Scheduling: Queues & Computation

achieving baseline performance efficiently

Devavrat Shah

LIDS, MIT
Outline

Two models

switched network and bandwidth sharing

Scheduling: desirable performance

queue-size scaling and computation

Both models are equivalent

from perspective of desirable scheduling performance

Performance of known scheduling algorithms

Discussion
Unit sized packets arrive as per independent Poisson processes
Switch: scheduling constraints

Each integral time instance
- each input can send at most one packet
- each output can receive at most one packet
Switch: scheduling constraints

Each integral time instance
- each input can send at most one packet
- each output can receive at most one packet

\[
\begin{bmatrix}
1 & 0 \\
0 & 1 \\
\end{bmatrix}
\]
The set of feasible rates

convex combination of possible schedules

In a 2-port switch, \( \lambda = [\lambda_{ij}] \) is feasible iff

\[
\begin{bmatrix}
\lambda_{11} & \lambda_{12} \\
\lambda_{21} & \lambda_{22}
\end{bmatrix} \leq \alpha \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \beta \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}
\]

s.t. \( \alpha + \beta \leq 1, \quad \alpha, \beta \geq 0. \)

Load of \( \lambda \): the smallest possible \( \alpha + \beta \) satisfying above
Switch: resource capacity

The set of feasible rates

convex combination of possible schedules

In a 2-port switch, \( \lambda = [\lambda_{ij}] \) is feasible iff

\[
\begin{align*}
\lambda_{11} + \lambda_{12} & \leq 1 \\
\lambda_{11} + \lambda_{21} & \leq 1 \\
\lambda_{21} + \lambda_{22} & \leq 1 \\
\lambda_{12} + \lambda_{22} & \leq 1
\end{align*}
\]

Load of \( \lambda \) : maximum of the above four terms
Switched network

Scheduling constraints

each time choose schedule $\sigma \in \mathcal{S}$

$\mathcal{S} \subset \mathbb{Z}_{\geq 0}^N$ is a finite monotone set

if $\sigma \in \mathcal{S}$ and $\sigma' \in \mathbb{Z}_{\geq 0}^N$, $\sigma' \leq \sigma$ then $\sigma' \in \mathcal{S}$
Switched network

Feasibility region

convex hull of \( S \)

can be represented as

\[
\lambda \in \mathbb{R}^N_{\geq 0} : A\lambda \leq b, \quad A \in \mathbb{R}^{M\times N}_{\geq 0}, \quad b \in \mathbb{R}^M_{> 0}
\]

given a feasible \( \lambda \), it’s load is defined as

\[
\rho(\lambda) = \min\{\rho \leq 1 : A\lambda \leq \rho b\}
\]
Maximum weight policy [Tassiulas-Ephremides ’92]

Current queue-sizes $Q = [Q_i]$

Choose schedule $\sigma \in \mathbb{S}$ so that

$$\sigma \in \arg \max_{\gamma \in \mathbb{S}} \left( \sum_i \gamma_i Q_i \right)$$
Max weight: example

\[ Q_{11} \quad Q_{12} \quad Q_{21} \quad Q_{22} \]

Input 1

Input 2

Output 1

Output 2

\[
\begin{bmatrix}
1 & 0 \\
0 & 1 \\
\end{bmatrix}
\]
Bandwidth sharing network

Jobs arrive as per independent Poisson processes

Each link has unit capacity

Jobs arrive as per independent Poisson processes

job sizes have exponential distribution (unit mean)
Rate allocation \( x = [x_i] \)
such that link capacity constraints are satisfied
Bandwidth sharing network

The set of feasible rates that satisfy all link capacity constraints

In our example, $\lambda = [\lambda_i]$ is feasible iff

\[
\begin{align*}
\lambda_1 + \lambda_2 & \leq 1 \\
\lambda_3 + \lambda_4 & \leq 1
\end{align*}
\]

Load of $\lambda$: maximum of the above four terms
Bandwidth sharing network

N routes/flows

M links

capacity of link $i$ is $b_i$, $1 \leq i \leq M$

Routing matrix: $A \in \mathbb{R}_{\geq 0}^{M \times N}$

$A_{ij}$: resource on link $i$ consumed by unit flow on route $j$

Rate or bandwidth allocation

$x \geq 0$, $Ax \leq b$
Bandwidth sharing network

Feasibility region

\[ \lambda \in \mathbb{R}^N_{\geq 0} : A\lambda \leq b, \quad A \in \mathbb{R}^{M \times N}_{\geq 0}, \quad b \in \mathbb{R}^M_{>0} \]

given a feasible \( \lambda \), it's load is defined as

\[ \rho(\lambda) = \min\{ \rho \leq 1 : A\lambda \leq \rho b \} \]
Proportional fair policy [Kelly ’97, Kelly-Maullo-Tan ’97]

Current queue-sizes $Q = [Q_i]$

Choose rate allocation $x = [x_i]$ s.t.

$$\text{maximize } \sum_i Q_i \log x_i$$

$$Ax \leq b, \ x \geq 0$$

$$x_i = 0 \text{ if } Q_i = 0$$
Proportional fair: example

\[ Q_i = 1, \; \forall \; i \; \Rightarrow \; x_i = \frac{1}{2}, \; \forall \; i \]
Queue-size

Scheduling or allocation policy determines queue-sizes

we’ll restrict to policies with steady-state distribution

for all $\lambda$ with $\rho(\lambda) < 1$

We’ll focus on

average total queue-size: $\mathbb{E} \left[ \sum_i Q_i \right]$

tail probability of total queue-size:

$$\frac{1}{t} \log \mathbb{P} \left( \sum_i Q_i > t \right) \text{ for large } t$$
Desired Queue-size scaling

Baseline performance cf. [Harrison et al ’15]

average total queue-size:

$$\mathbb{E} \left[ \sum_i Q_i \right] \leq \frac{M}{(1 - \rho)}$$

tail probability of total queue-size:

$$\mathbb{P} \left( \sum_i Q_i \geq t \right) \approx \exp \left( - (1 - \rho)t \right), \quad \text{for } \rho \approx 1, \text{ large } t.$$
Desired Queue-size scaling

Baseline performance cf. [Harrison et al ’15]

average total queue-size:

\[
\limsup_{N \to \infty, \rho \to 1_+} \sup_{\lambda : \rho(\lambda) = \rho} (1 - \rho) \mathbb{E}[\sum_i Q_i] \leq M.
\]

tail probability of total queue-size:

\[
\limsup_{N \to \infty, \rho \to 1_+} \sup_{\lambda : \rho(\lambda) = \rho} \limsup_t \frac{1}{t(1 - \rho)} \log \mathbb{P}(\sum_i Q_i > t) \leq -\text{const}.
\]

Note: order of limits $\rho \to 1_+$ followed by $N \to \infty$
Baseline performance cf. [Harrison et al '15]

average total queue-size: $N = n^2$ and $M = 2n$

for switch: $(1 - \rho) \mathbb{E}\left[ \sum_i Q_i \right] \leq 2n$

tail probability of total queue-size:

behaves like an $M/D/1$ queue for any system size!
Computation

Scheduling or allocation policy

finds allocation or schedule each time instance

Tractable computation

to find a schedule with queue-sizes $Q = [Q_i]$

total computation should be polynomial in $M$, $N$ and $\log \|Q\|_\infty$
Computation

Tractable computation

to find a schedule with queue-sizes \( Q = [Q_i] \)

total computation should be polynomial in \( M, N \) and \( \log \|Q\|_{\infty} \)

Can not expect exponential better dependence on \( M \)

while retaining good queue-size scaling [Shah-Tse-Tsitsiklis ’12]

With baseline performance: tractable computation

on average, polynomial in \( M, N \) and \( \log(1/(1 - \rho)) \)
Question

Does there exist scheduling/allocation policy

Baseline performance and tractable computation

For any switched and/or bandwidth sharing network

How are these networks related?

How well do max weight and prop. fair perform?

Consider other policy?
Switched and bandwidth sharing networks

Feasibility region of a switched network

convex hull of $\mathcal{S}$

can be represented

$$\lambda \in \mathbb{R}^{N}_{\geq 0} : A\lambda \leq b, \ A \in \mathbb{R}^{M \times N}_{\geq 0}, \ b \in \mathbb{R}^{M}_{> 0}$$

Corresponding “relaxed” bandwidth-sharing network

N routes/flows

M links

capacity of link $i$ is $b_i, \ 1 \leq i \leq M$

Rate allocation from convex hull of $\mathcal{S}$
Relaxation of switch

Input 1

Input 2

Output

Output
Switched network vs its relaxation

Both are single-hop networks

Both have same number of queues
    with identical arrival processes

Only difference is in scheduling constraints

switched network operates in discrete time
    can choose one of finitely many schedules

bandwidth sharing network operates in continuous time
    can choose any of the convex combination of schedules as feasible rate allocation
Switched network vs it’s relaxation

Any policy in a switched network

is a valid policy for it’s relaxation

but not vice versa

Any policy for bandwidth sharing network

with baseline performance leads to a policy for

corresponding switched network with baseline performance

[Shah-Walton-Zhong ’12, ’14]
Explanation via an example

\[
\begin{pmatrix}
\frac{1}{2} & \frac{1}{2} \\
\frac{1}{2} & \frac{1}{2}
\end{pmatrix}
\]

\[
\begin{pmatrix}
0 \\
0 
\end{pmatrix}
\]
Explanation via an example

\[
\begin{bmatrix}
\frac{1}{4} & \frac{3}{4} \\
\frac{3}{4} & \frac{1}{4}
\end{bmatrix}
+ \begin{bmatrix}
\frac{1}{2} & \frac{1}{2} \\
\frac{1}{2} & \frac{1}{2}
\end{bmatrix}
\begin{bmatrix}
\frac{1}{4} & \frac{3}{4} \\
\frac{3}{4} & \frac{1}{4}
\end{bmatrix}
\]
Explanation via an example

Time 2
\[
\begin{bmatrix}
\frac{1}{4} & \frac{3}{4} \\
\frac{3}{4} & \frac{1}{4}
\end{bmatrix}
\]

\[
\begin{bmatrix}
\frac{3}{4} & \frac{5}{4} \\
\frac{5}{4} & \frac{3}{4}
\end{bmatrix}
\begin{bmatrix}
0 & 1 \\
1 & 0
\end{bmatrix}
\]

\[
\begin{bmatrix}
0 & 1 \\
1 & 0
\end{bmatrix}
\]
Explanation via an example

Time 2

\[
\begin{bmatrix}
\frac{1}{4} & \frac{3}{4} \\
\frac{3}{4} & \frac{1}{4}
\end{bmatrix}
\]

\[
\begin{bmatrix}
\frac{3}{4} & \frac{1}{4} \\
\frac{1}{4} & \frac{3}{4}
\end{bmatrix}
\]

Lag

\[
\begin{bmatrix}
0 & 1 \\
1 & 0
\end{bmatrix}
\]
Explanation via an example

Time 3, etc

Lag

\[
\begin{bmatrix}
\frac{1}{2} & \frac{1}{2} \\
\frac{1}{2} & \frac{1}{2}
\end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & 0 \end{bmatrix} + \frac{1}{2} \begin{bmatrix} 0 & 1 \end{bmatrix}
\]

⇒ No matching
Explanation via an example

Time 3, etc

\[ \begin{bmatrix} \frac{3}{4} & \frac{5}{4} \\ \frac{5}{4} & \frac{3}{4} \end{bmatrix} = \frac{3}{4} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \frac{5}{4} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \]

\[ \Rightarrow \text{pick} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \]
Explanation via an example

Time 3, etc

\[
\begin{bmatrix}
\frac{7}{4} & \frac{5}{4} \\
\frac{5}{4} & \frac{7}{4}
\end{bmatrix} = \frac{7}{4} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \frac{5}{4} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}
\]

\[\Rightarrow \text{pick} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\]
Lemma. Total lag is at most $n^2$.

That is, difference in total q-size is at most $n^3$. 

Explanation via an example
So do we have such a policy?

<table>
<thead>
<tr>
<th>Policy</th>
<th>Baseline Performance</th>
<th>Tractable Computation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max Weight [Tassiulas-Ephremides ’92]</td>
<td>Not Clear (may be: Srikant’s Talk)</td>
<td>Yes</td>
</tr>
<tr>
<td>Store-and-Forward [Massoulie, Bonald-Proutiere ’03]</td>
<td>Yes [Zachary ’08]</td>
<td>Not Clear</td>
</tr>
</tbody>
</table>
Maximum weight policy

Computation using Ellipsoid algorithm

can evaluate membership in the convex hull of $S$

using computations polynomial in $M, N$

by Grotchel-Lovasz-Schrijver ‘79

can solve for maximum weight solution

using computation polynomial in $M, N$ and $\log Q$
Maximum weight policy

Queue-size scaling

Heavy traffic approximation in hope of obtaining crisp characterization
Stolyar ’04, Dai-Lin ’07, Shah-Wischik ’06 ’12, Kang-Williams ’06

Recent result by Maguluri and Srikant (talk coming up)

For switch, under constant fraction of nodes saturated

\[
\limsup_{N \to \infty, \rho \to 1} \sup_{\lambda: \rho(\lambda) = \rho} (1 - \rho) \mathbb{E}\left[ \sum_{ij} Q_{ij} \right] \leq 2n.
\]

Beyond this special case, it remains open
Maximum weight policy

Tail probability

[Stolyar ’08][Venkataramanan-Lin ’10] established an implicit result

max-weight policy optimizes the decay-rate

$$\lim_{t \to \infty} \frac{1}{t} \log \mathbb{P} \left( \sum_i Q_i^2 \geq t \right) = -\theta^*(\lambda).$$

[Shah-Walton-Zhong ’12] characterizes this for a class of networks

But not known for total of queue-sizes
Proportional fair policy

Average queue-size [Kang-Kelly-Lee-Williams ’08]

(nearly) achieves base-line performance in heavy traffic limit

associate independent exponential variable to each link

with appropriate parameter induced by original system

queue-size of a route is equal (in dist.) to

the (weighted) sum of these link variables on it’s route

under “local traffic” condition

pretty general, but not satisfied by switch, for example
An unusual policy for bandwidth sharing network (BSN) derived from an analogy to a multi-class network that has “product-form” stationary distribution achieves the baseline performance exactly!

The insensitivity result holds quite generally
Store and forward allocation

[Massoulie, Bonald-Proutiere ’03]

Analogous multi-class network (MCN)

- each link has a processor sharing queue with its fixed capacity
- each route defines a class
  - jobs of a class travel through servers on the route

Rate allocation

- let total jobs in MCN equal to that in BSN
- allocate rate to route in BSN equal to average departure rate in MCN
Store and forward allocation

Product-form stationary distribution for MCN

queue associated with links are independent

have geometric distribution

with parameter given by induced load on the link

Insensitivity established by [Zachary ’08]

For BSN

total jobs in BSN equal to (in dist.) total jobs in MCN
Store and forward allocation

Queue-size scaling property

average total queue-size:

\[ \limsup_{N \to \infty, \rho \to 1-} \sup_{\lambda: \rho(\lambda) = \rho} (1 - \rho) \mathbb{E}\left[ \sum_i Q_i \right] \leq M. \]

tail probability of total queue-size:

\[ \limsup_{N \to \infty, \rho \to 1-} \sup_{\lambda: \rho(\lambda) = \rho} \limsup_t \frac{1}{t(1 - \rho)} \log \mathbb{P}(\sum_i Q_i > t) \leq -\text{const}. \]
Store and forward allocation

Computation of rate allocation

Requires summation of exponentially many terms

Just like computation of Permanent

has polynomial time approx algorithm [Jerrum-Sinclair-Vigoda ’01]

Actually, there is a much closer connection between these objects

We believe there is a polynomial time approximation algorithm
Summary

Desirable scheduling algorithm

Baseline performance

Tractable computation

Exists for switched network

iff it does so for bandwidth sharing
Summary

Candidates for desirable scheduling algorithm

Maximum weight
  believable, but far for able to establish

Proportional fair allocation
  it achieves for all networks with “local traffic” condition

Store-and-forward
  computation is unresolved
Summary

Baseline performance need is not the best achievable

1. Hierarchical Greedy Performance suggested by [Harrison et al ’15]
   UFOS policy does better than baseline for few examples

2. Independent set model for switched network
   max weight induced avg queue size $O(N^2)$
   but $M$ ought be super-polynomial in $N$
   therefore baseline performance can be terribly off