Reversibility and network algorithms

Devavrat Shah
Massachusetts Institute of Technology
Switched network: model of interest

- Stochastic processing network of Harrison ‘00
- Switched networks: discrete-time instances
Switched network

- **Example:** dynamic resource sharing
  - Communication
    - Bandwidth sharing model of Internet
    - Wireless multi-hop a la mesh-network
  
- **Computation-Storage**
  - Cloud facility or data-center

- **Human Resource (HR)**
  - Project management in large industries

- **Transportation**
  - Road traffic signaling
Switched network

- **Basic operational task**
  - Scheduling or sharing of resources
    - Among various contending entities
  - Examples
    - Which laptop transmits over WiFi
    - Disk/CPU allocation to a Virtual Machine
    - Project assignments to skilled employees
    - Signaling mechanisms on road

- **Network performance**
  - Depends crucially on scheduling policy
Network performance

- Three metrics
  - Capacity
    - What is the effective resource
  - Queue-size, latency or delay
    - How long does it take to get serviced
  - Complexity
    - What sorts of implementations are feasible

- Interest is in understanding
  - Trade-offs between these metrics
Rest of the talk

- Role of reversibility (product-form distributions) in
  - Design and analysis of scheduling algorithms

- Specifically, we shall discuss
  - Scheduling *inside* queues
    - To achieve low network-wide delay
  - Scheduling *resources* among queues
    - To achieve low network-wide delay
  - Implementing scheduling policies
    - To achieve low-complexity, distributed design
Network without constraints

- Network of \( n \) queues
  - Exogenous Poisson packet arrival process for each queue
    - Packets are of unit size (require unit amount of service)
  - Each queue can serve packets in discrete time
    - One packet per unit time (= time slot)
    - Without any further constraint
  - Served packets depart or join another queue

\[ \begin{align*}
\lambda_1 & = \lambda + 0.5 \lambda_3 \\
\lambda_2 & = \lambda_2 + \lambda_4 \\
\lambda_3 & = \lambda_3 \\
\lambda_4 & = 0.5 \lambda_1 + \lambda_4 \\
\lambda_5 & = 0.5 \lambda_1 + 0.5 \lambda_3
\end{align*} \]
Network without constraints

- **Network of n queues**
  - Exogenous Poisson packet arrival process for each queue
  - Each queue can serve one packet per time slot
    - Without any further constraint
  - Scheduling required *inside* each queue
    - To decide which amongst the waiting packets to serve first

\[
\begin{align*}
\bar{g}_1 &= \lambda_1 + 0.5\lambda_3 \\
\bar{g}_6 &= \lambda_1 + 0.5\lambda_3 \\
\bar{g}_5 &= 0.5\lambda_1 + 0.5\lambda_3 \\
\bar{g}_4 &= 0.5\lambda_1 + \lambda_4 \\
\bar{g}_7 &= \lambda_3 + \lambda_4 \\
\bar{g}_3 &= \lambda_3 \\
\bar{g}_2 &= \lambda_2 + \lambda_4
\end{align*}
\]
Network without constraints

- Network of n queues in *continuous* time
  - Exogenous Poisson packet arrival process for each queue
  - Each queue has unit service capacity
  - Scheduling *inside* each queue as per
    - Pre-emptive Last In First Out (PL)
    - Which may serve a packet *in parts* unlike in discrete time
Network without constraints

- Network of n queues in *continuous* time
  - PL Scheduling *inside* each queue
    - Quasi-reversible queues (cf. Kelly '78)
  - Stationary distribution is *product-form* (cf. BCMP '74, Kelly ‘78)

\[
P(Q_1 = k_1, \ldots, Q_7 = k_7) \sim \frac{1}{g_1 \ldots g_7} g_j^{k_j}
\]
Network without constraints

- Network of n queues in continuous time
  - PL Scheduling inside each queue
  - The product-form distribution implies that
    - The average delay $E[D_i] = \sum_{j=1}^{n} \frac{1}{1 - \rho_j}$ for each route $i$
    - If all $\rho_j = \rho$, then delay of route $i$ scales as $(\text{num of hops})/(1-\rho)$
Network without constraints

- Network of \( n \) queues in *continuous* time
  - PL Scheduling *inside* each queue
  - The *product-form* distribution implies that
    - The average delay of route \( i \) scales as \((\text{num of hops})/(1-\rho)\)
- Can we obtain similar performance for *discrete* time setting?
  - That is, serving each packet in entirety

\[
\begin{align*}
\lambda_1 & = \lambda_1 + 0.5\lambda_3 \\
\lambda_2 & = \lambda_2 + \lambda_4 \\
\lambda_3 & = \lambda_3 \\
\lambda_4 & = 0.5\lambda_1 + \lambda_4 \\
\lambda_5 & = 0.5\lambda_1 + 0.5\lambda_3
\end{align*}
\]
Emulation Lemma.

It is possible to design scheduling at each queue so that
- The time a packet departs from each queue in discrete time network
- Is at most 1 more than that in the corresponding continuous time network with each node operating as per PL policy
- This “coupling” is distribution independent
Emulation Lemma

- The scheduling algorithm in discrete time network
  - Schedule at each queue as per the Last In First Out policy
  - With respect to \([A]\), where \(A\) is the arrival time of a packet
    - In this queue in the continuous time network operating with PL policy
    - Ties broken as per continuous time network

- In summary
  - By simulating continuous time network (in a causal manner)
    - It is possible to achieve delay per (packet-)flow
    - That is proportional to \((\text{num of hops})/(1-\rho)\)
Network without constraints

- The achievable delay scaling
  - \((\text{num of hops})/(1-\rho)\)

- For M/M/1 queues in tandem
  - This is the best achievable

- For queues in tandem serving packets
  - Delay scales as \((\text{num of hops}) + 1/(1-\rho)\)
    - The “pipe-lining” effect

- Question: which is the right scaling?
  - Single “bottleneck” link entirely avoids this
Network with constraints

- Network of $n$ queues
  - Exogenous Poisson packet arrival process for each queue
    - Packets are of unit size (require unit amount of service)
  - Each queue can serve packets in discrete time
    - One packet per unit time (= time slot)
- Scheduling constraints
  - Let $\sigma = [\sigma_i] \in \{0,1\}^n$ be subset of queues served
  - Then
    - $\sigma$ must satisfy certain constraints: represented by $\sigma \in S \subseteq \{0,1\}^n$

Question: how does the “optimal” queue-size/delay scale

- Depending upon $S$ and gap to the capacity $(1-\rho)$
Network with constraints

- **Example 1:**
  - Parallel queues, $n$ of them
  - The net average queue-size $Q_1 + \ldots + Q_n \approx n/(1-\rho)$
Network with constraints

- **Example 2:**
  - One server, n queues
  - The net average queue-size: $Q_1 + \ldots + Q_n \approx 1/(1-\rho)$
Network with constraints

- Example 3:
  - $N \times N$ switch: $n = N^2$ queues
  - Example of switch with $N = 2$:
Network with constraints

- **Example 3:**
  - $N \times N$ switch: $n = N^2$ queues
  - Average queue-size: $Q_1 + \ldots + Q_n$
    - conjectured*: $(\text{const}) \frac{N}{1-\rho}$
    - Known upper bound: $\frac{N^2}{1-\rho}$
    - Known lower bound: $\frac{N}{1-\rho}$

\[ \lambda_1 = \frac{\rho}{n}, \quad \lambda_2 = \frac{\rho}{n}, \quad \lambda_n = \frac{\rho}{n} \]

* = QUESTA open problem special issue
Network with constraints

- **Network of n queues**
  - With scheduling constraints represented by
    - Schedule \( \sigma \in S \subseteq \{0,1\}^n \)

- **The convex hull of \( S \) is the capacity region**
  - Let it be represented as (polytope)
    - \( \Lambda=\{x \in [0,1]^n : Ax \leq C\} \) with
      - A non-negative \( m \times n \) matrix
      - \( C \) non-negative valued \( m \)-vector

- **Effectively, any scheduling policy imposes constraint**
  - Service rate \( \sigma \in \Lambda \) (with abuse of notation)
Network with constraints

- Proportional fair policy: each time
  - Choose schedule so that induced service rate $\sigma$ is such that
    - It maximizes objective $\sum_i Q_i \log \sigma_i$ over all $\sigma \in \Lambda$
  - This is achieved by a simple randomized policy
    - Find $\sigma$ that solves above optimization problem
    - Decompose $\sigma$ as convex combination of actions in $S$
      - $\sigma = \sum_k \alpha_k \pi_k$ for $\pi_k \in S$ with $\sum_k \alpha_k = 1$
      - Choose $\pi_k$ with probability $\alpha_k$

- This has been well analyzed (for bandwidth sharing) by
  - Bandon-Massoulie ‘01, Kelly-Williams ‘04, Massoulie ‘06, Kang-Kelly-Lee-Williams ‘08, Ye-Yao ‘08
Network with constraints: prop. fair

- Kang-Kelly-Lee-Williams ‘08
  - Considered heavy traffic limit of such a network
    - With multiple links bottle-necked
    - Assumed
      - Matrix A full rank
      - Local traffic condition: for each j, there exists i s.t. $A_{ij} > 0, A_{ij}' = 0$ for all $j' \neq j$

- Characterized product-form stationary distribution
  - For diffusion approximation

- We establish validity of diffusion approximation in stationarity
  - The product-form distribution is the limit of stat. dist. of system
    - Under heavy-traffic scaling limit
  - That is, exchange of limits is valid
    - (Shah-Tsitsiklis-Zhong ‘11)
Network with constraints: prop. fair

- The product-form stationary distribution implies
  - The average queue-size is
    \[ \mathbb{E}[Q_i] \approx \lambda_i \sum_j \frac{A_{ji}}{c_j - (A\lambda)_j} \]
    \[ \leq \left| \{ j : A_{ji} \neq 0 \} \right| \cdot \max_{j : A_{ji} \neq 0} \left( \frac{\lambda_i A_{ji}}{c_j - (A\lambda)_j} \right) \]
  - And, for any policy
    \[ \mathbb{E}[Q_i] \geq \max_{j : A_{ji} \neq 0} \frac{\lambda_i A_{ji}}{c_j - (A\lambda)_j} \]
- That is, prop. fair is optimal
  - Up to the “number of hops”
Network with constraints: prop. fair

- Back to conjecture for switch
  - Assuming the KKLW ‘08 holds for $N \times N$ switch
    - Using Proportional fair scheduling policy
  - The net average queue-size would turn out to be
    - $2N/(1-\rho)$ : matches the conjecture !

- Recent progress (Shah-Tsitsiklis-Zhong ‘xx)
  - For uniform loading with $(1-\rho) = 1/N$
    - We show that the net average queue-size is $N^{5/2}$
  - Recall (for $(1-\rho) = 1/N$)
    - What was known: $N^3$
    - Conjecture is: $N^2$
Network w constraints: implementation

- **A reasonable policy**
  - At each time choose schedule $\sigma \in S$ such that
    - It maximizes objective $\sum_i F(\sigma_i)$
    - For some function $F$ which may depend on queue-size, etc.

- **Implementation:**
  - How to choose this schedule each time
    - Using simple algorithm
      - Low complexity
      - Minimal data-structure
    - Preferably in a distributed manner
      - With little protocol co-ordination overhead
Network w constraints: implementation

- **Product-form distribution**
  - Consider a Markov chain on $S$ with stationary distribution
    $$
    \pi(\sigma) \propto \exp\left( \sum_i F(\sigma_i) \right)
    $$
  - Then
    - Variational characterization of such distribution suggests
      $$
      \mathbb{E}_p \left[ \sum_i F(\sigma_i) \right] \geq \left( \max_{\pi \in S} \sum_i F(\pi_i) \right) - \log |S|
      $$
  - That is, effectively by *sampling* schedule at each time
    - As per stationary distribution of this Markov chain is what we want
Network w constraints: implementation

- Two issues
  - Designing Markov chain with such product-form distribution
    - Reversible construction a la Metropolis-Hasting’s Rule
    - The transitions of such a Markov chain are essentially distributed
      - Separable objective is particularly useful for this property
  - Sampling from stationary distribution of Markov chain
    - The objective keeps changing every time
    - And Markov chain makes only few transitions per unit time
    - By choice of slowly varying objective $F$
      - It is possible to essentially sample from stationary distribution at all times
        (Shah-Shin ‘08, ’10)
Discussion

- **Reversible networks are useful**
  - Primarily because of their product-form stationary distribution
    - Calculate average delay
      - Network without constraints
      - Network with constraints using proportional fair policy
    - Choose schedule that maximizes appropriate objective

- **Reversible networks are, however, too specific**
  - Therefore, *approximate* characterization can be quite useful
    - In expanding scope of these results
  - One such approximation is obtained means of
    - “Comparison” property (Shah-Shin-Tetali ‘11)
Consider two Markov chains on finite state space

Let

- $P = [P_{ij}]$ and $Q = [Q_{ij}]$ be the respective transition matrices
- $P$ and $Q$ have unique stationary distributions $p$ and $q$
- $n$ be the size of the state space
- $C_{ij} = \max(Q_{ij}/P_{ij}, P_{ij}/Q_{ij})$ and $C^* = \max_{ij} C_{ij}$

Then

$$D(p \mid q) \leq 2n \log C^*$$

And, this is tight

This can be used to characterize a large class of

- Approximate product-form distributions
- It has been extremely useful in designing distributed algorithms for us