# Communication Driven Remapping of Processing Element (PE) in Fault-tolerant NoC-based MPSoCs

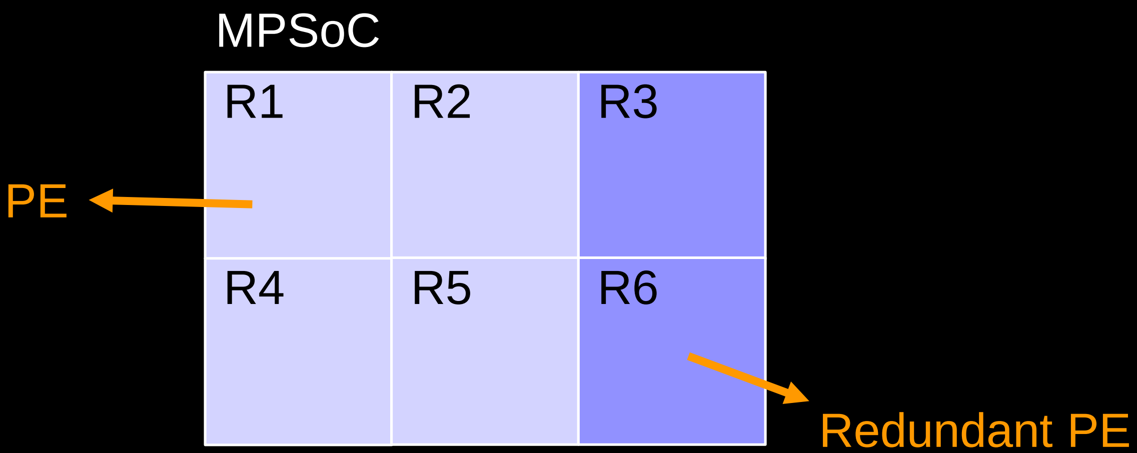
Chia-Ling Chen, Yen-Hao Chen and TingTing Hwang
Department of Computer Science
National Tsing Hua University, Taiwan

# Outline

- **Multi-Processor System-on-Chip (MPSoC)**
  - Task remapping for fault-tolerant
- **Communication driven remapping method**
  - Model communication cost on edges
  - Allow tasks be moved to non-neighboring PEs
- **Initial mapping improvement**
- **Experimental results**
- **Conclusions**

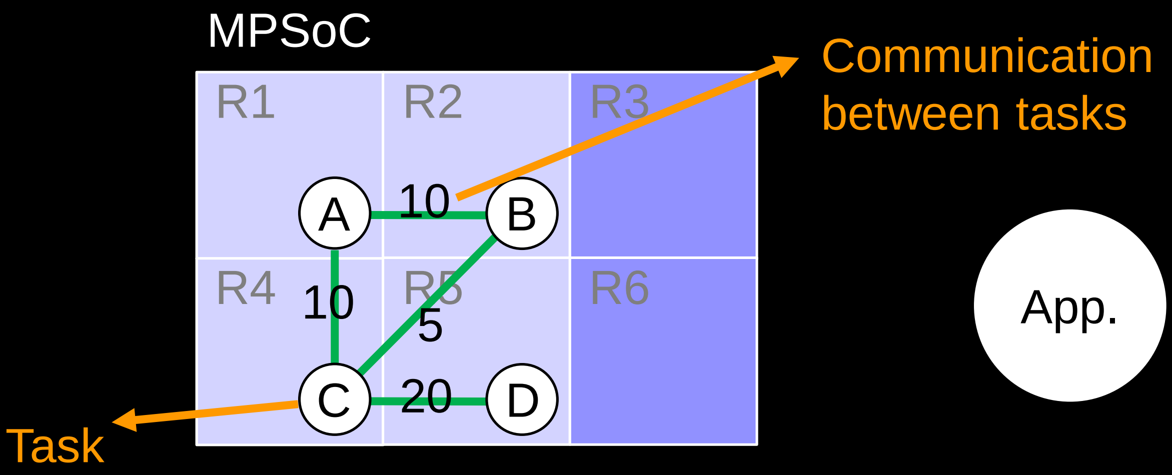
# Multi-Processor System-on-Chip

- **Processing elements (PEs) integrated on a chip**
  - Each PE can execute a task independently
  - Redundant PEs are included for fault tolerance



# Task Graph

- **Application is partitioned into several tasks**
  - Task (node) – basic executable unit
    - Must be mapped to a good PE for execution
  - Edge – communication overhead
    - The amount of data (data size) transferred between tasks
    - A shorter edge (transferred distance) is preferred
      - For lower communication overhead

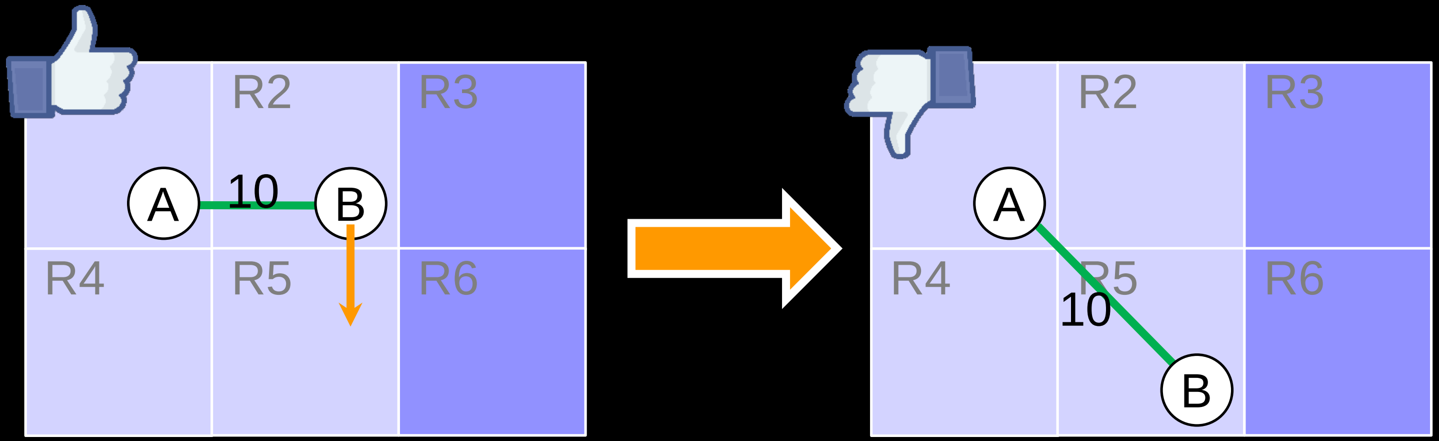


Communication between tasks

Task

# Task Mapping

- **Process of deciding PE of each task for execution**
- **Mapping with shorter transferred distance**
  - Less communication overhead
    - Improve system performance
    - Decrease energy consumption

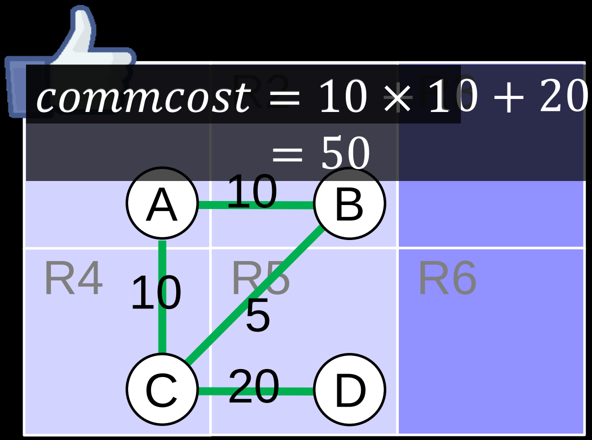


Transferred distance = 1

Transferred distance = 2

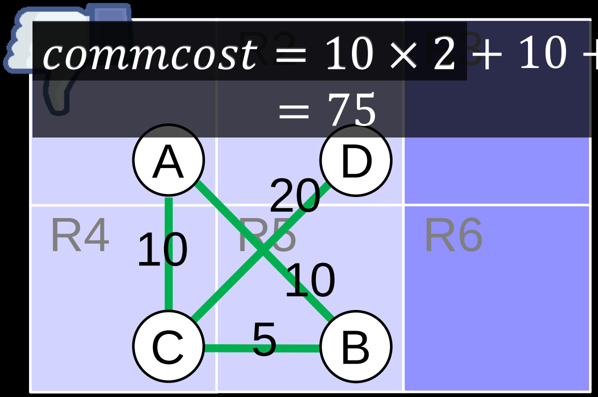
# Communication Cost Metric

- $commcost = \sum_{i \in transferred\_data} size_i \times distance_i$
  - Size – transferred data size
  - Distance – transferred distance



$commcost = 10 \times 10 + 20 + 5 \times 2 = 50$

Transferred distance = 1

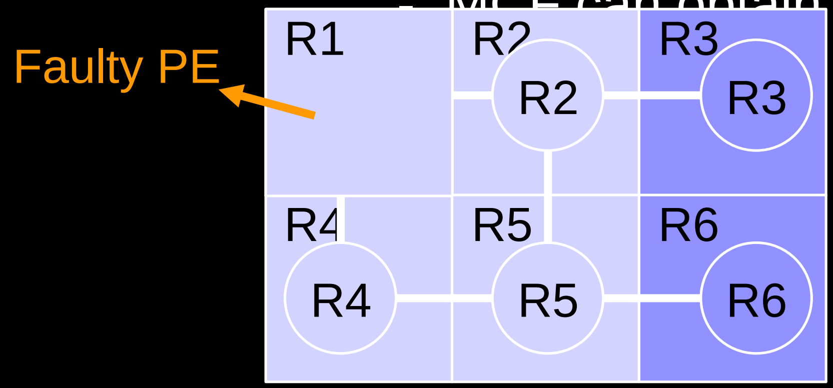


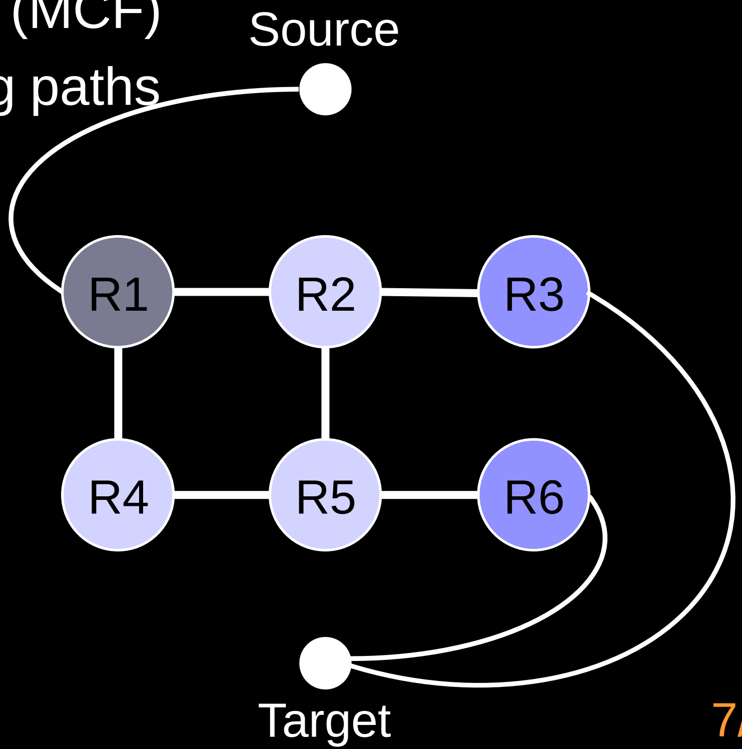
$commcost = 10 \times 2 + 10 + 20 \times 2 + 5 = 75$

Transferred distance = 2

# Task Remapping for Fault-tolerant

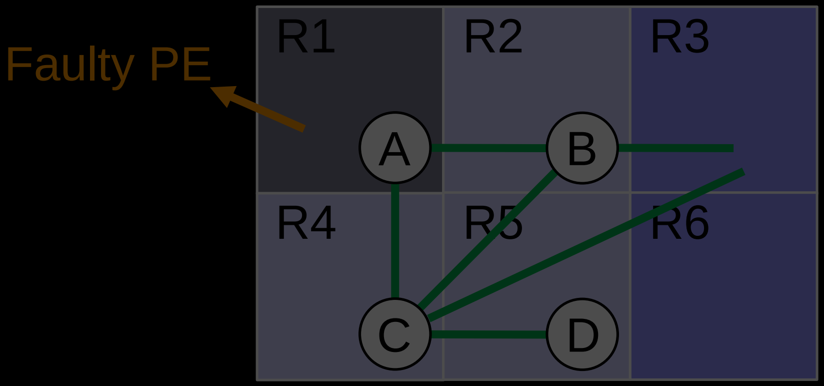
- **Remap all tasks again when faulty PE detected**
  - Topology graph construction
    - Repairing path
      - Each path from source to target is a remapping
    - Minimum-cost flow algorithm (MCF)
      - MCF can obtain repairing paths



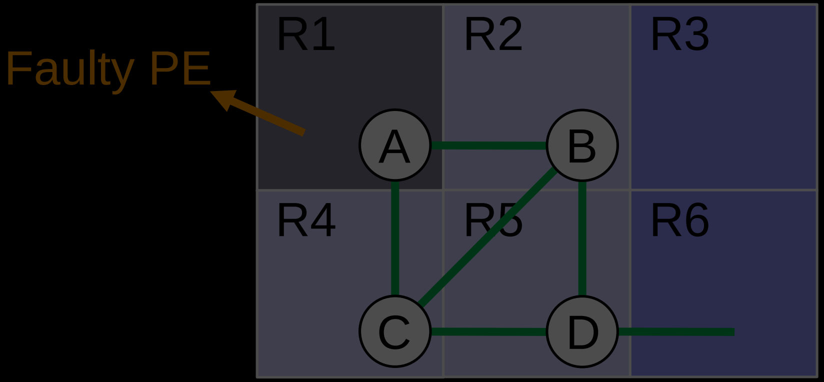


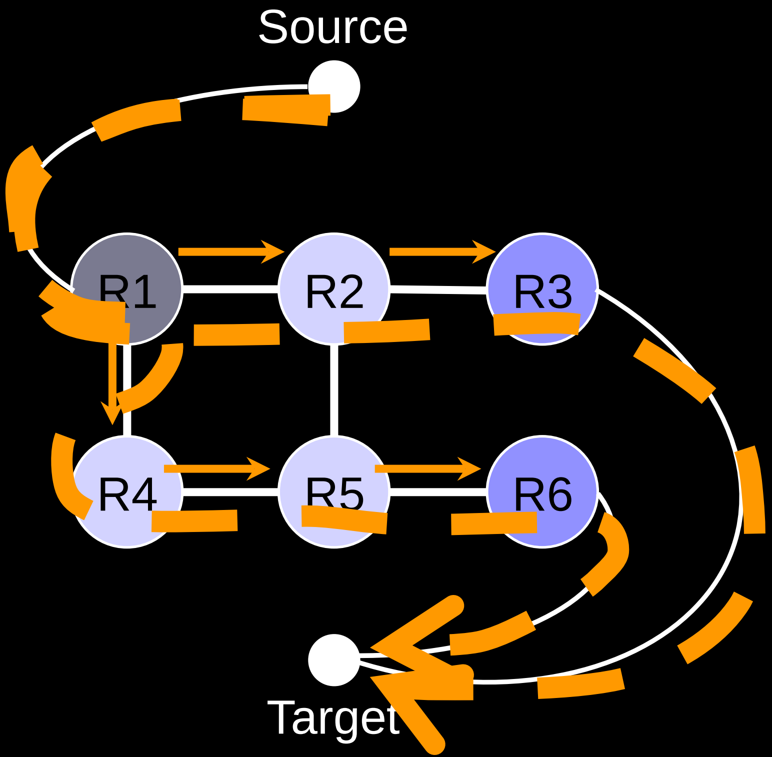
# Repairing Paths

- **A repairing path represents a remapping solution**
  - E.g. R1->R2->R3



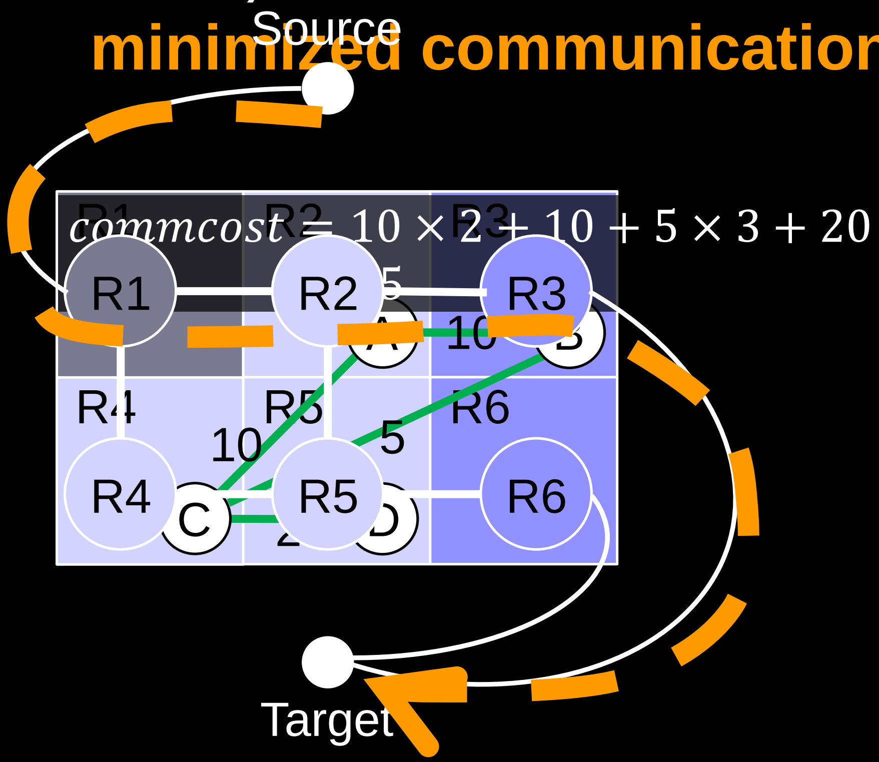
  - E.g. R1->R4->R5->R6



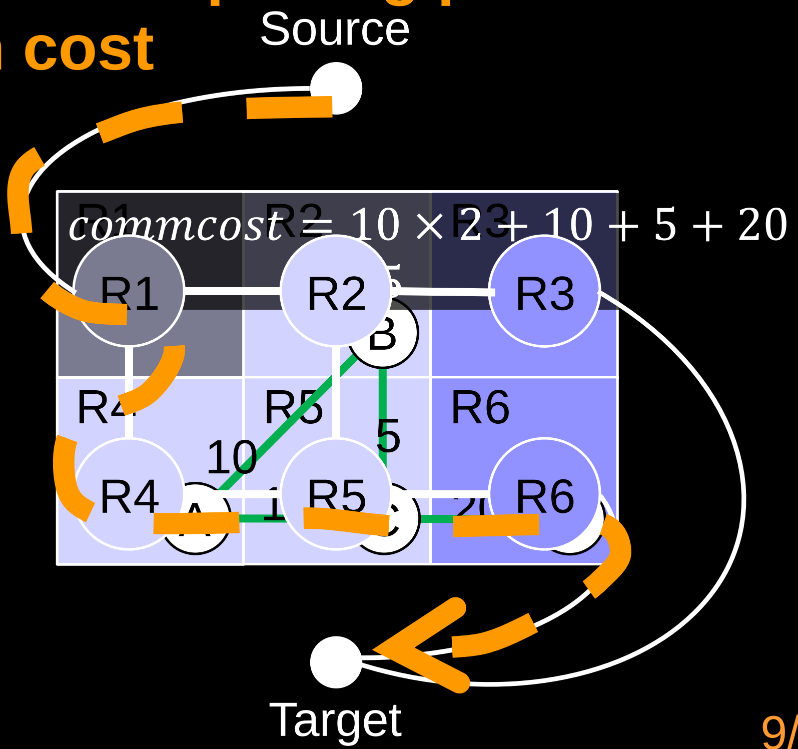


# Problem Description

- **Each repairing path indicates a valid remapping**
  - All can successfully remap all tasks to good PEs
- **But, the remappings have different $\boldsymbol{commcost}$**
- **Thus, we would like to find the repairing path with minimized communication cost**



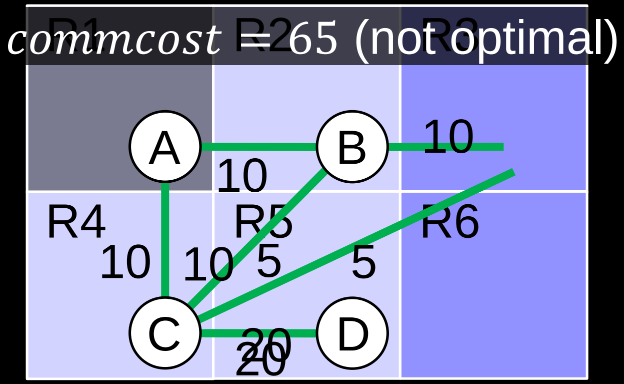
$commcost = 10 \times 2 + 10 + 5 \times 3 + 20$



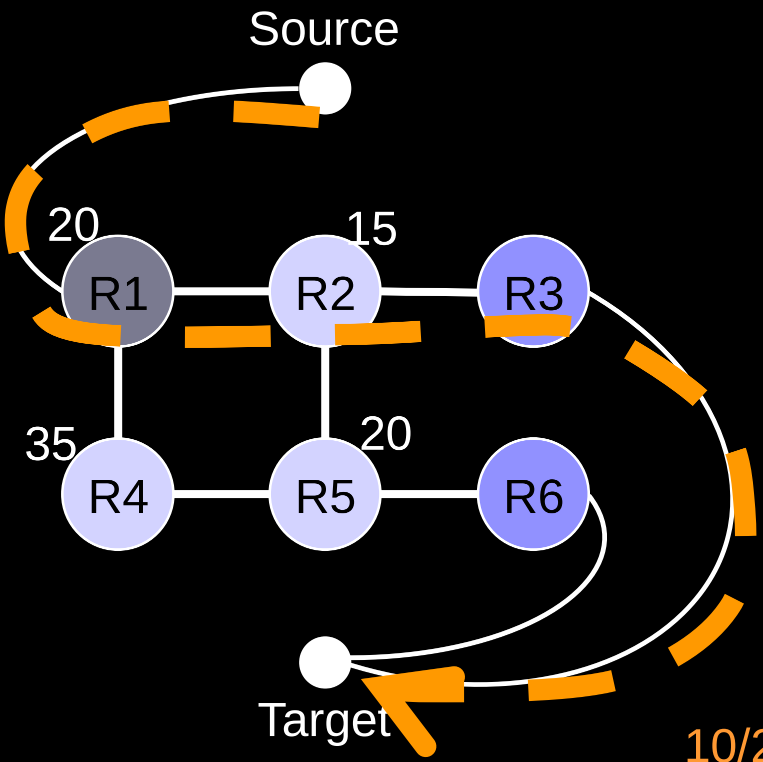
$commcost = 10 \times 2 + 10 + 5 + 20$

# Previous Work [1]

- **The communication cost is modeled on nodes**
  - Not precise enough to represent $commcost$
- **Tasks can be only moved to neighboring PEs**
  - Guarantee $\Delta distance = 1$
  - Reduce repair rate



$commcost = 65$ (not optimal)



# Outline

- **Multi-Processor System-on-Chip (MPSoC)**
  - Task remapping for fault-tolerant
- **Communication driven remapping method**
  - Model communication cost on edges
  - Allow tasks be moved to non-neighboring PEs
- **Initial mapping improvement**
- **Experimental results**
- **Conclusions**

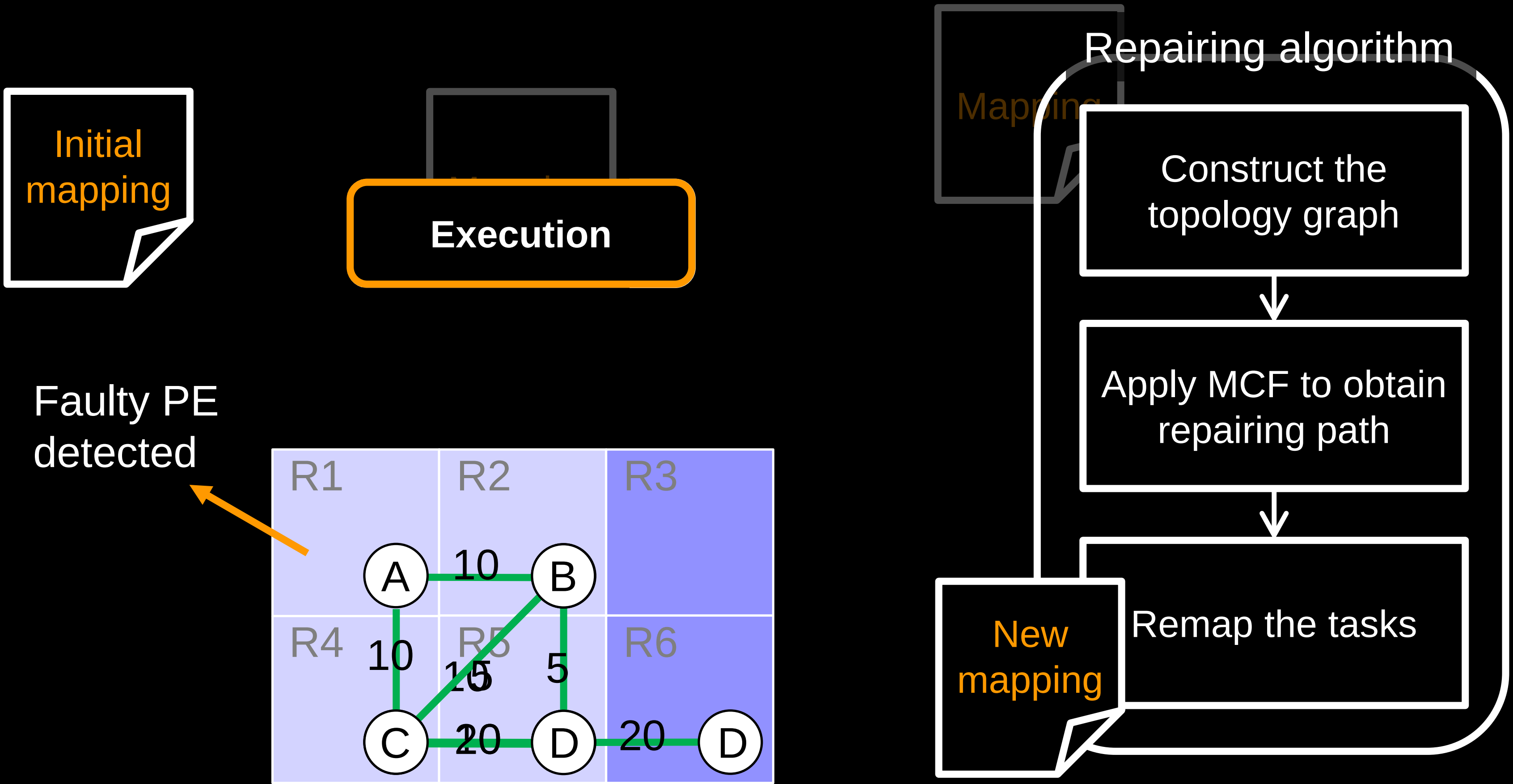
# Features of Our Method

- **Model communication cost on edges**
  - Precisely model the communication overhead

- **Allow task be moved to non-neighboring PEs**
  - Greatly increase the repairing flexibility
  - Achieve 100% repair rate across all test cases

# Flow Diagram of Task Remapping

- **Repairing is applied when detecting faulty PE**



Initial mapping

**Execution**

Mapping

Faulty PE detected

Repairing algorithm

Construct the topology graph

Apply MCF to obtain repairing path

Remap the tasks

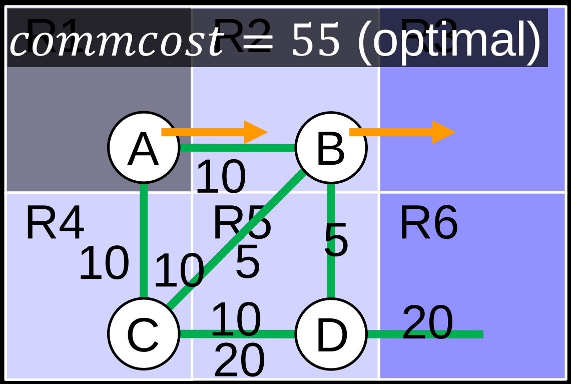
New mapping

# Communication Cost Model on Edges

- **Increment in $commcost$ when task remapped to PE**
  - $\Delta commcost(\mathrm{R} \to \mathrm{R'}) =$

$$\left(\sum_{lenghten\ communication} size - \sum_{shorten\ communication} size\right) \times \Delta distance$$

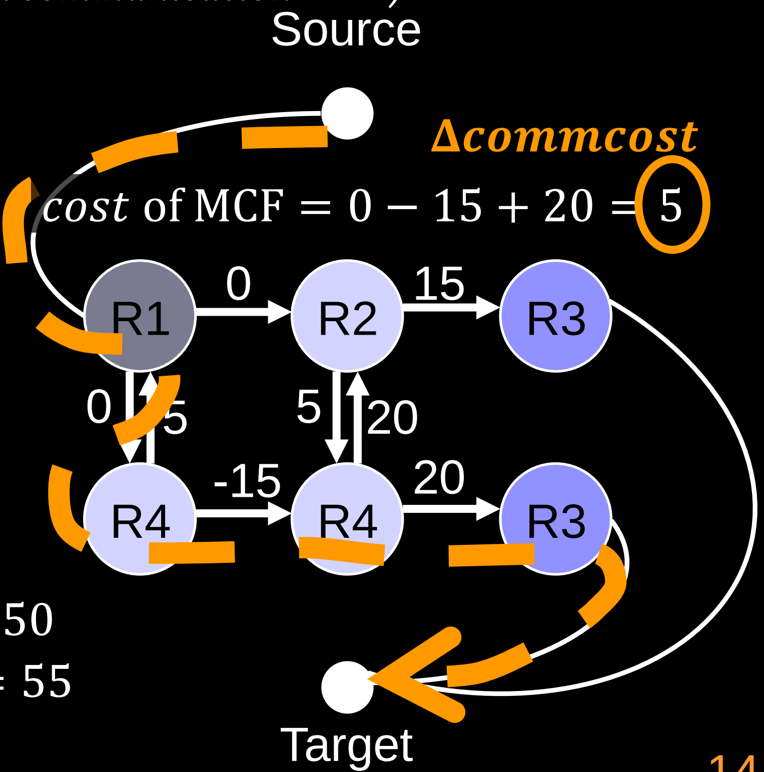
$$\Delta commcost(\mathrm{R1} \to \mathrm{R2}) = (10 + 5) \times 1$$
$$= 15$$



$commcost = 55$ (optimal)

$$commcost_{ori} = 1 * 10 + 1 * 10 + 1 * 20 + 2 * 5 = 50$$
$$commcost_{new} = 2 * 10 + 1 * 10 + 1 * 20 + 1 * 5 = 55$$

Correct $\Delta commcost$ is $55 - 50 = \boxed{5}$

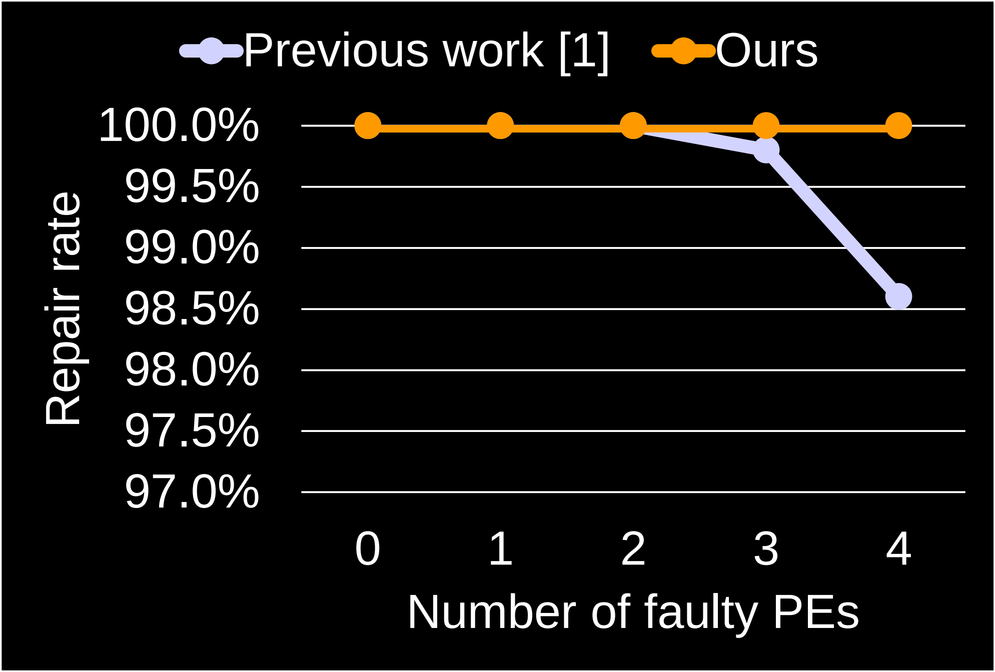


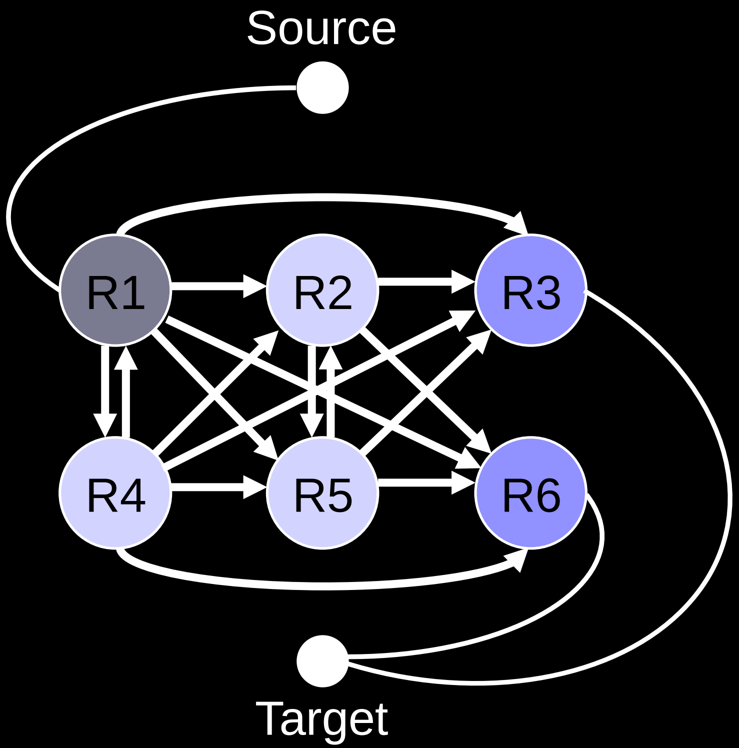
$\Delta commcost$

$cost\ \text{of MCF} = 0 - 15 + 20 = \boxed{5}$

# Tasks moved to Non-neighboring PEs

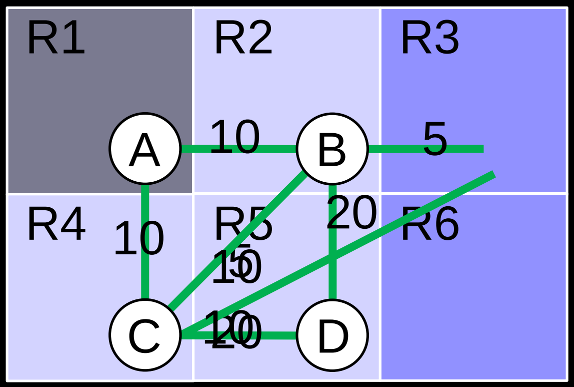
- **Explore larger solution space**
- **Achieve higher repair rate**
  - 100% repair rate for all experimental cases

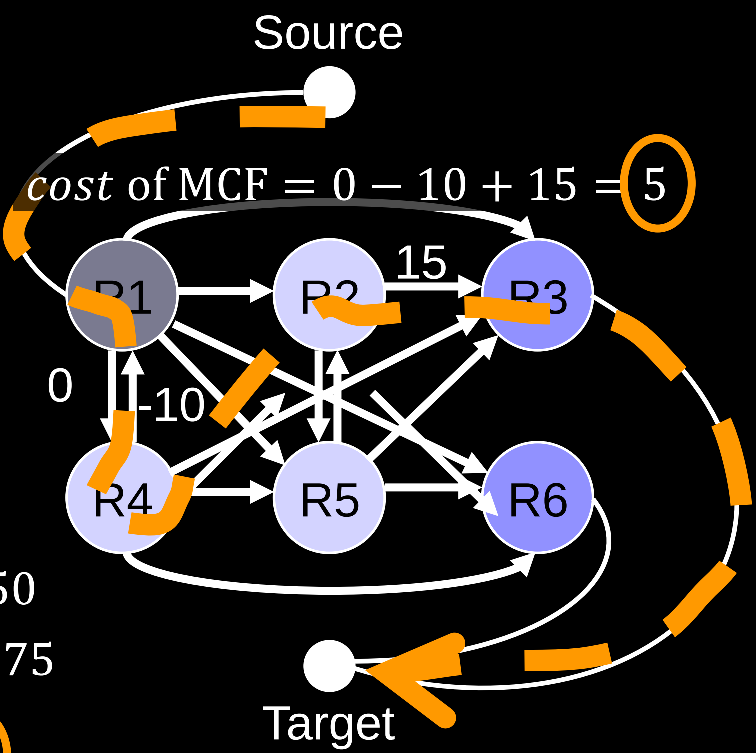




# Problem on Down-and-up Paths

- **The exact cost of some repairing paths can’t be calculated by the MCF algorithm**
  - With down-and-up (back-and-forth) movement





$cost$ of MCF $= 0 - 10 + 15 = \boxed{5}$

$commcost_{ori} = 1*10 + 1*10 + 1*20 + 2*5 = 50$

$commcost_{new} = 2*10 + 3*10 + 1*20 + 1*5 = 75$

Correct $\Delta commcost$ should be $75 - 50 = \boxed{25}$

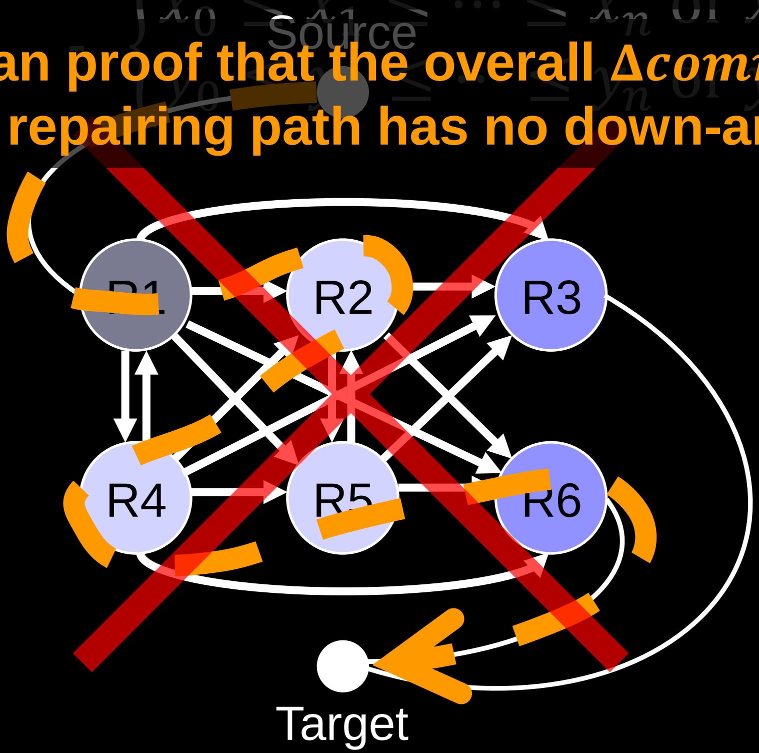
# Conditions on Repairing Path

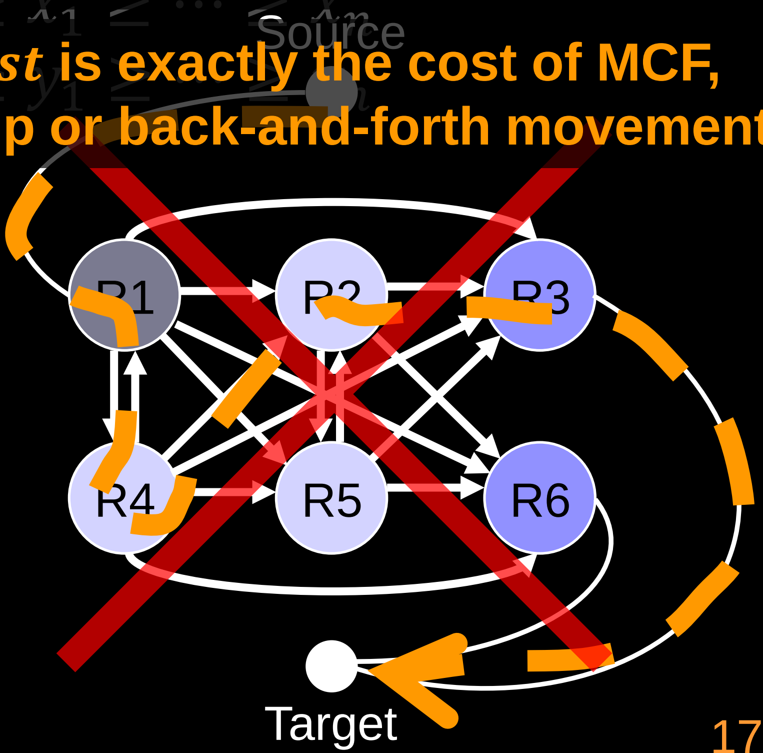
- Let $R_0 \rightarrow R_1 \rightarrow \cdots \rightarrow R_n$ be a given repairing path
- Let $R_k$ *be the* $k^{th}$ PE of the path located at $(x_k, y_k)$
- The repairing path must subjects to conditions:

Repairing path cannot have down-and-up or back-and-forth movement

$$\begin{cases} x_0 \le x_1 \le \cdots \le x_n \text{ or } x_0 \ge x_1 \ge \cdots \ge x_n \\ y_0 \le y_1 \le \cdots \le y_n \text{ or } y_0 \ge y_1 \ge \cdots \ge y_n \end{cases}$$

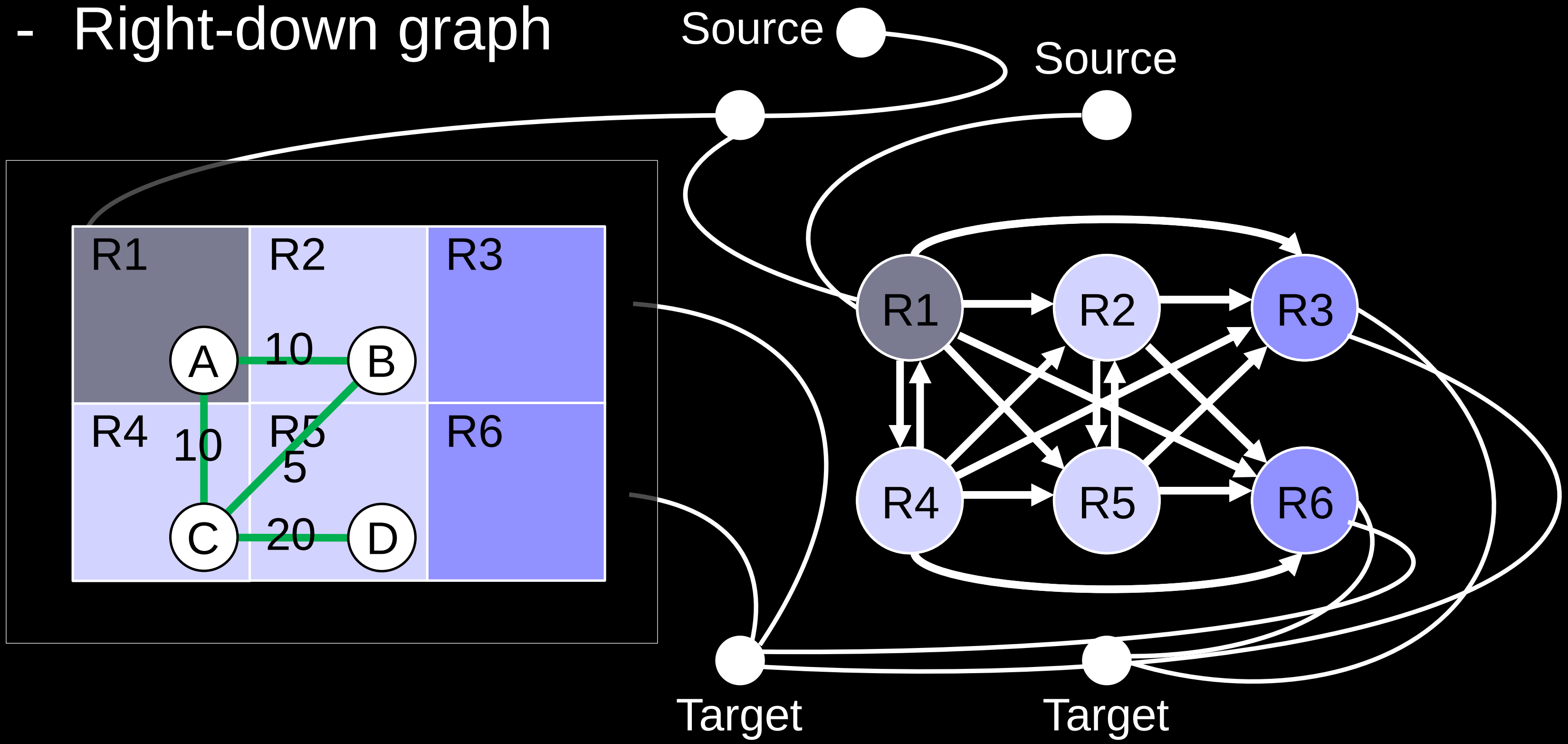
We can proof that the overall $\Delta commcost$ is exactly the cost of MCF, if the repairing path has no down-and-up or back-and-forth movement





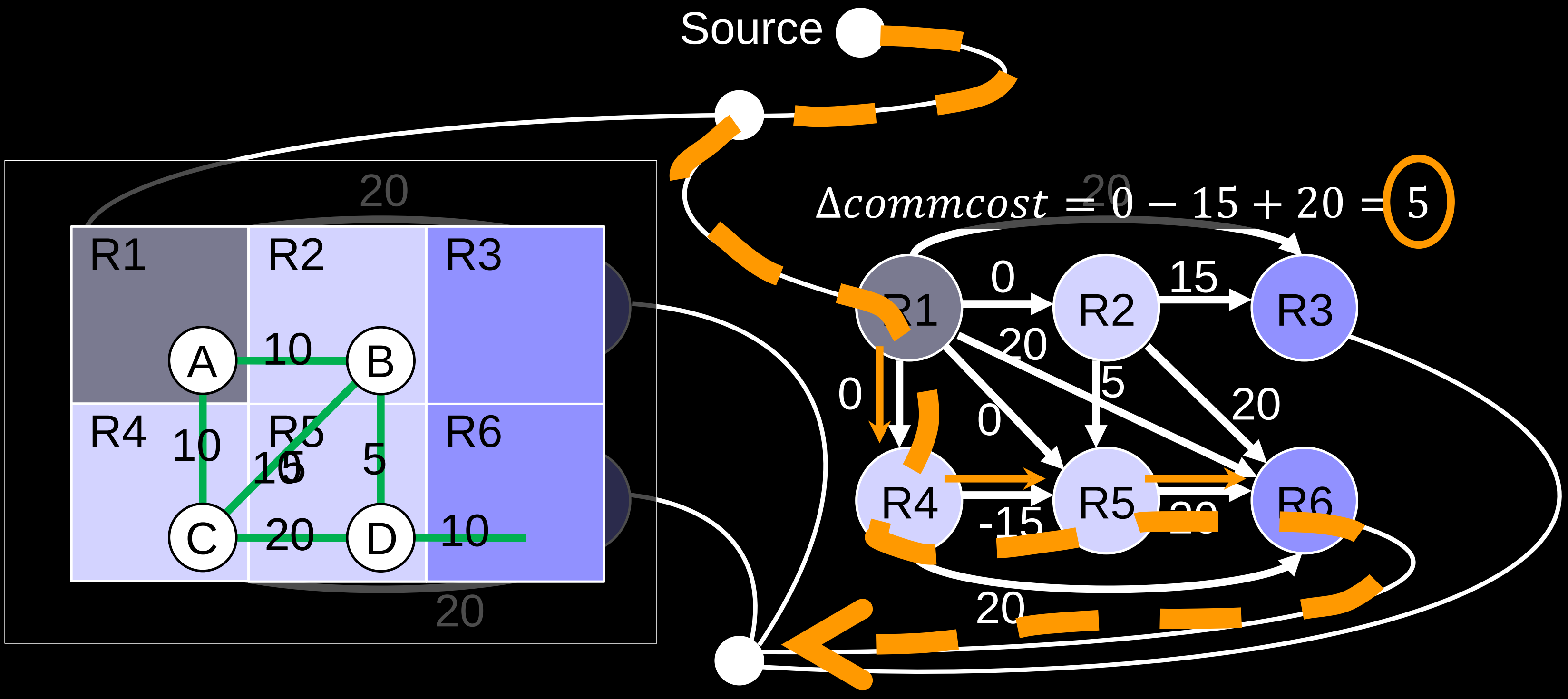
# Topology Graph Construction

- **Construct two topology graphs**
  - Right-up graph
  - Right-down graph



# A Complete Example

- **Obtain the repairing path with minimum overhead**
  - Repairing path: R1->R4->R5->R6



$\Delta commcost = 0 - 15 + 20 = \boxed{5}$

$commcost_{ori} = 1 * 10 + 1 * 10 + 1 * 20 + 2 * 5 = 50$

$commcost_{new} = 2 * 10 + 1 * 10 + 1 * 20 + 1 * 5 = 55$

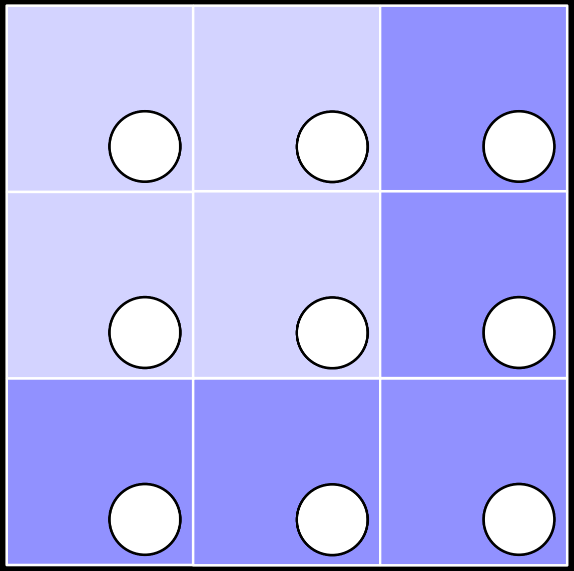
Correct $\Delta commcost$ is $55 - 50 = \boxed{5}$

# Outline

- **Multi-Processor System-on-Chip (MPSoC)**
  - Task remapping for fault-tolerant
- **Communication driven remapping method**
  - Model communication cost on edges
  - Allow tasks be moved to non-neighboring PEs
- **Initial mapping improvement**
- **Experimental results**
- **Conclusions**

# Initial Mapping Improvement

- **Our method can also be applied to improve the communication cost of a given initial mapping**



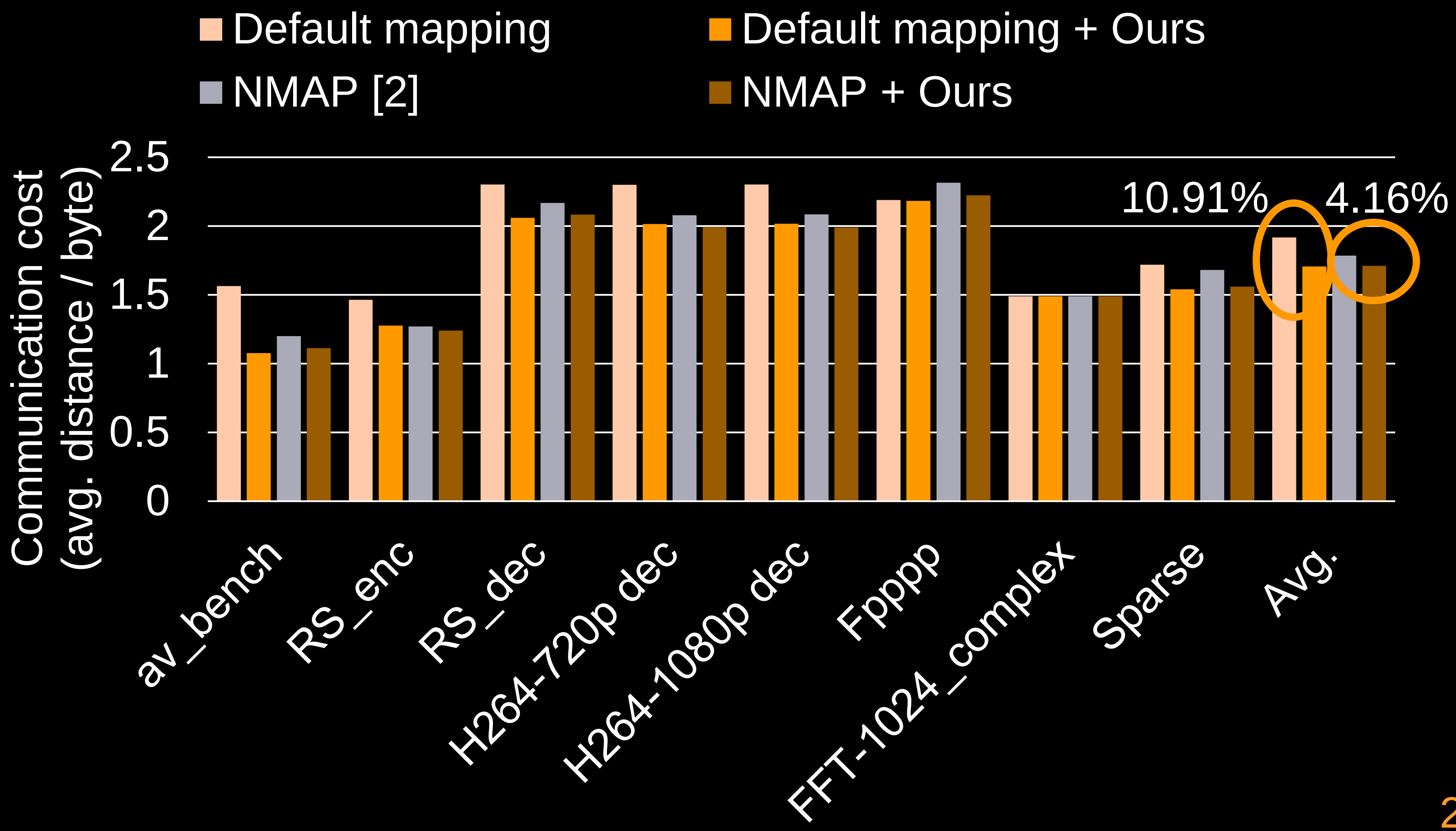
Step1: Place a faulty PE
Step2: Apply our repairing method
Step3: Repeat Step1 and Step2 for iterations
Step4: Select the mapping with lowest $commcost$

# Outline

- **Multi-Processor System-on-Chip (MPSoC)**
  - Task remapping for fault-tolerant
- **Communication driven remapping method**
  - Model communication cost on edges
  - Allow tasks be moved to non-neighboring PEs
- **Initial mapping improvement**
- **Experimental results**
- **Conclusions**

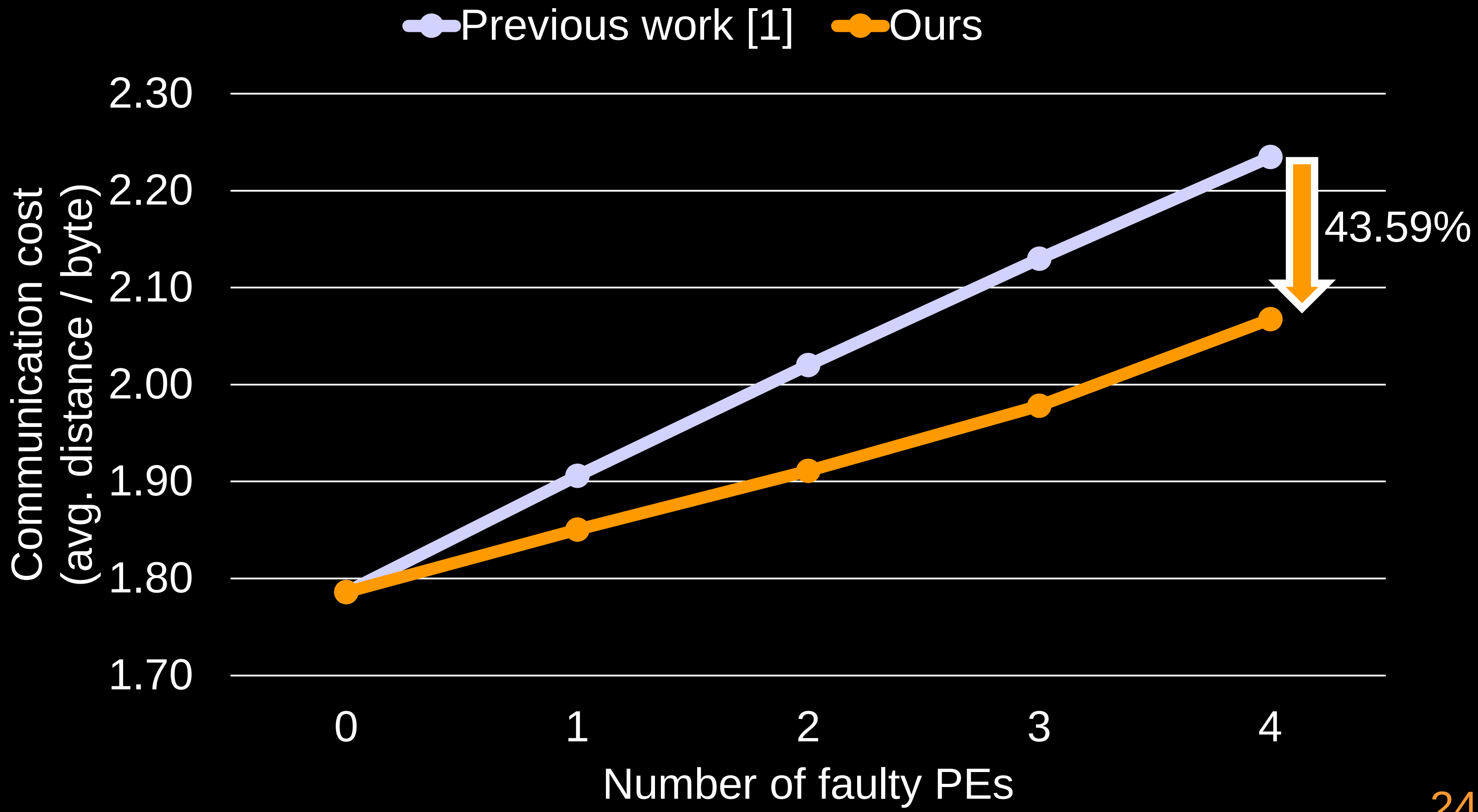
# Initial Mapping Improvement

- **10.91% improvement on default mapping**
- **4.16% improvement on NMAP [2]**



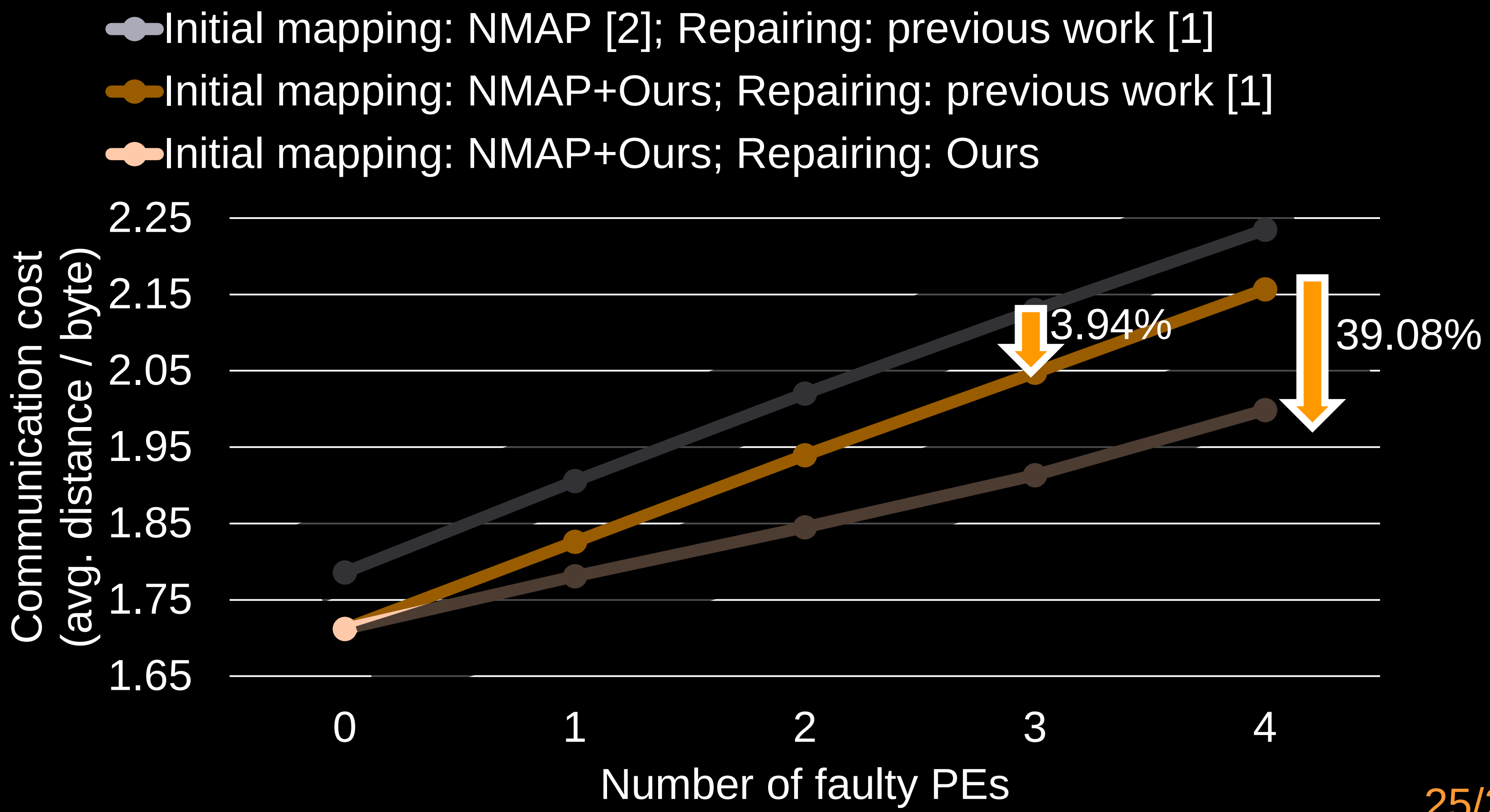
# Repairing Communication Overhead

- **43.59% less communication overhead on average**
  - Use a more precise communication cost model



# Overall Effect

- **3.94% fewer overhead on initial mapping**
- **39.08% less $commcost$ for remapping**



# Outline

- **Multi-Processor System-on-Chip (MPSoC)**
  - Task remapping for fault-tolerant
- **Communication driven remapping method**
  - Model communication cost on edges
  - Allow tasks be moved to non-neighboring PEs
- **Initial mapping improvement**
- **Experimental results**
- **Conclusions**

# Conclusions

- **A communication driven remapping algorithm**
  - Tolerate processing element failures
    - Use precise communication cost model
    - Allow flexible non-neighboring task movement
  - Improve the initial mapping
  - Be compared with previous work
    - Higher repair rate
    - Less communication overhead
      - Initial mapping (4.16%)
      - Remapping (43.59%)

# Thank you for listening

# Backup Slides

# Environment & Benchmarks

- **Transaction Generator (TG) [10]**
- **NoC traffic benchmark suite MCSL [12]**
- **Mesh $4 \times 5$ architecture**

| Application | # Tasks | # Communication Links | |
|---|---|---|---|
| | | Among tasks | Among PEs |
| av_bench [11] | 40 | 57 | 25 |
| RS_enc | 262 | 348 | 18 |
| RS_dec | 182 | 392 | 71 |
| H264-720p_dec | 2311 | 3461 | 65 |
| H264-1080p_dec | 5191 | 7781 | 65 |
| Fpppp | 334 | 1145 | 120 |
| FFT-1024_complex | 16384 | 25600 | 116 |
| Sparse | 96 | 67 | 34 |

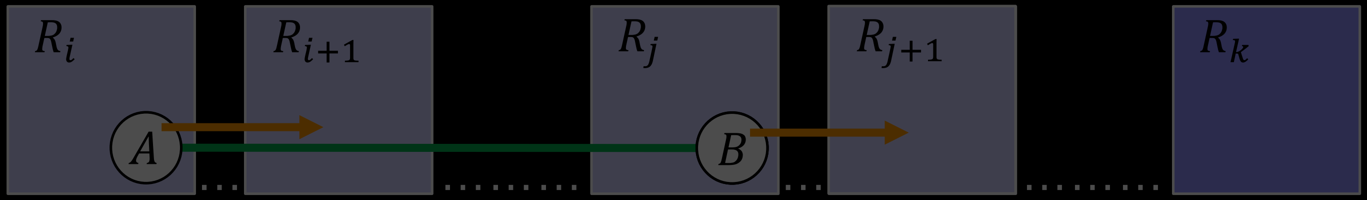
# Lemma of Topology Graph

$$\text{Let } \begin{cases} \text{task } A \text{ is moved from } \boldsymbol{R_i} = (\boldsymbol{x_i}, \boldsymbol{y_i}) \text{ to } \boldsymbol{R_{i+1}} = (\boldsymbol{x_{i+1}}, \boldsymbol{y_{i+1}}) \\ \text{task } B \text{ is moved from } \boldsymbol{R_j} = (\boldsymbol{x_j}, \boldsymbol{y_j}) \text{ to } \boldsymbol{R_{j+1}} = (\boldsymbol{x_{j+1}}, \boldsymbol{y_{j+1}}) \end{cases},$$

where $i < j$ and $i, j \in \{0,1,\dots,n\}$

$$\text{Suppose conditions } \begin{cases} x_0 \le x_1 \le \cdots \le x_n \text{ or } x_0 \ge x_1 \ge \cdots \ge x_n \\ y_0 \le y_1 \le \cdots \le y_n \text{ or } y_0 \ge y_1 \ge \cdots \ge y_n \end{cases} \text{ are satified}$$

$$\begin{aligned} \Delta commcost_{A,B} &= size_{A,B} \times \left( distance(\boldsymbol{R_{i+1}}, \boldsymbol{R_{j+1}}) - distance(\boldsymbol{R_i}, \boldsymbol{R_j}) \right) \\ &= size_{A,B} \times \left( \left( |x_{j+1} - x_{i+1}| + |y_{j+1} - y_{i+1}| \right) - \left( |x_j - x_i| + |y_j - y_i| \right) \right) \\ &= size_{A,B} \times \left( \left( |x_{j+1} - x_j| + |y_{j+1} - y_j| \right) - \left( |x_{i+1} - x_i| + |y_{i+1} - y_i| \right) \right) \\ &= size_{A,B} \times \left( distance(\boldsymbol{R_j}, \boldsymbol{R_{j+1}}) - distance(\boldsymbol{R_i}, \boldsymbol{R_{i+1}}) \right) \\ &= size_{A,B} \times \Delta distance(\boldsymbol{B}) - size_{A,B} \times \Delta distance(\boldsymbol{A}) \end{aligned}$$



# Cost of MCF Algorithm ($\Delta commcost$)

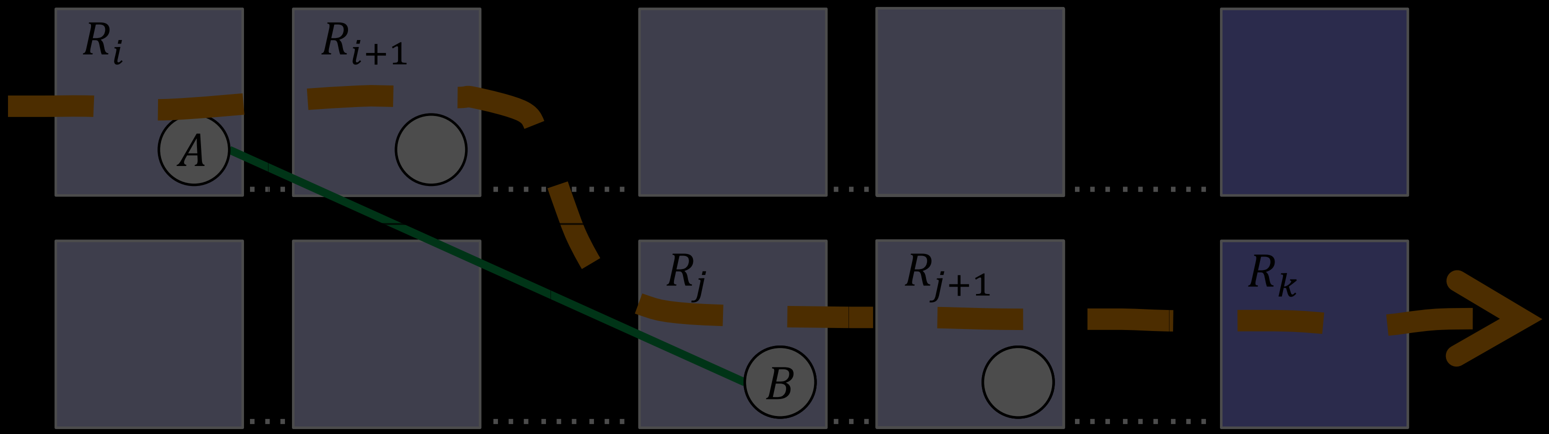
$$\Delta commcost = \sum_{\text{Task graph edge } A,B} \Delta commcost_{A,B}$$

$$= \sum_{\text{Task graph edge } A,B} size_{A,B} \times \Delta distance(\boldsymbol{A}) - size_{A,B} \times \Delta distance(\boldsymbol{B})$$

$$= \sum_{\text{Moved task } A} \left( \sum_{\text{shorten edge } A,B} size_{A,B} - \sum_{\text{lenthen edge } A,B} size_{A,B} \right) \times \Delta distance(\boldsymbol{A})$$

$$= \sum_{R_i \to R_{i+1} \in \text{repairing path}} \Delta commcost(\boldsymbol{R_i} \to \boldsymbol{R_{i+1}})$$

↖ Exactly the cost of the MCF algorithm


$R_i$
$R_{i+1}$
$A$
$R_j$
$R_{j+1}$
$R_k$
$B$