HARVEST

Adaptive Photonic Switching Schedules for Collective Communication in Scale-up Domains

Mahir Rahman, Samuel Joseph, Nihar Kodkani, Behnaz Arzani, Vamsi Addanki

PURDUE UNIVERSITY

STyGIANet

Microsoft

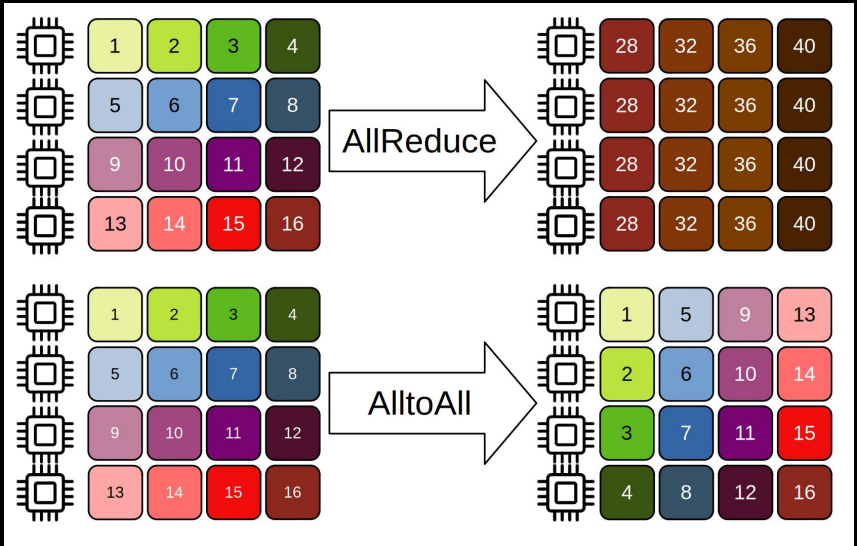
NSF

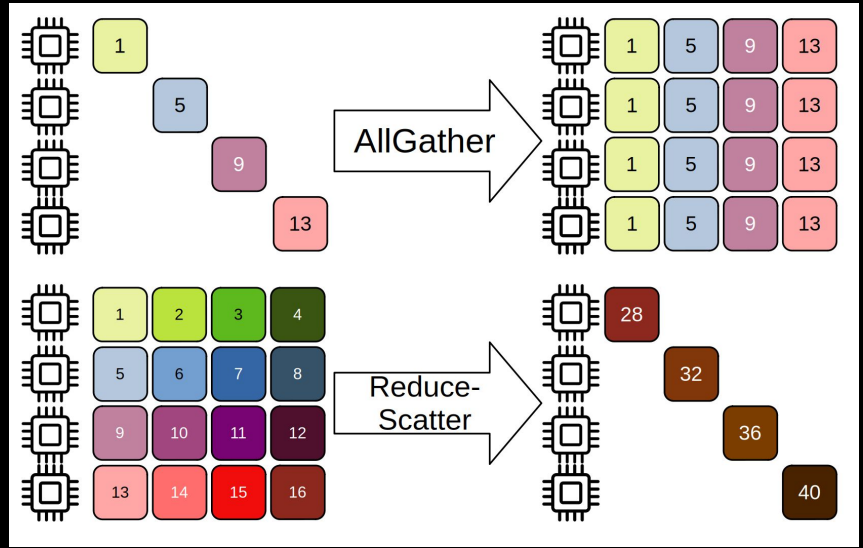
Explosive Growth of Model Sizes

Source: Epoch AI

HARVEST

Distributed Computing → Collective Communication

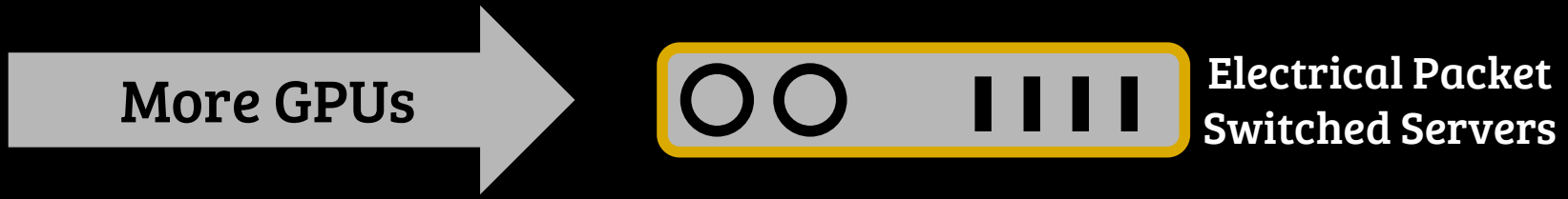




HARVEST

Electrical Interconnects are Hitting their Limits

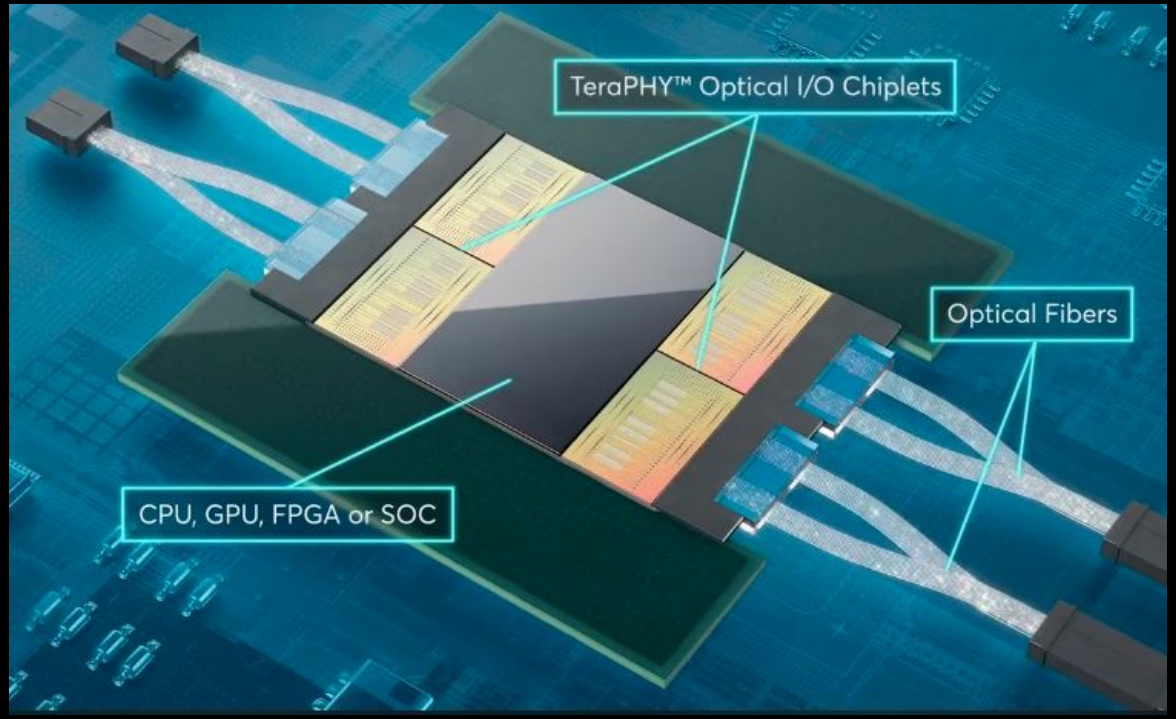
- Need for more compute resources in our scale-up domains
- However, electrical packet switches face:
  - Limited bandwidth scaling
  - Limited copper reach



Scalable ?

HARVEST

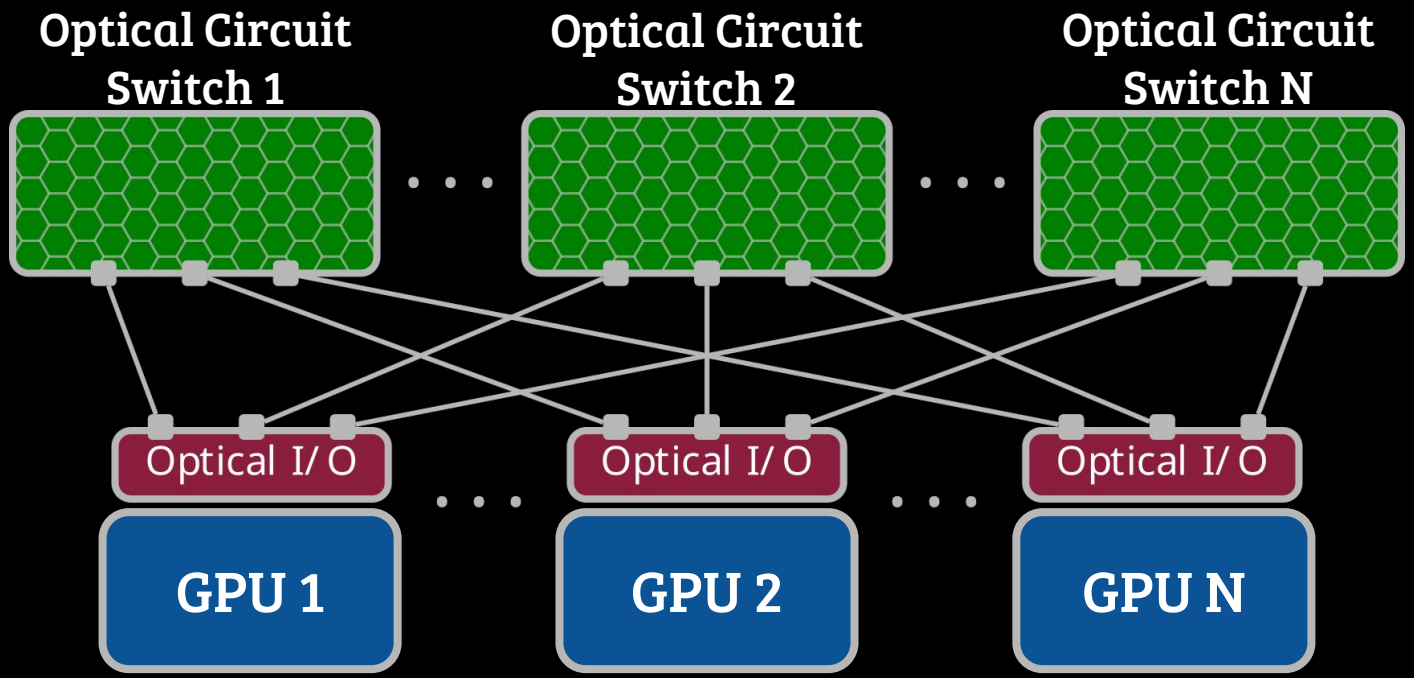
Enter Photonic Interconnects



Source: Ayar Labs

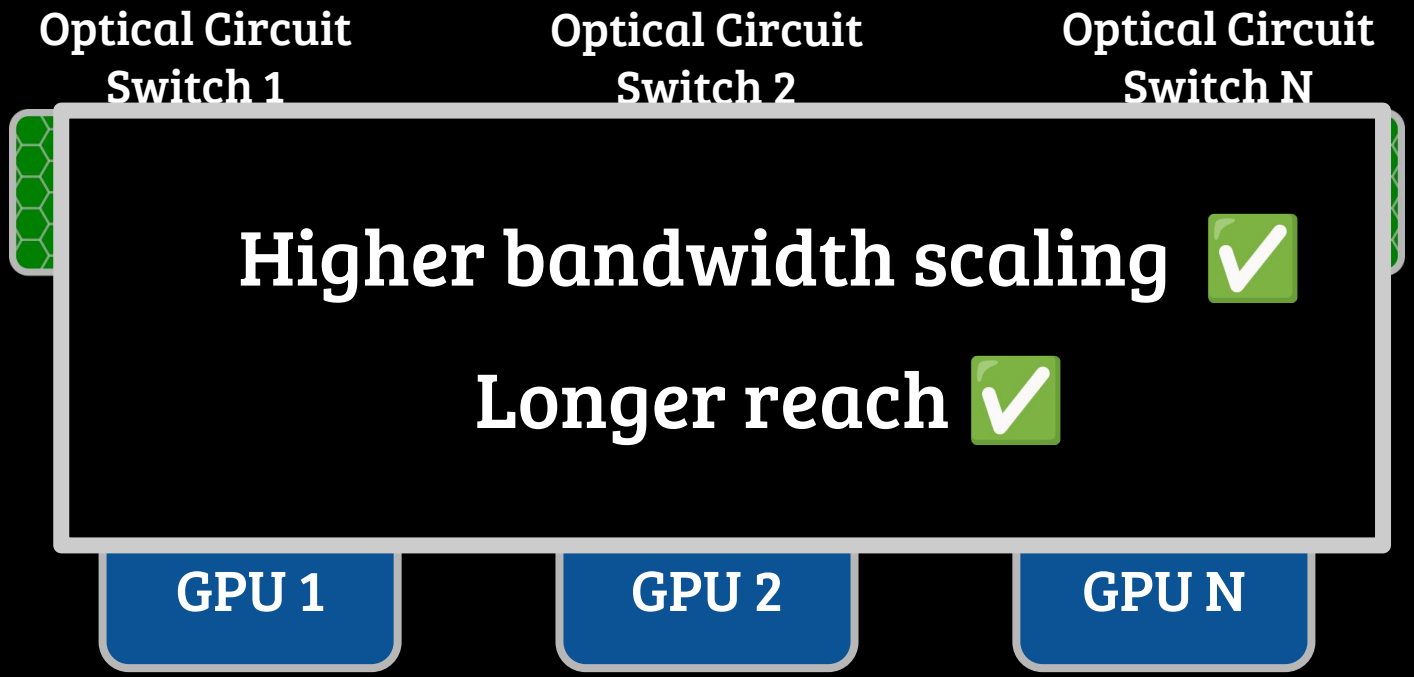
HARVEST

Chip-Chip Photonic Interconnects



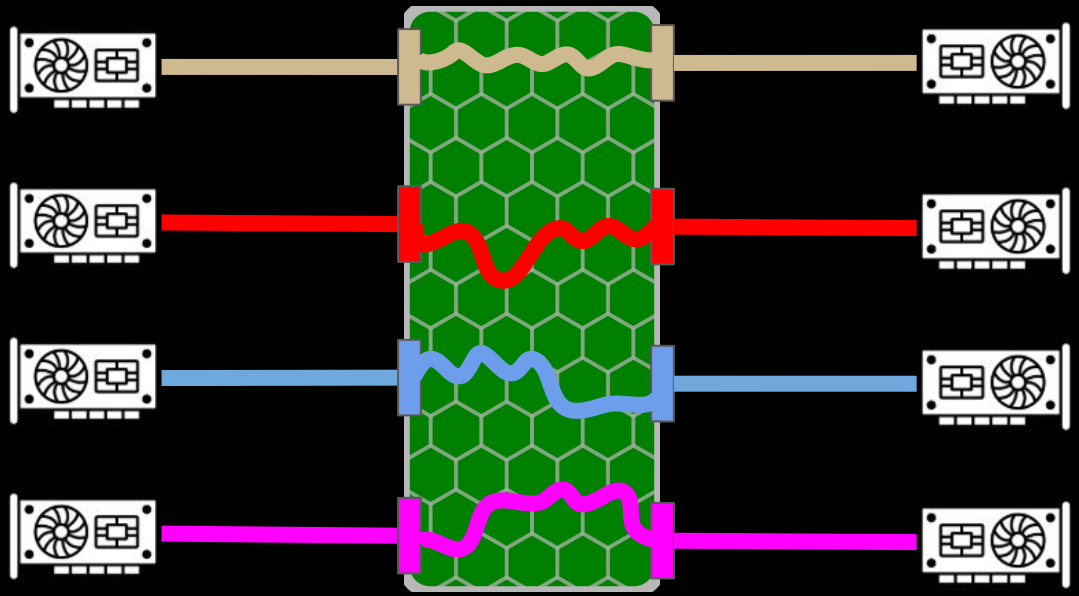
HARVEST

Chip-Chip Photonic Interconnects



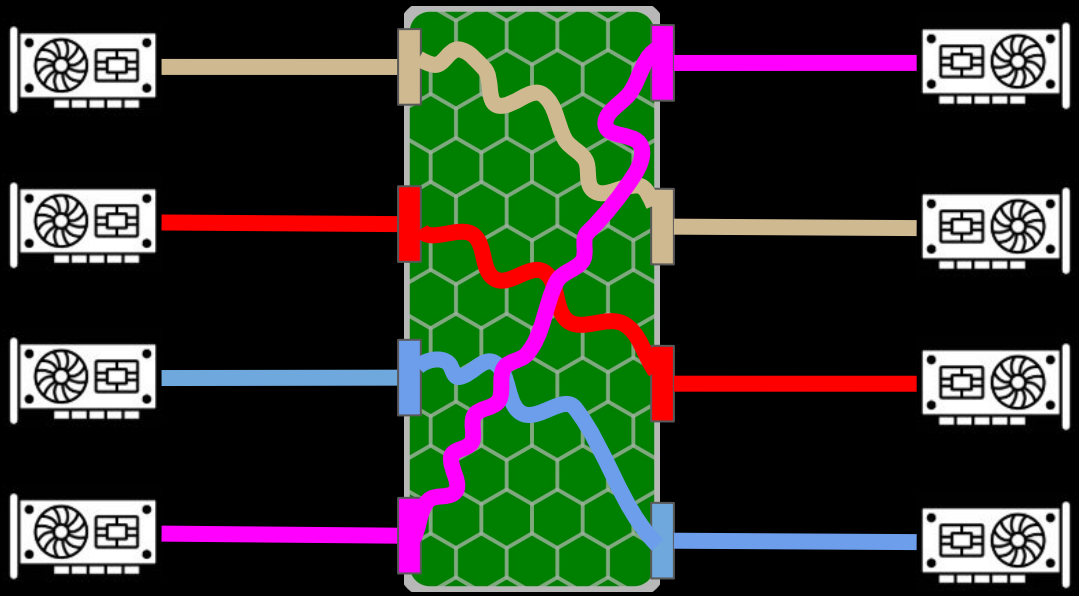
HARVEST

Chip-Chip Photonic Interconnects



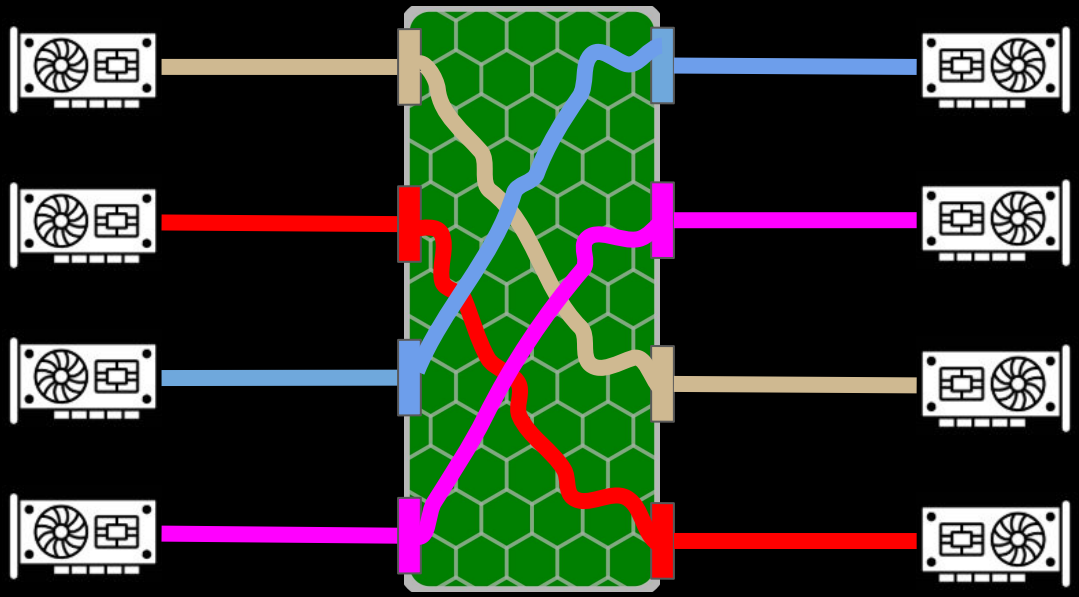
HARVEST

Chip-Chip Photonic Interconnects



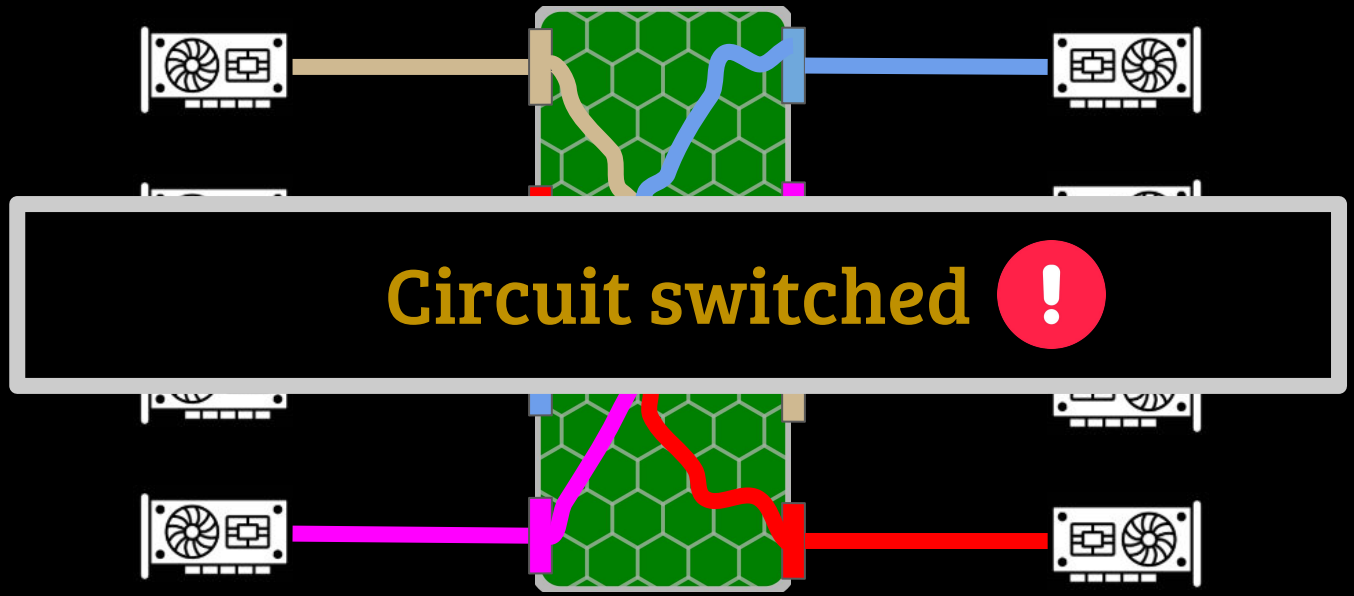
HARVEST

Chip-Chip Photonic Interconnects



HARVEST

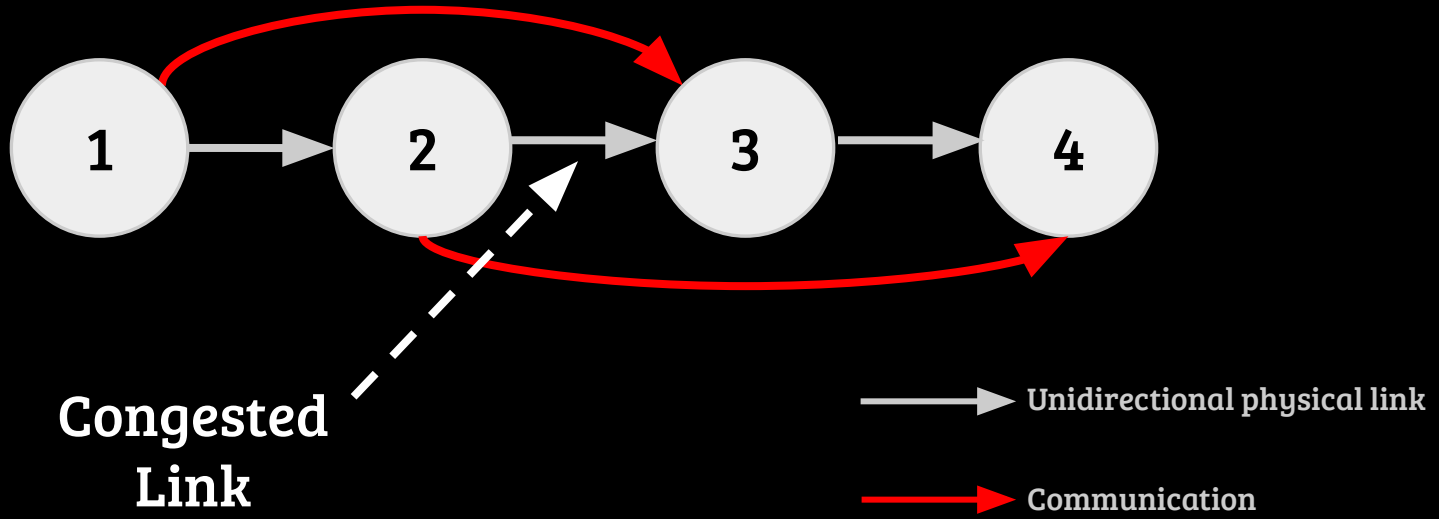
Chip-Chip Photonic Interconnects



HARVEST

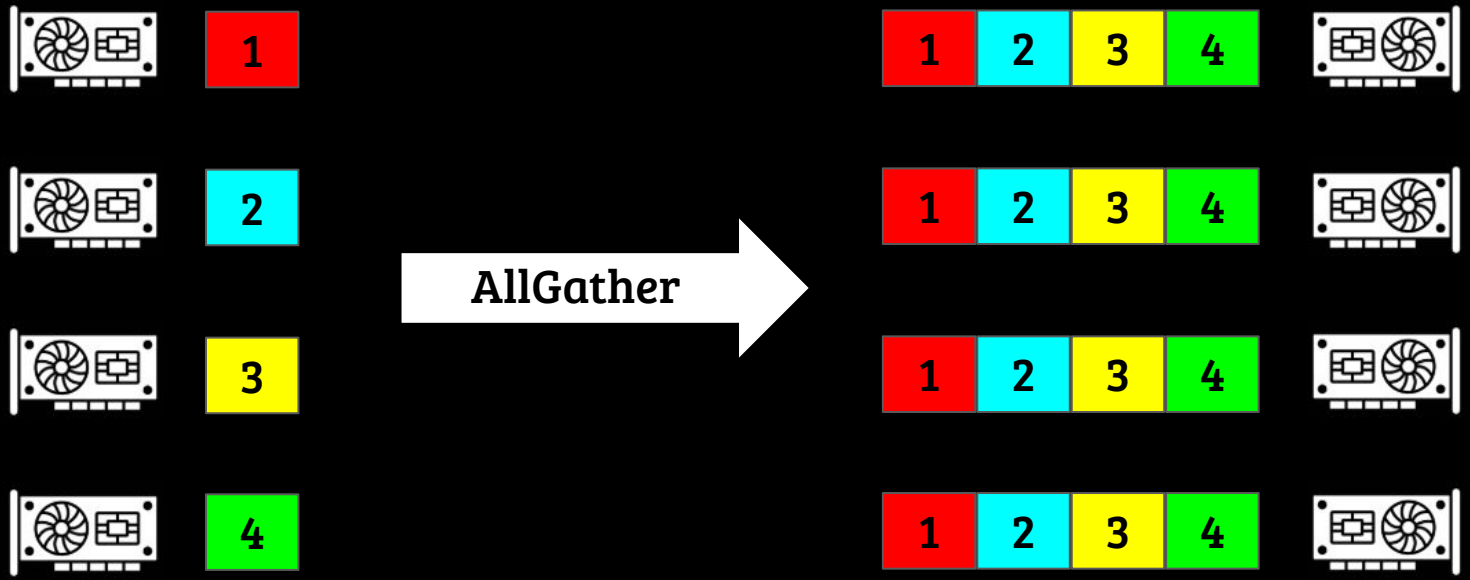
AI Workloads Face New Challenge

- GPU communication is bandwidth intensive and latency sensitive
- Worsens during congestion



Example: AllGather Collective Algorithm

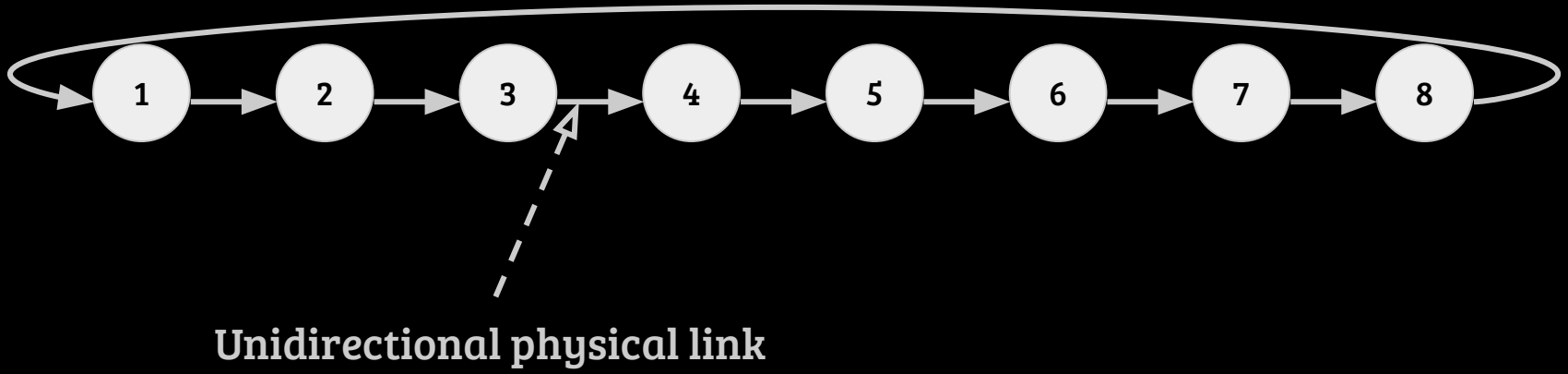
Every GPU transmits distinct chunks of its data to all other GPUs



HARVEST

AllGather: Recursive Doubling (Example)

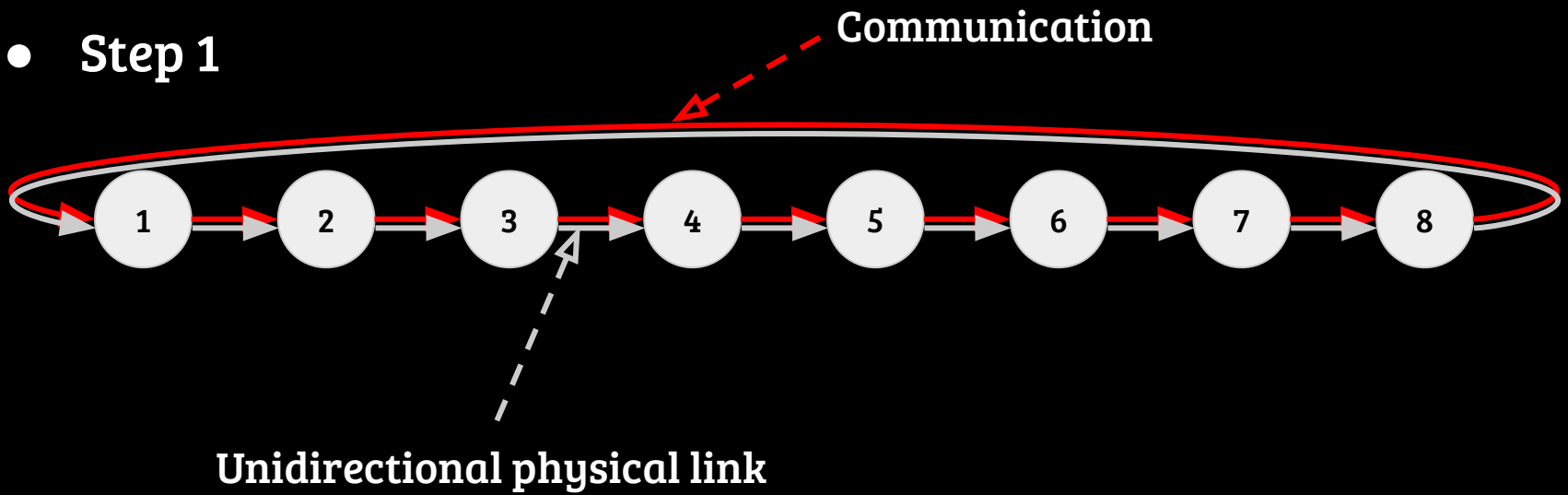
- 8 GPUs connected in a ring topology



HARVEST

AllGather: Recursive Doubling (Example)

- Step 1



No congestion ✅

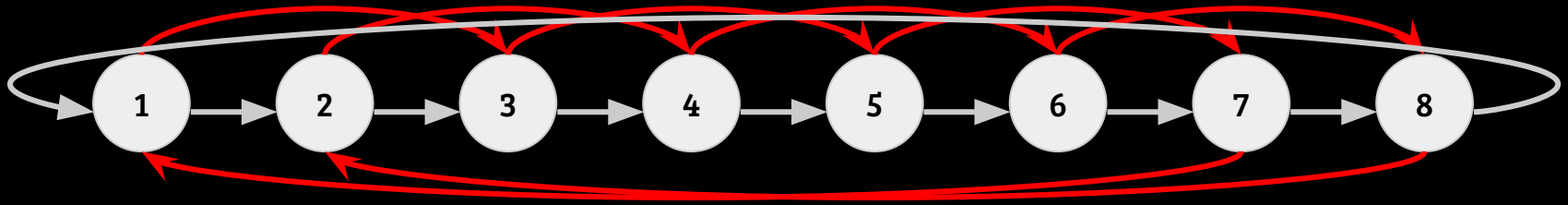
HARVEST

Unidirectional physical link

Communication

AllGather: Recursive Doubling (Example)

- Step 2



Moderate congestion→2 overlapping transfers

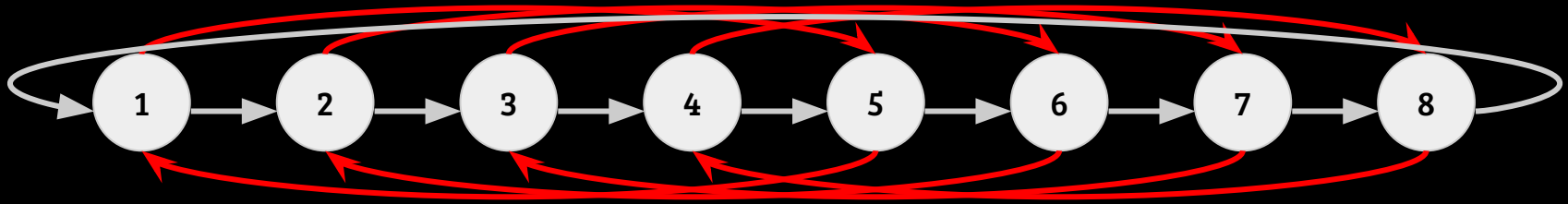
HARVEST

Unidirectional physical link

Communication

AllGather: Recursive Doubling (Example)

- Step 3



Severe congestion→4 overlapping transfers

HARVEST

An alternative option:

Reconfigure the topology!

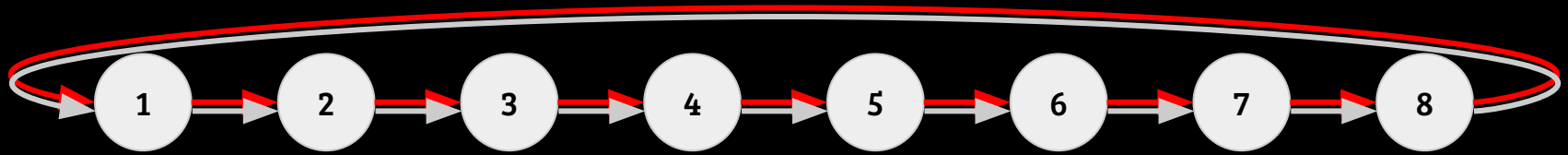
HARVEST

Unidirectional physical link

Communication

AllGather: Recursive Doubling (Example)

- Step 1



No congestion ✅

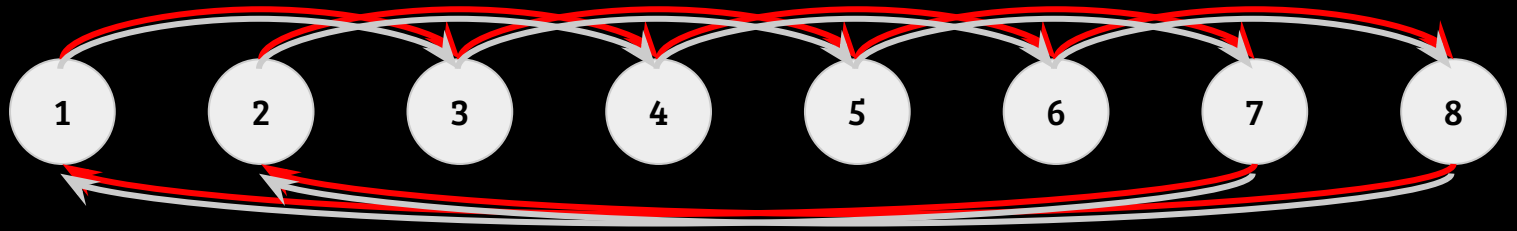
HARVEST

Unidirectional physical link

Communication

AllGather: Recursive Doubling (Example)

- Step 2



Reconfigure! No congestion ✅

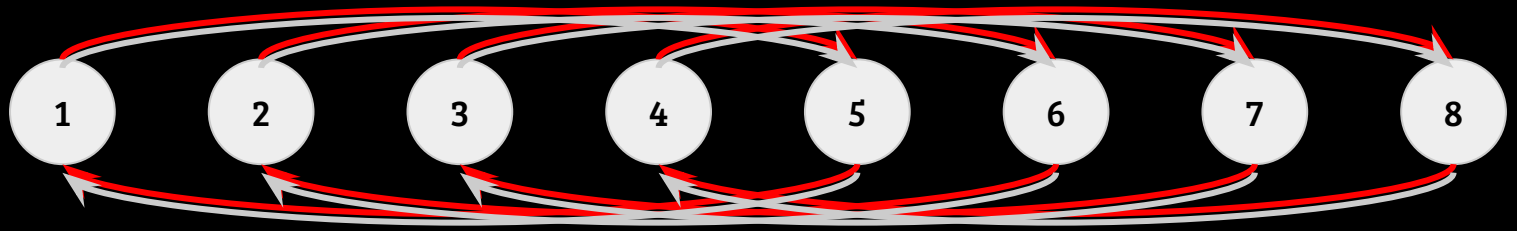
HARVEST

Unidirectional physical link

Communication

AllGather: Recursive Doubling (Example)

- Step 3



Reconfigure and no congestion again! ✅

HARVEST

Unidirectional physical link

Communication

AllGather: Recursive Doubling (Example)

- Step 3

Different Reconfiguration Schemes

- Periodic circuit switching
- Demand-aware Greedy Matchings
- Failure mitigation
- One-shot optimization
- Aggregate Demand Matrix Decomposition

HARVEST

Different Reconfiguration Schemes

- Periodic circuit switching
- Demand-aware Greedy Matchings

} **Assumption:** Very low reconfiguration delay

- Failure mitigation
- One-shot optimization

} **Assumption:** Very high reconfiguration delay

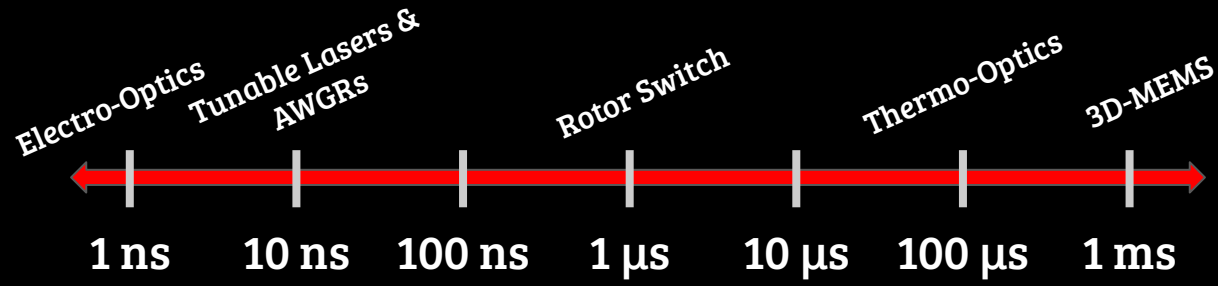
- Aggregate Demand Matrix Decomposition

} **Assumption:** No data dependency

HARVEST

In reality…

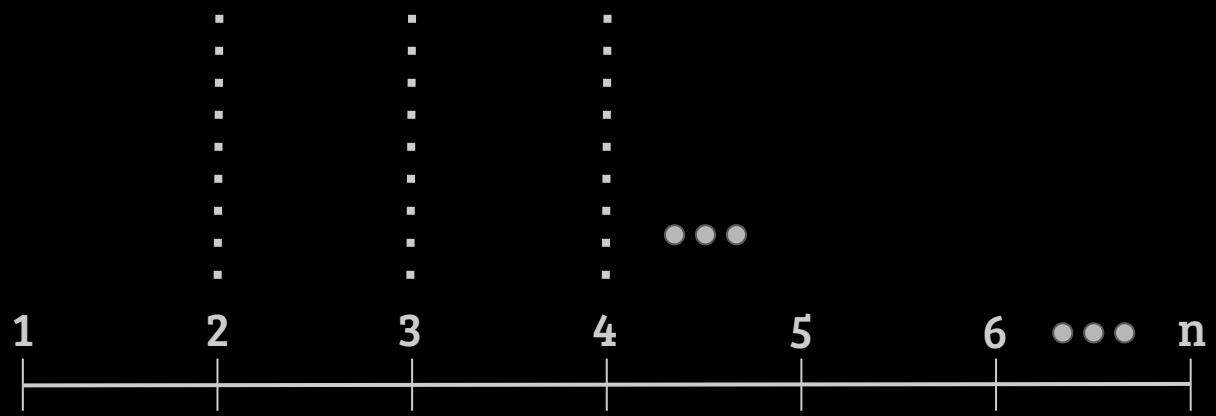
- Reconfiguration delays span a wide spectrum
- GPU communication has data dependencies
  - *Aggregate Demand Matrix is not a good abstraction!*



Reconfiguration Delays

HARVEST

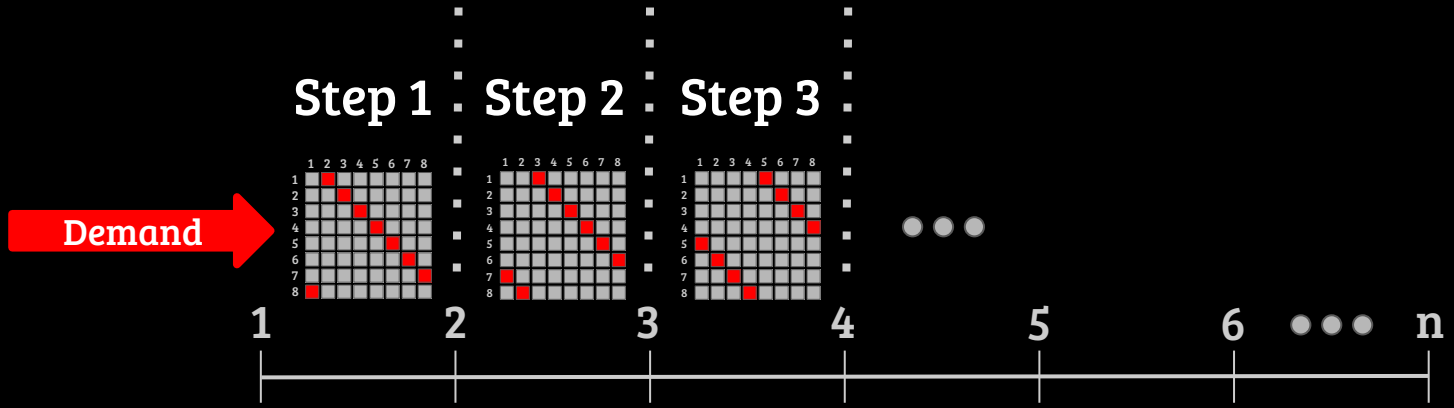
Collective Communication and Completion Time



Communication steps of a collective

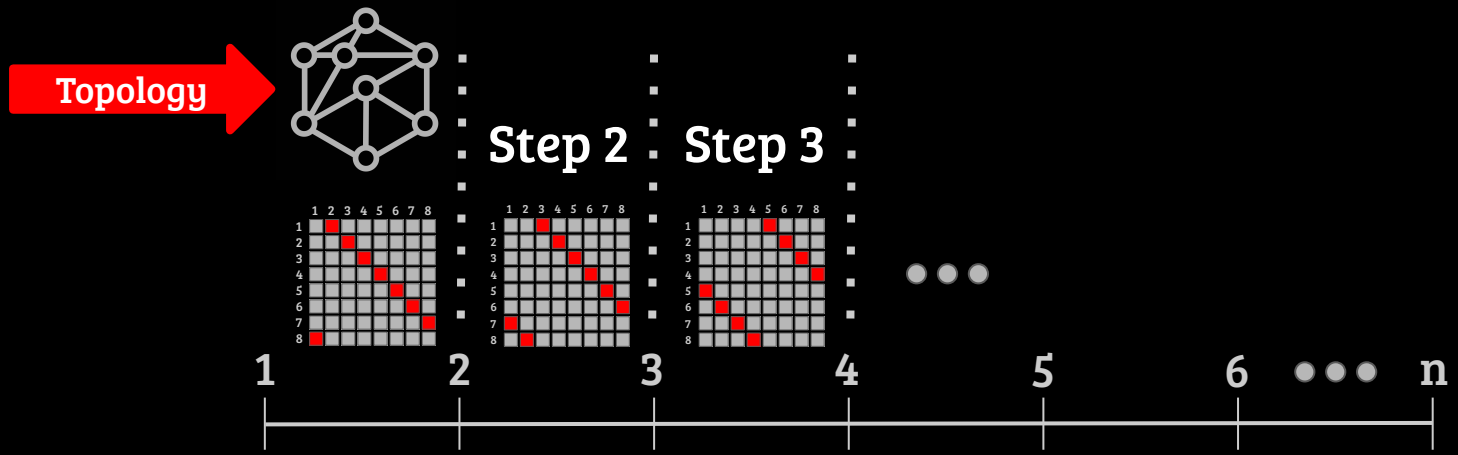
HARVEST

Collective Communication and Completion Time



Communication steps of a collective

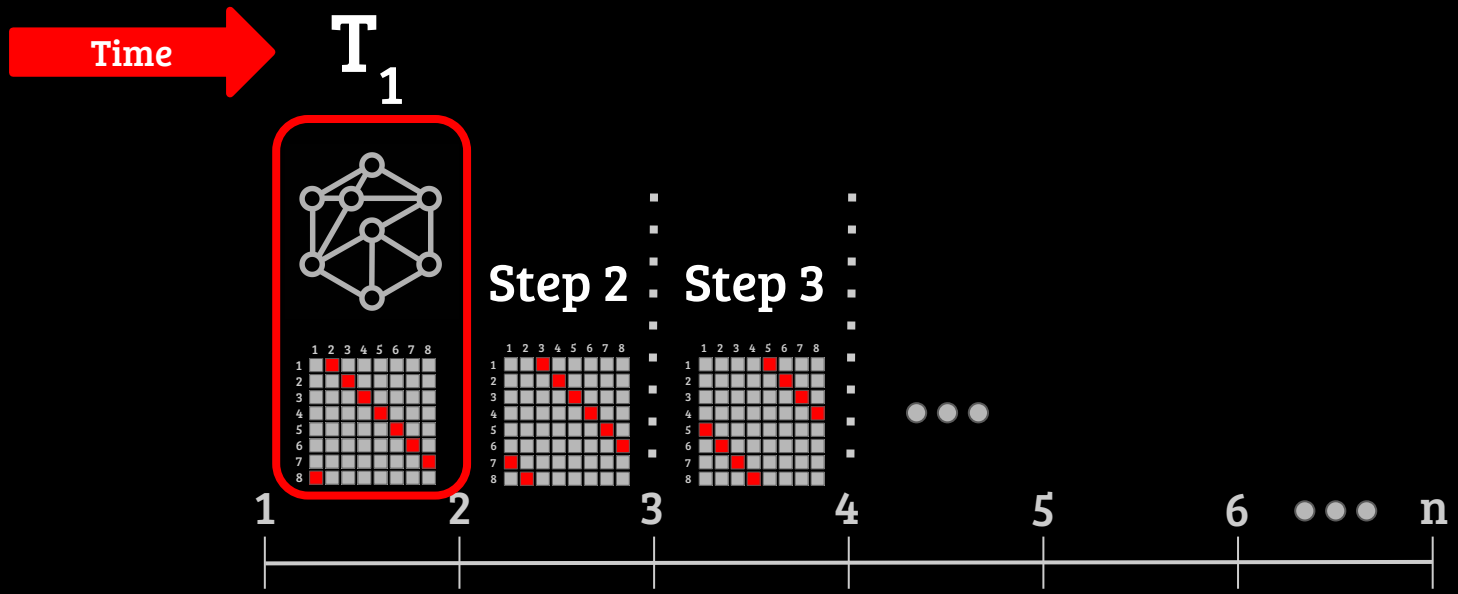
Collective Communication and Completion Time



Communication steps of a collective

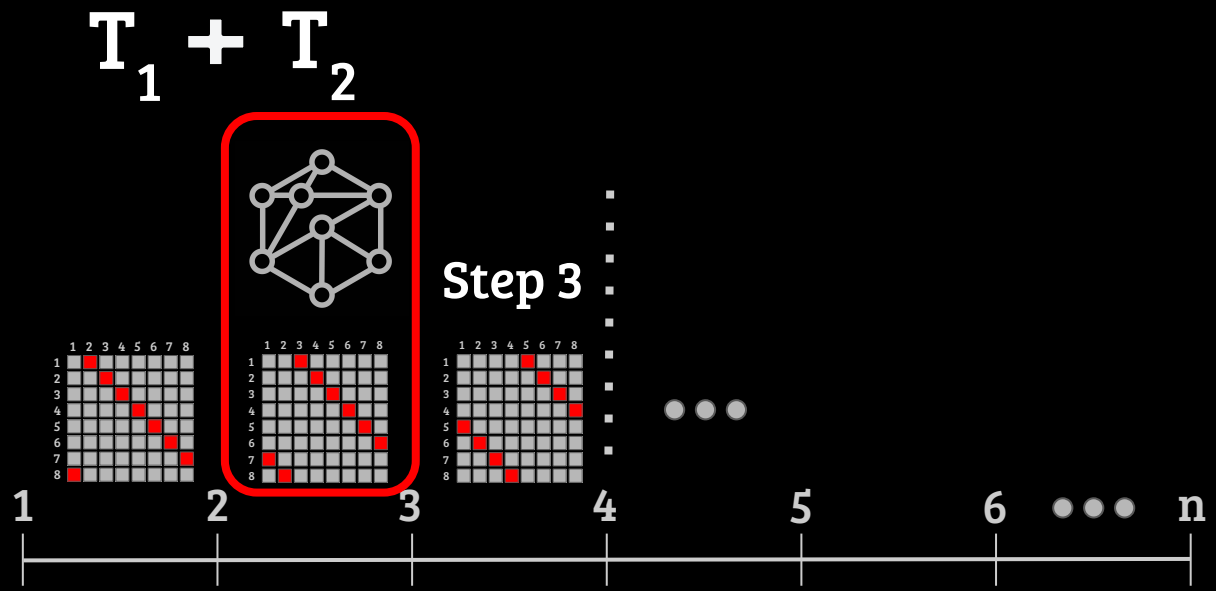
HARVEST

Collective Communication and Completion Time



HARVEST

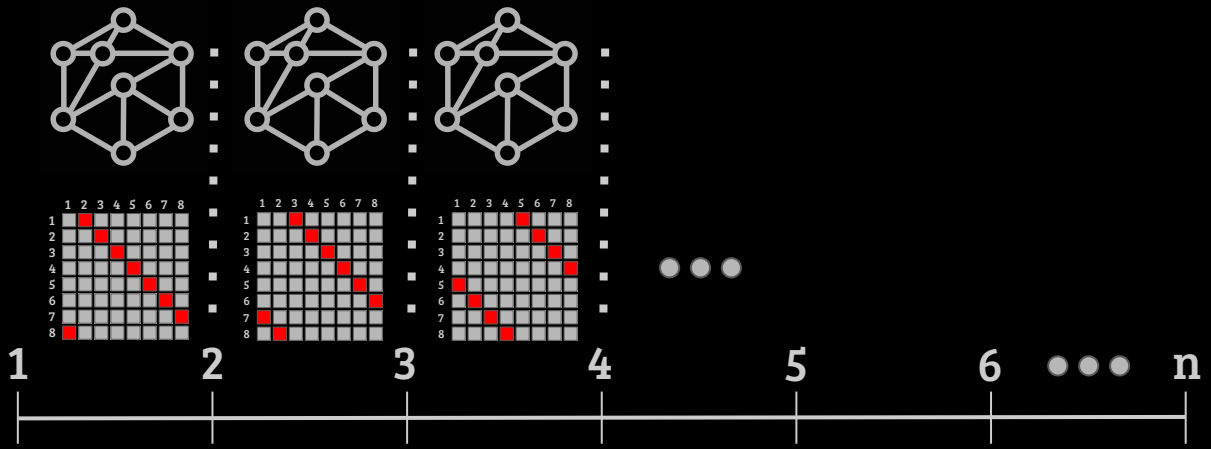
Collective Communication and Completion Time

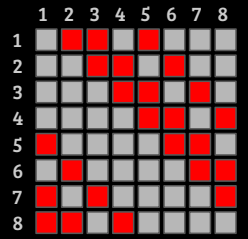


Communication steps of a collective

Collective Communication and Completion Time

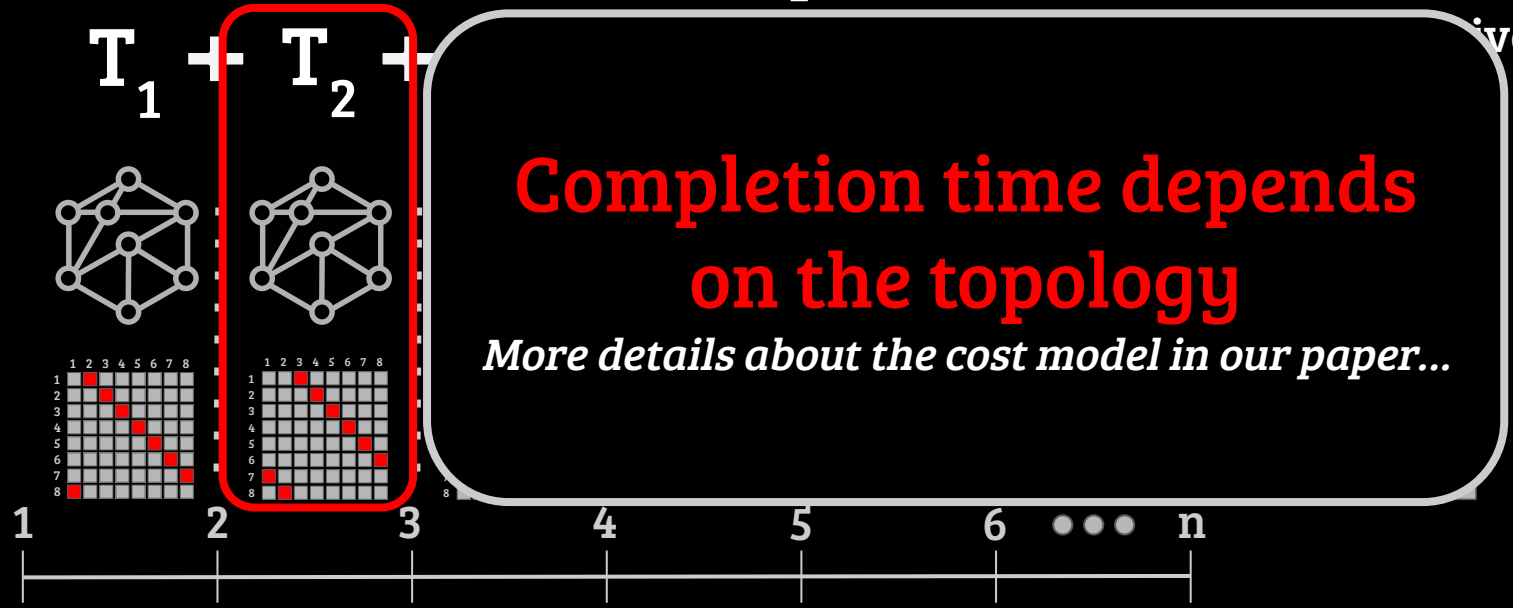
$$T_1 + T_2 + T_3 \cdots = \text{Total Collective Completion Time}$$





HARVEST

Collective Communication and Completion Time



HARVEST

An Optimization Opportunity

If we reconfigure:

- Reduced congestion and propagation delays
- Incur reconfiguration delay

Else:

- High congestion and propagation delays
- No reconfiguration delays

HARVEST

An Optimization Opportunity

Challenge:
When and ***how*** should we reconfigure?

HARVEST

HARVEST

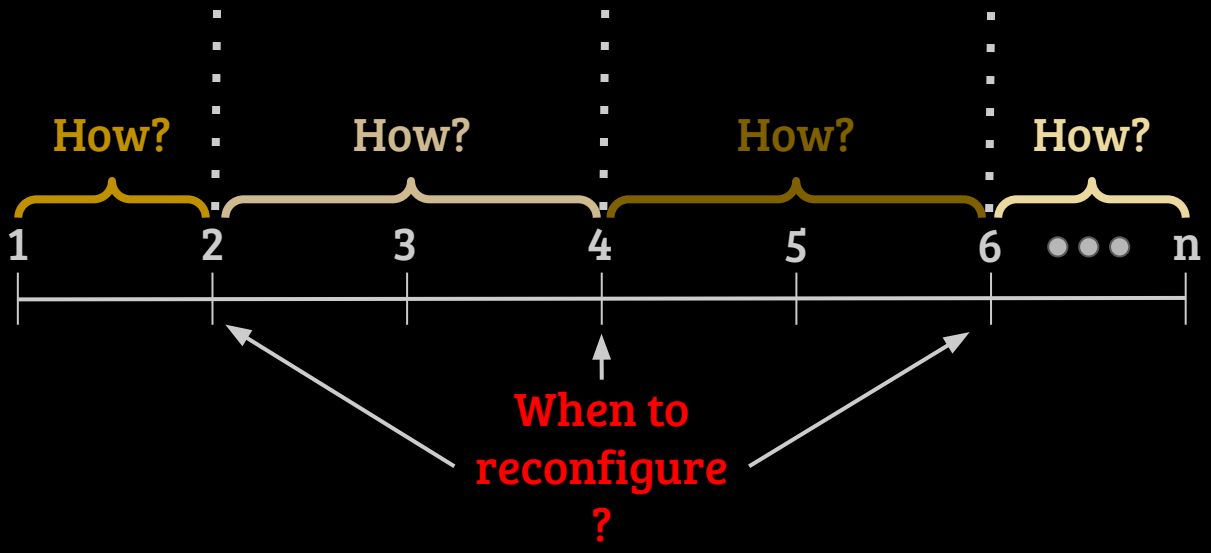
A novel framework that synthesizes optimal photonic switching schedules for collective algorithms

Collective Algorithm
Topology Constraints (# of links, nodes, etc.)
Reconfiguration Delay

Topology
Reconfiguration
Schedule

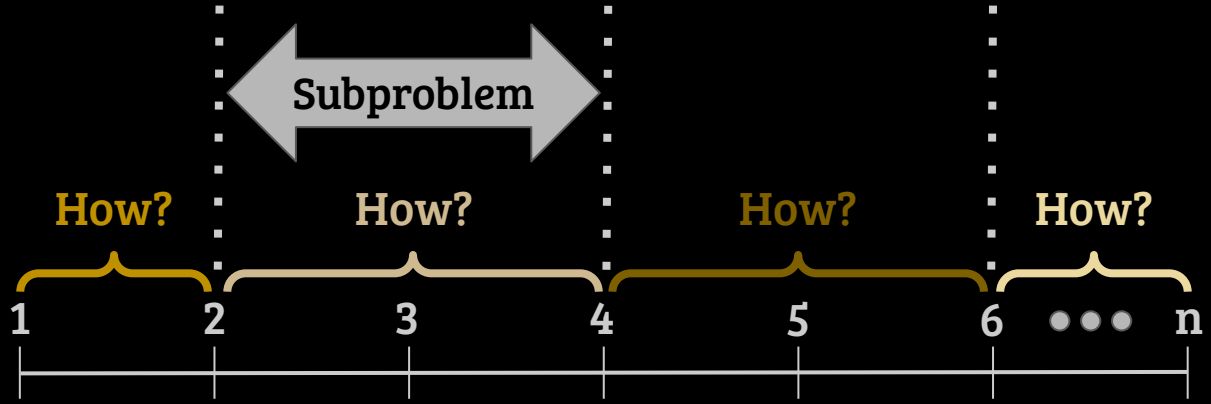
When and How to Reconfigure?

1. When? Partition the sequence of communication steps into intervals
2. How? Find an optimal static topology for each interval



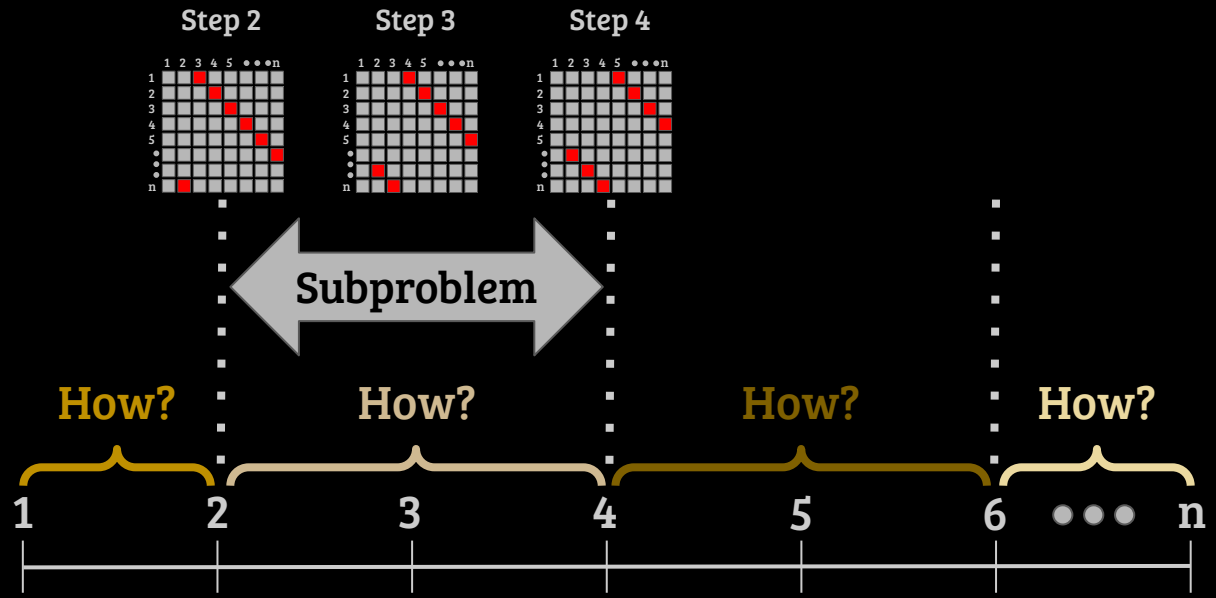
HARVEST

When and How to Reconfigure?



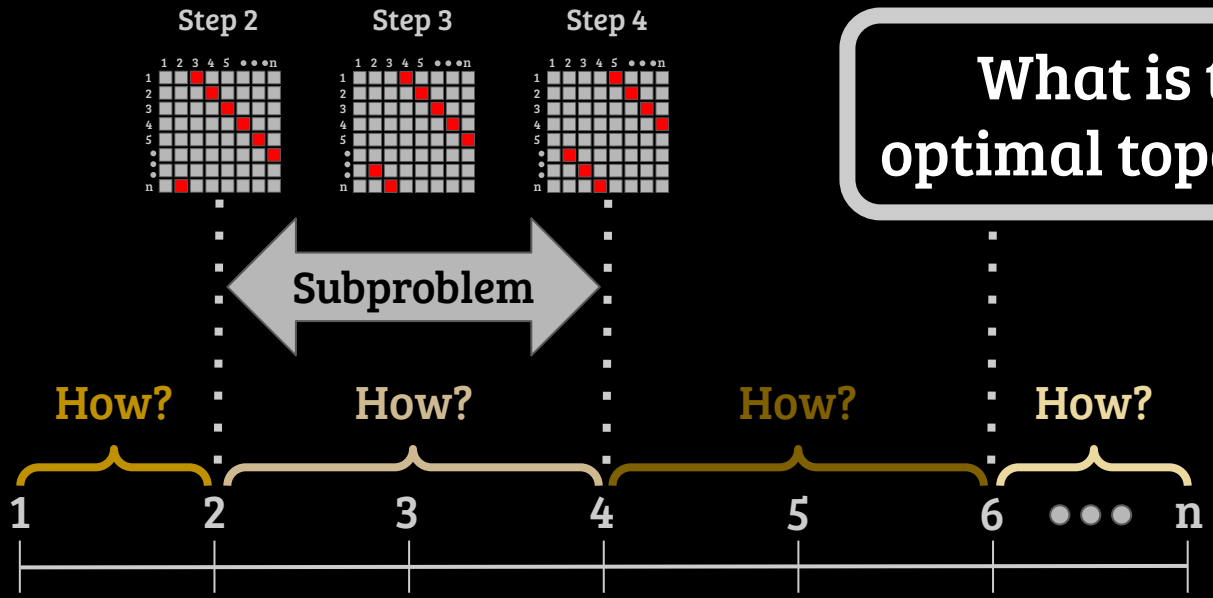
HARVEST

When and How to Reconfigure?



HARVEST

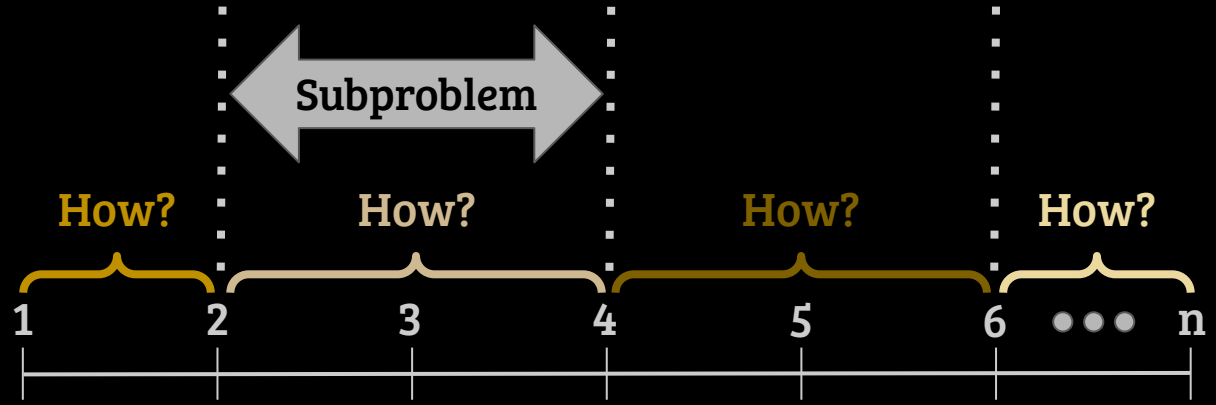
When and How to Reconfigure?



HARVEST

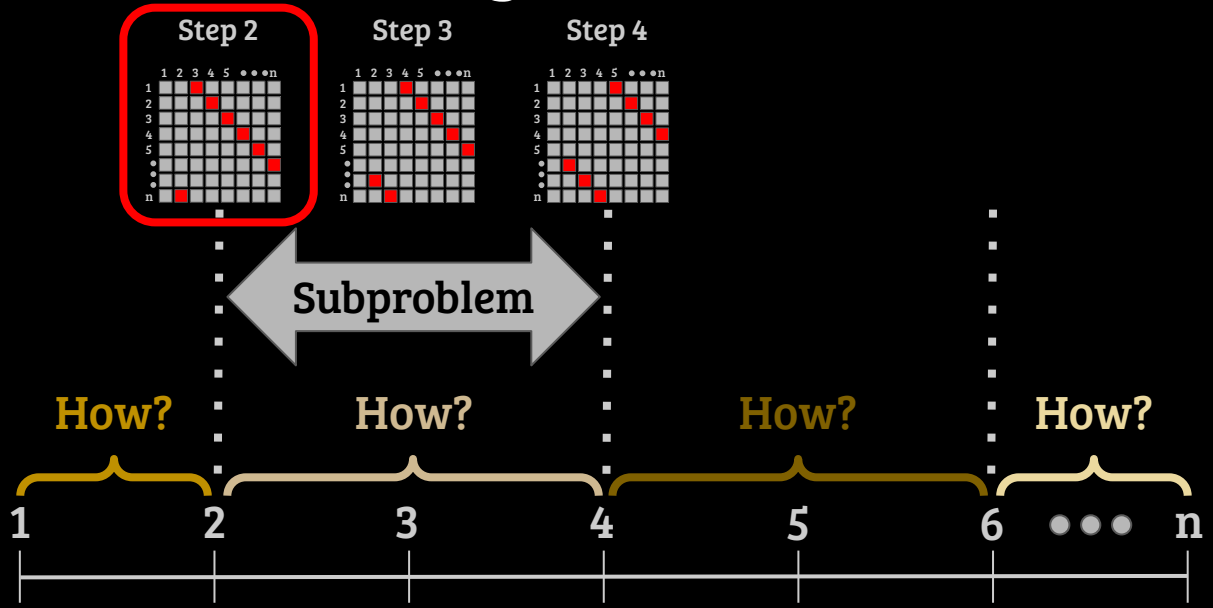
When and How to Reconfigure?

Mixed Integer Second Order Conic Problem



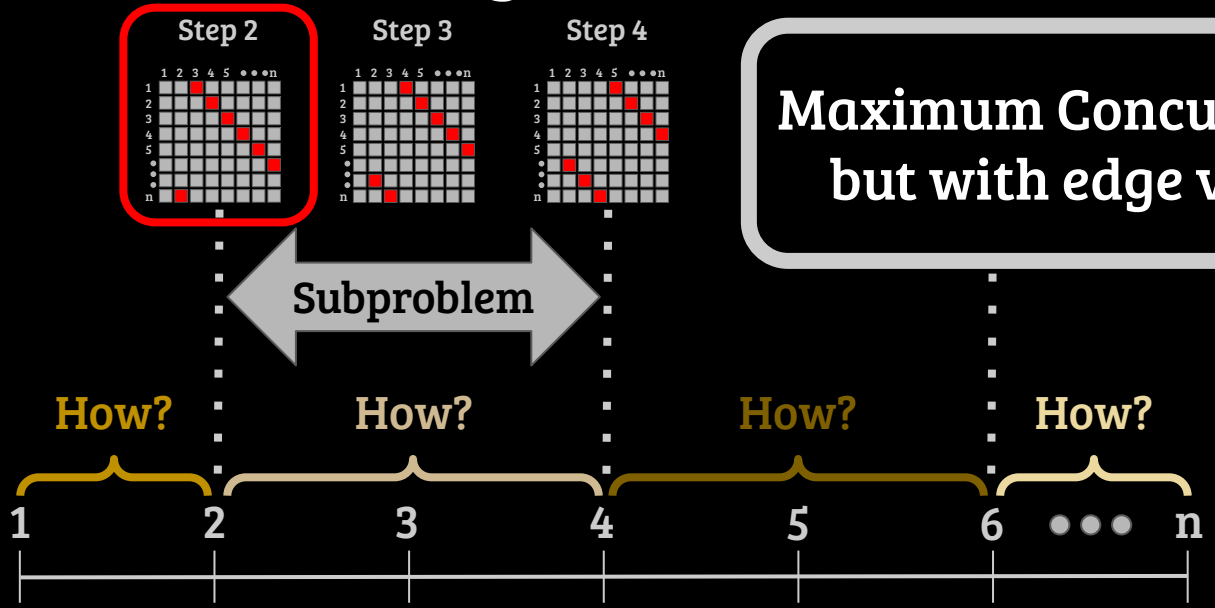
HARVEST

When and How to Reconfigure?



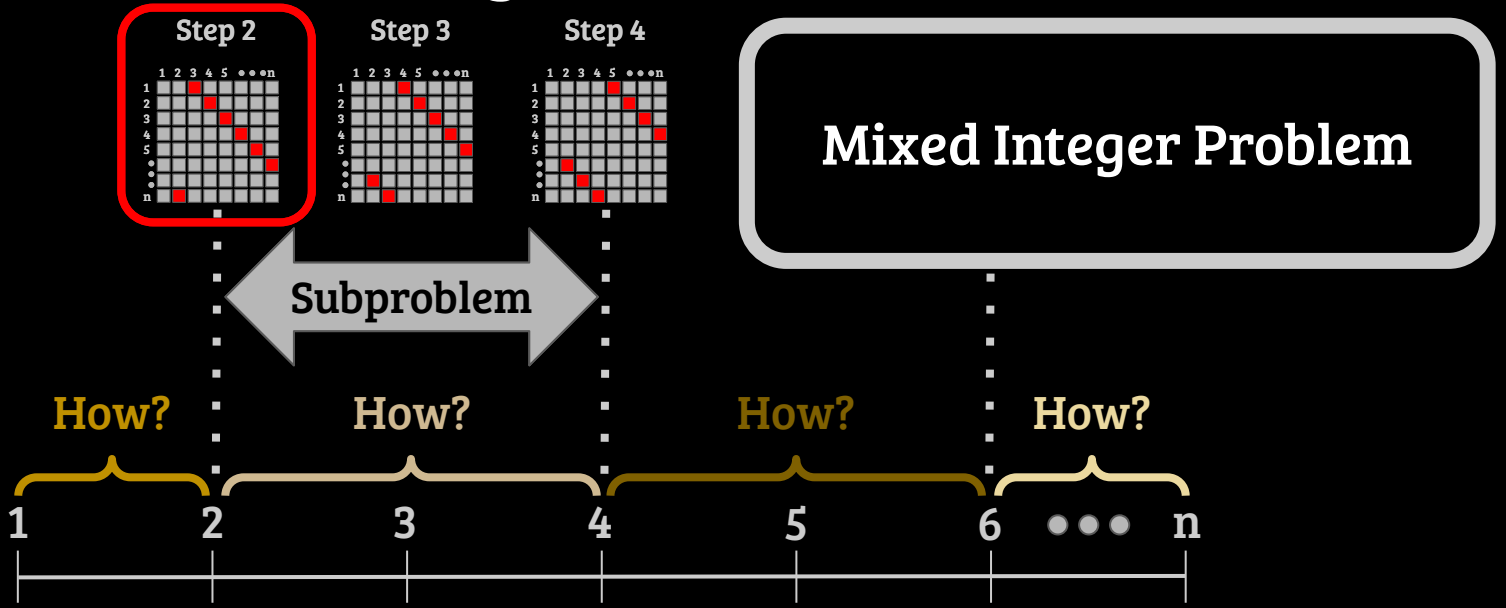
HARVEST

When and How to Reconfigure?



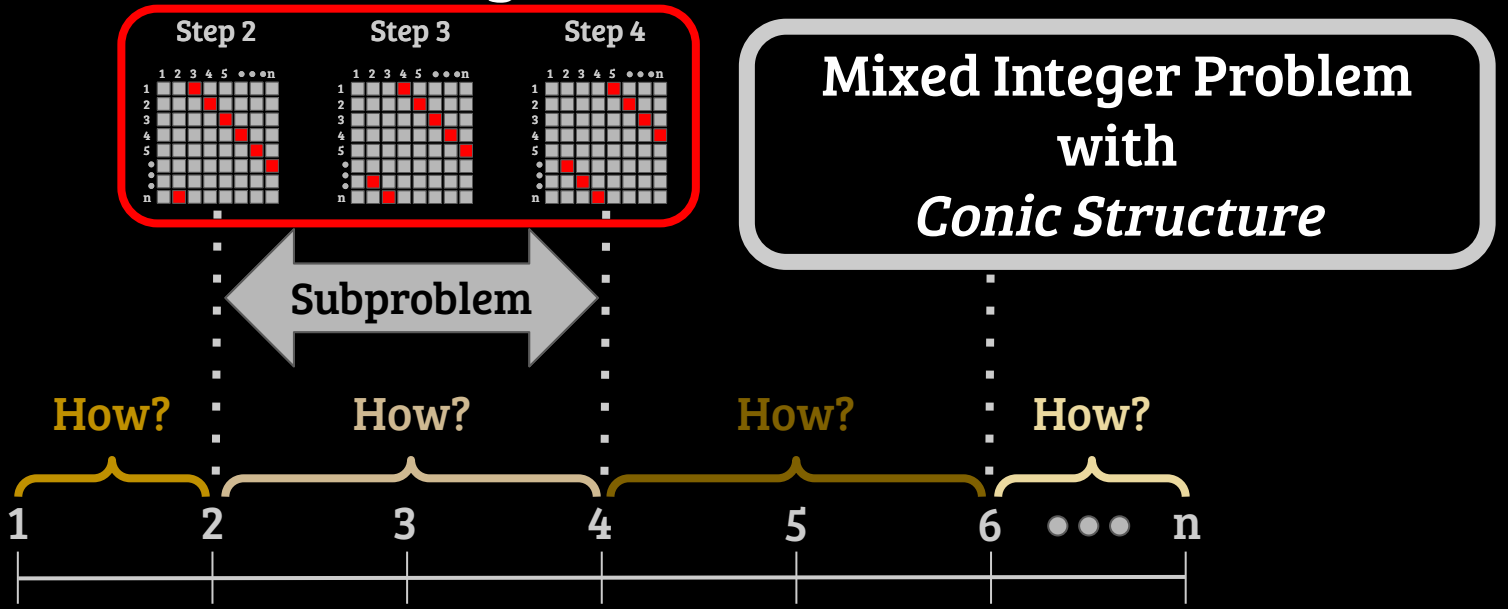
HARVEST

When and How to Reconfigure?



HARVEST

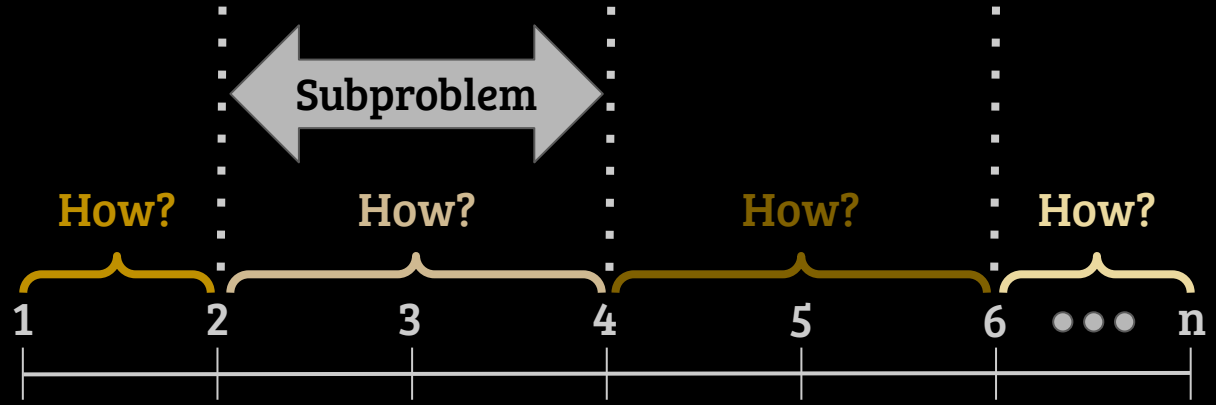
When and How to Reconfigure?



HARVEST

When and How to Reconfigure?

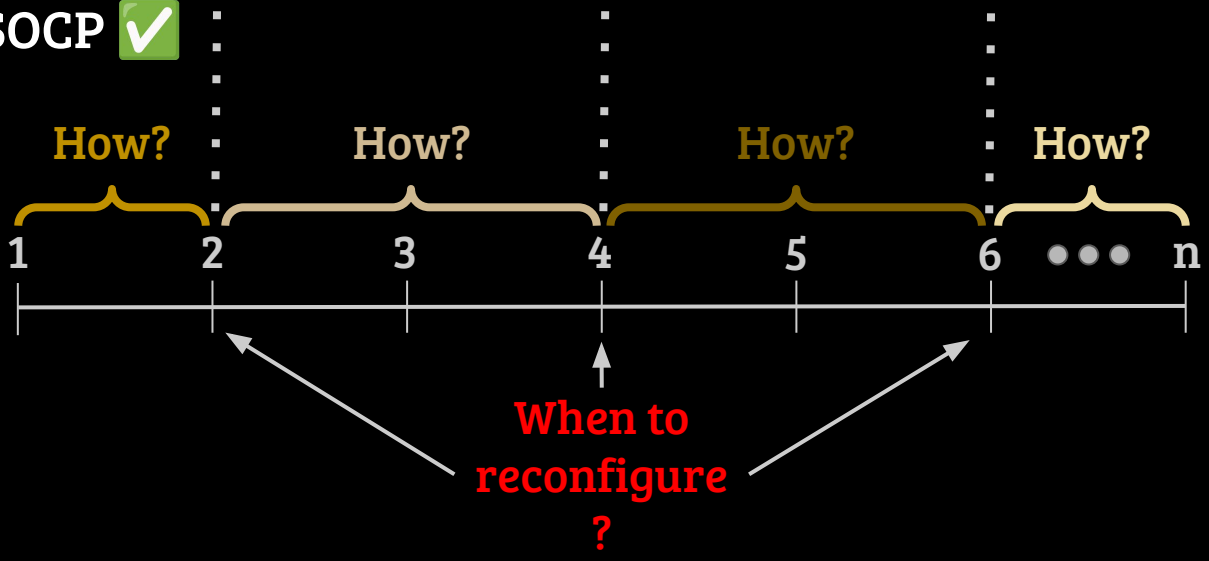
Mixed Integer Second Order Conic Problem



HARVEST

When and How to Reconfigure?

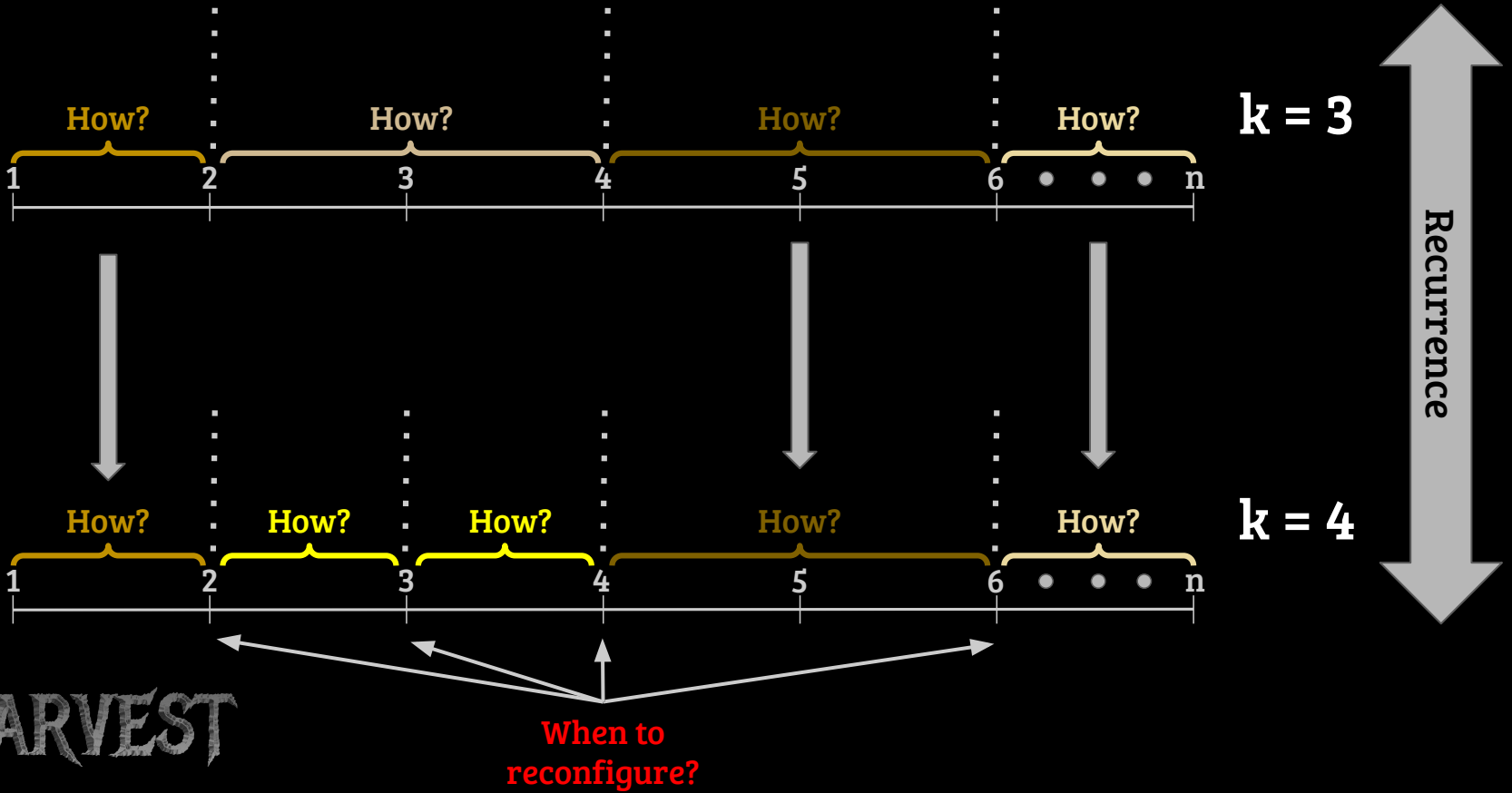
1. When? Partition the sequence of communication steps into intervals
2. ~~How? Find an optimal static topology for each interval~~
   a. MISOCP ✅



HARVEST

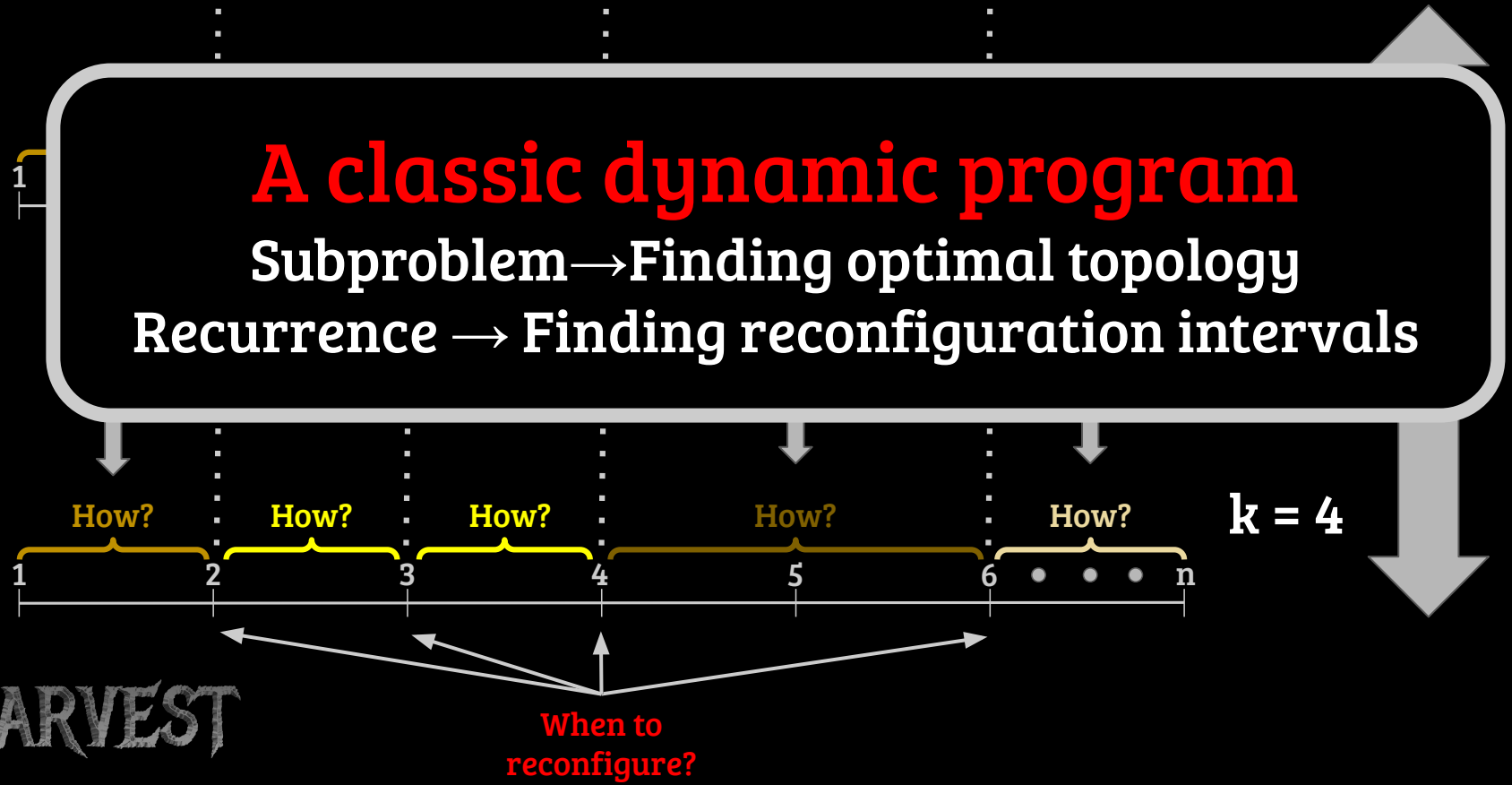
k -> number of reconfigurations

When and How to Reconfigure?



HARVEST

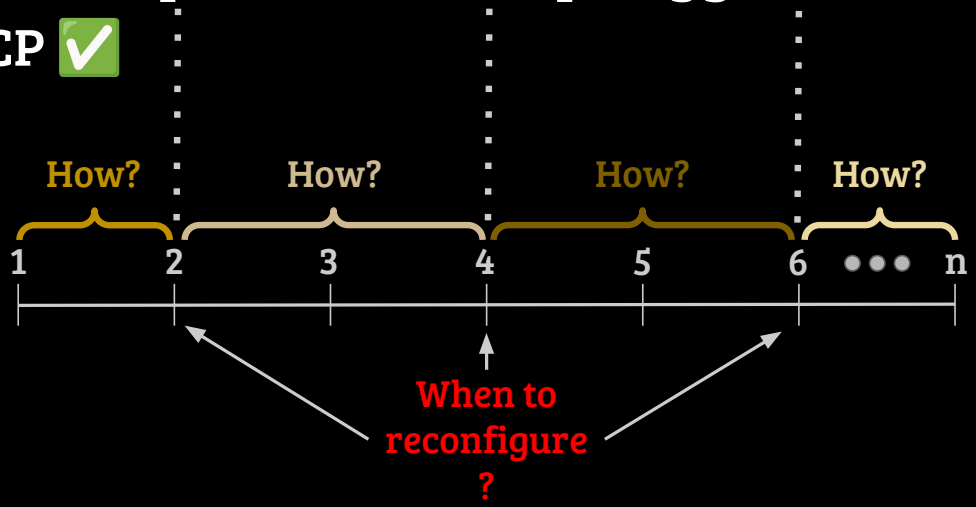
When and How to Reconfigure?



HARVEST

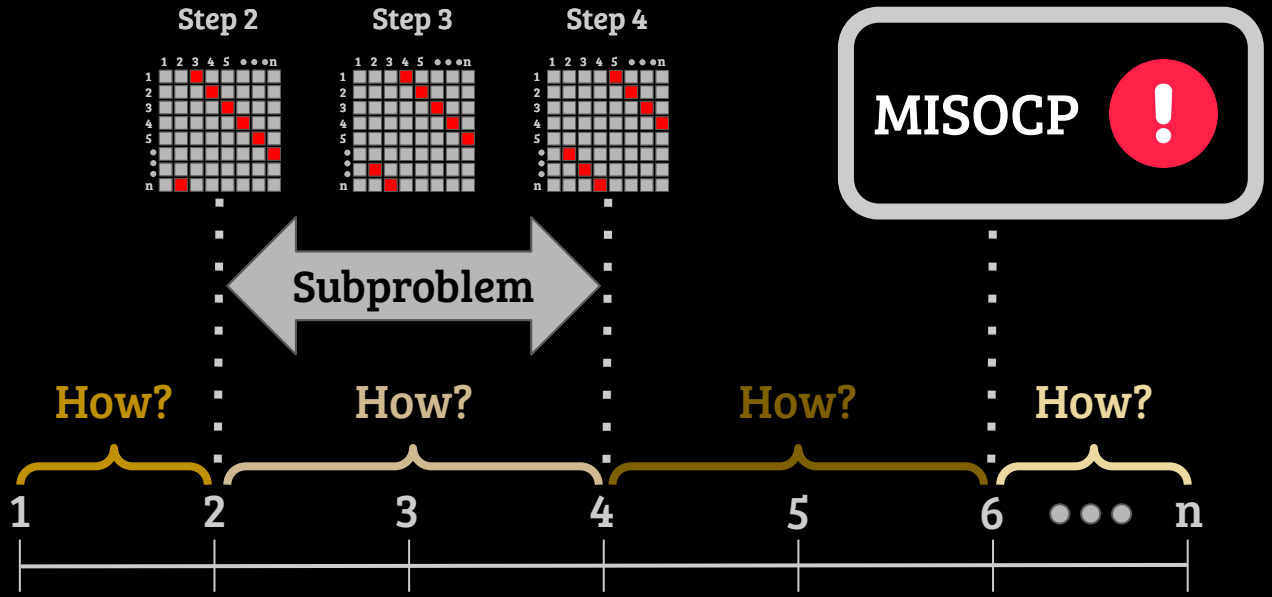
When and How to Reconfigure?

1. ~~When? Partition the sequence of communication steps into intervals~~
   a. Dynamic programming ✅
2. ~~How? Find an optimal static topology for each interval~~
   a. MISOCP ✅



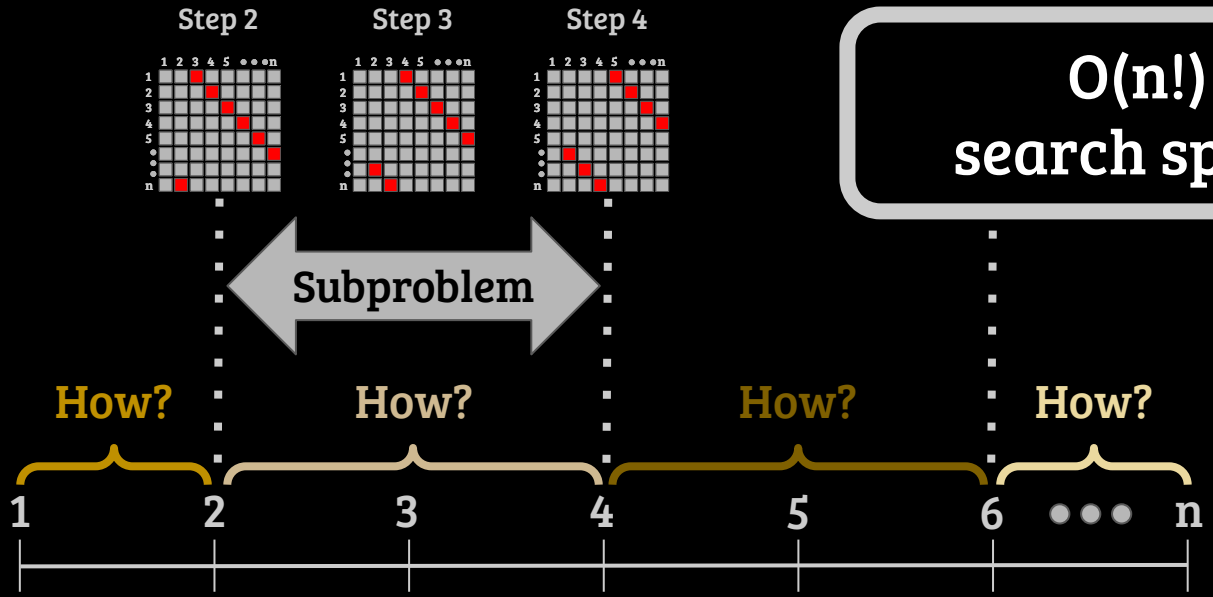
HARVEST

When and How to Reconfigure?



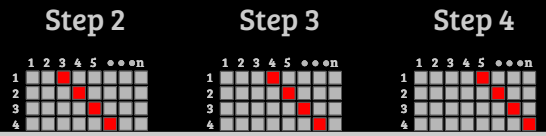
HARVEST

When and How to Reconfigure?



HARVEST

When and How to Reconfigure?



Can we reduce the search space without compromising optimality?

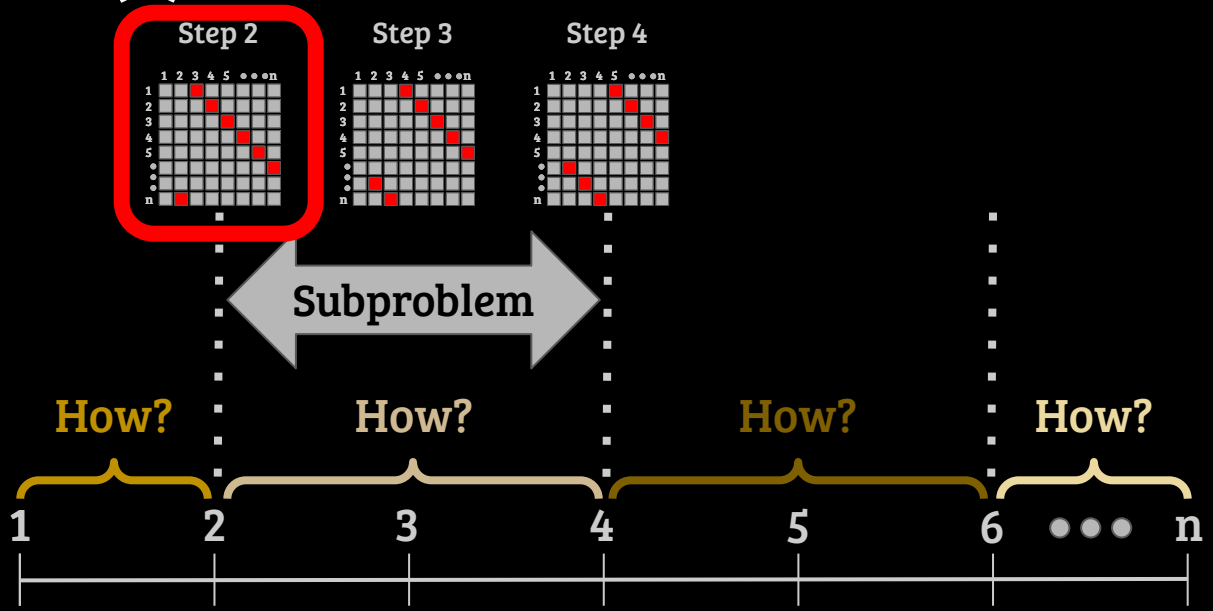
HARVEST

When and How to Reconfigure?

- Collectives with geometric distance progression
  node *i* communicates with node $i + d^k$ in step *k*
  - Recursive doubling: d = 2
  - Bruck's All-to-All and AllReduce: d = 3, 4, …, r
  - Hypercube All-to-All: d = 2
  - Trivance: d = 3

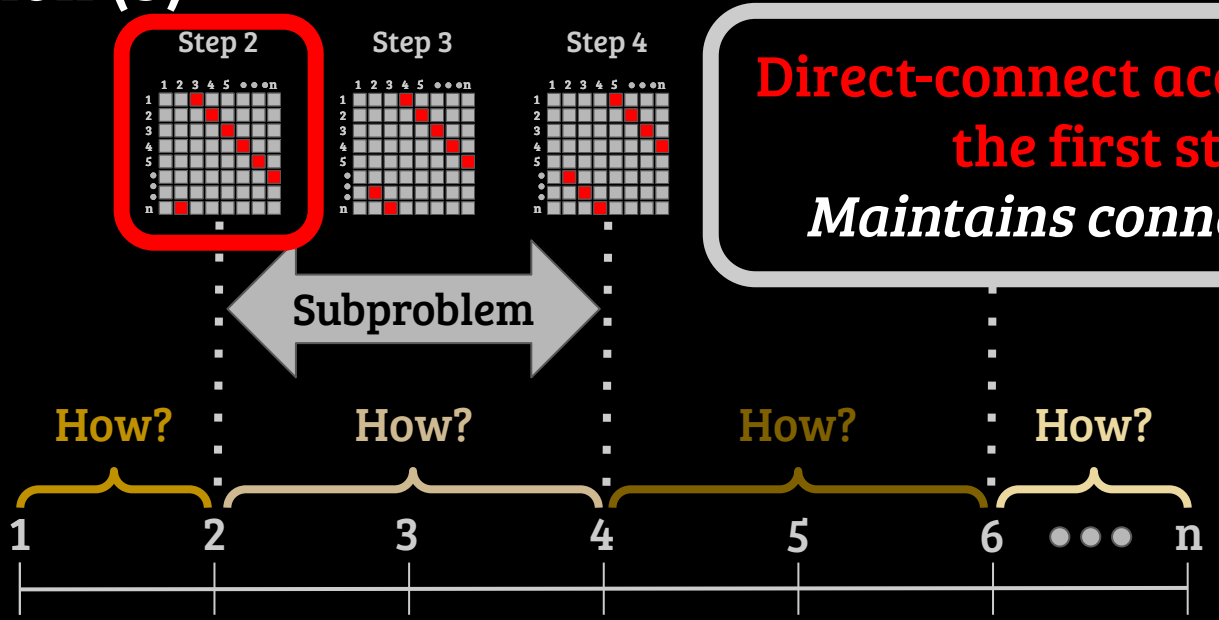
HARVEST

Observation (3)



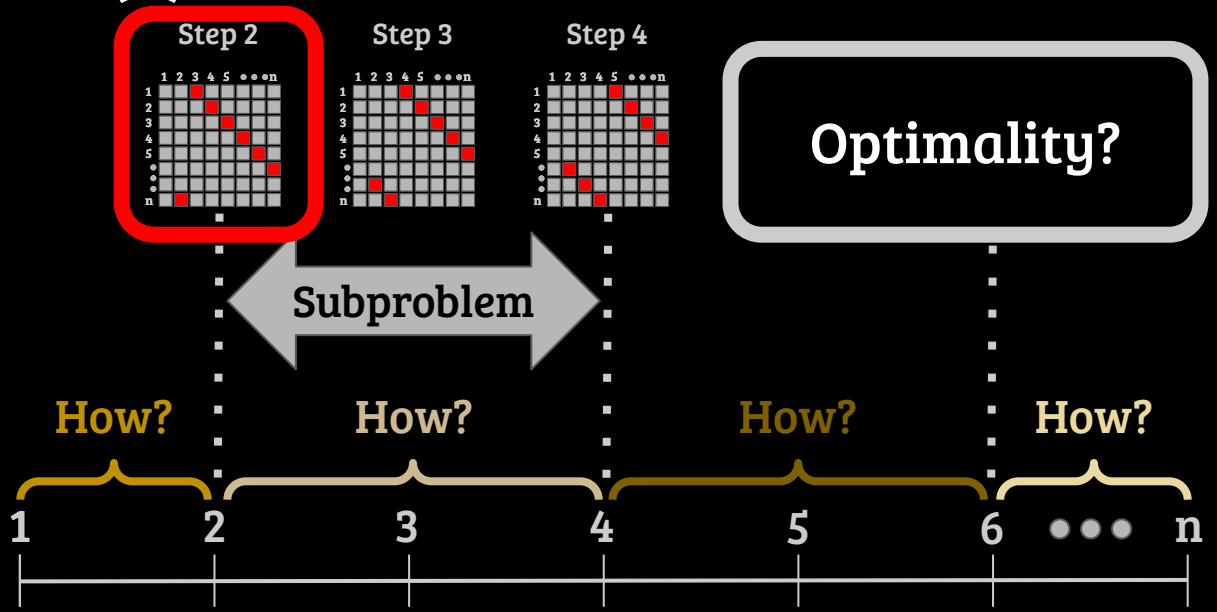
HARVEST

Observation (3)



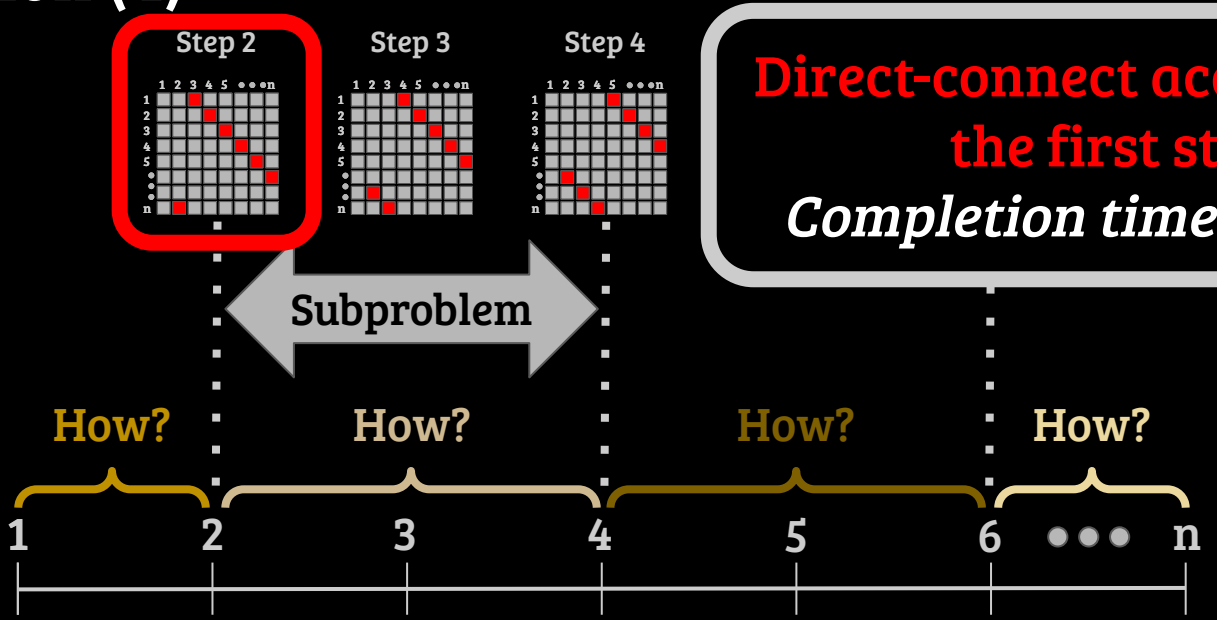
HARVEST

Observation (4)



HARVEST

Observation (4)

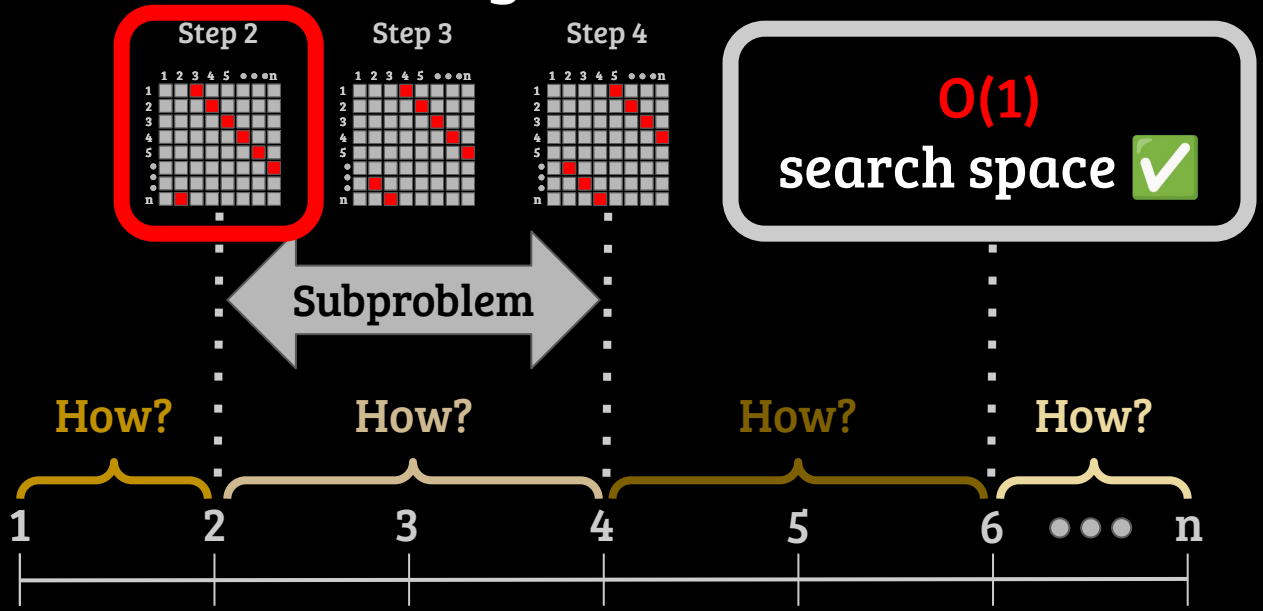


Direct-connect according to the first step

Completion time optimal!

HARVEST

When and How to Reconfigure?



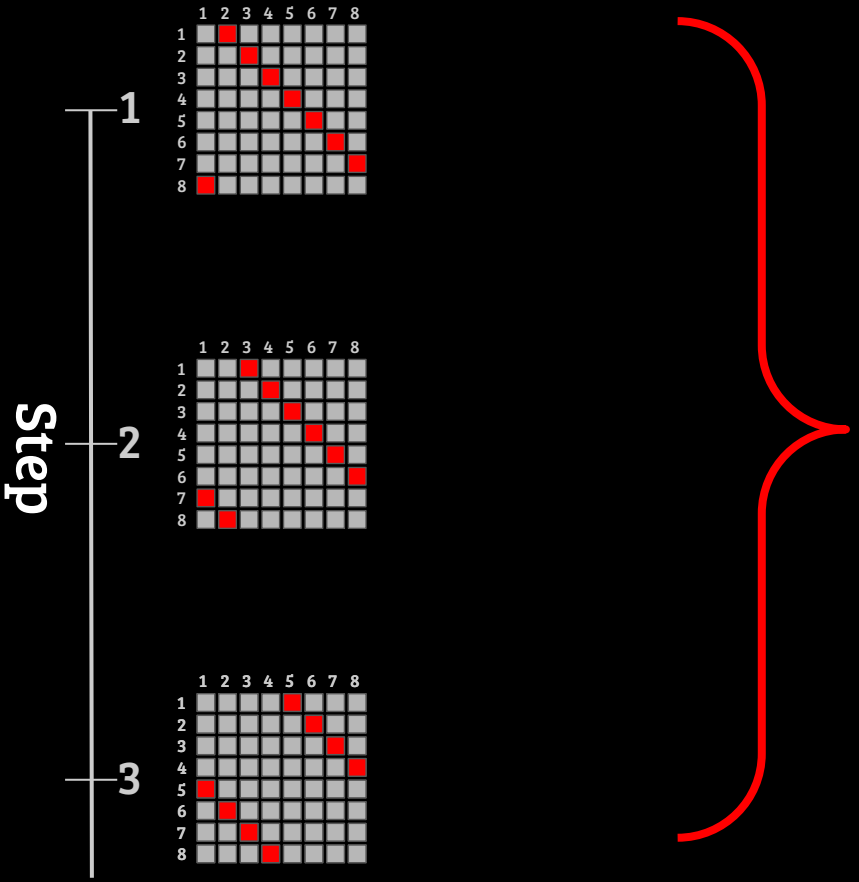
HARVEST

Example: Recursive Doubling (d = 2)

Unidirectional physical link

Communication

Demand Matrix

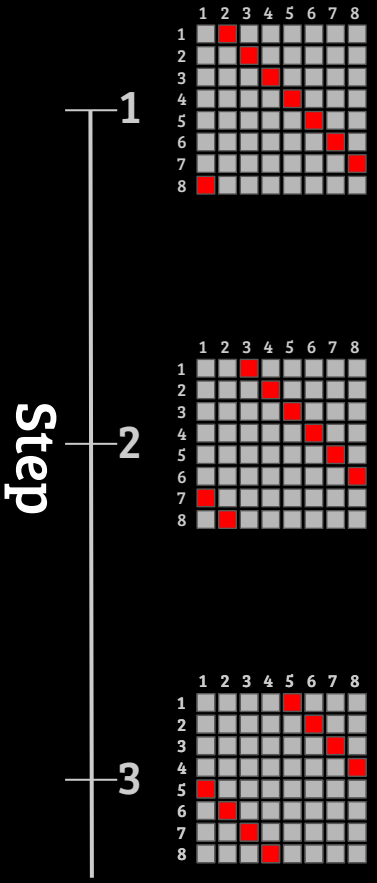


Example: Recursive Doubling (d = 2)

Unidirectional physical link

Communication

Demand Matrix



What is the optimal topology?

Example: Recursive Doubling (d = 2)

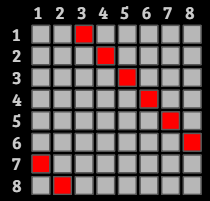
Unidirectional physical link

Communication

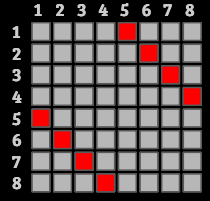
Demand Matrix

Step

1



2



3

Direct-connect according to the first step

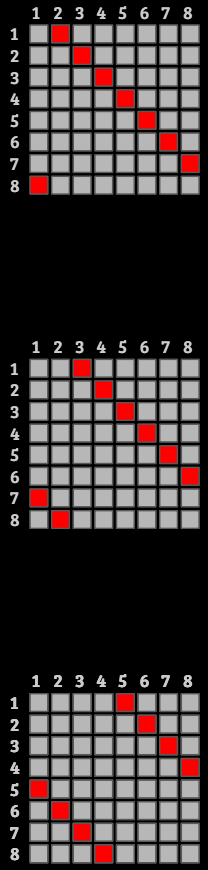
Maintains connectivity & optimal!

Example: Recursive Doubling (d = 2)

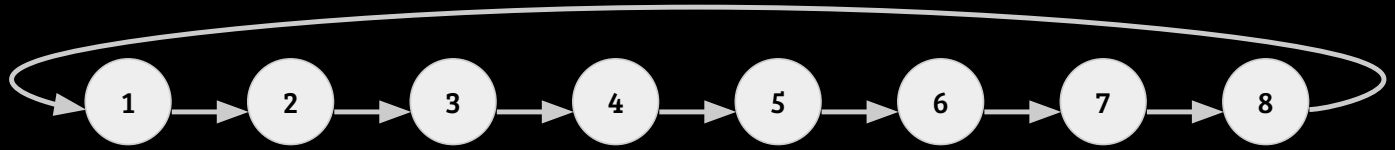
Unidirectional physical link

Communication

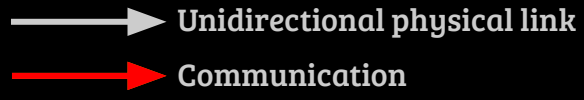
Demand Matrix



Topology



Example: Recursive Doubling (d = 2)



Demand Matrix

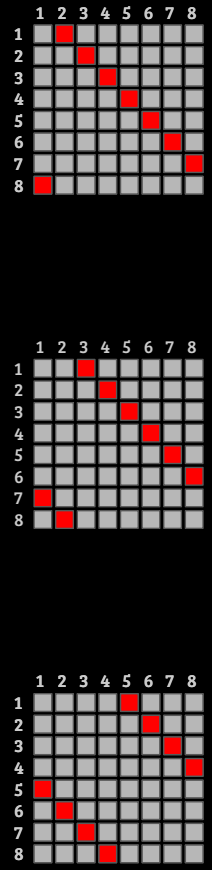
Topology

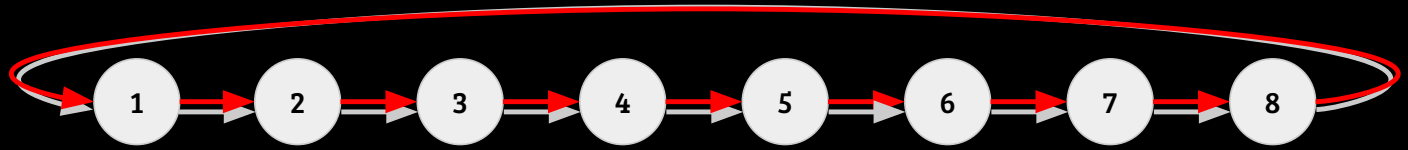
Step

1

2

3





Example: Recursive Doubling (d = 2)

Unidirectional physical link

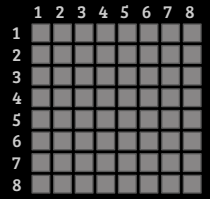
Communication

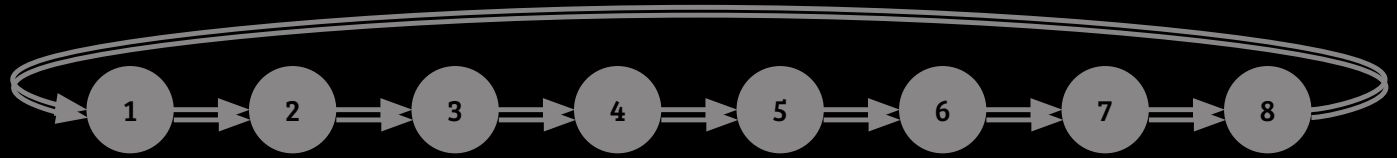
Demand Matrix

Topology

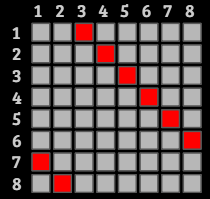
Step

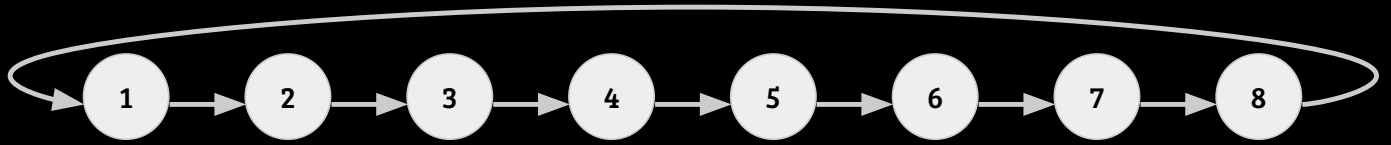
1



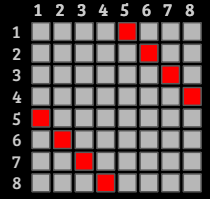


2





3



Example: Recursive Doubling (d = 2)

Unidirectional physical link

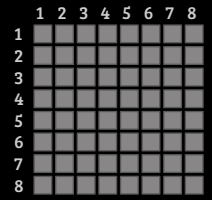
Communication

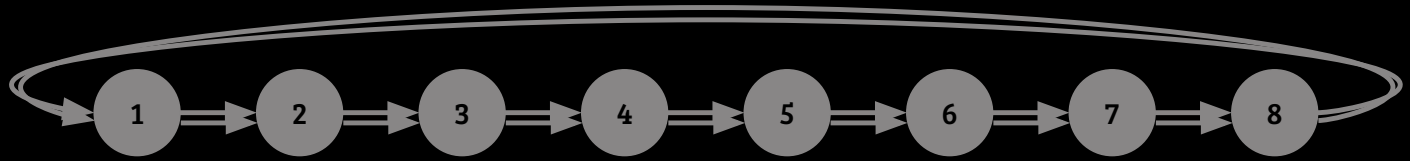
Demand Matrix

Topology

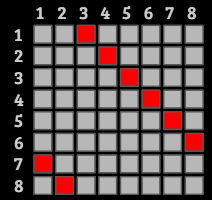
Step

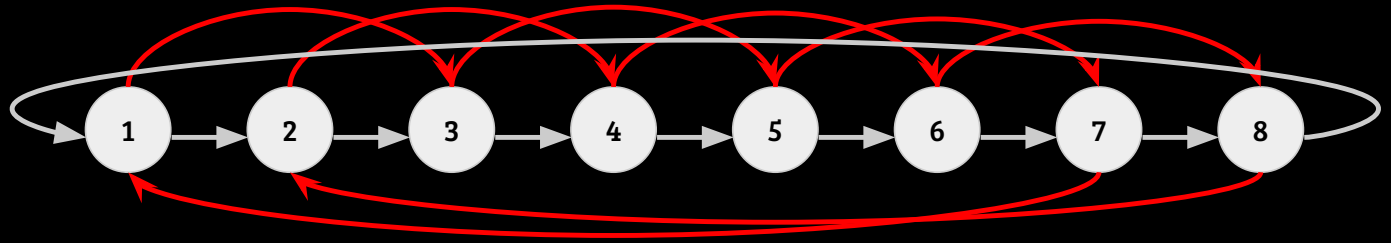
1



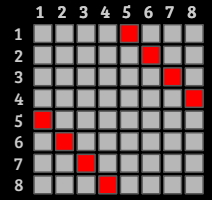


2





3



Example: Recursive Doubling (d = 2)

Unidirectional physical link

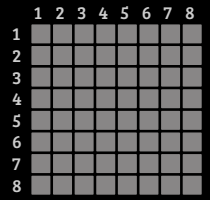
Communication

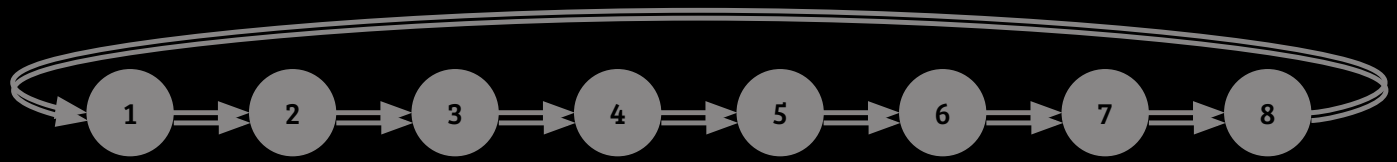
Demand Matrix

Topology

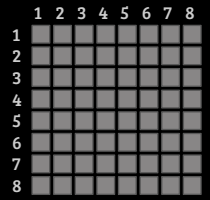
Step

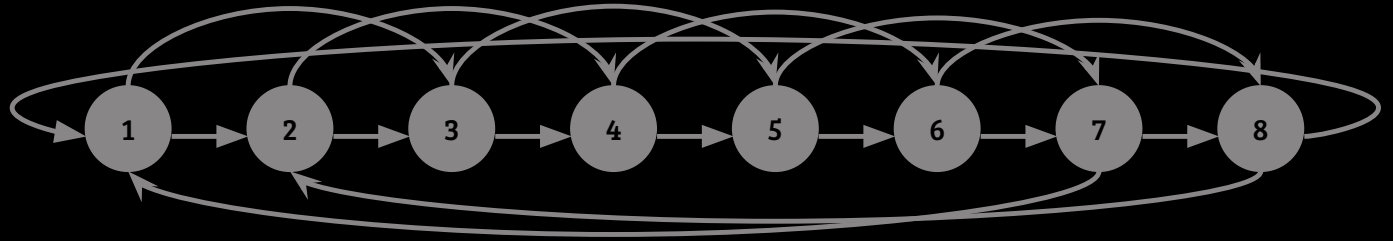
1



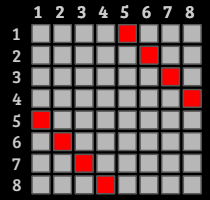


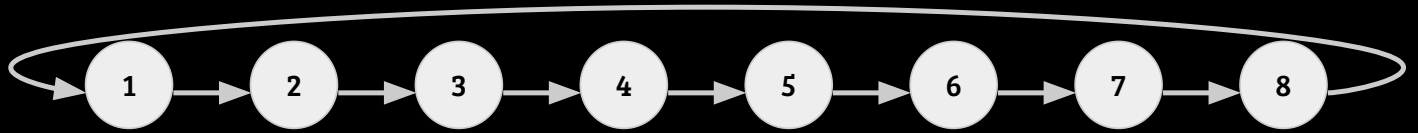
2



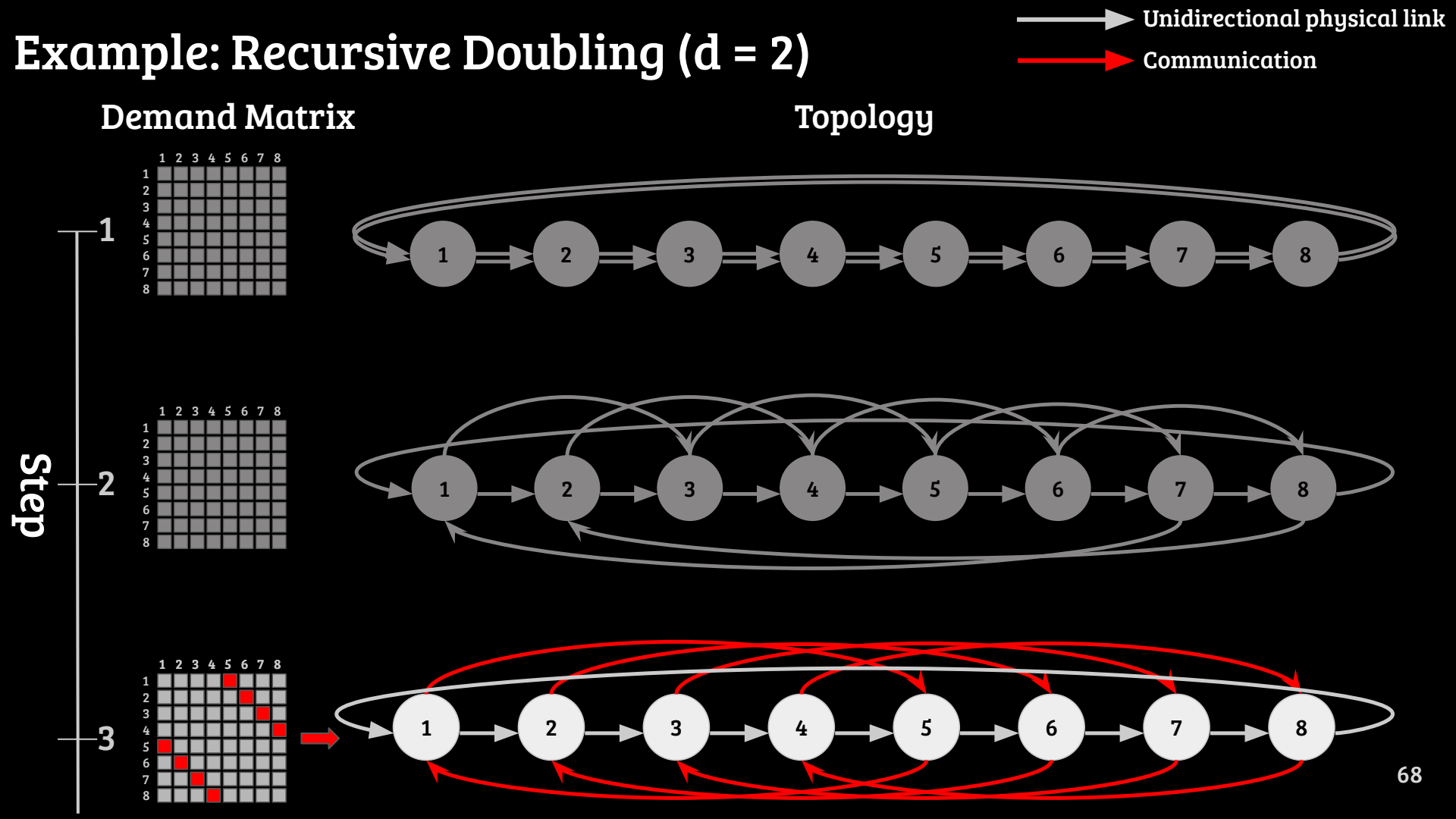


3





Example: Recursive Doubling (d = 2)



Example: Recursive Doubling (d = 2)

Unidirectional physical link

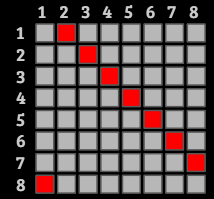
Communication

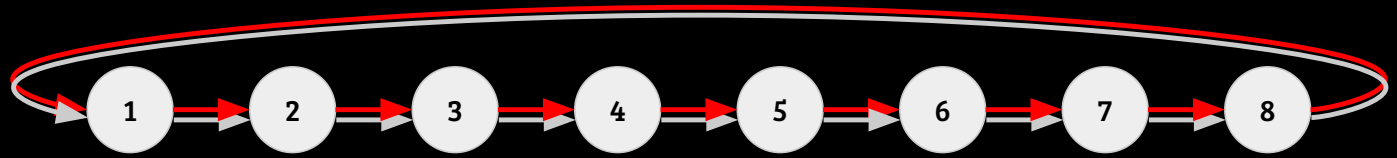
Demand Matrix

Topology

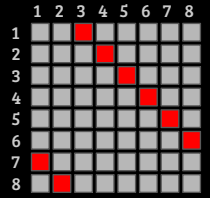
Step

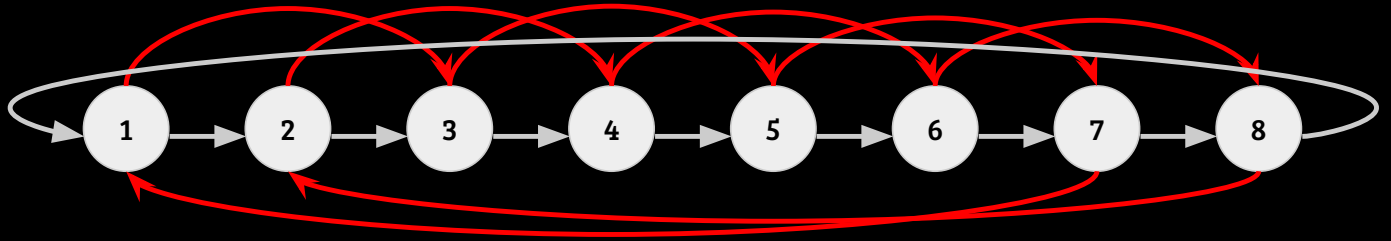
1



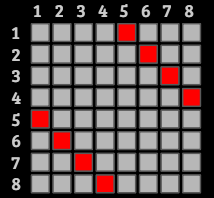


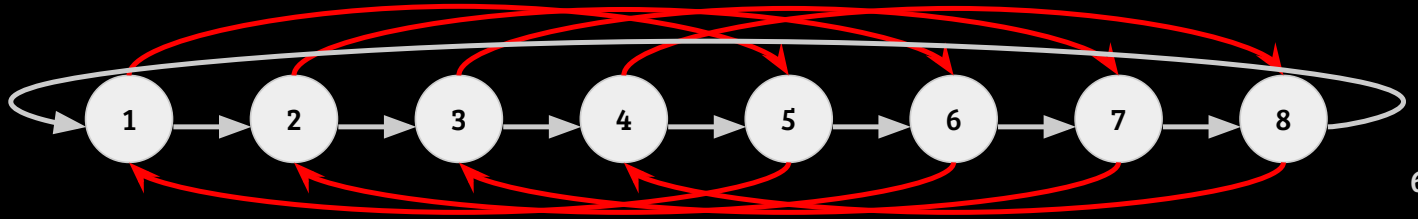
2





3





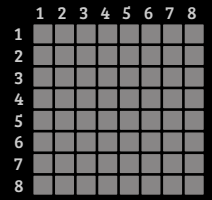
Example: Recursive Doubling (d = 2)

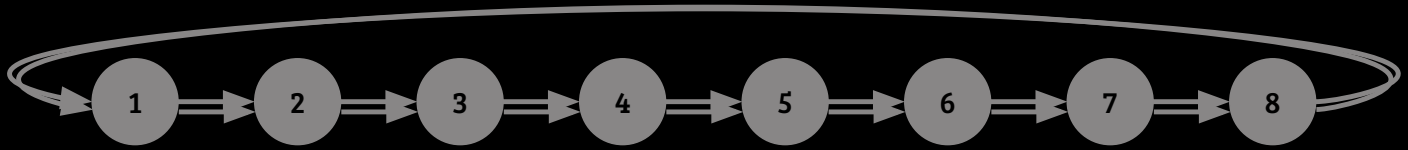
Unidirectional physical link

Communication

Demand Matrix

Topology



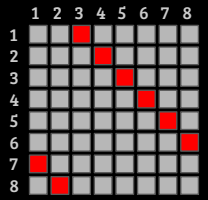


Reconfiguration event

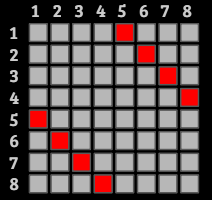
Step

1

2



3



What is the optimal topology?

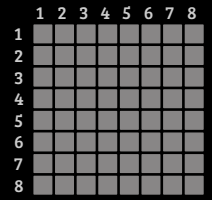
Example: Recursive Doubling (d = 2)

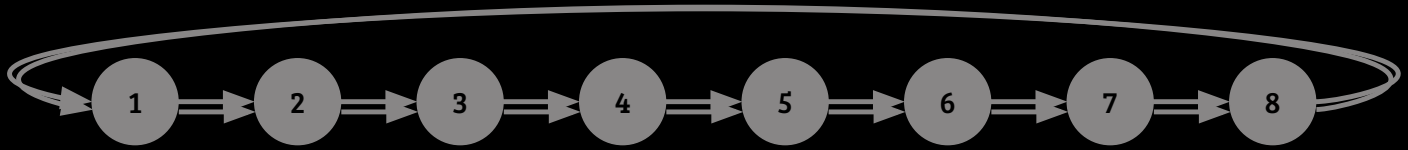
Unidirectional physical link

Communication

Demand Matrix

Topology





Reconfiguration event

Step

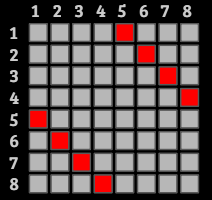
1

2

3

Direct-connect according to the first step

Maintains connectivity & optimal!



Example: Recursive Doubling (d = 2)

Unidirectional physical link

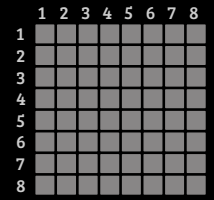
Communication

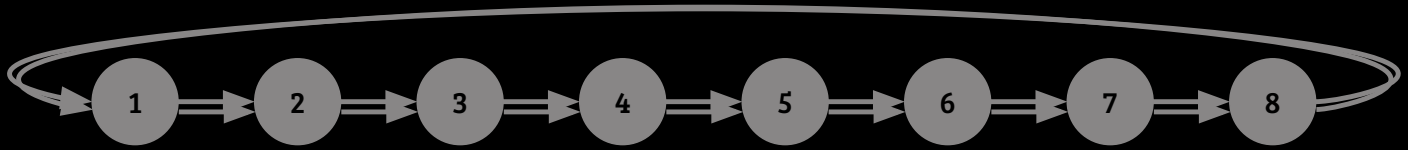
Demand Matrix

Topology

Step

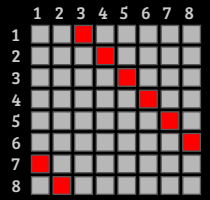
1

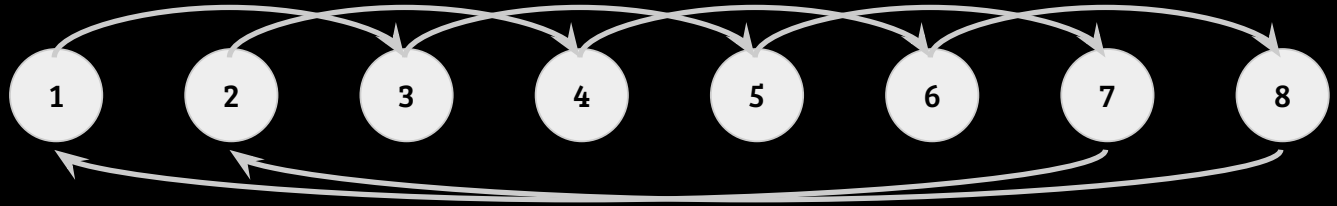




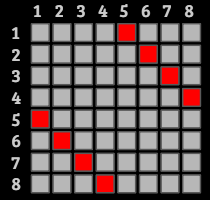
Reconfiguration event

2





3



Example: Recursive Doubling (d = 2)

Unidirectional physical link

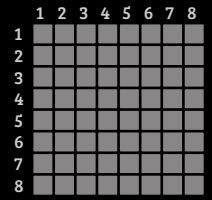
Communication

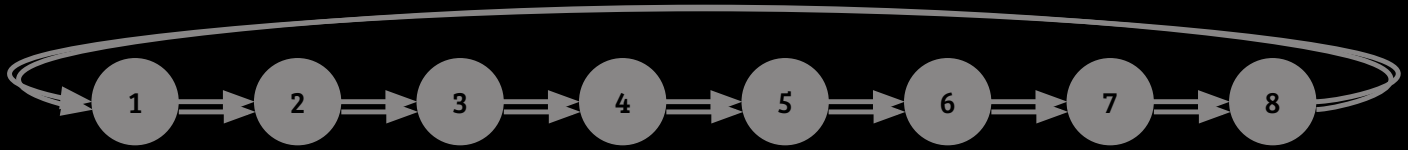
Demand Matrix

Topology

Step

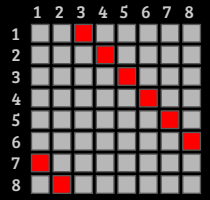
1

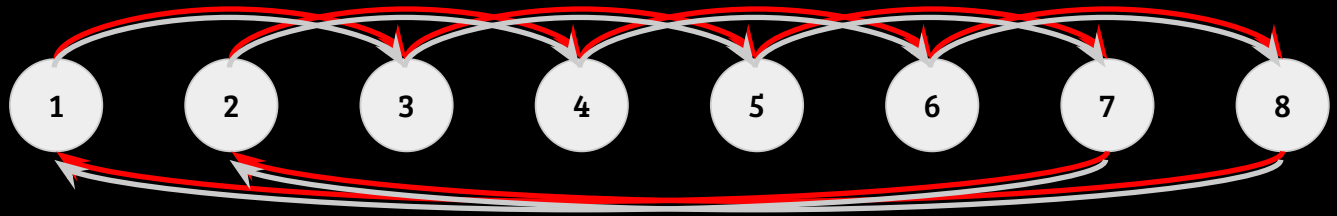




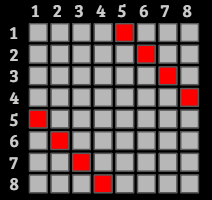
Reconfiguration event

2





3



Example: Recursive Doubling (d = 2)

Unidirectional physical link

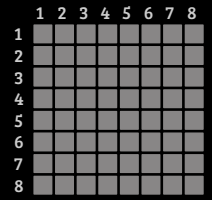
Communication

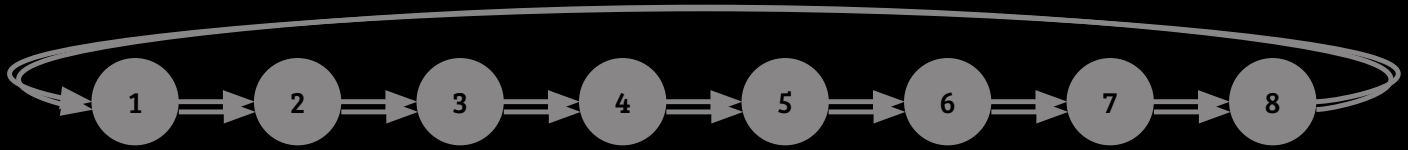
Demand Matrix

Topology

Step

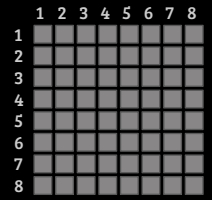
1

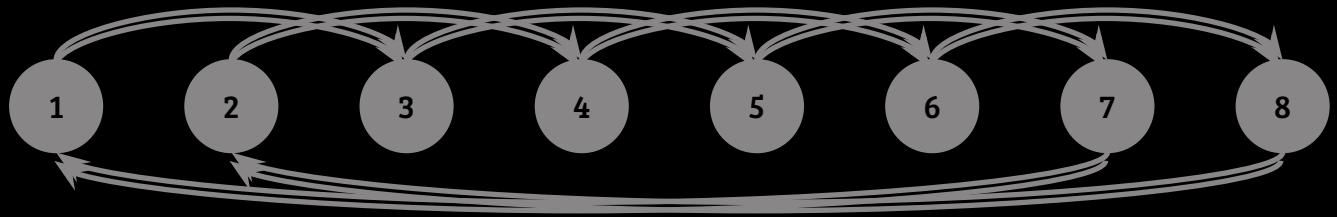




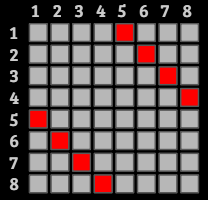
Reconfiguration event

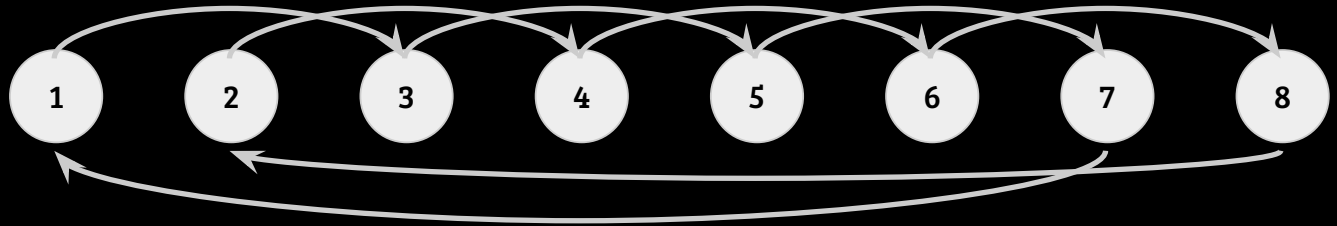
2





3





Example: Recursive Doubling (d = 2)

Unidirectional physical link

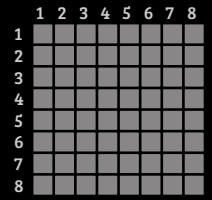
Communication

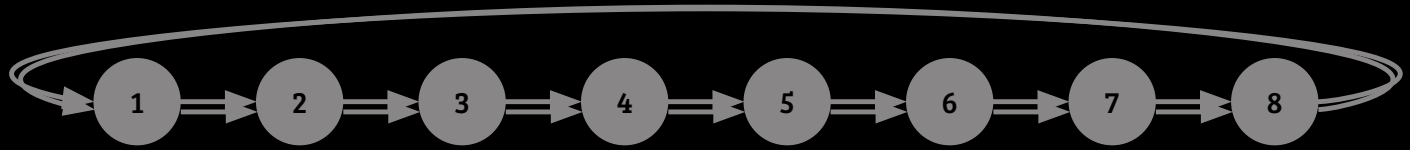
Demand Matrix

Topology

Step

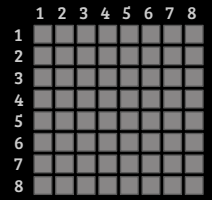
1

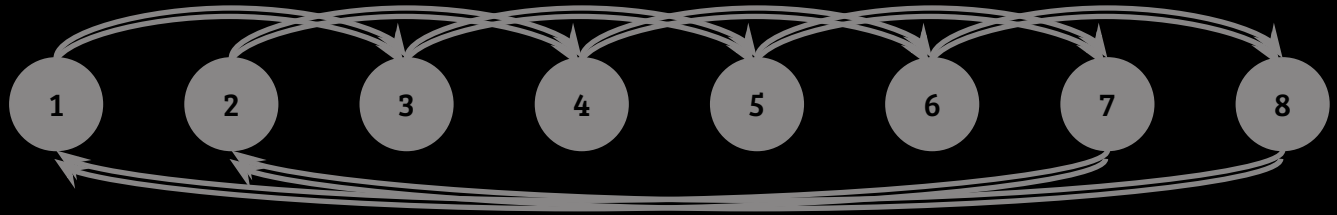




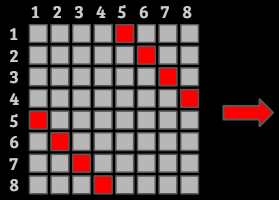
Reconfiguration event

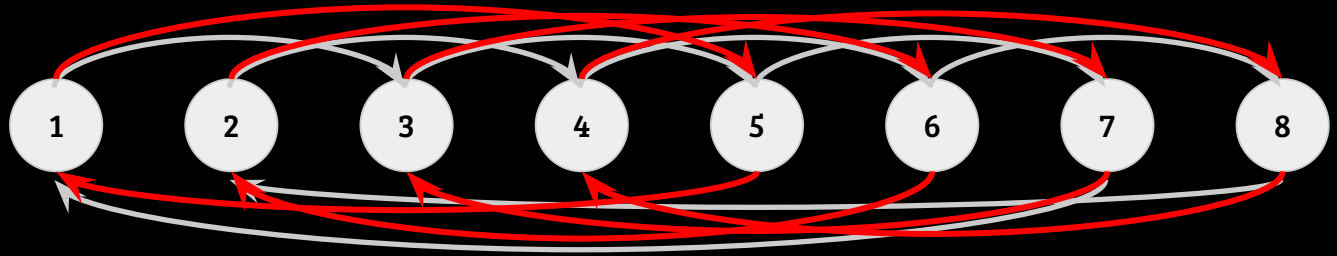
2





3





Example: Recursive Doubling (d = 2)

Unidirectional physical link

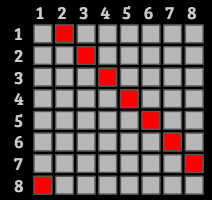
Communication

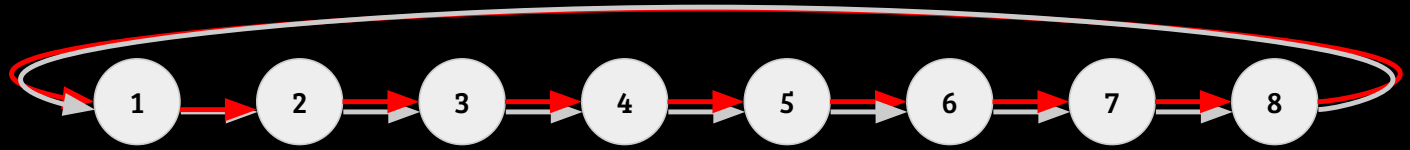
Demand Matrix

Topology

Step

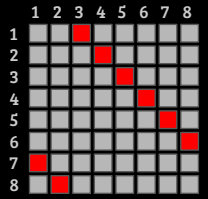
1

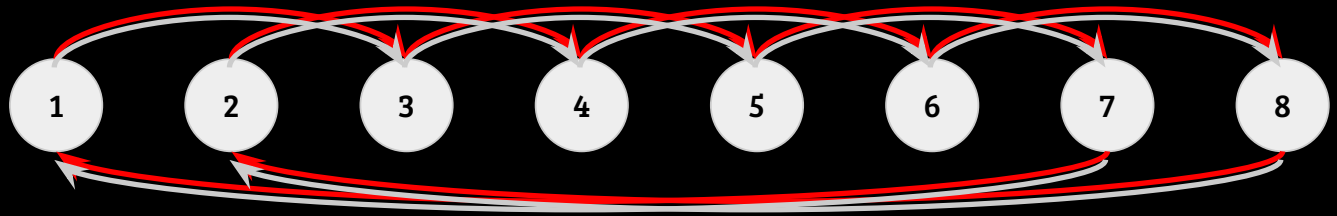




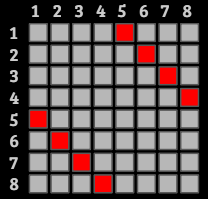
Reconfiguration event

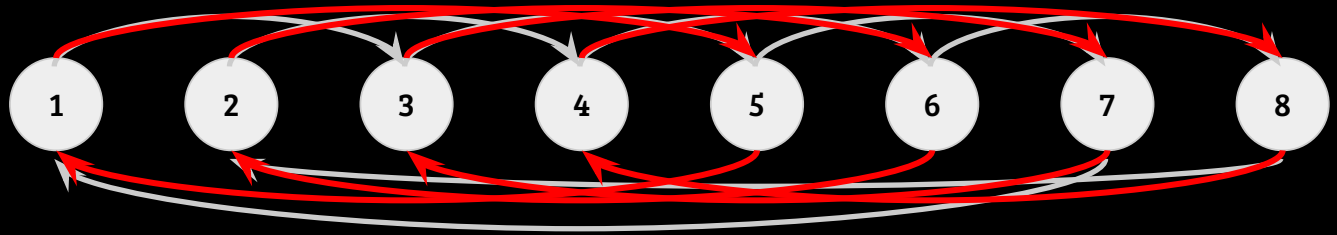
2





3





When and How to Reconfigure?

- Collectives with geometric distance progression
  node *i* communicates with node $i + d^k$ in step *k*
  - Recursive doubling: d = 2
  - Bruck's All-to-All and AllReduce: d = 3, 4, …, r
  - Hypercube All-to-All: d = 2
  - Trivance: d = 3

HARVEST

When and How to Reconfigure?

1. ~~When? Partition the sequence of communication steps into intervals~~
   a. Dynamic programming ✅
2. ~~How? Find an optimal static topology for each interval~~
   a. MISOCP ✅ (For generality)
   b. Direct-connect ✅ (For collectives with geometric distance progression)

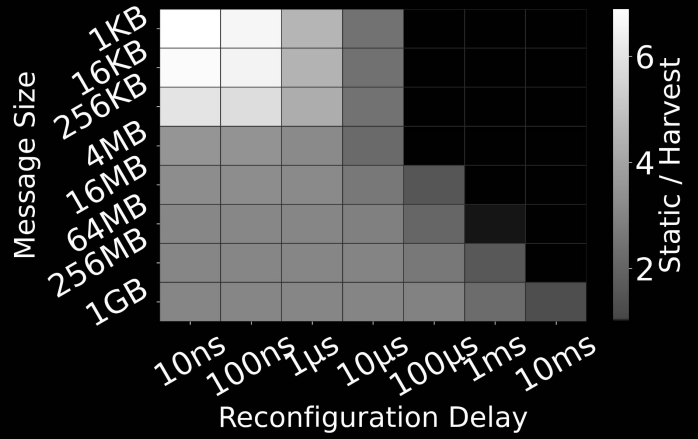
HARVEST

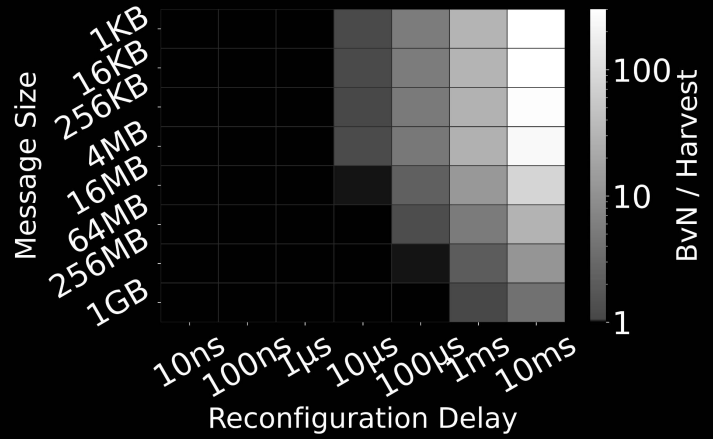
Evaluation Methodology

- We compare performance of Harvest against two baselines:
  - A static topology
  - Schedules that reconfigure every step (BvN)
- Network sizes: 8 to 64 GPUs
- Reconfiguration delays: 10 ns to 10 ms
- Message sizes: 1 KB to 1 GB

Speed-Up for Recursive Doubling AllReduce

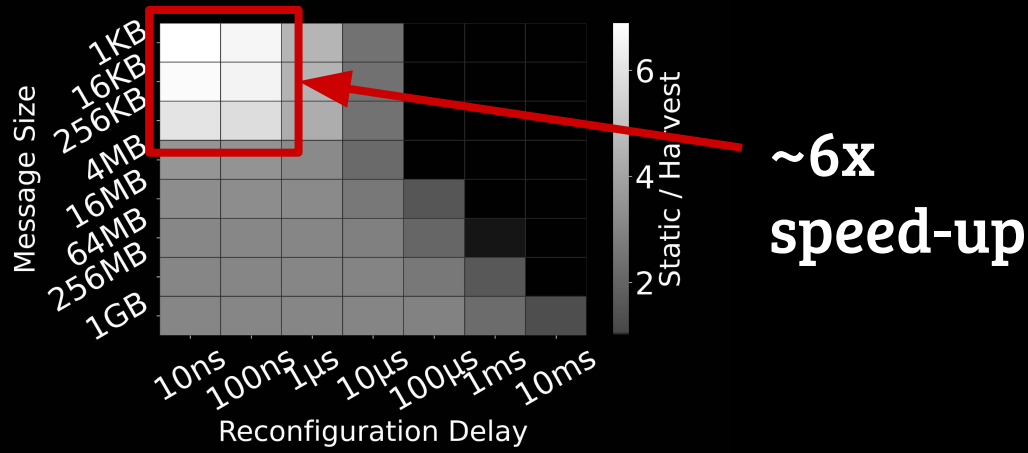
- Baseline/Harvest Ratio > 1 –> Harvest outperforms
- Baseline/Harvest Ratio == 1 –> Harvest matches performance





Speed-Up for Recursive Doubling AllReduce

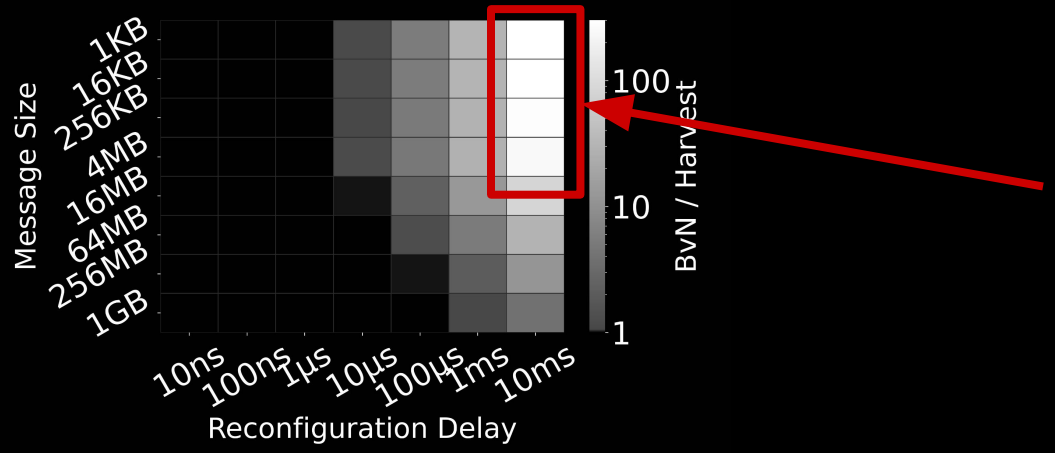
- Static vs Harvest
- Low message size and low reconfiguration delay domains show more than 6x speed-up over Static



HARVEST

Speed-Up for Recursive Doubling AllReduce

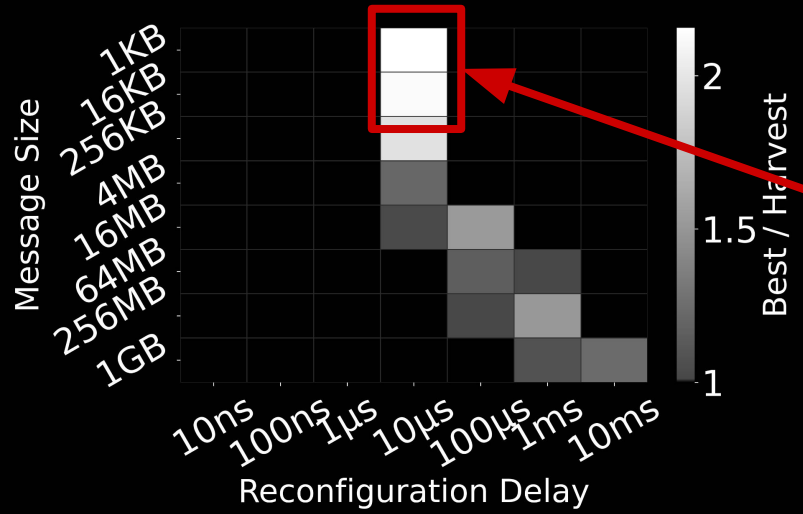
- Reconfigure every step schedule (BvN) vs Harvest
- Low message size and high reconfiguration delay domains show significant speed-up over BvN



HARVEST

Speed-Up for Recursive Doubling AllReduce

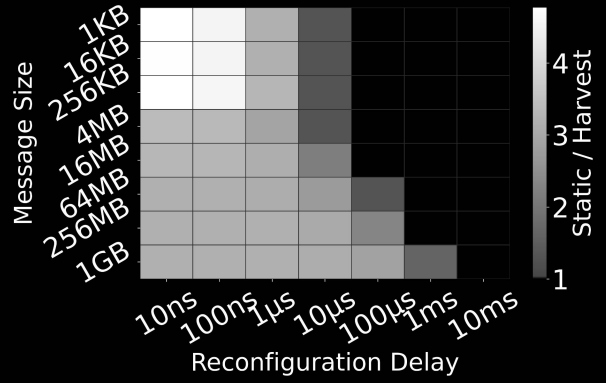
- Harvest vs Best of (Static, BvN)
- A new space where Harvest outperforms both



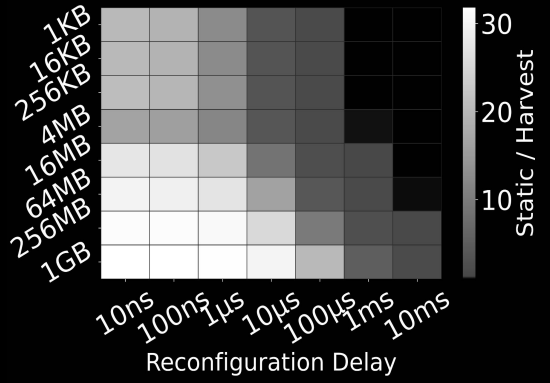
~2x speed-up

HARVEST

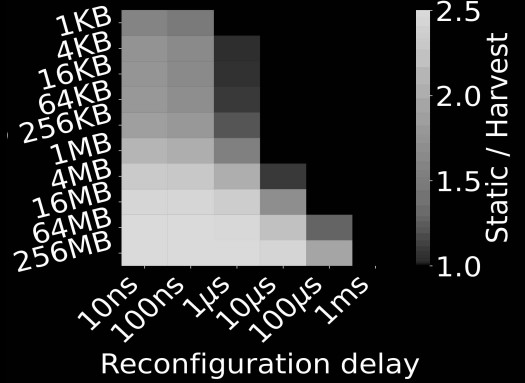
Performance comparison to other collectives



Swing



Cyclic All-to-All



Bruck r = 4 AllGather

And many more…

HARVEST

Conclusion

- There is a need to balance network congestion and reconfiguration delay to minimize collective completion time
- We present an optimization framework to synthesize optimal topologies
- O(1) reduction in topology search space for most well-known algorithms

HARVEST

Thank You

Mahir Rahman
rahman77@purdue.edu

HARVEST