Awake Efficient Distibuted Algorithms

William K. Moses Jr. (Billy)

Durham University

ADGA 2025

October 27th, 2025

Today's Plan

- Introduction to the rest of the talk
  - Motivation
  - Model
  - Brief mention of the themes of the talk
- Leveraging sleep to gain something
- Dealing with global problems - enforcing structure on communication
- Leveraging structure on communication in local problems
- Final thoughts

Outline

Motivation

- Metric with increasing importance today - energy
- Multiple ways to approach this
  Simple approach - if a device not in use, turn it off temporarily
- Studied in wireless networks for multiple decades now [Nakano & Olariu, ICPP 2000]
- Recently started to be studied in wired networks [Chatterjee, Gmyr, & Pandurangan, PODC 2020]
- Since then, a plethora of work has come out in the wired setting [Chatterjee, Gmyr, & Pandurangan, PODC 2020, Barenboim & Maimon, DISC 2021, Ghaffari & Portmann, SPAA 2022, Dufoulon, Moses Jr., & Pandurangan, PODC 2023, Ghaffari & Portmann, PODC 2023, Augustine, Moses Jr., & Pandurangan, SIROCCO 2024, Ghaffari & Trygub, PODC 2024, Balliu, Fraigniaud, Olivetti, & Rabie, PODC 2025]

Motivation

- Metric with increasing importance today - energy
- Multiple ways to approach this
  Simple approach - if a device not in use, turn it off temporarily
- Studied in wireless networks for multiple decades now [Nakano & Olariu, ICPP 2000]
- Recently started to be studied in wired networks [Chatterjee, Gmyr, & Pandurangan, PODC 2020]
- Since then, a plethora of work has come out in the wired setting [Chatterjee, Gmyr, & Pandurangan, PODC 2020, Barenboim & Maimon, DISC 2021, Ghaffari & Portmann, SPAA 2022, Dufoulon, Moses Jr., & Pandurangan, PODC 2023, Ghaffari & Portmann, PODC 2023, Augustine, Moses Jr., & Pandurangan, SIROCCO 2024, Ghaffari & Trygub, PODC 2024, Balliu, Fraigniaud, Olivetti, & Rabie, PODC 2025]

Motivation

- Metric with increasing importance today - energy
- Multiple ways to approach this
  Simple approach - if a device not in use, turn it off temporarily
- Studied in wireless networks for multiple decades now [Nakano & Olariu, ICPP 2000]
- Recently started to be studied in wired networks [Chatterjee, Gmyr, & Pandurangan, PODC 2020]
- Since then, a plethora of work has come out in the wired setting [Chatterjee, Gmyr, & Pandurangan, PODC 2020, Barenboim & Maimon, DISC 2021, Ghaffari & Portmann, SPAA 2022, Dufoulon, Moses Jr., & Pandurangan, PODC 2023, Ghaffari & Portmann, PODC 2023, Augustine, Moses Jr., & Pandurangan, SIROCCO 2024, Ghaffari & Trygub, PODC 2024, Balliu, Fraigniaud, Olivetti, & Rabie, PODC 2025]

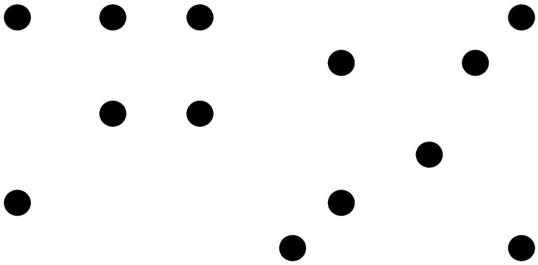
Motivation

- Metric with increasing importance today - energy
- Multiple ways to approach this
  Simple approach - if a device not in use, turn it off temporarily
- Studied in wireless networks for multiple decades now [Nakano & Olariu, ICPP 2000]
- Recently started to be studied in wired networks [Chatterjee, Gmyr, & Pandurangan, PODC 2020]
- Since then, a plethora of work has come out in the wired setting [Chatterjee, Gmyr, & Pandurangan, PODC 2020, Barenboim & Maimon, DISC 2021, Ghaffari & Portmann, SPAA 2022, Dufoulon, Moses Jr., & Pandurangan, PODC 2023, Ghaffari & Portmann, PODC 2023, Augustine, Moses Jr., & Pandurangan, SIROCCO 2024, Ghaffari & Trygub, PODC 2024, Balliu, Fraigniaud, Olivetti, & Rabie, PODC 2025]

Motivation

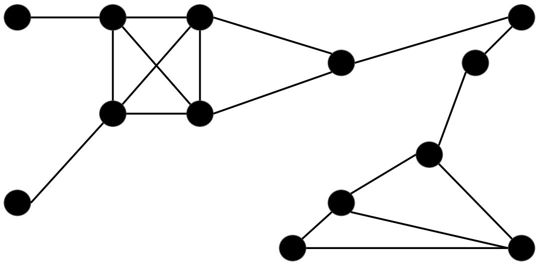
- Metric with increasing importance today - energy
- Multiple ways to approach this
  Simple approach - if a device not in use, turn it off temporarily
- Studied in wireless networks for multiple decades now [Nakano & Olariu, ICPP 2000]
- Recently started to be studied in wired networks [Chatterjee, Gmyr, & Pandurangan, PODC 2020]
- Since then, a plethora of work has come out in the wired setting [Chatterjee, Gmyr, & Pandurangan, PODC 2020, Barenboim & Maimon, DISC 2021, Ghaffari & Portmann, SPAA 2022, Dufoulon, Moses Jr., & Pandurangan, PODC 2023, Ghaffari & Portmann, PODC 2023, Augustine, Moses Jr., & Pandurangan, SIROCCO 2024, Ghaffari & Trygub, PODC 2024, Balliu, Fraigniaud, Olivetti, & Rabie, PODC 2025]

Distributed Computing Model



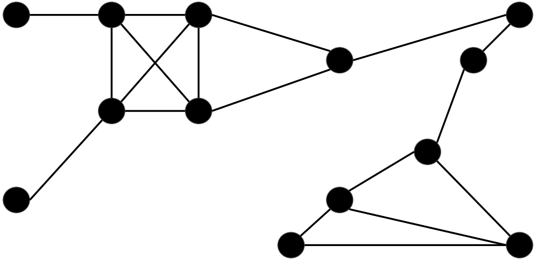
- n nodes with unique IDs $\in [1, N]$, $N = poly(n)$
- Arbitrary graph
- Global knowledge: n known to all nodes, sometimes N known to all nodes
- CONGEST model: $O(\log n)$ bits per message over an edge;
  LOCAL model: unlimited bandwidth

Distributed Computing Model



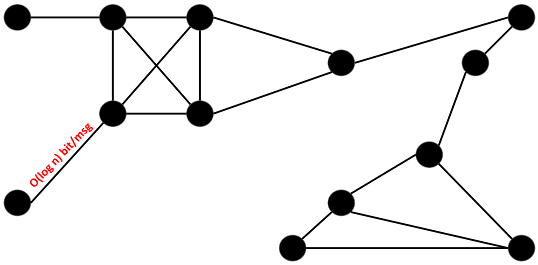
- n nodes with unique IDs $\in [1, N]$, $N = poly(n)$
- Arbitrary graph
- Global knowledge: n known to all nodes, sometimes N known to all nodes
- CONGEST model: $O(\log n)$ bits per message over an edge;
  LOCAL model: unlimited bandwidth

Distributed Computing Model



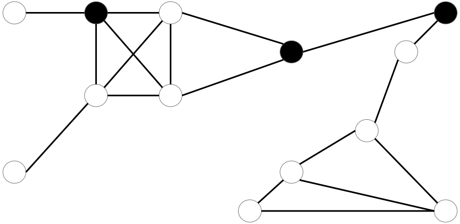
- n nodes with unique IDs $\in [1, N]$, $N = poly(n)$
- Arbitrary graph
- Global knowledge: n known to all nodes, sometimes N known to all nodes
- CONGEST model: $O(\log n)$ bits per message over an edge;
  LOCAL model: unlimited bandwidth

Distributed Computing Model



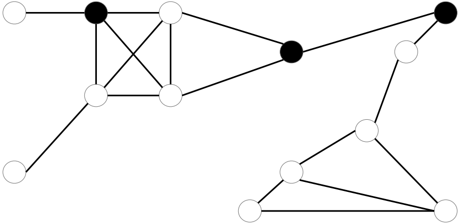
- n nodes with unique IDs $\in [1, N]$, $N = poly(n)$
- Arbitrary graph
- Global knowledge: n known to all nodes, sometimes N known to all nodes
- CONGEST model: $O(\log n)$ bits per message over an edge;
  LOCAL model: unlimited bandwidth

Awake/Energy Efficiency Defined & Measured



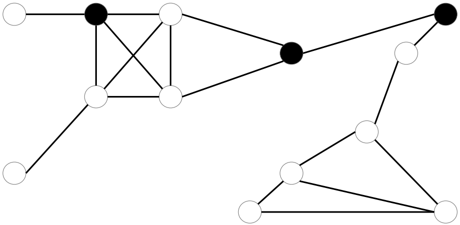
- Synchronous system
- In each round, each node's state $\in$ {awake, asleep}
- Algorithm designer controls when nodes are awake, asleep. Nodes know current round #
- Message exchange on an edge (u, v) only possible when u,v awake in same round
- **Metric 1:** awake time; **Metric 2:** running time/round complexity

Awake/Energy Efficiency Defined & Measured



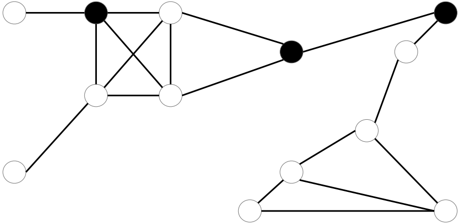
- Synchronous system
- In each round, each node's state $\in$ {awake, asleep}
- Algorithm designer controls when nodes are awake, asleep. Nodes know current round #
- Message exchange on an edge (u, v) only possible when u,v awake in same round
- **Metric 1:** awake time; **Metric 2:** running time/round complexity

Awake/Energy Efficiency Defined & Measured



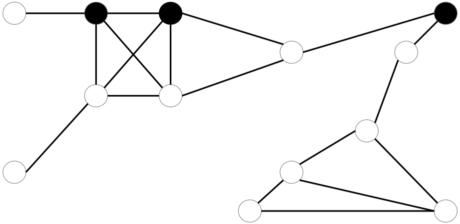
- Synchronous system
- In each round, each node's state $\in$ {awake, asleep}
- Algorithm designer controls when nodes are awake, asleep. Nodes know current round #
- Message exchange on an edge (u, v) only possible when u,v awake in same round
- **Metric 1:** awake time; **Metric 2:** running time/round complexity

Awake/Energy Efficiency Defined & Measured



- Synchronous system
- In each round, each node's state $\in$ {awake, asleep}
- Algorithm designer controls when nodes are awake, asleep. Nodes know current round #
- Message exchange on an edge (u, v) only possible when u,v awake in same round
- **Metric 1:** awake time; **Metric 2:** running time/round complexity

Awake/Energy Efficiency Defined & Measured



- Synchronous system
- In each round, each node's state $\in$ {awake, asleep}
- Algorithm designer controls when nodes are awake, asleep. Nodes know current round #
- Message exchange on an edge (u, v) only possible when u,v awake in same round
- **Metric 1:** awake time; **Metric 2:** running time/round complexity

General Themes of Talk

- Biggest algorithmic design hurdle: Figuring out when to communicate between neighbors
- Think of solutions as finding the communication structure inherent to pre-existing techniques, and then exploiting them
- It's ok to lose run time to enforce a structure which saves us awake time
- A little structure goes a long way, even for local problems

General Themes of Talk

- Biggest algorithmic design hurdle: Figuring out when to communicate between neighbors
- Think of solutions as finding the communication structure inherent to pre-existing techniques, and then exploiting them
- It's ok to lose run time to enforce a structure which saves us awake time
- A little structure goes a long way, even for local problems

General Themes of Talk

- Biggest algorithmic design hurdle: Figuring out when to communicate between neighbors
- Think of solutions as finding the communication structure inherent to pre-existing techniques, and then exploiting them
- It's ok to lose run time to enforce a structure which saves us awake time
- A little structure goes a long way, even for local problems

General Themes of Talk

- Biggest algorithmic design hurdle: Figuring out when to communicate between neighbors
- Think of solutions as finding the communication structure inherent to pre-existing techniques, and then exploiting them
- It's ok to lose run time to enforce a structure which saves us awake time
- A little structure goes a long way, even for local problems

Outline

Section Outline

Plan for Section

- Introduce solution to Maximal Independent Set (MIS) from [Chatterjee, Gmyr, & Pandurangan, PODC 2020]
- **Big Idea:** Cleverly putting nodes to sleep saves average node-awake time
- **Why care:** Lower bound [Kuhn, Moscibroda, & Wattenhofer, JACM 2016] - $\Omega(\min\{\frac{\log \Delta}{\log \log \Delta}, \sqrt{\frac{\log n}{\log \log n}}\})$.
  For high degree graphs, if nodes always awake, then total energy/awake time - $\omega(n)$
- [Chatterjee, Gmyr, & Pandurangan, PODC 2020]: $O(1)$ node-average awake time $\implies$ $O(n)$ total energy/awake time

Section Outline

Plan for Section

- Introduce solution to Maximal Independent Set (MIS) from [Chatterjee, Gmyr, & Pandurangan, PODC 2020]
- **Big Idea:** Cleverly putting nodes to sleep saves average node-awake time
- **Why care:** Lower bound [Kuhn, Moscibroda, & Wattenhofer, JACM 2016] - $\Omega(\min\{\frac{\log \Delta}{\log\log \Delta}, \sqrt{\frac{\log n}{\log\log n}}\})$.
  For high degree graphs, if nodes always awake, then total energy/awake time - $\omega(n)$
- [Chatterjee, Gmyr, & Pandurangan, PODC 2020]: $O(1)$ node-average awake time $\implies$ $O(n)$ total energy/awake time

Section Outline

Plan for Section

- Introduce solution to Maximal Independent Set (MIS) from [Chatterjee, Gmyr, & Pandurangan, PODC 2020]
- **Big Idea:** Cleverly putting nodes to sleep saves average node-awake time
- **Why care:** Lower bound [Kuhn, Moscibroda, & Wattenhofer, JACM 2016] - $\Omega(\min\{\frac{\log \Delta}{\log\log \Delta}, \sqrt{\frac{\log n}{\log\log n}}\})$.
  For high degree graphs, if nodes always awake, then total energy/awake time - $\omega(n)$
- [Chatterjee, Gmyr, & Pandurangan, PODC 2020]: $O(1)$ node-average awake time $\implies$ $O(n)$ total energy/awake time

Section Outline

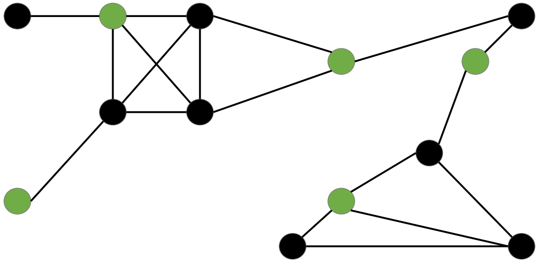
Plan for Section

- Introduce solution to Maximal Independent Set (MIS) from [Chatterjee, Gmyr, & Pandurangan, PODC 2020]
- **Big Idea:** Cleverly putting nodes to sleep saves average node-awake time
- **Why care:** Lower bound [Kuhn, Moscibroda, & Wattenhofer, JACM 2016] - $\Omega(\min\{\frac{\log \Delta}{\log \log \Delta}, \sqrt{\frac{\log n}{\log \log n}}\})$.
  For high degree graphs, if nodes always awake, then total energy/awake time - $\omega(n)$
- [Chatterjee, Gmyr, & Pandurangan, PODC 2020]: $O(1)$ node-average awake time $\implies$ $O(n)$ total energy/awake time

Maximal Independent Set (MIS) - Problem Definition

Maximal Independent Set (MIS)

For a graph $G(V, E)$, the maximal independent set of G is a set of nodes S such that for all nodes $u, v \in S$, the edge (u, v) does not exist and every node not in S has a neighbor in S.



Solution Outline

Each node samples $K = O(\log n)$ fair coin tosses

Each node participates in a recursive process MIS(K):

- Bottom out - either MIS(0) or you're awake & no one around you is in MIS (you're isolated): join MIS
- If undecided: if Kth coin toss heads participate in MIS($K - 1$), else sleep for duration of MIS(K-1)
- Inform neighbors of updates to your state (synchronize)
- If neighbor in MIS, don't join MIS
- If you become isolated, then join MIS
- If you're still undecided, participate in MIS($K - 1$), else sleep

Solution Outline

Each node samples $K = O(\log n)$ fair coin tosses

Each node participates in a recursive process MIS(K):

- Bottom out - either MIS(0) or you're awake & no one around you is in MIS (you're isolated): join MIS
- If undecided: if Kth coin toss heads participate in MIS($K-1$), else sleep for duration of MIS(K-1)
- Inform neighbors of updates to your state (synchronize)
- If neighbor in MIS, don't join MIS
- If you become isolated, then join MIS
- If you're still undecided, participate in MIS($K-1$), else sleep

Solution Outline

Each node samples $K = O(\log n)$ fair coin tosses
Each node participates in a recursive process MIS(K):

- Bottom out - either MIS(0) or you're awake & no one around you is in MIS (you're isolated): join MIS
- If undecided: if Kth coin toss heads participate in MIS($K-1$), else sleep for duration of MIS(K-1)
- Inform neighbors of updates to your state (synchronize)
- If neighbor in MIS, don't join MIS
- If you become isolated, then join MIS
- If you're still undecided, participate in MIS($K-1$), else sleep

Solution Outline

Each node samples $K = O(\log n)$ fair coin tosses
Each node participates in a recursive process MIS(K):

- Bottom out - either MIS(0) or you're awake & no one around you is in MIS (you're isolated): join MIS
- If undecided: if Kth coin toss heads participate in MIS($K - 1$), else sleep for duration of MIS(K-1)
- Inform neighbors of updates to your state (synchronize)
- If neighbor in MIS, don't join MIS
- If you become isolated, then join MIS
- If you're still undecided, participate in MIS($K - 1$), else sleep

Solution Outline

Each node samples $K = O(\log n)$ fair coin tosses
Each node participates in a recursive process MIS(K):

- Bottom out - either MIS(0) or you're awake & no one around you is in MIS (you're isolated): join MIS
- If undecided: if Kth coin toss heads participate in MIS($K-1$), else sleep for duration of MIS(K-1)
- Inform neighbors of updates to your state (synchronize)
- If neighbor in MIS, don't join MIS
- If you become isolated, then join MIS
- If you're still undecided, participate in MIS($K-1$), else sleep

Solution Outline

Each node samples $K = O(\log n)$ fair coin tosses
Each node participates in a recursive process MIS(K):

- Bottom out - either MIS(0) or you're awake & no one around you is in MIS (you're isolated): join MIS
- If undecided: if Kth coin toss heads participate in MIS($K-1$), else sleep for duration of MIS(K-1)
- Inform neighbors of updates to your state (synchronize)
- If neighbor in MIS, don't join MIS
- If you become isolated, then join MIS
- If you're still undecided, participate in MIS($K-1$), else sleep

Solution Outline

Each node samples $K = O(\log n)$ fair coin tosses
Each node participates in a recursive process MIS(K):

- Bottom out - either MIS(0) or you're awake & no one around you is in MIS (you're isolated): join MIS
- If undecided: if Kth coin toss heads participate in MIS($K-1$), else sleep for duration of MIS(K-1)
- Inform neighbors of updates to your state (synchronize)
- If neighbor in MIS, don't join MIS
- If you become isolated, then join MIS
- If you're still undecided, participate in MIS($K-1$), else sleep

Solution Outline

Each node samples $K = O(\log n)$ fair coin tosses
Each node participates in a recursive process MIS(K):

- Bottom out - either MIS(0) or you're awake & no one around you is in MIS (you're isolated): join MIS
- If undecided: if Kth coin toss heads participate in MIS($K - 1$), else sleep for duration of MIS(K-1)
- Inform neighbors of updates to your state (synchronize)
- If neighbor in MIS, don't join MIS
- If you become isolated, then join MIS
- If you're still undecided, participate in MIS($K - 1$), else sleep

Analysis

- Simple analysis - each node participates in $O(\log n)$ recursion, so each node $O(\log n)$ awake time
- **Insight:** At each level of the recursion, set of awake nodes A and sleeping nodes S. Nodes in set $MIS(A)$ have neighbors in set S.
  A constant fraction of S are pruned thanks to the actions of awake nodes
- In particular, at every level, $1/4$ of nodes in S pruned by nodes in $MIS(A)$. So at each level of recursion i, $(3/4)^i n$ nodes participate.
  These pruned nodes each need only $O(1)$ awake time for whole MIS process
- For non-pruned nodes, node average awake time $= (1/n)\sum_{i=0}^{K}(3/4)^i \cdot n = O(1)$

Analysis

- Simple analysis - each node participates in $O(\log n)$ recursion, so each node $O(\log n)$ awake time
- **Insight:** At each level of the recursion, set of awake nodes A and sleeping nodes S.
  Nodes in set $MIS(A)$ have neighbors in set S.
  A constant fraction of S are pruned thanks to the actions of awake nodes
- In particular, at every level, $1/4$ of nodes in S pruned by nodes in $MIS(A)$. So at each level of recursion i, $(3/4)^i n$ nodes participate.
  These pruned nodes each need only $O(1)$ awake time for whole MIS process
- For non-pruned nodes, node average awake time $= (1/n) \sum_{i=0}^{K} (3/4)^i \cdot n = O(1)$

Analysis

- Simple analysis - each node participates in $O(\log n)$ recursion, so each node $O(\log n)$ awake time
- **Insight:** At each level of the recursion, set of awake nodes A and sleeping nodes S. Nodes in set $MIS(A)$ have neighbors in set S.
  A constant fraction of S are pruned thanks to the actions of awake nodes
- In particular, at every level, $1/4$ of nodes in S pruned by nodes in $MIS(A)$. So at each level of recursion i, $(3/4)^i n$ nodes participate.
  These pruned nodes each need only $O(1)$ awake time for whole MIS process
- For non-pruned nodes, node average awake time $= (1/n) \sum_{i=0}^{K} (3/4)^i \cdot n = O(1)$

Analysis

- Simple analysis - each node participates in $O(\log n)$ recursion, so each node $O(\log n)$ awake time
- **Insight:** At each level of the recursion, set of awake nodes A and sleeping nodes S.
  Nodes in set $MIS(A)$ have neighbors in set S.
  A constant fraction of S are pruned thanks to the actions of awake nodes
- In particular, at every level, $1/4$ of nodes in S pruned by nodes in $MIS(A)$. So at each level of recursion i, $(3/4)^i n$ nodes participate.
  These pruned nodes each need only $O(1)$ awake time for whole MIS process
- For non-pruned nodes, node average awake time $= (1/n)\sum_{i=0}^{K}(3/4)^i \cdot n = O(1)$

Notes

- The given solution sacrificed run time to save awake time
  $O(\log^{3.41})$ run time
- However, it's possible to avoid doing so [Ghaffari & Portmann, SPAA 2022]
  $O(\log n)$ run time
  Note that $O(1)$ node-average awake time is w.h.p.
- [Ghaffari & Portmann, SPAA 2022] also solves maximal matching awake efficiently

Notes

- The given solution sacrificed run time to save awake time
  $O(\log^{3.41})$ run time
- However, it's possible to avoid doing so [Ghaffari & Portmann, SPAA 2022]
  $O(\log n)$ run time
  Note that $O(1)$ node-average awake time is w.h.p.
- [Ghaffari & Portmann, SPAA 2022] also solves maximal matching awake efficiently

Notes

- The given solution sacrificed run time to save awake time
  $O(\log^{3.41})$ run time
- However, it's possible to avoid doing so [Ghaffari & Portmann, SPAA 2022]
  $O(\log n)$ run time
  Note that $O(1)$ node-average awake time is w.h.p.
- [Ghaffari & Portmann, SPAA 2022] also solves maximal matching awake efficiently

Outline

Section Outline

Plan for Section

- [Barenboim & Maimon, DISC 2021]: Identified structure on a global scale.
  Leveraged it to solve spanning tree.
  Also devised awake-efficient solution for class of problems called $O-LOCAL$
- **Focus of section** [Augustine, Moses Jr., & Pandurangan, SIROCCO 2024]: Tweaked structure so that minimum spanning tree could be solved. Additionally, run time improved
- **Big Idea:** Break up solution into smaller chunks of time where you can guarantee certain nodes are not awake at the same time
- **Why care:** Global problem, but $O(\log n)$ awake time. Also, the structure observed and built has applications to various other problems, both global and local

Section Outline

Plan for Section

- [Barenboim & Maimon, DISC 2021]: Identified structure on a global scale. Leveraged it to solve spanning tree. Also devised awake-efficient solution for class of problems called $O - LOCAL$
- **Focus of section** [Augustine, Moses Jr., & Pandurangan, SIROCCO 2024]: Tweaked structure so that minimum spanning tree could be solved. Additionally, run time improved
- **Big Idea:** Break up solution into smaller chunks of time where you can guarantee certain nodes are not awake at the same time
- **Why care:** Global problem, but $O(\log n)$ awake time. Also, the structure observed and built has applications to various other problems, both global and local

Section Outline

Plan for Section

- [Barenboim & Maimon, DISC 2021]: Identified structure on a global scale. Leveraged it to solve spanning tree. Also devised awake-efficient solution for class of problems called $O - LOCAL$
- **Focus of section** [Augustine, Moses Jr., & Pandurangan, SIROCCO 2024]: Tweaked structure so that minimum spanning tree could be solved. Additionally, run time improved
- **Big Idea:** Break up solution into smaller chunks of time where you can guarantee certain nodes are not awake at the same time
- **Why care:** Global problem, but $O(\log n)$ awake time. Also, the structure observed and built has applications to various other problems, both global and local

Section Outline

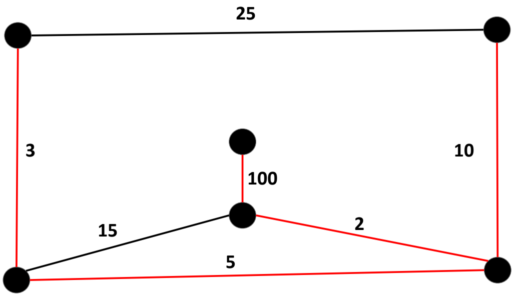
Plan for Section

- [Barenboim & Maimon, DISC 2021]: Identified structure on a global scale.
  Leveraged it to solve spanning tree.
  Also devised awake-efficient solution for class of problems called $O - LOCAL$
- **Focus of section** [Augustine, Moses Jr., & Pandurangan, SIROCCO 2024]: Tweaked structure so that minimum spanning tree could be solved. Additionally, run time improved
- **Big Idea:** Break up solution into smaller chunks of time where you can guarantee certain nodes are not awake at the same time
- **Why care:** Global problem, but $O(\log n)$ awake time. Also, the structure observed and built has applications to various other problems, both global and local

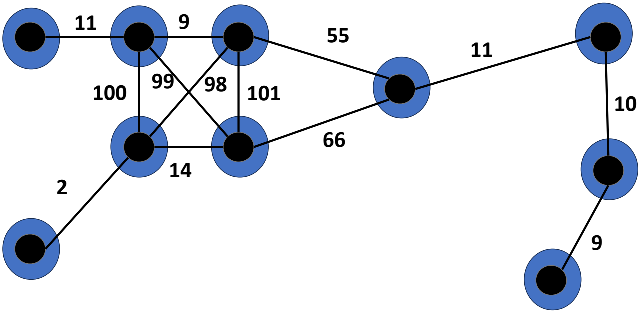
Minimum Spanning Tree (MST) - Problem Definition

Minimum Spanning Tree (MST)

For a graph $G(V,E)$ where each edge $e \in E$ has associated weight, the minimum spanning tree of G is a connected subgraph of G that connects all nodes of V, has no cycles, and has the minimum sum of edge weights amongst all possible subgraphs satisfying the previous two conditions.

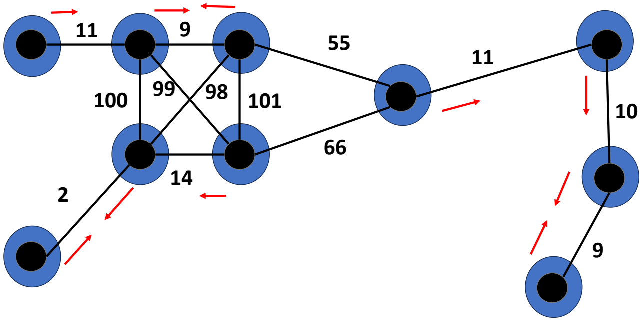


Identifying Structure



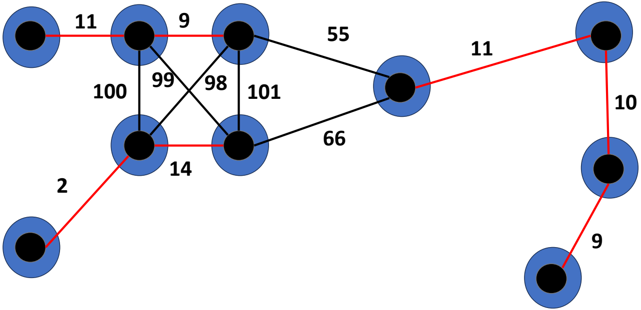
- Consider usual approach: GHS style merging of clusters
- Clusters (initially each node) merge using MOEs into larger clusters
- Each phase, constant fraction of clusters merge. Totally $O(\log n)$ phases

Identifying Structure



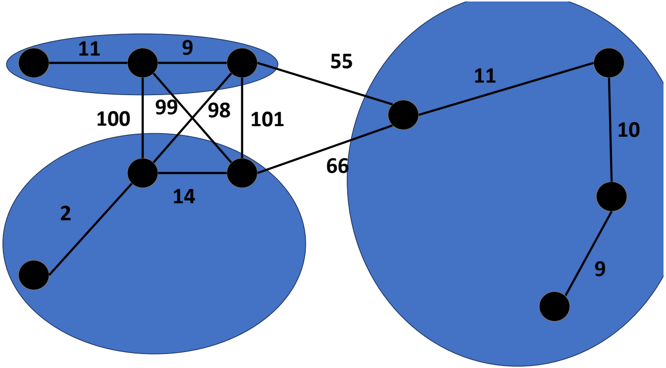
- Consider usual approach: GHS style merging of clusters
- Clusters (initially each node) merge using MOEs into larger clusters
- Each phase, constant fraction of clusters merge. Totally $O(\log n)$ phases

Identifying Structure



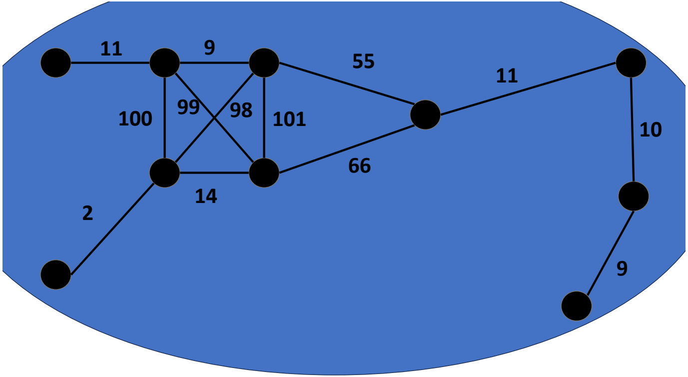
- Consider usual approach: GHS style merging of clusters
- Clusters (initially each node) merge using MOEs into larger clusters
- Each phase, constant fraction of clusters merge. Totally $O(\log n)$ phases

Identifying Structure



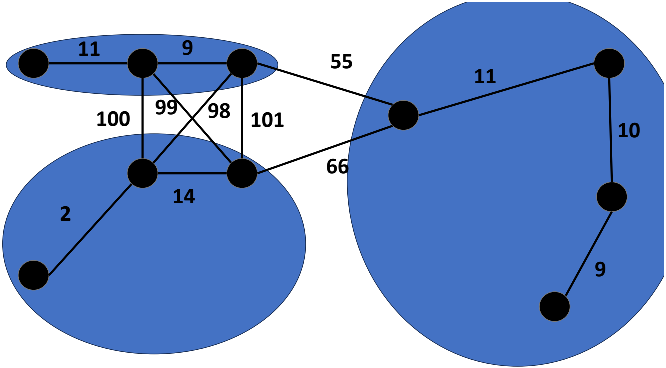
- Consider usual approach: GHS style merging of clusters
- Clusters (initially each node) merge using MOEs into larger clusters
- Each phase, constant fraction of clusters merge. Totally $O(\log n)$ phases

Identifying Structure



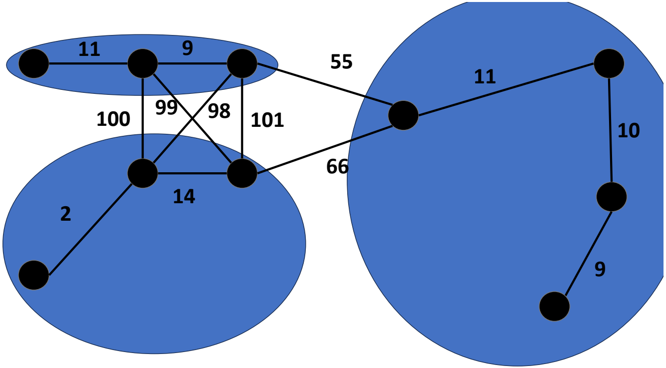
- Consider usual approach: GHS style merging of clusters
- Clusters (initially each node) merge using MOEs into larger clusters
- Each phase, constant fraction of clusters merge. Totally $O(\log n)$ phases

How To Leverage this Structure?



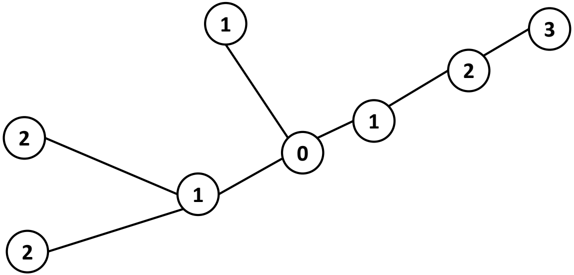
- **Notice:** Not all nodes in a cluster need to be awake at all times in a phase
- Communication req.: aggregation & info spread within cluster, adjacent cluster msgs
- **Labeled Distance Tree (LDT)** - rooted tree, each node knows its distance to root
- **Transmission schedule $(2n+1)$** - $2n+1$ rounds, each node in LDT awake $O(1)$ times

How To Leverage this Structure?



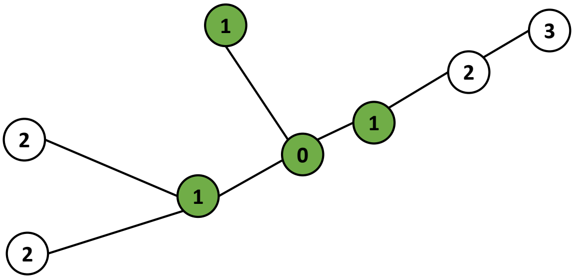
- **Notice:** Not all nodes in a cluster need to be awake at all times in a phase
- Communication req.: aggregation & info spread within cluster, adjacent cluster msgs
- **Labeled Distance Tree (LDT)** - rooted tree, each node knows its distance to root
- **Transmission schedule** $(2n+1)$ - $2n+1$ rounds, each node in LDT awake $O(1)$ times

How To Leverage this Structure?



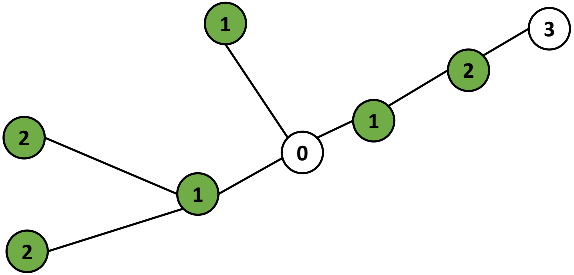
- **Notice:** Not all nodes in a cluster need to be awake at all times in a phase
- Communication req.: aggregation & info spread within cluster, adjacent cluster msgs
- **Labeled Distance Tree (LDT)** - rooted tree, each node knows its distance to root
- **Transmission schedule $(2n+1)$** - $2n+1$ rounds, each node in LDT awake $O(1)$ times

How To Leverage this Structure?



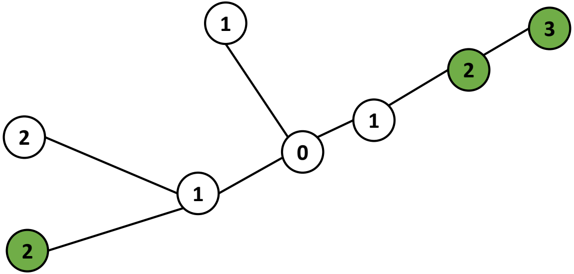
- **Notice:** Not all nodes in a cluster need to be awake at all times in a phase
- Communication req.: aggregation & info spread within cluster, adjacent cluster msgs
- **Labeled Distance Tree (LDT)** - rooted tree, each node knows its distance to root
- **Transmission schedule $(2n+1)$** - $2n+1$ rounds, each node in LDT awake $O(1)$ times

How To Leverage this Structure?



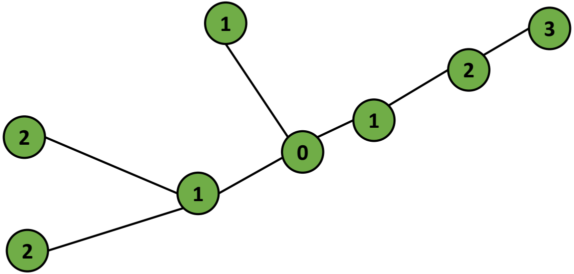
- **Notice:** Not all nodes in a cluster need to be awake at all times in a phase
- Communication req.: aggregation & info spread within cluster, adjacent cluster msgs
- **Labeled Distance Tree (LDT)** - rooted tree, each node knows its distance to root
- **Transmission schedule $(2n+1)$** - $2n+1$ rounds, each node in LDT awake $O(1)$ times

How To Leverage this Structure?



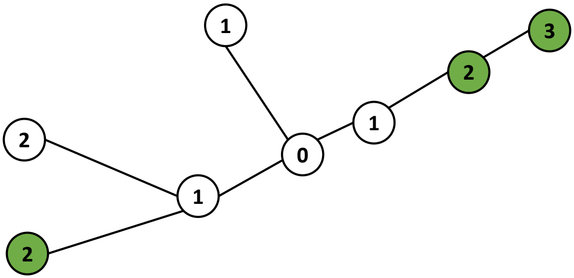
- **Notice:** Not all nodes in a cluster need to be awake at all times in a phase
- Communication req.: aggregation & info spread within cluster, adjacent cluster msgs
- **Labeled Distance Tree (LDT)** - rooted tree, each node knows its distance to root
- **Transmission schedule $(2n+1)$** - $2n+1$ rounds, each node in LDT awake $O(1)$ times

How To Leverage this Structure?



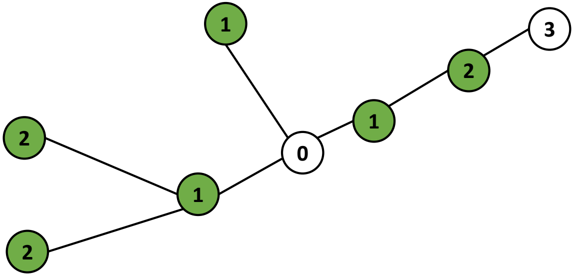
- **Notice:** Not all nodes in a cluster need to be awake at all times in a phase
- Communication req.: aggregation & info spread within cluster, adjacent cluster msgs
- **Labeled Distance Tree (LDT)** - rooted tree, each node knows its distance to root
- **Transmission schedule $(2n+1)$** - $2n+1$ rounds, each node in LDT awake $O(1)$ times

How To Leverage this Structure?



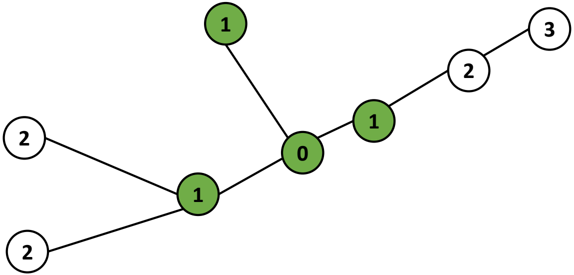
- **Notice:** Not all nodes in a cluster need to be awake at all times in a phase
- Communication req.: aggregation & info spread within cluster, adjacent cluster msgs
- **Labeled Distance Tree (LDT)** - rooted tree, each node knows its distance to root
- **Transmission schedule ($2n+1$)** - $2n+1$ rounds, each node in LDT awake $O(1)$ times

How To Leverage this Structure?



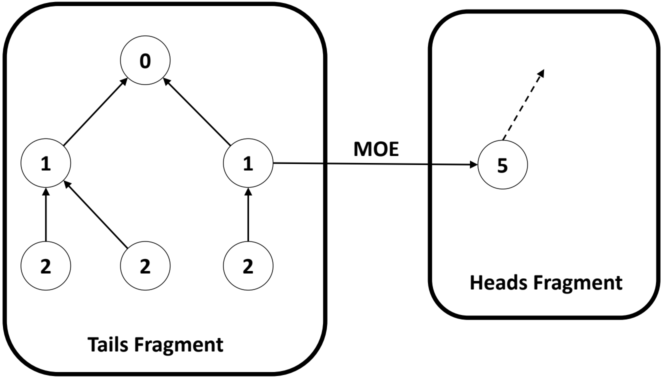
- **Notice:** Not all nodes in a cluster need to be awake at all times in a phase
- Communication req.: aggregation & info spread within cluster, adjacent cluster msgs
- **Labeled Distance Tree (LDT)** - rooted tree, each node knows its distance to root
- **Transmission schedule ($2n + 1$)** - $2n + 1$ rounds, each node in LDT awake $O(1)$ times

How To Leverage this Structure?



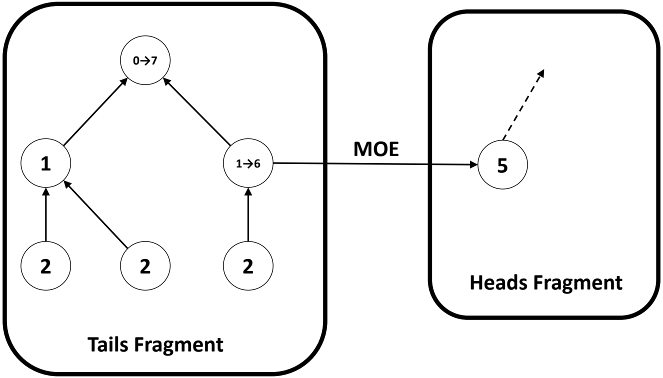
- **Notice:** Not all nodes in a cluster need to be awake at all times in a phase
- Communication req.: aggregation & info spread within cluster, adjacent cluster msgs
- **Labeled Distance Tree (LDT)** - rooted tree, each node knows its distance to root
- **Transmission schedule ($2n + 1$)** - $2n + 1$ rounds, each node in LDT awake $O(1)$ times

Randomized Algorithm



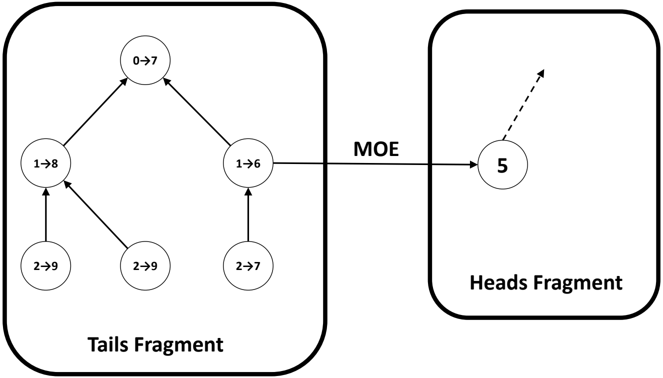
- So in each phase of GHS, merge LDTs (fragments/clusters). **Goal:** only one LDT left.
- In each phase: (i) Each LDT finds MOE (ii) Merge LDTs along MOEs.

Randomized Algorithm

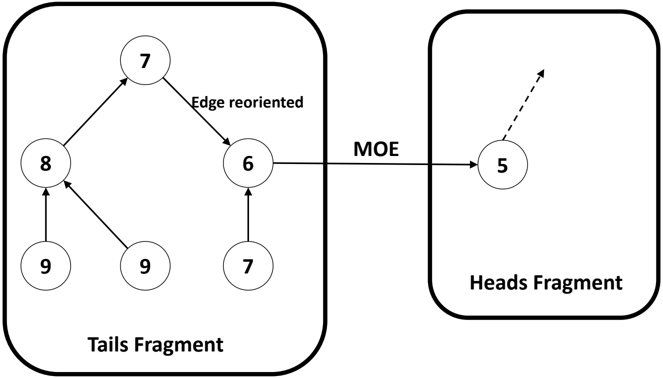


- So in each phase of GHS, merge LDTs (fragments/clusters). **Goal:** only one LDT left.
- In each phase: (i) Each LDT finds MOE (ii) Merge LDTs along MOEs.

Randomized Algorithm

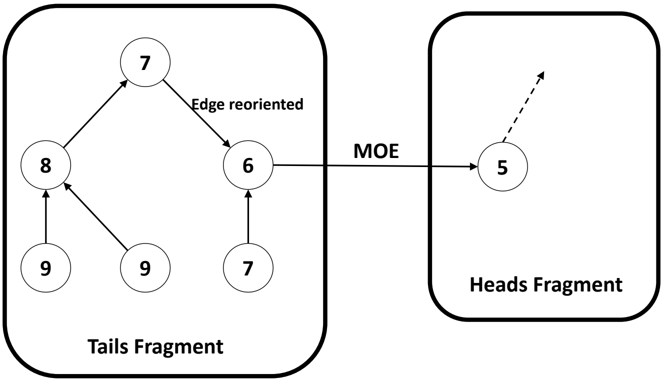


- So in each phase of GHS, merge LDTs (fragments/clusters). **Goal:** only one LDT left.
- In each phase: (i) Each LDT finds MOE (ii) Merge LDTs along MOEs.



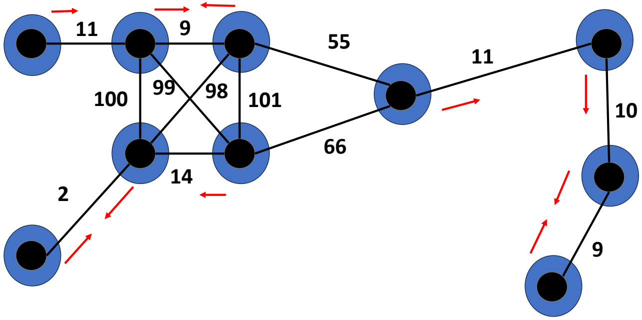
- So in each phase of GHS, merge LDTs (fragments/clusters). **Goal:** only one LDT left.
- In each phase: (i) Each LDT finds MOE (ii) Merge LDTs along MOEs.

Randomized Algorithm



- So in each phase of GHS, merge LDTs (fragments/clusters). **Goal:** only one LDT left.
- In each phase: (i) Each LDT finds MOE (ii) Merge LDTs along MOEs.
- Where could a problem occur?

Randomized Algorithm



- So in each phase of GHS, merge LDTs (fragments/clusters). **Goal:** only one LDT left.
- In each phase: (i) Each LDT finds MOE (ii) Merge LDTs along MOEs.
- Where could a problem occur?
- **Hint:** we want $O(1)$ awake time per node per phase.

Randomized Algorithm

- **Problem:** "Long lines" of LDTs trying to merge.
- **Solution:** Each LDT flips a coin. "Valid" MOEs are from Tails to Heads. Ignore others.
- Through analysis, we show that a constant fraction of LDTs are merged in each phase.
- So $O(\log n)$ phases.
- Each phase, $O(1)$ awake complexity per node and $O(n)$ run time.
- **Result:** MST constructed w.h.p. in $O(\log n)$ awake complexity and $O(n \log n)$ round complexity.

Randomized Algorithm

- **Problem:** "Long lines" of LDTs trying to merge.
- **Solution:** Each LDT flips a coin. "Valid" MOEs are from Tails to Heads. Ignore others.
- Through analysis, we show that a constant fraction of LDTs are merged in each phase.
- So $O(\log n)$ phases.
- Each phase, $O(1)$ awake complexity per node and $O(n)$ run time.
- **Result:** MST constructed w.h.p. in $O(\log n)$ awake complexity and $O(n \log n)$ round complexity.

Randomized Algorithm

- **Problem:** "Long lines" of LDTs trying to merge.
- **Solution:** Each LDT flips a coin. "Valid" MOEs are from Tails to Heads. Ignore others.
- Through analysis, we show that a constant fraction of LDTs are merged in each phase.
- So $O(\log n)$ phases.
- Each phase, $O(1)$ awake complexity per node and $O(n)$ run time.
- **Result:** MST constructed w.h.p. in $O(\log n)$ awake complexity and $O(n \log n)$ round complexity.

Randomized Algorithm

- **Problem:** "Long lines" of LDTs trying to merge.
- **Solution:** Each LDT flips a coin. "Valid" MOEs are from Tails to Heads. Ignore others.
- Through analysis, we show that a constant fraction of LDTs are merged in each phase.
- So $O(\log n)$ phases.
- Each phase, $O(1)$ awake complexity per node and $O(n)$ run time.
- **Result:** MST constructed w.h.p. in $O(\log n)$ awake complexity and $O(n \log n)$ round complexity.

Randomized Algorithm

- **Problem:** "Long lines" of LDTs trying to merge.
- **Solution:** Each LDT flips a coin. "Valid" MOEs are from Tails to Heads. Ignore others.
- Through analysis, we show that a constant fraction of LDTs are merged in each phase.
- So $O(\log n)$ phases.
- Each phase, $O(1)$ awake complexity per node and $O(n)$ run time.
- **Result:** MST constructed w.h.p. in $O(\log n)$ awake complexity and $O(n \log n)$ round complexity.

Randomized Algorithm

- **Problem:** "Long lines" of LDTs trying to merge.
- **Solution:** Each LDT flips a coin. "Valid" MOEs are from Tails to Heads. Ignore others.
- Through analysis, we show that a constant fraction of LDTs are merged in each phase.
- So $O(\log n)$ phases.
- Each phase, $O(1)$ awake complexity per node and $O(n)$ run time.
- **Result:** MST constructed w.h.p. in $O(\log n)$ awake complexity and $O(n \log n)$ round complexity.

Notes

- O-LOCAL - a wide variety of problems solved in $O(\log \Delta + \log^* n)$ awake time [Barenboim & Maimon, DISC 2021]
- [Balliu, Fraigniaud, Olivetti, & Rabie, PODC 2025] - O-LOCAL problems solved in $O(\sqrt{\log n} \log^* n)$ awake time
- LDT - used to help solve global problem SSSP in [Ghaffari & Trygub, PODC 2024]
- LDT - used to help solve local problem MIS in next section

Notes

- O-LOCAL - a wide variety of problems solved in $O(\log \Delta + \log^* n)$ awake time [Barenboim & Maimon, DISC 2021]
- [Balliu, Fraigniaud, Olivetti, & Rabie, PODC 2025] - O-LOCAL problems solved in $O(\sqrt{\log n} \log^* n)$ awake time
- LDT - used to help solve global problem SSSP in [Ghaffari & Trygub, PODC 2024]
- LDT - used to help solve local problem MIS in next section

Notes

- O-LOCAL - a wide variety of problems solved in $O(\log \Delta + \log^* n)$ awake time [Barenboim & Maimon, DISC 2021]
- [Balliu, Fraigniaud, Olivetti, & Rabie, PODC 2025] - O-LOCAL problems solved in $O(\sqrt{\log n} \log^* n)$ awake time
- LDT - used to help solve global problem SSSP in [Ghaffari & Trygub, PODC 2024]
- LDT - used to help solve local problem MIS in next section

Notes

- O-LOCAL - a wide variety of problems solved in $O(\log \Delta + \log^* n)$ awake time [Barenboim & Maimon, DISC 2021]
- [Balliu, Fraigniaud, Olivetti, & Rabie, PODC 2025] - O-LOCAL problems solved in $O(\sqrt{\log n} \log^* n)$ awake time
- LDT - used to help solve global problem SSSP in [Ghaffari & Trygub, PODC 2024]
- LDT - used to help solve local problem MIS in next section

Outline

Section Outline

Plan for Section

- Introduce solution to Maximal Independent Set (MIS) from [Dufoulon, Moses Jr., & Pandurangan, PODC 2023]
- **Big Idea:** Instead of reducing awake time through only analysis, what if we identify a structure that speeds up MIS computation, and leverage algorithmic techniques & analysis to constantly get that structure. Also, target CONGEST
- **Why care:** Lower bound [Kuhn, Moscibroda, & Wattenhofer, JACM 2016] - $\Omega(\min\{\frac{\log \Delta}{\log\log \Delta}, \sqrt{\frac{\log n}{\log\log n}}\})$.
  For high degree graphs, run time $= \Omega(\sqrt{\frac{\log n}{\log\log n}})$
- [Chatterjee, Gmyr, & Pandurangan, PODC 2020, Ghaffari & Portmann, SPAA 2022]: $O(1)$ node-average awake time
  But! - $O(\log n)$ awake time (not average, but worst case)
- O-LOCAL: only LOCAL solutions, and when $\Delta = poly(n)$, $O(\log n)$ awake time
- [Dufoulon, Moses Jr., & Pandurangan, PODC 2023] - $O(\log\log n)$ awake time (asymptotically better than lower bound for high degree graphs)

Section Outline

Plan for Section

- Introduce solution to Maximal Independent Set (MIS) from [Dufoulon, Moses Jr., & Pandurangan, PODC 2023]
- **Big Idea:** Instead of reducing awake time through only analysis, what if we identify a structure that speeds up MIS computation, and leverage algorithmic techniques & analysis to constantly get that structure. Also, target CONGEST
- **Why care:** Lower bound [Kuhn, Moscibroda, & Wattenhofer, JACM 2016] - $\Omega(\min\{\frac{\log \Delta}{\log\log \Delta}, \sqrt{\frac{\log n}{\log\log n}}\})$.

  For high degree graphs, run time $= \Omega(\sqrt{\frac{\log n}{\log\log n}})$
- [Chatterjee, Gmyr, & Pandurangan, PODC 2020, Ghaffari & Portmann, SPAA 2022]: $O(1)$ node-average awake time

  But! - $O(\log n)$ awake time (not average, but worst case)
- O-LOCAL: only LOCAL solutions, and when $\Delta = poly(n)$, $O(\log n)$ awake time
- [Dufoulon, Moses Jr., & Pandurangan, PODC 2023] - $O(\log\log n)$ awake time (asymptotically better than lower bound for high degree graphs)

Section Outline

Plan for Section

- Introduce solution to Maximal Independent Set (MIS) from [Dufoulon, Moses Jr., & Pandurangan, PODC 2023]
- **Big Idea:** Instead of reducing awake time through only analysis, what if we identify a structure that speeds up MIS computation, and leverage algorithmic techniques & analysis to constantly get that structure. Also, target CONGEST
- **Why care:** Lower bound [Kuhn, Moscibroda, & Wattenhofer, JACM 2016] - $\Omega(\min\{\frac{\log \Delta}{\log\log \Delta}, \sqrt{\frac{\log n}{\log\log n}}\})$.
  For high degree graphs, run time $= \Omega(\sqrt{\frac{\log n}{\log\log n}})$
- [Chatterjee, Gmyr, & Pandurangan, PODC 2020, Ghaffari & Portmann, SPAA 2022]: $O(1)$ node-average awake time
  But! - $O(\log n)$ awake time (not average, but worst case)
- O-LOCAL: only LOCAL solutions, and when $\Delta = poly(n)$, $O(\log n)$ awake time
- [Dufoulon, Moses Jr., & Pandurangan, PODC 2023] - $O(\log\log n)$ awake time (asymptotically better than lower bound for high degree graphs)

Section Outline

Plan for Section

- Introduce solution to Maximal Independent Set (MIS) from [Dufoulon, Moses Jr., & Pandurangan, PODC 2023]
- **Big Idea:** Instead of reducing awake time through only analysis, what if we identify a structure that speeds up MIS computation, and leverage algorithmic techniques & analysis to constantly get that structure. Also, target CONGEST
- **Why care:** Lower bound [Kuhn, Moscibroda, & Wattenhofer, JACM 2016] - $\Omega(\min\{\frac{\log \Delta}{\log \log \Delta}, \sqrt{\frac{\log n}{\log \log n}}\})$.
  For high degree graphs, run time $= \Omega(\sqrt{\frac{\log n}{\log \log n}})$
- [Chatterjee, Gmyr, & Pandurangan, PODC 2020, Ghaffari & Portmann, SPAA 2022]: $O(1)$ node-average awake time
  But! - $O(\log n)$ awake time (not average, but worst case)
- O-LOCAL: only LOCAL solutions, and when $\Delta = poly(n)$, $O(\log n)$ awake time
- [Dufoulon, Moses Jr., & Pandurangan, PODC 2023] - $O(\log \log n)$ awake time (asymptotically better than lower bound for high degree graphs)

The Big Ideas

- Structure leveraged: LDT, but of size $O(\log n)$
- Analysis technique leveraged: Konrad's lemma [Konrad, arXiv 2018]
- Second communication structure leveraged: Virtual binary tree (similar to that in [Barenboim & Maimon, DISC 2021])
- Plan: Explain each technique/idea. Then put them together

The Big Ideas

- Structure leveraged: LDT, but of size $O(\log n)$
- Analysis technique leveraged: Konrad's lemma [Konrad, arXiv 2018]
- Second communication structure leveraged: Virtual binary tree (similar to that in [Barenboim & Maimon, DISC 2021])
- Plan: Explain each technique/idea. Then put them together

The Big Ideas

- Structure leveraged: LDT, but of size $O(\log n)$
- Analysis technique leveraged: Konrad's lemma [Konrad, arXiv 2018]
- Second communication structure leveraged: Virtual binary tree (similar to that in [Barenboim & Maimon, DISC 2021])
- Plan: Explain each technique/idea. Then put them together

The Big Ideas

- Structure leveraged: LDT, but of size $O(\log n)$
- Analysis technique leveraged: Konrad's lemma [Konrad, arXiv 2018]
- Second communication structure leveraged: Virtual binary tree (similar to that in [Barenboim & Maimon, DISC 2021])
- Plan: Explain each technique/idea. Then put them together

LDT of Size $O(\log n)$

- **Idea:** Shatter graph into disjoint non-neighboring sets of nodes
- Somehow ensure each set has at most $O(\log n)$ nodes
- Form an LDT on each set of nodes in $O(\log \log n)$ awake time per node
- Computing MIS on each LDT takes $O(\log \log n)$ awake time per node:
  (i) Generate IDs for all nodes and let them know in $O(\log \log n)$ awake time
  (ii) Use virtual binary tree (see next next slide) to get LFMIS in $O(\log \log n)$ awake time
- **Issue:** How do we "somehow ensure" size?

LDT of Size $O(\log n)$

- **Idea:** Shatter graph into disjoint non-neighboring sets of nodes
- Somehow ensure each set has at most $O(\log n)$ nodes
- Form an LDT on each set of nodes in $O(\log \log n)$ awake time per node
- Computing MIS on each LDT takes $O(\log \log n)$ awake time per node:
  (i) Generate IDs for all nodes and let them know in $O(\log \log n)$ awake time
  (ii) Use virtual binary tree (see next next slide) to get LFMIS in $O(\log \log n)$ awake time
- **Issue:** How do we "somehow ensure" size?

LDT of Size $O(\log n)$

- **Idea:** Shatter graph into disjoint non-neighboring sets of nodes
- Somehow ensure each set has at most $O(\log n)$ nodes
- Form an LDT on each set of nodes in $O(\log \log n)$ awake time per node
- Computing MIS on each LDT takes $O(\log \log n)$ awake time per node:
  (i) Generate IDs for all nodes and let them know in $O(\log \log n)$ awake time
  (ii) Use virtual binary tree (see next next slide) to get LFMIS in $O(\log \log n)$ awake time
- **Issue:** How do we "somehow ensure" size?

LDT of Size $O(\log n)$

- **Idea:** Shatter graph into disjoint non-neighboring sets of nodes
- Somehow ensure each set has at most $O(\log n)$ nodes
- Form an LDT on each set of nodes in $O(\log \log n)$ awake time per node
- Computing MIS on each LDT takes $O(\log \log n)$ awake time per node:
  (i) Generate IDs for all nodes and let them know in $O(\log \log n)$ awake time
  (ii) Use virtual binary tree (see next next slide) to get LFMIS in $O(\log \log n)$ awake time
- **Issue:** How do we "somehow ensure" size?

LDT of Size $O(\log n)$

- **Idea:** Shatter graph into disjoint non-neighboring sets of nodes
- Somehow ensure each set has at most $O(\log n)$ nodes
- Form an LDT on each set of nodes in $O(\log \log n)$ awake time per node
- Computing MIS on each LDT takes $O(\log \log n)$ awake time per node:
  (i) Generate IDs for all nodes and let them know in $O(\log \log n)$ awake time
  (ii) Use virtual binary tree (see next next slide) to get LFMIS in $O(\log \log n)$ awake time
- **Issue:** How do we "somehow ensure" size?

Konrad's Lemma

- **Idea:** Use a Lexicographically First MIS (LFMIS) (process nodes in inc. order of IDs)
- **Lemma:** Let t, t' be two integers such that $1 \leq t < t' \leq n$. Let V_t denote the (set of the) first t nodes, $V_{t'}$ the (set of the) first t' nodes and M_t the LFMIS over $G[V_t]$. Then, for any $\epsilon > 0$, $G[V_{t'} \setminus N(M_t)]$ has maximum degree at most $\frac{t'}{t}\ln(n/\epsilon)$ with probability at least $1 - \epsilon$
- Process first $O(\log n)$ nodes (based on ID) as a **batch**. Graphs have degree $O(\log n)$.
- Dividing nodes into $O(\log n)$ **buckets** u.a.r. guarantees any connected subgraph in each bucket has $O(\log n)$ nodes w.h.p.
- Run LDT on each of these connected subgraphs to get MIS, one bucket at a time
- Then once all buckets processed, process next batch of $O(\log^2 n)$ nodes. By Konrad's lemma, graphs have degree $O(\log n)$. Rinse & repeat
- Two issues (but really same issue): MIS from one bucket affects nodes in next. MIS from batch of smaller ID nodes affects MIS next one
- Essentially, how to communicate outcomes from smaller buckets/batch to larger buckets/batch?

Konrad's Lemma

- **Idea:** Use a Lexicographically First MIS (LFMIS) (process nodes in inc. order of IDs)
- **Lemma:** Let t, t' be two integers such that $1 \leq t < t' \leq n$. Let V_t denote the (set of the) first t nodes, $V_{t'}$ the (set of the) first t' nodes and M_t the LFMIS over $G[V_t]$. Then, for any $\epsilon > 0$, $G[V_{t'} \setminus N(M_t)]$ has maximum degree at most $\frac{t'}{t} \ln(n/\epsilon)$ with probability at least $1 - \epsilon$
- Process first $O(\log n)$ nodes (based on ID) as a **batch**. Graphs have degree $O(\log n)$.
- Dividing nodes into $O(\log n)$ **buckets** u.a.r. guarantees any connected subgraph in each bucket has $O(\log n)$ nodes w.h.p.
- Run LDT on each of these connected subgraphs to get MIS, one bucket at a time
- Then once all buckets processed, process next batch of $O(\log^2 n)$ nodes. By Konrad's lemma, graphs have degree $O(\log n)$. Rinse & repeat
- Two issues (but really same issue): MIS from one bucket affects nodes in next. MIS from batch of smaller ID nodes affects MIS next one
- Essentially, how to communicate outcomes from smaller buckets/batch to larger buckets/batch?

Konrad's Lemma

- **Idea:** Use a Lexicographically First MIS (LFMIS) (process nodes in inc. order of IDs)
- **Lemma:** Let t, t' be two integers such that $1 \leq t < t' \leq n$. Let V_t denote the (set of the) first t nodes, $V_{t'}$ the (set of the) first t' nodes and M_t the LFMIS over $G[V_t]$. Then, for any $\epsilon > 0$, $G[V_{t'} \setminus N(M_t)]$ has maximum degree at most $\frac{t'}{t} \ln(n/\epsilon)$ with probability at least $1 - \epsilon$
- Process first $O(\log n)$ nodes (based on ID) as a **batch**. Graphs have degree $O(\log n)$.
- Dividing nodes into $O(\log n)$ **buckets** u.a.r. guarantees any connected subgraph in each bucket has $O(\log n)$ nodes w.h.p.
- Run LDT on each of these connected subgraphs to get MIS, one bucket at a time
- Then once all buckets processed, process next batch of $O(\log^2 n)$ nodes. By Konrad's lemma, graphs have degree $O(\log n)$. Rinse & repeat
- Two issues (but really same issue): MIS from one bucket affects nodes in next. MIS from batch of smaller ID nodes affects MIS next one
- Essentially, how to communicate outcomes from smaller buckets/batch to larger buckets/batch?

Konrad's Lemma

- **Idea:** Use a Lexicographically First MIS (LFMIS) (process nodes in inc. order of IDs)
- **Lemma:** Let t, t' be two integers such that $1 \leq t < t' \leq n$. Let V_t denote the (set of the) first t nodes, $V_{t'}$ the (set of the) first t' nodes and M_t the LFMIS over $G[V_t]$. Then, for any $\epsilon > 0$, $G[V_{t'} \setminus N(M_t)]$ has maximum degree at most $\frac{t'}{t} \ln(n/\epsilon)$ with probability at least $1 - \epsilon$
- Process first $O(\log n)$ nodes (based on ID) as a **batch**. Graphs have degree $O(\log n)$.
- Dividing nodes into $O(\log n)$ **buckets** u.a.r. guarantees any connected subgraph in each bucket has $O(\log n)$ nodes w.h.p.
- Run LDT on each of these connected subgraphs to get MIS, one bucket at a time
- Then once all buckets processed, process next batch of $O(\log^2 n)$ nodes. By Konrad's lemma, graphs have degree $O(\log n)$. Rinse & repeat
- Two issues (but really same issue): MIS from one bucket affects nodes in next. MIS from batch of smaller ID nodes affects MIS next one
- Essentially, how to communicate outcomes from smaller buckets/batch to larger buckets/batch?

Konrad's Lemma

- **Idea:** Use a Lexicographically First MIS (LFMIS) (process nodes in inc. order of IDs)
- **Lemma:** Let t, t' be two integers such that $1 \leq t < t' \leq n$. Let V_t denote the (set of the) first t nodes, $V_{t'}$ the (set of the) first t' nodes and M_t the LFMIS over $G[V_t]$. Then, for any $\epsilon > 0$, $G[V_{t'} \setminus N(M_t)]$ has maximum degree at most $\frac{t'}{t} \ln(n/\epsilon)$ with probability at least $1 - \epsilon$
- Process first $O(\log n)$ nodes (based on ID) as a **batch**. Graphs have degree $O(\log n)$.
- Dividing nodes into $O(\log n)$ **buckets** u.a.r. guarantees any connected subgraph in each bucket has $O(\log n)$ nodes w.h.p.
- Run LDT on each of these connected subgraphs to get MIS, one bucket at a time
- Then once all buckets processed, process next batch of $O(\log^2 n)$ nodes. By Konrad's lemma, graphs have degree $O(\log n)$. Rinse & repeat
- Two issues (but really same issue): MIS from one bucket affects nodes in next. MIS from batch of smaller ID nodes affects MIS next one
- Essentially, how to communicate outcomes from smaller buckets/batch to larger buckets/batch?

Konrad's Lemma

- **Idea:** Use a Lexicographically First MIS (LFMIS) (process nodes in inc. order of IDs)
- **Lemma:** Let t, t' be two integers such that $1 \leq t < t' \leq n$. Let V_t denote the (set of the) first t nodes, $V_{t'}$ the (set of the) first t' nodes and M_t the LFMIS over $G[V_t]$. Then, for any $\epsilon > 0$, $G[V_{t'} \setminus N(M_t)]$ has maximum degree at most $\frac{t'}{t} \ln(n/\epsilon)$ with probability at least $1 - \epsilon$
- Process first $O(\log n)$ nodes (based on ID) as a **batch**. Graphs have degree $O(\log n)$.
- Dividing nodes into $O(\log n)$ **buckets** u.a.r. guarantees any connected subgraph in each bucket has $O(\log n)$ nodes w.h.p.
- Run LDT on each of these connected subgraphs to get MIS, one bucket at a time
- Then once all buckets processed, process next batch of $O(\log^2 n)$ nodes. By Konrad's lemma, graphs have degree $O(\log n)$. Rinse & repeat
- Two issues (but really same issue): MIS from one bucket affects nodes in next. MIS from batch of smaller ID nodes affects MIS next one
- Essentially, how to communicate outcomes from smaller buckets/batch to larger buckets/batch?

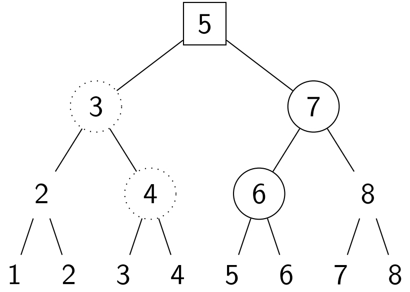
Konrad's Lemma

- **Idea:** Use a Lexicographically First MIS (LFMIS) (process nodes in inc. order of IDs)
- **Lemma:** Let t, t' be two integers such that $1 \leq t < t' \leq n$. Let V_t denote the (set of the) first t nodes, $V_{t'}$ the (set of the) first t' nodes and M_t the LFMIS over $G[V_t]$. Then, for any $\epsilon > 0$, $G[V_{t'} \setminus N(M_t)]$ has maximum degree at most $\frac{t'}{t} \ln(n/\epsilon)$ with probability at least $1 - \epsilon$
- Process first $O(\log n)$ nodes (based on ID) as a **batch**. Graphs have degree $O(\log n)$.
- Dividing nodes into $O(\log n)$ **buckets** u.a.r. guarantees any connected subgraph in each bucket has $O(\log n)$ nodes w.h.p.
- Run LDT on each of these connected subgraphs to get MIS, one bucket at a time
- Then once all buckets processed, process next batch of $O(\log^2 n)$ nodes. By Konrad's lemma, graphs have degree $O(\log n)$. Rinse & repeat
- Two issues (but really same issue): MIS from one bucket affects nodes in next. MIS from batch of smaller ID nodes affects MIS next one
- Essentially, how to communicate outcomes from smaller buckets/batch to larger buckets/batch?

Konrad's Lemma

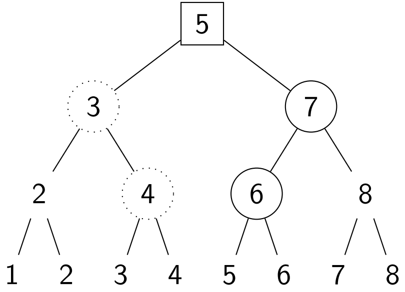
- **Idea:** Use a Lexicographically First MIS (LFMIS) (process nodes in inc. order of IDs)
- **Lemma:** Let t, t' be two integers such that $1 \leq t < t' \leq n$. Let V_t denote the (set of the) first t nodes, $V_{t'}$ the (set of the) first t' nodes and M_t the LFMIS over $G[V_t]$. Then, for any $\epsilon > 0$, $G[V_{t'} \setminus N(M_t)]$ has maximum degree at most $\frac{t'}{t} \ln(n/\epsilon)$ with probability at least $1 - \epsilon$
- Process first $O(\log n)$ nodes (based on ID) as a **batch**. Graphs have degree $O(\log n)$.
- Dividing nodes into $O(\log n)$ **buckets** u.a.r. guarantees any connected subgraph in each bucket has $O(\log n)$ nodes w.h.p.
- Run LDT on each of these connected subgraphs to get MIS, one bucket at a time
- Then once all buckets processed, process next batch of $O(\log^2 n)$ nodes. By Konrad's lemma, graphs have degree $O(\log n)$. Rinse & repeat
- Two issues (but really same issue): MIS from one bucket affects nodes in next. MIS from batch of smaller ID nodes affects MIS next one
- Essentially, how to communicate outcomes from smaller buckets/batch to larger buckets/batch?

Virtual Binary Tree



- **Idea:** Consider a binary tree where the ID of each bucket/batch is a leaf node. Systematic way to send messages from lower buckets to higher buckets in order. Ancestors of a leaf represent round numbers to be awake in a schedule the size of the tree
- **Important:** Ensure binary tree has only $O(\log n)$ nodes. Then height of tree (# of awake rounds per node) is $O(\log \log n)$

Virtual Binary Tree



- **Idea:** Consider a binary tree where the ID of each bucket/batch is a leaf node. Systematic way to send messages from lower buckets to higher buckets in order. Ancestors of a leaf represent round numbers to be awake in a schedule the size of the tree
- **Important:** Ensure binary tree has only $O(\log n)$ nodes. Then height of tree (# of awake rounds per node) is $O(\log \log n)$

Putting it all Together

- Start with the whole graph
- Nodes choose one of $O(\log n)$ batches u.a.r. - X random bits
- Nodes choose one of $O(\log n)$ buckets u.a.r. - Y random bits
- Nodes choose one of $O(\log n)$ IDs in LDT u.a.r. - Z random bits
- Each node has ID corresponding roughly to XYZ.
  Virtual binary tree takes $O(\log \log n)$ awake time to (i) solve LFMIS on LDT, (ii) communicate between buckets, (iii) communicate between batches
- LFMIS maintained. Totally $O(\log \log n)$ awake time per node

Putting it all Together

- Start with the whole graph
- Nodes choose one of $O(\log n)$ batches u.a.r. - X random bits
- Nodes choose one of $O(\log n)$ buckets u.a.r. - Y random bits
- Nodes choose one of $O(\log n)$ IDs in LDT u.a.r. - Z random bits
- Each node has ID corresponding roughly to XYZ.
  Virtual binary tree takes $O(\log \log n)$ awake time to (i) solve LFMIS on LDT, (ii) communicate between buckets, (iii) communicate between batches
- LFMIS maintained. Totally $O(\log \log n)$ awake time per node

Putting it all Together

- Start with the whole graph
- Nodes choose one of $O(\log n)$ batches u.a.r. - X random bits
- Nodes choose one of $O(\log n)$ buckets u.a.r. - Y random bits
- Nodes choose one of $O(\log n)$ IDs in LDT u.a.r. - Z random bits
- Each node has ID corresponding roughly to XYZ.
  Virtual binary tree takes $O(\log\log n)$ awake time to (i) solve LFMIS on LDT, (ii) communicate between buckets, (iii) communicate between batches
- LFMIS maintained. Totally $O(\log\log n)$ awake time per node

Putting it all Together

- Start with the whole graph
- Nodes choose one of $O(\log n)$ batches u.a.r. - X random bits
- Nodes choose one of $O(\log n)$ buckets u.a.r. - Y random bits
- Nodes choose one of $O(\log n)$ IDs in LDT u.a.r. - Z random bits
- Each node has ID corresponding roughly to XYZ.
  Virtual binary tree takes $O(\log\log n)$ awake time to (i) solve LFMIS on LDT, (ii) communicate between buckets, (iii) communicate between batches
- LFMIS maintained. Totally $O(\log\log n)$ awake time per node

Putting it all Together

- Start with the whole graph
- Nodes choose one of $O(\log n)$ batches u.a.r. - X random bits
- Nodes choose one of $O(\log n)$ buckets u.a.r. - Y random bits
- Nodes choose one of $O(\log n)$ IDs in LDT u.a.r. - Z random bits
- Each node has ID corresponding roughly to XYZ.
  Virtual binary tree takes $O(\log \log n)$ awake time to (i) solve LFMIS on LDT, (ii) communicate between buckets, (iii) communicate between batches
- LFMIS maintained. Totally $O(\log \log n)$ awake time per node

Putting it all Together

- Start with the whole graph
- Nodes choose one of $O(\log n)$ batches u.a.r. - X random bits
- Nodes choose one of $O(\log n)$ buckets u.a.r. - Y random bits
- Nodes choose one of $O(\log n)$ IDs in LDT u.a.r. - Z random bits
- Each node has ID corresponding roughly to XYZ.
  Virtual binary tree takes $O(\log \log n)$ awake time to (i) solve LFMIS on LDT, (ii) communicate between buckets, (iii) communicate between batches
- LFMIS maintained. Totally $O(\log \log n)$ awake time per node

Notes

- $O(\log \log n)$ awake time, $O((\log^7 n) \log \log n)$ run time.
  Variation: $O((\log \log n) \log^* n)$ awake time, $O((\log^3 n)(\log \log n) \log^* n)$ run time
- [Ghaffari & Portmann, PODC 2023] solve MIS in CONGEST - $O(\log \log n)$ awake time, $O(\log^2 n)$ run time.
  Variation: $O((\log \log n)^2)$ awake time, $O(\log n(\log \log n) \log^* n)$ run time

Notes

- $O(\log\log n)$ awake time, $O((\log^7 n)\log\log n)$ run time.
  Variation: $O((\log\log n)\log^* n)$ awake time, $O((\log^3 n)(\log\log n)\log^* n)$ run time
- [Ghaffari & Portmann, PODC 2023] solve MIS in CONGEST - $O(\log\log n)$ awake time, $O(\log^2 n)$ run time.
  Variation: $O((\log\log n)^2)$ awake time, $O(\log n(\log\log n)\log^* n)$ run time

Outline

Final Thoughts

- Many more papers exist, not all covered in this talk
- Ideas from wireless models can be leveraged in wired settings
- Many problems still unstudied/scope for further study
- **Personal View:** The key to solving problems in this setting is to look at existing techniques for a given problem, identify who communicates with whom, and see if various techniques play well together on that front or if you can artificially make them play nice

Final Thoughts

- Many more papers exist, not all covered in this talk
- Ideas from wireless models can be leveraged in wired settings
- Many problems still unstudied/scope for further study
- **Personal View:** The key to solving problems in this setting is to look at existing techniques for a given problem, identify who communicates with whom, and see if various techniques play well together on that front or if you can artificially make them play nice

Final Thoughts

- Many more papers exist, not all covered in this talk
- Ideas from wireless models can be leveraged in wired settings
- Many problems still unstudied/scope for further study
- **Personal View:** The key to solving problems in this setting is to look at existing techniques for a given problem, identify who communicates with whom, and see if various techniques play well together on that front or if you can artificially make them play nice

Final Thoughts

- Many more papers exist, not all covered in this talk
- Ideas from wireless models can be leveraged in wired settings
- Many problems still unstudied/scope for further study
- **Personal View:** The key to solving problems in this setting is to look at existing techniques for a given problem, identify who communicates with whom, and see if various techniques play well together on that front or if you can artificially make them play nice

References

John Augustine, William K. Moses Jr., and Gopal Pandurangan.
Awake complexity of distributed minimum spanning tree.
In *International Colloquium on Structural Information and Communication Complexity*, pages 45–63. Springer, 2024.

Alkida Balliu, Pierre Fraigniaud, Dennis Olivetti, and Mikaël Rabie.
Solving sequential greedy problems distributedly with sub-logarithmic energy cost.
In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, pages 417–427, 2025.

Leonid Barenboim and Tzalik Maimon.
Deterministic logarithmic completeness in the distributed sleeping model.
In *35th International Symposium on Distributed Computing*, 2021.

Soumyottam Chatterjee, Robert Gmyr, and Gopal Pandurangan.
Sleeping is efficient: Mis in o (1)-rounds node-averaged awake complexity.
In *Proceedings of the 39th Symposium on Principles of Distributed Computing*, pages 99–108, 2020.

Fabien Dufoulon, William K. Moses Jr., and Gopal Pandurangan.
Distributed mis in o (log log n) awake complexity.
In *Proceedings of the 2023 ACM Symposium on Principles of Distributed Computing*, pages 135–145, 2023.

References

Mohsen Ghaffari and Julian Portmann.
Average awake complexity of mis and matching.
In *Proceedings of the 34th ACM Symposium on Parallelism in Algorithms and Architectures*, pages 45–55, 2022.

Mohsen Ghaffari and Julian Portmann.
Distributed mis with low energy and time complexities.
In *Proceedings of the 2023 ACM Symposium on Principles of Distributed Computing*, pages 146–156, 2023.

Mohsen Ghaffari and Anton Trygub.
A near-optimal low-energy deterministic distributed sssp with ramifications on congestion and apsp.
In *Proceedings of the 43rd ACM Symposium on Principles of Distributed Computing*, pages 401–411, 2024.

Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer.
Local computation: Lower and upper bounds.
Journal of the ACM (JACM), 63(2):1–44, 2016.

Christian Konrad.
MIS in the congested clique model in $O(\log\log\Delta)$ rounds.
arXiv preprint arXiv:1802.07647, 2018.

References

Koji Nakano and Stephan Olariu.
Energy-efficient initialization protocols for radio networks with no collision detection.
In *Proceedings 2000 International Conference on Parallel Processing*, pages 263–270. IEEE, 2000.

The End