Network algorithms made distributed

Devavrat Shah Jinwoo Shin

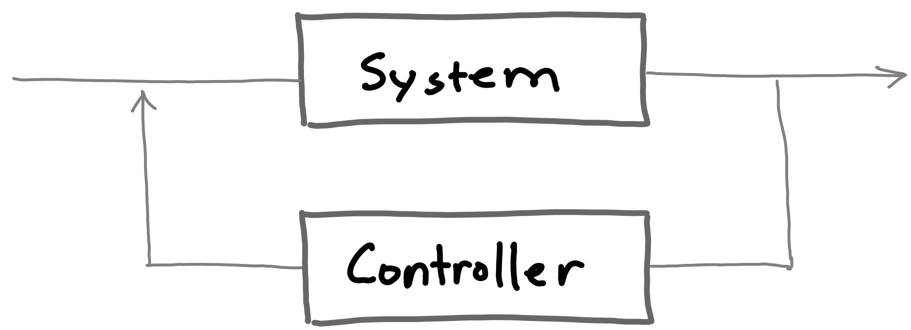
Laboratory for Information and Decision Systems

Massachusetts Institute of Technology

Communication Networks

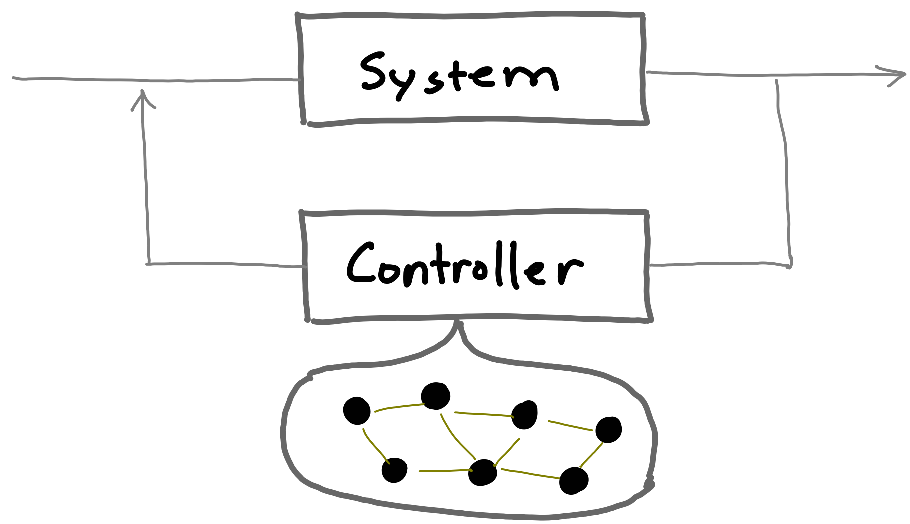
- Network algorithms
  - ○ Required for efficient network resource allocation
    - – that is, they must be *high-performance*
  - ○ Required to operate with system constraints
    - – that is, they should be *implementable*
- Primary challenge
  - ○ Resolution of tension: performance vs. implementation
  - ○ Ideally, implementable without loss of performance
- This talk
  - ○ A method to address this challenge

A bit of Philosophy



- System design and control
  - Generic controller *observes* system parameters
    - based on which it computes control
    - usually, corresponds to solving an optimization problem

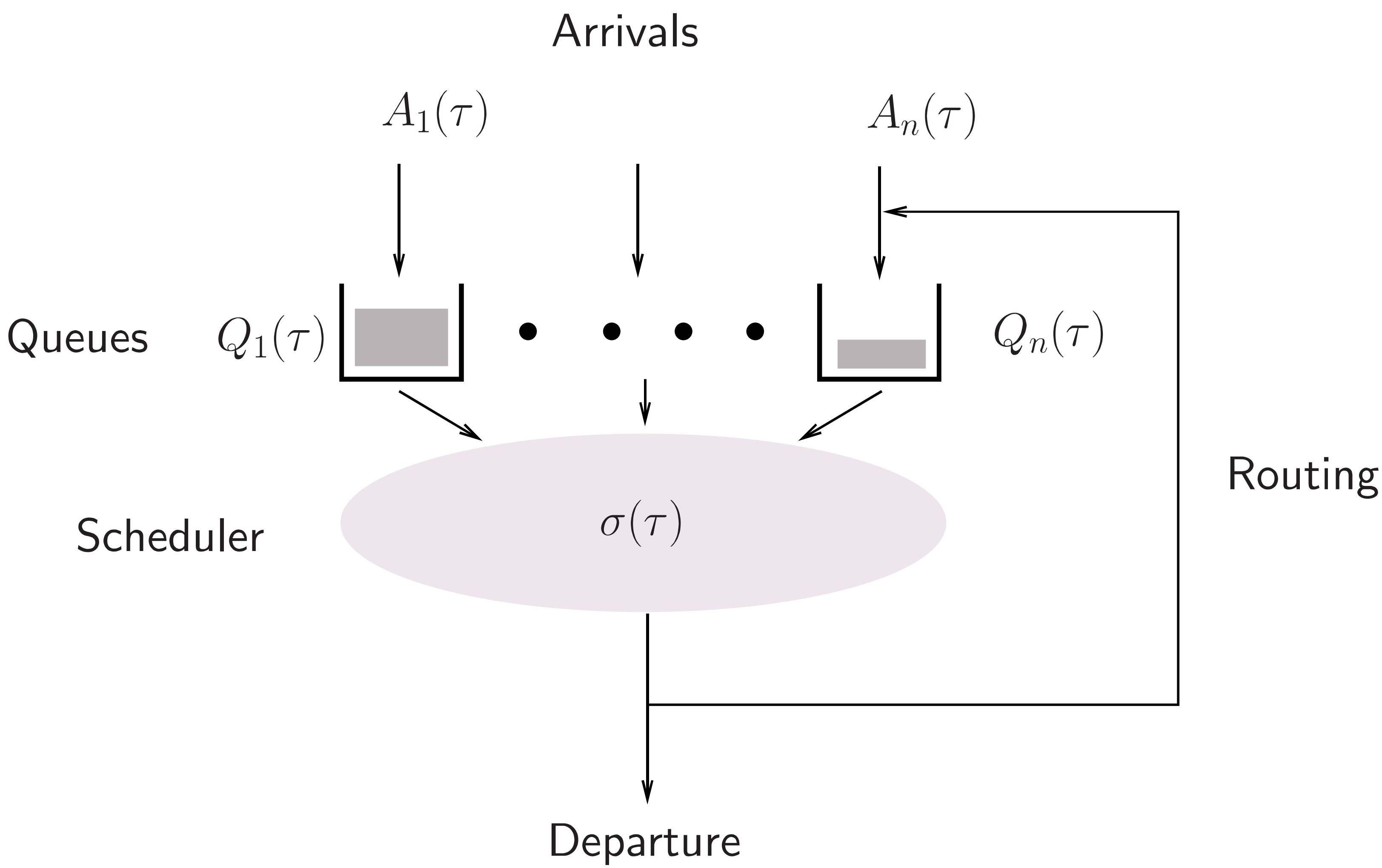
A bit of Philosophy



- In a networked system, control is distributed at nodes
  - ○ Requires network nodes to solve a global optimization
    - – usually, by means of iterative algorithm
  - ○ Usual analytic limitation
    - – algorithm and system operate at same time scale
    - – but design assumes 'time scale' separation

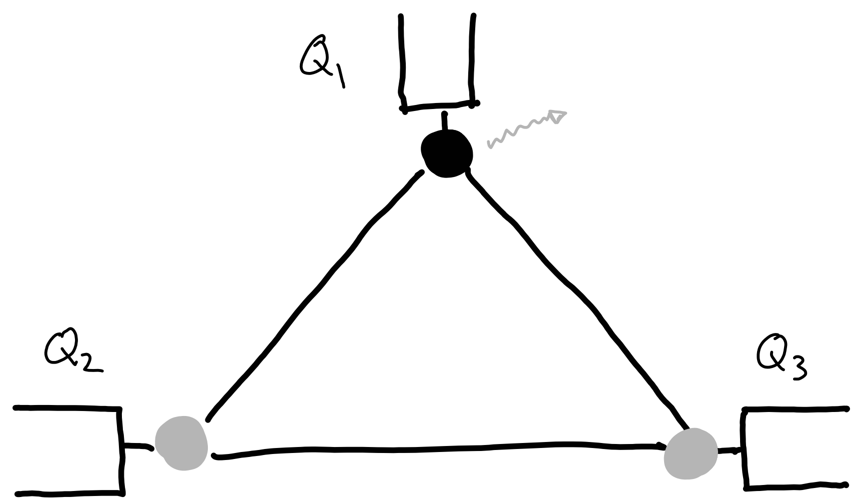
Generic Resource Allocation: Packet-level Model

- A network of n queues



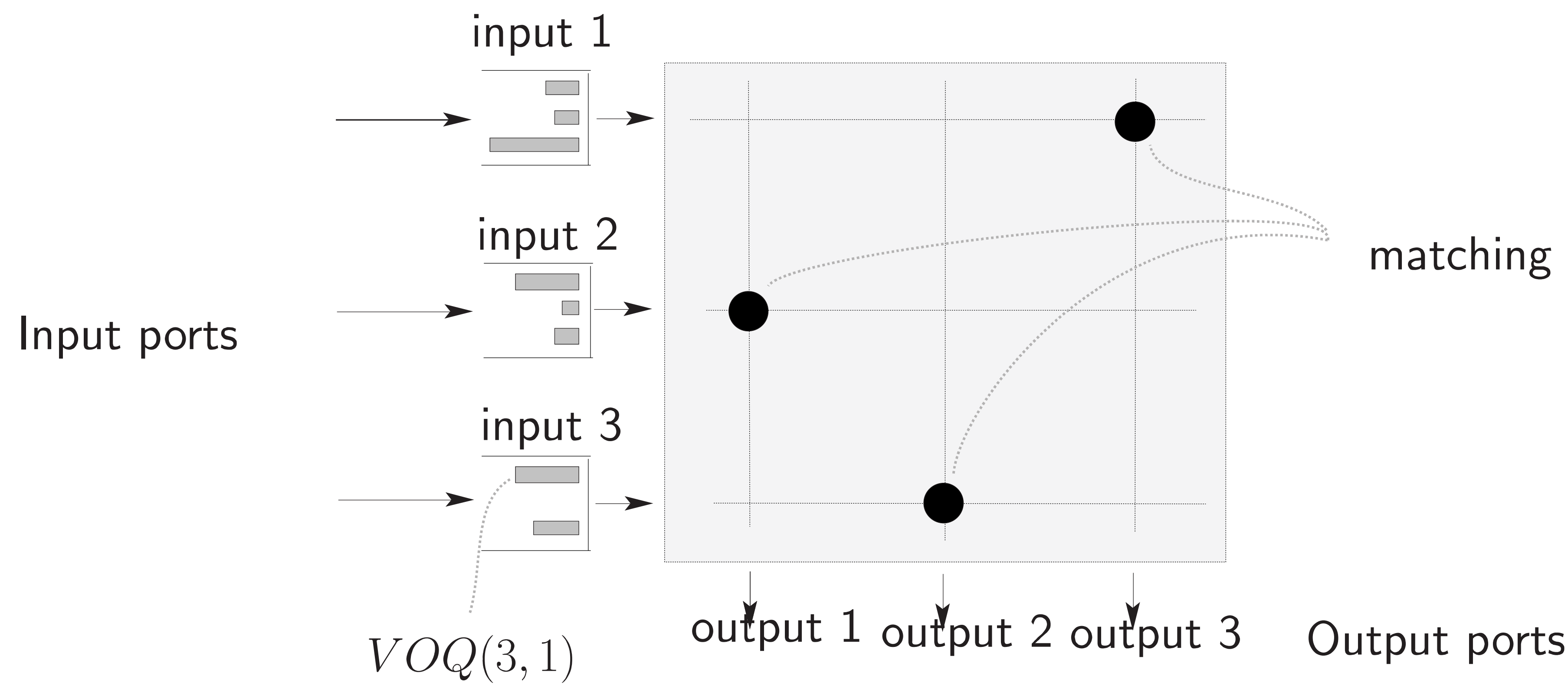
Generic Resource Allocation: Packet-level Model

- Example 1: wireless network



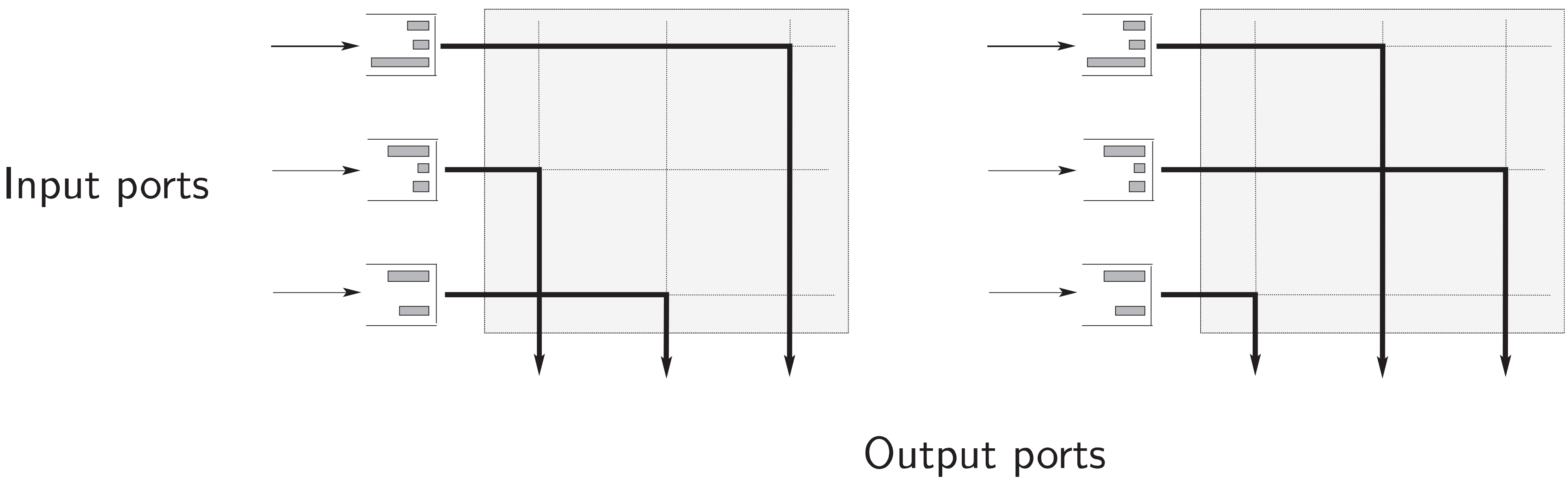
Generic Resource Allocation: Packet-level Model

- Example 2: switch in an Internet router



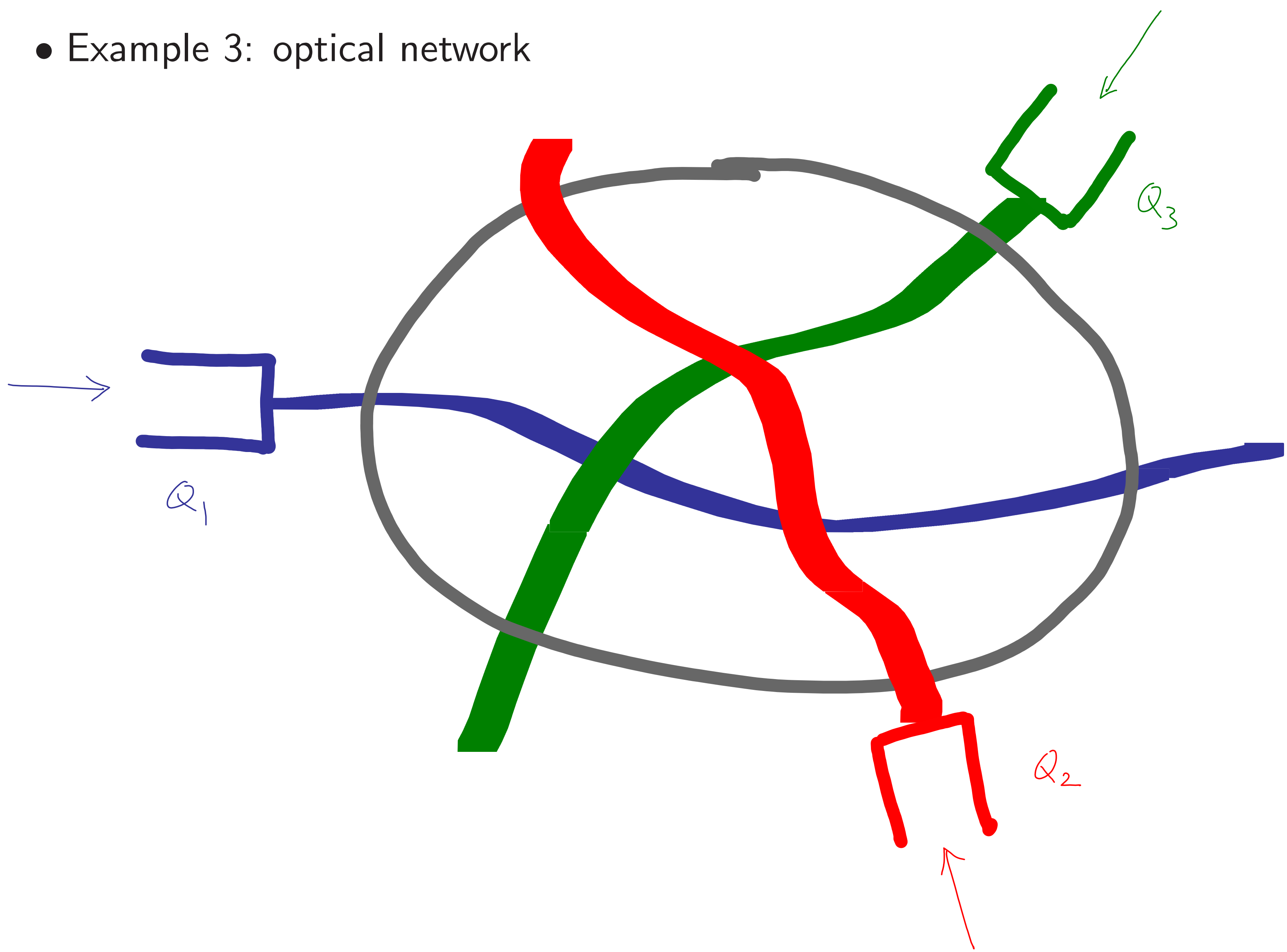
Generic Resource Allocation: Packet-level Model

- Example 2: switch in an Internet router



Generic Resource Allocation: Packet-level Model

- Example 3: optical network



Generic Resource Allocation: Packet-level Model

- Performance metric: throughput
  - ◦ Λ be the set of all arrival rates $\lambda = [\lambda_i]$ s.t.
    - – there exists an algorithm that can keep queues finite under λ
  - ◦ An algorithm is throughput optimal, if
    - – it keeps queues finite for any $\lambda \in \Lambda$
- Key question
  - ◦ Design of throughput-optimal algorithm that is
    - – implementable

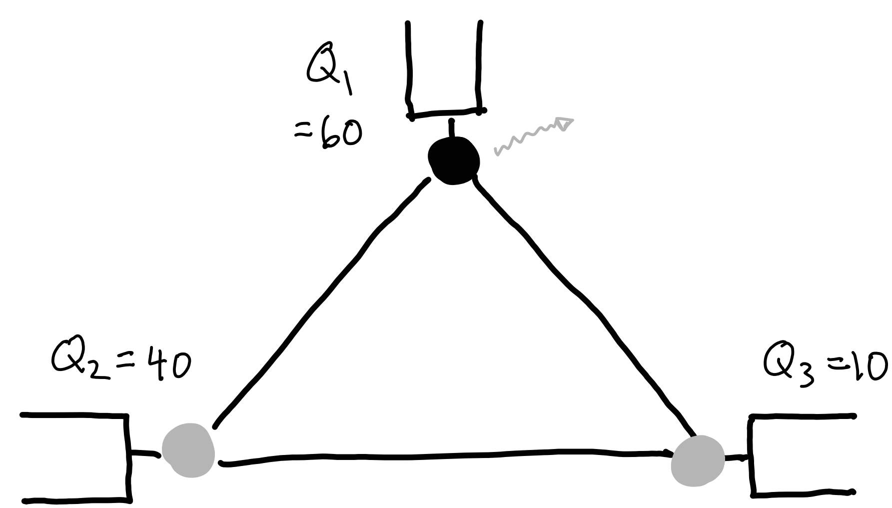
Generic Resource Allocation: Packet-level Model

- Implementable algorithm
  - Simple: few logical operation
    - because, in a switch packets arrive roughly 10ns apart
  - Distributed: little or no co-ordination
    - because, communication between chips or over a wireless medium is v. expensive or even impossible
  - Low memory-bandwidth: few reads/writes
    - because, memory bandwidth is a bottleneck
- Key question: design of a
  - Throughput-optimal, simple, distributed scheduling algorithm

Brief History

- Known throughput optimal algorithm : maximum weight scheduling
  - ◦ Choose a valid collection of queues to serve so that
    - – sum of the queue-sizes of the served queues is maximized
  - ◦ Due to Tassiulas and Ephremides (1992)
    - – it's variant has good delay/latency properties

      cf. Stolyar (2004), Shah and Wischik (2006, 08, 09)



Brief History

- Known throughput optimal algorithm : maximum weight scheduling
  - ○ Choose a valid collection of queues to serve so that
    - – sum of the queue-sizes of the served queues is maximized
  - ○ Due to Tassiulas and Ephremides (1992)
    - – it's variant has good delay/latency properties
      cf. Stolyar (2004), Shah and Wischik (2006, 08, 09)

- Implementation
  - ○ As is, extremely complex
    - – even if its polynomial time, can be costly
    - – e.g. for a 30-port switch, requires 27000 ops in 10ns !

Brief History

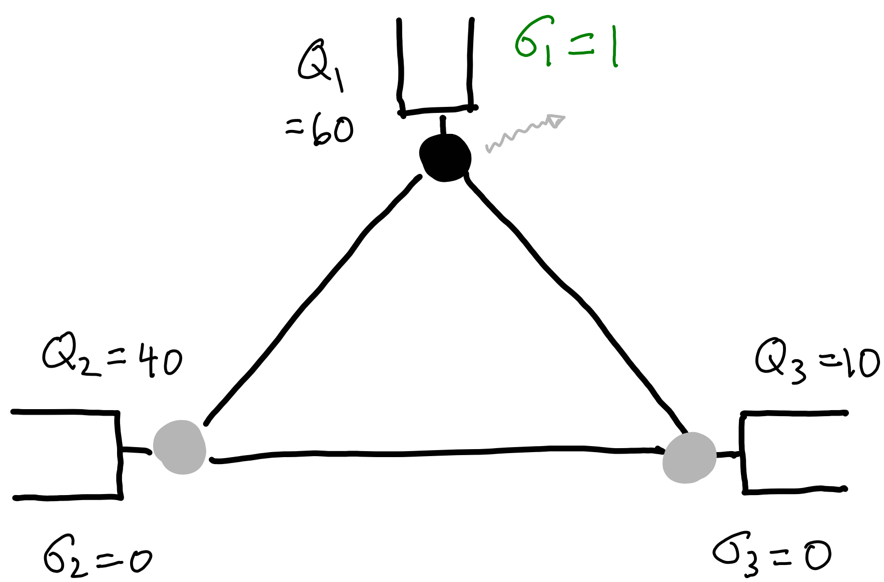
- Known throughput optimal algorithm : maximum weight scheduling
  - Choose a valid collection of queues to serve so that
    - sum of the queue-sizes of the served queues is maximized
  - Due to Tassiulas and Ephremides (1992)
    - it's variant has good delay/latency properties
      cf. Stolyar (2004), Shah and Wischik (2006, 08, 09)
- Implementation
  - As is, extremely complex
  - A lot ($= \infty$) of research, motivated by implementation concern
  - But, all work suffer from one or more of the following
    - too specific an approach
    - generic, but requires global co-ordination (over time)
    - and, lack of elegance/simplicity

Main Result

- A scheduling algorithm, that
  - Is v. simple, elegant and distributed
  - Throughput optimal
  - And, applies to the general problem setup
  - A perfect analogy: Metropolis-Hastings Method
- Distributed means
  - With respect to constraint network graph
  - And, beyond that little or *no* control information need to be exchanged
- Next, for the example of wireless network
  - Description of the algorithm
  - Intuition behind its efficiency

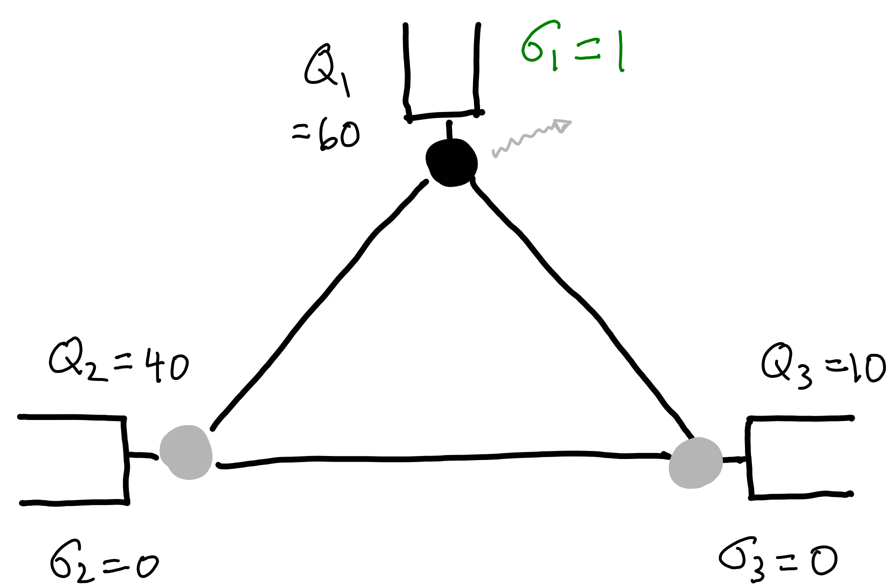
Algorithm

- Given a wireless network of n nodes
  - Network graph $G = (V, E)$
  - Let $\mathbf{Q}(t) = [Q_i(t)]$ be queue-sizes at time t
  - Let $\sigma(t) = [\sigma_i(t)]$ be schedule at time t
    - $\sigma_i(t) \in \{0, 1\}$ for all i,
    - $\sigma_i(t) = 1$ means node i is transmitting, and
    - $\sigma_i(t) + \sigma_j(t) \leq 1$ for all $(i, j) \in E$



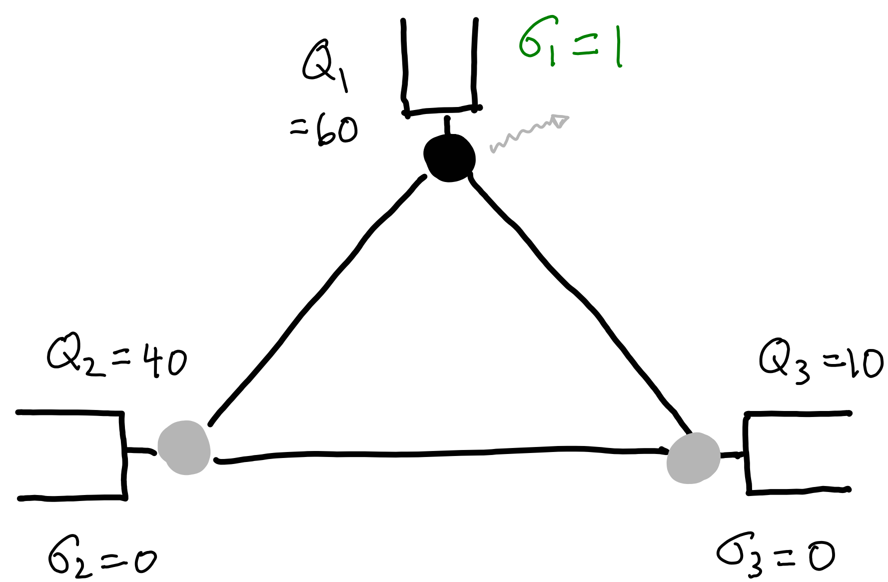
Algorithm

- Given a wireless network of n nodes
  - Network graph $G = (V, E)$
  - Let $\mathbf{Q}(t) = [Q_i(t)]$ be queue-sizes at time t
  - Let $\sigma(t) = [\sigma_i(t)]$ be schedule at time t
    - let $\mathcal{I}(G)$ be set of all feasible schedules
    - and maximum weight schedule is $\arg\max_{\rho \in \mathcal{I}(G)} \sum_i \rho_i Q_i(t)$



Algorithm

- Given a wireless network of n nodes
  - Network graph $G = (V, E)$
  - Let $\mathbf{Q}(t) = [Q_i(t)]$ be queue-sizes at time t
  - Let $\sigma(t) = [\sigma_i(t)]$ be schedule at time t
  - Carrier sense capability
    - if any node i is transmitting, then
    - all of its neighbors can *sense* it



Algorithm

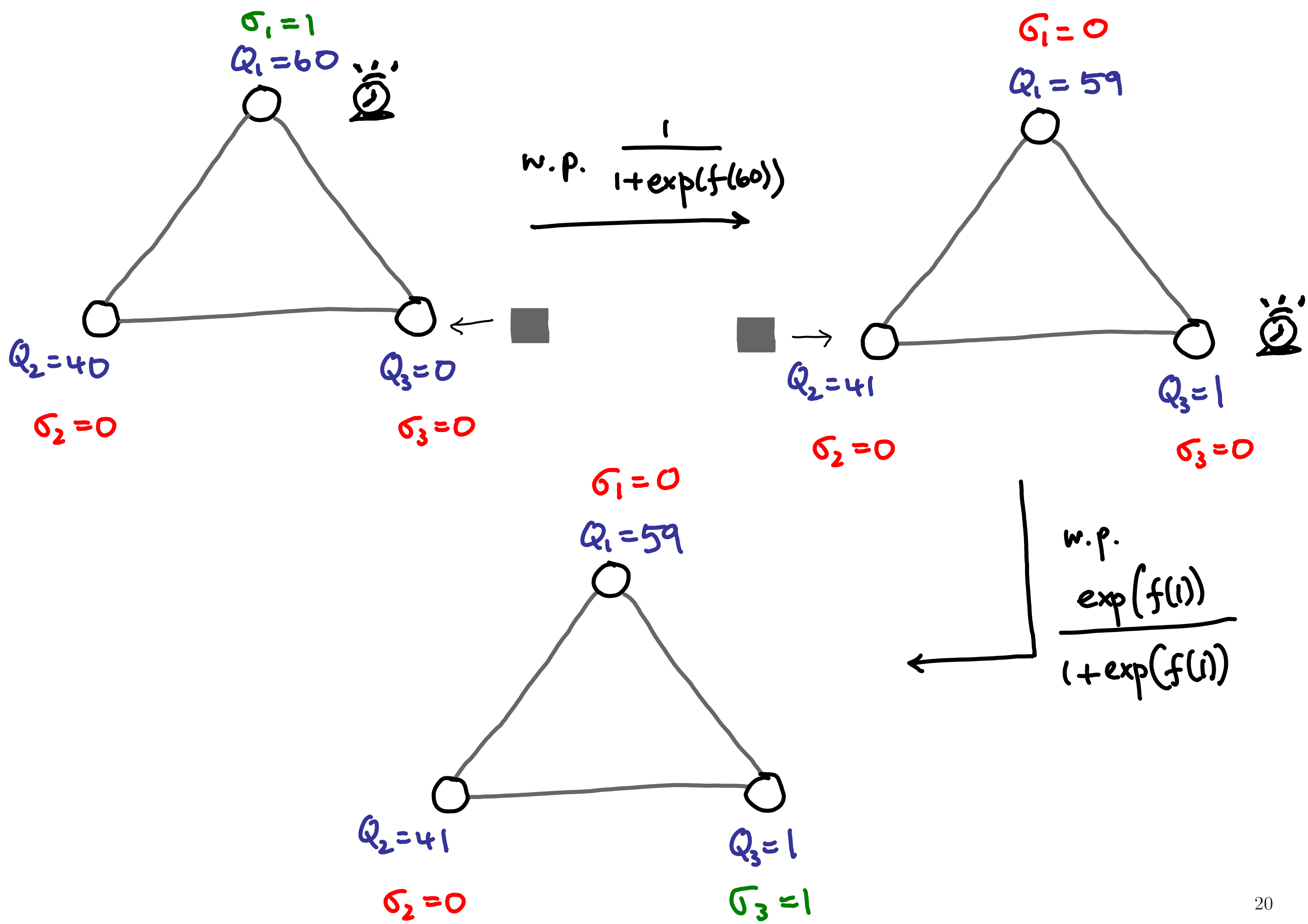
- Each node $i \in V$ has an independent Exponential clock of rate 1
  - When a node, say i's clock ticks, say at time s, do
    - node i checks if medium is FREE or $\sigma_i(s^-) = 1$
    - if yes, then it sets

$$\sigma_i(s^+) = \begin{cases} 1, & \text{w. prob. } \frac{\exp(f(Q_i(s)))}{1+\exp(f(Q_i(s)))} \\ 0, & \text{o.w.} \end{cases}$$

    - else (i.e. medium is BUSY), it sets $\sigma_i(s^+) = 0$ w.p. 1

- An example

Algorithm



$\sigma_1=1$
$Q_1=60$
$Q_2=40$
$Q_3=0$
$\sigma_2=0$
$\sigma_3=0$
w.p. $\frac{1}{1+\exp(f(60))}$
$\sigma_1=0$
$Q_1=59$
$Q_2=41$
$Q_3=1$
$\sigma_2=0$
$\sigma_3=0$
w.p. $\frac{\exp(f(1))}{1+\exp(f(1))}$
$\sigma_1=0$
$Q_1=59$
$Q_2=41$
$Q_3=1$
$\sigma_2=0$
$\sigma_3=1$

Algorithm

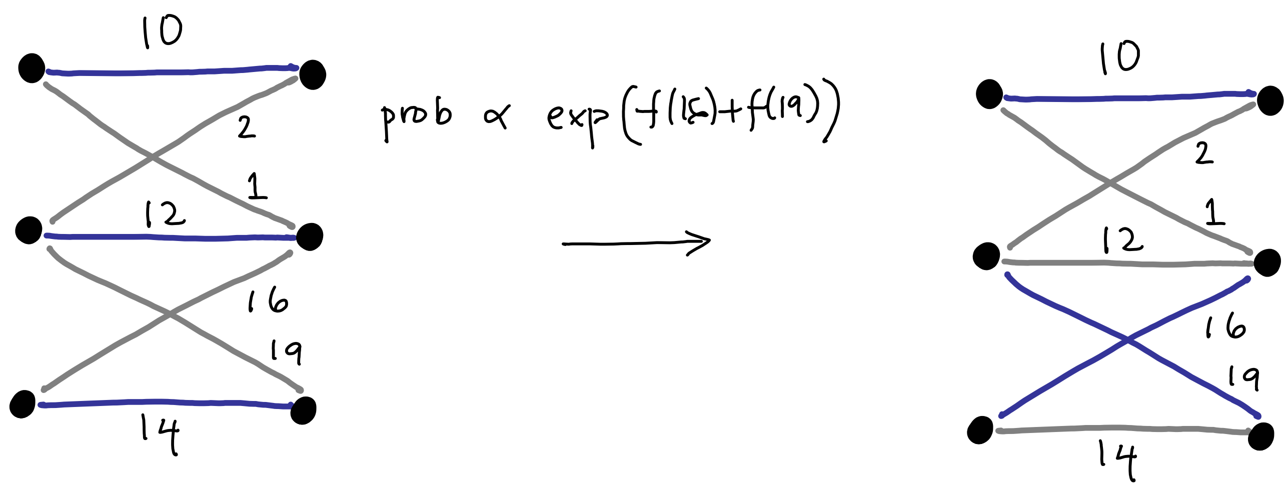
- **Theorem 1.** With the proper choice of f, the algorithm is efficient.

- We'll go through the proof intuition and
  - Search for the proper choice of f

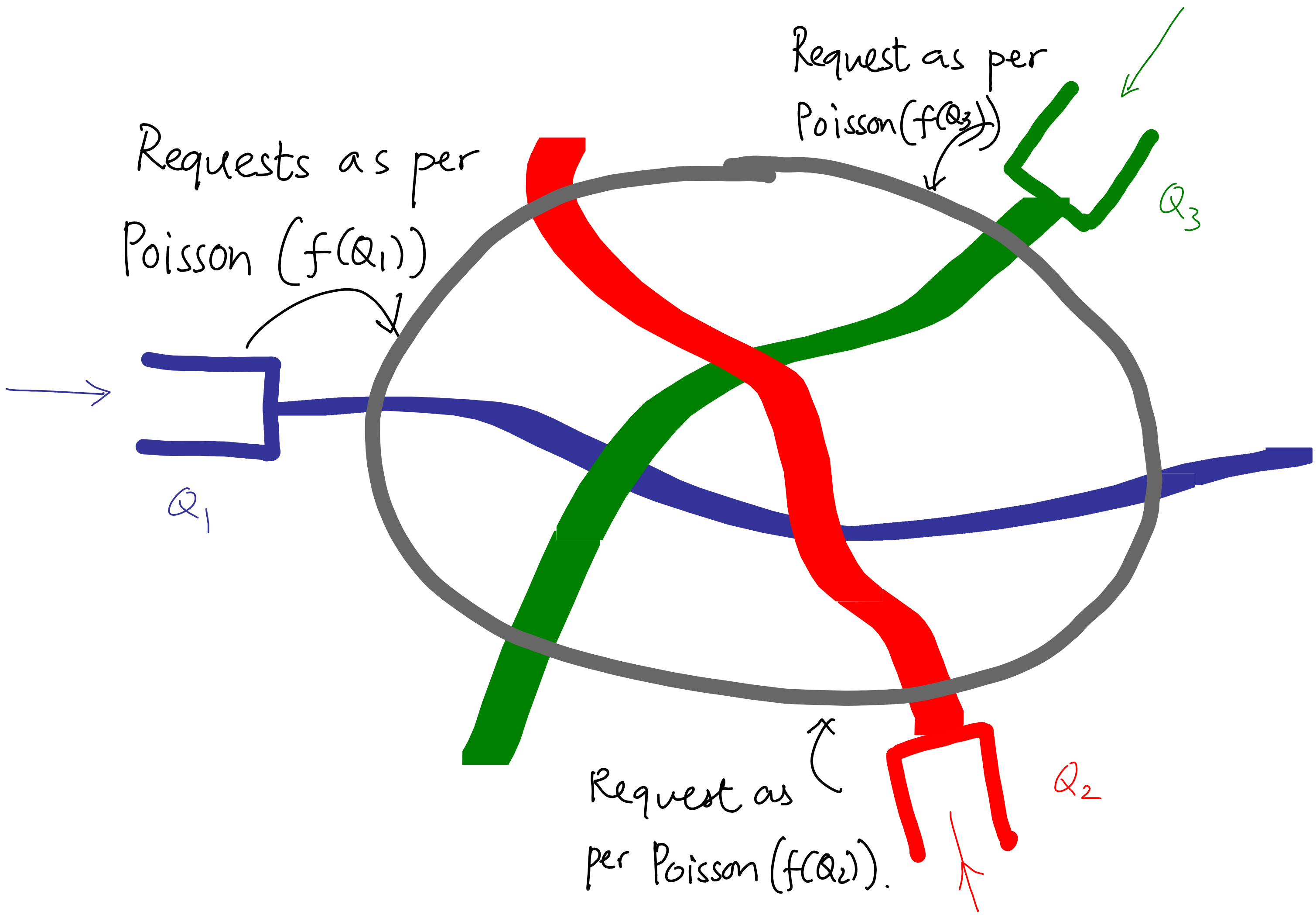
    Essentially, $f(x) = \log\log(x + e)$

- Note that, exactly same algorithmic result holds for the generic model
  - Switch scheduling : a slotted time algorithm
  - Optical core-network scheduling : variable length packets
    - both are throughput optimal, and
    - use appropriate reversible Markov chain on space of schedules
    - next, quick explanation through examples

Algorithm : Switch scheduling



Algorithm : Optical network scheduling



Back to Wireless: Algorithm $\approx$ Glauber dynamics

- Assume: $\mathbf{Q}(t) = \mathbf{Q}$ is *fixed*

- Each node $i \in V$ has an Exponential clock of rate 1
  - ○ When a node, say i's clock ticks, say at time s
    - – node i checks if medium is FREE or $\sigma_i(s^-) = 1$
    - – if yes, then it sets

$$\sigma_i(s^+) = \begin{cases} 1, & \text{w. prob. } \frac{\exp(f(Q_i))}{1+\exp(f(Q_i))} \\ 0, & \text{o.w.} \end{cases}$$

    - – else (i.e. medium is BUSY), it sets $\sigma_i(s^+) = 0$ w.p. 1

Glauber dynamics

- Assume: $\mathbf{Q}(t) = \mathbf{Q}$ is *fixed*
- Glauber dynamics
  - ◦ Induces irreducible, finite state reversible Markov chain on $\mathcal{I}(G)$
  - ◦ Has a unique stationary distribution, say $\pi_{\mathbf{Q}}$
- **Lemma 1.** The $\pi_{\mathbf{Q}}$ has the following properties:

  1. $\pi_{\mathbf{Q}}(\sigma) \propto \exp(\sum_i \sigma_i f(Q_i))$ for every $\sigma \in \mathcal{I}(G)$, and

Glauber dynamics

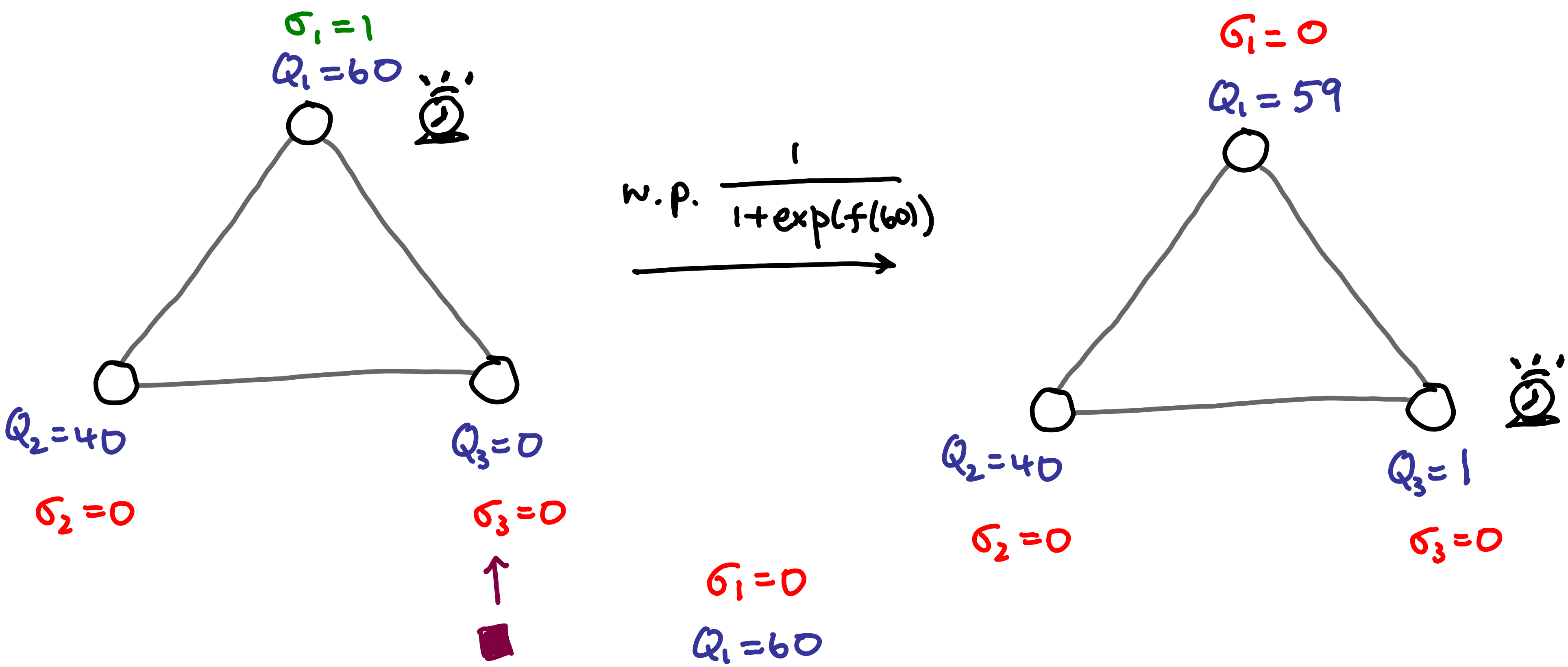
- Assume: $\mathbf{Q}(t) = \mathbf{Q}$ is *fixed*
- Glauber dynamics
  - ∘ Induces irreducible, finite state reversible Markov chain on $\mathcal{I}(G)$
  - ∘ Has a unique stationary distribution, say $\pi_{\mathbf{Q}}$
- **Lemma 1.** The $\pi_{\mathbf{Q}}$ has the following properties:
  1. $\pi_{\mathbf{Q}}(\sigma) \propto \exp(\sum_i \sigma_i f(Q_i))$ for every $\sigma \in \mathcal{I}(G)$, and
  2. $\mathbb{E}_{\pi_{\mathbf{Q}}}[\sum_i \sigma_i f(Q_i)] \geq \left(\max_{\rho \in \mathcal{I}(G)} \sum_i \rho_i f(Q_i)\right) - n$.

→ If $\mathbf{Q}(t)$ *fixed*, then Glauber dynamics chooses $\sigma(t)$ s.t.

- ∘ It is essentially maximum weight w.r.t. $f(Q)$
- ∘ Which is efficient for any increasing f
  - – e.g. $f(x) = x,\ x^\alpha,\ \log(x+1),\ \log\log(x+e)$, ...

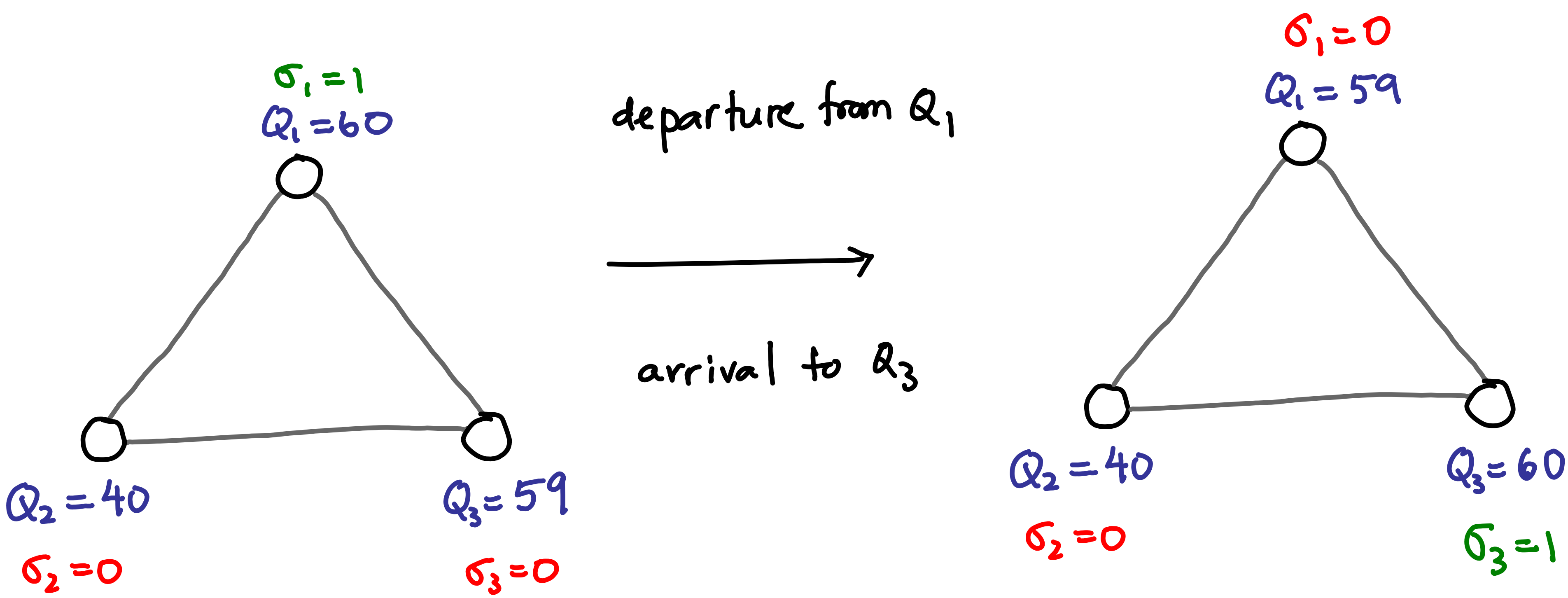
Glauber dynamics

- Assuming $\mathbf{Q}(t) = \mathbf{Q}$ *fixed*,
  - Glauber dynamics leads to throughput optimal performance
  - But, the fact ramains – queues *do* change
- For example



Glauber dynamics

- Assuming $\mathbf{Q}(t) = \mathbf{Q}$ *fixed*,
  - ○ Glauber dynamics leads to throughput optimal performance
  - ○ But, the fact remains – queues *do* change
    - – they change essentially at unit rate
    - – and can severely affect the performance



Glauber dynamics: Cost of change

- If $\mathbf{Q}(t)$ does not change,
  - Glauber dynamics leads to throughput optimal performance

- But $\mathbf{Q}(t)$ changes and want to find its "cost"
  - Let $\Delta\mathbf{Q}(t)$ be change in $\mathbf{Q}(t)$ in unit time
  - Let $\mathrm{T}_{\mathsf{mix}}(n, f(\mathbf{Q}(t)))$ be the *mixing time* of Glauber dynamics
    - time to reach stationary distribution with f ($\mathbf{Q}(t)$ fixed)

- Key analytic result: for algorithm with given f
  - The cost of change $\Delta\mathbf{Q}(t)$ is (of the order of)
    - $C_f(n, \mathbf{Q}(t)) \sim \mathrm{T}_{\mathsf{mix}}(n, f(\mathbf{Q}(t)))f'(\mathbf{Q}(t))\Delta\mathbf{Q}(t)$ time steps

Glauber dynamics: Cost of change

- If $\mathbf{Q}(t)$ does not change,
  - ∘ Glauber dynamics leads to throughput optimal performance
- Accounting for change at unit rate in $\mathbf{Q}(t)$: $\Delta\mathbf{Q}(t) = 1$
  - ∘ A general bound:
    - $\mathrm{T}_{\mathsf{mix}}(n, f(\mathbf{Q}(t))) \sim \exp\left(\phi_n |f(\mathbf{Q}(t))|\right)$
    - where ϕ_n depends on $n = |V|$
  - ∘ Cost in terms of time-steps of algorithm

$$\begin{aligned} C_f(n, \mathbf{Q}(t)) &\sim \exp\left(\phi_n \, |f(\mathbf{Q}(t))|\right) \cdot f'(\mathbf{Q}(t))\Delta\mathbf{Q}(t) \\ &\sim \exp\left(\phi_n \, |f(\mathbf{Q}(t))|\right) f'(\mathbf{Q}(t)). \end{aligned}$$

- For algorithm to be stable, we need cost small (< 1)
  - ∘ Let us check different f

Glauber dynamics

- Cost of change for weight function f

$$C_f(n, \mathbf{Q}(t)) \sim \exp\left(\phi_n |f(\mathbf{Q}(t)|\right) \cdot |f'(\mathbf{Q}(t))|.$$

  - Ideally, we wish to have it really small

- Consider "costs" for various f

  - Recall, for $f(x) = x$

$$C_f(n, \mathbf{Q}(t)) \sim \exp\left(\phi_n |\mathbf{Q}(t)|\right).$$

$\rightarrow$ Too large !

Glauber dynamics

- Cost of change for weight function f

$$C_f(n, \mathbf{Q}(t)) \sim \exp\left(\phi_n |f(\mathbf{Q}(t)|\right) \cdot |f'(\mathbf{Q}(t))|.$$

  - Ideally, we wish to have it really small

- Consider "costs" for various f
  - For $f(x) = \log x$

$$\begin{aligned} C_f(n, \mathbf{Q}(t)) &\sim \exp\left(\phi_n \log \mathbf{Q}(t)\right) \cdot \frac{1}{\mathbf{Q}(t)} \\ &\sim \text{poly}(\mathbf{Q}(t)). \end{aligned}$$

$\rightarrow$ Still, too large !

Glauber dynamics

- Cost of change for weight function f

$$C_f(n, \mathbf{Q}(t)) \sim \exp\left(\phi_n |f(\mathbf{Q}(t)|\right) \cdot |f'(\mathbf{Q}(t))|.$$

  - Ideally, we wish to have it really small

- Consider "costs" for various f

  - For $f(x) = \log\log x$

$$\begin{aligned} C_f(n, \mathbf{Q}(t)) &\sim \exp\left(\phi_n \log\log \mathbf{Q}(t)\right) \cdot \frac{1}{\mathbf{Q}(t)\log \mathbf{Q}(t)} \\ &\sim \frac{\text{poly}(\log \mathbf{Q}(t))}{\mathbf{Q}(t)} \\ &\sim \frac{1}{\mathbf{Q}(t)}. \end{aligned}$$

$\rightarrow$ Goes to 0 as $\mathbf{Q}(t)$ goes to ∞

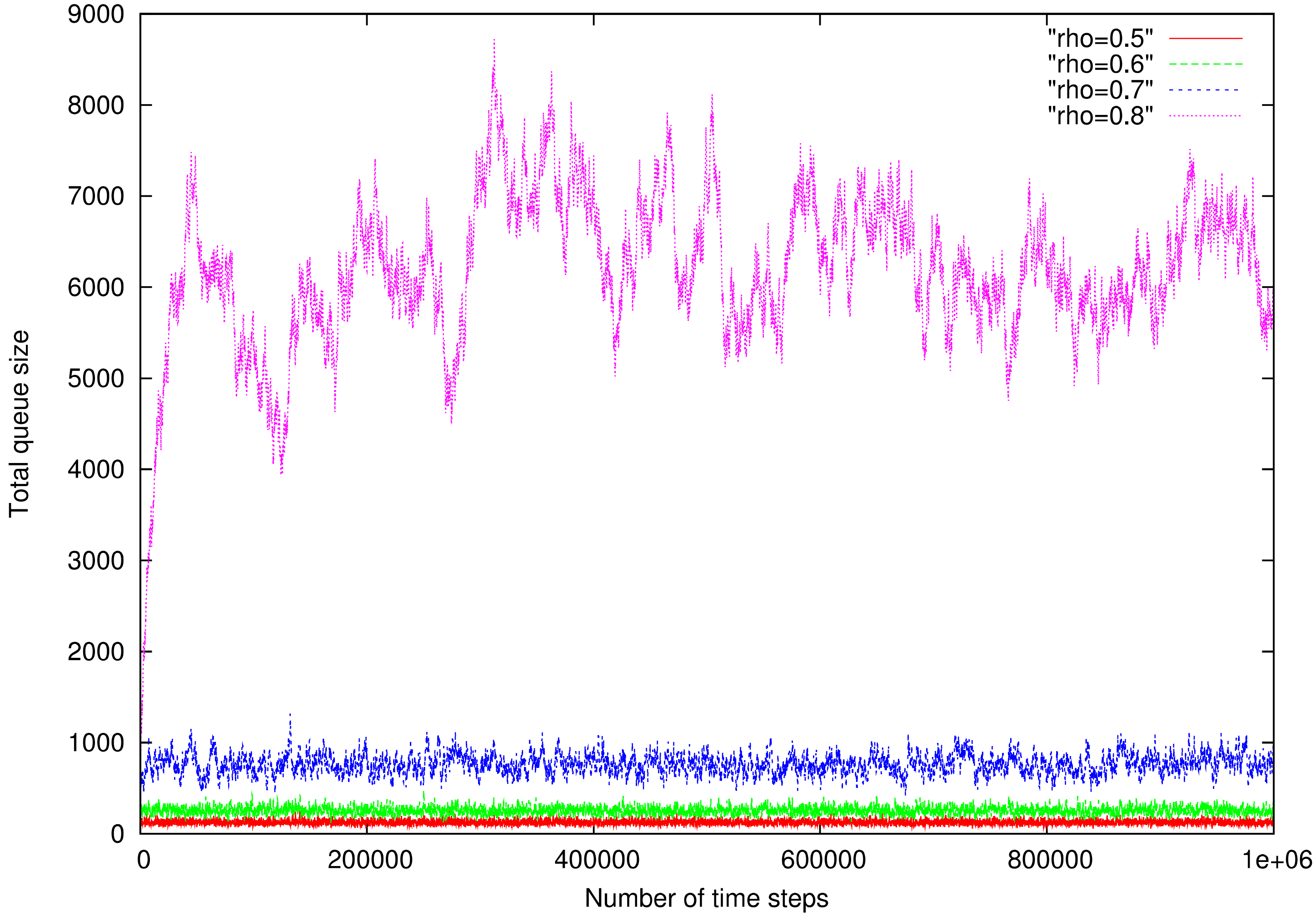
Glauber dynamics

- Thus, for weight function $f(x) = \log \log(x + e)$
  - The cost of change becomes negligible if $\mathbf{Q}(t)$ grows
  - Hence system can operate with one-step Glauber dynamics
    - and provide efficient performance !
- **Theorem 2.** Under the Glauber dynamics based algorithm* with weight function $f(x) = \log \log(x + e)$, the resulting network Markov process is positive Harris recurrent for $\lambda \in \Lambda$.

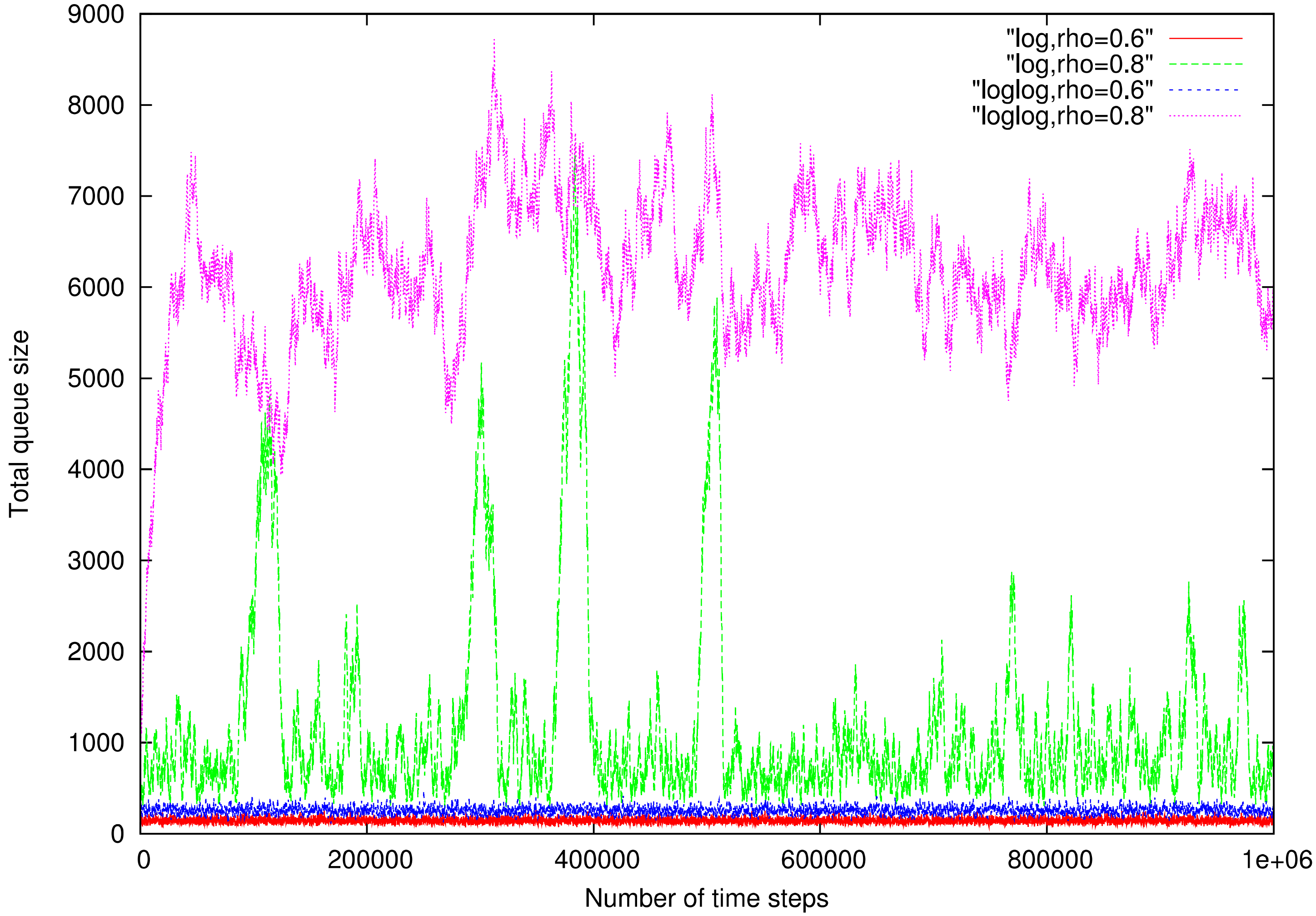
Glauber dynamics

- **Theorem 2.** Under the Glauber dynamics based algorithm* with weight function $f(x) = \log\log(x+e)$, the resulting network Markov process is positive Harris recurrent for $\lambda \in \Lambda$.
- Here, $\star$ means a minor modification of weight $f(Q_i(t))$
  - ◦ Using estimate of $Q_{\text{max}}(t) = \max_k Q_k(t)$ along with $Q_i(t)$
    - – requires minimal global information
  - ◦ Or, a *learning* based mechanism is needed
    - – Algorithm by Jiang and Walrand (2008)
    - – Positive recurrence in Jiang, Shah, Shin and Walrand (2009)
  - ◦ We *strongly* believe that
    - – the 'vanilla' algorithm works
- Adaptation to the slotted time setup
  - ◦ A clever trick introduced by Ni and Srikant (2009)

Delay or Queue-size: $\log\log$ weight



Delay or Queue-size: $\log \log$ vs. $\log$ weight



Delay

- Latency or delay or queue-size of algorithm depends on mixing time
  - For wireless network (independent set)
    - it scales like : $\exp(n \log n)$
  - For switch (matching)
    - it scales like : poly(n)
  - For general network, can not expect delay polynomial in n
    - unless, $P = NP$ (or $NP \subset BPP$)
      cf. Shah, Tse and Tsitsiklis (2009)
  - However, practical network topologies are not 'adversarial'
    - they posses 'geometry'

Delay

- Algorithm for networks with geometry
  - ◦ Those with *polynomial* growth

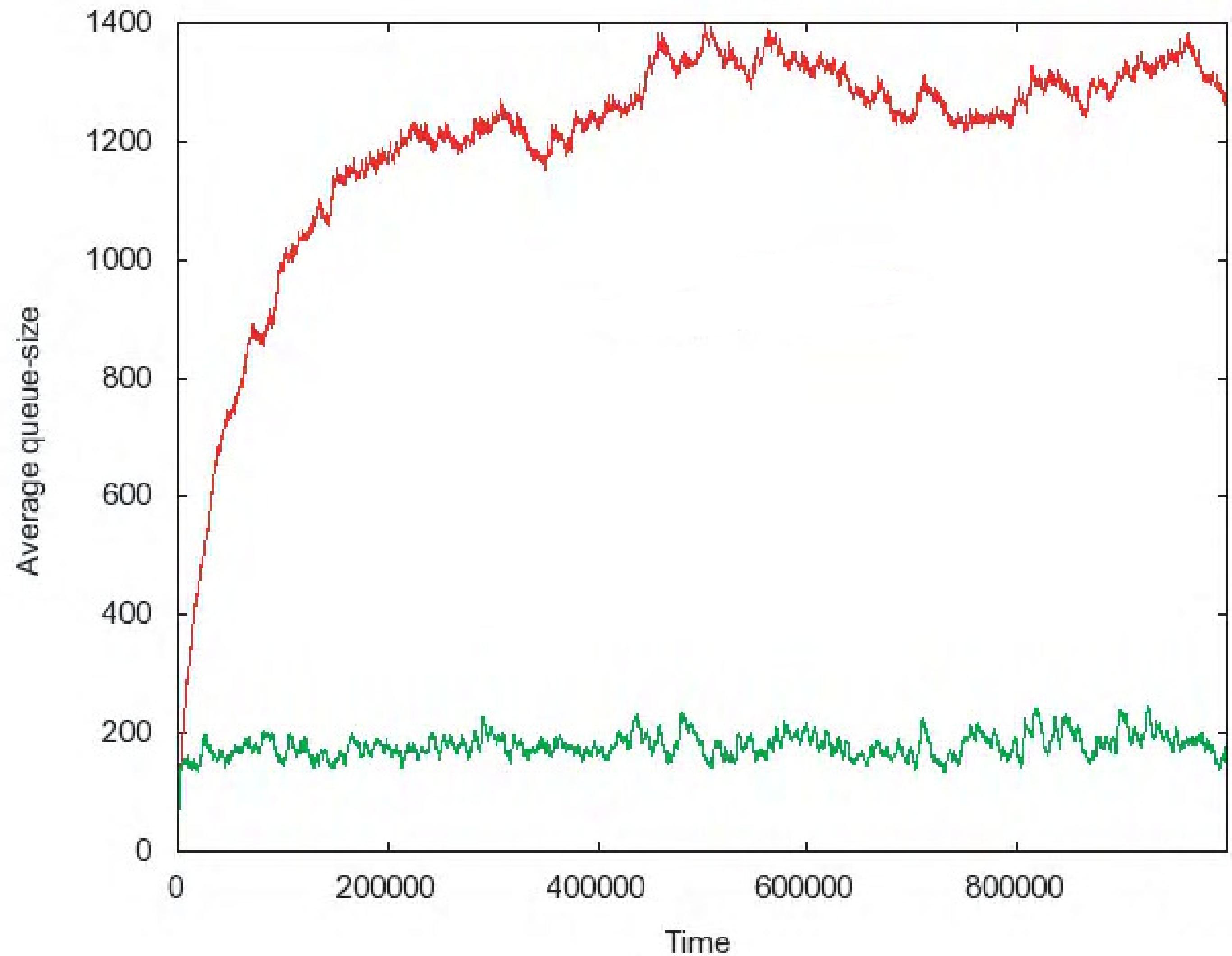
- Key insights
  - ◦ Geometry leads to decomposability of scheduling
    - – possible to solve global problem locally
  - ◦ Algorithm described could help solve problem locally
    - – issue is unboundedness of $\log\log$ weight

Delay

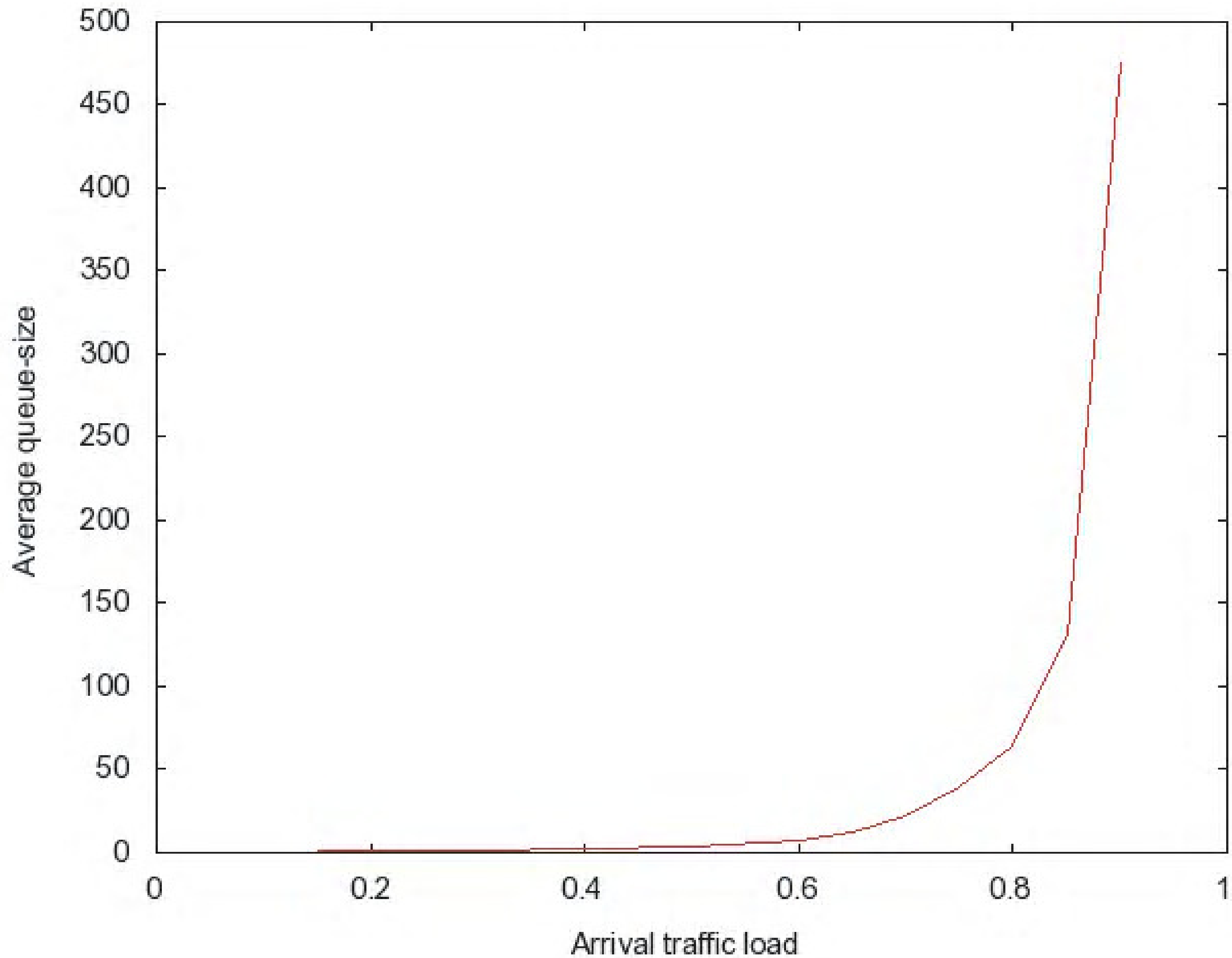
- Algorithm for networks with geometry
  - Those with *polynomial* growth

- Description of algorithm: at a high-level
  - At a given time, a tiny fraction of nodes are *frozen*
    - decided through a randomized local, distributed algo.
  - A *frozen* node
    - maintains its state and does not change it
  - A node, that is not *frozen*
    - executes queue-based CSMA described earlier
    - but with weight Q_i/Q_i^*
    - where Q_i^* is max of q-size in ‘local’ neighborhood of i

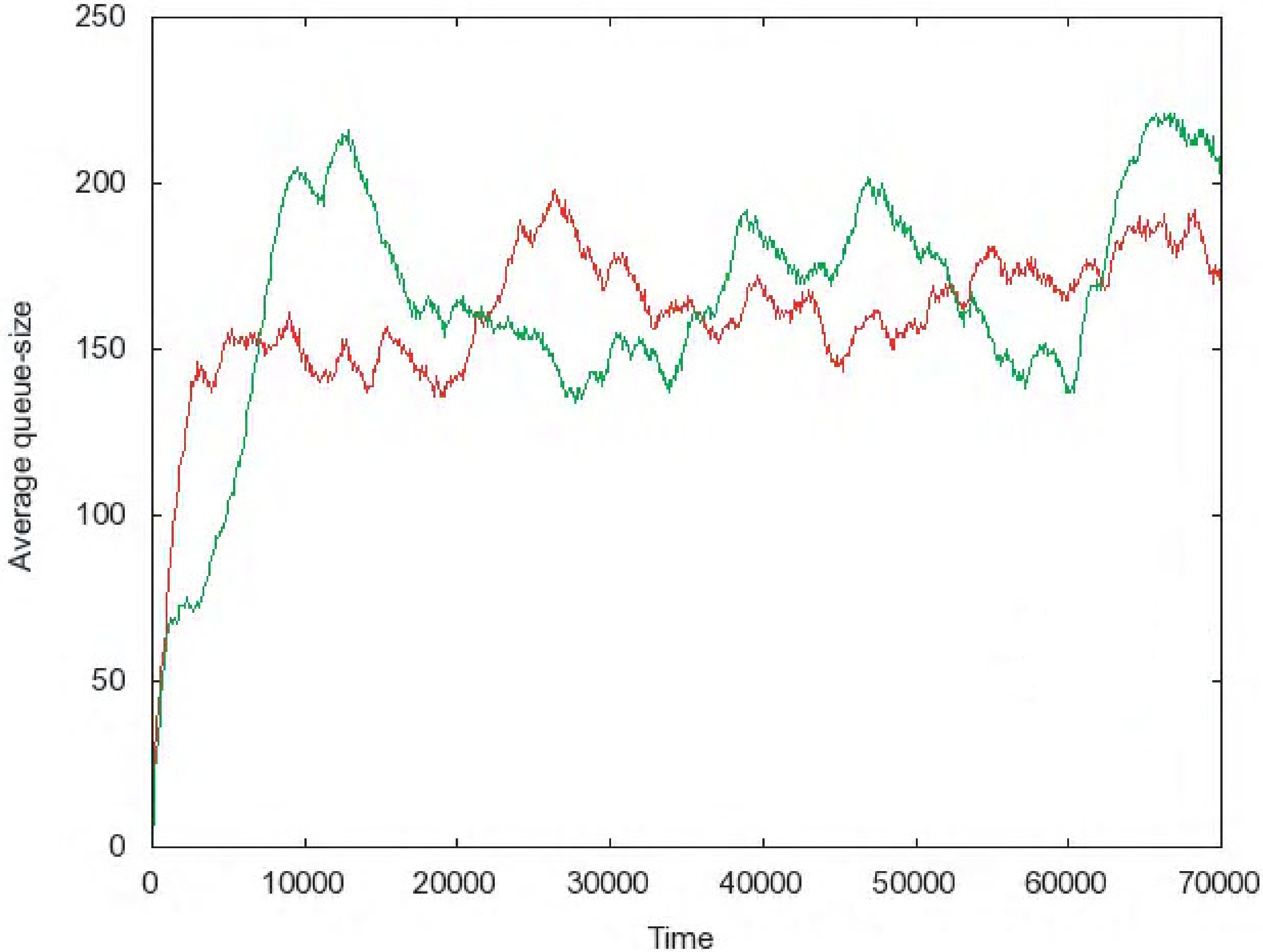
Delay or Queue-size: $\log\log$ versus Geometric algorithm



Delay or Queue-size for Geom. algo.: effect of load



Geom. algo.: Effect of freezing



Discussion

- Resource allocation
  - ○ Key algorithmic problem in a communication network
  - ○ Requires simple, distributed and efficient algorithm
- We presented such an algorithm
  - ○ Essentially, it runs time-varying version of 'Glauber dynamics'
    - – or, Metropolis-Hastings' sampling algorithm
  - ○ Key is to choose the right weight
    - – $f(x) = \log\log(x + e)$ for throughput in general network
    - – $f(x) = x/x^*$ for low delay in geometric networks
- Methodically, advances in understanding of
  - ○ Effect of dynamics on the network performance