Network algorithms made distributed

Devavrat Shah   Jinwoo Shin

Laboratory for Information and Decision Systems

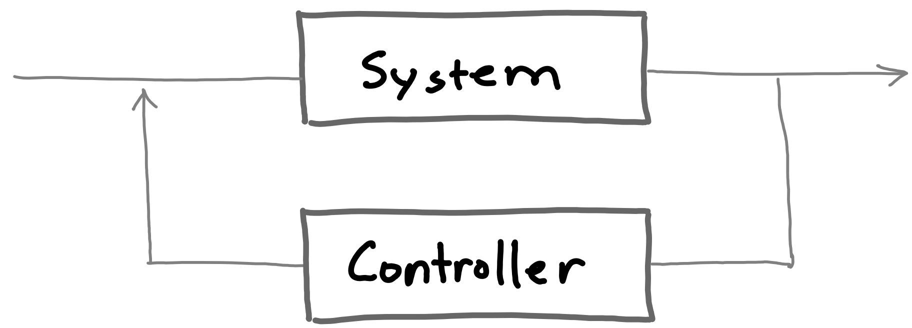
Massachusetts Institute of Technology

http://arxiv.org/abs/0908.3670

Communication Networks

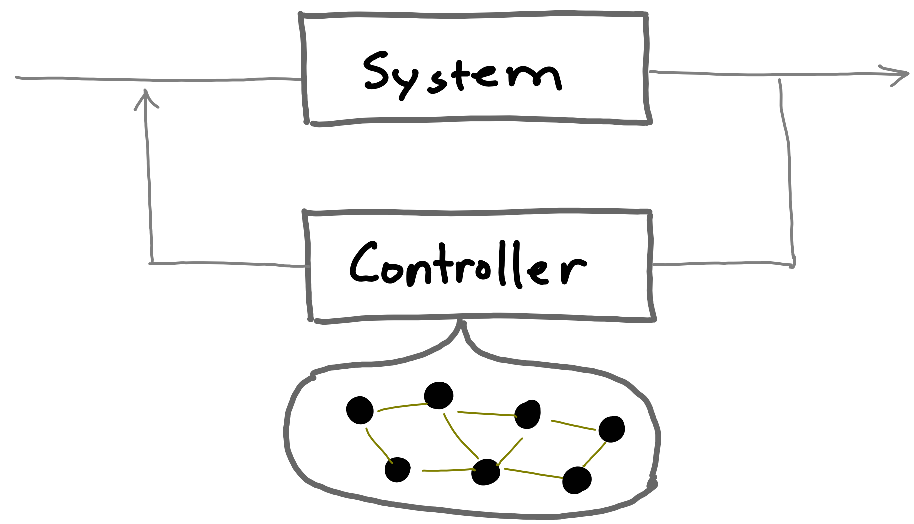
- Network algorithms
  - Required for efficient network resource allocation
    - that is, they must be *high-performance*
  - Required to operate with system constraints
    - that is, they should be *implementable*
- Primary challenge
  - Resolution of tension: performance vs. implementation
  - Ideally,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

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

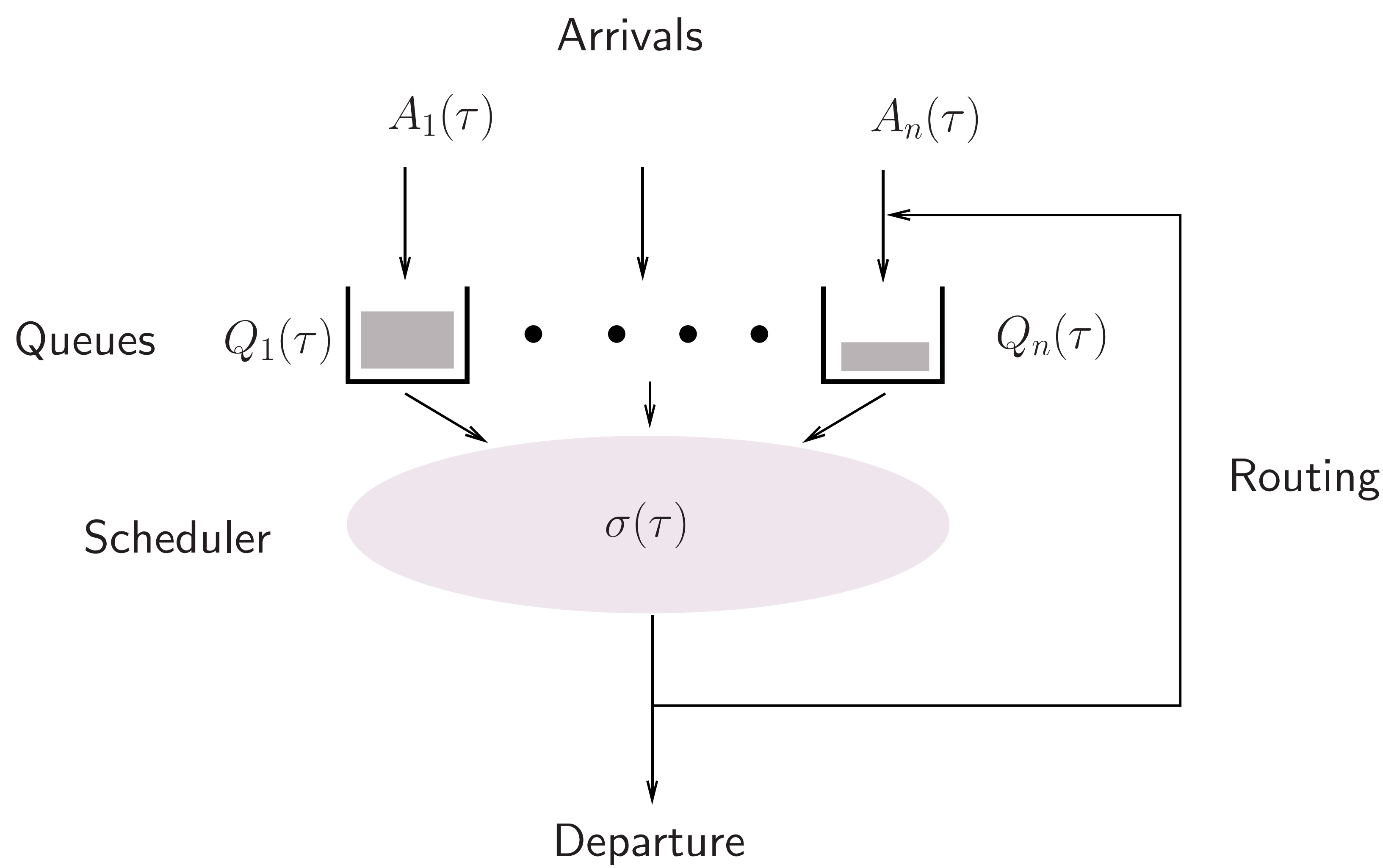
A bit of Philosophy



- 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

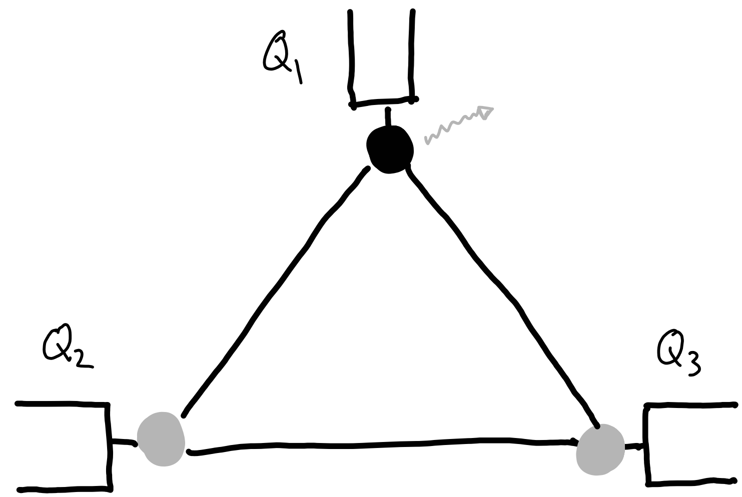
Generic Resource Allocation: Switched Network

- A network of n queues



Generic Resource Allocation: Wireless Network

- Network graph $G = (V, E)$
  - Transmitters $V = \{1, \ldots, n\}$
  - Interference constraints $E = \{(i,j) : i \text{ and } j \text{ interfere}\}$



- Scheduling: choose $\sigma(t) = [\sigma_i(t)] \in \{0,1\}^n$ s.t.
  - If $\sigma_i(t) = 1$, then i transmits at time t
  - Avoid interference, that is

$$\sigma_i(t) + \sigma_j(t) \leq 1, \text{ for all } (i,j) \in E$$

  - That is, $\sigma(t) \in \mathcal{I}(G)$ with

$$\mathcal{I}(G) = \{\sigma \in \{0,1\}^n : \sigma_i + \sigma_j \leq 1, \ \forall \ (i,j) \in E\}.$$

Generic Resource Allocation: Wireless Network

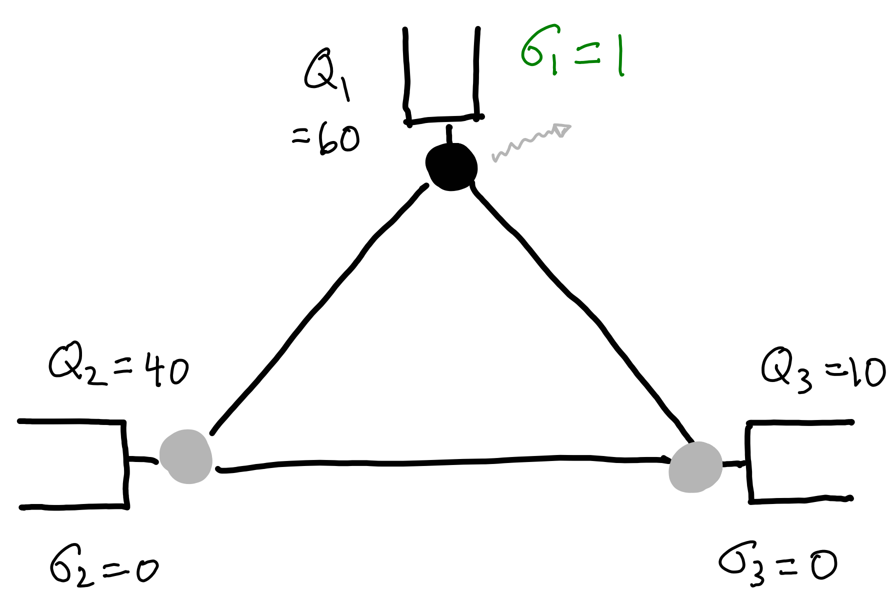
- Network graph $G = (V, E)$
  - Transmitters $V = \{1, \ldots, n\}$
  - Interference constraints $E = \{(i, j) : i \text{ and } j \text{ interfere}\}$

- Scheduling: choose $\sigma(t) = [\sigma_i(t)] \in \mathcal{I}(G)$ for all t

- Arrival:
  - Unit sized packets arrive to each queue as per
    - independent Bernoulli process w. parameter λ_i
  - And, $\lambda = [\lambda_i] \in \Lambda^o$ where
    - $\Lambda = \text{Co}(\mathcal{I}(G))$

Generic Resource Allocation: Wireless Network

- Scheduling algorithm: given wireless network of n nodes
  - With network graph $G = (V, E)$
  - Packet arrival process with $\lambda \in \Lambda^o$
  - Choose $\sigma(t) \in \mathcal{I}(G)$ using information
    - $\mathbf{Q}(t) = [Q_i(t)]$ queue-sizes at time t

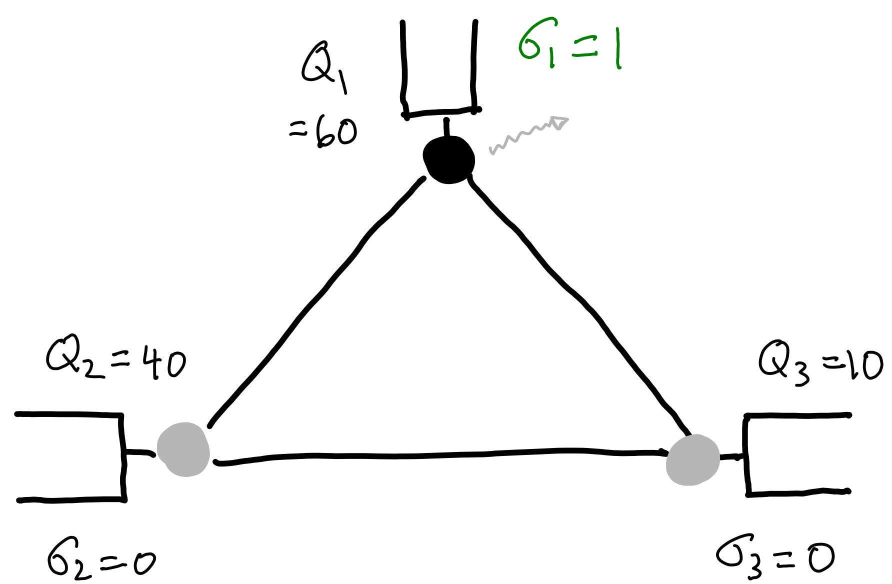


Generic Resource Allocation: Wireless Network

- Scheduling algorithm: given wireless network of n nodes
  - ∘ With network graph $G = (V, E)$
  - ∘ Packet arrival process with $\lambda \in \Lambda^o$
  - ∘ Choose $\sigma(t) \in \mathcal{I}(G)$ using information
    - – $\mathbf{Q}(t) = [Q_i(t)]$ queue-sizes at time t
    - – and, perfect carrier sensing

      if any node i is transmitting, i.e. $\sigma_i(t) = 1$, then

      all of its neighbors can *sense* it, i.e. know $\sigma_i(t) = 1$



Generic Resource Allocation: Wireless Network

- Scheduling algorithm: given wireless network of n nodes
  - With network graph $G = (V, E)$
  - Packet arrival process with $\lambda \in \Lambda^o$
  - Choose $\sigma(t) \in \mathcal{I}(G)$ using information
    - $\mathbf{Q}(t) = [Q_i(t)]$ queue-sizes at time t
    - and, perfect carrier sensing

      if any node i is transmitting, i.e. $\sigma_i(t) = 1$, then
      all of its neighbors can *sense* it, i.e. know $\sigma_i(t) = 1$
  - So that the system is stable, or positive recurrent

Generic Resource Allocation: Wireless Network

- Scheduling algorithm: given wireless network of n nodes
  - With network graph $G = (V, E)$
  - Packet arrival process with $\lambda \in \Lambda^o$
  - Choose $\sigma(t) \in \mathcal{I}(G)$ using information
    - $\mathbf{Q}(t) = [Q_i(t)]$ queue-sizes at time t
    - and, perfect carrier sensing

      if any node i is transmitting, i.e. $\sigma_i(t) = 1$, then
      all of its neighbors can *sense* it, i.e. know $\sigma_i(t) = 1$
  - So that the system is stable, or positive recurrent

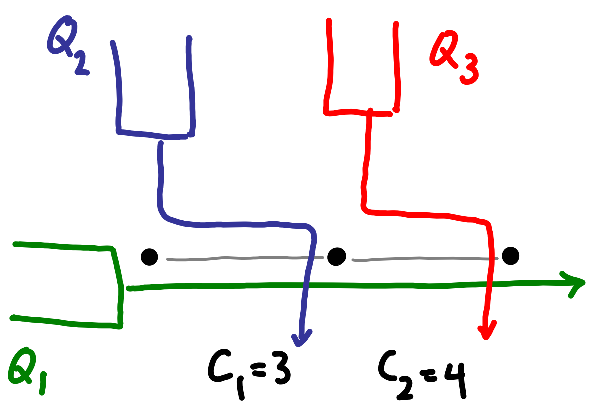
- Thus, challenge is design of simple, distributed algorithm
  - That is, stable or throughput optimal

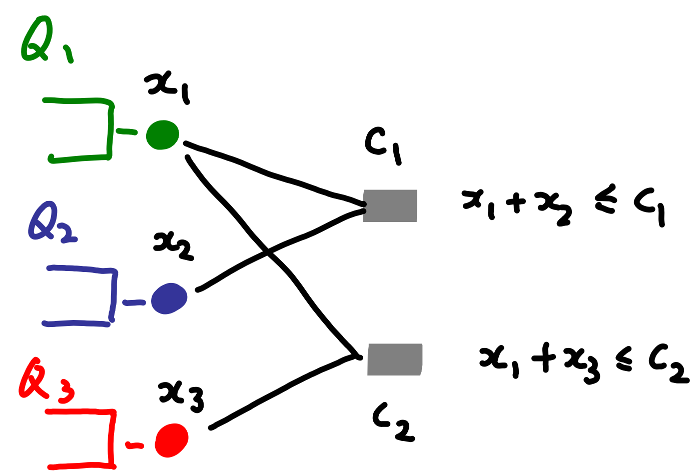
Generic Resource Allocation: Optical Core Network

- Network of capacitated links $\mathcal{L} = \{1, \ldots, L\}$
  - $C = [C_\ell]$ with C_ℓ being capacity of link $\ell \in \mathcal{L}$

- Routes $\mathcal{R} = \{1, \ldots, R\}$
  - Routing matrix $A = [A_{\ell r}]$ with

$$A_{\ell r} = \begin{cases} 1 & \text{if } \ell \in r \\ 0 & \text{otherwise.} \end{cases}$$





Generic Resource Allocation: Optical Core Network

- Network of capacitated links $\mathcal{L} = \{1, \ldots, L\}$
  - $C = [C_\ell]$ with C_ℓ being capacity of link $\ell \in \mathcal{L}$

- Routes $\mathcal{R} = \{1, \ldots, R\}$
  - Routing matrix $A = [A_{\ell r}]$ with

$$A_{\ell r} = \begin{cases} 1 & \text{if } \ell \in r \\ 0 & \text{otherwise.} \end{cases}$$

- Scheduling: assign capacity to routes $x(t) \in \mathbb{N}^R$ s.t.
  - Capacity constraints are satisfied, i.e. $Ax(t) \leq C$
  - Let $\mathcal{X} = \{y \in \mathbb{N}^R : Ay \leq C\}$

Generic Resource Allocation: Optical Core Network

- Network of capacitated links $\mathcal{L}$ and routes $\mathcal{R}$
  - $C = [C_\ell]$ with C_ℓ being capacity of link $\ell \in \mathcal{L}$
  - Routing matrix $A = [A_{\ell r}]$
  - Scheduling requires assigning capacity to routes $x(t) \in \mathcal{X}$

- Arrival and Service requirement
  - Requests on $r \in \mathcal{R}$ arrive as per ind. Poisson proc. of rate λ_r
  - Service requirement of each request is IID exponential of mean 1
  - A request utilizes unit resource on each link when processed/assigned/scheduled
  - $\lambda = [\lambda_r] \in \Lambda^o$ where
    - $\Lambda^o = \text{Co}(\mathcal{X})$.

Generic Resource Allocation: Optical Core Network

- Scheduling algorithm: given an optical core network with R routes
  - ◦ Poisson arrival process with $\lambda \in \Lambda^o$
    - – with IID exponential mean 1 service requirement
  - ◦ Choose $x(t) \in \mathcal{X}$ using information
    - – $\mathbf{Q}(t) = [Q_r(t)]$, the number of backlogged requests at time t
    - – and, bandwidth availability information, i.e. answer to
      is it possible to increase $x_r(t) \to x_r(t) + 1$
    - – in a non preemptive manner
  - ◦ So that the system is stable, or positive recurrent
- Thus, again challenge is design of simple, distributed algorithm
  - ◦ That is, stable or throughput optimal

Generic Resource Allocation

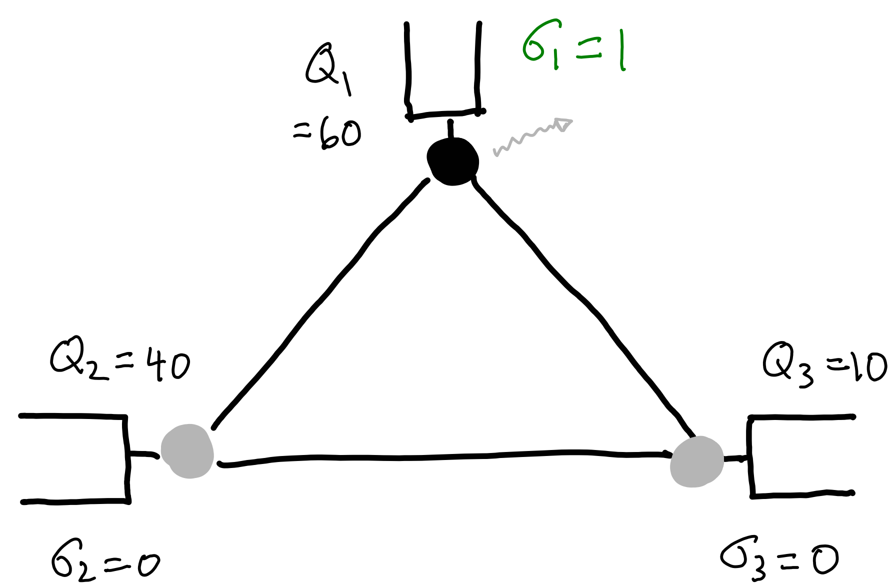
- Implementable scheduling algorithm
  - Distributed: little or no co-ordination
    - for wireless, using sensing and queue-size information
    - for optical, using bandwidth availability and queue-size

- Key question: design of a such an algorithm that is
  - Throughput-optimal

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 is extremely complex
  - For wireless network requires solving
    - hard max. wt. independent set problem !
  - For optical core network, no straightforward implementation
    - due to non preemptiveness and non-packetized scenario

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),distributed and througput optimal
  - Applies more generally
  - A perfect analogy: Metropolis-Hastings Method

- Next, description of the algorithm for
  - Wireless network, followed by optical core network
  - Intuition behind their efficiency
  - Some back-of-envelop calculations

Algorithm: Wireless Network

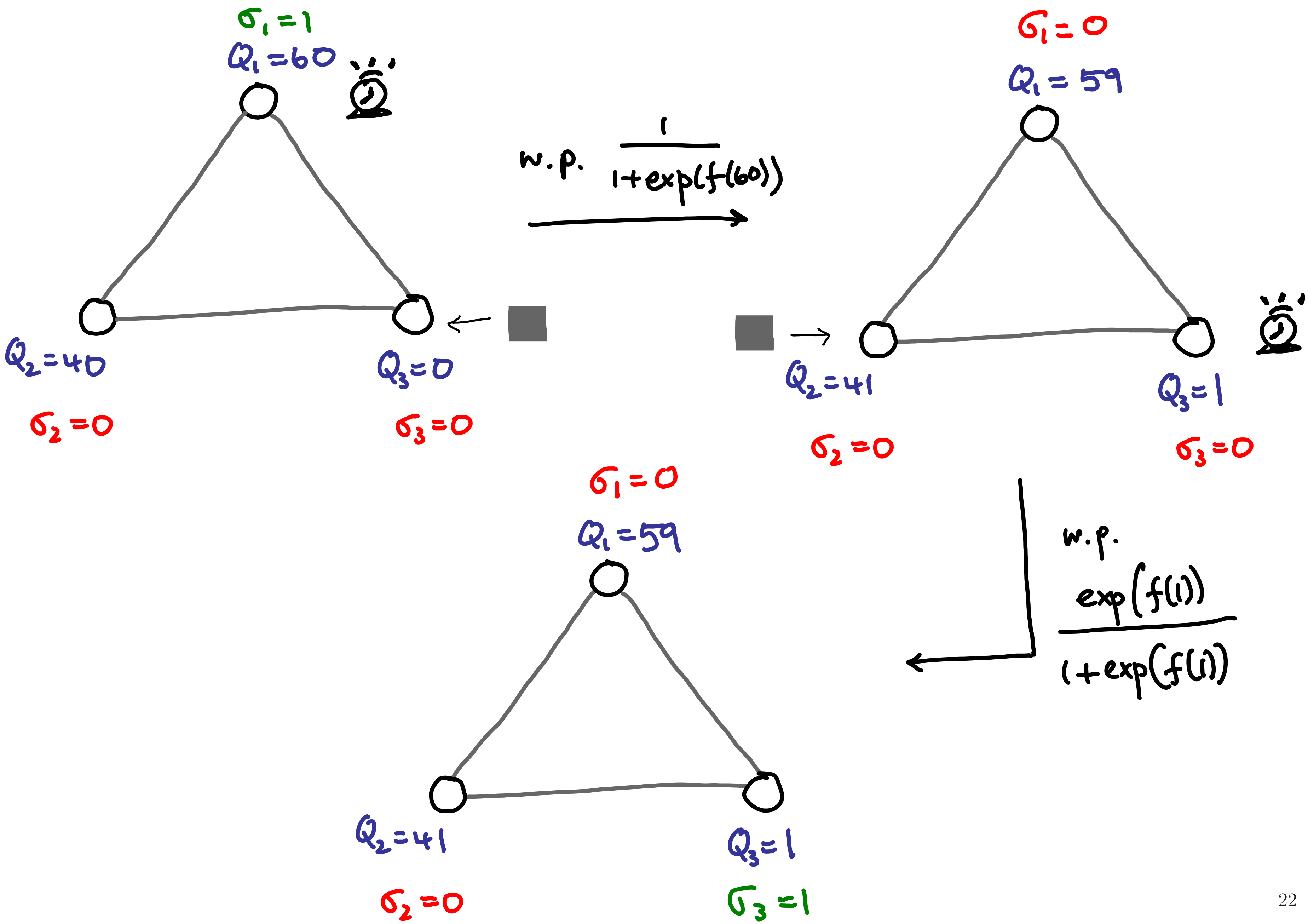
- Each node $i \in V$ has an independent Exponential clock of rate 1
  - When a node, say i's clock ticks, say at time s, do
    - node i checks if medium is FREE or $\sigma_i(s^-) = 1$
    - if yes, then it sets

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

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

- An example

Algorithm: Wireless Network



Algorithm: Wireless Network

- **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

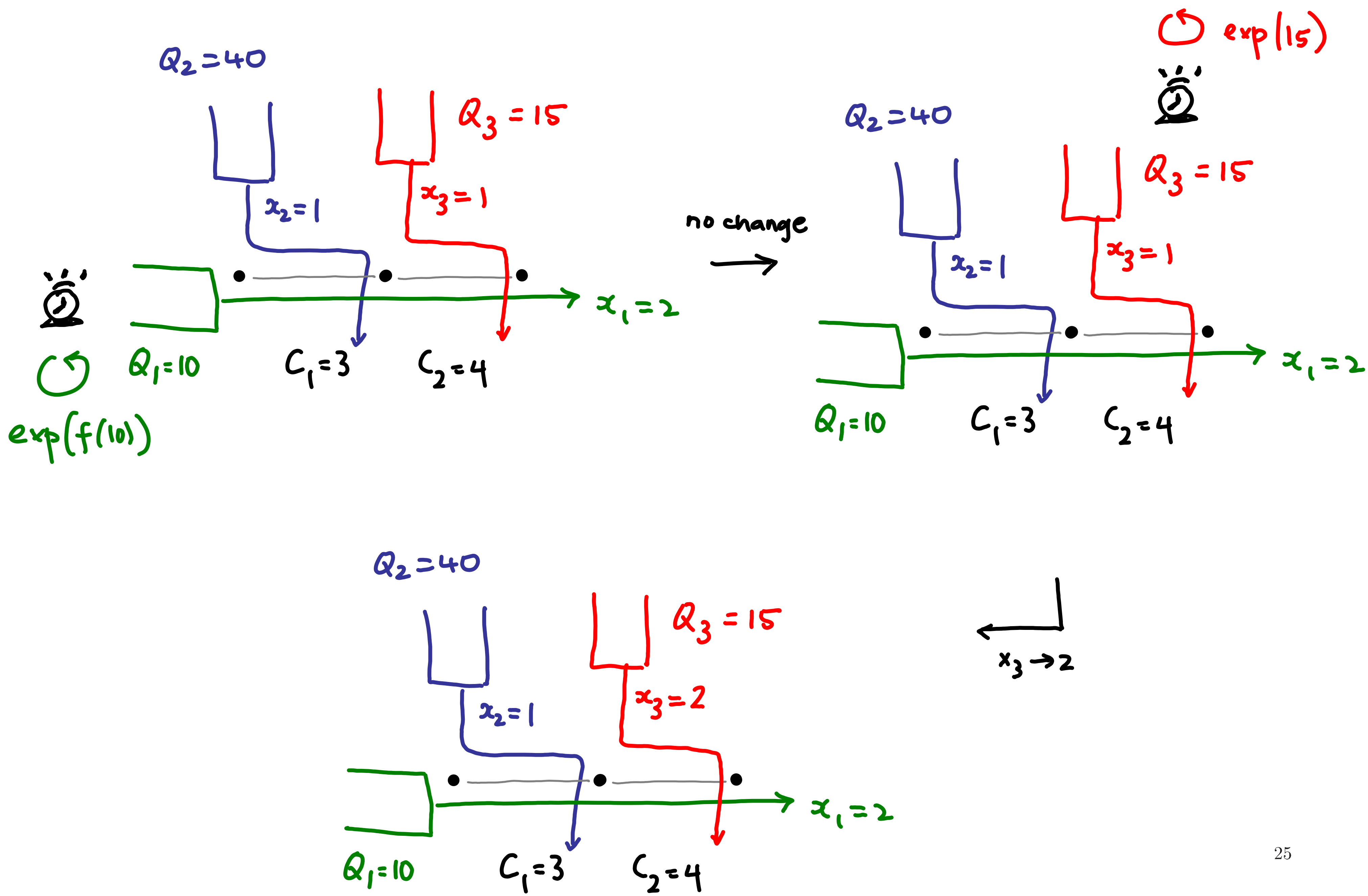
    Essentially, $f(x) = \log\log(x + e)$

- But, before that, algorithm for optical core network

Algorithm: Optical Network

- Queue at ingress of each route $r \in \mathcal{R}$ has an independent clock
    - ∘ Ticks as per Poisson process of rate $\exp(f(Q_r(t)))$
- When a route, say r's clock ticks, say at time t
    - ∘ Check if additional bandwidth is available on its route
        - – if yes, then it acquires it,
            - ∘ sets $x_r(t^+) = x_r(t^-) + 1$,
            - ∘ the head-of-line request starts using it, and
            - ∘ $Q_r(t^+) = Q_r(t^-) - 1$
        - – else, nothing happens
- A request on any route, upon completion
    - ∘ Frees the acquired unit resource of links along its route

Algorithm: Optical Network



Algorithm: Optical Network

- **Theorem 2.** With the proper choice of f, the algorithm is efficient.
  Essentially, $f(x) = \log\log(x + e)$

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

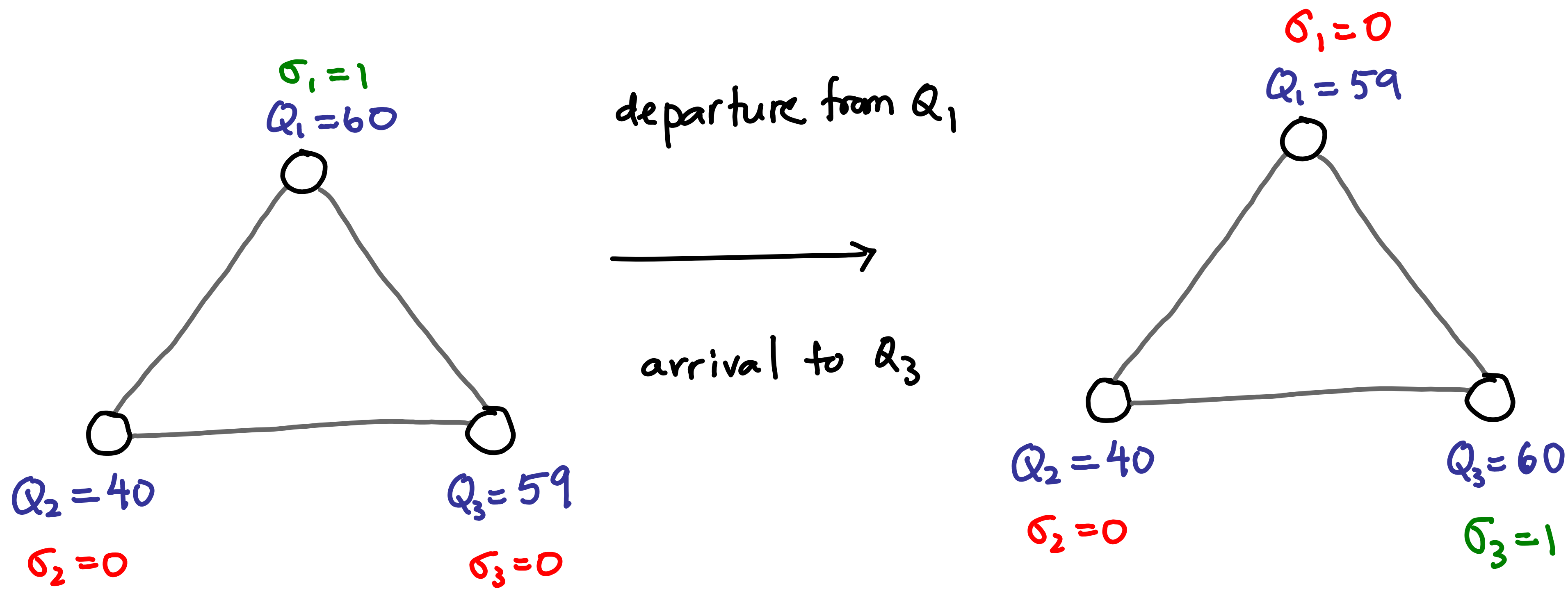
- Assume: $\mathbf{Q}(t) = \mathbf{Q}$ is *fixed*
- Glauber dynamics
  - ∘ Induces irreducible, finite state reversible Markov chain on $\mathcal{I}(G)$
  - ∘ Has a unique stationary distribution, say $\pi_{\mathbf{Q}}$
- **Lemma 1.** The $\pi_{\mathbf{Q}}$ has the following properties:
  1. $\pi_{\mathbf{Q}}(\sigma) \propto \exp(\sum_i \sigma_i f(Q_i))$ 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 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 $\mathrm{T}_{\textsf{mix}}(n, f(\mathbf{Q}(t)))$ 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)
    - $C_f(n, \mathbf{Q}(t)) \sim \mathrm{T}_{\textsf{mix}}(n, f(\mathbf{Q}(t)))f'(\mathbf{Q}(t))\Delta\mathbf{Q}(t)$ 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 $f(x) = x$

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

→ 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 $f(x) = \log x$

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

$\longrightarrow$ 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

- **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 $Q_{\max}(t) = \max_k Q_k(t)$ along with $Q_i(t)$
  - ∘ That is, its $\max\left(f(Q_i(t), \sqrt{f(Q_{\max}(t))}\right)$
    - – requires minimal global information
  - ∘ Or, a *learning* based mechanism is needed
    - – Algorithm by Jiang and Walrand (2008)
    - – Rate optimality & Positive recurrence established in Jiang, Shah, Shin and Walrand (2009)
  - ∘ We *strongly* believe that
    - – the 'vanilla' algorithm works

Back to Optical: Algorithm $\approx$ Loss network

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

- When a route, say r's clock ticks, say at time s
  - Check if additional bandwidth is available on its route
    - if yes, then acquire it
      - set $x_r(t^+) = x_r(t^-) + 1$,
      - the head-of-line request starts using it, and
      - $Q_r(t^+) = Q_r(t^-) - 1$
    - else, nothing happens

- A request on any route, upon completion
  - Frees the acquired unit resource of links along its route

Loss Network

- Assume: $\mathbf{Q}(t) = \mathbf{Q}$ is *fixed*
- Loss network
  - ○ Induces irreducible, finite state reversible Markov chain on $\mathcal{X}$
  - ○ Has a unique stationary distribution, say $\pi_{\mathbf{Q}}$
- **Lemma 2.** The $\pi_{\mathbf{Q}}$ has the following properties:

  1. For each $\mathbf{x} \in \mathcal{X}$, with $\rho_r = \exp(f(Q_r))$

$$\pi_{\mathbf{Q}}(\mathbf{x}) \propto \prod_r \frac{\rho_r^{x_r}}{x_r!},$$

Loss Network

- Assume: $\mathbf{Q}(t) = \mathbf{Q}$ is *fixed*
- Loss network
  - ∘ Induces irreducible, finite state reversible Markov chain on $\mathcal{X}$
  - ∘ Has a unique stationary distribution, say $\pi_{\mathbf{Q}}$
- **Lemma 2.** The $\pi_{\mathbf{Q}}$ has the following properties:

  1. For each $\mathbf{x} \in \mathcal{X}$, with $\rho_r = \exp(f(Q_r))$

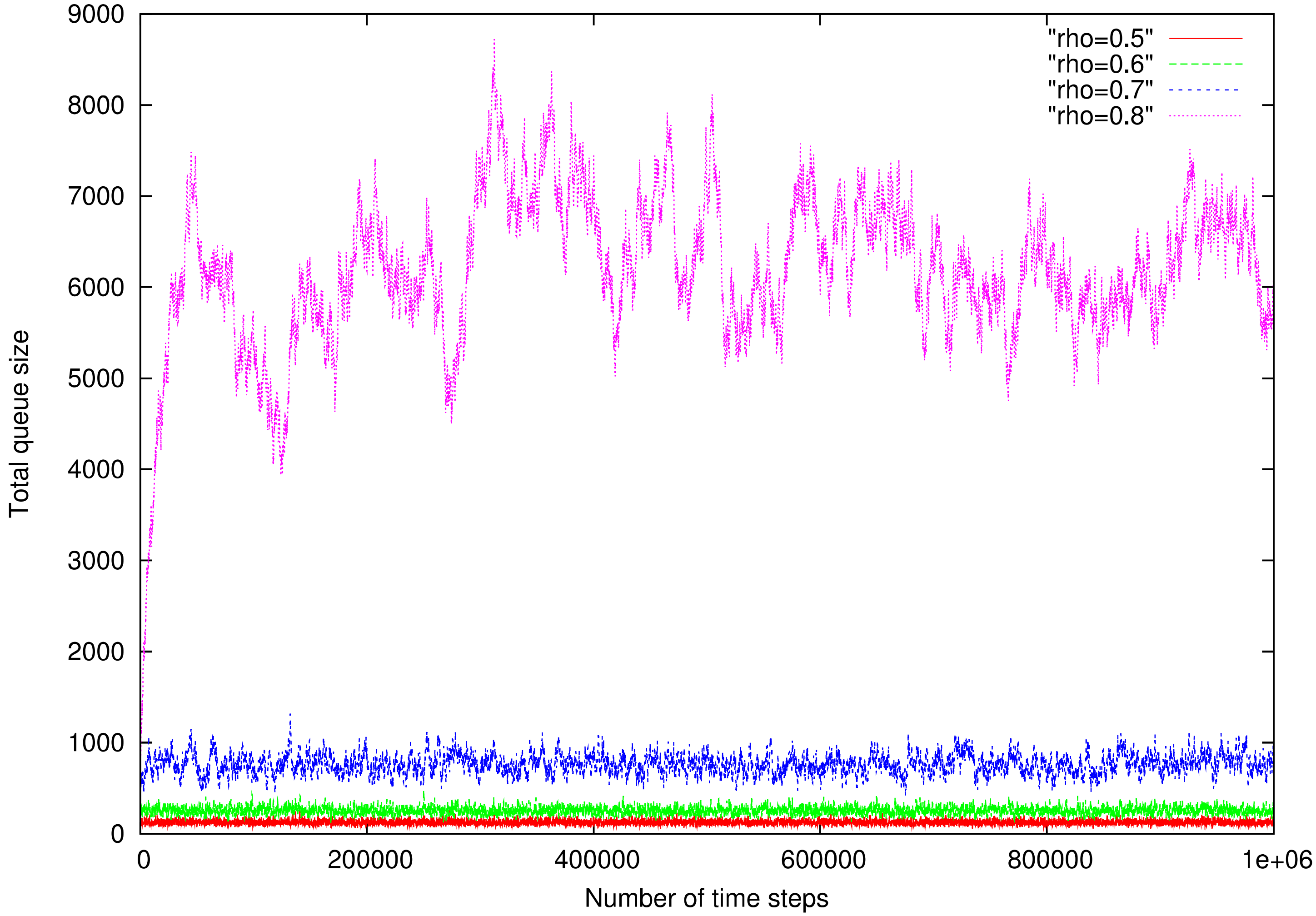
  $$\pi_{\mathbf{Q}}(\mathbf{x}) \propto \prod_r \frac{\rho_r^{x_r}}{x_r!},$$

  2. $\mathbb{E}_{\pi_{\mathbf{Q}}}\left[\sum_i x_r f(Q_r)\right] \geq \left(\max_{\mathbf{y}\in\mathcal{X}} \sum_r y_r f(Q_r)\right) - C$.

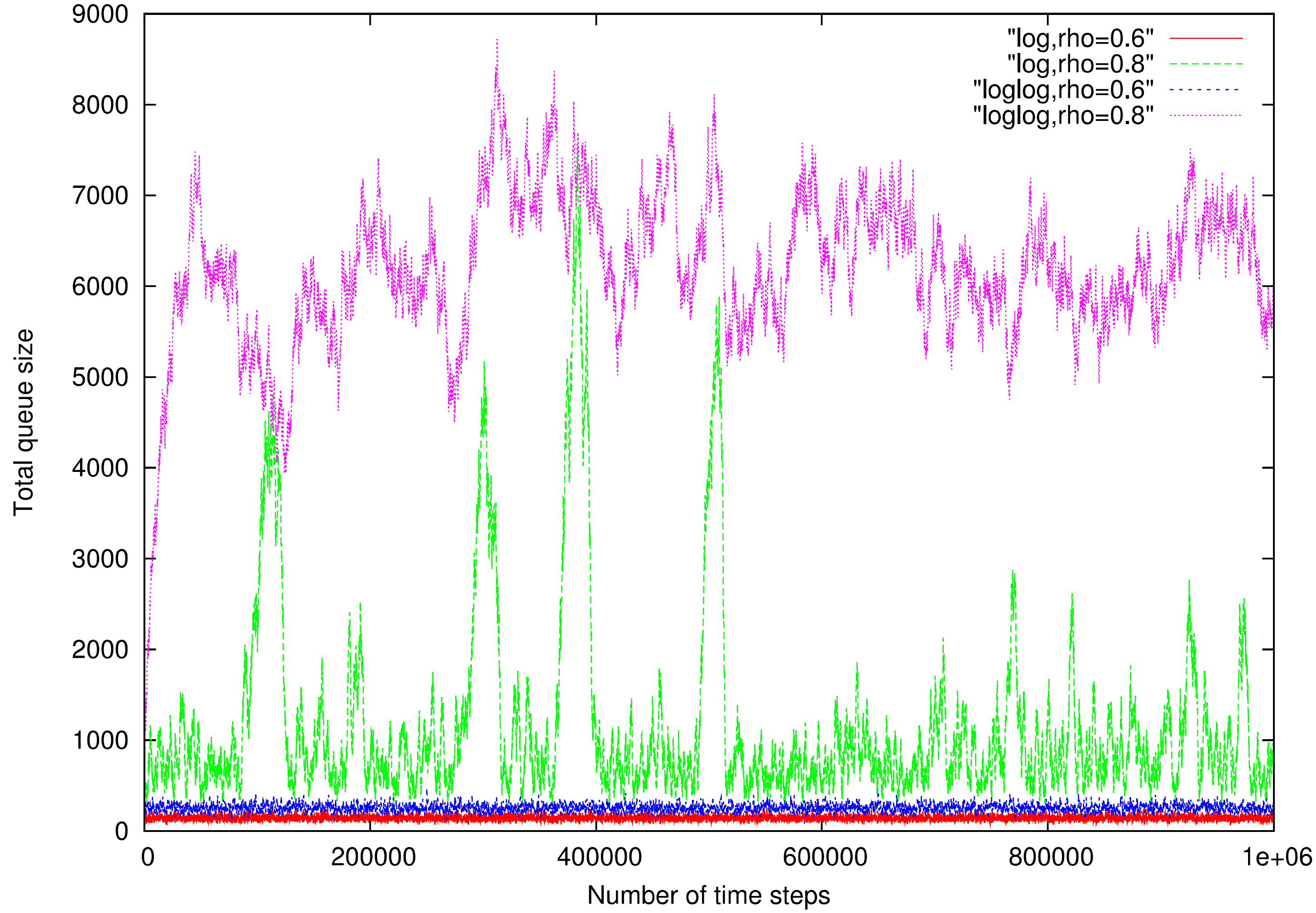
→ If $\mathbf{Q}(t)$ *fixed*, then Loss network chooses $\mathbf{x}(t)$ s.t.

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

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



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



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 : $f(x) = \log\log(x + e)$

- Methodically, advances in understanding of
  - Effect of dynamics on the network performance

Discussion

- 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 better than $\exp(n)$
    - unless, $P = NP$

      cf. Shah, Tse and Tsitsiklis (2009)
  - However, for networks with 'geometry'
    - possible to adapt our algorithm to obtain low delay

      cf. Shah and Shin (20XX)