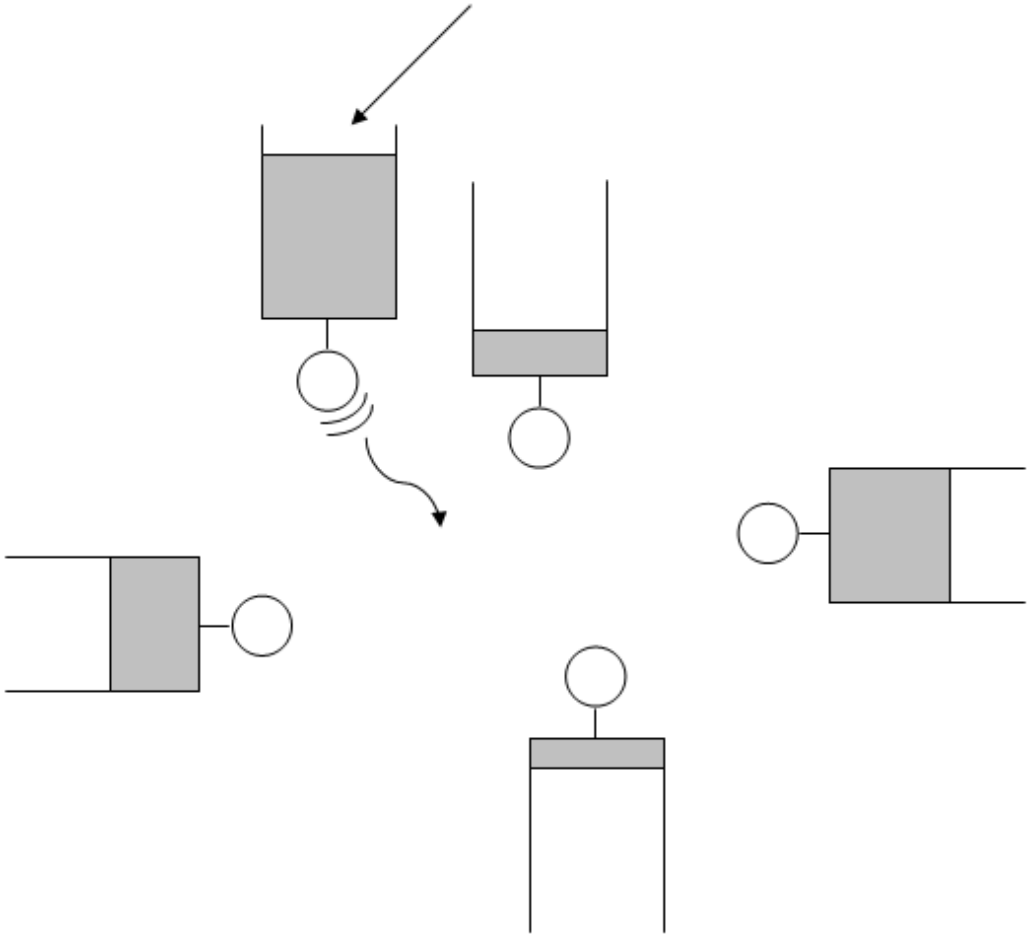
An Efficient Randomized Protocol for Contention Resolution
Network Adiabatic Theorem

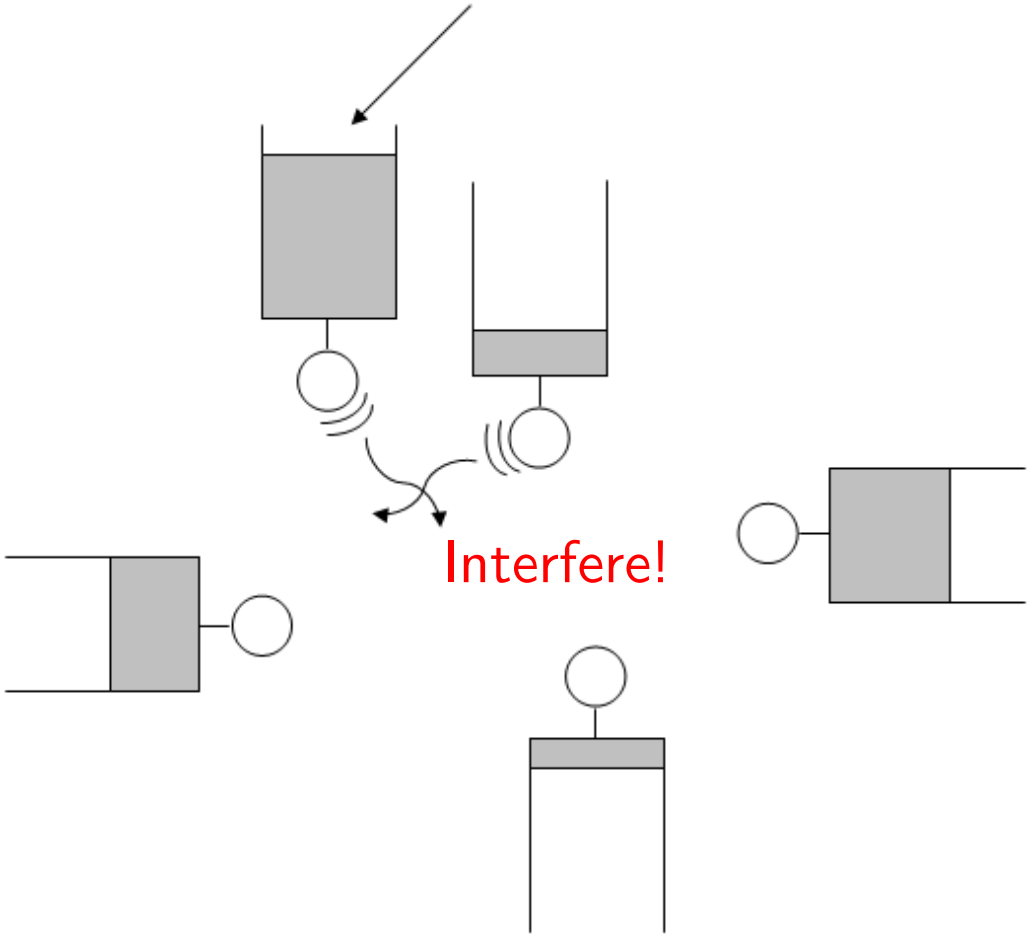
Shreevatsa Rajagopalan Devavrat Shah Jinwoo Shin

Laboratory for Information and Decision Systems
Massachusetts Institute of Technology

Wireless Network

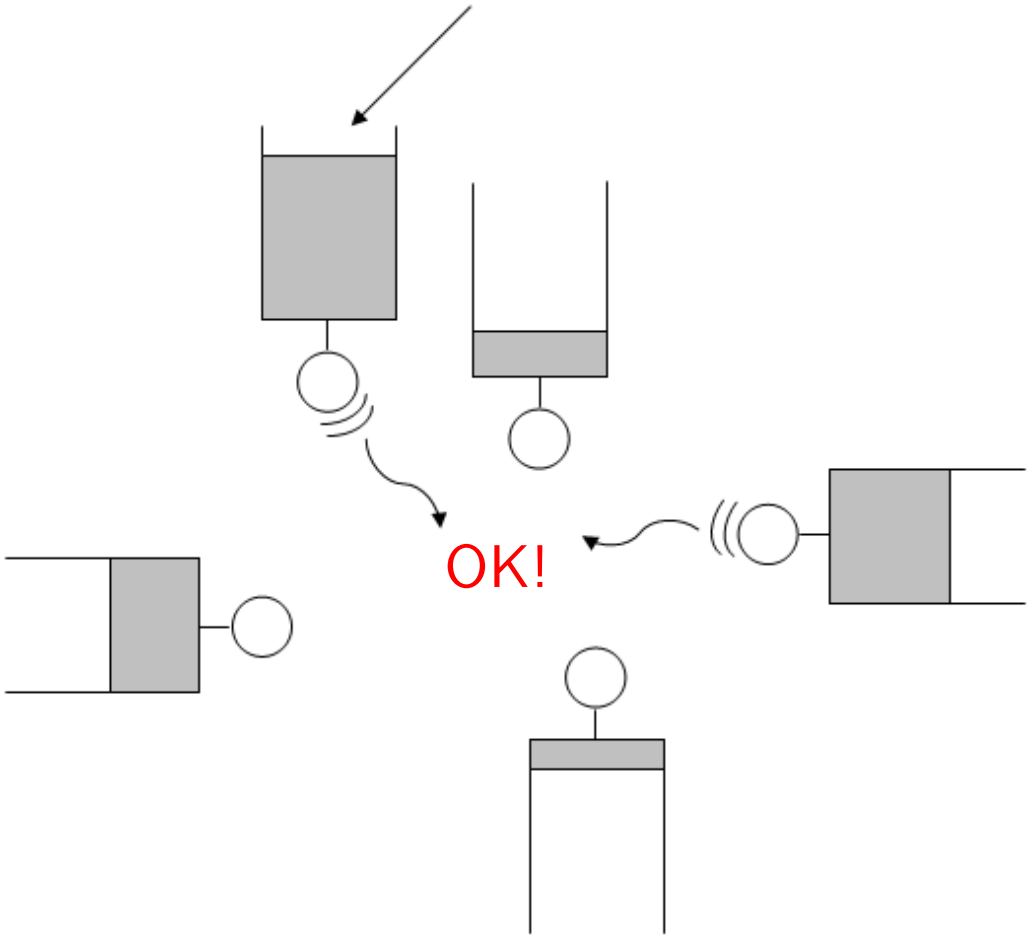


Wireless Network



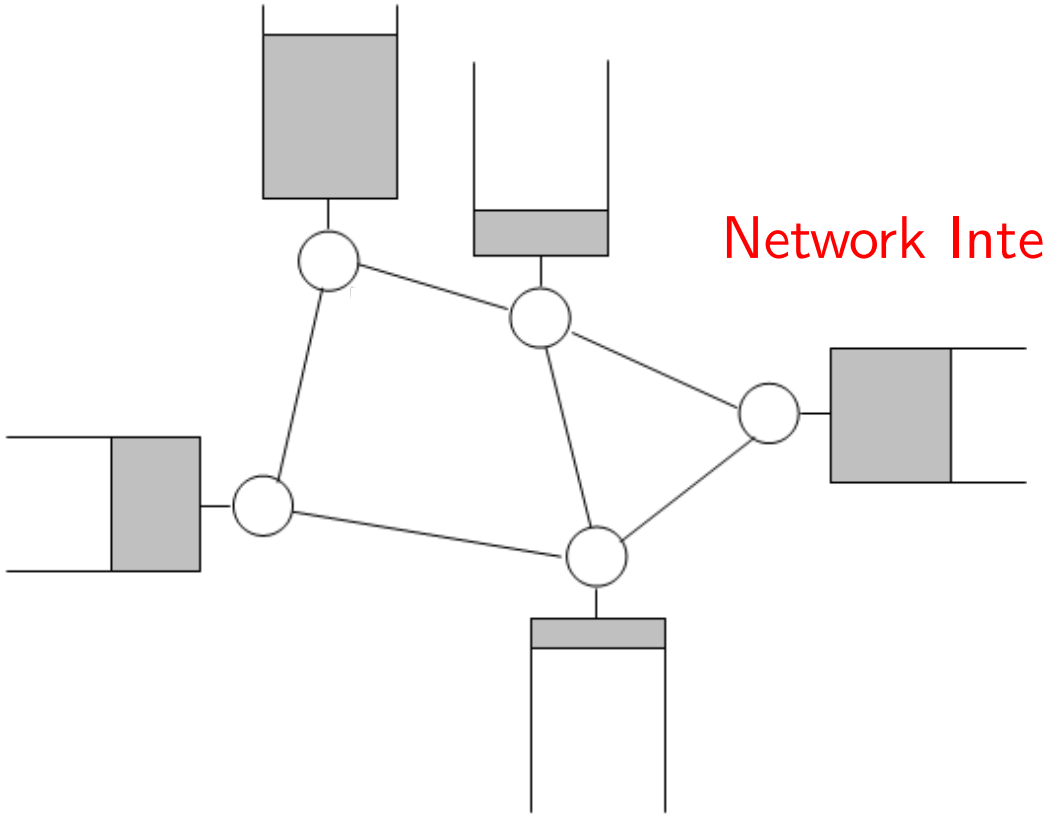
- Constraints
  - Two simultaneously transmitting nodes interfere with each other.
  - Each node has only "local" information.

Wireless Network



- Constraints
  - Two simultaneously transmitting nodes interfere with each other.
  - Each node has only "local" information.

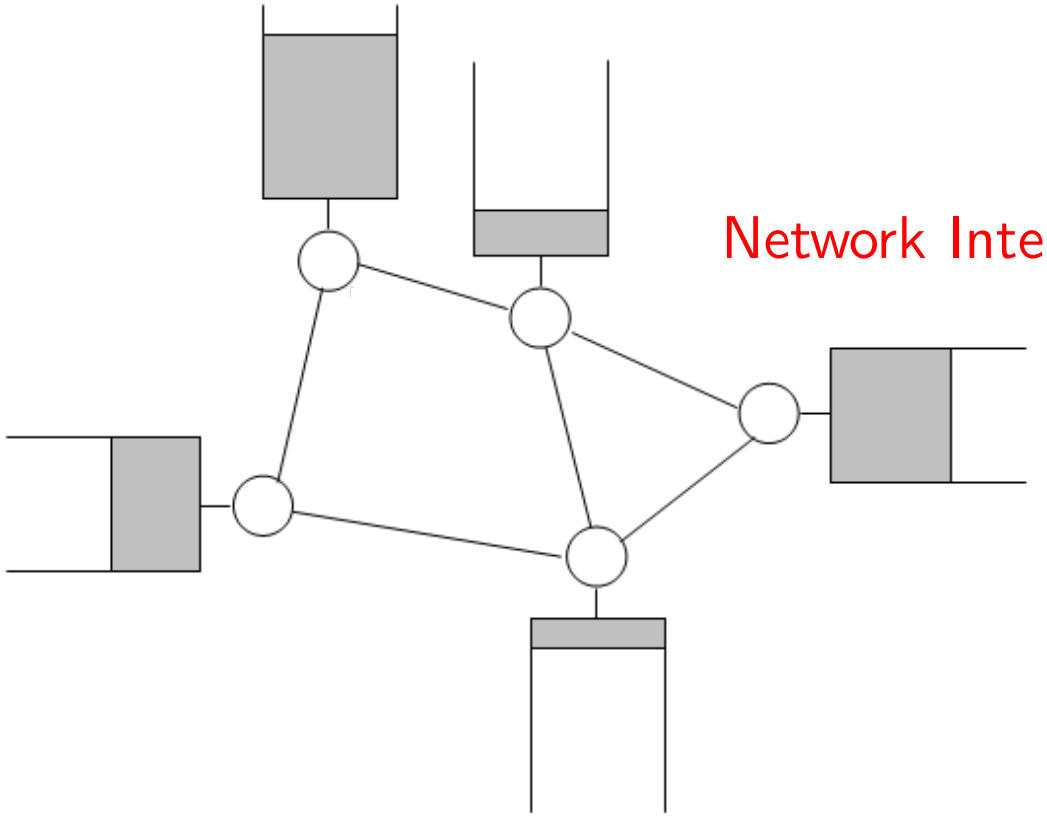
Wireless Network



Network Interference Graph

- Constraints
  - Two simultaneously transmitting nodes interfere with each other.
  - Each node has only "local" information.

Wireless Network



Network Interference Graph

- Question
  - Which nodes should transmit simultaneously using local information.
  - So that performance is not compromised.

Wireless Network: Scheduling Algorithm

- Constraint:
  - Interfering nodes cannot transmit simultaneously.

- Information structure:
  - Slotted-time version
    - Attempt to transmit at the beginning of each time-step.
    - Determine its success/failure at the end of each time-step.
  - Asynchronous-time version
    - Attempt to transmit at any time.
    - Determine its success/failure instantly (Perfect Carrier Sensing).

Wireless Network: Scheduling Algorithm

- Our algorithm would be "Asynchronous-time version"
  - With PCS (Perfect Carrier Sensing).
  - However, possible to apply our algorithm
    - To "Slotted-time version" without PCS
    - Due to [Ni and Srikant 09].

Previous Works

- Single shared medium:
  - Usually slotted-time
    1. Queue-free model
    2. Queueing model

- Network setup:
  - Usually message-passing/PCS(asynchronous-time)
    3. Max-Weight inspired algorithms
    4. Random access algorithms

Single Shared Medium: Queue-free Model

- Each message can attempt to use the medium in each time-step.
- Random back-off (Aloha) protocols:
  - Started from Aloha-system [Abramson and Kuo 73].
  - No sub-exponential back-off protocol achieves the arrival rate $\lambda > 0$ [Kelly et al.87].
  - Binary exponential back-off protocol cannot achieve $\lambda > 0$ [Aldous 87].
  - No back-off protocol achieves $\lambda > .42$ [Goldberg et al.00].
- Full sensing protocols:
  - Every node listens to the medium.
  - "Tree protocol" achieves $\lambda \approx .487$ [Mosely et al. 85].
  - No protocol achieves $\lambda > .568$ [Tsybakov et al. 87].

Single Shared Medium: Queueing Model

- Each queue can attempt to use the medium in each time-step.
  - Fixed N queues.
- Throughput optimal algorithm [Hastad, Leighton, and Rogoff 96]
  - Superlinear polynomial back-off protocol achieves
    - $\lambda < 1$ i.e. throughput-optimal.
  - No binary exponential back-off protocol achieves
    - $\lambda > 0.568$ with the uniform $\lambda_i = \lambda/N$.
- More in the website by Goldberg
  - URL: http://www.csc.liv.ac.uk/∼leslie/contention.html

Network Setup: MW Algorithm

- Select non-interfering nodes in the queueing network
  - So that the summation of their queue-size is maximized.
- Throughput-optimal [Tassiulas and Ephremides 92]
  - Good-delay property [Shah and Wischik 06]
  - Myopic
- Non implementable
  - Not using local information
  - Computationally expensive

Network Setup: MW-inspired Algorithms - 1

Greedy Algorithms

- Serve the longest queue first in a greedy manner.
- Sometimes throughput optimal [Dimakis and Walrand 06]
  - But, not in general [Dai and Prabhakar 00] [Joo, Lin and Shroff 08] [Leconte, Ni and Srikant 09]
- More implementable than MW algorithm
  - But decentralization cost is high depending on network topology.

Network Setup: MW-inspired Algorithms - 2

Randomized Algorithms

- Simpler MW-implementations [Tassiulas 98] [Giaccone, Prabhakar and Shah 03]
  - But still centralized.
- First distributed MW-implementation [Modiano, Shah and Zussman 06]
  - But $O(N^3)$-overhead per each schedule i.e. not practical.
- Constant-overhead distributed algorithm [Sanghavi, Bui and Srikant 07]
  - Only for matching-constraints.
  - Constant is large for throughput-optimality.

Network Setup: Random Access Algorithms

- Find access probabilities
  - Based on the knowledge of the network information
  - References: [Marbach 07] [Eryilmaz, Marbach and Ozdaglar 07], [Gupta and Stolyar 06] [Liu and Stolyar 07] [Stolyar 08]
  - However,
    - *Saturated system* analysis
    - Too much information-exchange
    - *Pareto* throughput-optimal

Network Setup: Random Access Algorithms

- Find access probabilities
  - Based on the knowledge of the network information
  - References: [Marbach 07] [Eryilmaz, Marbach and Ozdaglar 07], [Gupta and Stolyar 06] [Liu and Stolyar 07] [Stolyar 08]
  - However,
    - *Saturated system* analysis
    - Too much information-exchange
    - *Pareto* throughput-optimal

- Adaptively choose access probabilities [Jiang and Walrand 08]
  - Assuming Perfect Carrier Sensing
  - No proof for throughput-optimality

Summary and Goal

Summary of History

- All previous algorithms suffer from one or more of the followings:
  - Non implementable
    - Centralized or large overhead for decentralization
    - Lack of simplicity and elegance
  - Not throughput-optimal
  - Too specific approach or no network

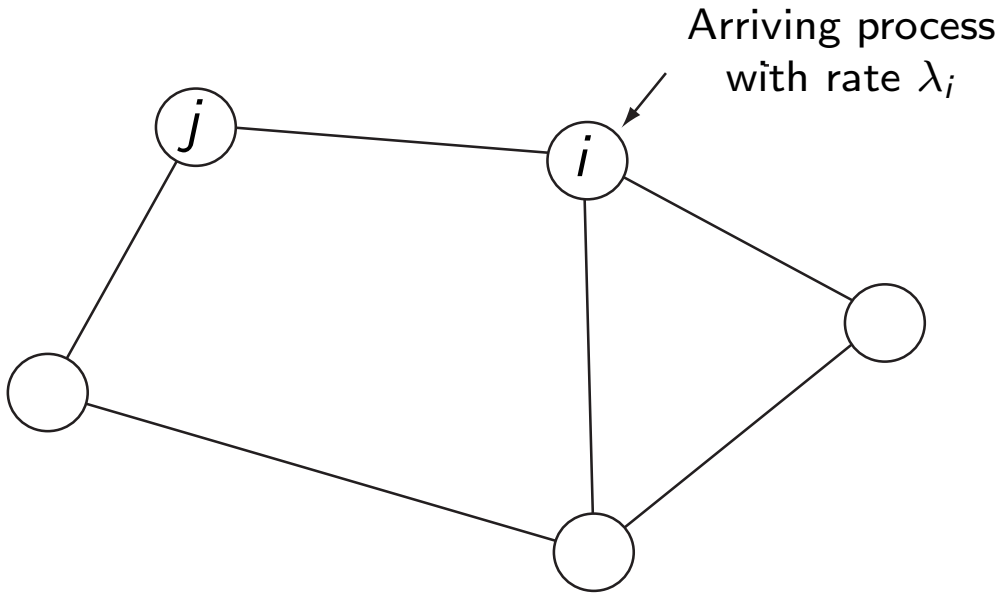
Our Goal

- Design a scheduling algorithm which is
  - Implementable
    - Distributed: low communication cost
    - Simple: low computation & memory cost
  - Throughput-optimal
  - Generic

Outline

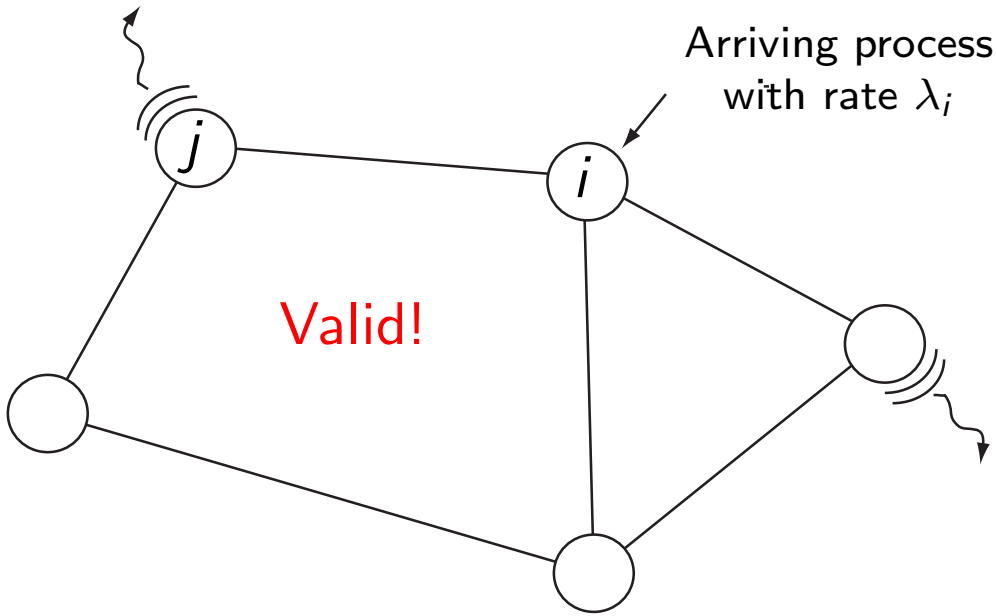
1. Our Algorithm
2. Why it works?
3. Discussions & Summary

Model



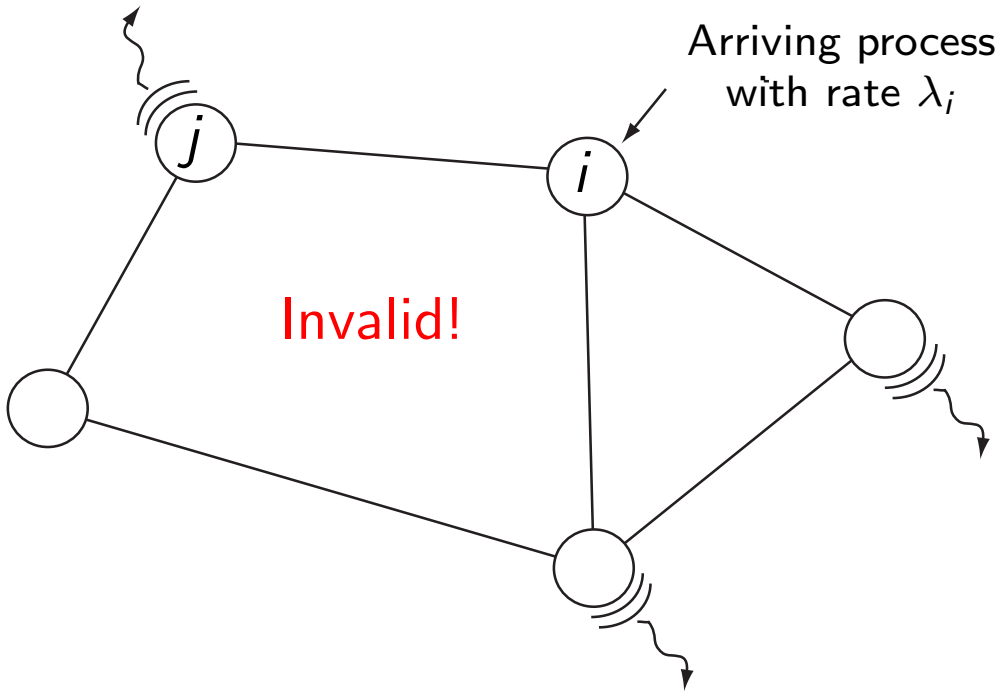
- Network interference graph G of N queues s.t.
  - Independent Poisson packet-arriving process with rate λ_i for queue i.
  - If (i, j) is an edge, i and j cannot transmit simultaneously.
- Scheduling algorithm chooses a valid collection of queues to serve at each time.

Model



- Network interference graph G of N queues s.t.
  - Independent Poisson packet-arriving process with rate λ_i for queue i.
  - If (i, j) is an edge, i and j cannot transmit simultaneously.
- Scheduling algorithm chooses a valid collection of queues to serve at each time.

Model



- Network interference graph G of N queues s.t.
  - Independent Poisson packet-arriving process with rate λ_i for queue i.
  - If (i, j) is an edge, i and j cannot transmit simultaneously.
- Scheduling algorithm chooses a valid collection of queues to serve at each time.

Performance Metric

Recall our goal: Design a simple, distributed, throughput-optimal and generic scheduling algorithm.

Throughput-optimal means

- The capacity region Λ be the set of all arrival rate $\lambda = [\lambda_i]$ s.t.
  - There exists an algorithm that can keep queues finite under λ.
- An scheduling algorithm is throughput-optimal if
  - It keeps queues finite under $\lambda \in \Lambda$.

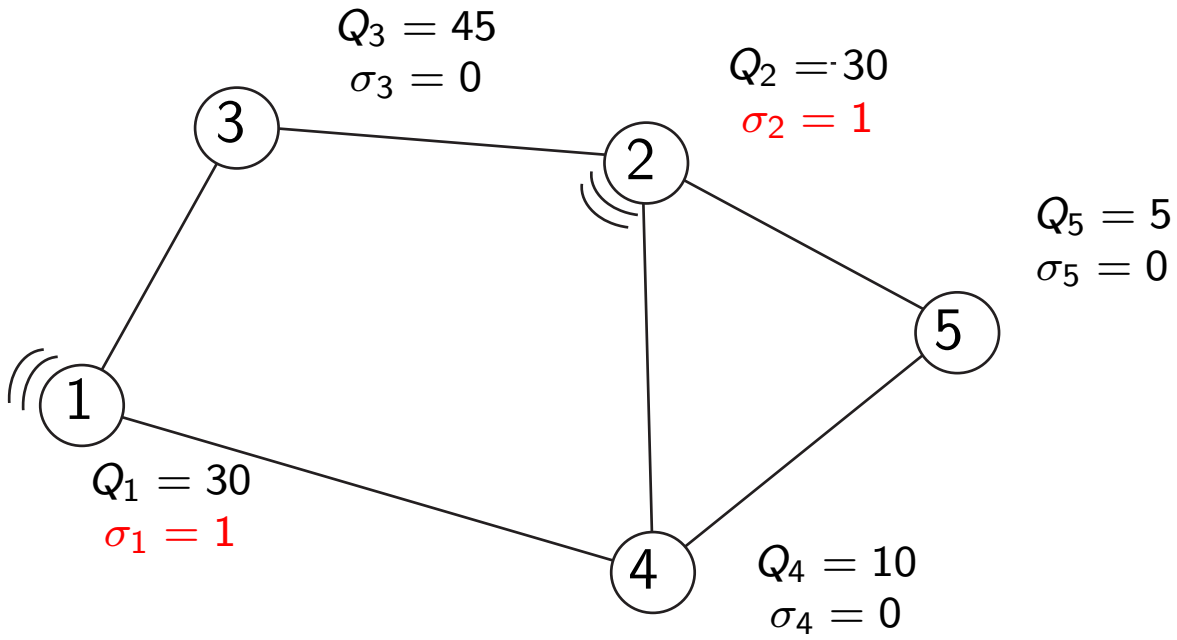
Distributed means

- With respect to the interference graph G.

Notations

- Network interference graph $G = (V, E)$ of N queues.
- $\mathbf{Q}(t) = [Q_i(t)] \in \mathbb{R}_+^N$ be the queue-sizes at time t.
- $\boldsymbol{\sigma}(t) = [\sigma_i(t)] \in \{0,1\}^N$ be the schedule at time t.
  - $\sigma_i(t) = 1$ means the queue i is transmitting at time t.
  - $\boldsymbol{\sigma}(t) \in \mathcal{I}(G) := \{\boldsymbol{\sigma} \in \{0,1\}^N : \sigma_i + \sigma_j \leq 1 \text{ for all } (i,j) \in E\}$.

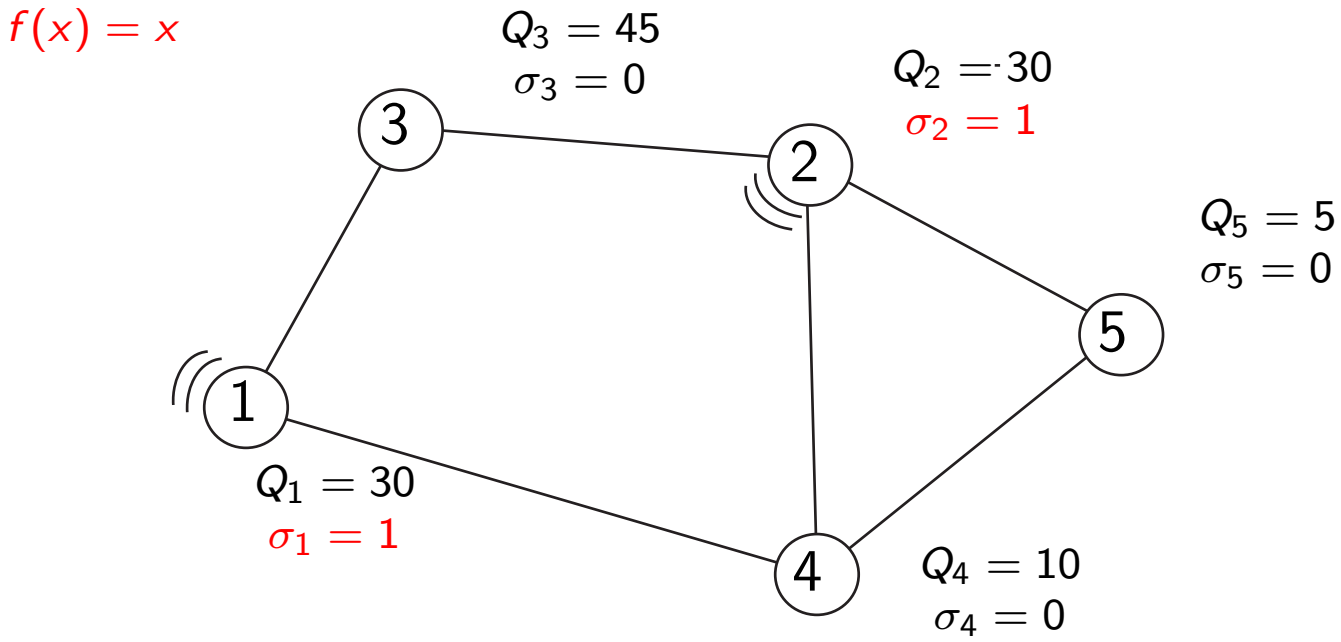
Recall MW Algorithm



MW Algorithm

- Choose a valid collection of queues at each time so that their summation is maximized.
  - $\boldsymbol{\sigma}(t) = \arg\max_{\boldsymbol{\sigma} \in \mathcal{I}(G)} \sum_i Q_i(t) \cdot \sigma_i$.
- Throughput-optimal [Tassiulas and Ephremides 92]
- But high complexity for implementation

Recall MW Algorithm



f-MW Algorithm

- Choose a valid collection of queues at each time so that their summation is maximized with respect to f.
  - $\boldsymbol{\sigma}(t) = \arg\max_{\boldsymbol{\sigma} \in \mathcal{I}(G)} \sum_i f(Q_i(t)) \cdot \sigma_i$.
- Throughput-optimal [Shah and Wischik 06]
- But high complexity for implementation

Our Algorithm

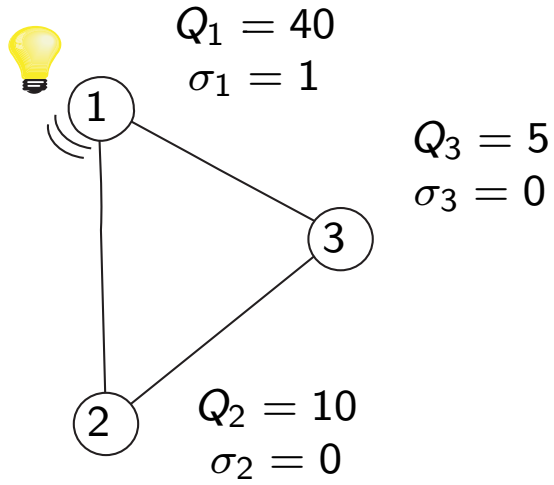
- Each queue has an independent Exponential clock of rate 1.
- When the clock of the queue i ticks at time t,
  - i checks whether the medium is free
    - i.e. no neighbor of i is transmitting.
  - If yes,

$$\sigma_i(t^+) = \begin{cases} 1 & \text{with probability } \frac{\exp[f(Q_i(t))]}{1+\exp[f(Q_i(t))]} \\ 0 & \text{otherwise.} \end{cases}.$$

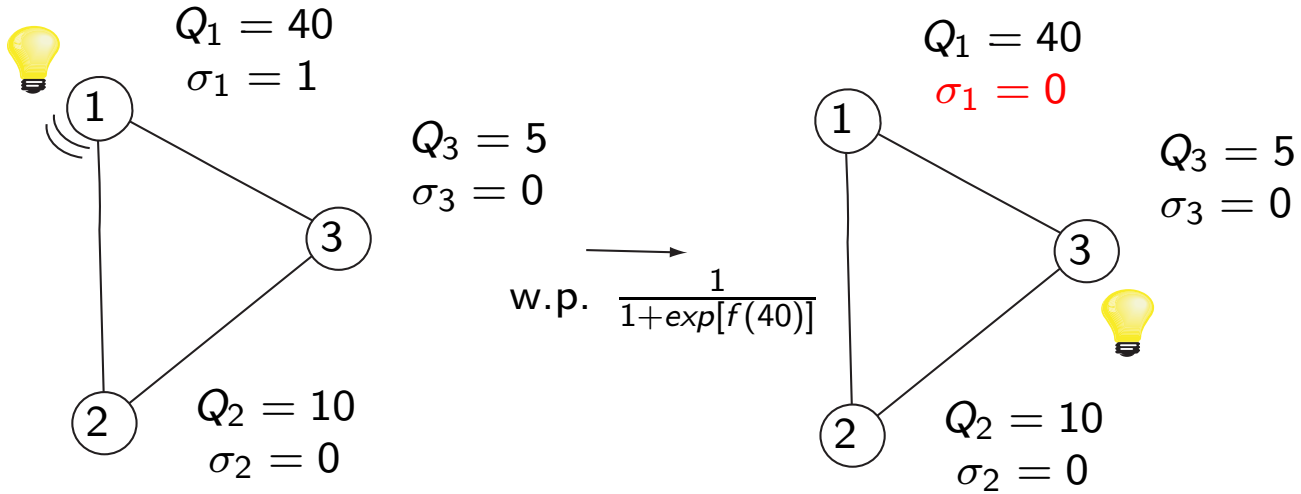
  - Else, do nothing.

$\star$ f will be decided later.

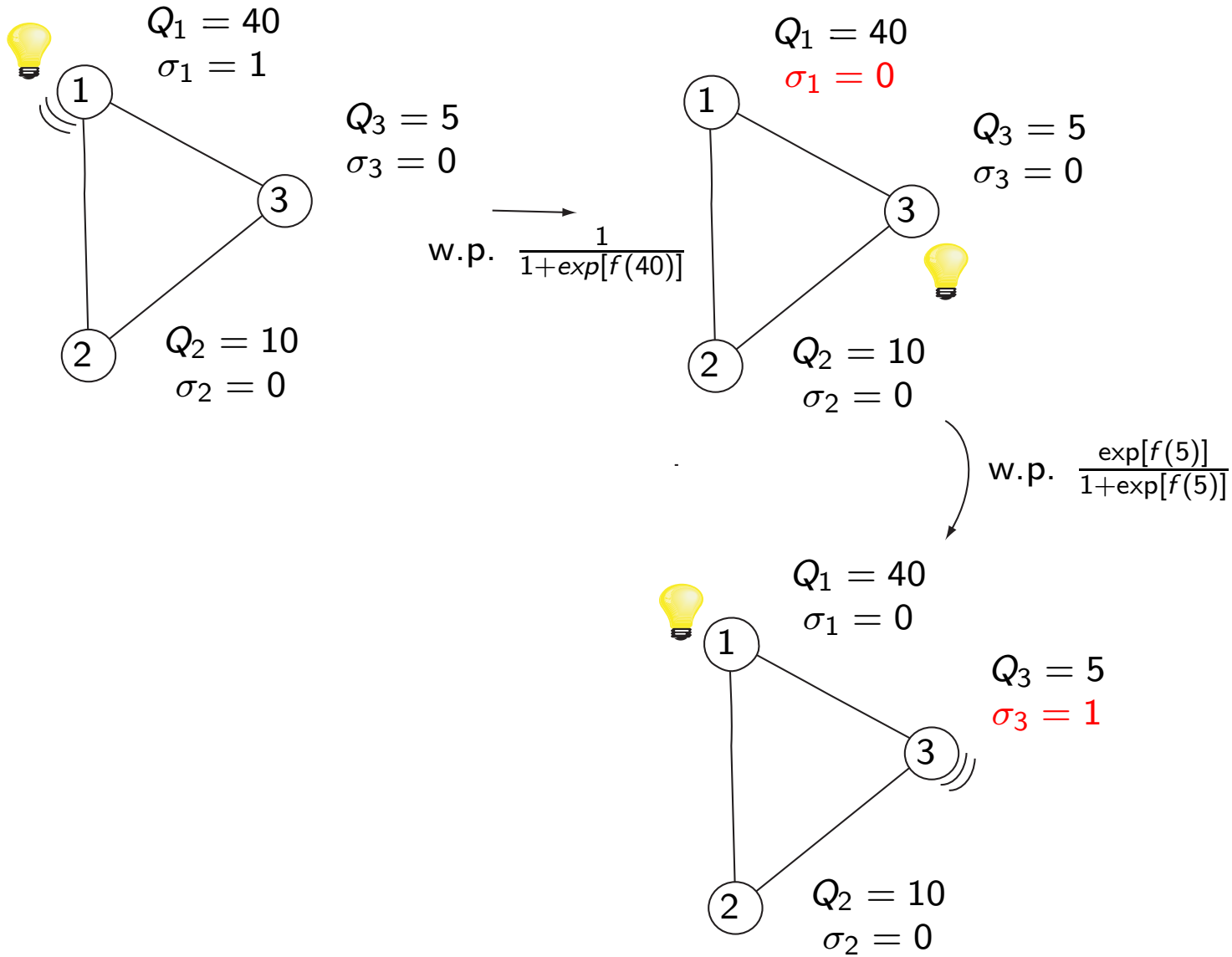
Our Algorithm: Example



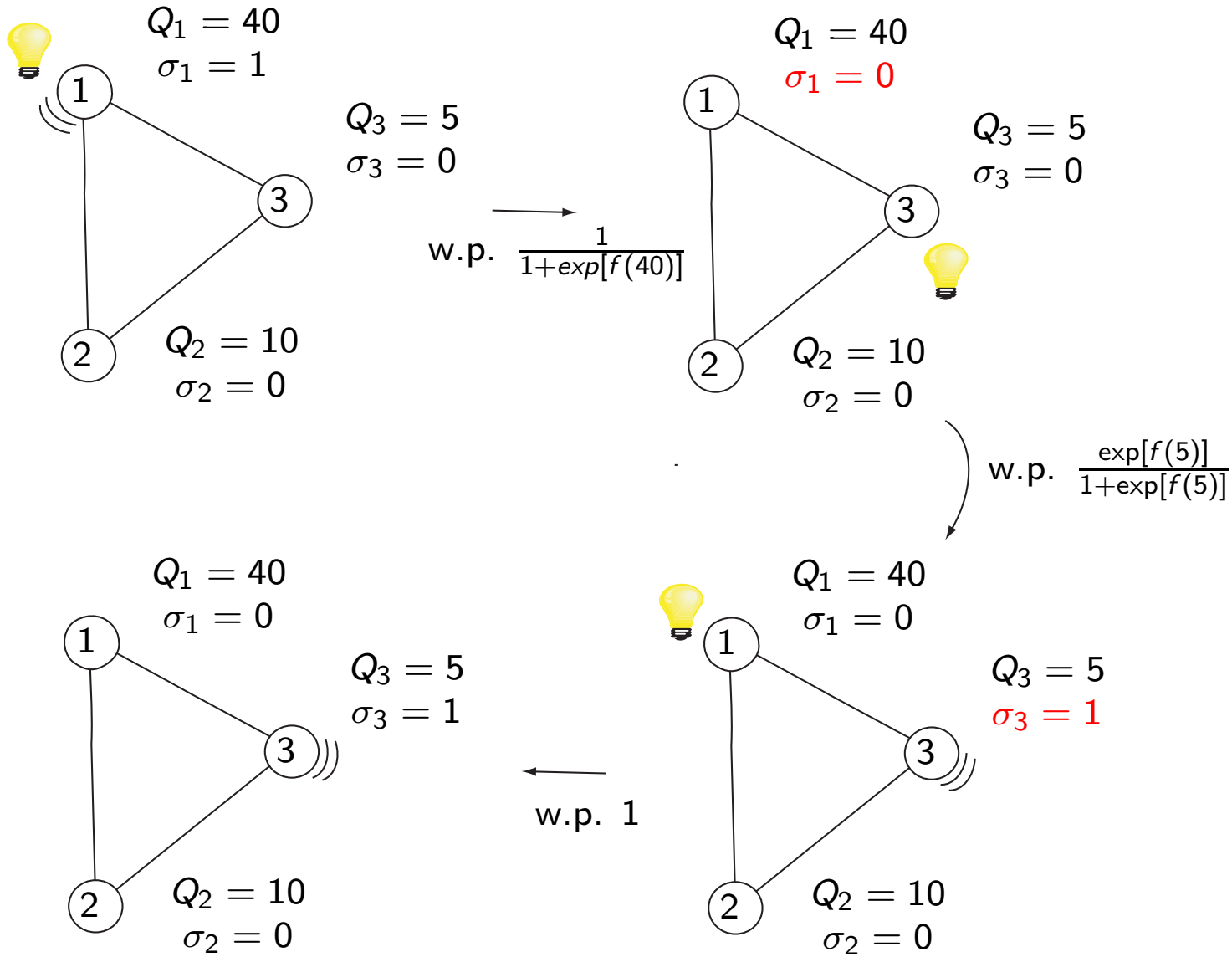
Our Algorithm: Example



Our Algorithm: Example



Our Algorithm: Example



Our Algorithm: Throughput-optimality

Theorem

The algorithm is throughput-optimal with an appropriate choice of f.

- We will go through the proof intuition and search for a proper f.
  - Essentially $f(x) \approx \log\log x$.

Our Algorithm ≈ Glauber Dynamics

- Assume $\mathbf{Q}(t)$ is fixed.

Recall our algorithm

- Each queue has an independent Exponential clock of rate 1.
- When the clock of the queue i ticks at time t,
  - i checks whether the medium is free
  - If yes,

$$\sigma_i(t^+) = \begin{cases} 1 & \text{with probability } \frac{\exp[f(Q_i(t))]}{1+\exp[f(Q_i(t))]} \\ 0 & \text{otherwise.} \end{cases}.$$

  - Else, do nothing.

- Our algorithm runs a finite state, reversible Markov chain on $\mathcal{I}(G)$.
  - It is well-known as *Glauber dynamics*.

Our Algorithm $\approx$ Glauber Dynamics

- Assuming $\mathbf{Q}(t) = \mathbf{Q}$ is fixed,
  - Our algorithm runs Glauber dynamics.

Properties of Glauber Dynamics

- It has the unique stationary distribution π s.t.
  - $\pi(\boldsymbol{\sigma}) \propto \exp\left[\sum_i f(Q_i) \cdot \sigma_i\right]$.
    - High mass on large $f(\mathbf{Q})$-weighted schedules

Our Algorithm $\approx$ Glauber Dynamics

- Assuming $\mathbf{Q}(t) = \mathbf{Q}$ is fixed,
  - Our algorithm runs Glauber dynamics.

Properties of Glauber Dynamics

- It has the unique stationary distribution π s.t.
  - $\pi(\boldsymbol{\sigma}) \propto \exp\left[\sum_i f(Q_i) \cdot \sigma_i\right]$.
    - High mass on large $f(\mathbf{Q})$-weighted schedules
  - $\mathbb{E}_\pi\left[\sum_i f(Q_i) \cdot \sigma_i\right] \geq \left(\arg\max_{\boldsymbol{\rho} \in \mathcal{I}(G)} \sum_i f(Q_i) \cdot \rho_i\right) - N$
    - Sampling $\boldsymbol{\sigma}$ w.r.t π is essentially a f-MW choice!

Our Algorithm $\approx$ Glauber Dynamics

- Assuming $\mathbf{Q}(t) = \mathbf{Q}$ is fixed,
  - Our algorithm runs Glauber dynamics.

Properties of Glauber Dynamics

- It has the unique stationary distribution π s.t.
  - $\mathbb{E}_\pi \left[\sum_i f(Q_i) \cdot \sigma_i\right] \geq \left(\arg\max_{\boldsymbol{\rho} \in \mathcal{I}(G)} \sum_i f(Q_i) \cdot \rho_i\right) - N$
    - Sampling $\boldsymbol{\sigma}$ w.r.t π is essentially a f-MW choice!
- The actual sampling distribution $\mu(t)$ converges to π.

Our Algorithm $\approx$ Glauber Dynamics

- Assuming $\mathbf{Q}(t) = \mathbf{Q}$ is fixed,
  - Our algorithm samples essentially the f-MW schedule.
    - Actual sampling distribution $\mu(t)$ converges to f-MW distribution π.
  - Hence, throughput-optimal for any increasing f
    - For example, $f(x) = x$, $\sqrt{x}$, $\log x$, $\log\log x$, $\ldots$.

Our Algorithm $\approx$ Glauber Dynamics

- Assuming $\mathbf{Q}(t) = \mathbf{Q}$ is fixed,
  - Our algorithm samples essentially the f-MW schedule.
    - Actual sampling distribution $\mu(t)$ converges to f-MW distribution π.
  - Hence, throughput-optimal for any increasing f
    - For example, $f(x) = x,\ \sqrt{x},\ \log x,\ \log\log x,\ \ldots$.

- However, $\mathbf{Q}(t)$ does change.
  - Our algorithm runs time-varying Glauber dynamics.
  - $\pi = \pi(t)$ also changes.

Our Algorithm $\approx$ Glauber Dynamics

- Assuming $\mathbf{Q}(t) = \mathbf{Q}$ is fixed,
  - Our algorithm samples essentially the f-MW schedule.
    - Actual sampling distribution $\mu(t)$ converges to f-MW distribution π.
  - Hence, throughput-optimal for any increasing f
    - For example, $f(x) = x$, $\sqrt{x}$, $\log x$, $\log\log x$, $\ldots$.

- However, $\mathbf{Q}(t)$ does change.
  - Our algorithm runs time-varying Glauber dynamics.
  - $\pi = \pi(t)$ also changes.
  - Question: $\mu(t)$ still converges to $\pi(t)$?

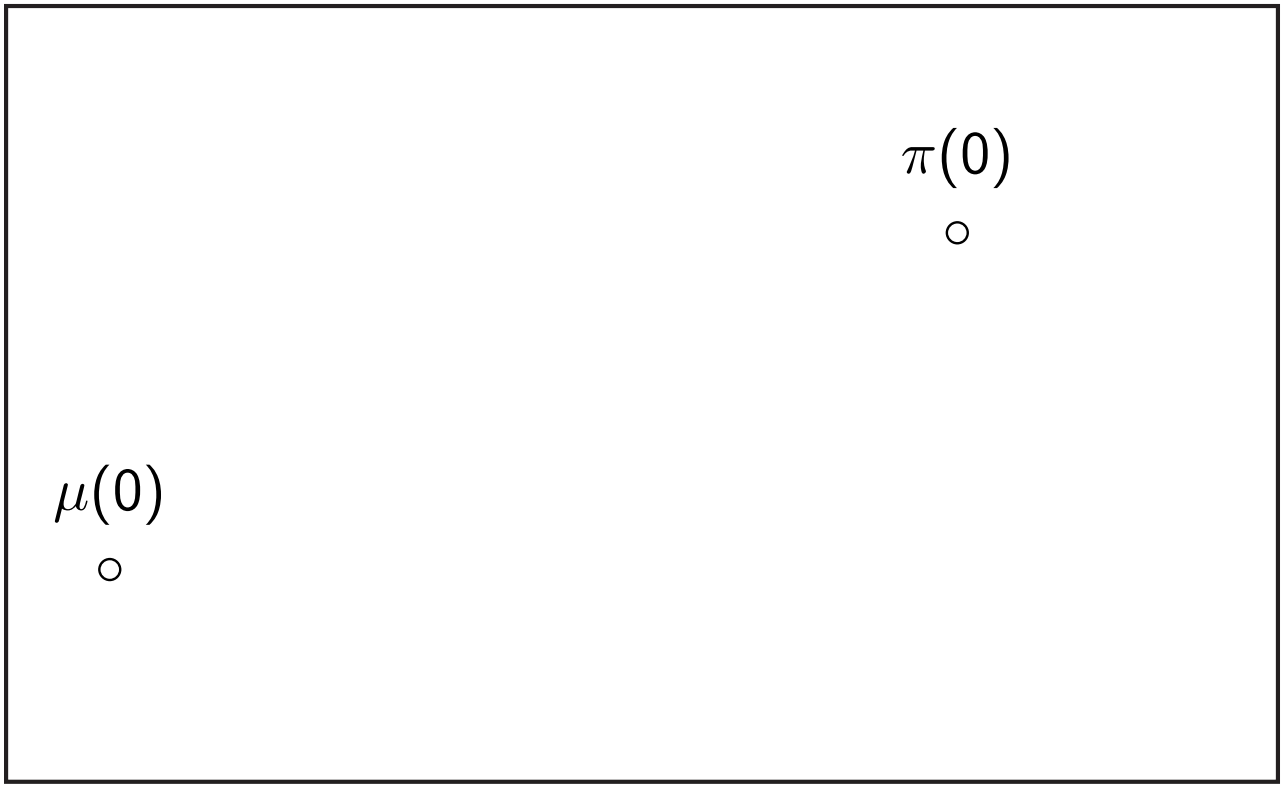
Our Algorithm $\approx$ Glauber Dynamics

- Assuming $\mathbf{Q}(t) = \mathbf{Q}$ is fixed,
  - Our algorithm samples essentially the f-MW schedule.
    - Actual sampling distribution $\mu(t)$ converges to f-MW distribution π.
  - Hence, throughput-optimal for any increasing f
    - For example, $f(x) = x$, $\sqrt{x}$, $\log x$, $\log\log x$, $\ldots$.

- However, $\mathbf{Q}(t)$ does change.
  - Our algorithm runs time-varying Glauber dynamics.
  - $\pi = \pi(t)$ also changes.
  - Question: $\mu(t)$ still converges to $\pi(t)$?
  - If yes, our algorithm is throughput-optimal!

Our Algorithm $\approx$ Time-varying Glauber Dynamics

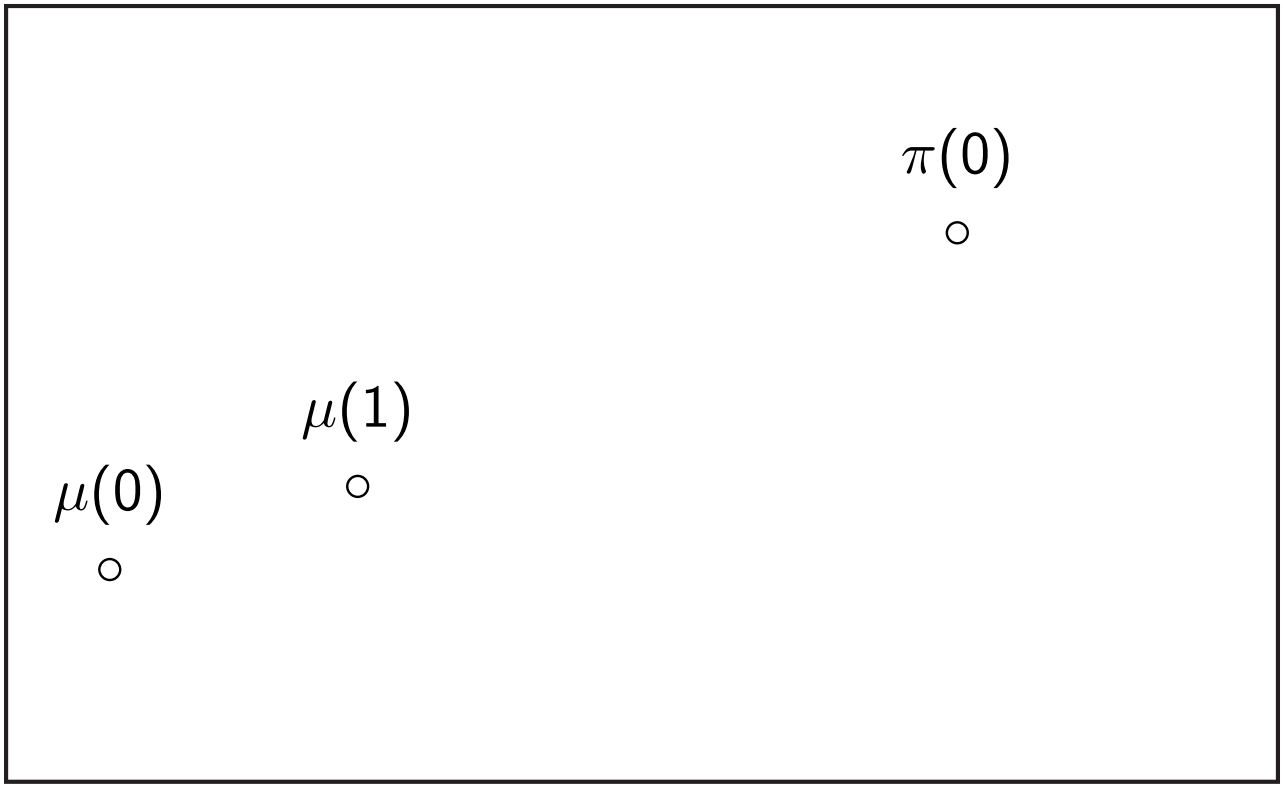
- Question: $\mu(t)$ still converges to $\pi(t)$?



- Consider $\pi(0)$ and $\mu(0)$ in the space of probability distributions.

Our Algorithm $\approx$ Time-varying Glauber Dynamics

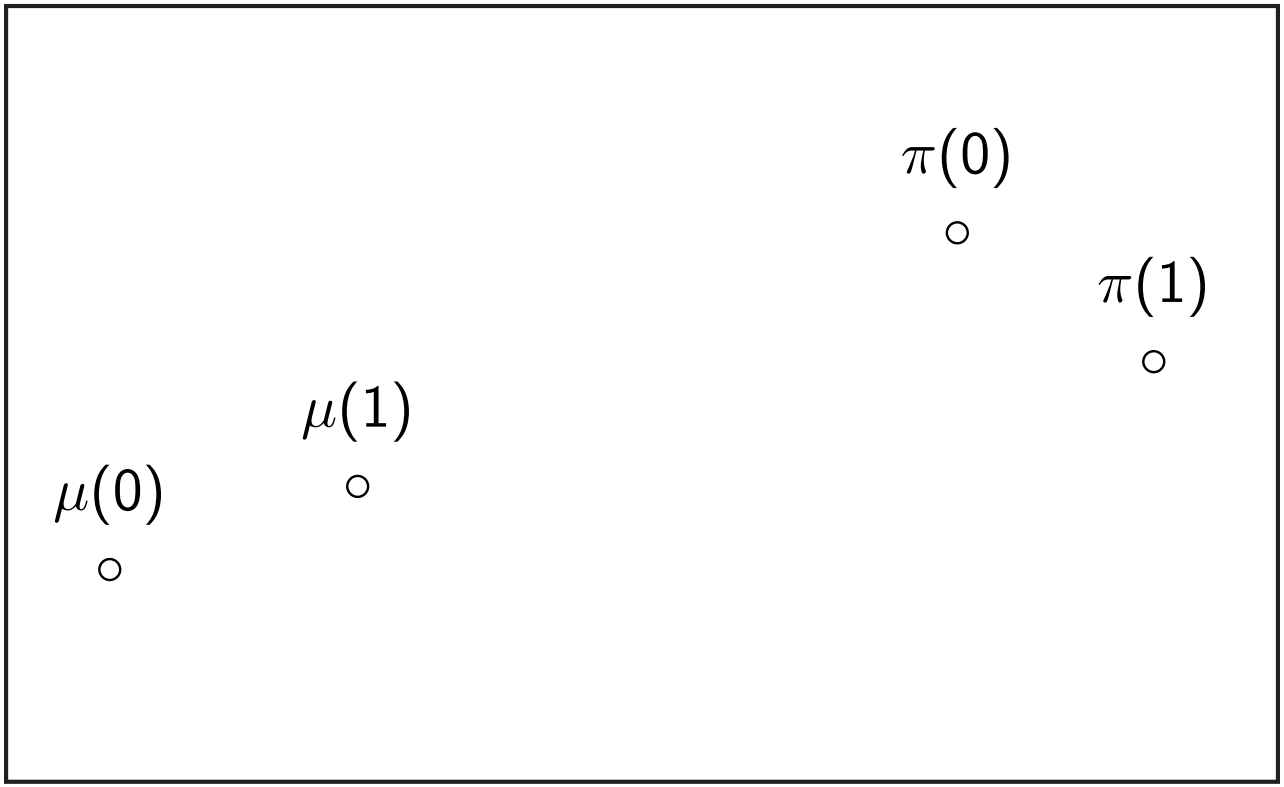
- Question: $\mu(t)$ still converges to $\pi(t)$?



- $\mu(0)$ moves toward $\pi(0)$.

Our Algorithm $\approx$ Time-varying Glauber Dynamics

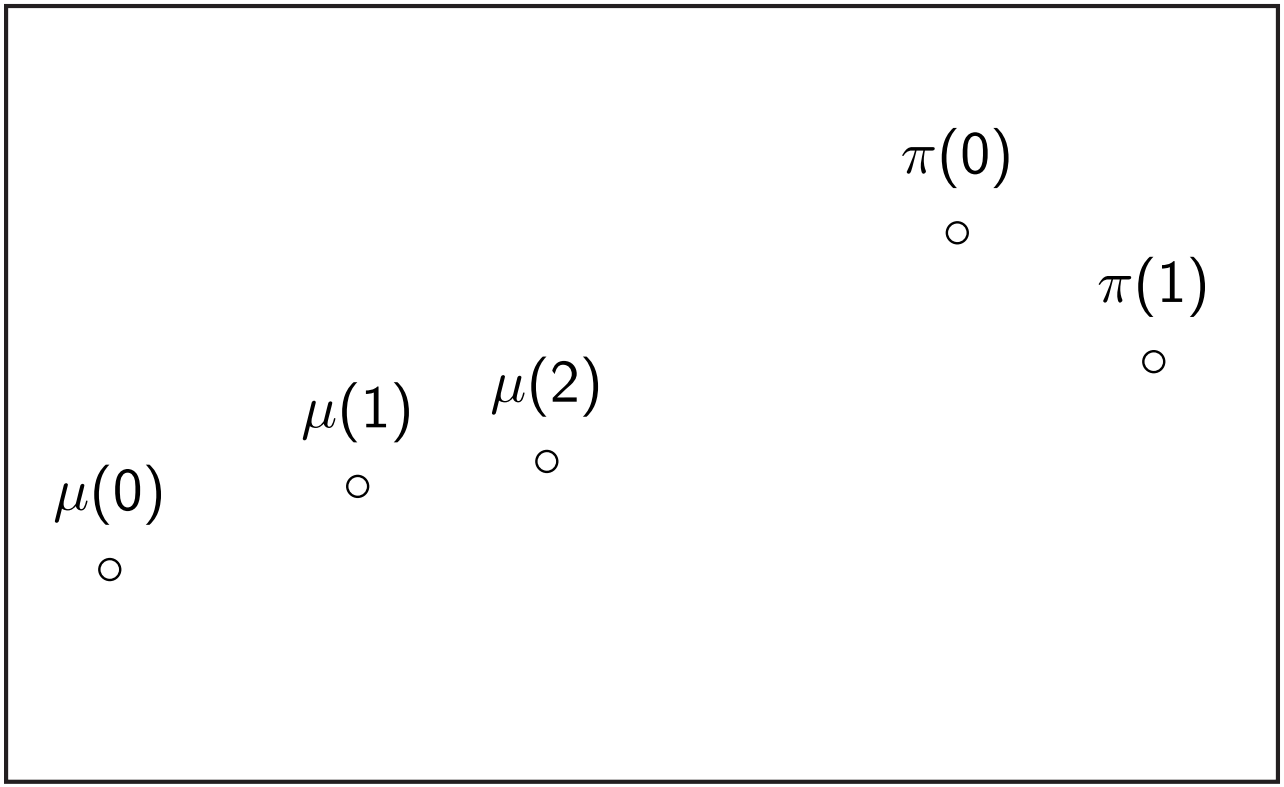
- Question: $\mu(t)$ still converges to $\pi(t)$?



- $\pi(0)$ may runs away.

Our Algorithm $\approx$ Time-varying Glauber Dynamics

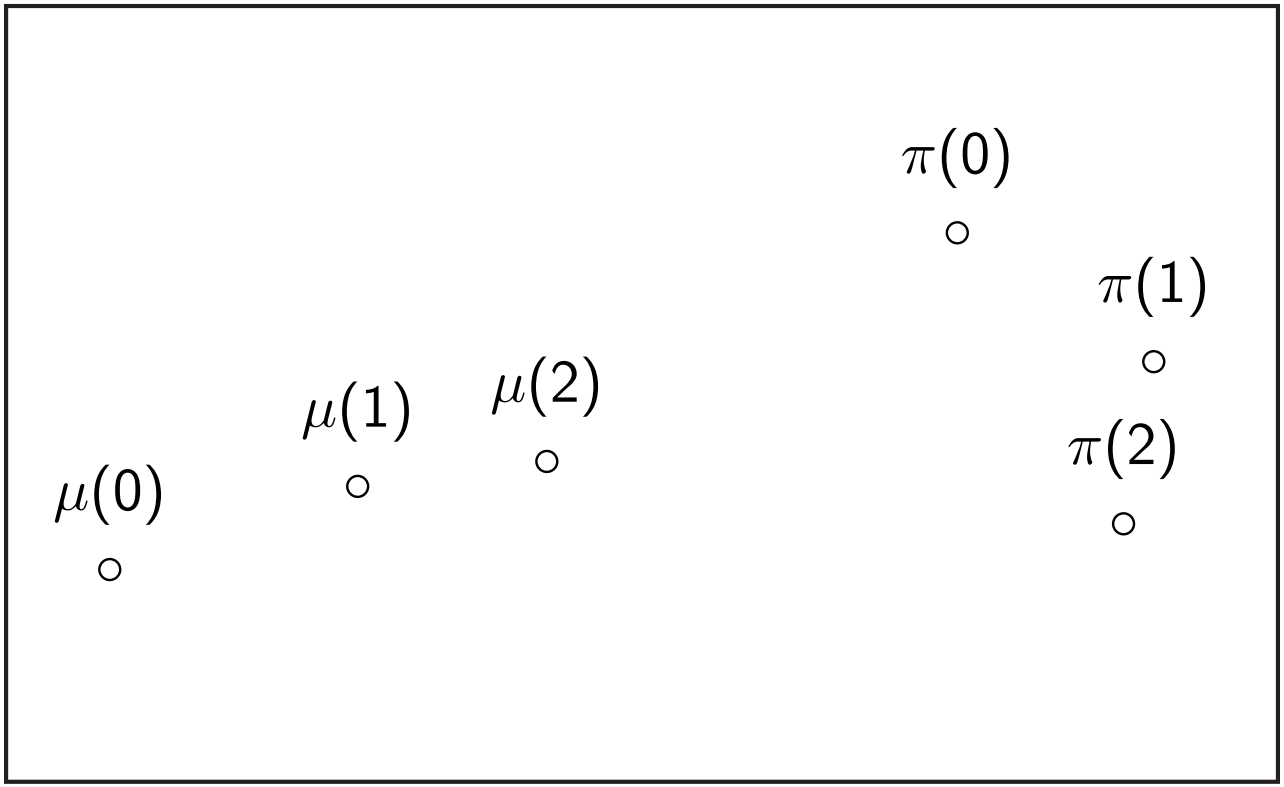
- Question: $\mu(t)$ still converges to $\pi(t)$?



- $\mu(1)$ moves toward $\pi(1)$.

Our Algorithm $\approx$ Time-varying Glauber Dynamics

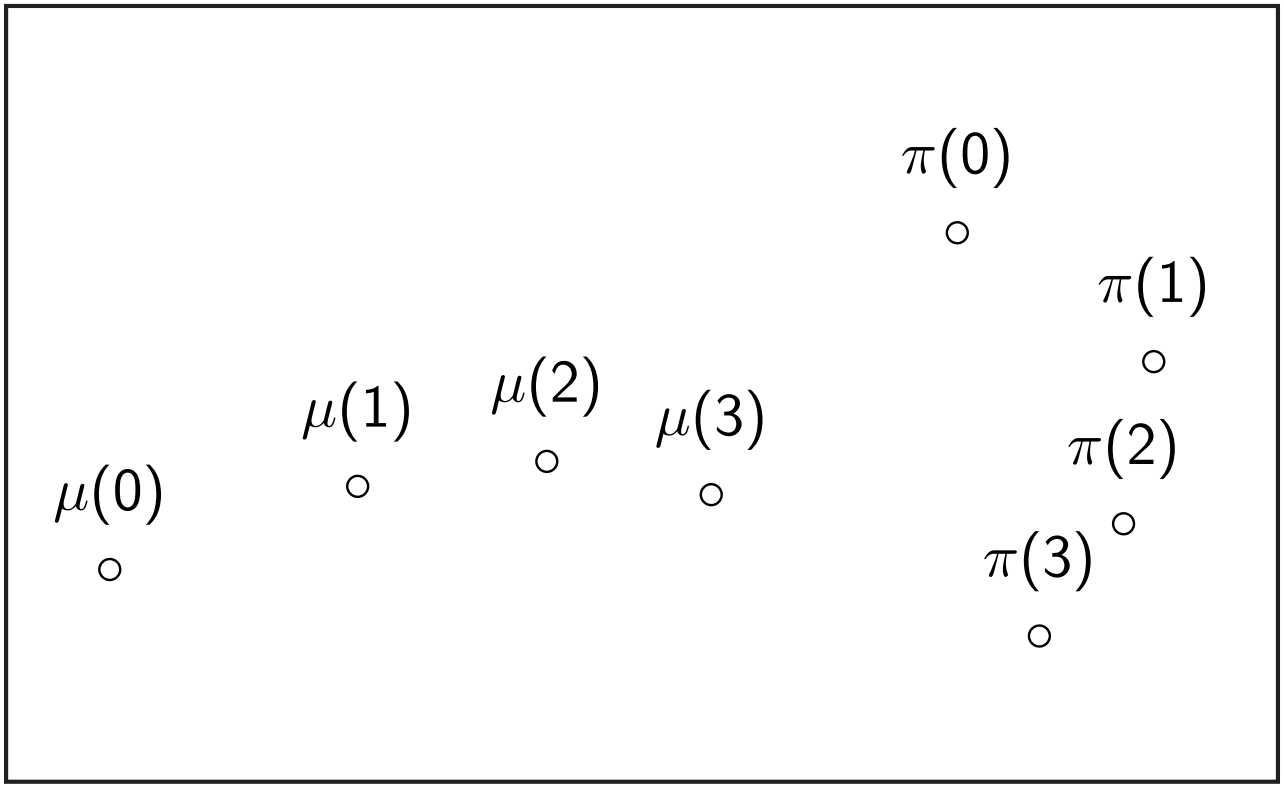
- Question: $\mu(t)$ still converges to $\pi(t)$?



- $\pi(1)$ may runs away.

Our Algorithm $\approx$ Time-varying Glauber Dynamics

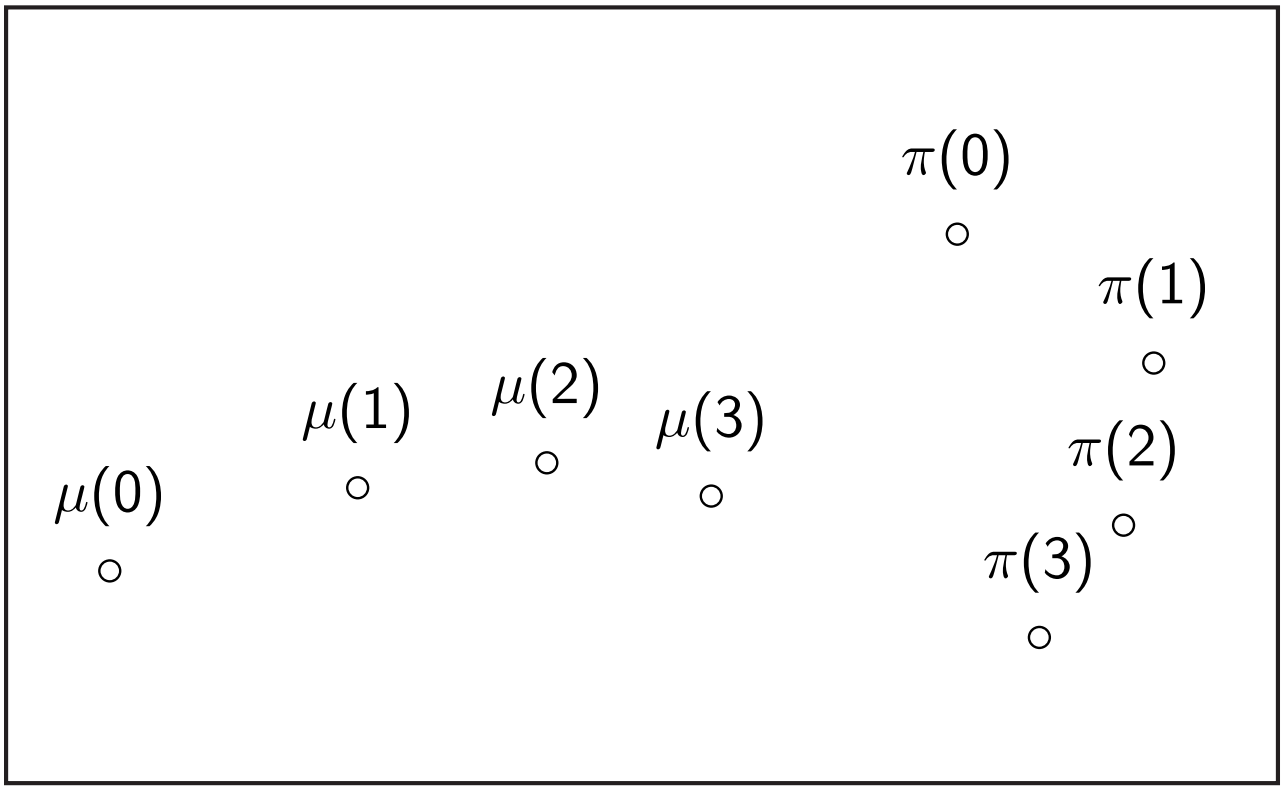
- Question: $\mu(t)$ still converges to $\pi(t)$?



- Continue ...

Our Algorithm $\approx$ Time-varying Glauber Dynamics

- Question: $\mu(t)$ still converges to $\pi(t)$?



- If $\pi(t)$ moves slower than $\mu(t)$,
  - $\mu(t)$ eventually catch up $\pi(t)$!
  - Need to analyze the speed of $\pi(t)$ and $\mu(t)$.

Our Algorithm $\approx$ Time-varying Glauber Dynamics

Speed of $\pi(t)$

- $Q(t)$ changes at unit rate.

Our Algorithm $\approx$ Time-varying Glauber Dynamics

Speed of $\pi(t)$

- $Q(t)$ changes at unit rate.
- $\Delta\pi(t) \approx \Delta f(Q(t)) = f'(Q(t))$.

Our Algorithm $\approx$ Time-varying Glauber Dynamics

Speed of $\pi(t)$

- $Q(t)$ changes at unit rate.
- $\Delta\pi(t) \approx \Delta f(Q(t)) = f'(Q(t))$.

Speed of $\mu(t)$

- The mixing time T of Glauber dynamics determines the speed of $\mu(t)$.

Our Algorithm $\approx$ Time-varying Glauber Dynamics

Speed of $\pi(t)$

- $Q(t)$ changes at unit rate.
- $\Delta\pi(t) \approx \Delta f(Q(t)) = f'(Q(t))$.

Speed of $\mu(t)$

- The mixing time T of Glauber dynamics determines the speed of $\mu(t)$.
- $\Delta\mu(t) \approx \frac{1}{T} \approx \frac{1}{\exp[f(Q(t))]}$.

Our Algorithm $\approx$ Time-varying Glauber Dynamics

Speed of $\pi(t)$

- $Q(t)$ changes at unit rate.
- $\Delta\pi(t) \approx \Delta f(Q(t)) = f'(Q(t))$.

Speed of $\mu(t)$

- The mixing time T of Glauber dynamics determines the speed of $\mu(t)$.
- $\Delta\mu(t) \approx \frac{1}{T} \approx \frac{1}{\exp[f(Q(t))]}$.

- Main issue:

$$\Delta\pi(t) \ll \Delta\mu(t)\,? \qquad \Longleftrightarrow \qquad f'(Q(t)) \ll \frac{1}{\exp[f(Q(t))]}\,?$$

Our Algorithm $\approx$ Time-varying Glauber Dynamics

- Want
$$f'(Q(t)) \ll \frac{1}{\exp[f(Q(t))]}.$$

- Try $f(x) = x$.
  - $$f'(Q(t)) = 1 \qquad \text{and} \qquad \frac{1}{\exp[f(Q(t))]} = \frac{1}{\exp[Q(t)]}.$$
  - No!

Our Algorithm $\approx$ Time-varying Glauber Dynamics

- Want
$$f'(Q(t)) \ll \frac{1}{\exp[f(Q(t))]}.$$

- Try $f(x) = \log x$.
  - $$f'(Q(t)) = \frac{1}{Q(t)} \quad \text{and} \quad \frac{1}{\exp[f(Q(t))]} = \frac{1}{Q(t)}.$$
  - Close, but still not enough!

Our Algorithm $\approx$ Time-varying Glauber Dynamics

- Want
$$f'(Q(t)) \ll \frac{1}{\exp[f(Q(t))]}.$$

- Try $f(x) = \log\log x$.
  -
$$f'(Q(t)) = \frac{1}{Q(t)\log Q(t)} \quad \text{and} \quad \frac{1}{\exp[f(Q(t))]} = \frac{1}{\log Q(t)}.$$
  - Good if $Q(t)$ is large!
  - Analysis for Throughput optimality only cares the case when $Q(t)$ is large.

Throughput-optimality

Theorem

The underlying network Markov process induced by our algorithm* with $f(x) = \log\log x$ is Positive (Harris) recurrence if $\lambda \in \Lambda$.

Throughput-optimality

Theorem

The underlying network Markov process induced by our algorithm* with $f(x) = \log\log x$ is Positive (Harris) recurrence if $\lambda \in \Lambda$.

Here, $\star$ means a minor modification of the weight $f(Q_i)$ in our algorithm

- Using an estimation of $Q_{\max}(t) = \max_i Q_i(t)$
  - Minimal global information
  - For such an estimation,
    - Only one-bit message needs to communicate between neighbors per each time!

Our Algorithm*

- Each queue has an independent Exponential clock of rate 1.
- When the clock of the queue i ticks at time t,
  - i checks whether the medium is free
  - If yes,

$$\sigma_i(t^+) = \begin{cases} 1 & \text{with probability } \frac{\exp[W_i(t)]}{1+\exp[W_i(t)]} \\ 0 & \text{otherwise.} \end{cases}.$$

  - Else, do nothing.
- $W_i(t) = f(Q_i(t))$.

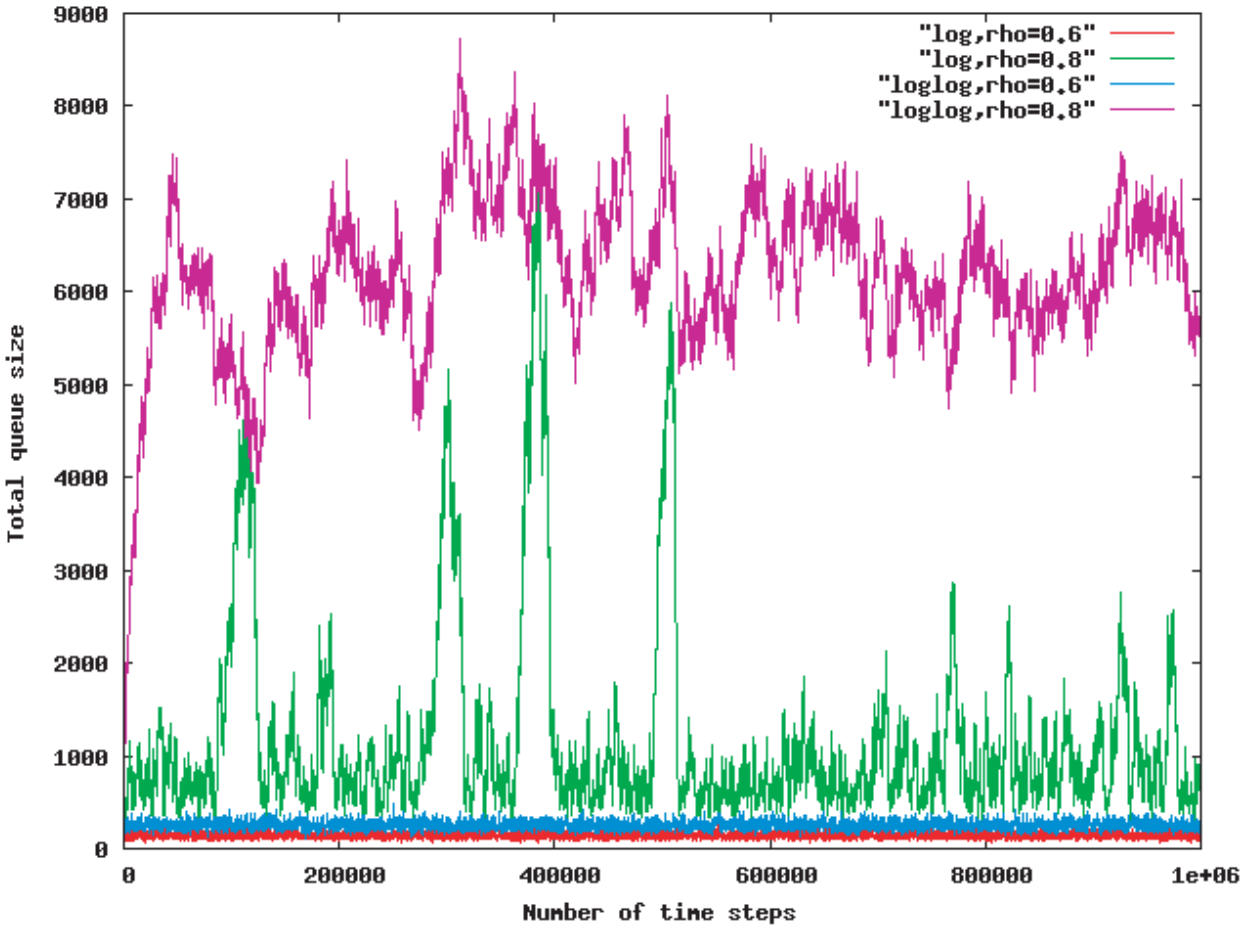
Our Algorithm*

- Each queue has an independent Exponential clock of rate 1.
- When the clock of the queue i ticks at time t,
  - i checks whether the medium is free
  - If yes,

$$\sigma_i(t^+) = \begin{cases} 1 & \text{with probability } \frac{\exp[W_i(t)]}{1+\exp[W_i(t)]} \\ 0 & \text{otherwise.} \end{cases}.$$

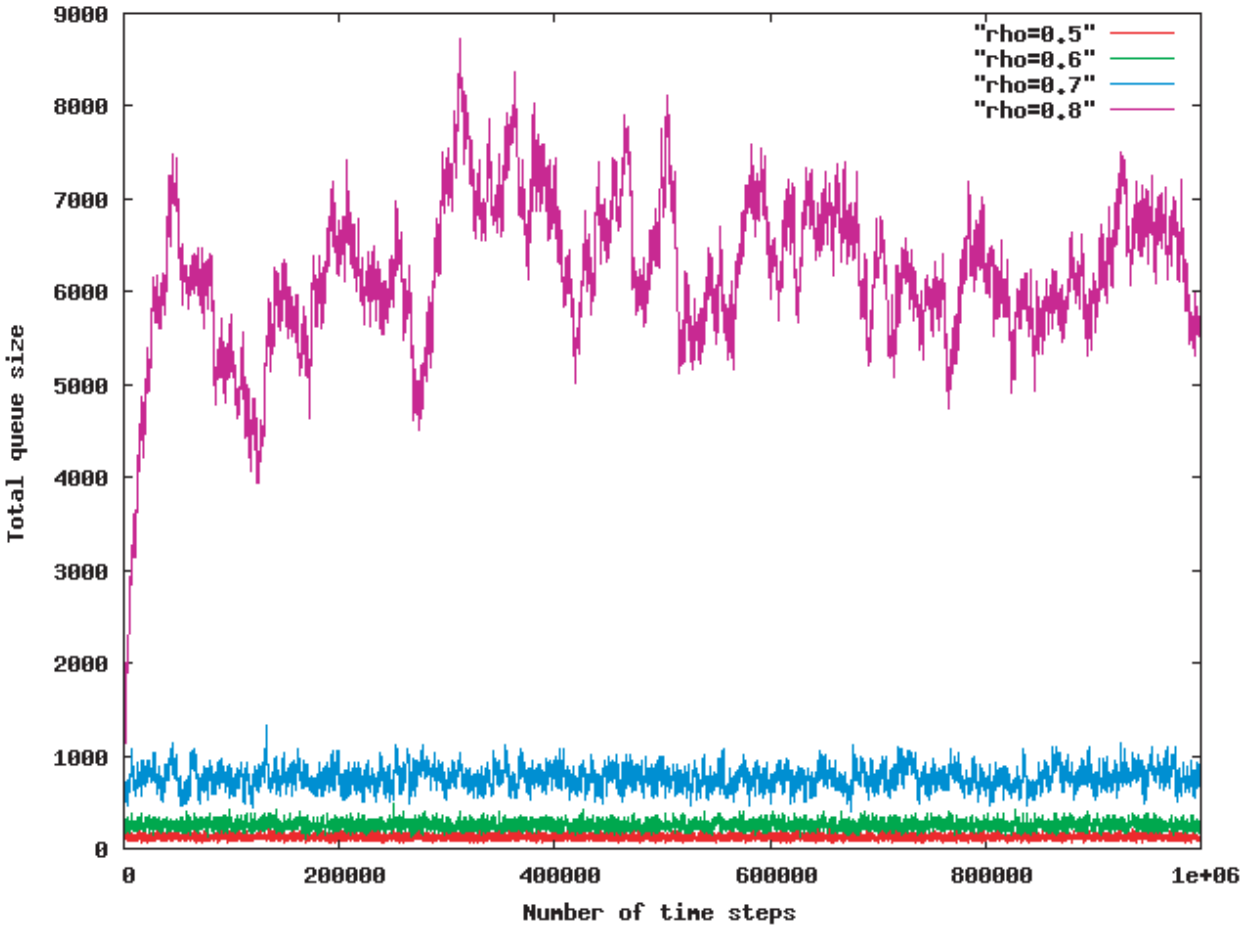
  - Else, do nothing.
- $W_i(t) = \max\{f(Q_i(t)), \varepsilon f(\widehat{Q}_{\max,i}(t))\}$ for small constant $\varepsilon > 0$.
  - $\widehat{Q}_{\max,i}(t)$ is an estimation of $Q_{\max}(t)$ at node i and time t.
  - However, we believe that this modification is not necessary.

Discussion 1 - $\log \log x$ is the best?

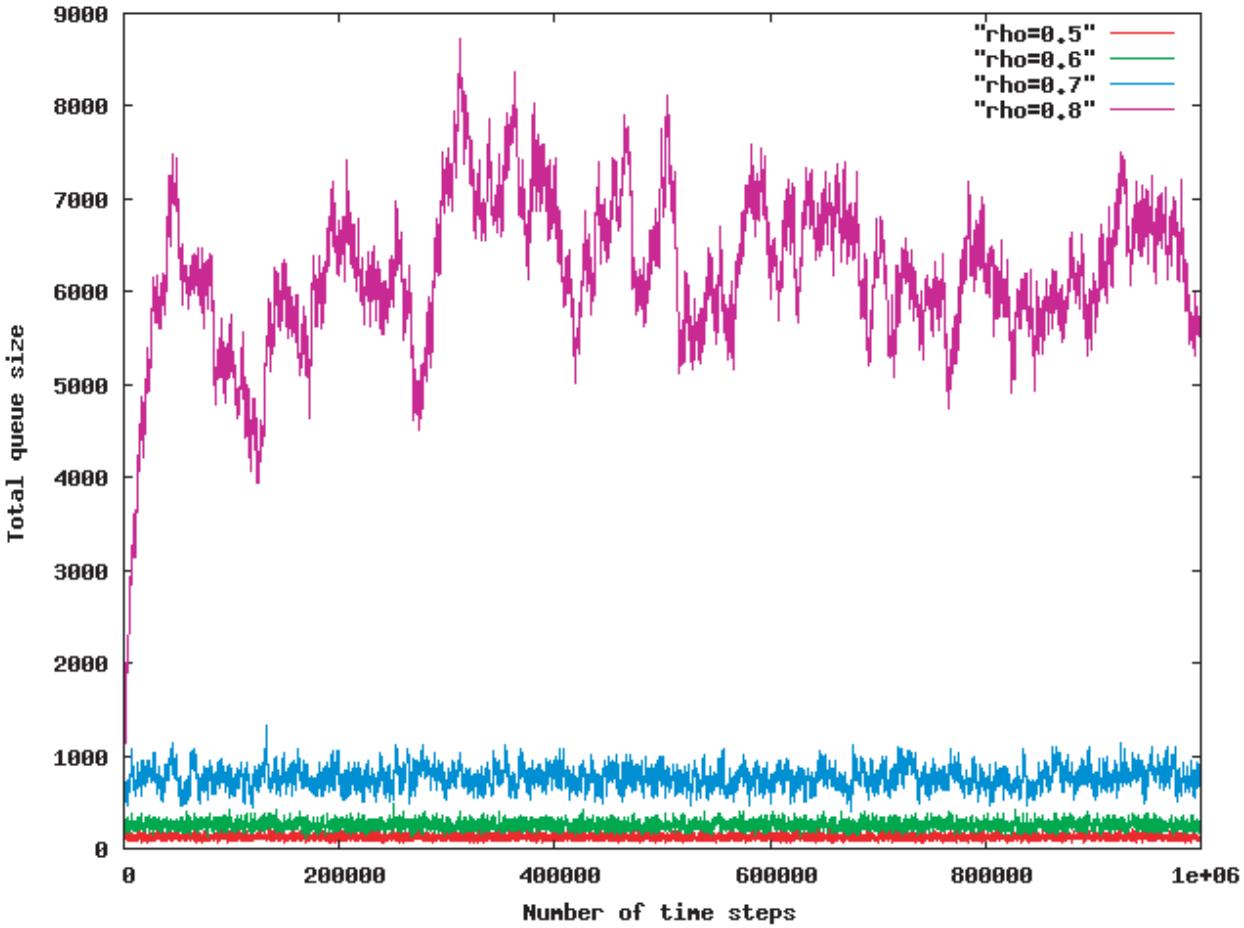


- Slower increasing function gives more stability, but higher delay.

Discussion 2 - Modification is necessary?

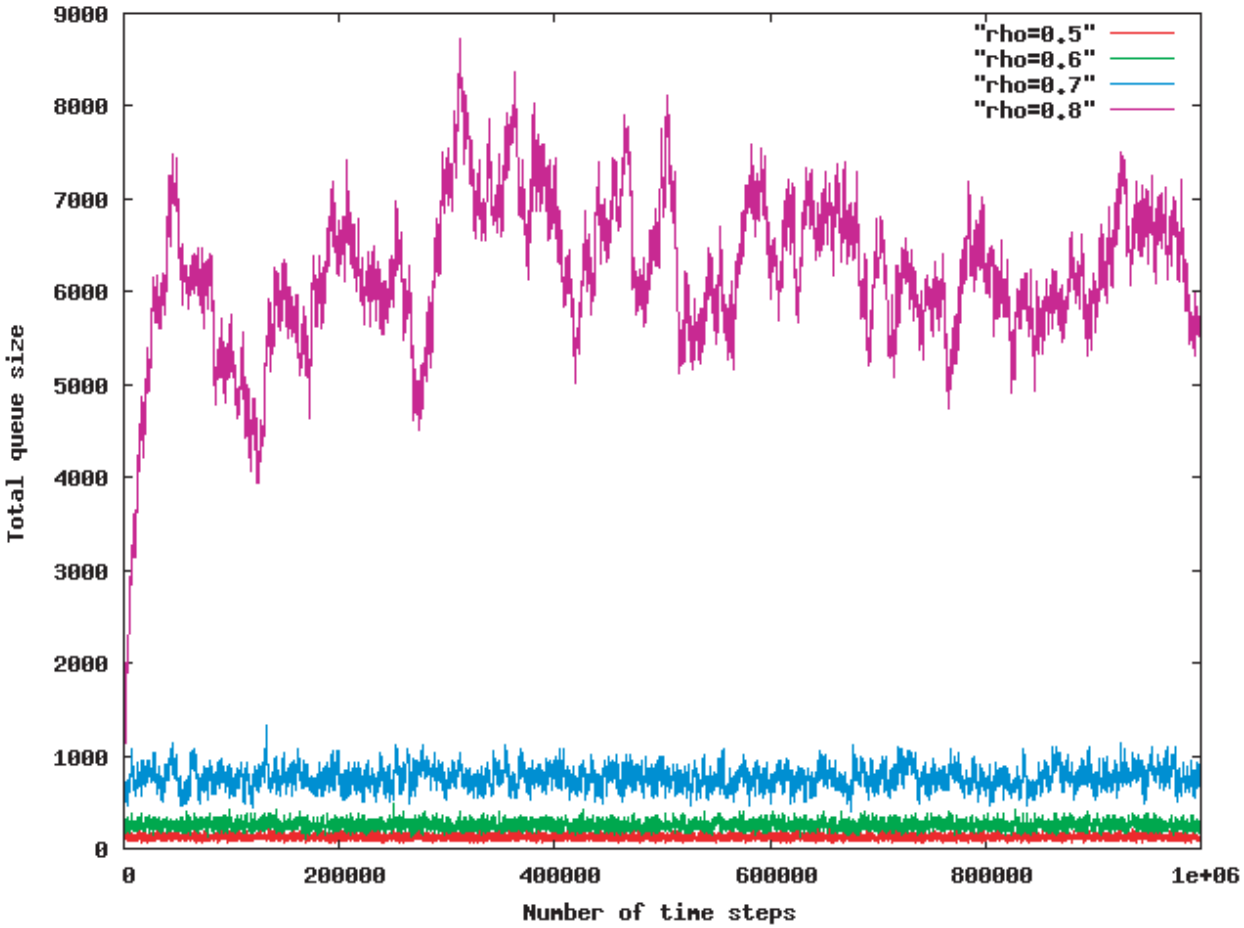


Discussion 2 - Modification is necessary?



- CSMA-based algorithm without any information-exchange can be possible to be throughput-optimal?

Discussion 2 - Modification is necessary?



- CSMA-based algorithm without any information-exchange can be possible to be throughput-optimal?
  - Yes [Jiang, Shah, Shin and Walrand 09]

Discussion 3 - Without Perfect Carrier Sensing?

- Recall PCS Assumption: Each node can sense instantly whether the medium is free.
  - Collisions cannot occur.

- Relax this assumption [Ni and Srikant 09]
  - Does not assume PCS.
  - Slotted the discrete time into a control slot and a data slot.
  - Exchange control messages for avoiding collisions in a data slot.
  - However, collisions are allowed in a control slot.

Summary

- Contention Resolution
  - Key algorithmic problem in the communication network
  - Requires a simple, distributed and efficient algorithm

- We provide such an algorithm
  - Essentially, it is a time-varying 'Glauber dynamics'.
  - Key is to guarantee $\mu(t) \approx \pi(t)$.
    - We choose the right weight function $f(x) = \log \log x$.
    - We call it *Network Adiabatic Theorem*.
    - Our algorithm essentially simulates the f-MW algorithm.
  - Generic
    - We consider the wireless network
    - Our algorithm is extendable to other networks
    - For example, the optical network [Shah and Shin 09]
  - Intimate relation between distributed algorithms and reversible networks

Summary

- Contention Resolution
  - Key algorithmic problem in the communication network
  - Requires a simple, distributed and efficient algorithm
- We provide such an algorithm
  - Essentially, it is a time-varying 'Glauber dynamics'.
  - Key is to guarantee $\mu(t) \approx \pi(t)$.
    - We choose the right weight function $f(x) = \log \log x$.
    - We call it *Network Adiabatic Theorem*.
    - Our algorithm essentially simulates the f-MW algorithm.
  - Generic
    - We consider the wireless network
    - Our algorithm is extendable to other networks
    - For example, the optical network [Shah and Shin 09]
  - Intimate relation between distributed algorithms and reversible networks
- Thank you!