Towards Topology-Hiding Computation from Oblivious Transfer

Marshall Ball, Alexander Bienstock, Lisa Kohl, Pierre Meyer

TCC 2023

What is Topology-Hiding Computation (THC)?

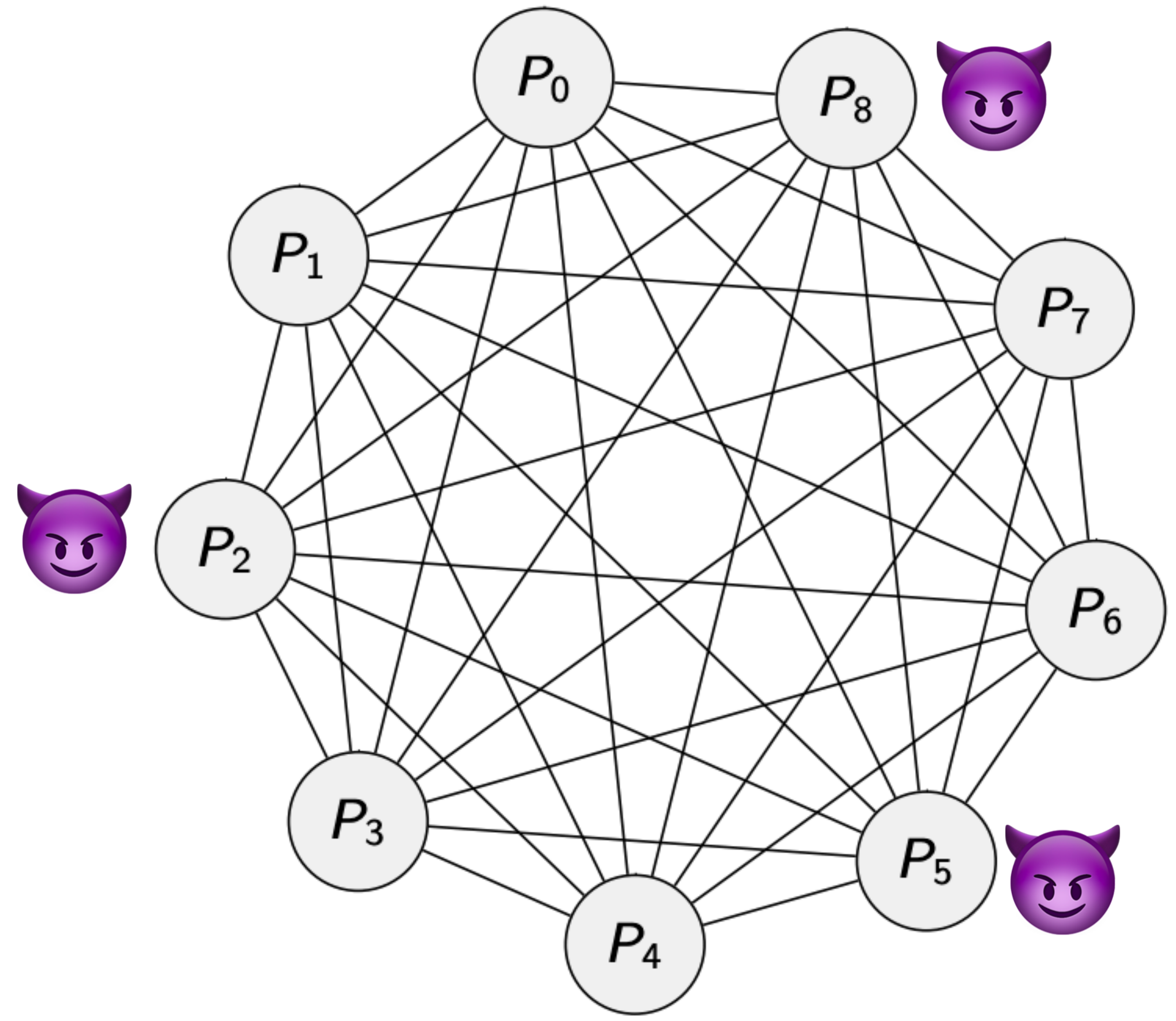
What is Topology-Hiding Computation (THC)?

Starting Point: Secure Multi-Party Computation

What is Topology-Hiding Computation (THC)?

Starting Point: Secure Multi-Party Computation

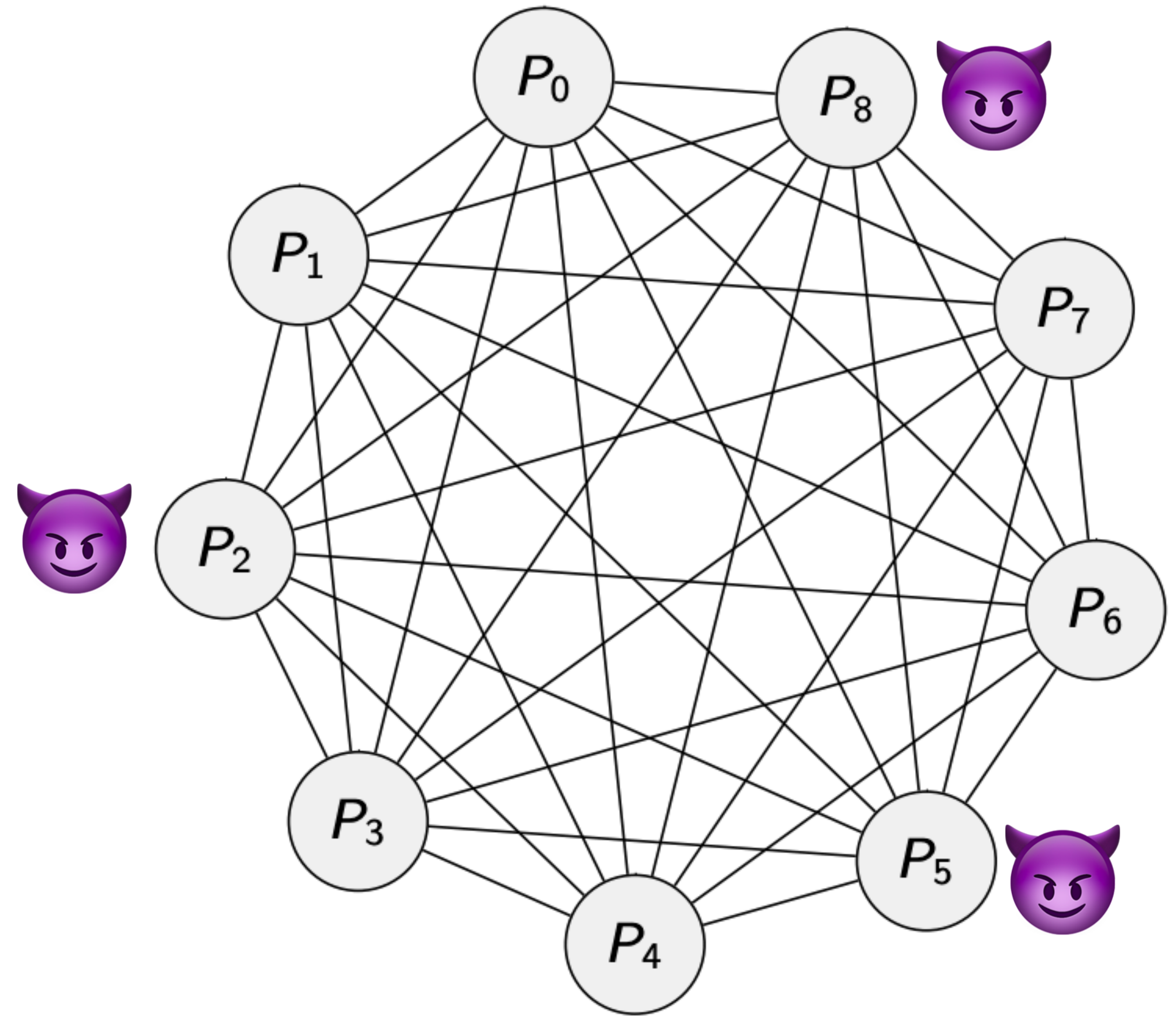
- Parties $P_0, \ldots, P_{n-1}$ with inputs $x_0, \ldots, x_{n-1}$ *privately* compute function $\mathcal{F}(x_0, \ldots, x_{n-1})$
    - Adversary (corrupting possibly many parties) learns nothing beyond inputs of corrupted parties and output



What is Topology-Hiding Computation (THC)?

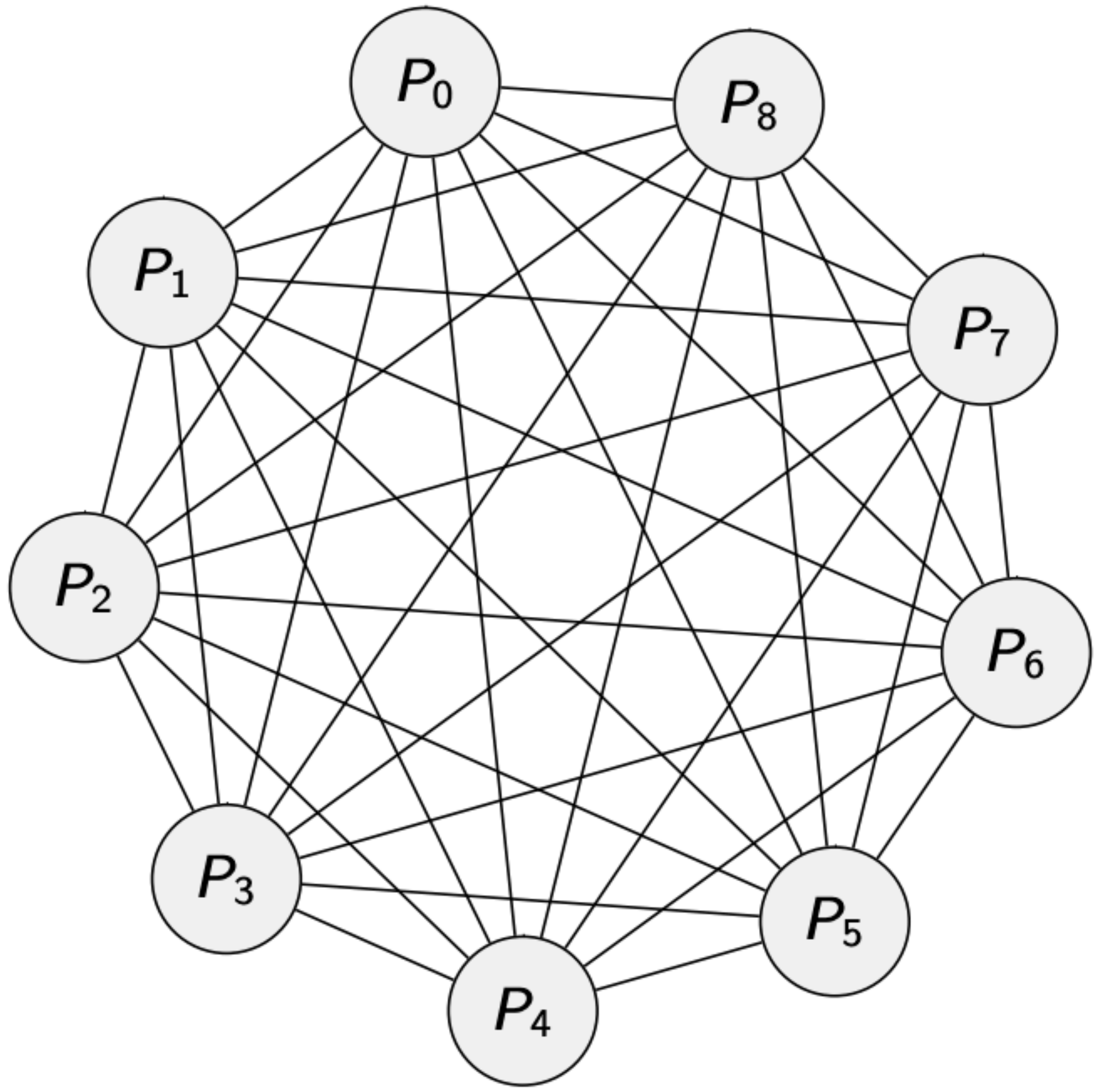
Starting Point: Secure Multi-Party Computation

- Parties $P_0, \ldots, P_{n-1}$ with inputs $x_0, \ldots, x_{n-1}$ *privately* compute function $\mathcal{F}(x_0, \ldots, x_{n-1})$
  - Adversary (corrupting possibly many parties) learns nothing beyond inputs of corrupted parties and output
- Typically -- assume secure point-to-point communication channels between all parties



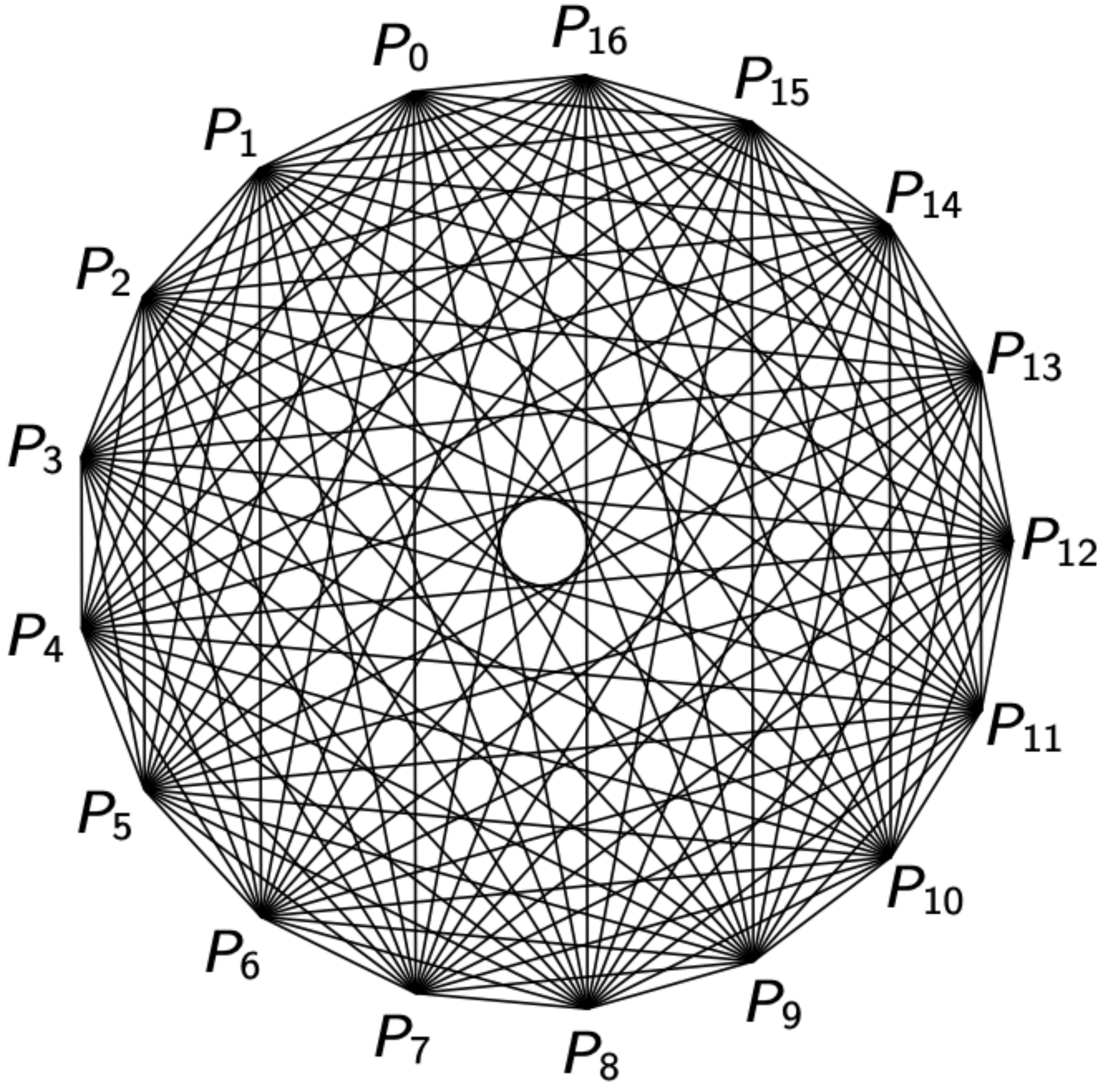
What is Topology-Hiding Computation (THC)?

Connectivity in MPC



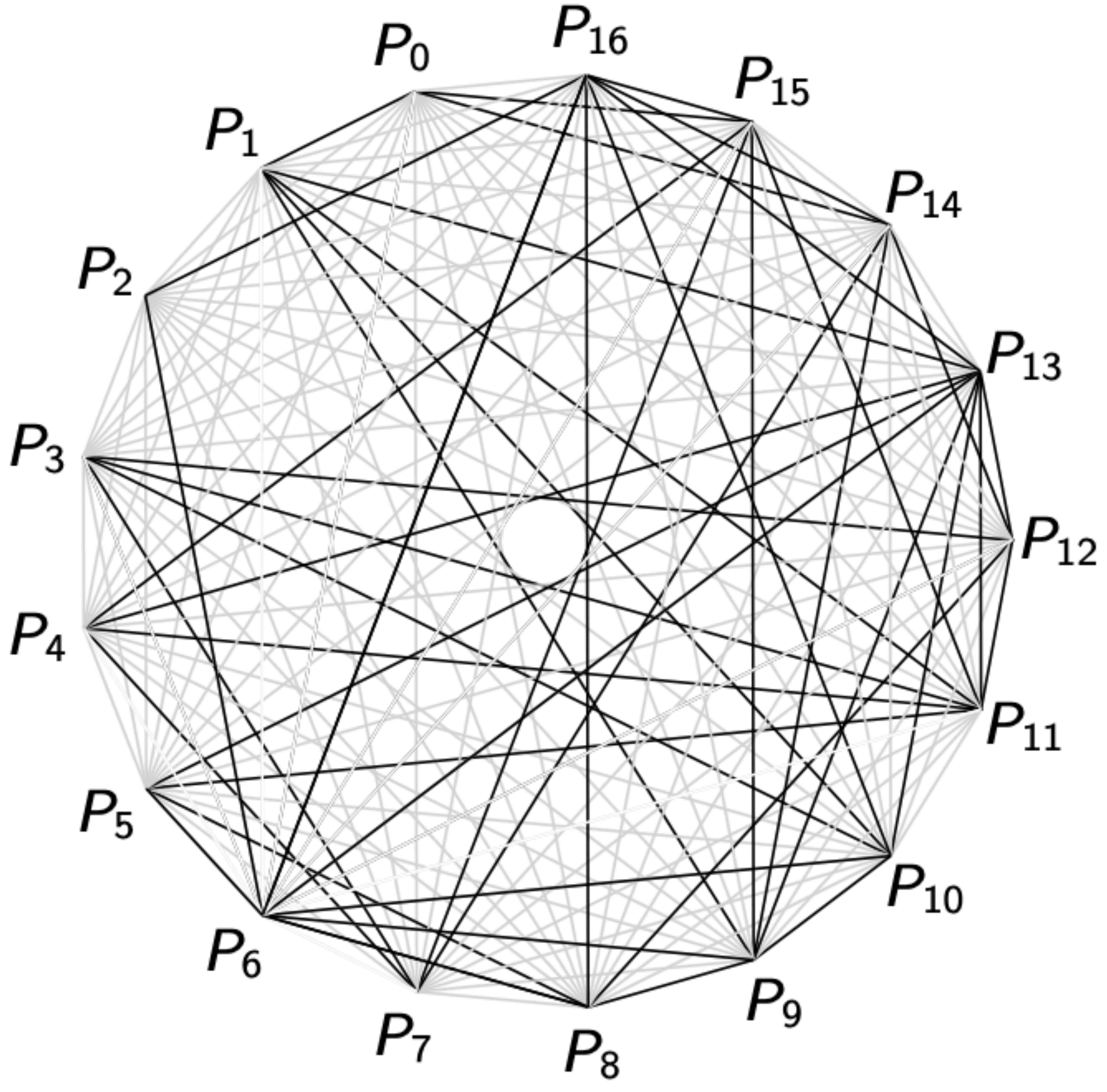
What is Topology-Hiding Computation (THC)?

Connectivity in MPC



What is Topology-Hiding Computation (THC)?

Connectivity in MPC



What is Topology-Hiding Computation (THC)?

Topology Privacy

What is Topology-Hiding Computation (THC)?

Topology Privacy

- Should we keep features of the topology private?

What is Topology-Hiding Computation (THC)?

Topology Privacy

- Should we keep features of the topology private?
  - Maybe topology is based on location data (e.g., vehicle-to-vehicle comms)

What is Topology-Hiding Computation (THC)?

Topology Privacy

- Should we keep features of the topology private?
  - Maybe topology is based on location data (e.g., vehicle-to-vehicle comms)
  - Maybe topology is based on users' relationships (e.g., social networks)

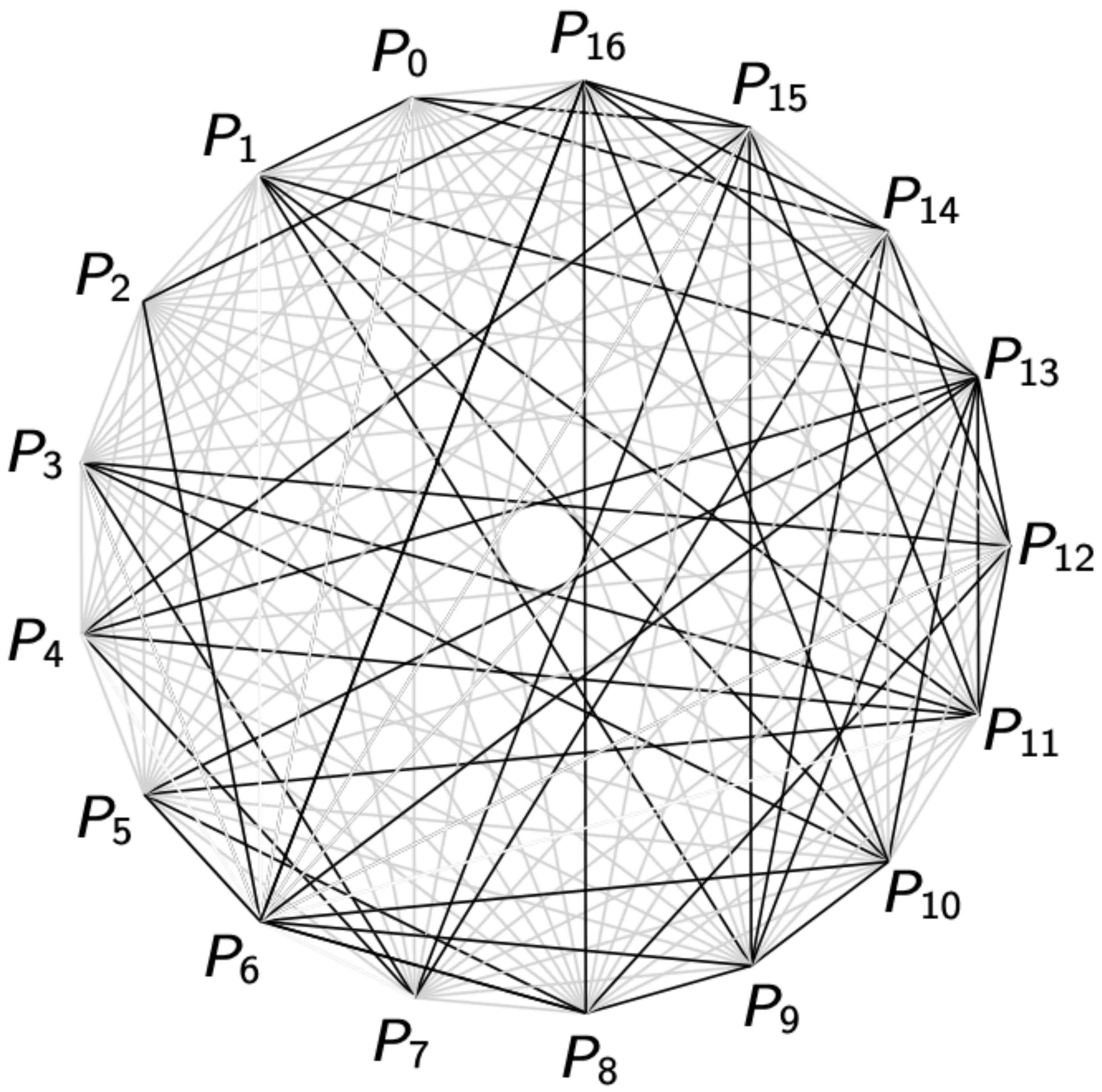
What is Topology-Hiding Computation (THC)?

Topology Privacy

- Should we keep features of the topology private?
  - Maybe topology is based on location data (e.g., vehicle-to-vehicle comms)
  - Maybe topology is based on users' relationships (e.g., social networks)
- What does this mean?

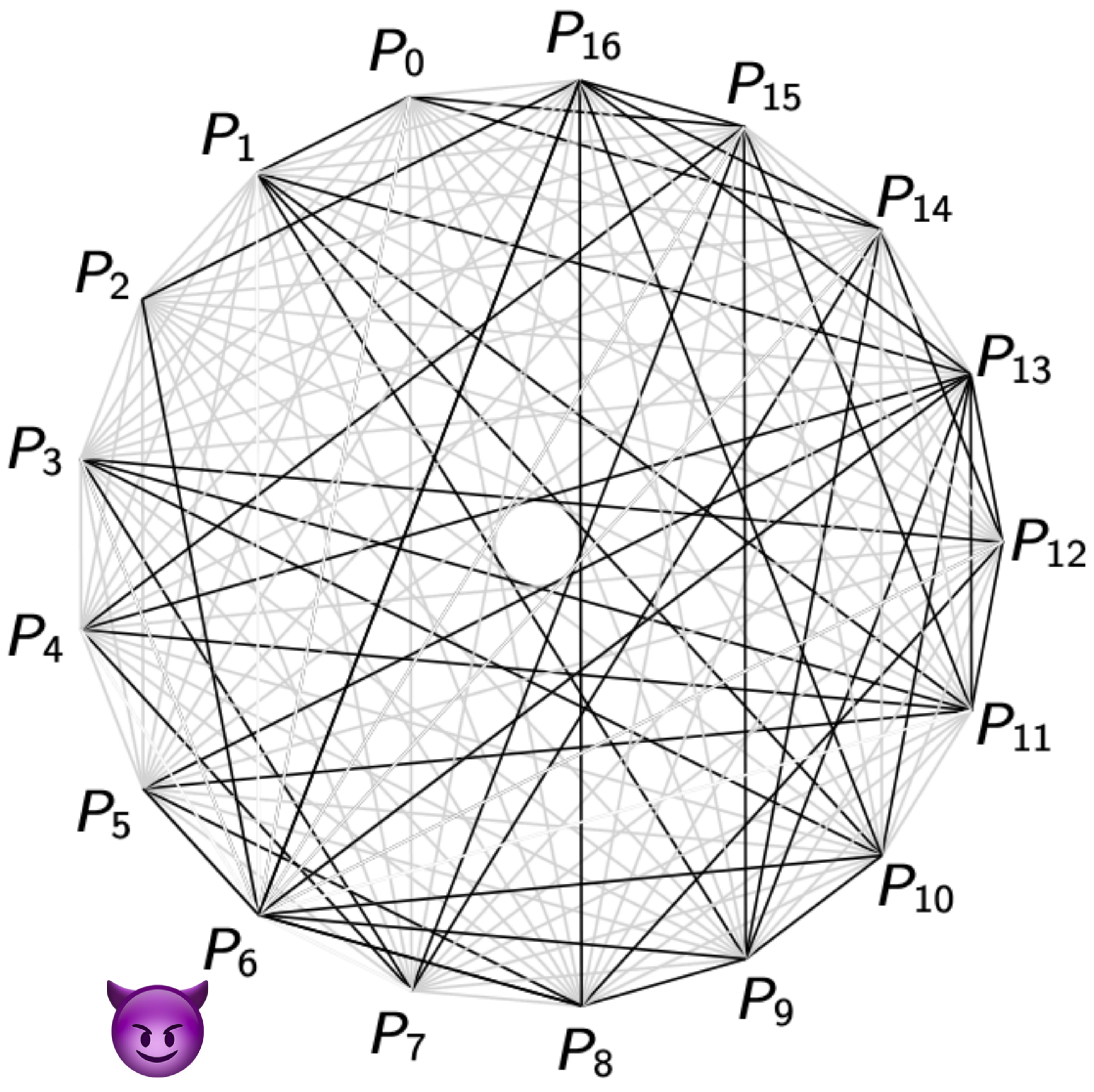
What is Topology-Hiding Computation (THC)?

Topology Privacy



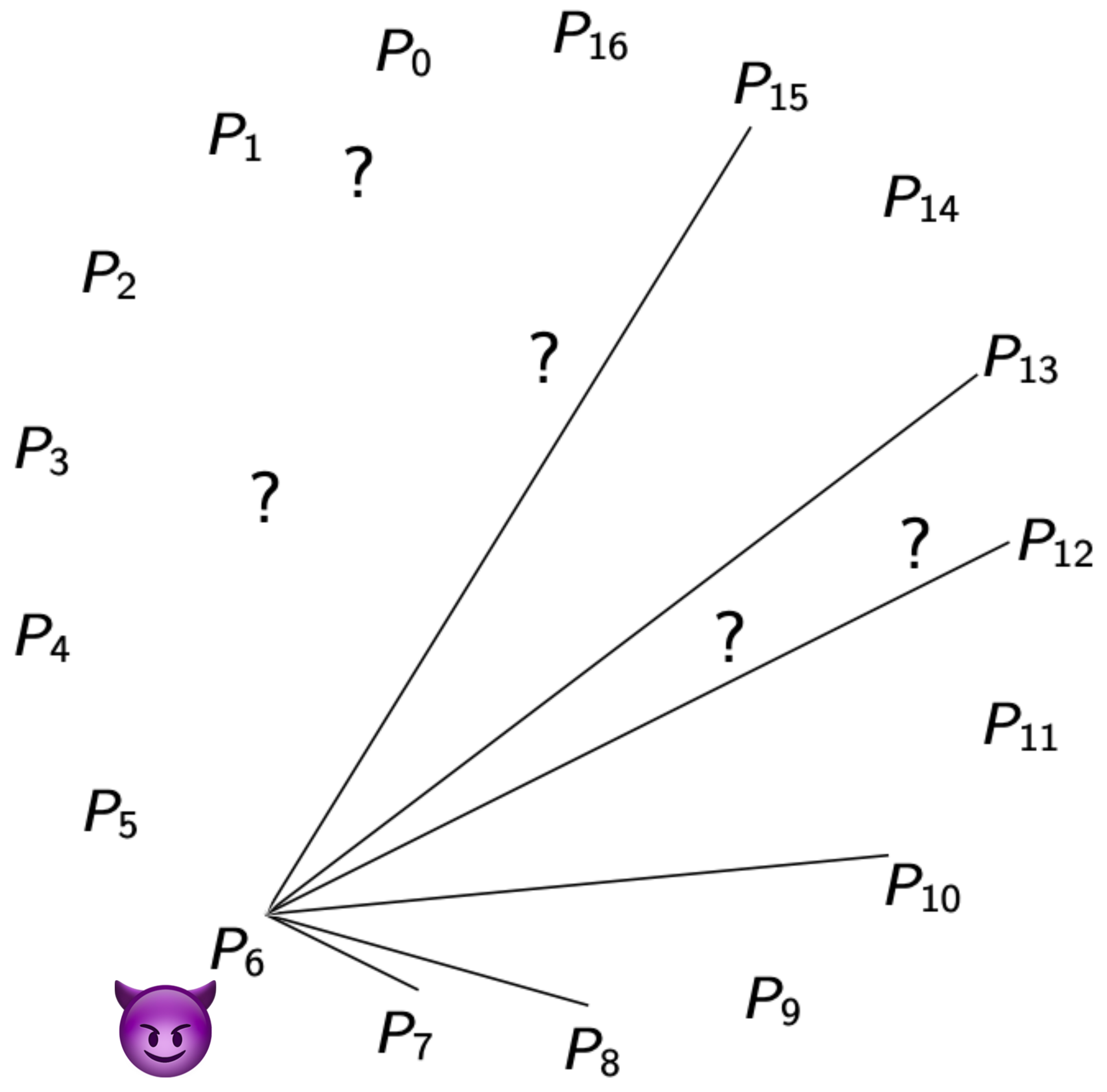
What is Topology-Hiding Computation (THC)?

Topology Privacy



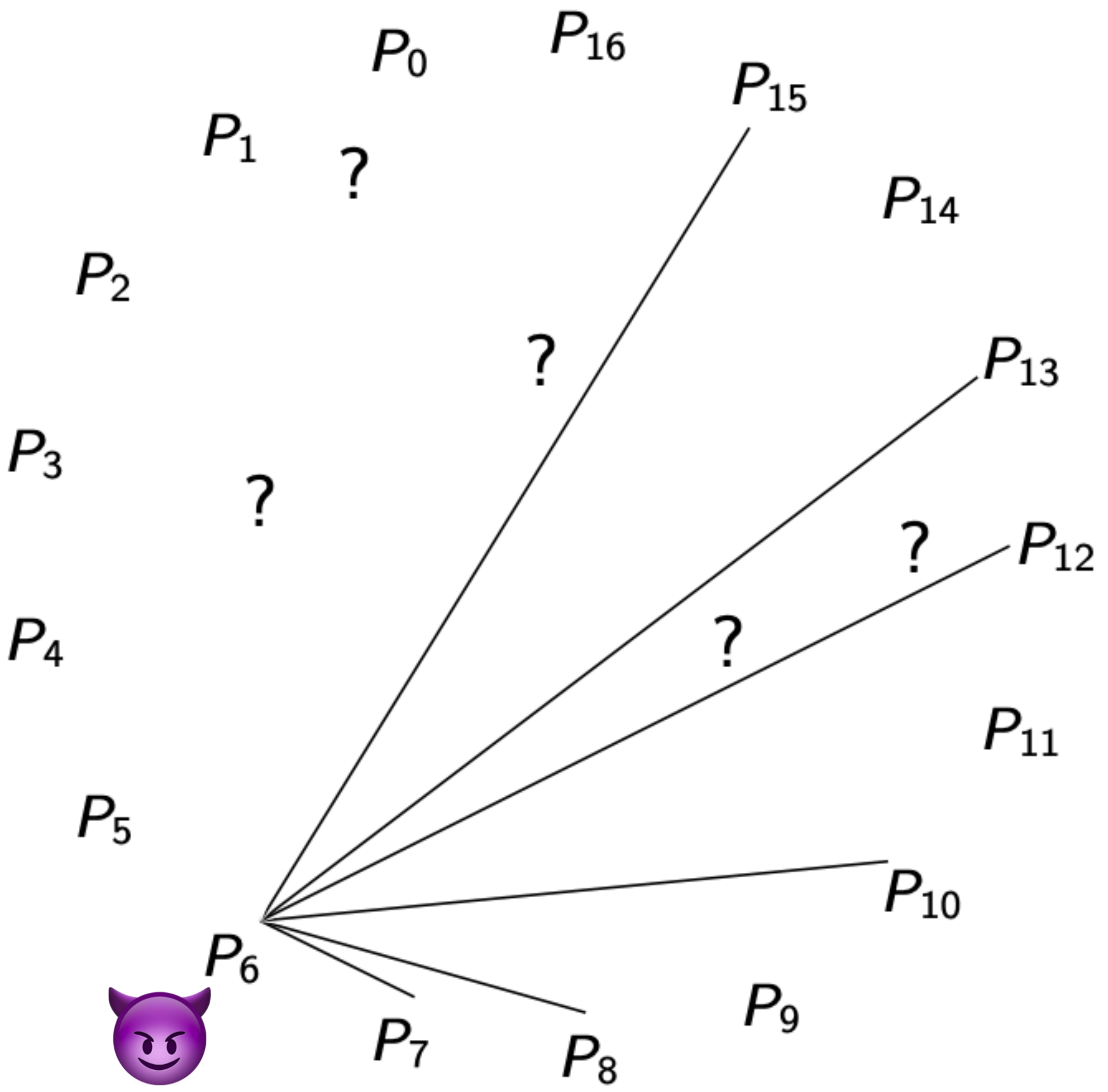
What is Topology-Hiding Computation (THC)?

Topology Privacy



What is Topology-Hiding Computation (THC)?

Topology Privacy



Goal: P_6 should learn nothing else beyond its local view (its neighbors)

What is Topology-Hiding Computation (THC)?

Definition [MOR15]

What is Topology-Hiding Computation (THC)?

Definition [MOR15]

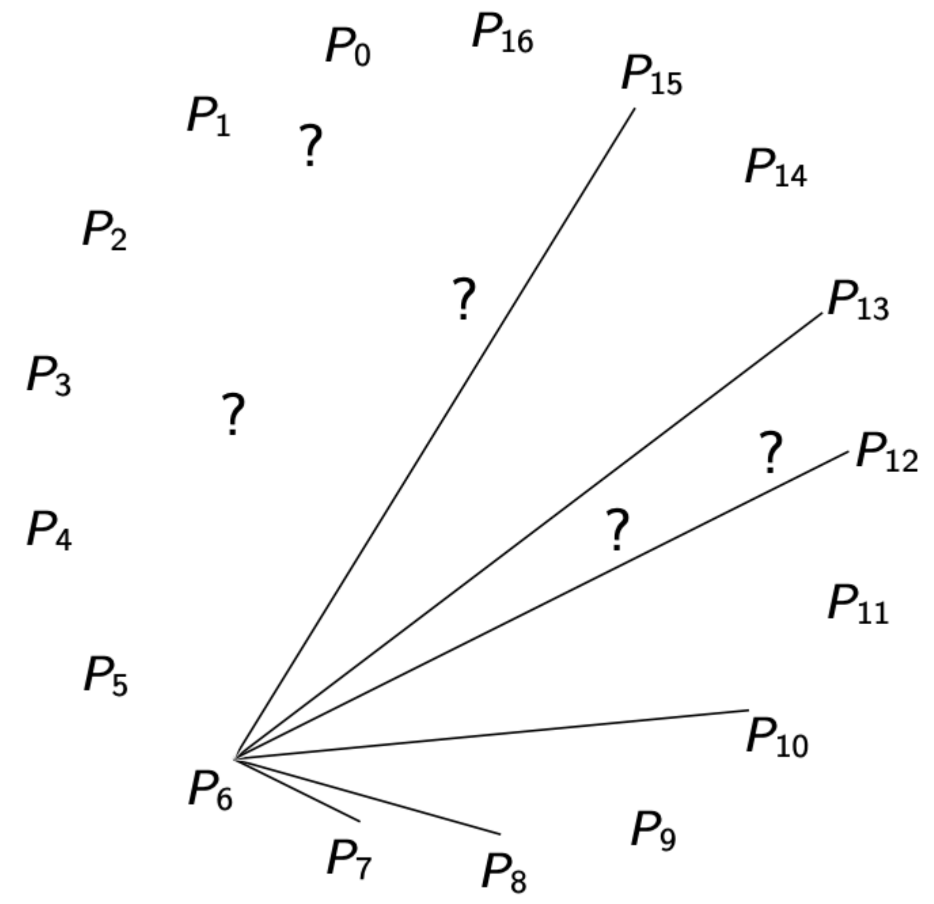
- A protocol Π is a *Topology-Hiding Computation* Protocol for a graph class $\mathscr{G}$ and functionality $\mathscr{F}$ if:

What is Topology-Hiding Computation (THC)?

Definition [MOR15]

- A protocol Π is a *Topology-Hiding Computation* Protocol for a graph class $\mathcal{G}$ and functionality $\mathcal{F}$ if:

$$\mathrm{Sim}(\mathcal{G}, x_6, \mathcal{F}(x_0, \ldots, x_{n-1}),$$



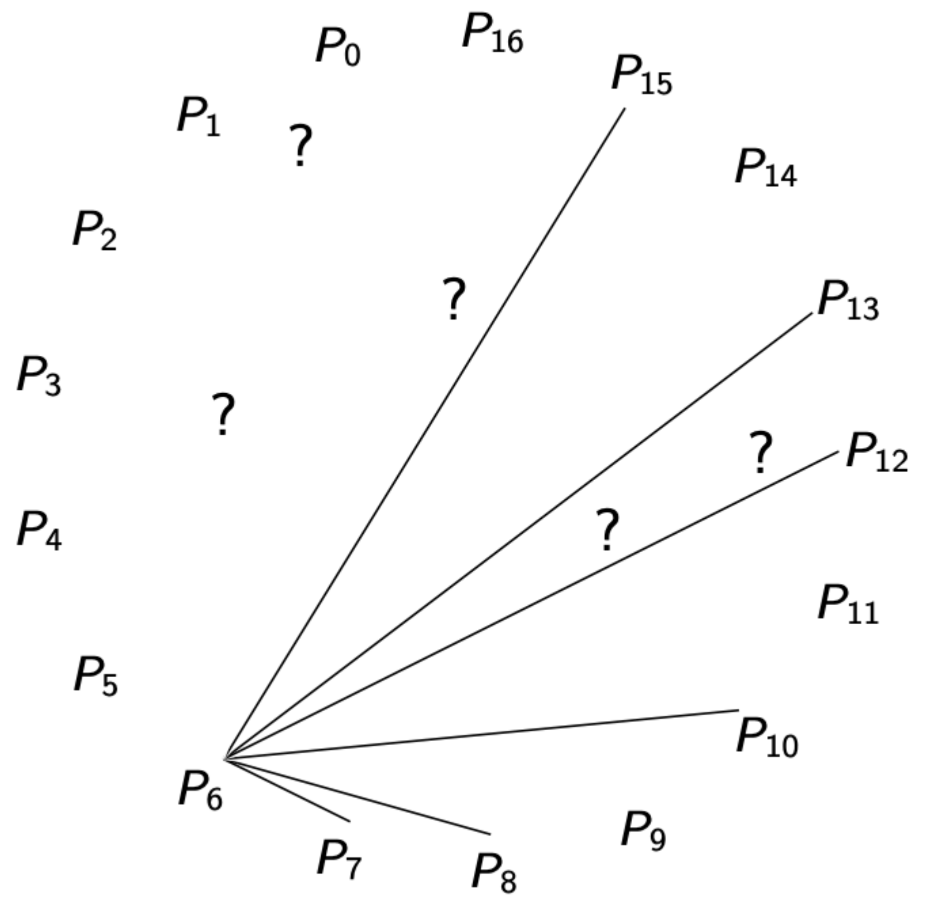
$$) \approx \mathrm{View}(\Pi)_6$$

What is Topology-Hiding Computation (THC)?

Definition [MOR15]

- A protocol Π is a *Topology-Hiding Computation* Protocol for a graph class $\mathcal{G}$ and functionality $\mathcal{F}$ if:

$$\text{Sim}(\mathcal{G}, x_6, \mathcal{F}(x_0, \ldots, x_{n-1}),$$



$$) \approx \text{View}(\Pi)_6$$

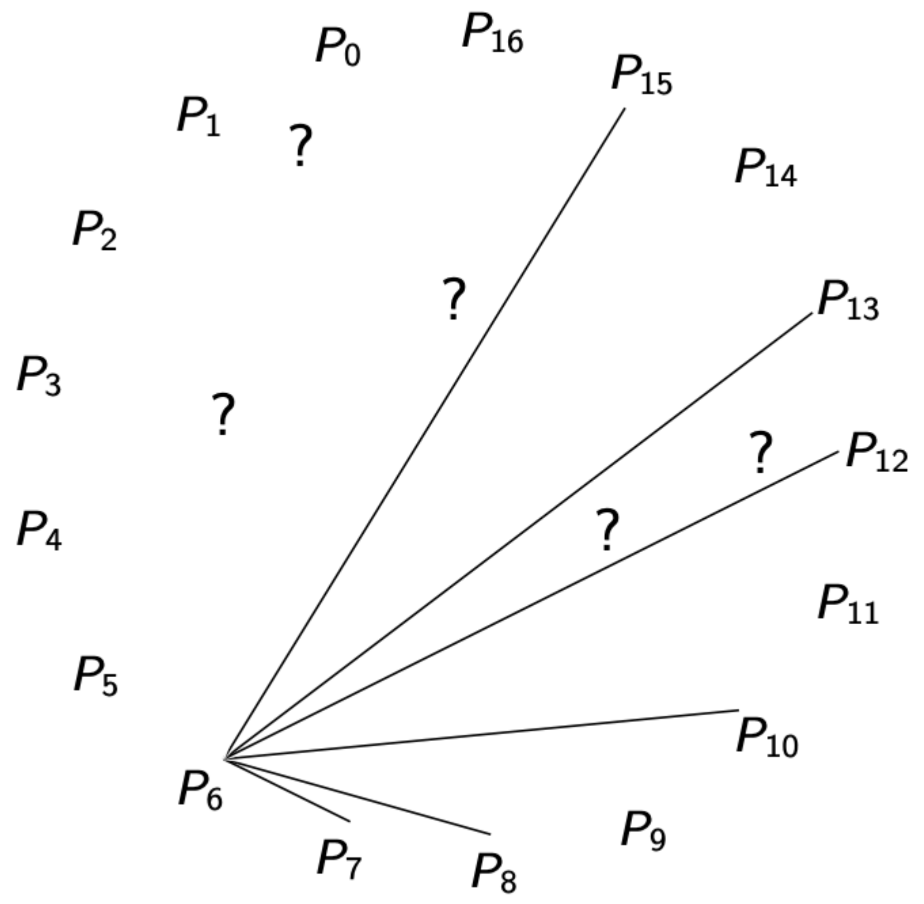
- (Protocol reveals nothing beyond input/output, and neighbors in graph)

What is Topology-Hiding Computation (THC)?

Definition [MOR15]

- A protocol Π is a *Topology-Hiding Computation* Protocol for a graph class $\mathscr{G}$ and functionality $\mathscr{F}$ if:

$$\mathsf{Sim}(\mathscr{G}, x_6, \mathscr{F}(x_0, \ldots, x_{n-1}),$$

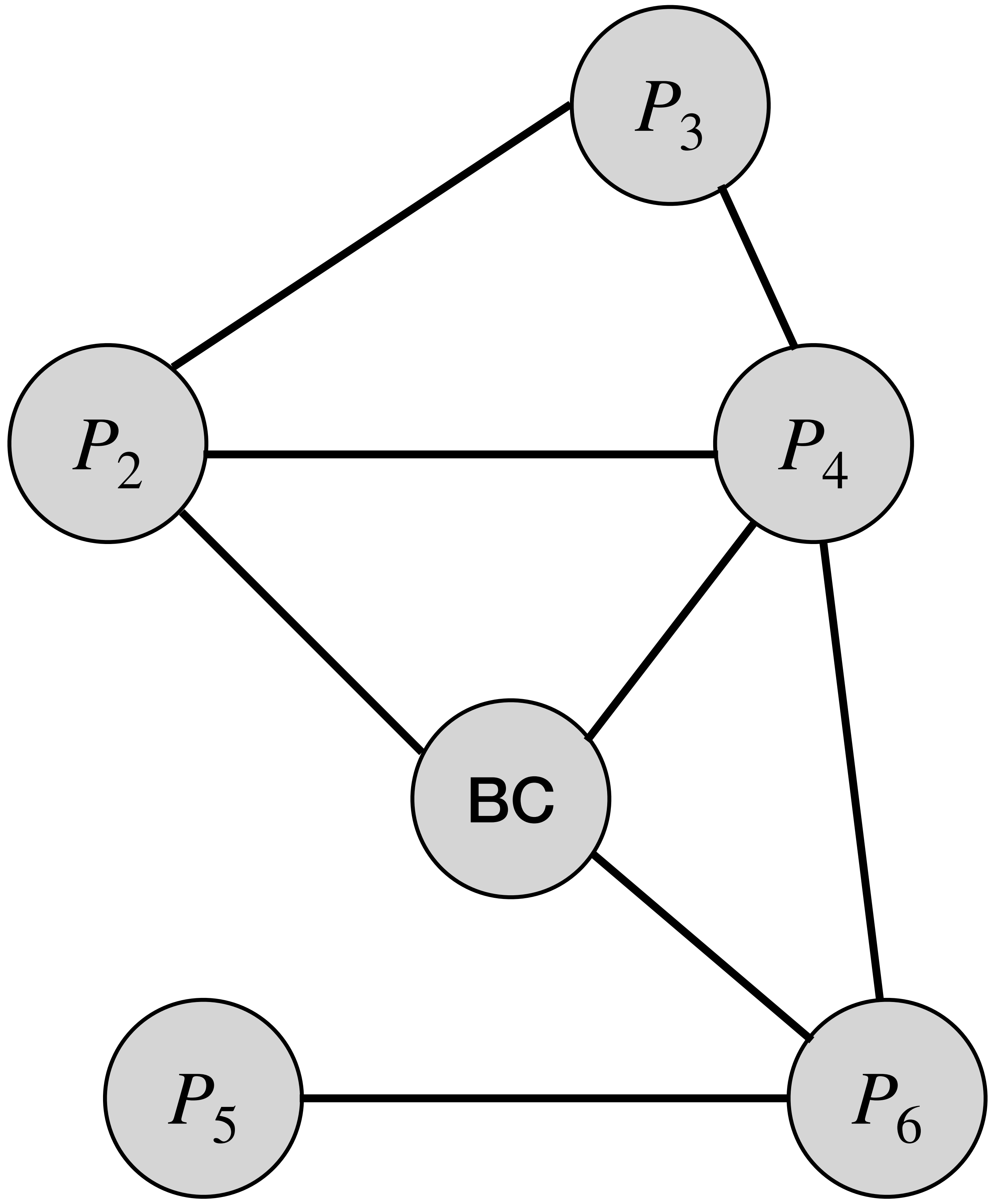


$$) \approx \mathsf{View}(\Pi)_6$$

- (Protocol reveals nothing beyond input/output, and neighbors in graph)
- Can consider different number of corruptions, passive/active, static/adaptive

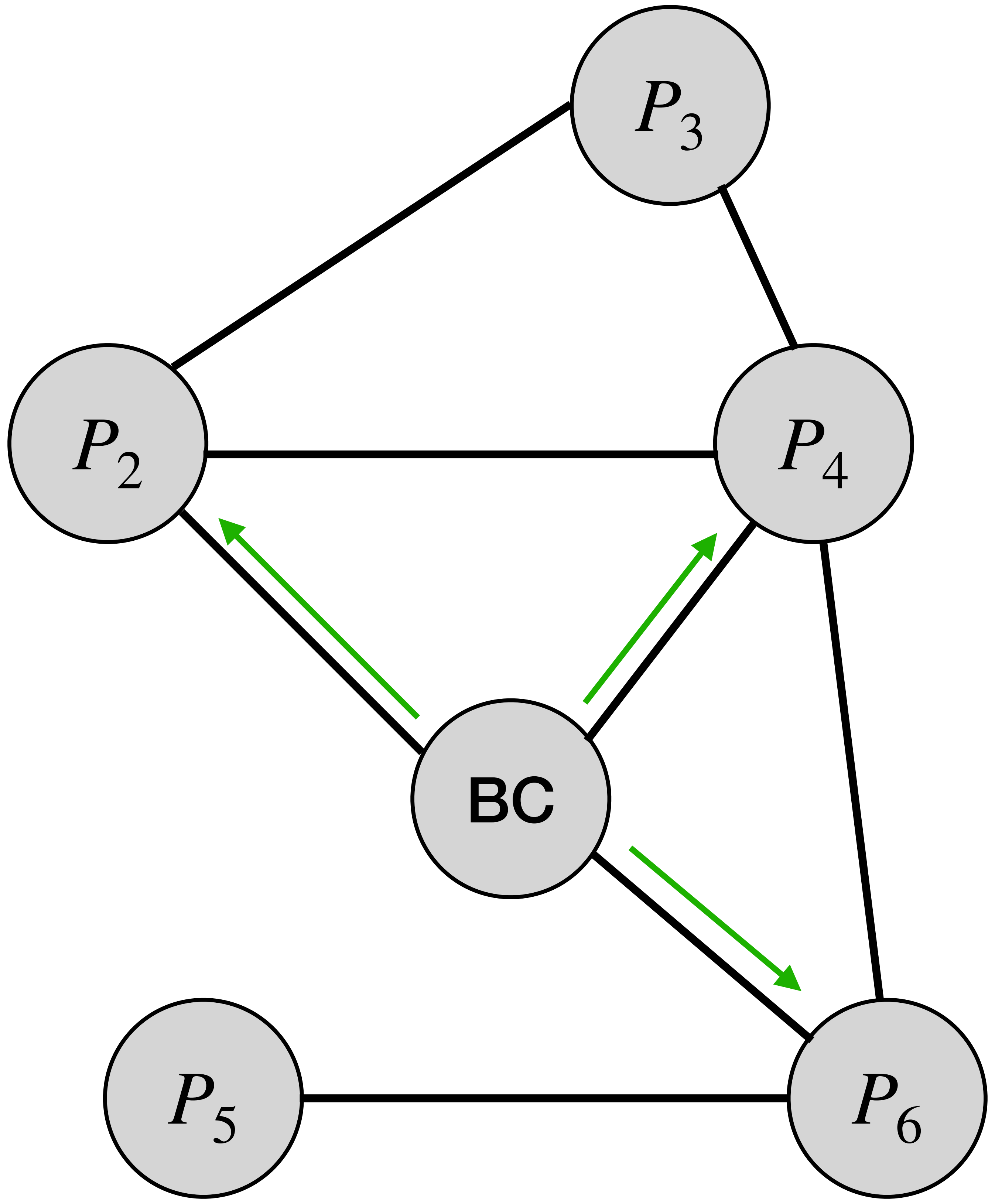
Why is THC hard (even with passive adversaries)?

Flooding Broadcast



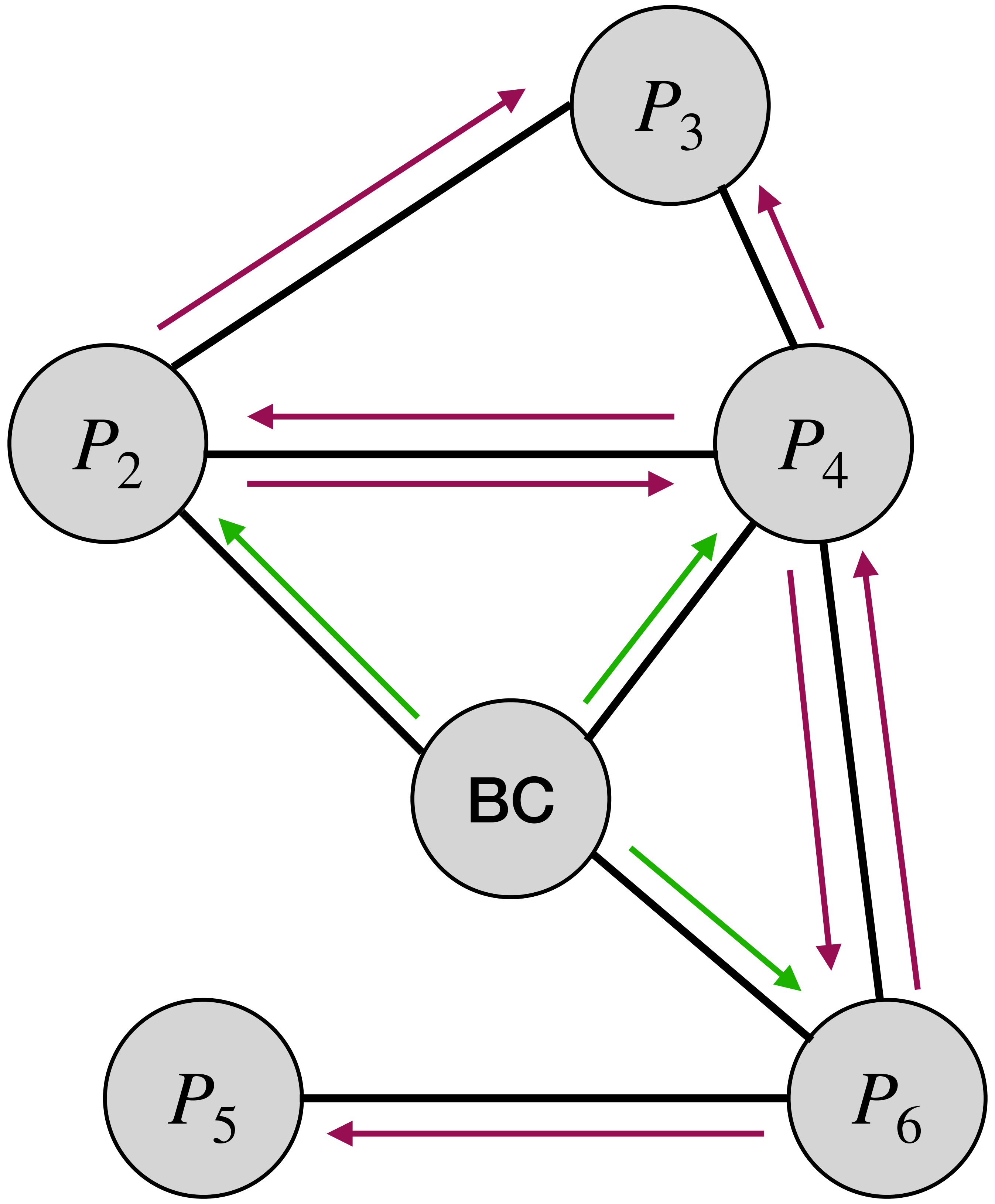
Why is THC hard (even with passive adversaries)?

Flooding Broadcast



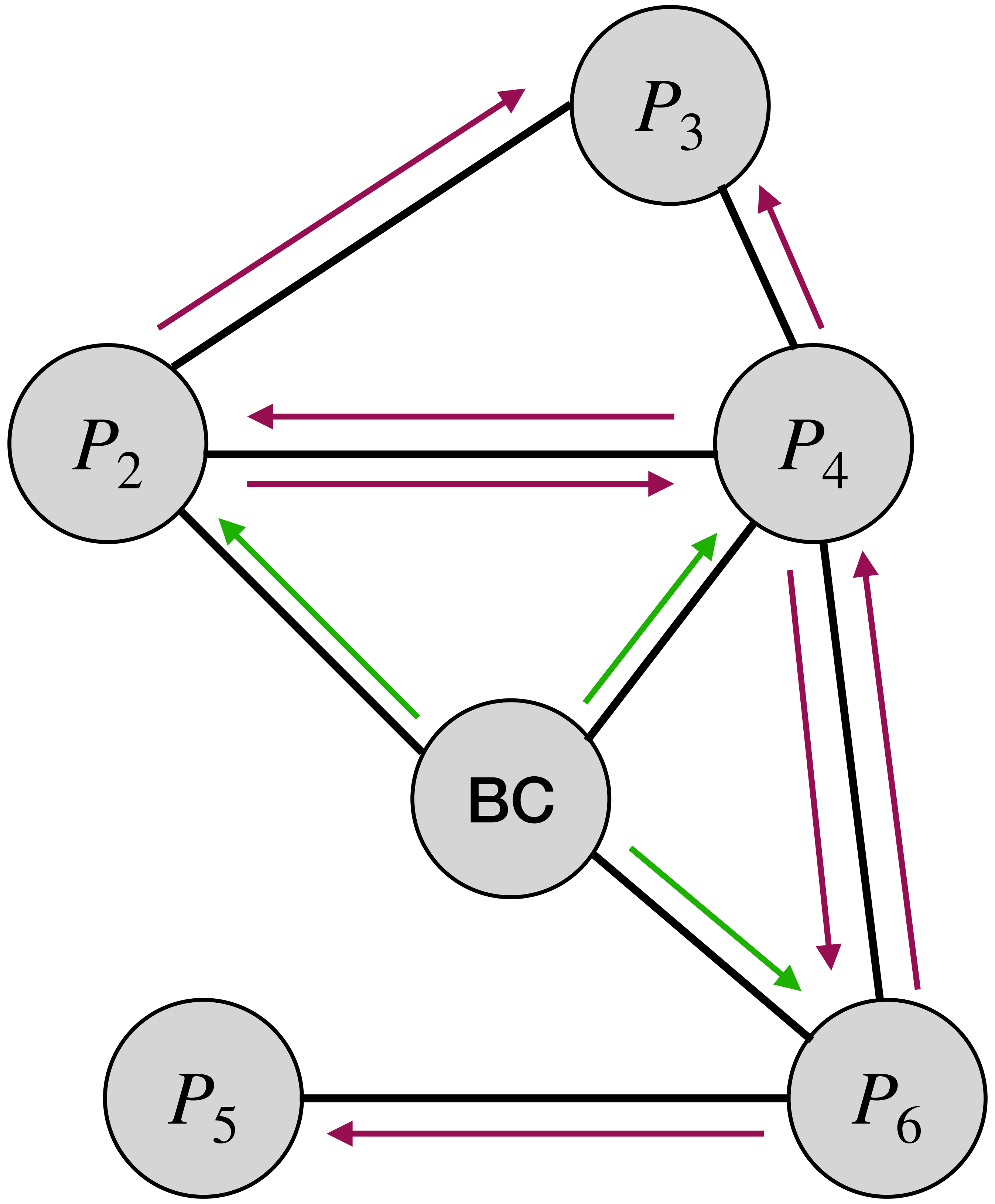
Why is THC hard (even with passive adversaries)?

Flooding Broadcast



Why is THC hard (even with passive adversaries)?

Flooding Broadcast



Each party learns:

- Distance to BC
- Nbrs' distance to BC

Not Topology-Hiding!

Our Question

THC Oblivious Transfer MPC

Our Question

THC

Oblivious Transfer ⇄ MPC

Our Question

THC → Oblivious Transfer ⇄ MPC

Our Question

Known Results

Semi-Honest, Static, Arbitrary Number of Corruptions

Known Results

Semi-Honest, Static, Arbitrary Number of Corruptions

- [MOR15]: Constant Round MPC with Constant Overhead => THC for graphs with constant degree and logarithmic diameter

Known Results

Semi-Honest, Static, Arbitrary Number of Corruptions

- [MOR15]: Constant Round MPC with Constant Overhead => THC for graphs with constant degree and logarithmic diameter
- [AM17,ALM17,LZM+18]: Key-homomorphic, re-randomizable encryption => THC for all graphs
  - Only known from structured algebraic assumptions; e.g., QR, DDH, LWE

Our Contributions

Our Contributions

- Assuming constant round 2PC with constant overhead, THC for semi-honest, static adversaries corrupting all-but-one parties exists for all graphs

Our Contributions

- Assuming constant round 2PC with constant overhead, THC for semi-honest, static adversaries corrupting all-but-one parties exists for all graphs

    1. Define Locally-Simulatable MPC (for any fixed Graph)

Our Contributions

- Assuming constant round 2PC with constant overhead, THC for semi-honest, static adversaries corrupting all-but-one parties exists for all graphs

  1. Define Locally-Simulatable MPC (for any fixed Graph)

  2. Use Locally-Simulatability + Correlated Random Walks [ALM17] to reduce THC to Locally-Simulatable OR on a Path

Our Contributions

- Assuming constant round 2PC with constant overhead, THC for semi-honest, static adversaries corrupting all-but-one parties exists for all graphs

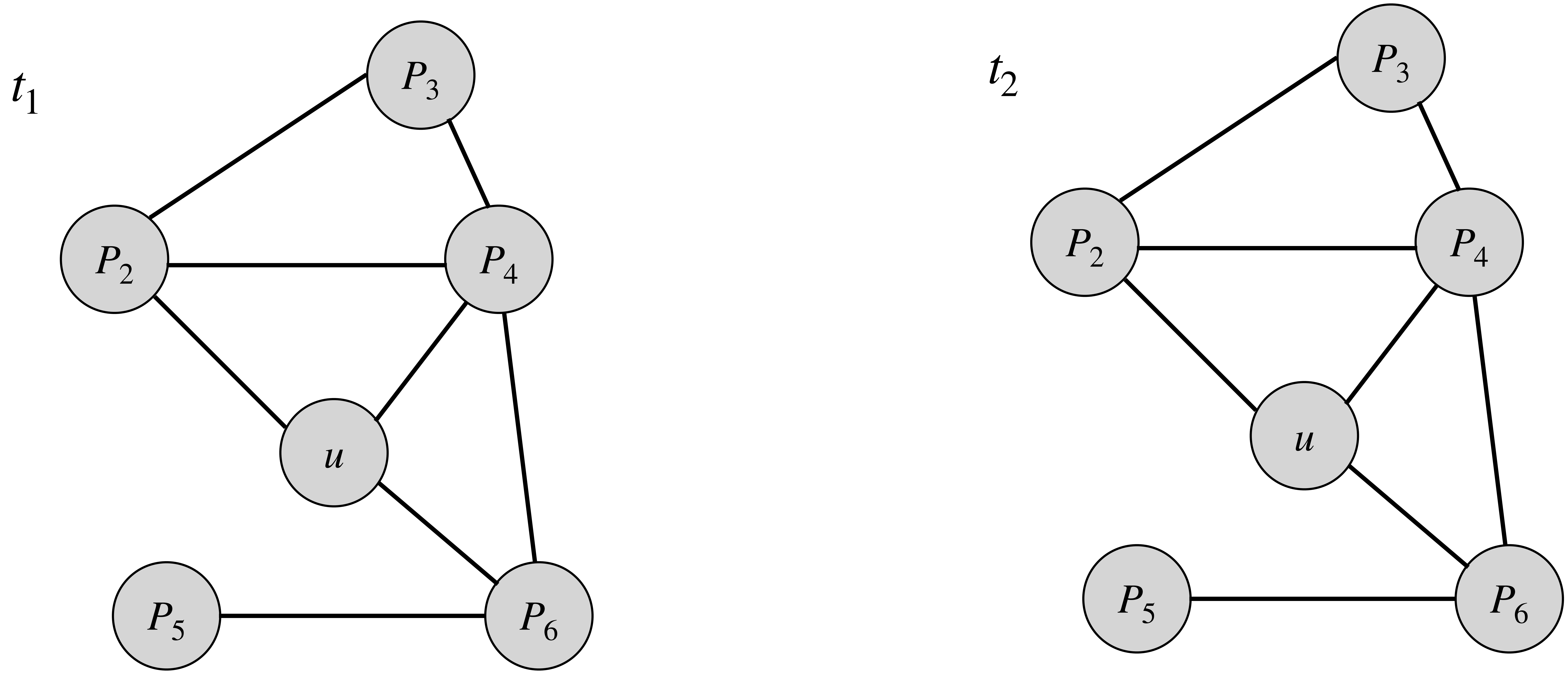
  1. Define Locally-Simulatable MPC (for any fixed Graph)

  2. Use Locally-Simulatability + Correlated Random Walks [ALM17] to reduce THC to Locally-Simulatable OR on a Path

  3. Construct Locally-Simulatable OR on a Path from constant round 2PC with constant overhead

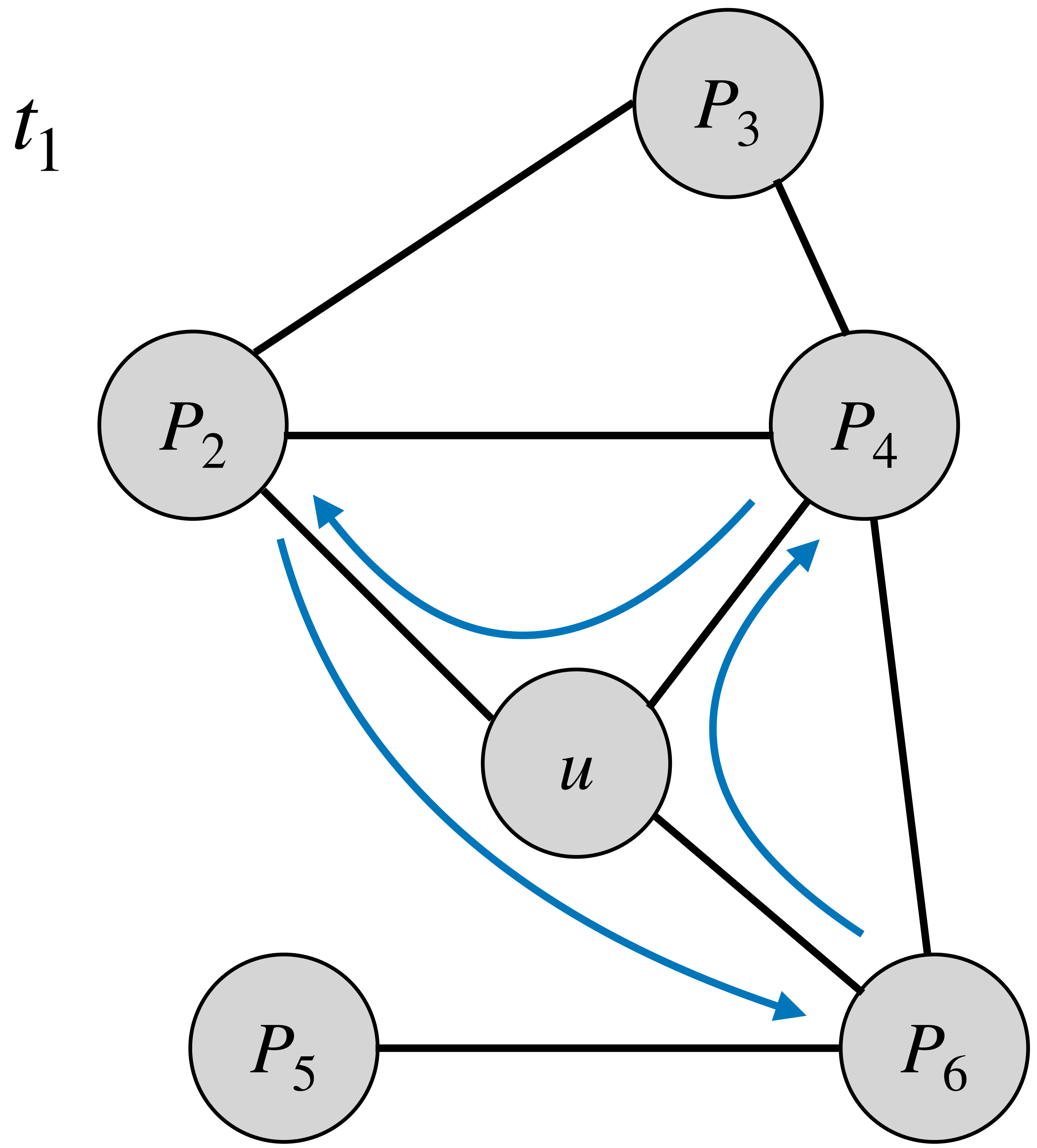
Correlated Random Walks [ALM17]

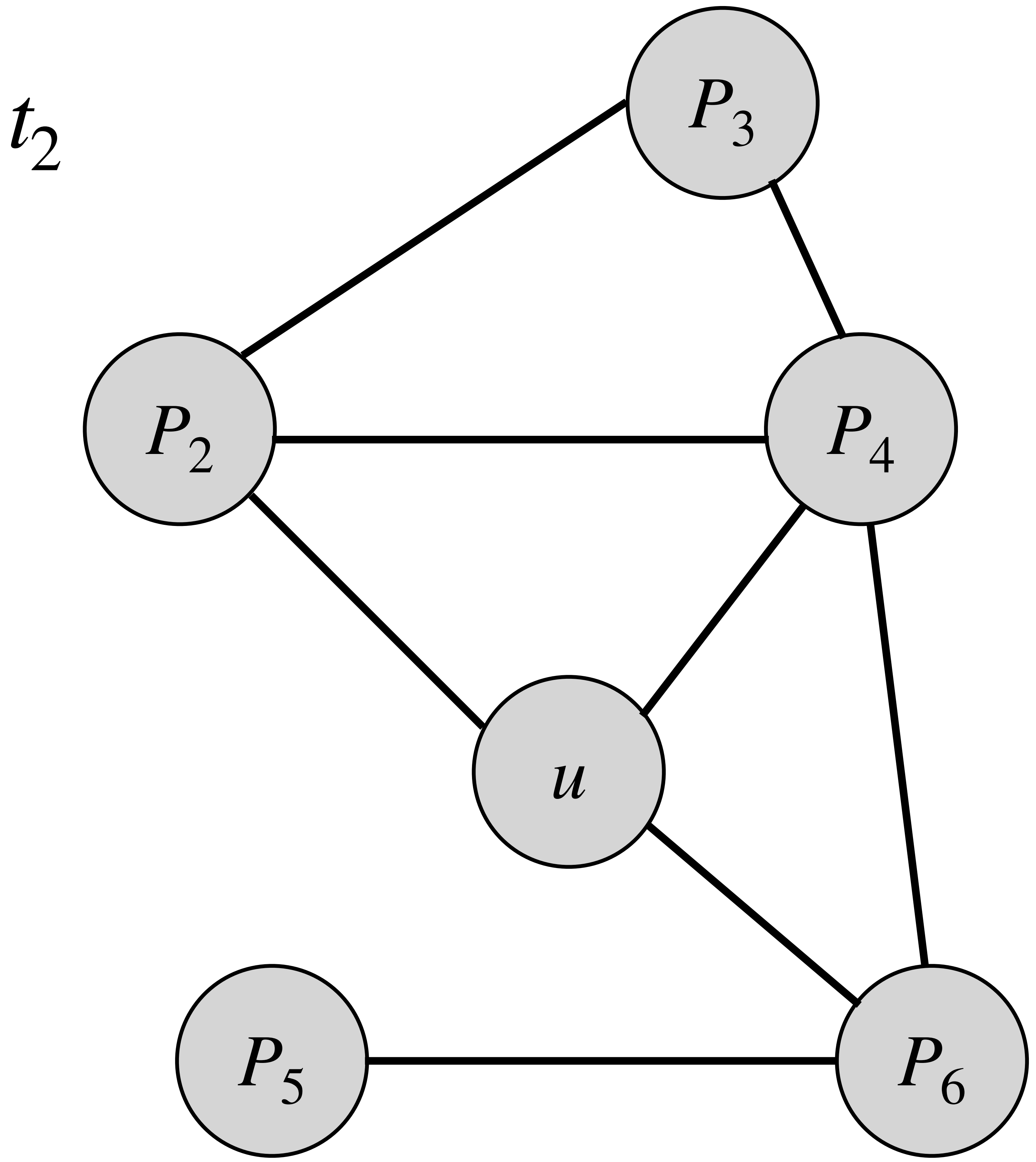
For each time step, each node u defines a permutation on its neighbors



Correlated Random Walks [ALM17]

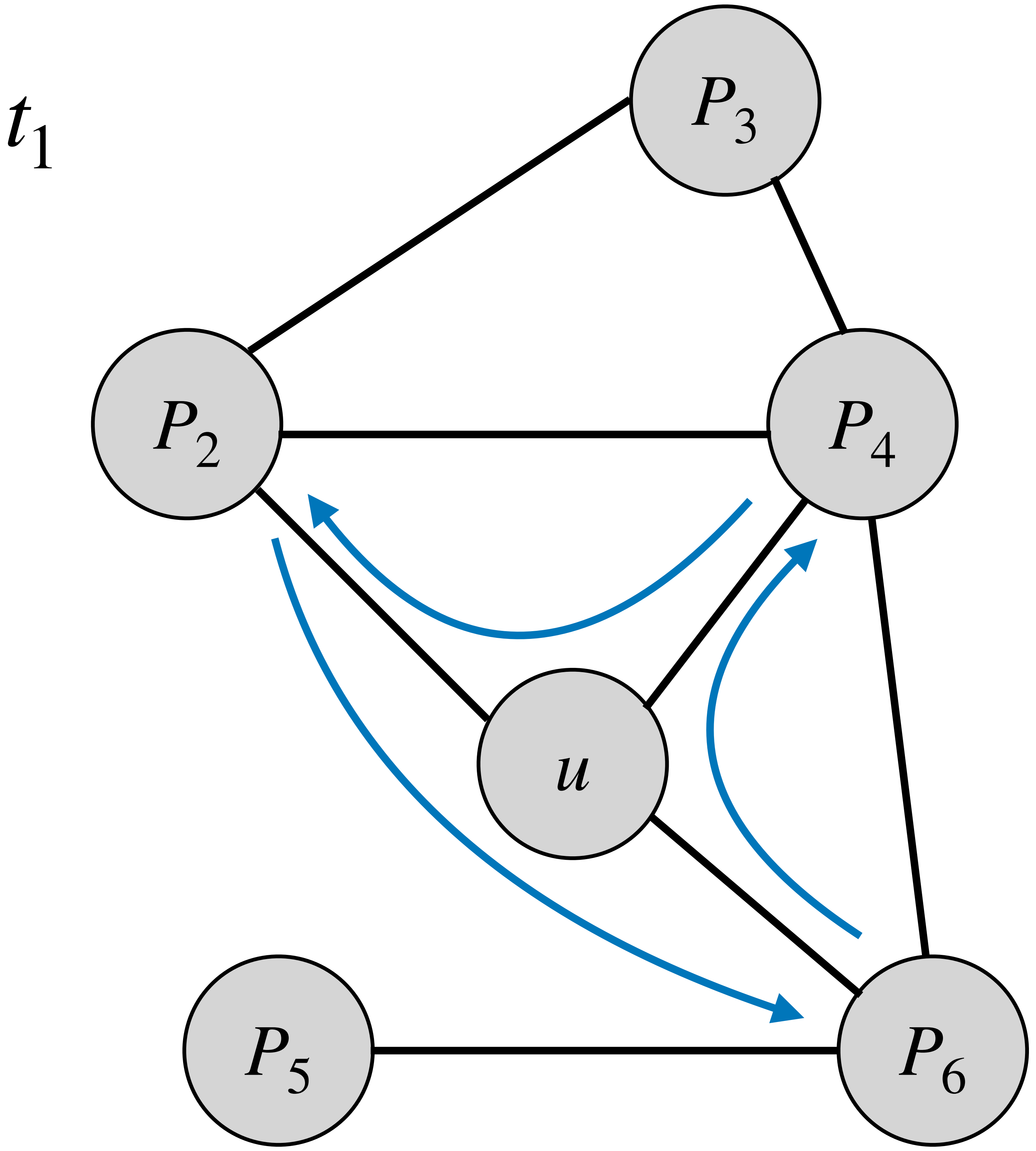
For each time step, each node u defines a permutation on its neighbors

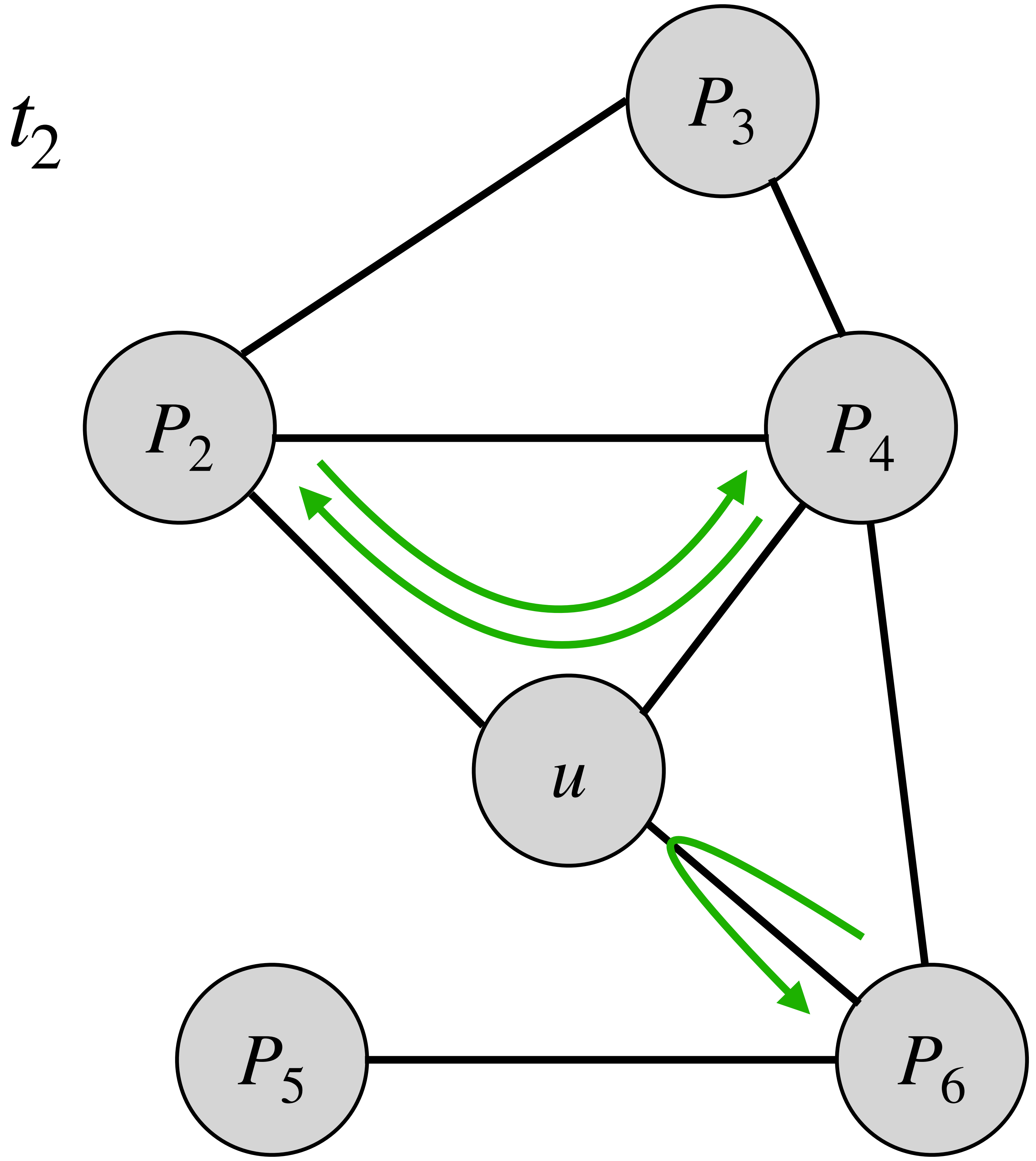




Correlated Random Walks [ALM17]

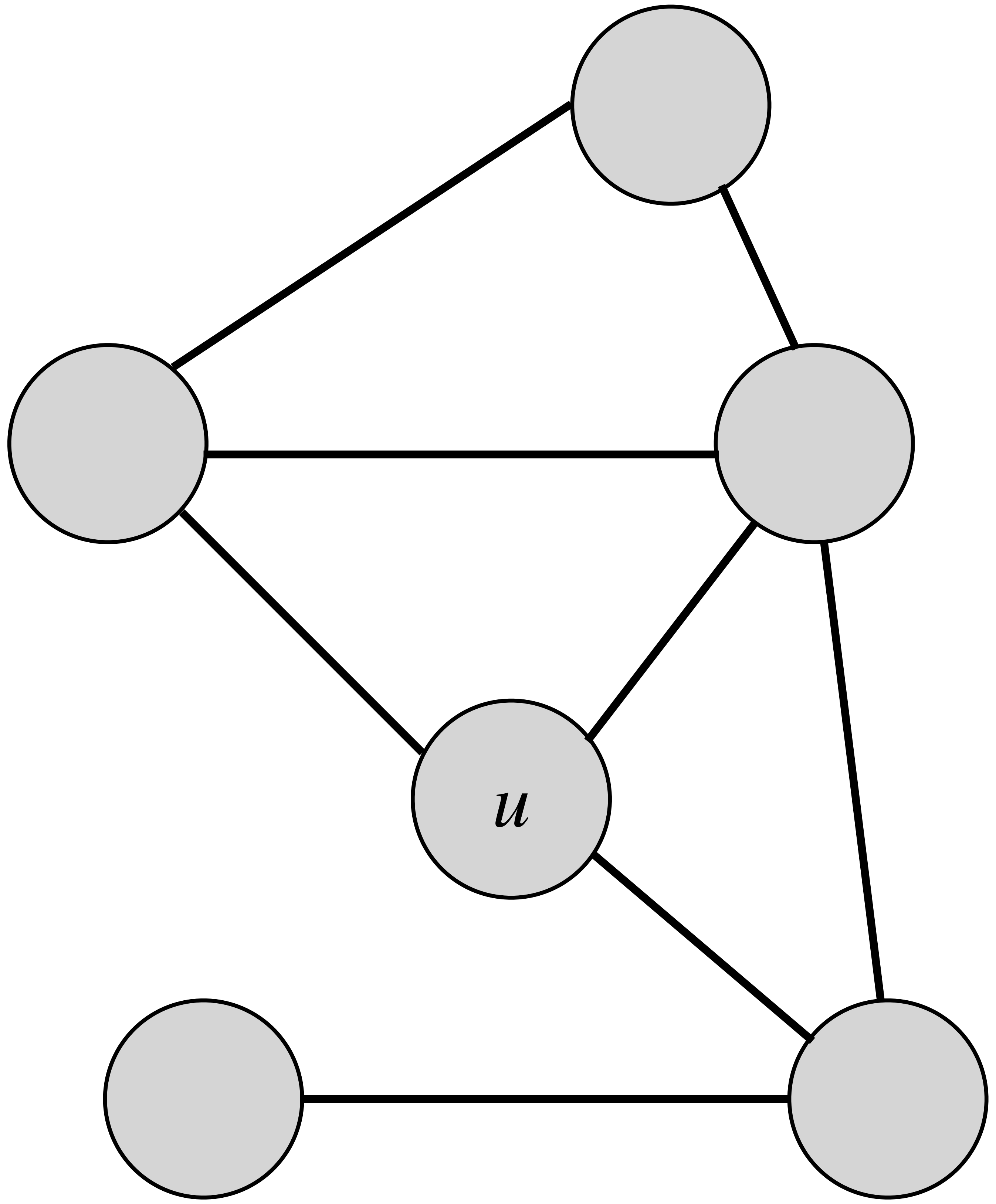
For each time step, each node u defines a permutation on its neighbors





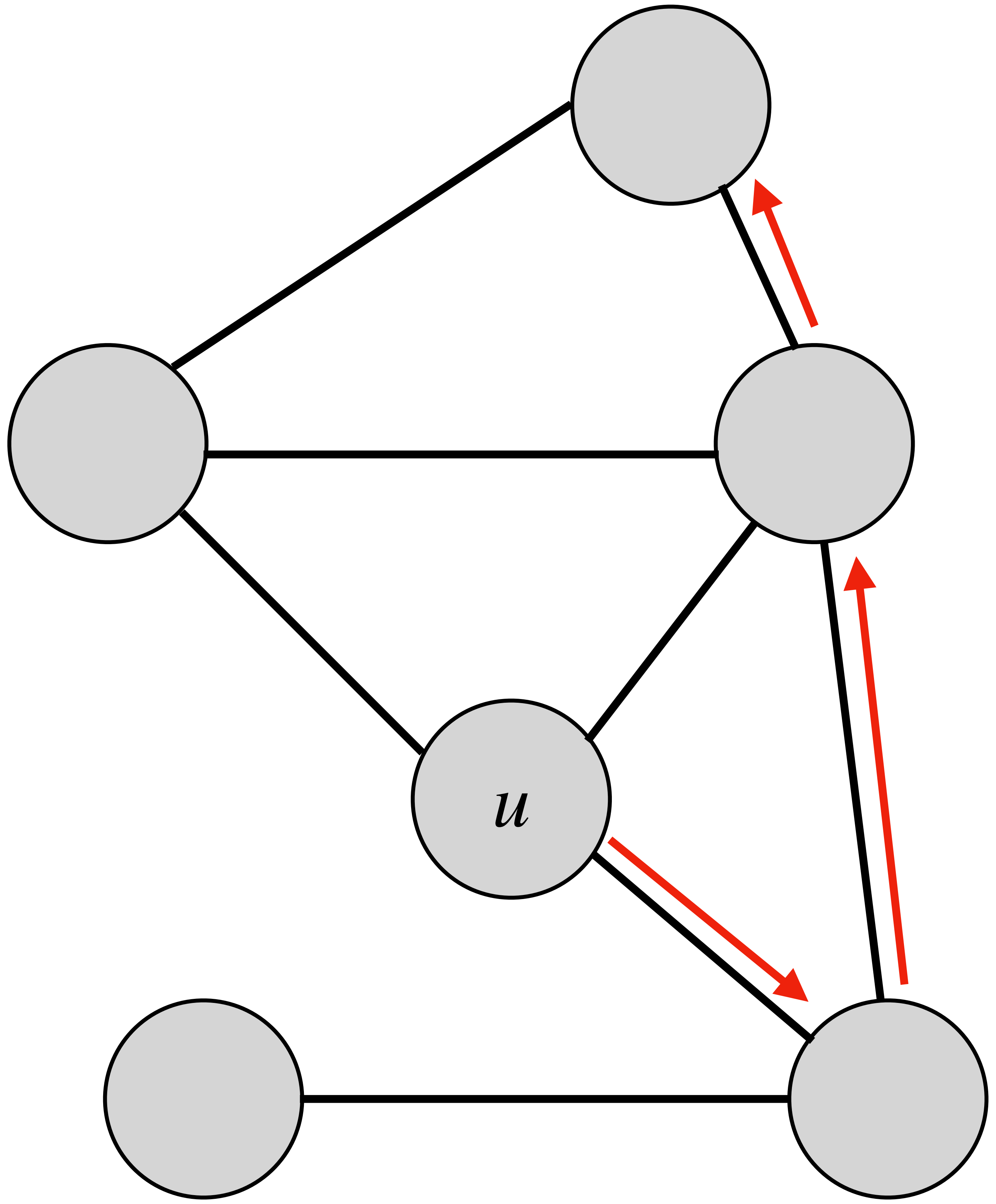
Correlated Random Walks [ALM17]

This establishes several (correlated) random walks



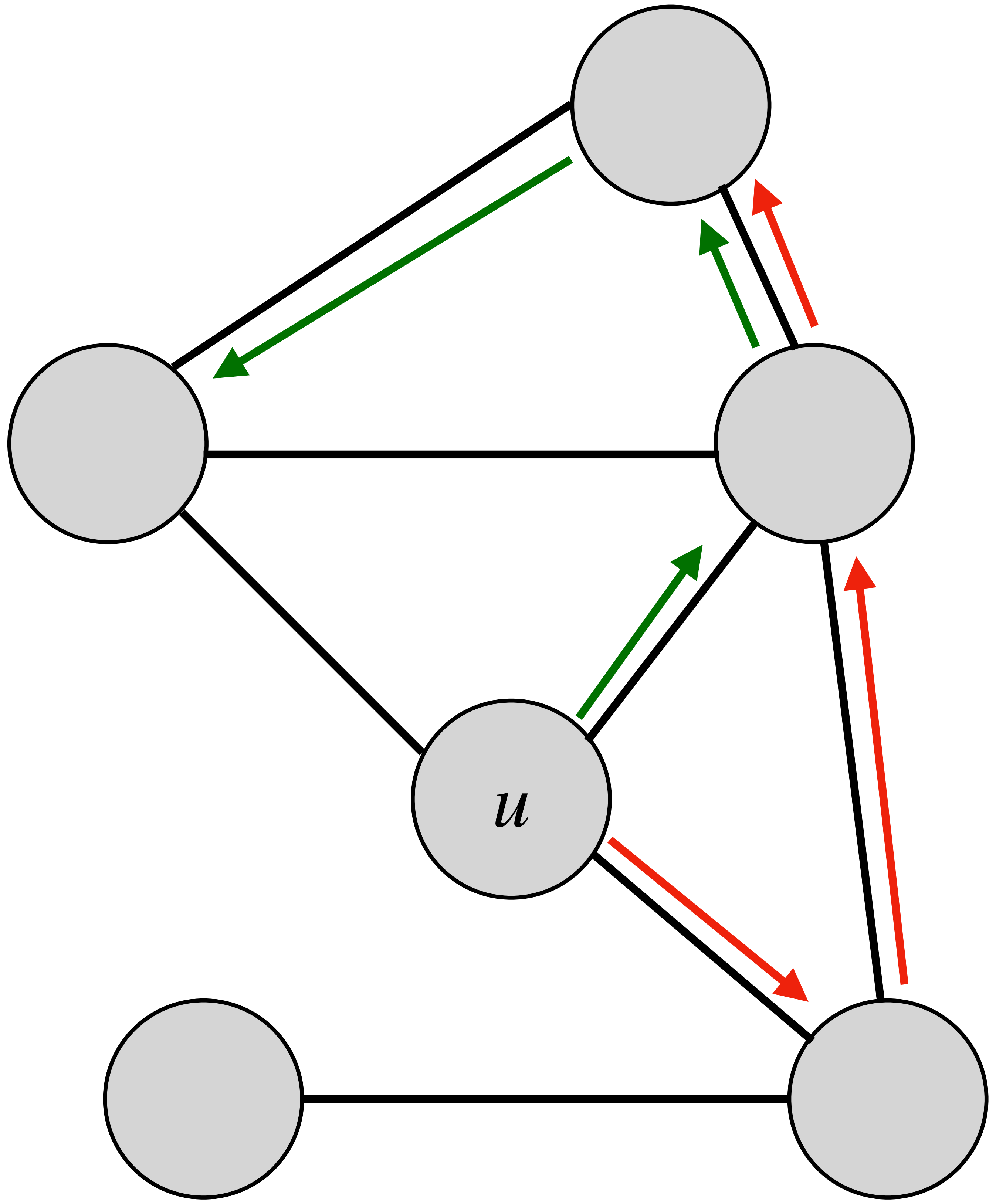
Correlated Random Walks [ALM17]

This establishes several (correlated) random walks



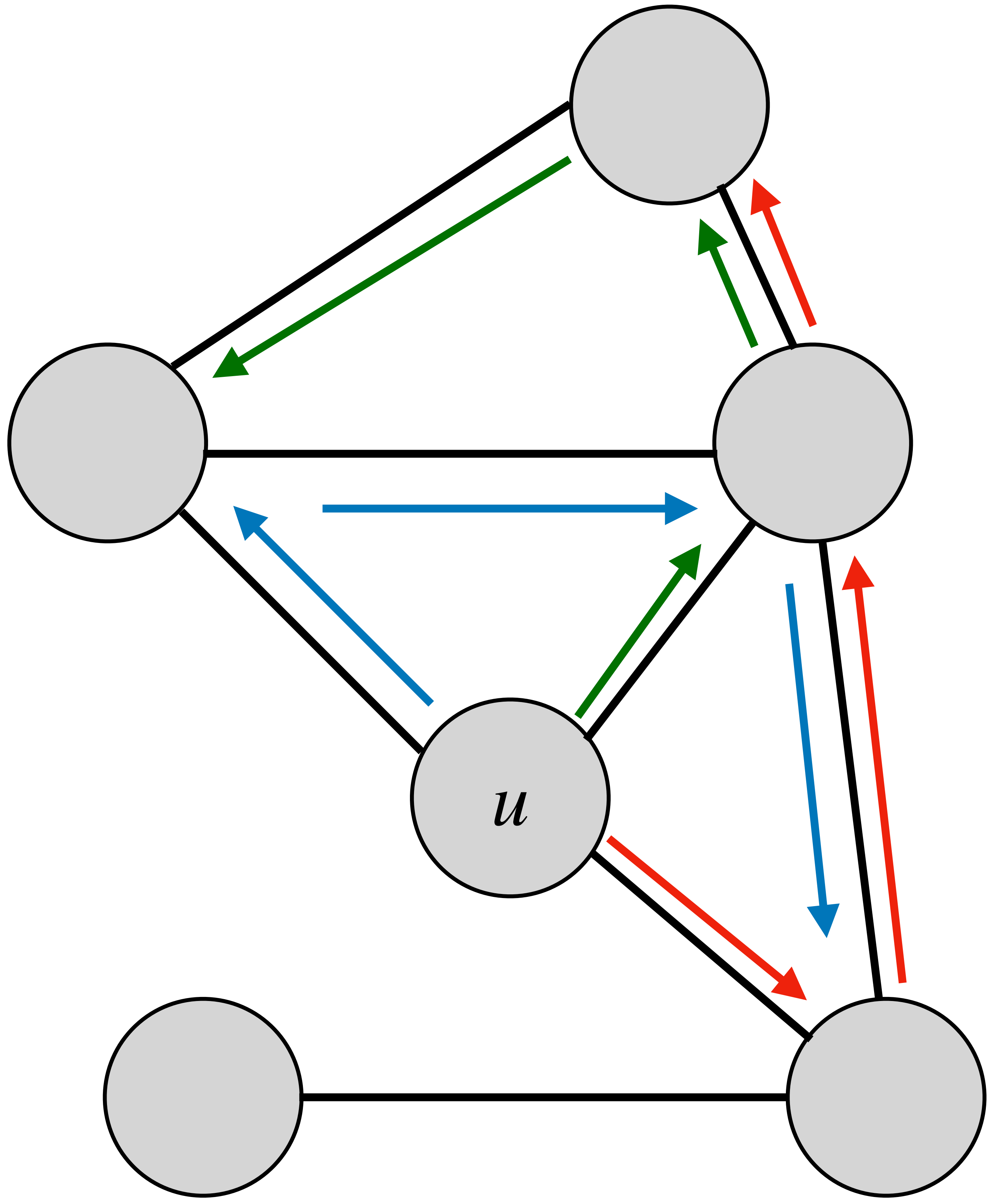
Correlated Random Walks [ALM17]

This establishes several (correlated) random walks



Correlated Random Walks [ALM17]

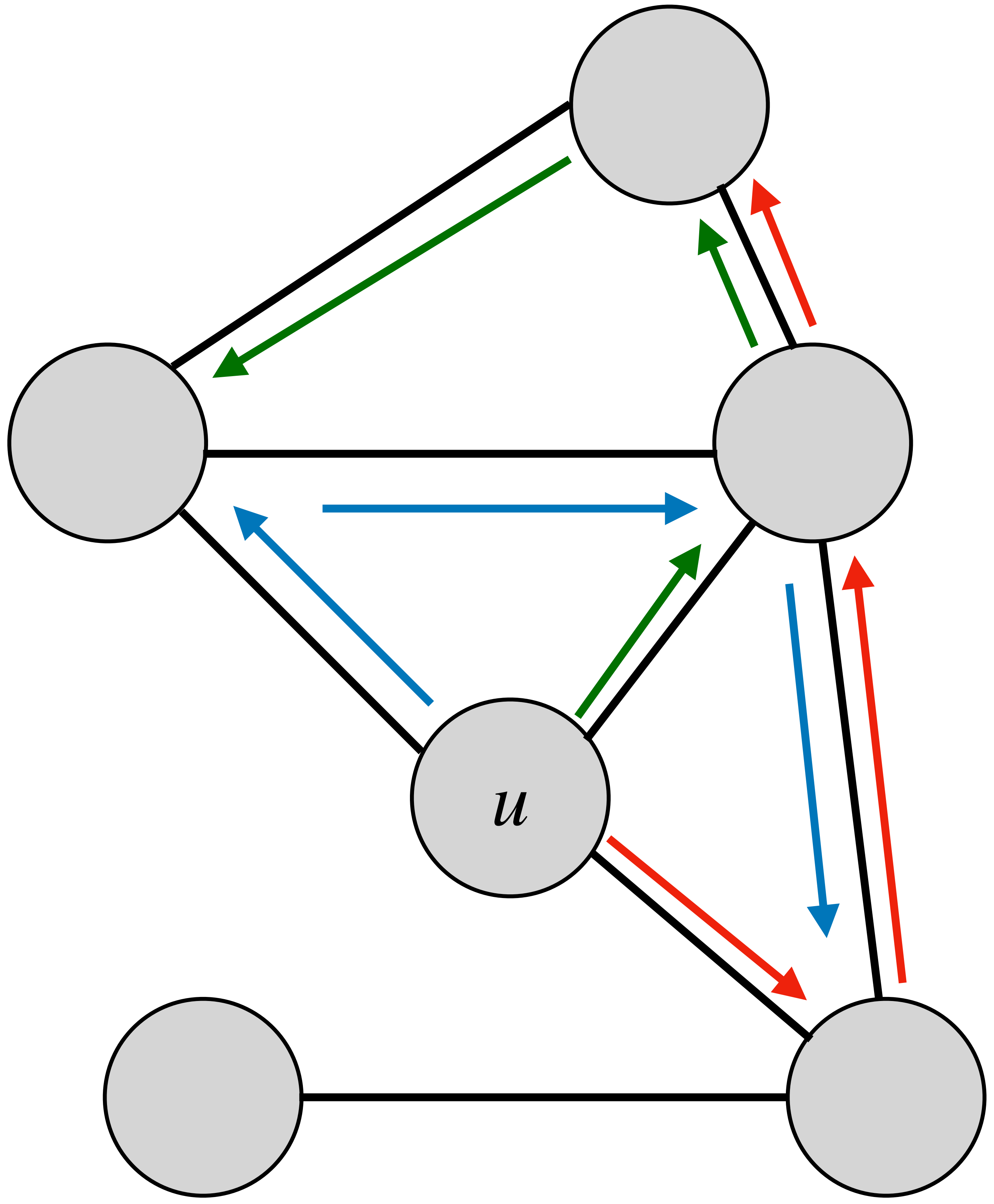
This establishes several (correlated) random walks



Correlated Random Walks [ALM17]

This establishes several (correlated) random walks

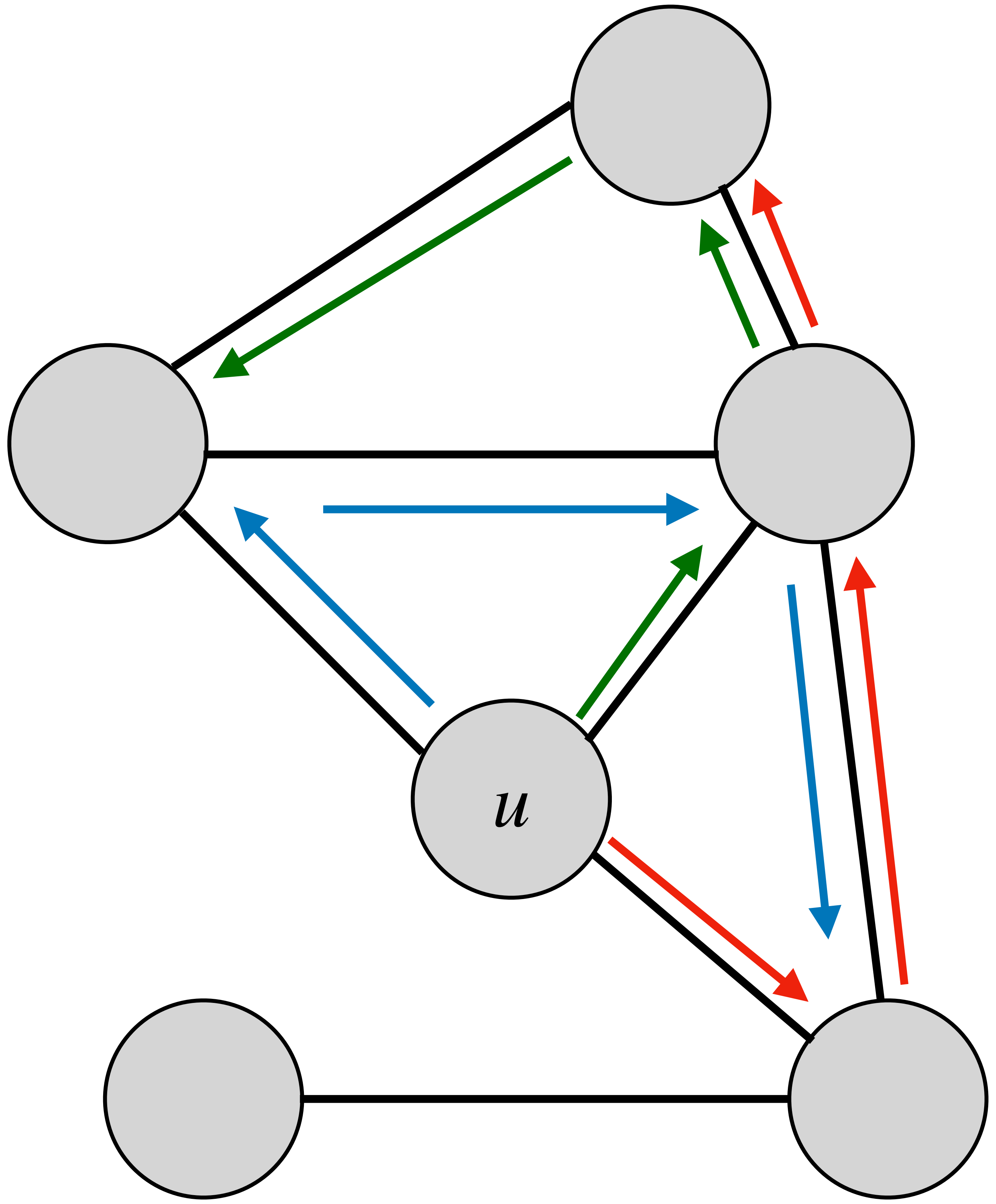
(Walks start from every node, not just u)



Correlated Random Walks [ALM17]

This establishes several (correlated) random walks

(Walks start from every node, not just u)

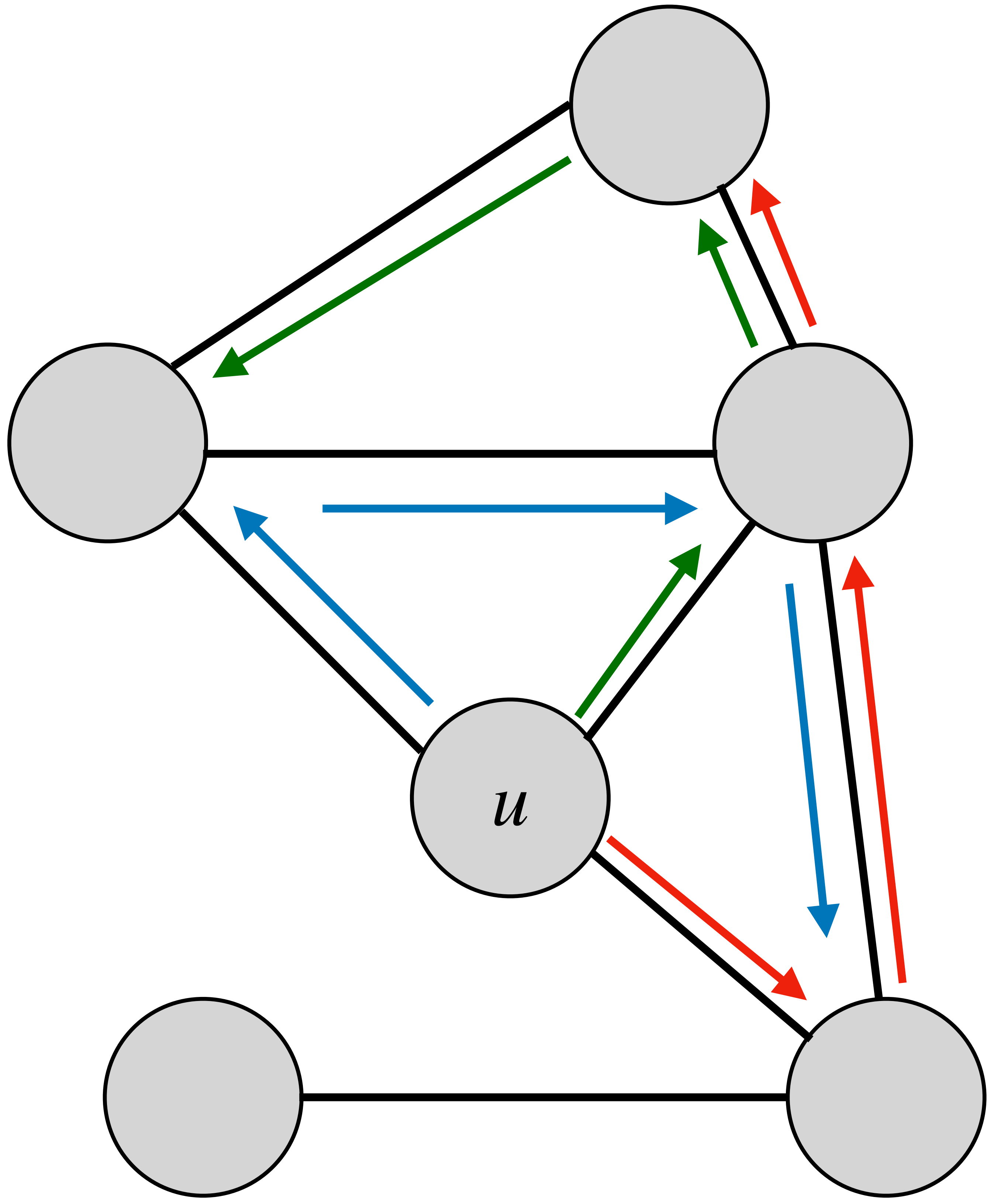


- "Correlated" -- exactly one walk through each edge at each time step

Correlated Random Walks [ALM17]

This establishes several (correlated) random walks

(Walks start from every node, not just u)

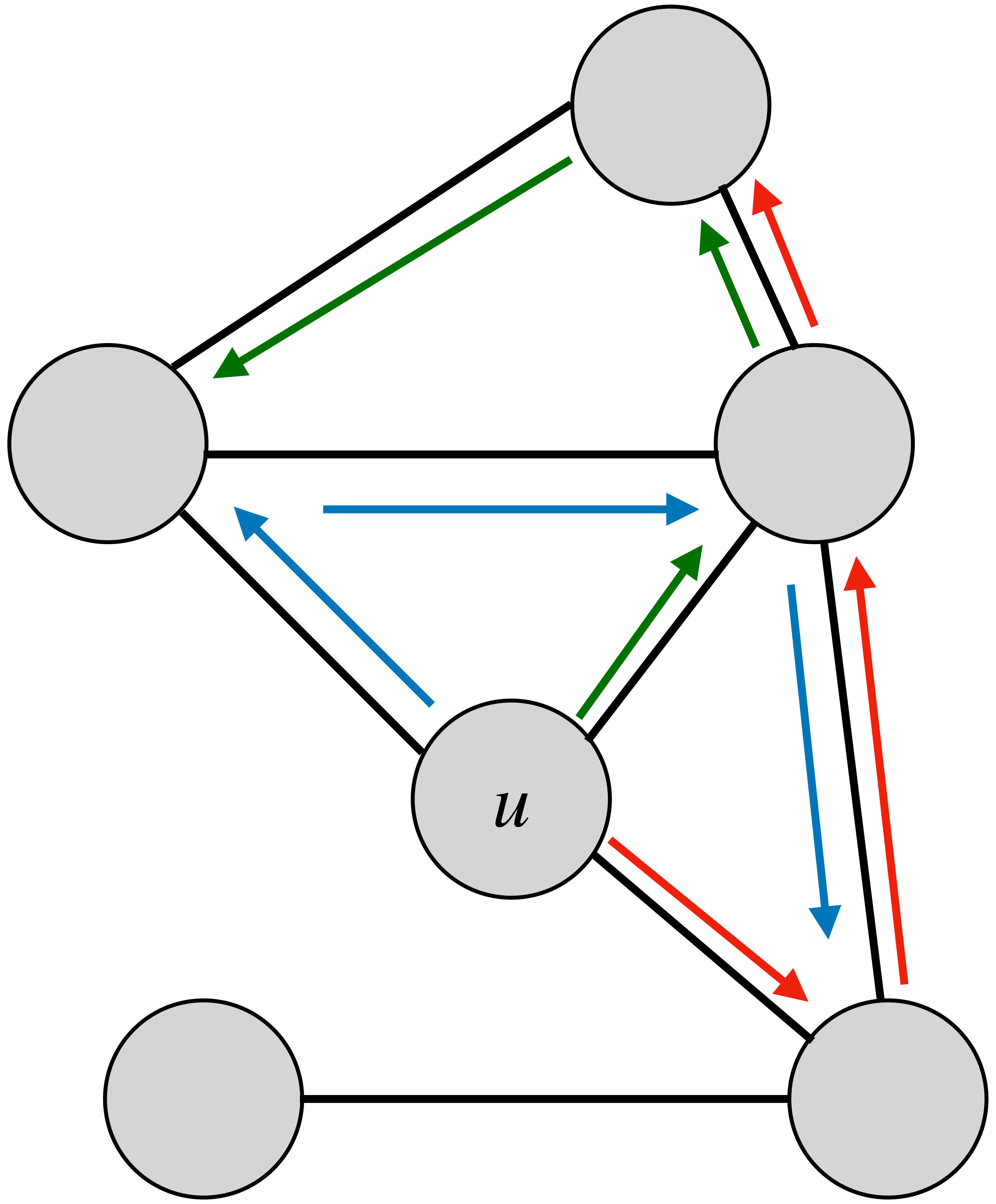


- "Correlated" -- exactly one walk through each edge at each time step
- "Random" -- The law of each individual walk is random

Correlated Random Walks [ALM17]

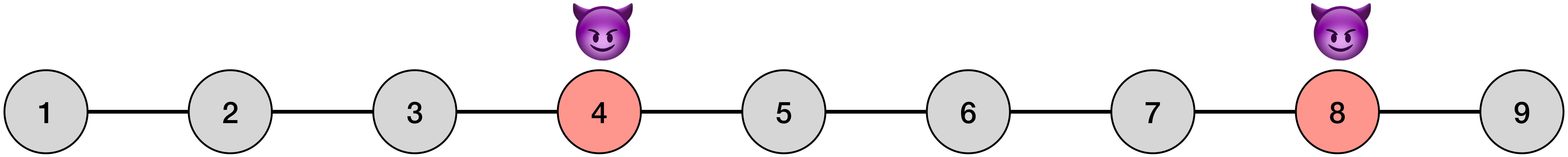
This establishes several (correlated) random walks

(Walks start from every node, not just u)

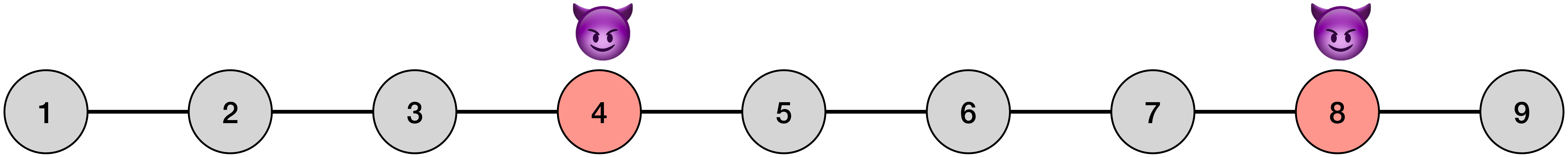


- "Correlated" -- exactly one walk through each edge at each time step
- "Random" -- The law of each individual walk is random
- Maybe we can design protocols just for paths now?

Locally Simulatable MPC on a Path



Locally Simulatable MPC on a Path



- New Requirement: Can simulate views of 4 and 8 independently --

$$\{\mathsf{Sim}(x_4, y; r_4), \mathsf{Sim}(x_8, y; r_8)\} \approx \mathsf{View}(\Pi)_{4,8}$$

- Namely, r_4, r_8 independent

Reducing THC to Locally Simulatable OR on a Path

Reducing THC to Locally Simulatable OR on a Path

- Fact: Once you have Topology-Hiding Broadcast (THB), can easily build THC

Reducing THC to Locally Simulatable OR on a Path

- Fact: Once you have Topology-Hiding Broadcast (THB), can easily build THC
  - Using THB, can first instantiate (topology-hiding) secure channels between parties using Key Exchange

Reducing THC to Locally Simulatable OR on a Path

- Fact: Once you have Topology-Hiding Broadcast (THB), can easily build THC
  - Using THB, can first instantiate (topology-hiding) secure channels between parties using Key Exchange
  - With these secure channels, can use any (standard) MPC to get THC

Reducing THC to Locally Simulatable OR on a Path

- Fact: Once you have Topology-Hiding Broadcast (THB), can easily build THC
  - Using THB, can first instantiate (topology-hiding) secure channels between parties using Key Exchange
  - With these secure channels, can use any (standard) MPC to get THC
- Broadcast == OR function where broadcaster inputs broadcast bit, everyone else inputs 0

Reducing THC to Locally Simulatable OR on a Path

THB Protocol

Reducing THC to Locally Simulatable OR on a Path

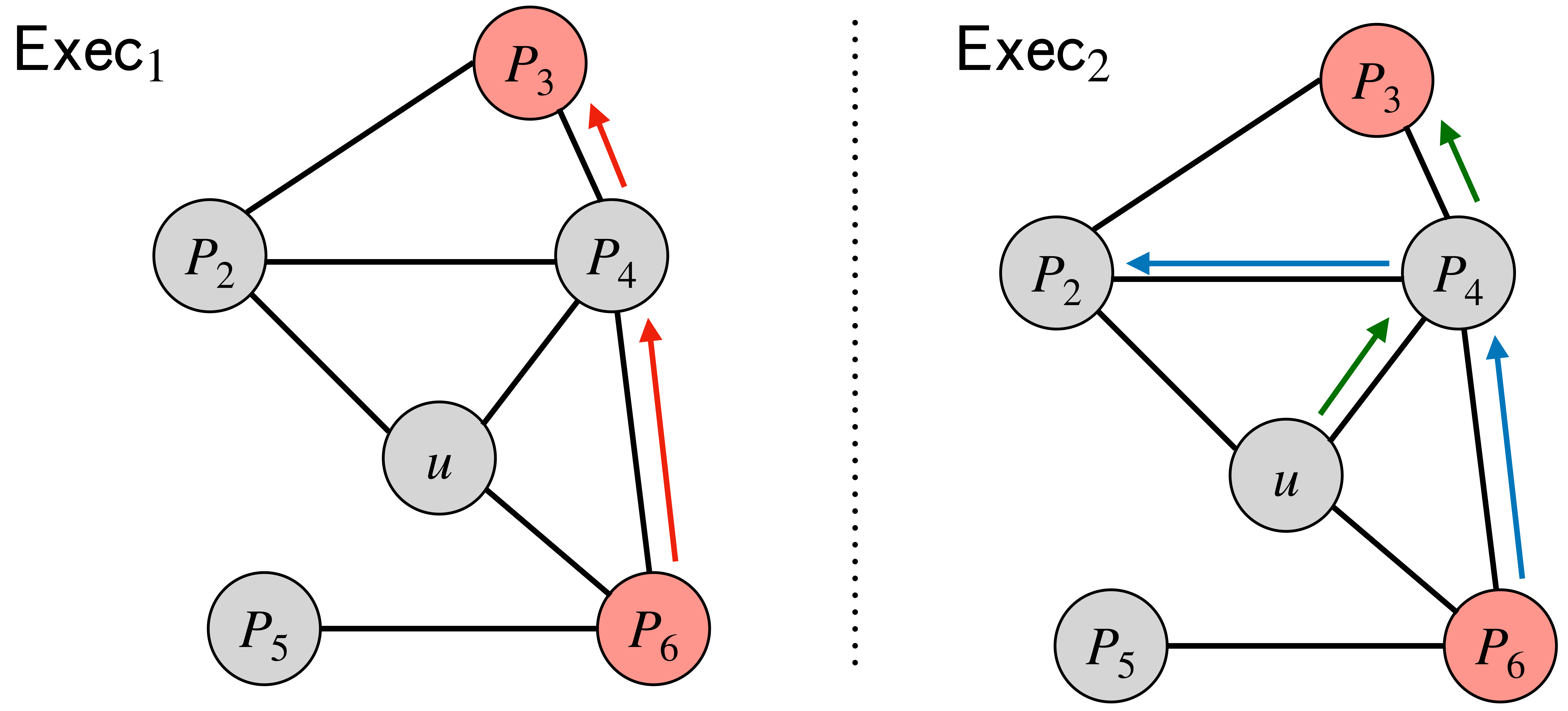
THB Protocol

- Π: Build Correlated Random Walks, and for each walk, run Locally Simulatable OR protocol

Reducing THC to Locally Simulatable OR on a Path

THB Protocol

- Π: Build Correlated Random Walks, and for each walk, run Locally Simulatable OR protocol

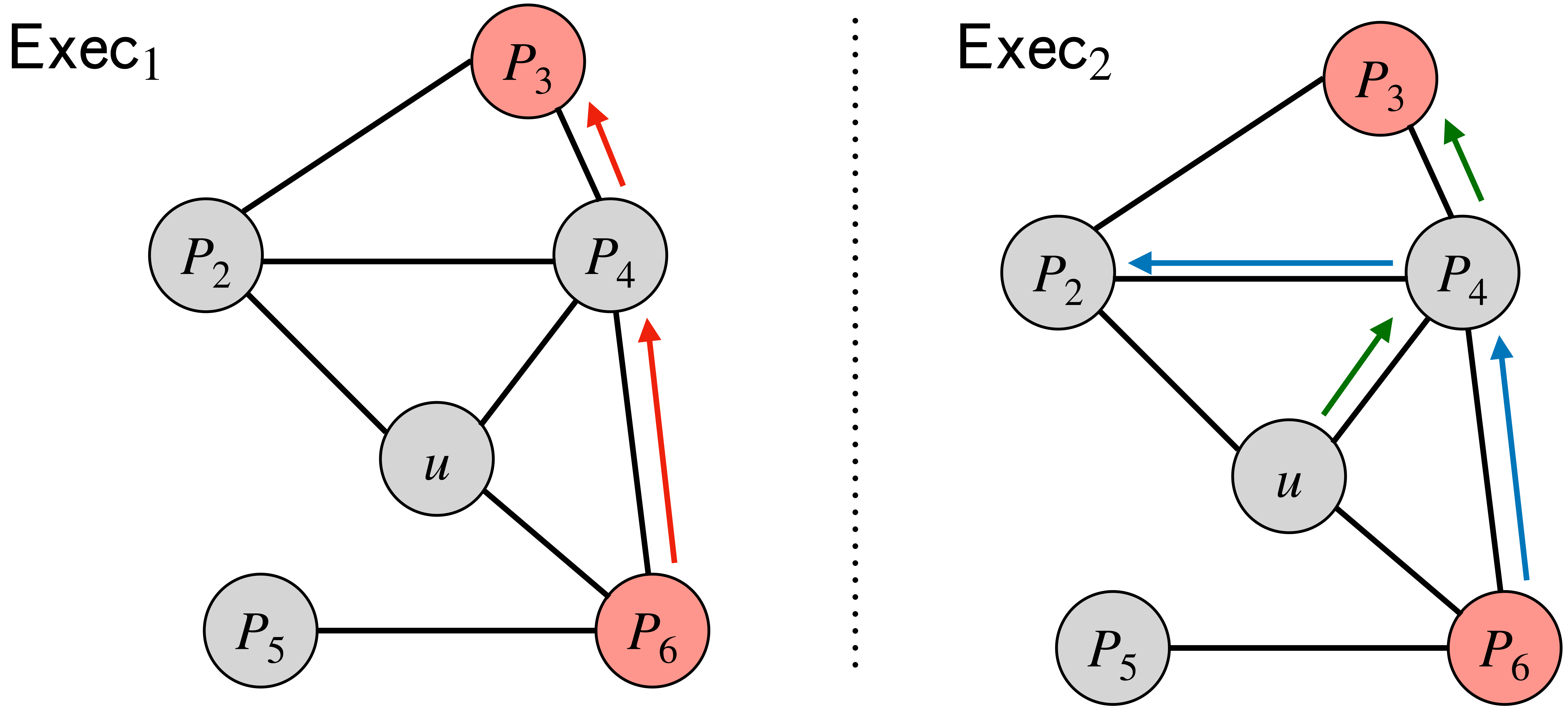


Reducing THC to Locally Simulatable OR on a Path

THB Protocol

- Π: Build Correlated Random Walks, and for each walk, run Locally Simulatable OR protocol

- From view of P_3,P_6, these look the same due to locally simulatability



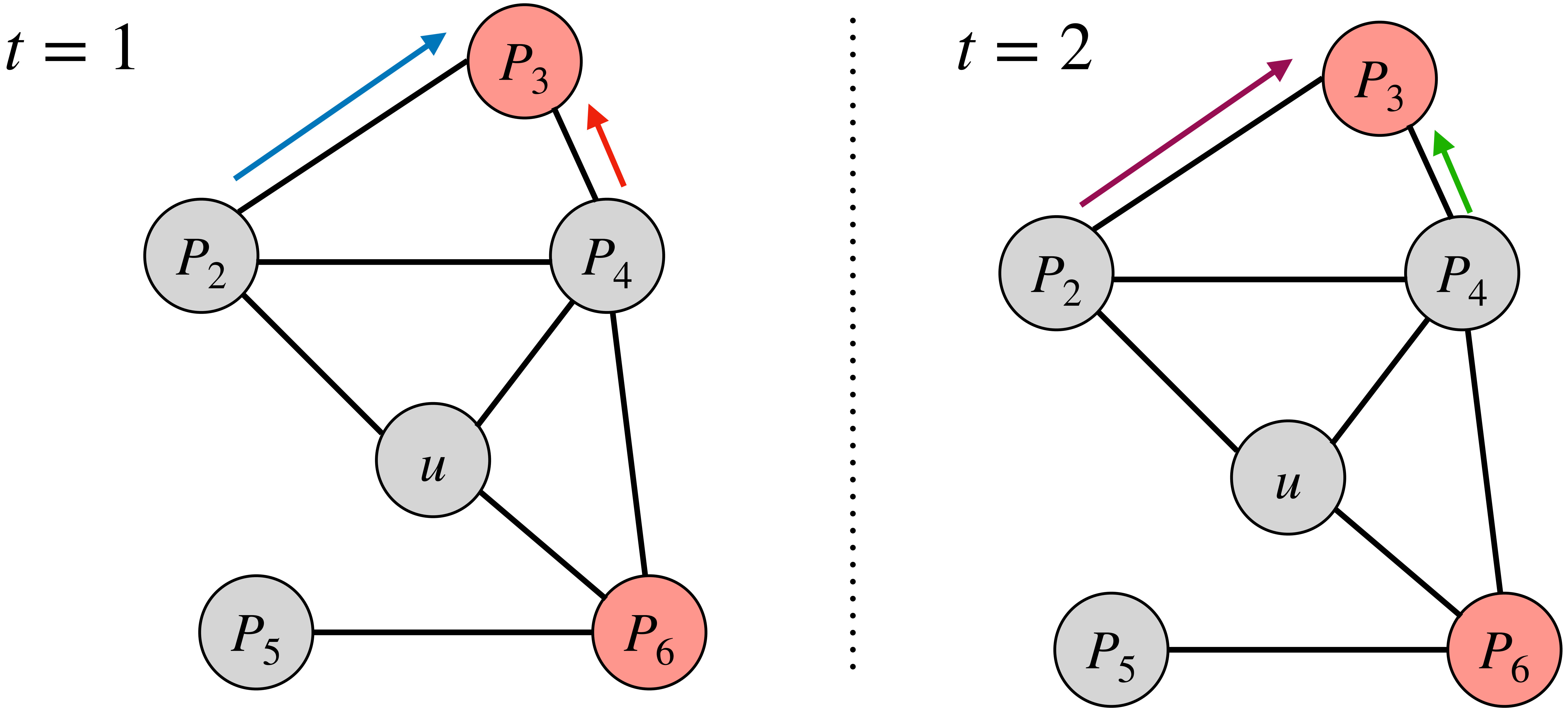
Reduction to Locally Simulatable OR on a Path

Reduction to Locally Simulatable OR on a Path

- For each step t, just need to simulate independently P_3's view as the t-th node on several paths (same for P_6)

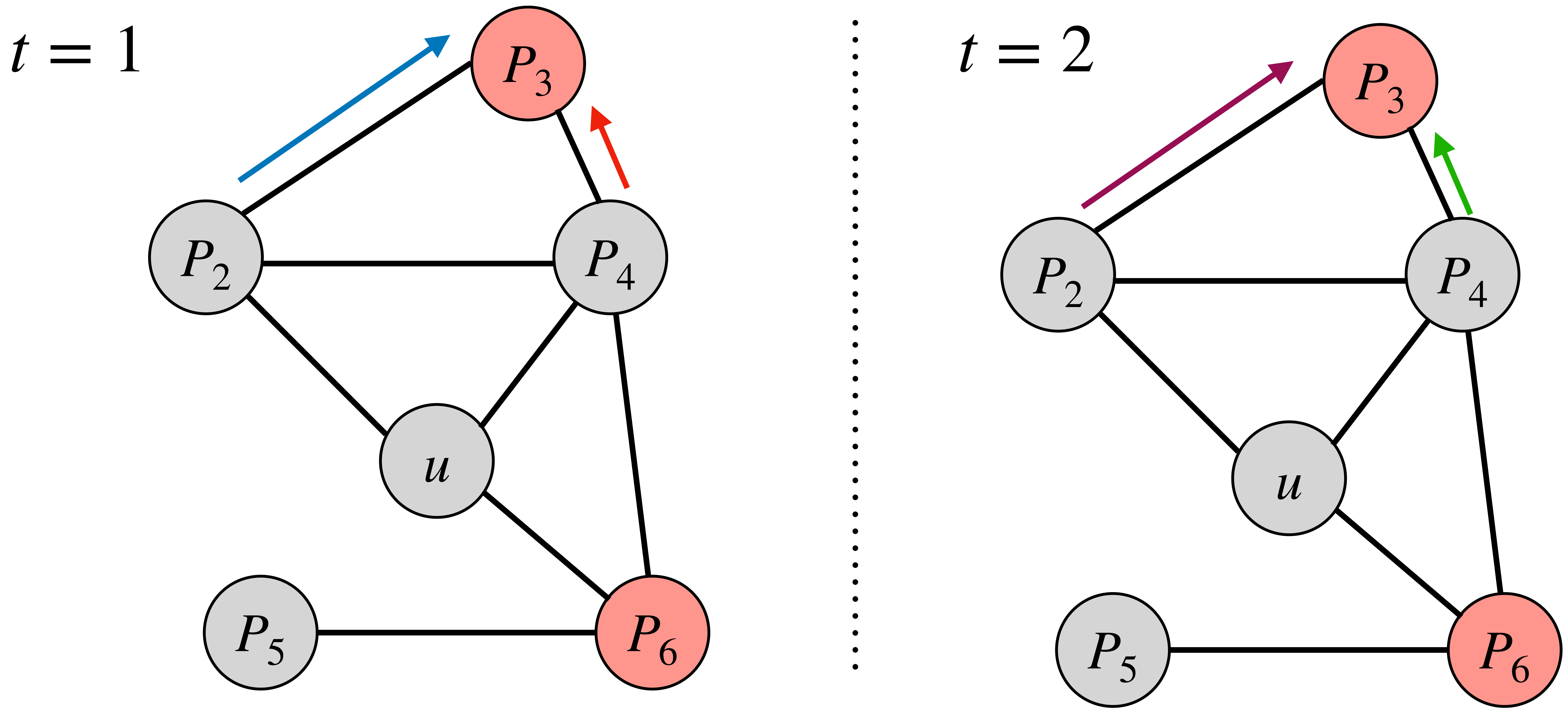
Reduction to Locally Simulatable OR on a Path

- For each step t, just need to simulate independently P_3's view as the t-th node on several paths (same for P_6)



Reduction to Locally Simulatable OR on a Path

- For each step t, just need to simulate independently P_3's view as the t-th node on several paths (same for P_6)
  - Can be done using local simulation



Locally Simulatable OR on a Path

$n = 8$ **parties**

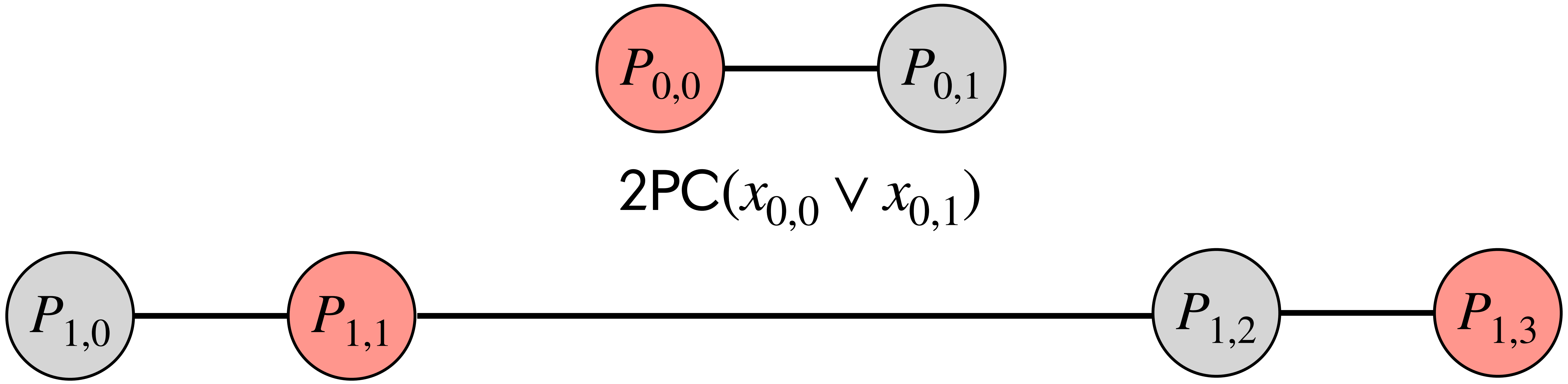
Locally Simulatable OR on a Path

$n = 8$ parties

2PC$(x_{0,0} \vee x_{0,1})$

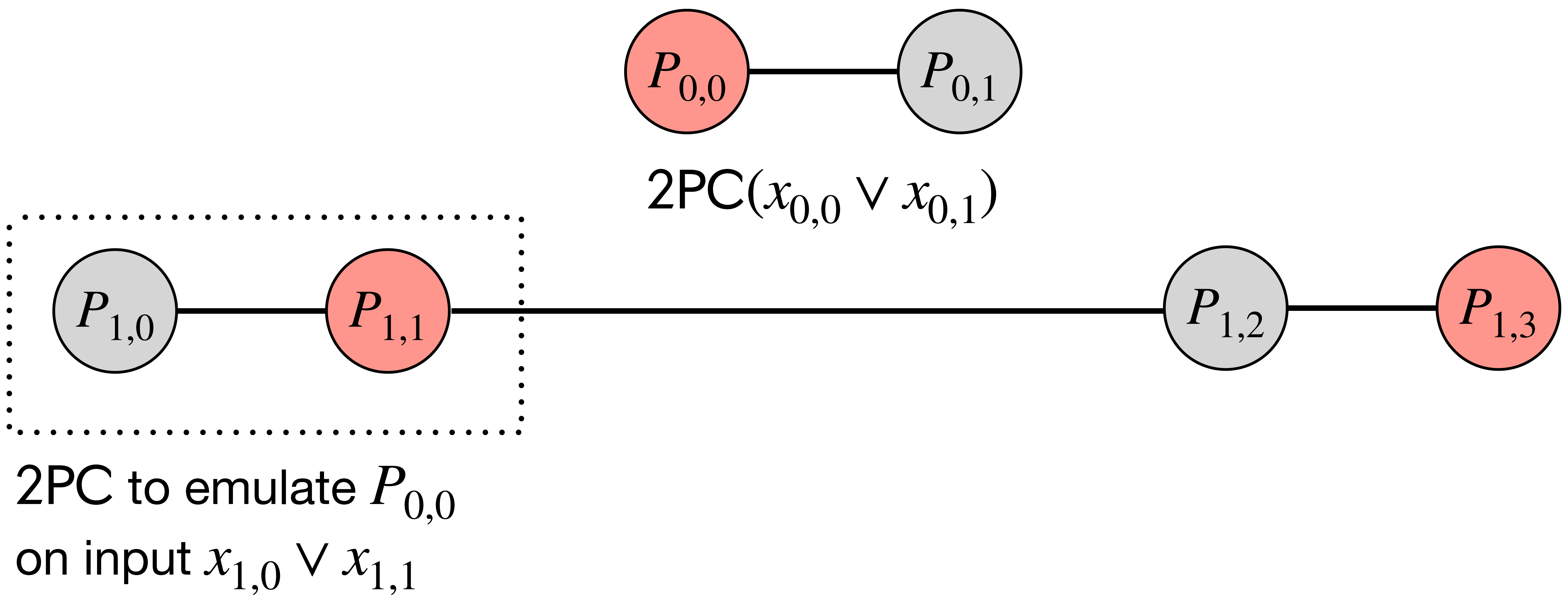
Locally Simulatable OR on a Path

$n = 8$ parties



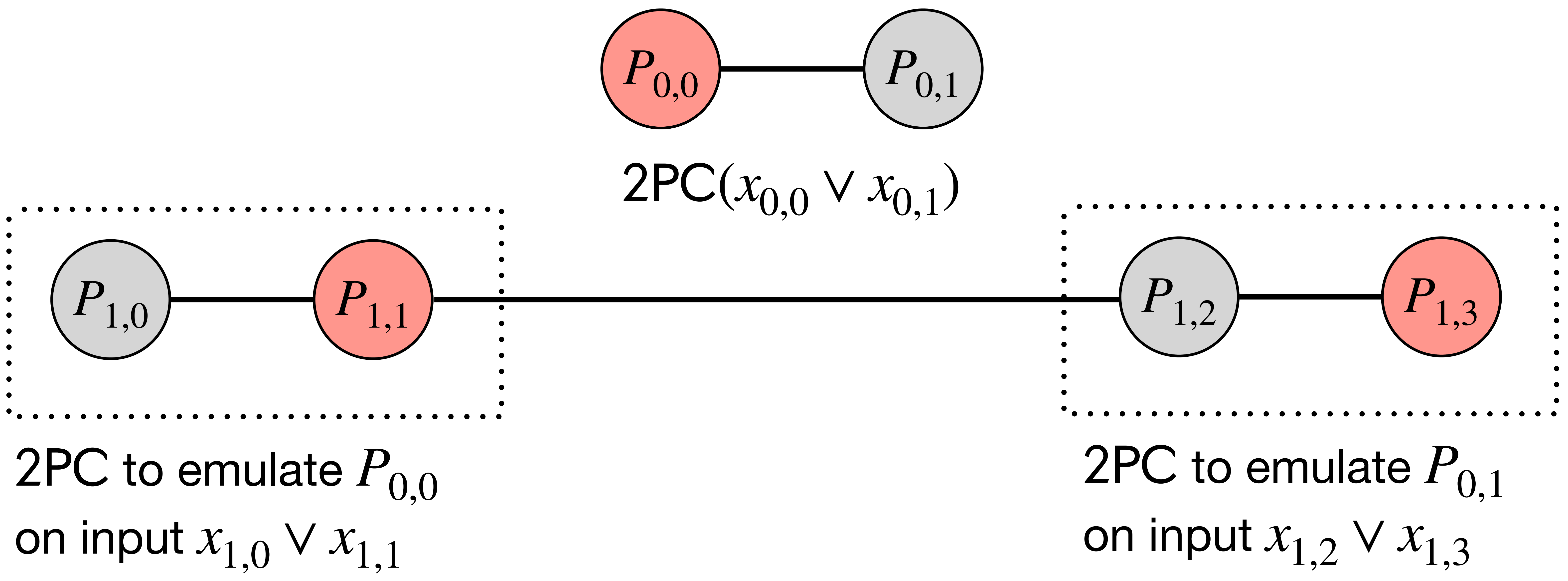
Locally Simulatable OR on a Path

$n = 8$ parties



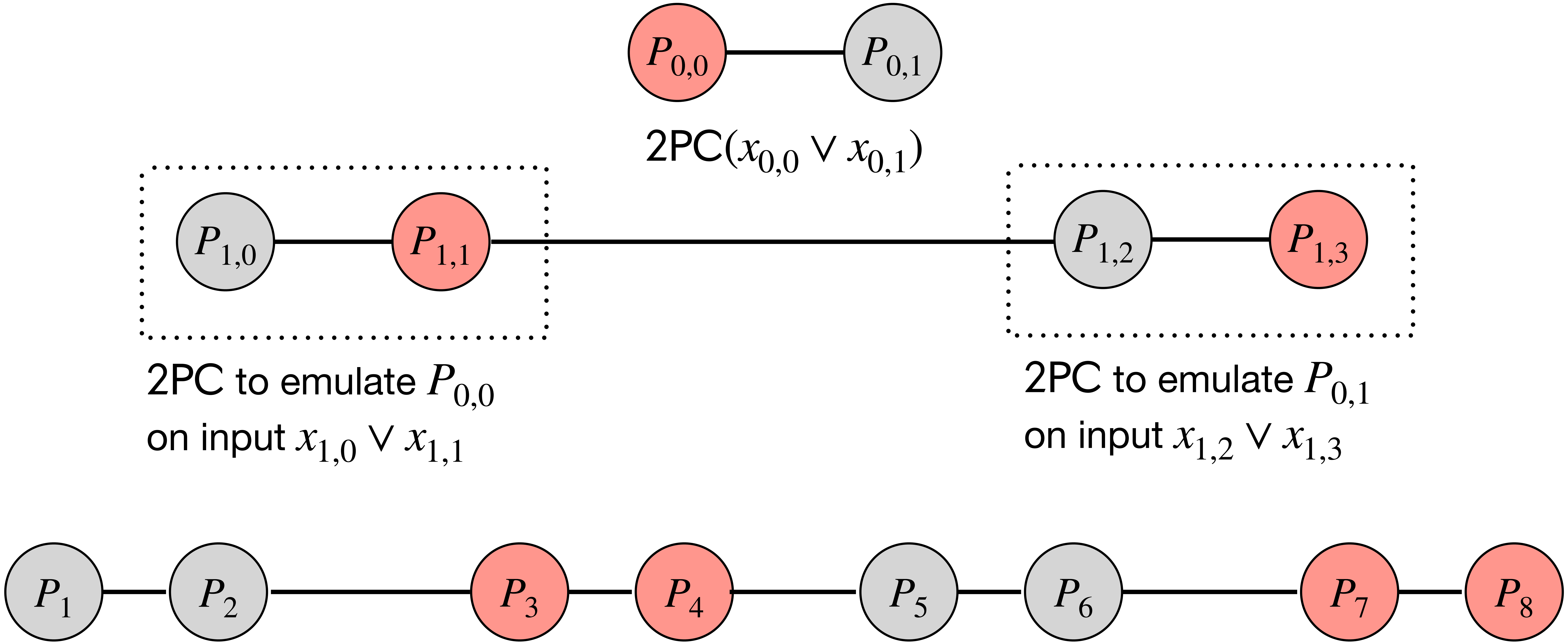
Locally Simulatable OR on a Path

$n = 8$ parties



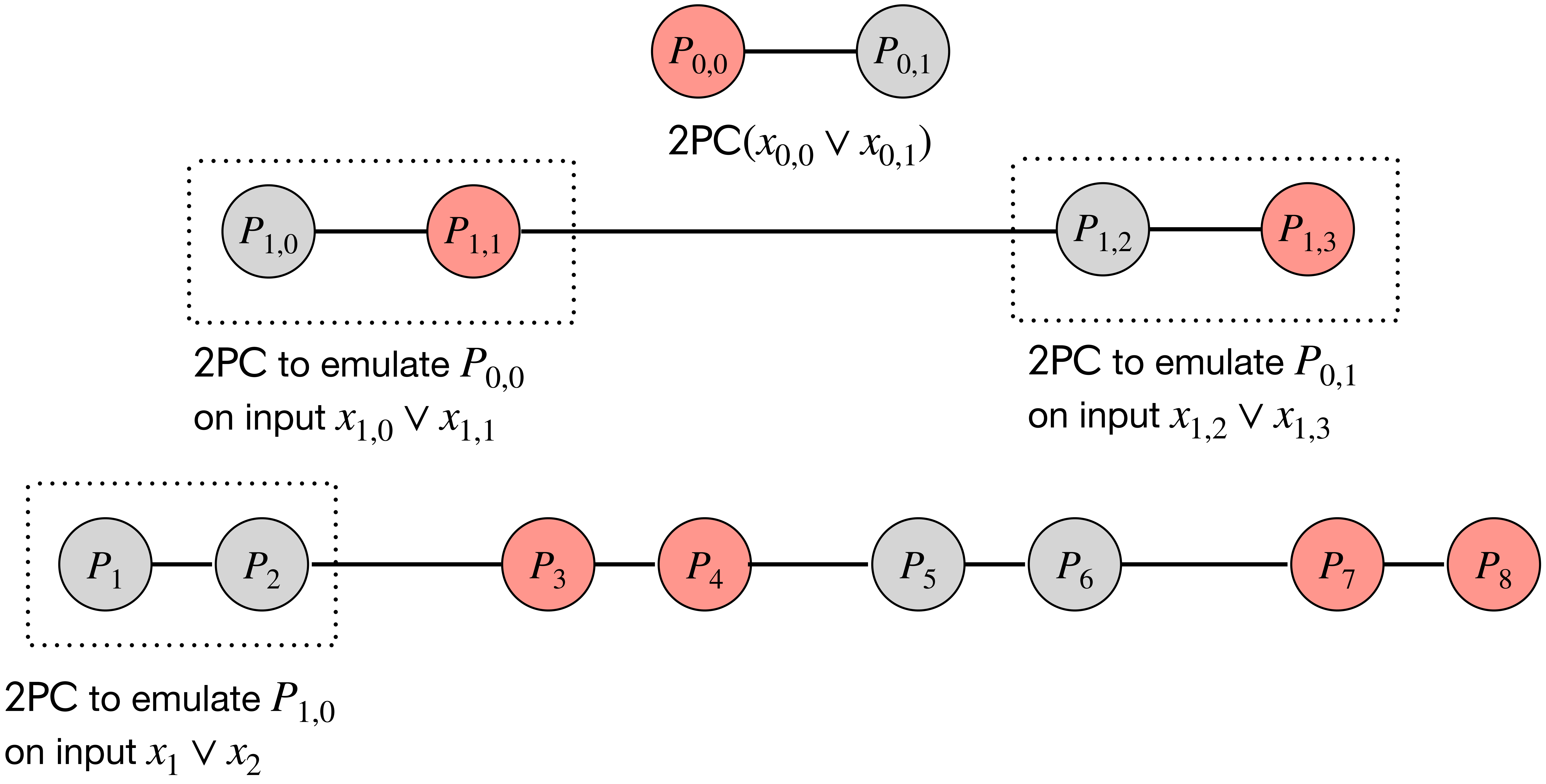
Locally Simulatable OR on a Path

$n = 8$ parties



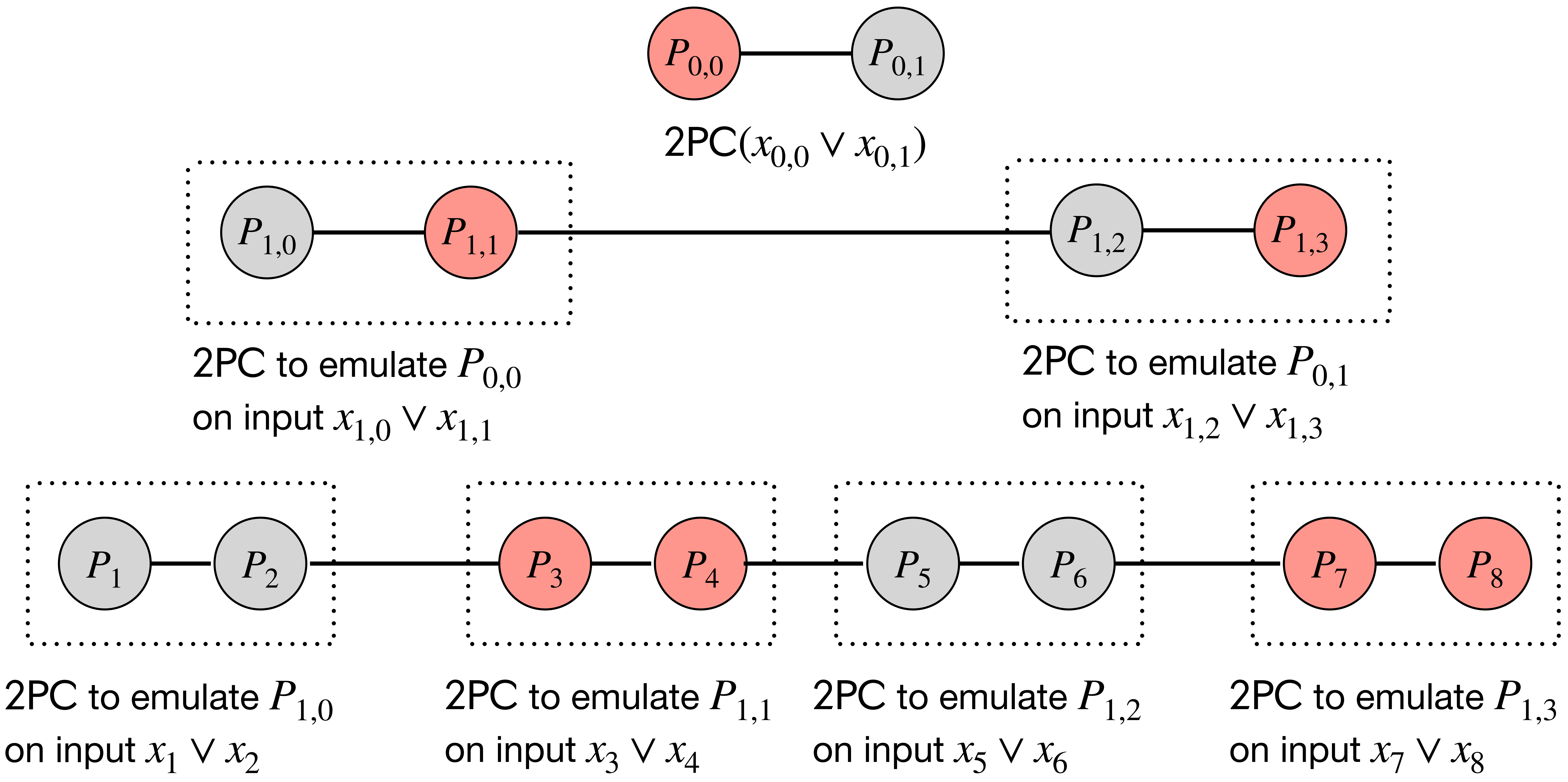
Locally Simulatable OR on a Path

$n = 8$ parties



Locally Simulatable OR on a Path

$n = 8$ **parties**



Locally Simulatable OR on a Path

Some Remarks

Locally Simulatable OR on a Path

Some Remarks

- Cannot use [MOR15] protocol here (diameter is n -- so too inefficient)
  - [MOR15] is also recursive; but ours is more efficient

Locally Simulatable OR on a Path

Some Remarks

- Cannot use [MOR15] protocol here (diameter is n -- so too inefficient)
  - [MOR15] is also recursive; but ours is more efficient
- Some subtleties in proof
  - Need to simulate messages for virtual party without entire input (not an issue for [MOR15])

Locally Simulatable OR on a Path

Some Remarks

- Cannot use [MOR15] protocol here (diameter is n -- so too inefficient)
  - [MOR15] is also recursive; but ours is more efficient
- Some subtleties in proof
  - Need to simulate messages for virtual party without entire input (not an issue for [MOR15])
    - If OR = 0, Sim knows all honest inputs = 0 (easy)

Locally Simulatable OR on a Path

Some Remarks

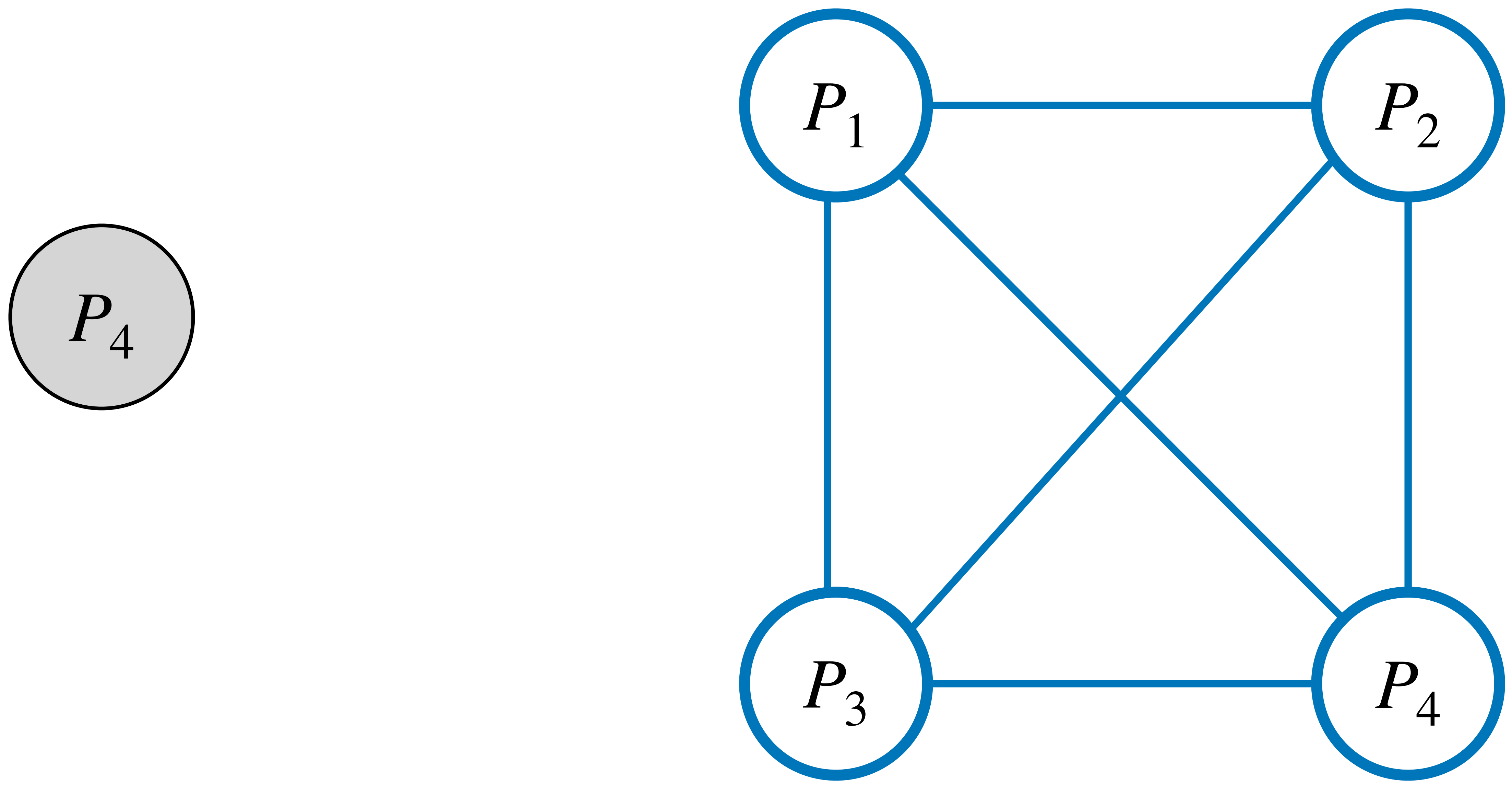
- Cannot use [MOR15] protocol here (diameter is n -- so too inefficient)
  - [MOR15] is also recursive; but ours is more efficient
- Some subtleties in proof
  - Need to simulate messages for virtual party without entire input (not an issue for [MOR15])
    - If OR = 0, Sim knows all honest inputs = 0 (easy)
    - If OR = 1, Sim pretends all honest inputs = 1 -- OK since simulated messages should be independent of virtual party's input

Conclusion

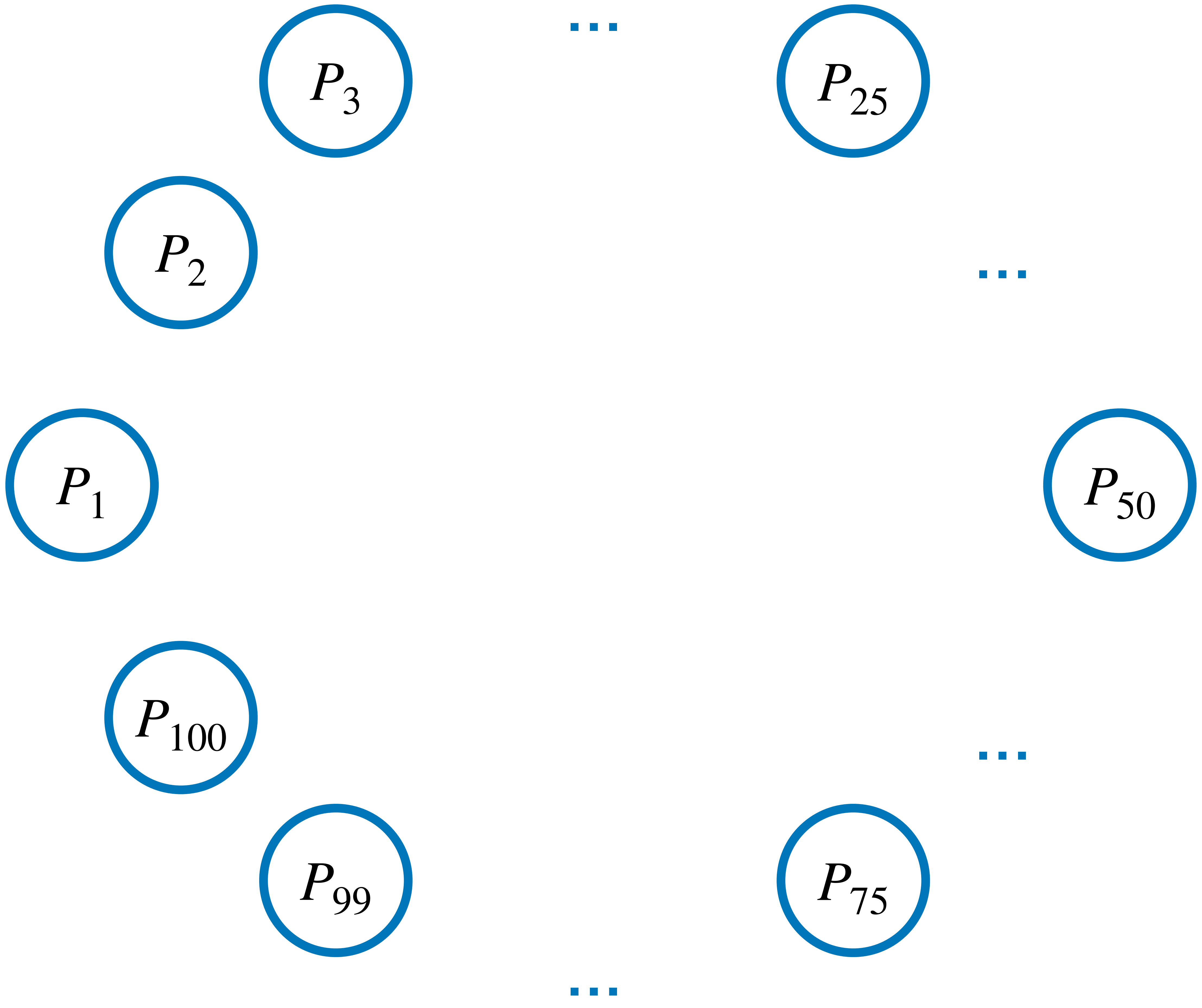
- We build THC from constant round 2PC with constant overhead
    - First such result for all graph classes (even with constant round/overhead)
- We define Locally Simulatable MPC (may be of independent interest)
- **Still Open:**
    - THC for all graph classes with *only* (polynomial-round) Oblivious Transfer?

Thanks!

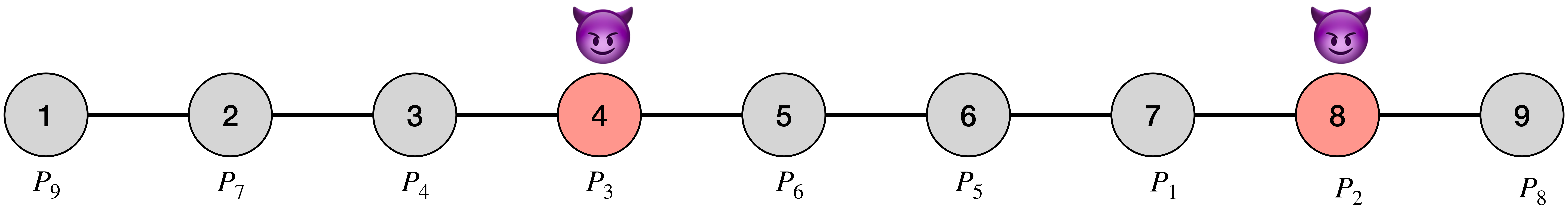
What is Topology-Hiding Computation (THC)?



What is Topology-Hiding Computation (THC)?



Local Simulation on a Path

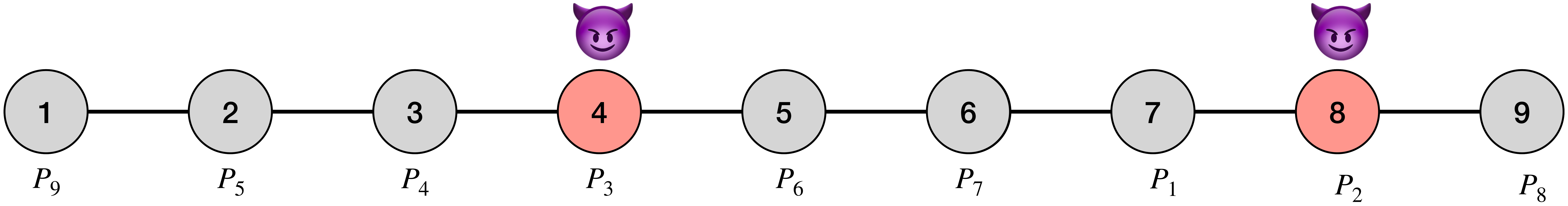


- Requirement: Can simulate views of P_3 and P_2 independently --

$$\{\mathsf{Sim}(x_3, y, (4\text{-}3\text{-}6)), \mathsf{Sim}(x_2, y, (1\text{-}2\text{-}8))\} \approx \mathsf{View}(\Pi)_{3,2}$$

- No Topology-Hiding (Directly) Needed!!

Local Simulation on a Path



- Requirement: Can simulate views of P_3 and P_2 independently --

$$\{\mathsf{Sim}(x_3, y, (4\text{-}3\text{-}6)), \mathsf{Sim}(x_2, y, (1\text{-}2\text{-}8))\} \approx \mathsf{View}(\Pi)_{3,2}$$

- No Topology-Hiding (Directly) Needed!!