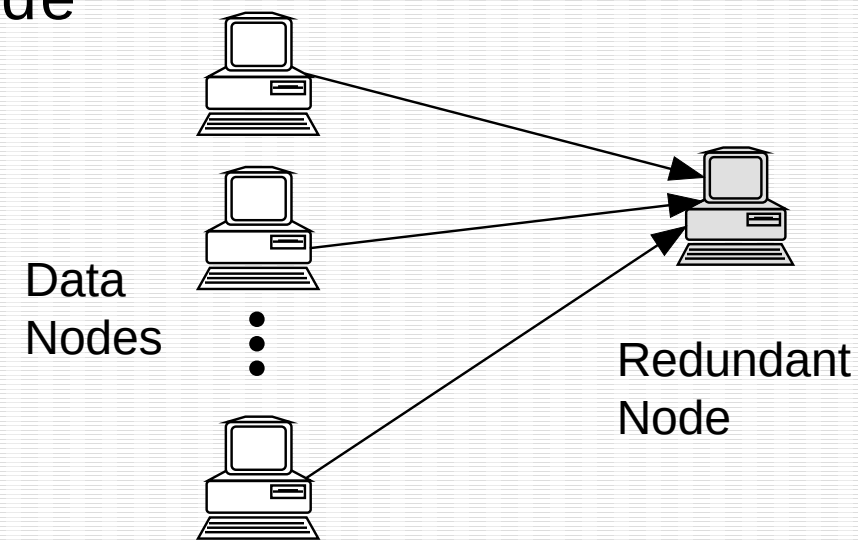
Background

- New nodes join the system
- Video data blocks are reorganized to utilize their storage and streaming capacity
- Video data blocks in the parity group change
- Redundant data blocks need updating
- Transmitting video data blocks to redundant nodes



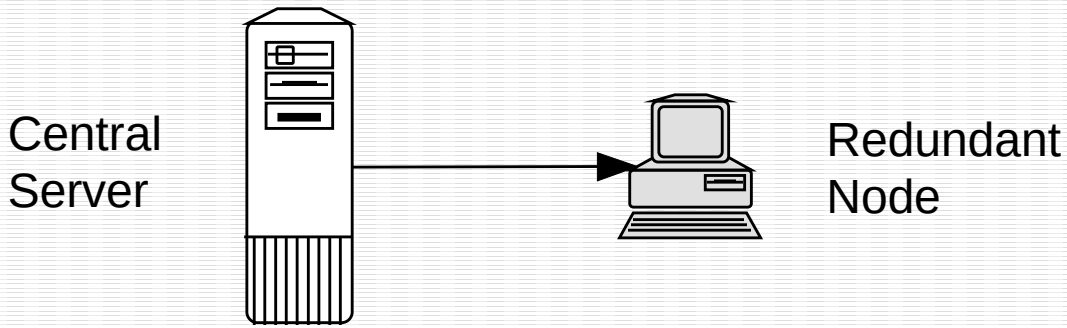
Previous Works

- Redundant Data Regeneration
  - Data blocks are fully distributed
    - Transmit all data blocks to redundant node
    - Compute the new redundant data in redundant node
    - Overhead is too high



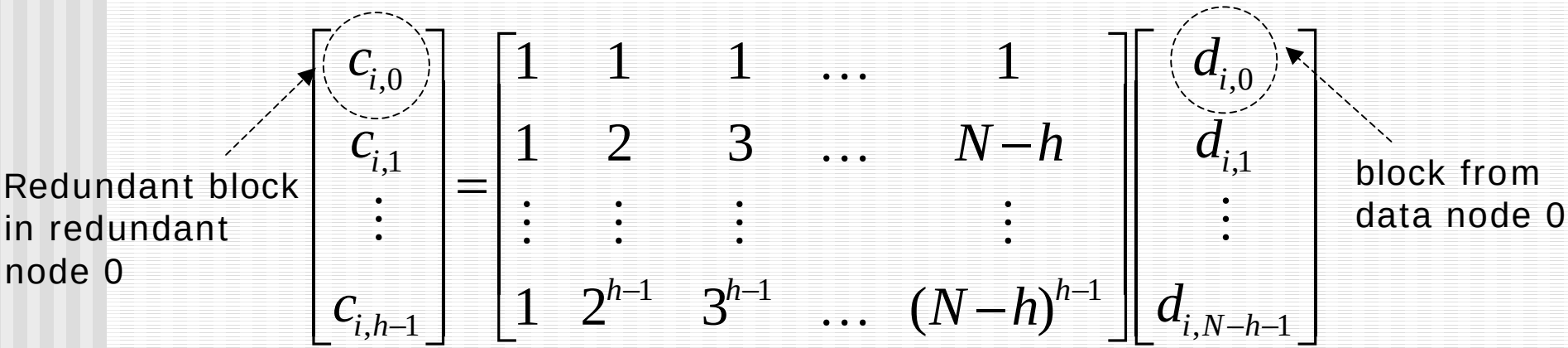
Previous Works

- Redundant Data Regeneration
  - All data blocks stored in central server
    - Regenerate new redundant data in server
    - Transmit them to redundant node
    - Overhead is low
    - Costly for server, not desirable/ feasible



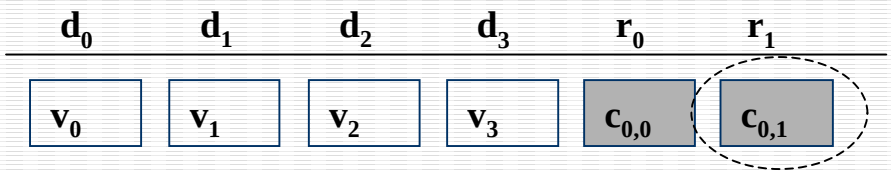
Previous Works

- Sequential Redundant Data Update (SRDU)
  - Old redundant blocks contain valuable information
  - Possible to reuse for linear code, e.g. RSE code
  - Vandermonde matrix
    - Number of nodes, *N*
    - Number of redundant nodes, *h*
    - Element at row *i*, column *j* = j^{i-1}



Previous Works

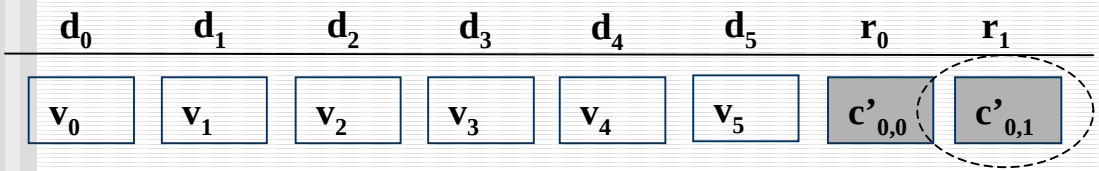
- Updating parity group 0
  - Before addition of nodes



$$\begin{bmatrix} c_{0,0} \\ c_{0,1} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \end{bmatrix} \begin{bmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

$$c_{0,1} = 1*v_0 + 2*v_1 + 3*v_2 + 4*v_3$$

  - After addition of two nodes



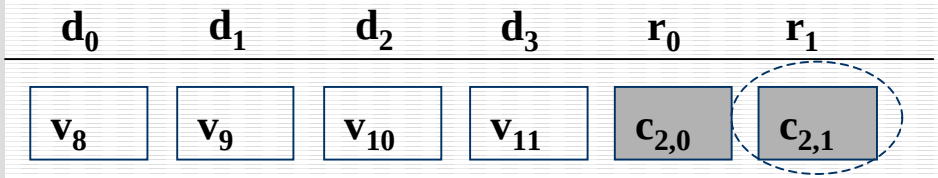
$$\begin{bmatrix} c'_{0,0} \\ c'_{0,1} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 \end{bmatrix} \begin{bmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{bmatrix}$$

$$c'_{0,1} = c_{0,1} + 5*v_4 + 6*v_5$$

Only need to send v_4, v_5

Previous Works

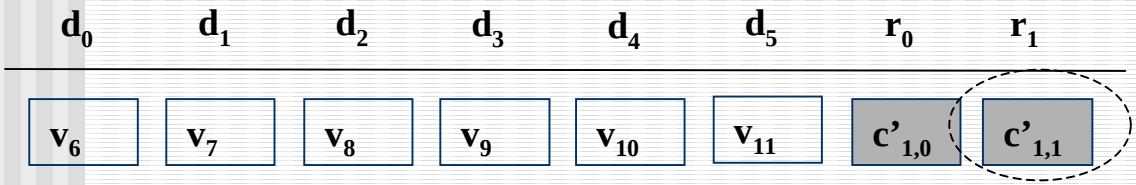
- Updating parity group 1
  - Before addition of nodes



$$\begin{bmatrix} c_{2,0} \\ c_{2,1} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \end{bmatrix} \begin{bmatrix} v_8 \\ v_9 \\ v_{10} \\ v_{11} \end{bmatrix}$$

$$c_{2,1} = 1*v_8 + 2*v_9 + 3*v_{10} + 4*v_{11}$$

  - After addition of two nodes



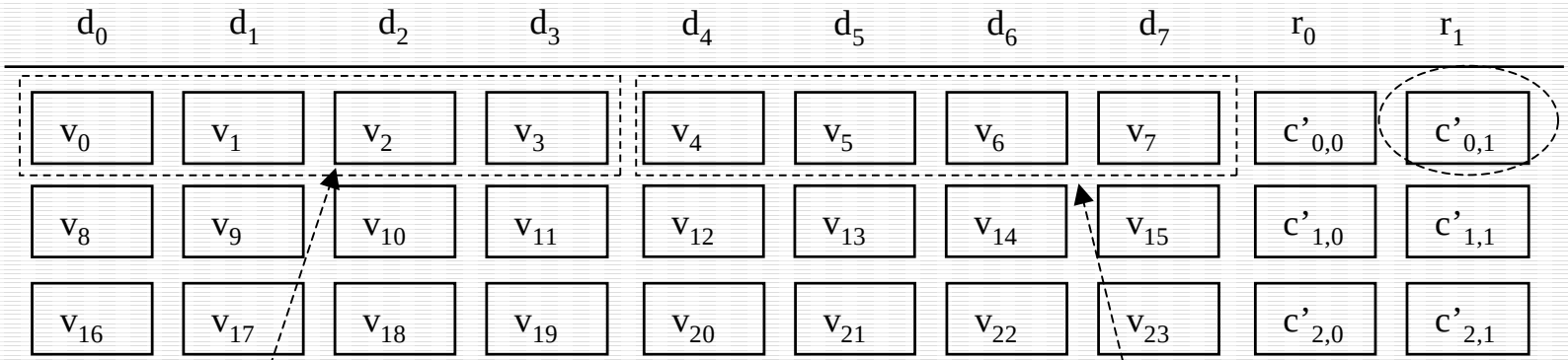
$$\begin{bmatrix} c'_{1,0} \\ c'_{1,1} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 \end{bmatrix} \begin{bmatrix} v_8 \\ v_9 \\ v_{10} \\ v_{11} \\ v_6 \\ v_7 \end{bmatrix}$$

$$c'_{1,1} = c_{2,1} + 5*v_6 + 6*v_7$$

Only need to send v_6, v_7

Previous Works

- Problem in SRDU
  - Reusing of old redundant block is limited
  - $c'_{0,1}$ can only be in term of one of $c_{0,1}$ and $c_{1,1}$
  - Both $c_{0,1}$ and $c_{1,1}$ are constructed by multiplying 1 to 4 with the data blocks



$$c'_{0,1} = c_{0,1} + \sum_{j=4}^{7} (j+1) * v_j \qquad c'_{0,1} = c_{1,1} + \sum_{j=0}^{3} (j+5) * v_j$$

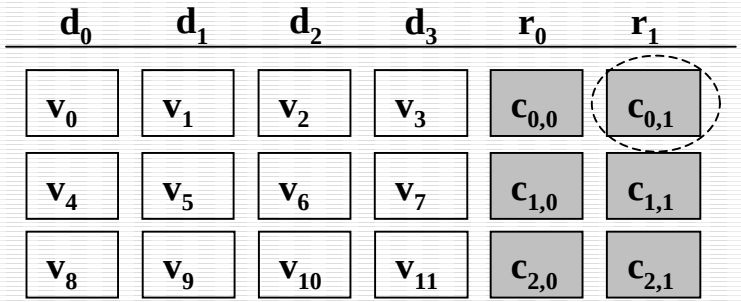
Transpositional Redundant Data Update

- Compute redundant block using transposed coefficients
- Vandermonde matrix
  - (N_{max}-h) columns
  - N_{max} is max system size, irrespective of current size N

$$\begin{bmatrix} c_{i,0} \\ c_{i,1} \\ \vdots \\ c_{i,h-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & 2 & 3 & \dots & N_{\max}-h \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & 2^{h-1} & 3^{h-1} & \dots & (N_{\max}-h)^{h-1} \end{bmatrix} \begin{bmatrix} d_{i,0} \\ d_{i,1} \\ \vdots \\ d_{i,N_{\max}-h-1} \end{bmatrix}$$

Transpositional Redundant Data Update

- For parity group *i*, the starting position is shifted by $(i(N\text{-}h) \bmod (N_{max}\text{-}h))$
- All other elements are zeroed
- Example
  - Let $(N_{max}\text{-}h)$ be 10
  - Parity group 0

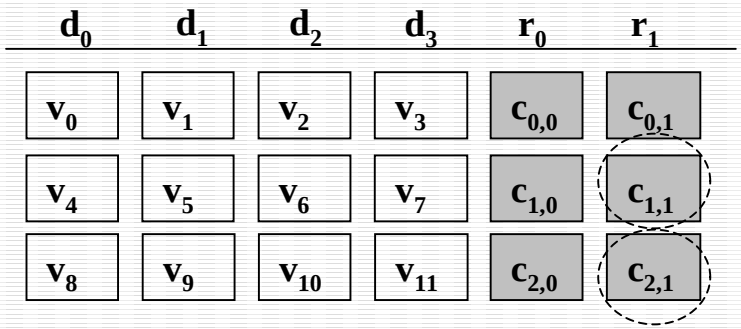


$$\begin{bmatrix} c_{0,0} \\ c_{0,1} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & 2 & 3 & 4 & 5 & \cdots & 10 \end{bmatrix} \begin{bmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$c_{0,1} = 1*v_0 + 2*v_1 + 3*v_2 + 4*v_3$$

Transpositional Redundant Data Update

- Parity group 1 and 2



$$c_{1,1} = 5*v_4 + 6*v_5 + 7*v_6 + 8*v_7$$

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ v_4 \\ \vdots \\ v_7 \\ 0 \\ 0 \end{bmatrix}$$

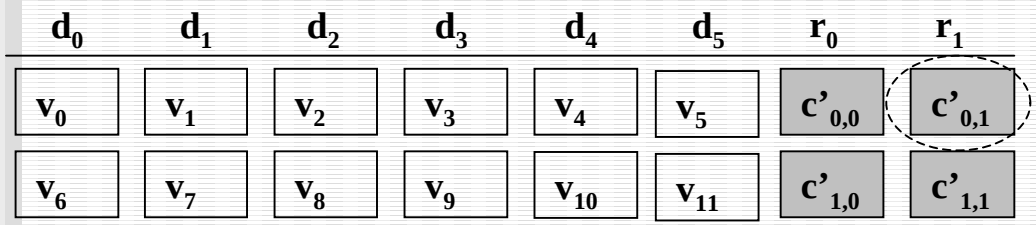
$$c_{2,1} = 9*v_8 + 10*v_9 + 1*v_{10} + 2*v_{11}$$

$$\begin{bmatrix} v_{10} \\ v_{11} \\ 0 \\ \vdots \\ 0 \\ v_8 \\ v_9 \end{bmatrix}$$

- Repeat coefficient in $c_{2,1}$
- Max size of 10 prevents v_0 and v_{10} in residing in the same parity group

Transpositional Redundant Data Update

- Two nodes are added
  - Updating parity group 0



$$c'_{0,1} = 1*v_0 + 2*v_1 + 3*v_2 + 4*v_3 + 5*v_4 + 6*v_5$$

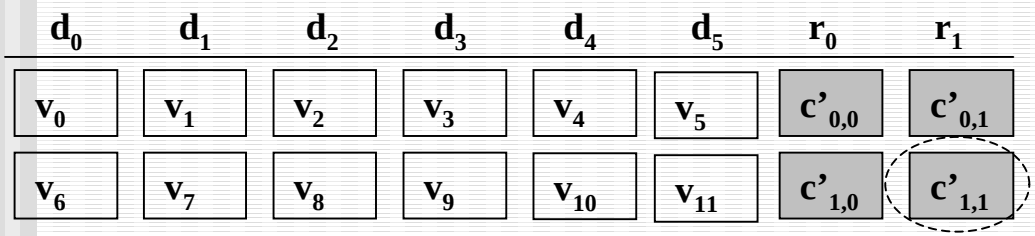
$$= c_{0,1} + 5*v_4 + 6*v_5$$

$$\begin{bmatrix} c'_{0,0} \\ c'_{0,1} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & 2 & 3 & 4 & 5 & \cdots & 10 \end{bmatrix} \begin{bmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Need to send v_4, v_5

Transpositional Redundant Data Update

- Updating parity group 1



$$\begin{bmatrix} c'_{1,0} \\ c'_{1,1} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & 2 & 3 & 4 & 5 & \cdots & 10 \end{bmatrix} \begin{bmatrix} v_{10} \\ v_{11} \\ 0 \\ \vdots \\ 0 \\ v_6 \\ v_7 \\ v_8 \\ v_9 \end{bmatrix}$$

No need to send

$$c'_{1,1} = (7*v_6 + 8*v_7) + (9*v_8 + 10*v_9 + 1*v_{10} + 2*v_{11})$$

$$= (c_{1,1} - 5*v_4 - 6*v_5) + c_{2,1}$$

Cached in previous update

- This equation is not allowed in SRDU, as v_6 shares the same coefficient with v_{10}

Transpositional Redundant Data Update

- In SRDU
  - Before addition of nodes

$$c_{1,1} = 1*v_4 + 2*v_5 + 3*v_6 + 4*v_7$$

$$c_{2,1} = 1*v_8 + 2*v_9 + 3*v_{10} + 4*v_{11}$$

  - After addition of two nodes

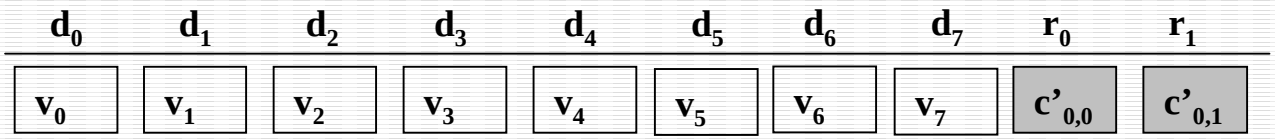
$$\begin{bmatrix} c'_{1,0} \\ c'_{1,1} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 \end{bmatrix} \begin{bmatrix} v_8 \\ v_9 \\ v_{10} \\ v_{11} \\ v_6 \\ v_7 \end{bmatrix}$$

$$c'_{1,1} = (1*v_8 + 2*v_9 + 3*v_{10} + 4*v_{11}) + (5*v_6 + 6*v_7)$$

$$= c_{2,1} + 5*v_6 + 6*v_7$$

Transpositional Redundant Data Update

- Zero overhead point
  - Number of additional nodes equals integral multiples of original system size

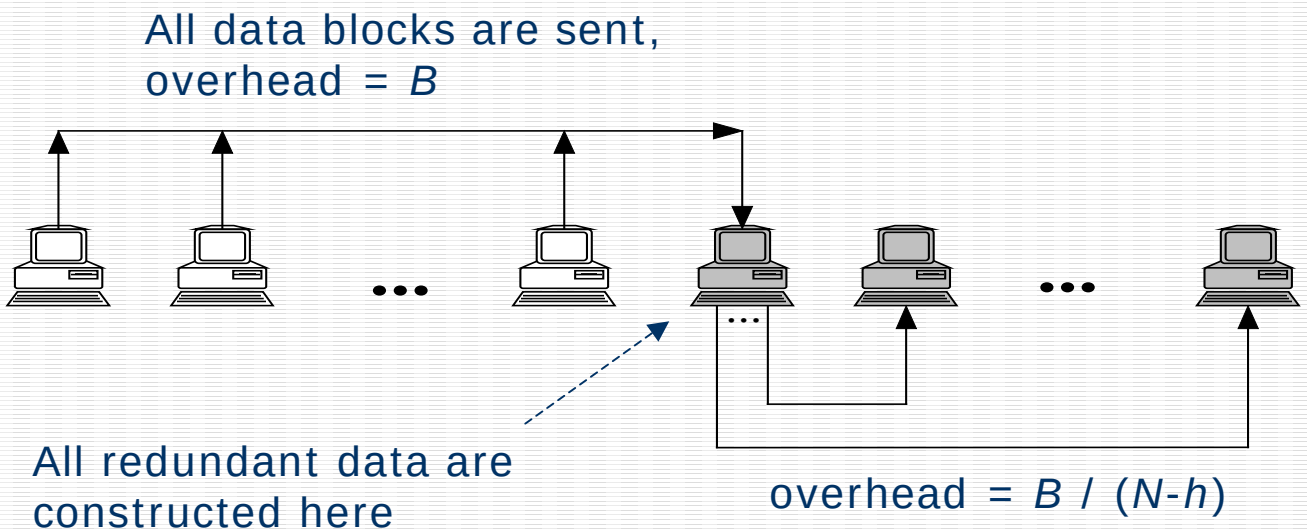


$$c'_{0,1} = c_{0,1} + c_{1,1}$$

- Tradeoffs
  - Specify the max system size N_{max}
  - Larger Vandermonde matrix increases the computational complexity in decoding

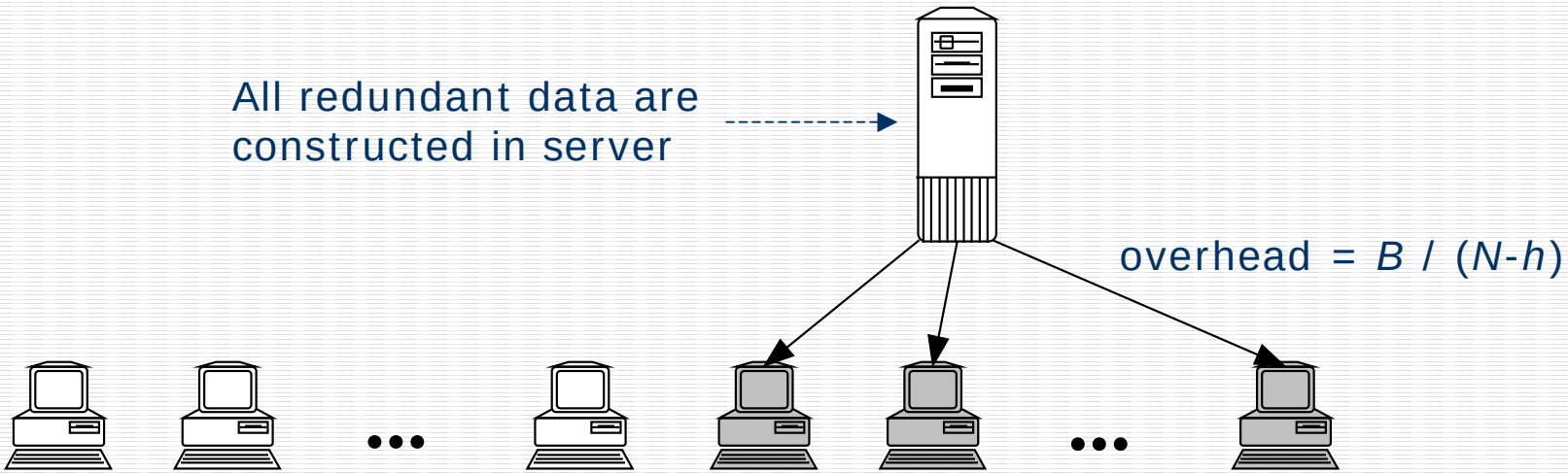
Multiple Redundant Nodes

- Total number of data blocks, *B*
- Redundant Data Regeneration
  - Fully distributed data blocks
    - Total overhead = $B + (h\text{-}1)(B/(N\text{-}h))$



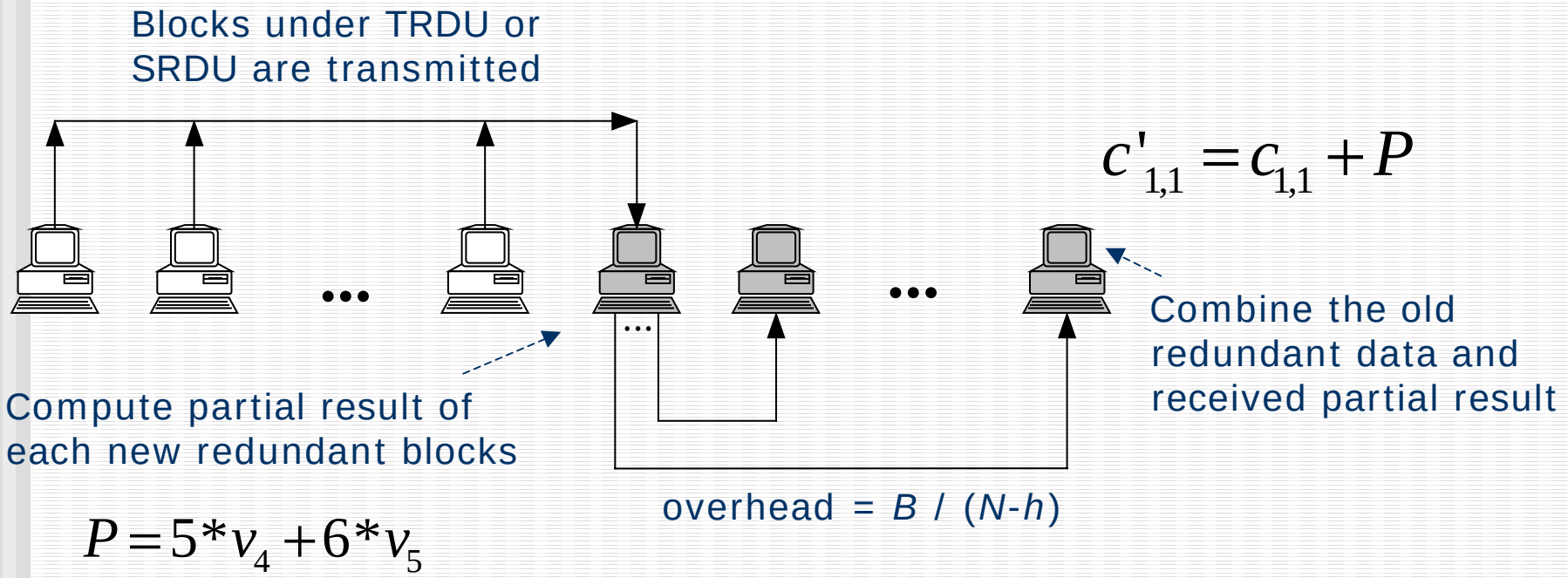
Multiple Redundant Nodes

- Data blocks are stored in central server
  - Total overhead = $h * B/(N-h)$



Multiple Redundant Nodes

- SRDU/TRDU
  - Total overhead = blocks under SRDU/TRDU + (*h*-1)(*B*/(*N*-*h*))



Multiple Redundant Nodes

Redundant Data Regeneration	$B+(h-1)(B/(N-h))$
Regeneration by archive server	$B/(N-h)+$ $(h-1)(B/(N-h))$
Sequential Redundant Data Update (SRDU)	Block movement under SRDU+ $(h-1)(B/(N-h))$
Transpositional Redundant Data Update (TRDU)	Block movement under TRDU+ $(h-1)(B/(N-h))$

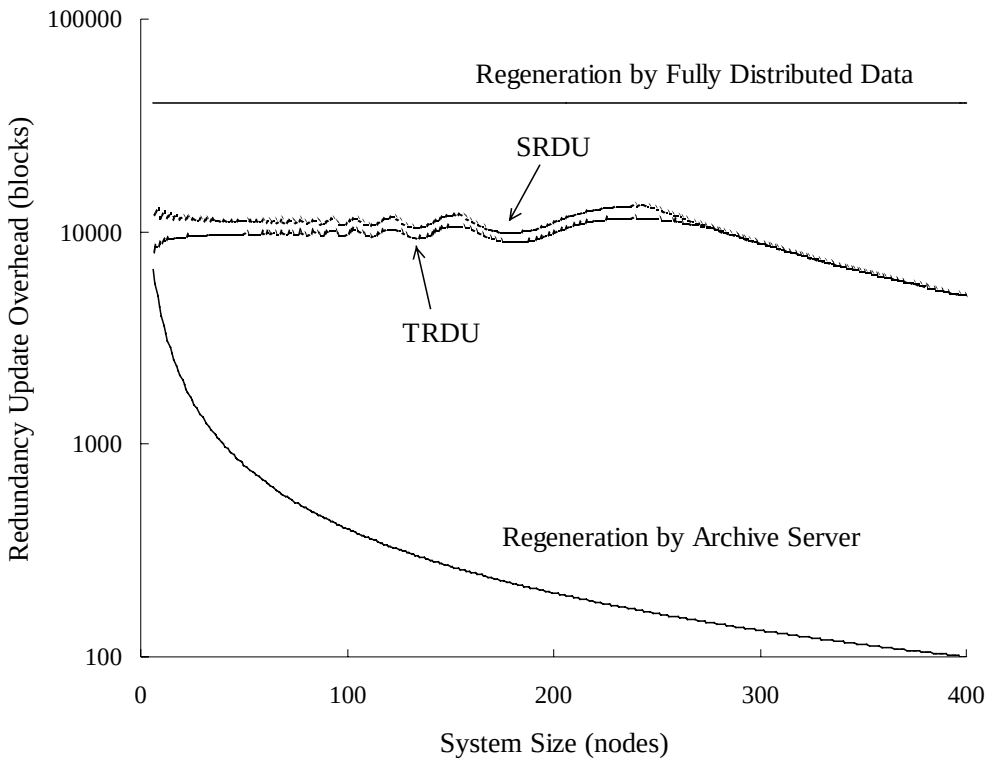
- Overhead is dominated by updating the first redundant node

Performance Evaluation

- Simulation Details
  - One redundant node
  - One video title
  - 40,000 data blocks
- Metric
  - Redundancy Update Overhead
    - Number of video data blocks needed to transmit for generating the new redundant data

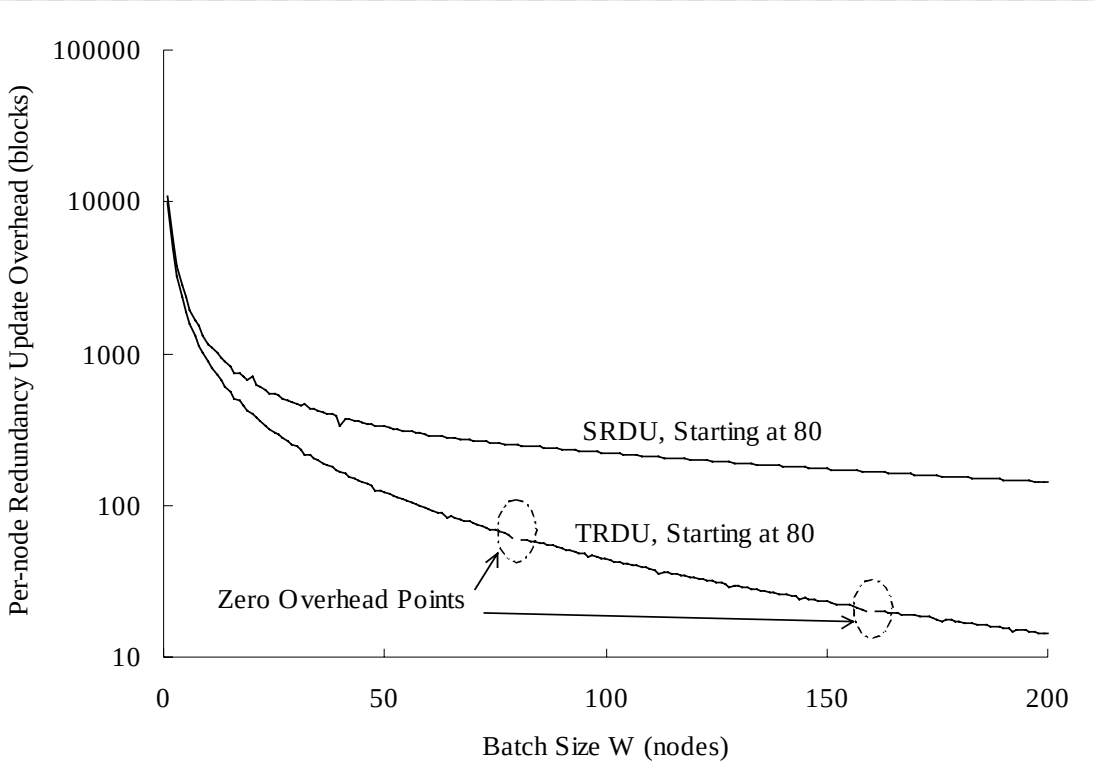
Performance Evaluation

- Continuous system growth
  - Grow from 1 node to 400 nodes
  - One node added each time



Performance Evaluation

- Batched redundancy update
  - Defer update until a fixed number of nodes, *W*
  - Initial system size = 80, *W* ranges from 1 to 200



Conclusion

- Reduce the redundancy update overhead significantly by
  - TRDU
  - Batched update
- Overhead in multiple redundant nodes is investigated
  - Additional overhead is insignificant

Future Works

- Node Removal
- Addition of redundant nodes
- Scheduling problem
- Optimal batch size

Thank you

- Q&A Section