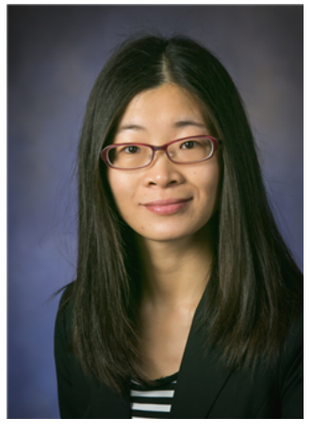
Approximately Reversible Stochastic Processing Networks

Devavrat Shah

Massachusetts Institute of Technology

Acknowledgements:

Jinwoo Shin
KAIST



Qiaomin Xie
Cornell

Yuan Zhong
U Chicago

Prelude

A Canonical Task

Resource Allocation

Sharing resources amongst competing jobs

A Canonical Task

Resource Allocation

- Sharing resources amongst competing jobs
- Subject to resource constraints

A Canonical Task

Resource Allocation

Sharing resources amongst competing jobs

Subject to resource constraints

So that

A Canonical Task

Resource Allocation

Sharing resources amongst competing jobs

Subject to resource constraints

So that

Maximally utilize resources

A Canonical Task

Resource Allocation

Sharing resources amongst competing jobs

Subject to resource constraints

So that

Maximally utilize resources

Minimize latency (completion time) incurred by jobs

A Canonical Task

Resource Allocation

Sharing resources amongst competing jobs

Subject to resource constraints

So that

Maximally utilize resources

Minimize latency (completion time) incurred by jobs

Using *Implementable* policy for allocating resources

A Canonical Task

Resource Allocation

Sharing resources amongst competing jobs

Subject to resource constraints

So that

Maximally utilize resources

Minimize latency (completion time) incurred by jobs

Using *Implementable* policy for allocating resources

It is self evident that they are everywhere

A Canonical Task

Performance Metric

Resource utilization: Capacity

Latency: Queue-size

Implementation: Computation, System constraints

A Canonical Task

Performance Metric

- Resource utilization: Capacity
- Latency: Queue-size
- Implementation: Computation, System constraints

Depends upon

- System load
- System size
- Structure of resource sharing constraints

Ideal Performance

Consider an M/M/1 Queue

Ideal Performance

Consider an M/M/1 Queue

Ideal Performance

Consider an M/M/1 Queue

System load: $\rho = \dfrac{\lambda}{\mu}$

Ideal Performance

Consider an M/M/1 Queue

System load: $\rho = \dfrac{\lambda}{\mu}$

Capacity: $\rho < 1$

Ideal Performance

Consider an M/M/1 Queue

System load: $\rho = \dfrac{\lambda}{\mu}$

Capacity: $\rho < 1$

Queue-size: $\mathbb{E}[Q] = \dfrac{\rho}{1-\rho}$ and $\mathbb{P}(Q > t) \approx \exp(-(1-\rho)t)$

Ideal Performance

Consider an M/M/1 Queue

System load: $\rho = \dfrac{\lambda}{\mu}$

Capacity: $\rho < 1$

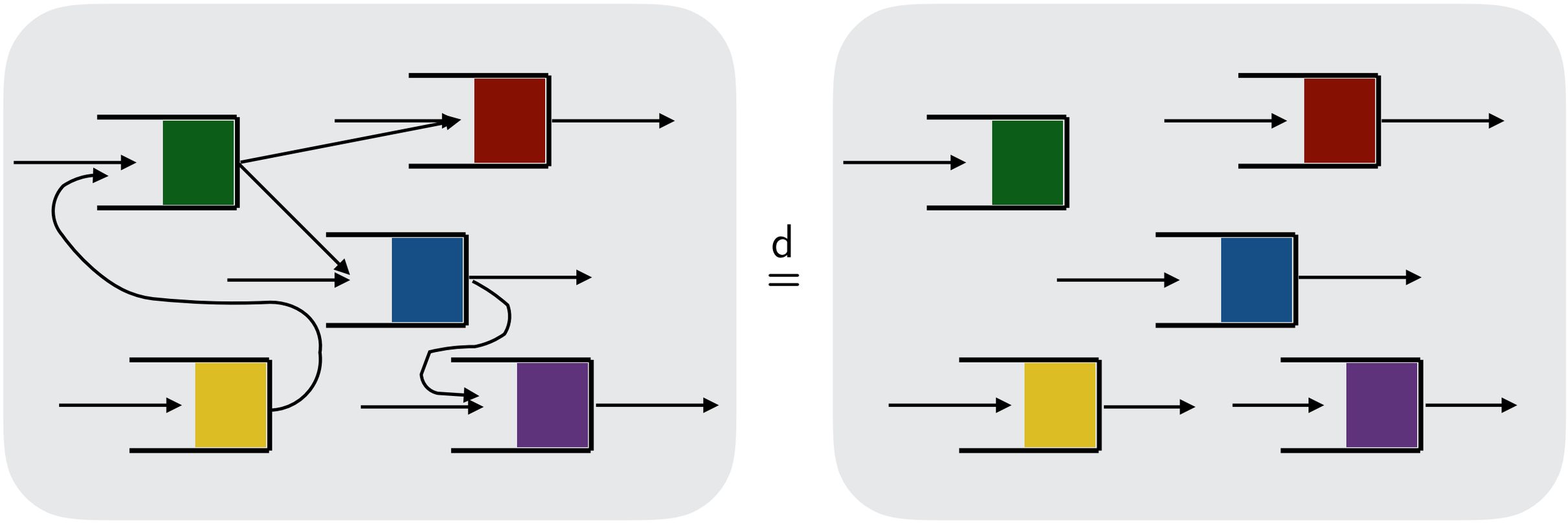
Queue-size: $\mathbb{E}[Q] = \dfrac{\rho}{1-\rho}$ and $\mathbb{P}(Q > t) \approx \exp(-(1-\rho)t)$

First-come-first-serve (work conserving) policy

Ideal Performance: Beyond Simple Example

Jackson (1957)/Basket-Chandy-Munz-Palacois (1975)/Kelly Network (1976)

"product-form" queueing networks



Network of Queues

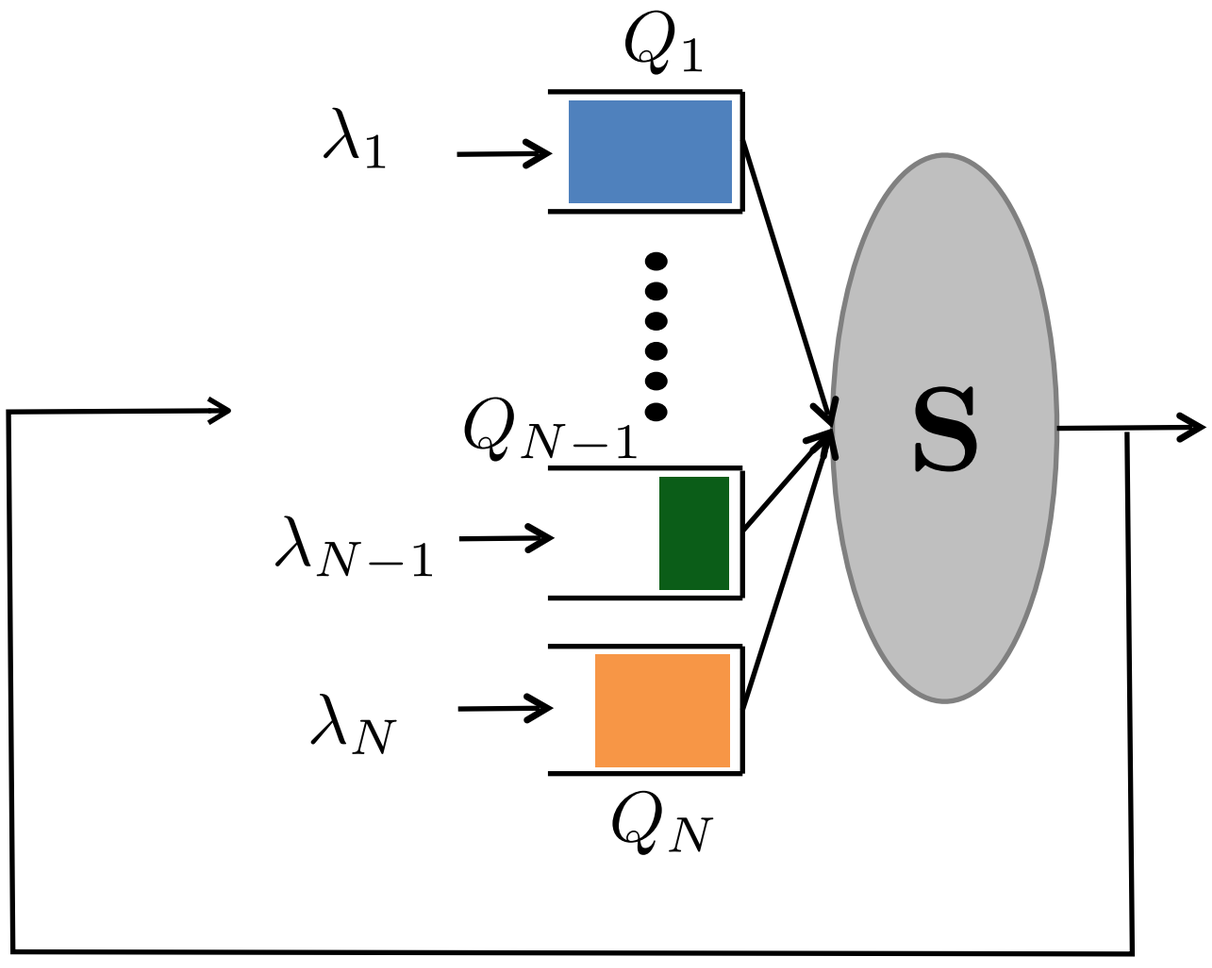
Independent M/M/1 Queues

Policies (LCFS / PS) achieving these are simple

Total queue-size scales linearly with network size - best one can hope for

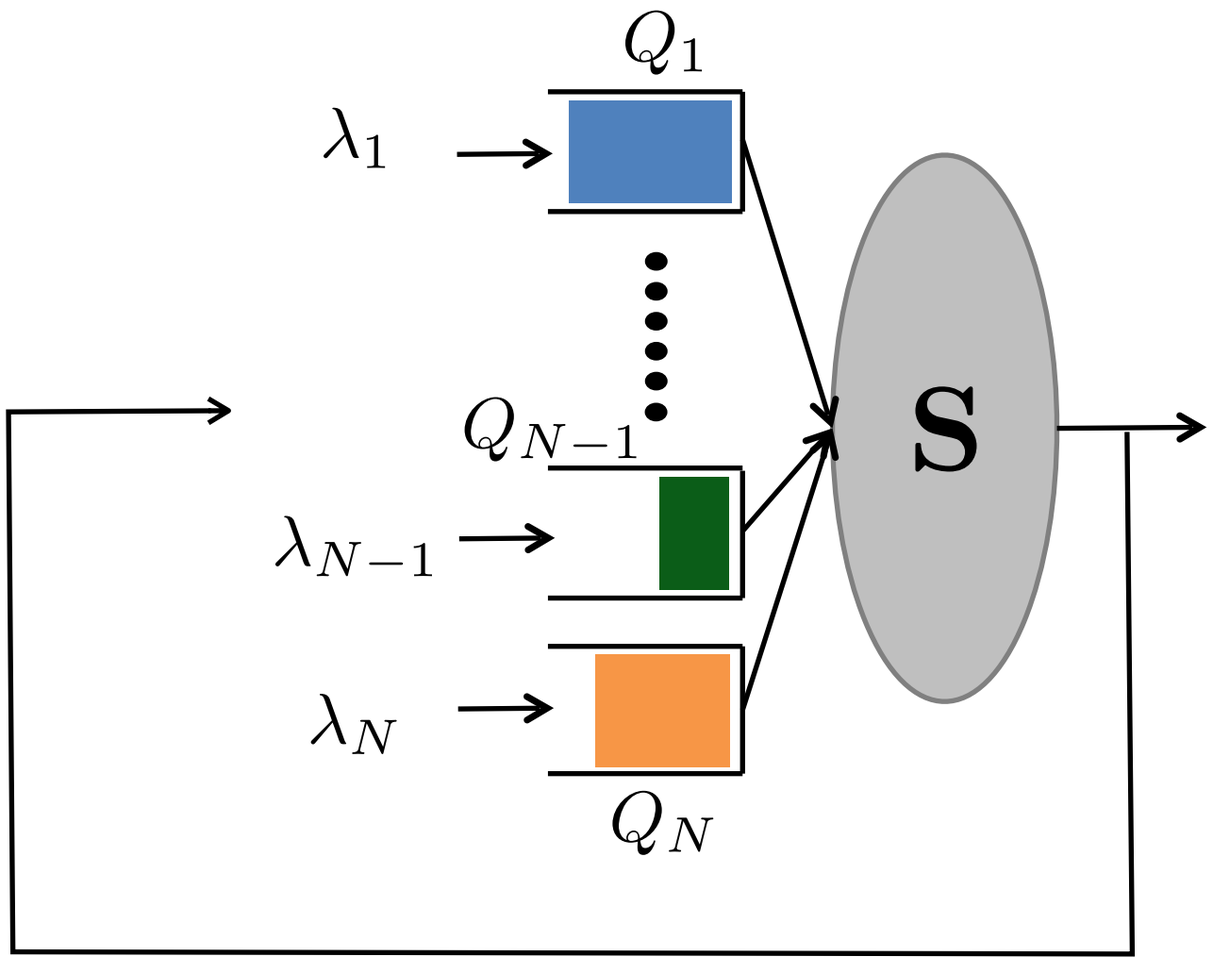
Stochastic Processing Networks

Harrison (2000): A parsimonious model



Stochastic Processing Networks

Harrison (2000): A parsimonious model



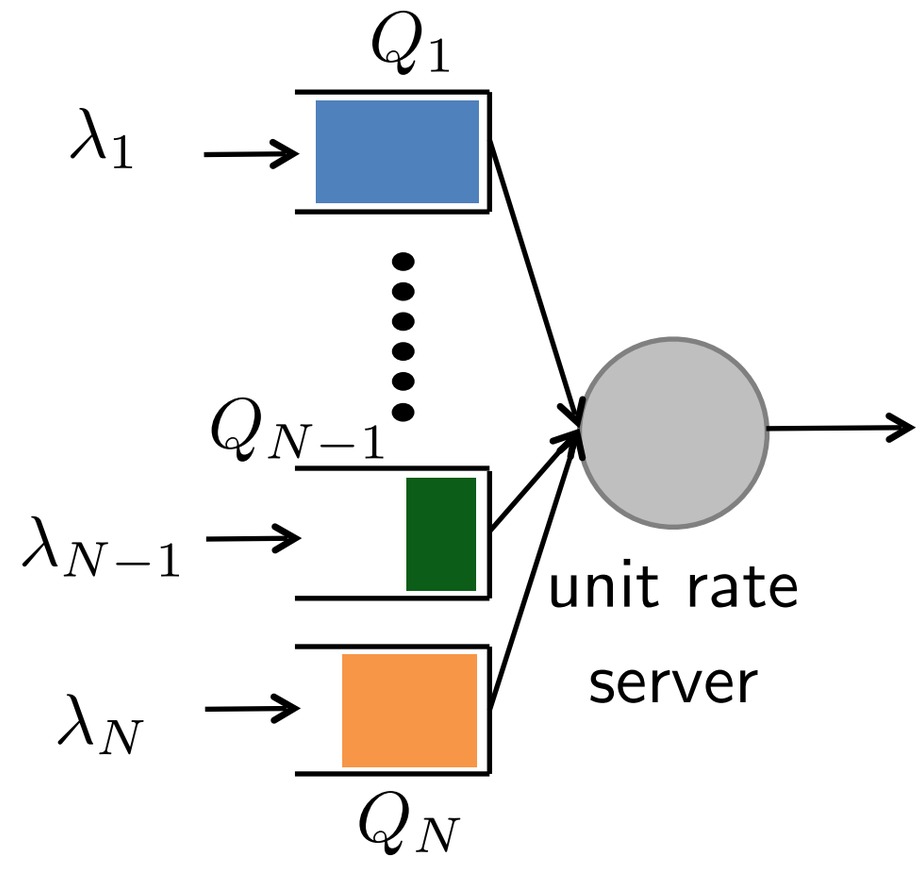
But, *complicated*

Stochastic Processing Networks

Policies

Maximum Weight (MW)(back-pressure) of Tassiulas-Ephremides (1992)

An Example:



Constraint: can serve only one queue at a time

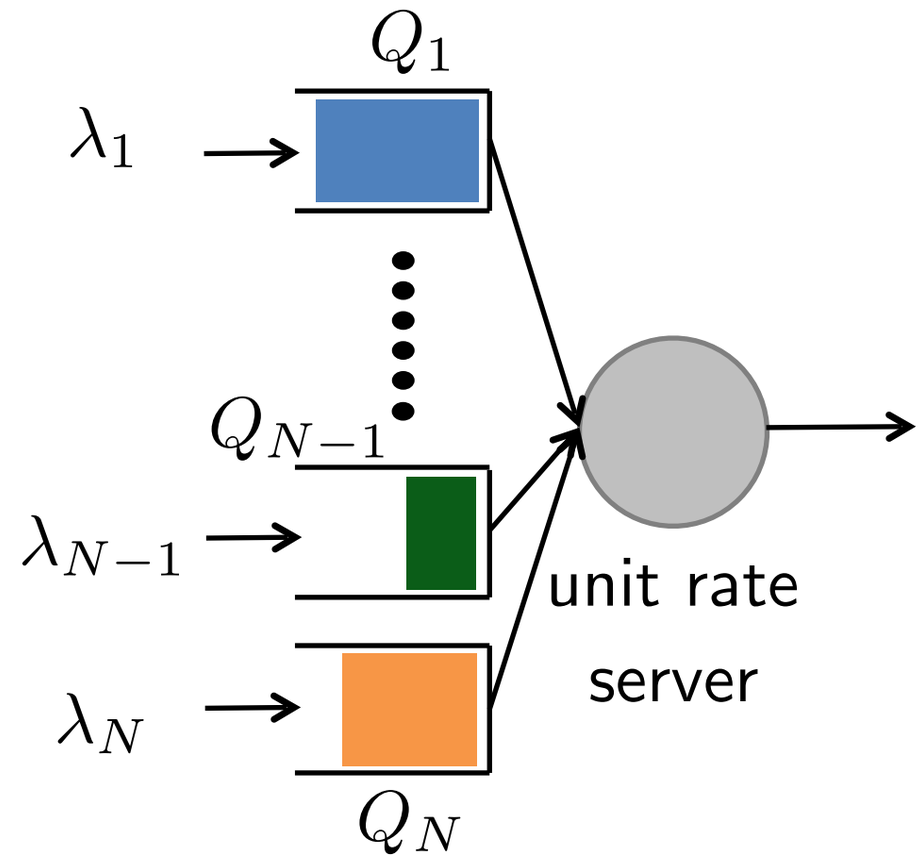
MW: always serve the longest queue

Stochastic Processing Networks

Policies

Proportional Fair (PF) of Kelly-Maullo-Tan (1997)

An Example:



Constraint: service rate bounded by 1, can split in any manner

PF: every non-empty queue is served with positive rate,
longer queue with higher rate

Stochastic Processing Networks

Stochastic Processing Networks

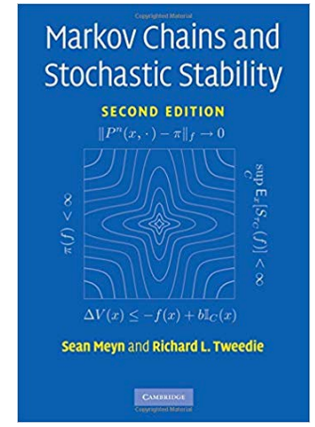
Performance analytic methods

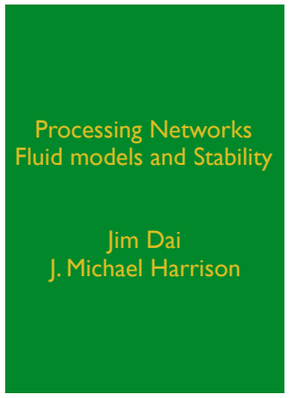
Capacity via positive recurrence of Markov chain

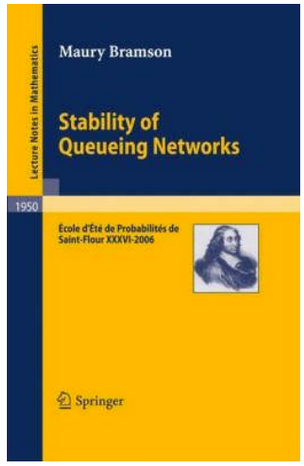
Foster and Lyapunov criterion cf. Meyn-Tweedie (1990s)

Fluid model cf. Dai (1995), Bramson (2008)

+ Dai-Harrison (20XX)





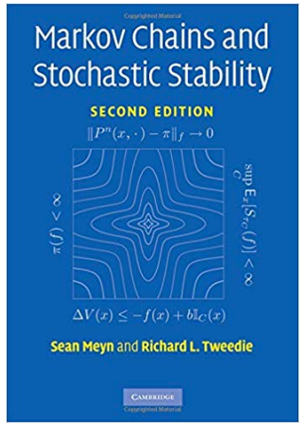


Stochastic Processing Networks

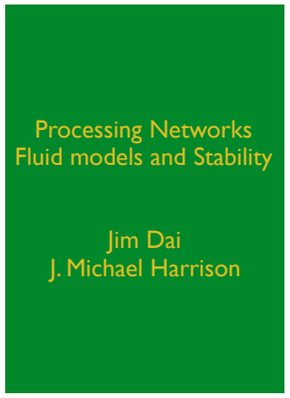
Performance analytic methods

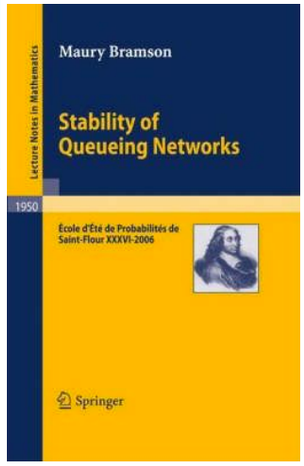
Capacity via positive recurrence of Markov chain

Foster and Lyapunov criterion cf. Meyn-Tweedie (1990s)



Fluid model cf. Dai (1995), Bramson (2008)





+ Dai-Harrison (20XX)

Implication:

Optimal capacity is achieved by

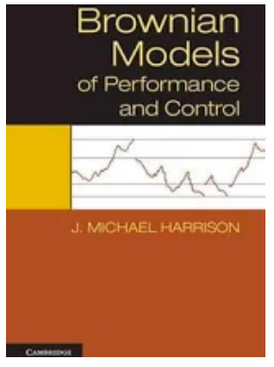
MW (back-pressure) by Tassiulas-Ephremides (1992), Stolyar (2004)

PF by Bonald-Massoulie(2001), de Vecianna-Lee-Konstantopoulos(2001)
Ye (2003)

Stochastic Processing Networks

Stochastic Processing Networks

Performance analytic methods



Queue-size scaling via diffusion or heavy traffic approximation

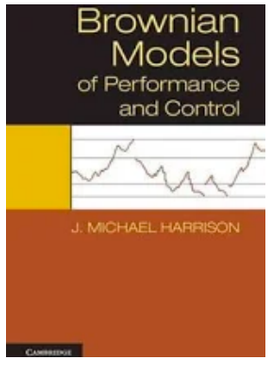
cf. Harrison (1985, 2013)

State-space collapse + Invariance Principle

cf. Bramson (1998), Williams (1998)

State-space collapse + Lyapunov function cf. Maguluri-Srikant (2016)

Stochastic Processing Networks



Performance analytic methods

Queue-size scaling via diffusion or heavy traffic approximation

cf. Harrison (1985, 2013)

State-space collapse + Invariance Principle

cf. Bramson (1998), Williams (1998)

State-space collapse + Lyapunov function cf. Maguluri-Srikant (2016)

Implication:

MW (restricted optimality in various forms)

cf. Stolyar (2004), Shah-Wischik (2006), Maguluri-Srikant (2016)

PF leads to asymptotic "product-form" (for specific networks)

cf. Kang-Kelly-Lee-Williams (2006), Ye-Yao (2012)

Stochastic Processing Networks

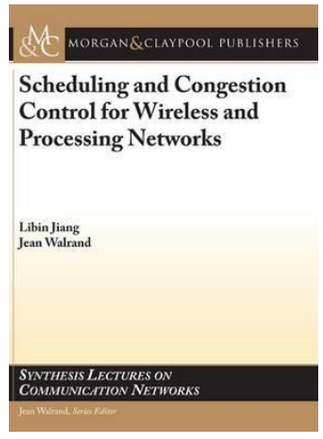
Policy design methods

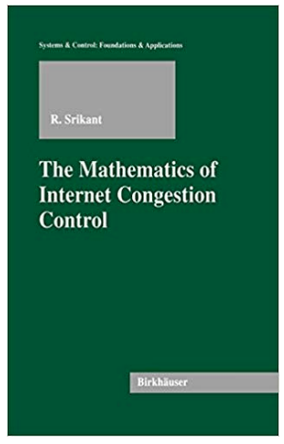
Randomization cf. Tassiulas (1998), Giaccone-Prabhakar-Shah (2002)

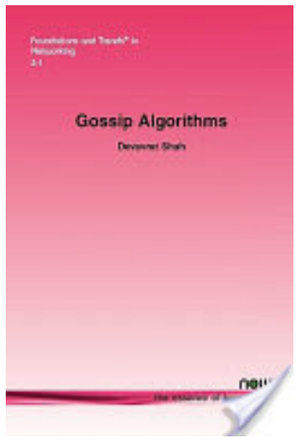
Primal-dual cf. Kelly-Maullo-Tan (1997), Low (1998), Srikant (2000)

Distributed Implementation cf. Modiano-Shah-Zussman (2005)

Message-passing cf. Shah (2006)



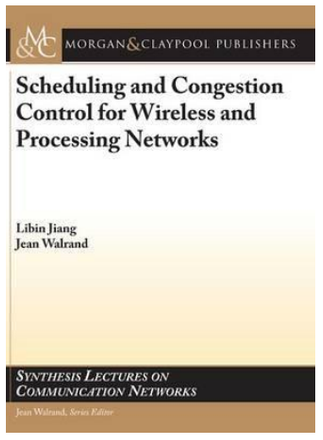


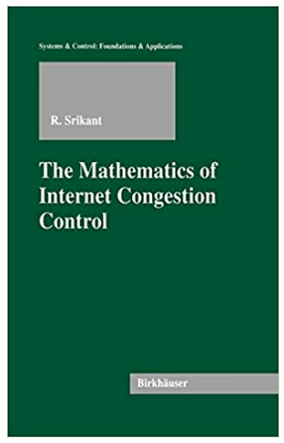


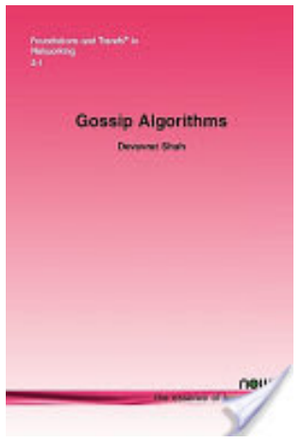
Stochastic Processing Networks

Policy design methods

- Randomization cf. Tassiulas (1998), Giaccone-Prabhakar-Shah (2002)
- Primal-dual cf. Kelly-Maullo-Tan (1997), Low (1998), Srikant (2000)
- Distributed Implementation cf. Modiano-Shah-Zussman (2005)
- Message-passing cf. Shah (2006)







Implication:

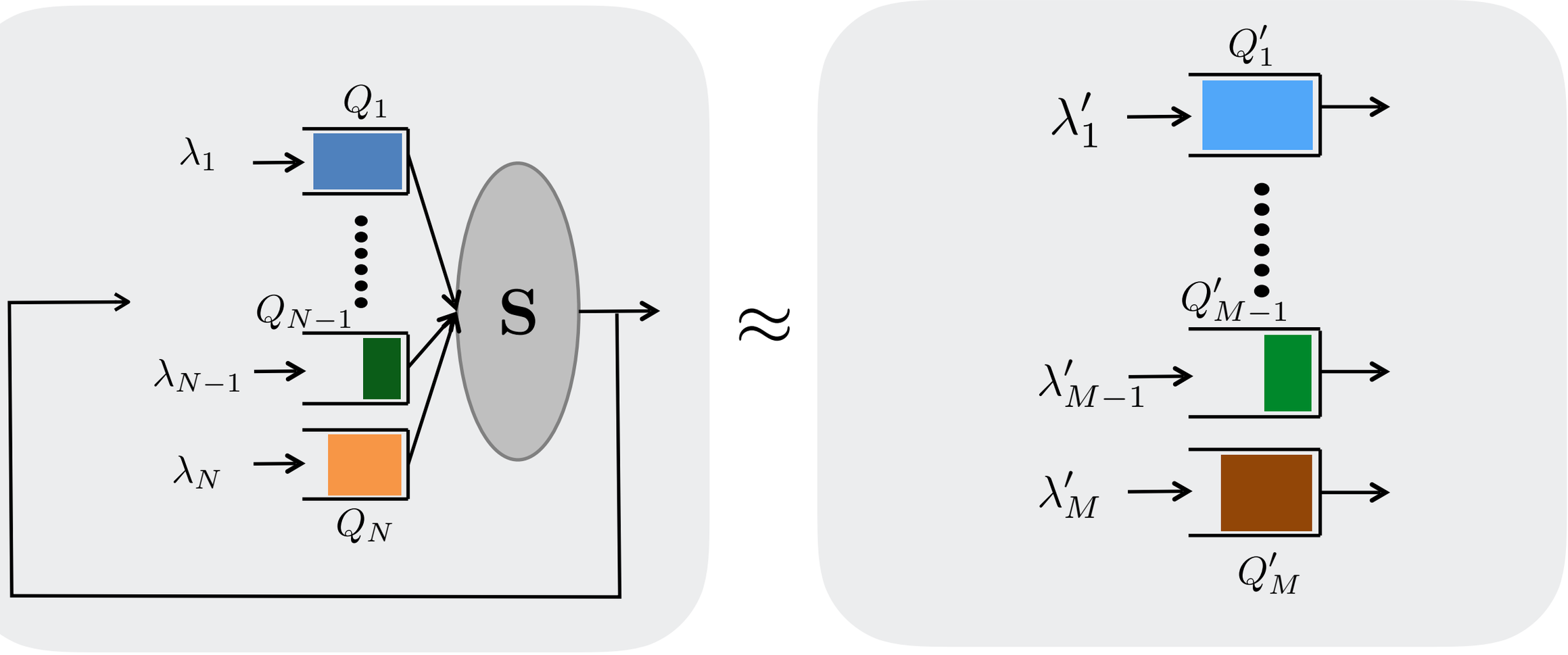
- MW has capacity achieving low-complexity implementation
- PF has iterative, distributed low-complexity implementation

Stochastic Processing Networks: Baseline Performance

An *Interesting* Goal: design resource allocation policy so that

network of queues becomes equivalent to "product-form"

called baseline performance cf. Harrison-Mandayam-Shah-Yang (2014)

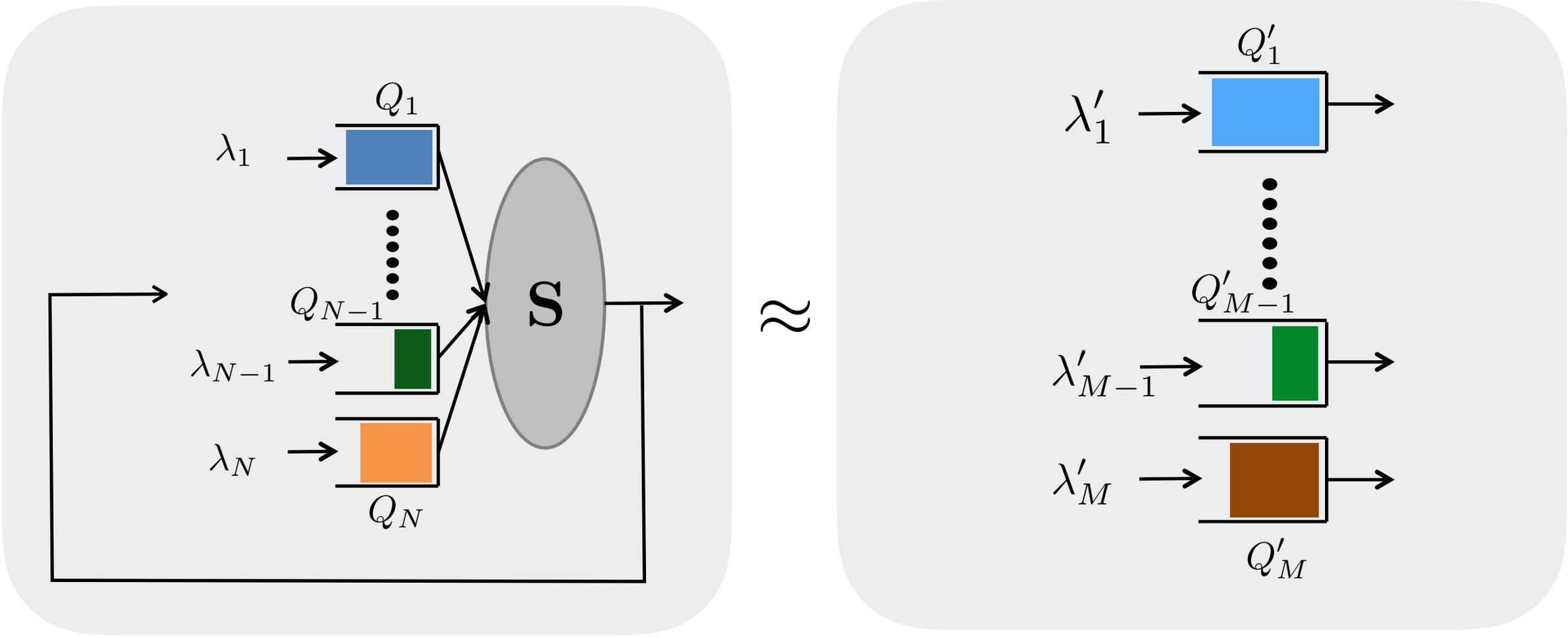


Stochastic Processing Networks: Baseline Performance

An *Interesting* Goal: design resource allocation policy so that

network of queues becomes equivalent to "product-form"

called baseline performance cf. Harrison-Mandayam-Shah-Yang (2014)



Questions:

is it feasible? done with low complexity? is it a good thing to do?

This Tutorial

This Tutorial

We will describe method to achieve this *interesting* goal

For two different objectives

Maximal capacity, minimal queue-size scaling

leads to notion of *baseline* performance

Maximal capacity, fully distributed implementation

This Tutorial

We will describe method to achieve this *interesting* goal

For two different objectives

Maximal capacity, minimal queue-size scaling

leads to notion of *baseline* performance

Maximal capacity, fully distributed implementation

We will discuss

Challenges in achieving maximal capacity, minimal queue-size and fully distributed implementation simultaneously

Relation of baseline performance to extension complexity

Some open directions

Outline

- Remainder of the tutorial
  - Baseline performance
    - Switched network, Single Hop
    - Switched flow network, Multi Hop
  - Distributed implementation
    - Wireless network
- Discussion

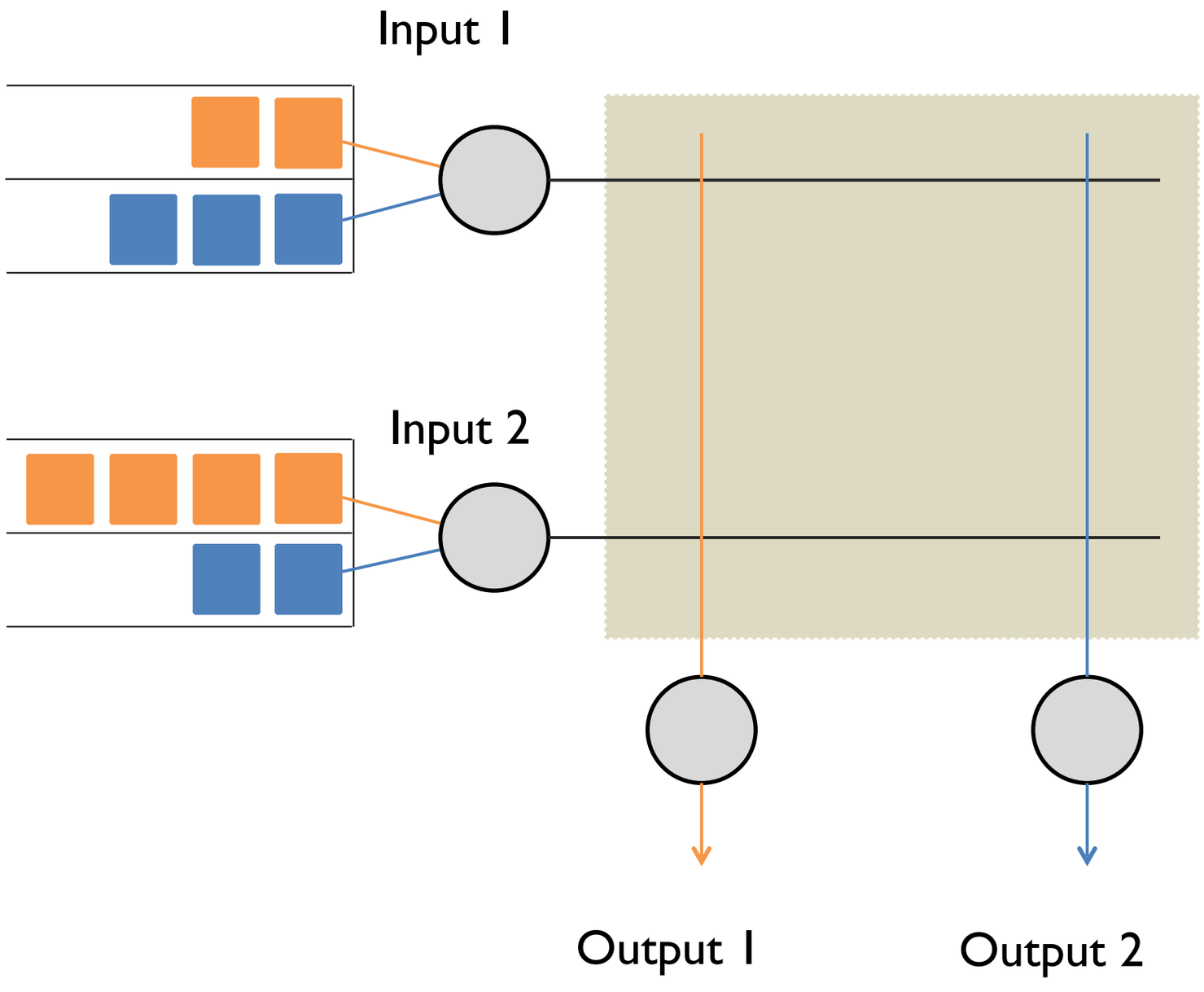
Switched Networks:

Single-hop

D. Shah, N. Walton and Y. Zhong, "Optimal queue-size scaling in switched networks”, The Annals of Applied Probability, 2014

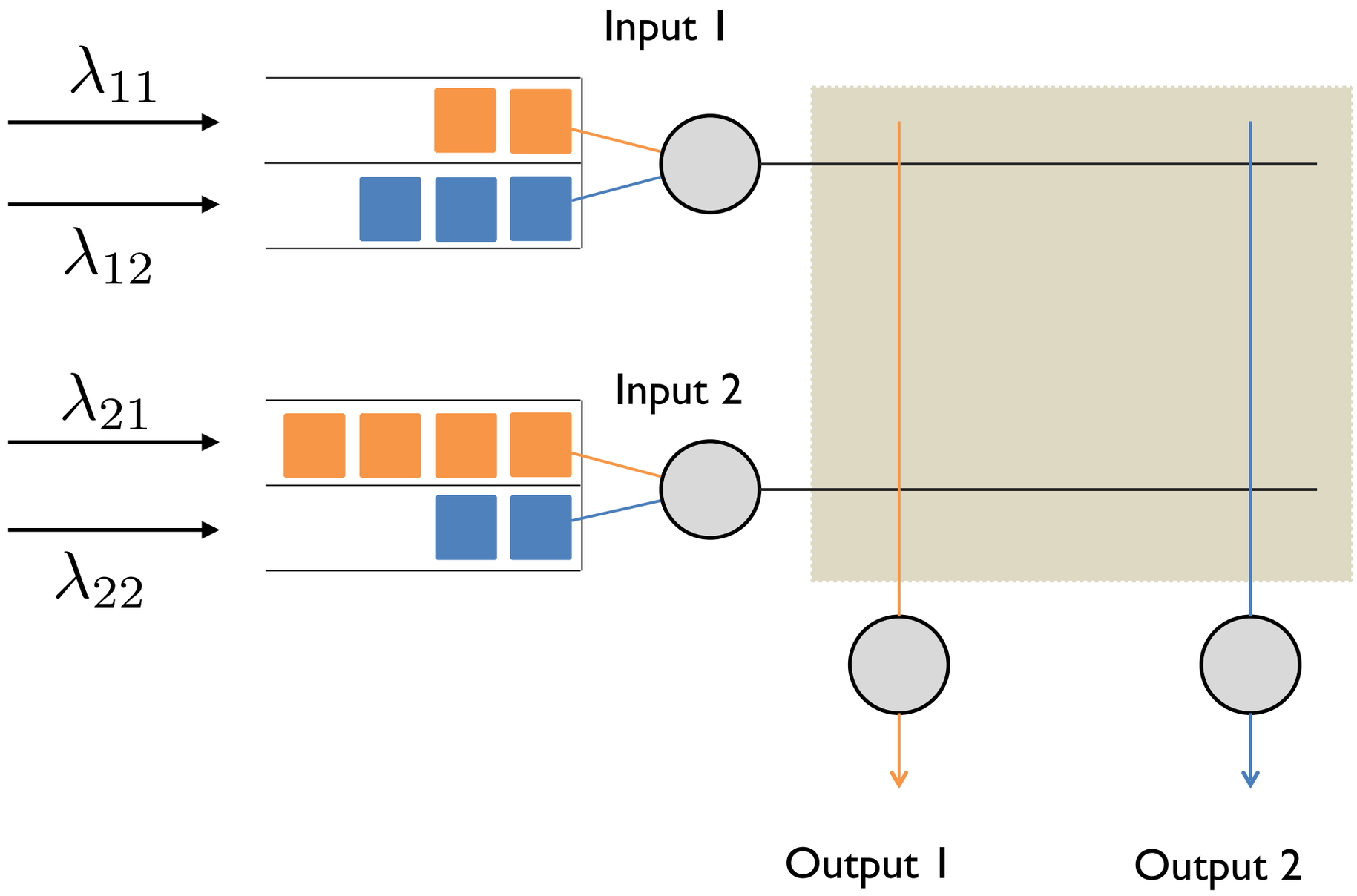
Switched Networks

A useful example: input-queued switch



Switched Networks

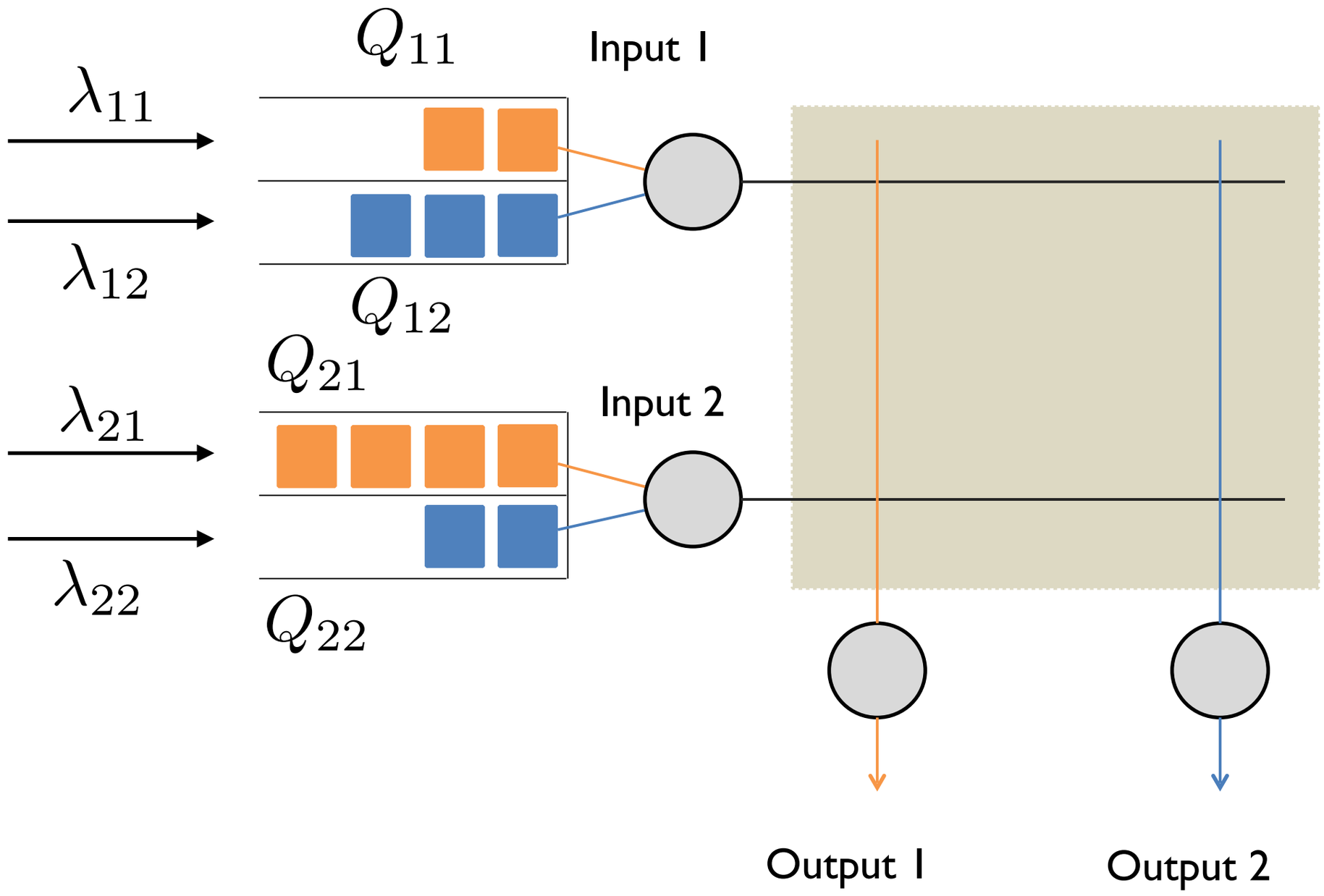
A useful example: input-queued switch



Unit sized packets arrive as per independent Poisson processes

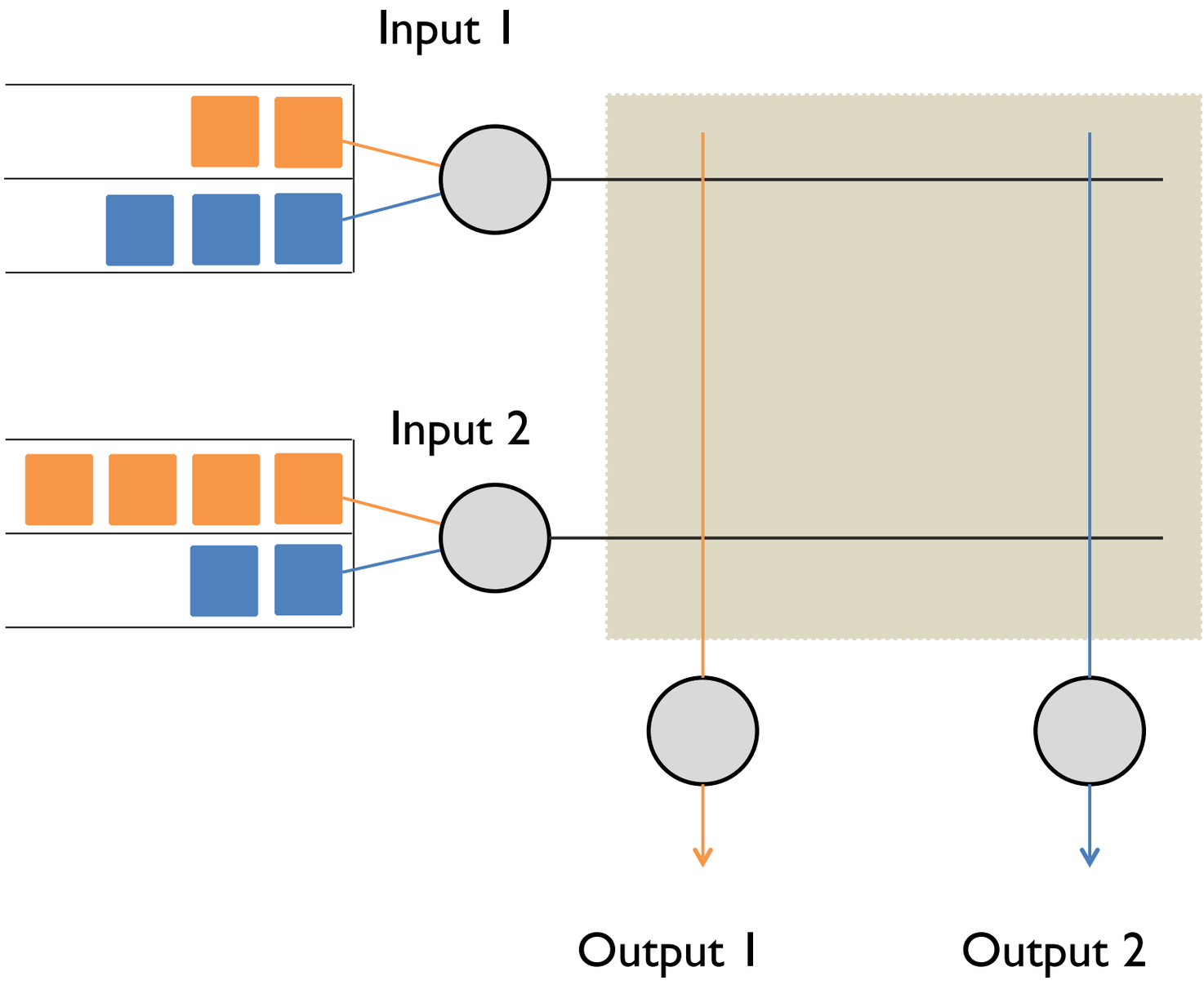
Switched Networks

A useful example: input-queued switch



Unit sized packets arrive as per independent Poisson processes

Switched Networks

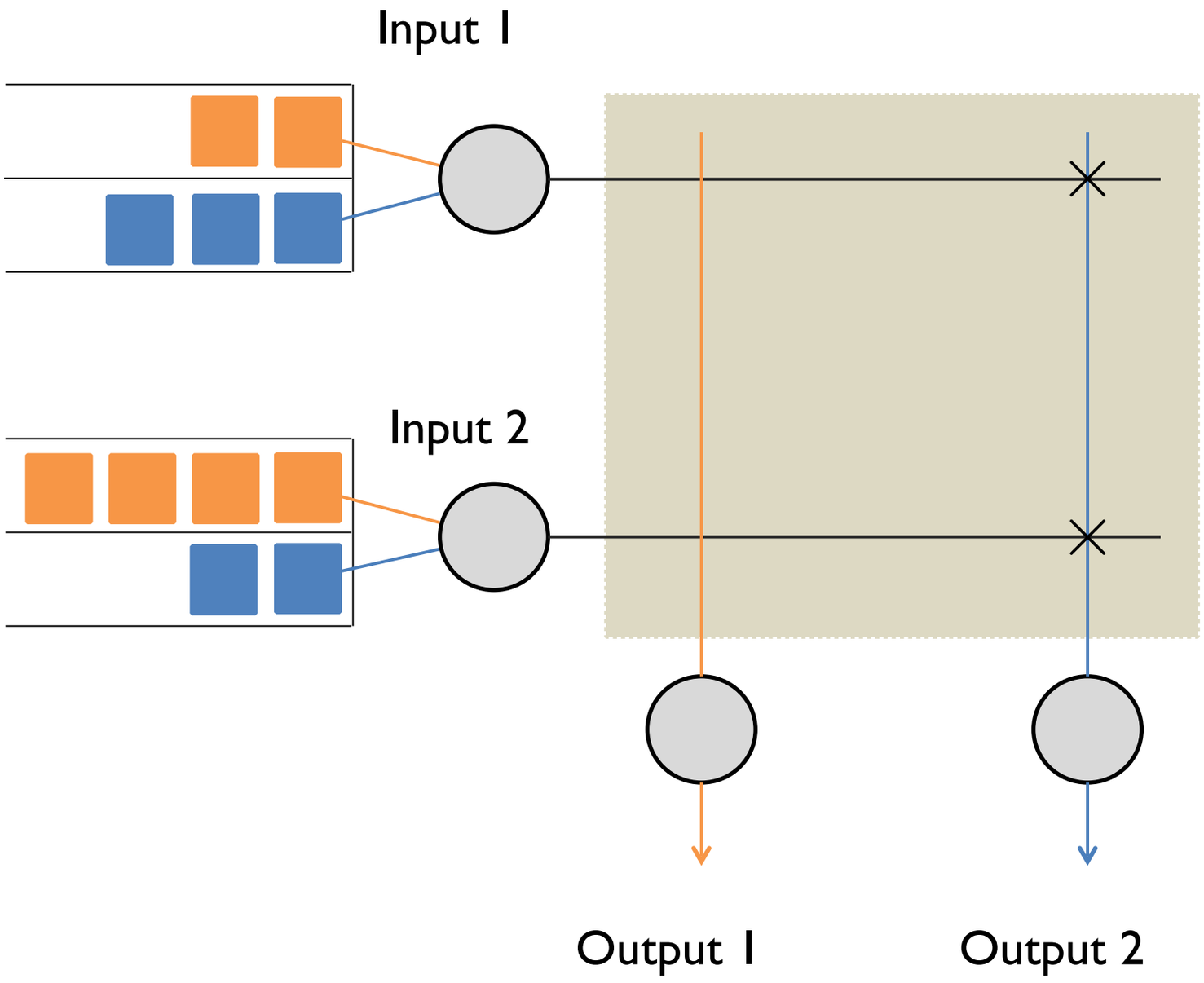


Each integral time instance

each input can send at most one packet

each output can receive at most one packet

Switched Networks

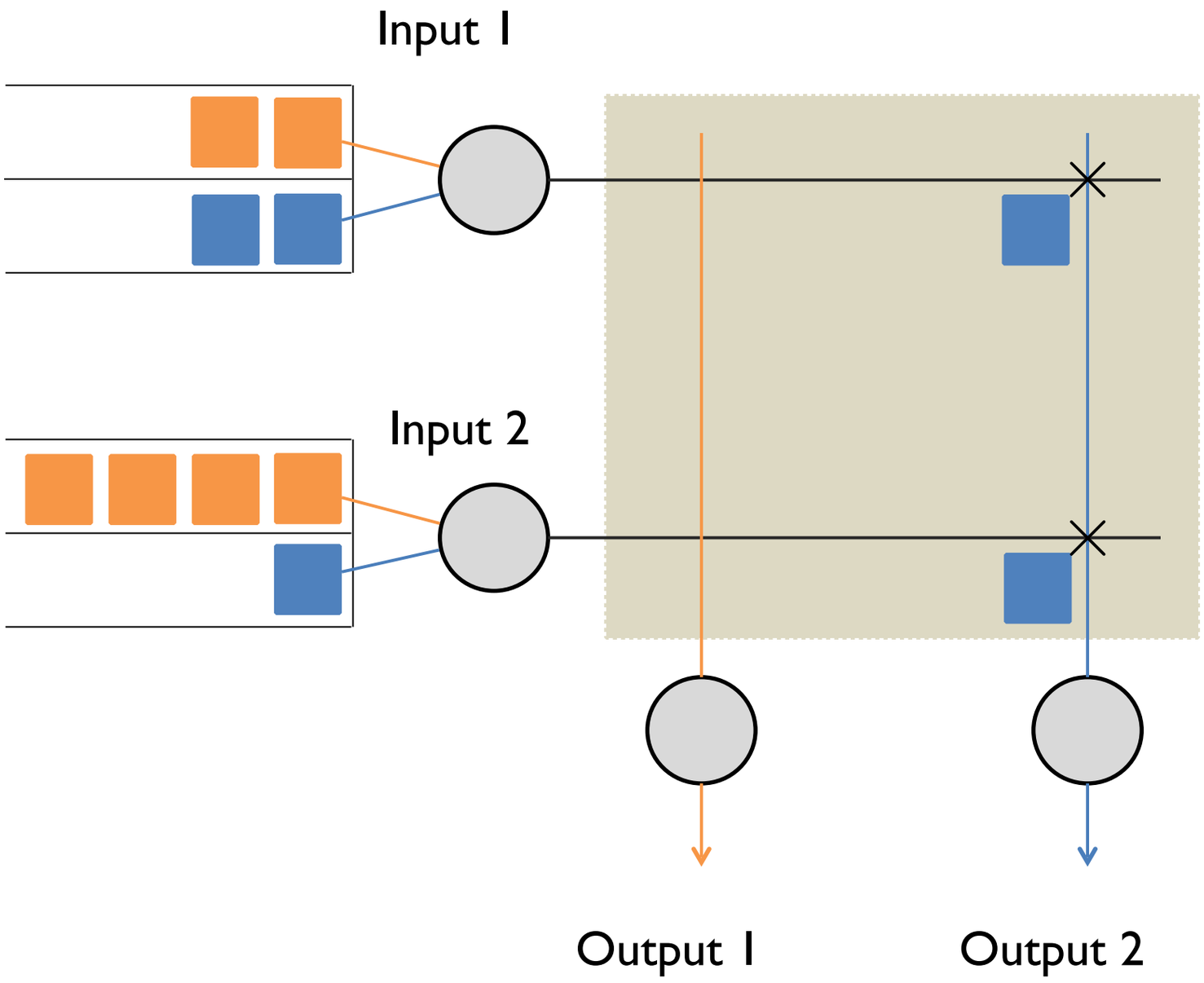


Each integral time instance

each input can send at most one packet

each output can receive at most one packet

Switched Networks

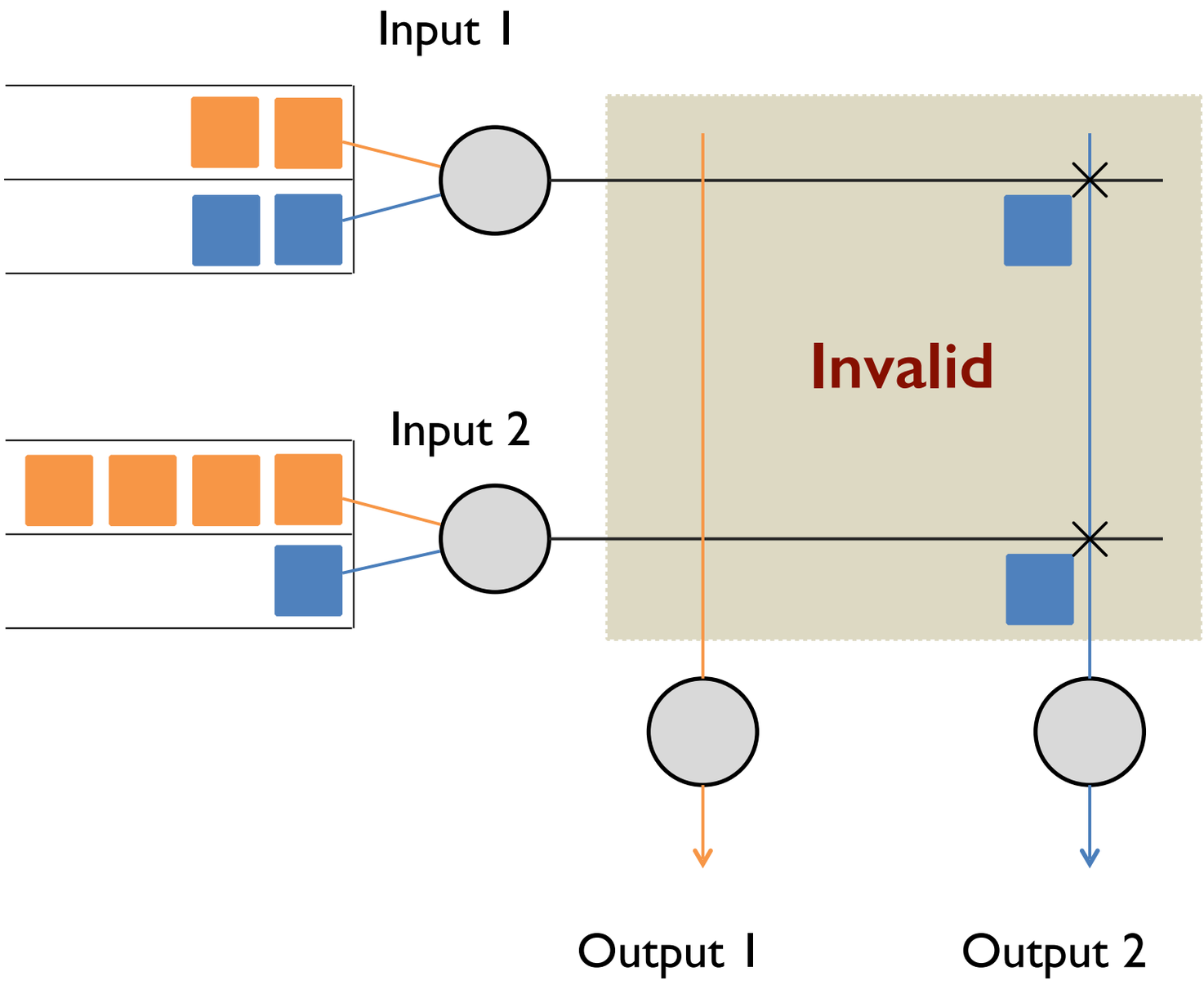


Each integral time instance

each input can send at most one packet

each output can receive at most one packet

Switched Networks

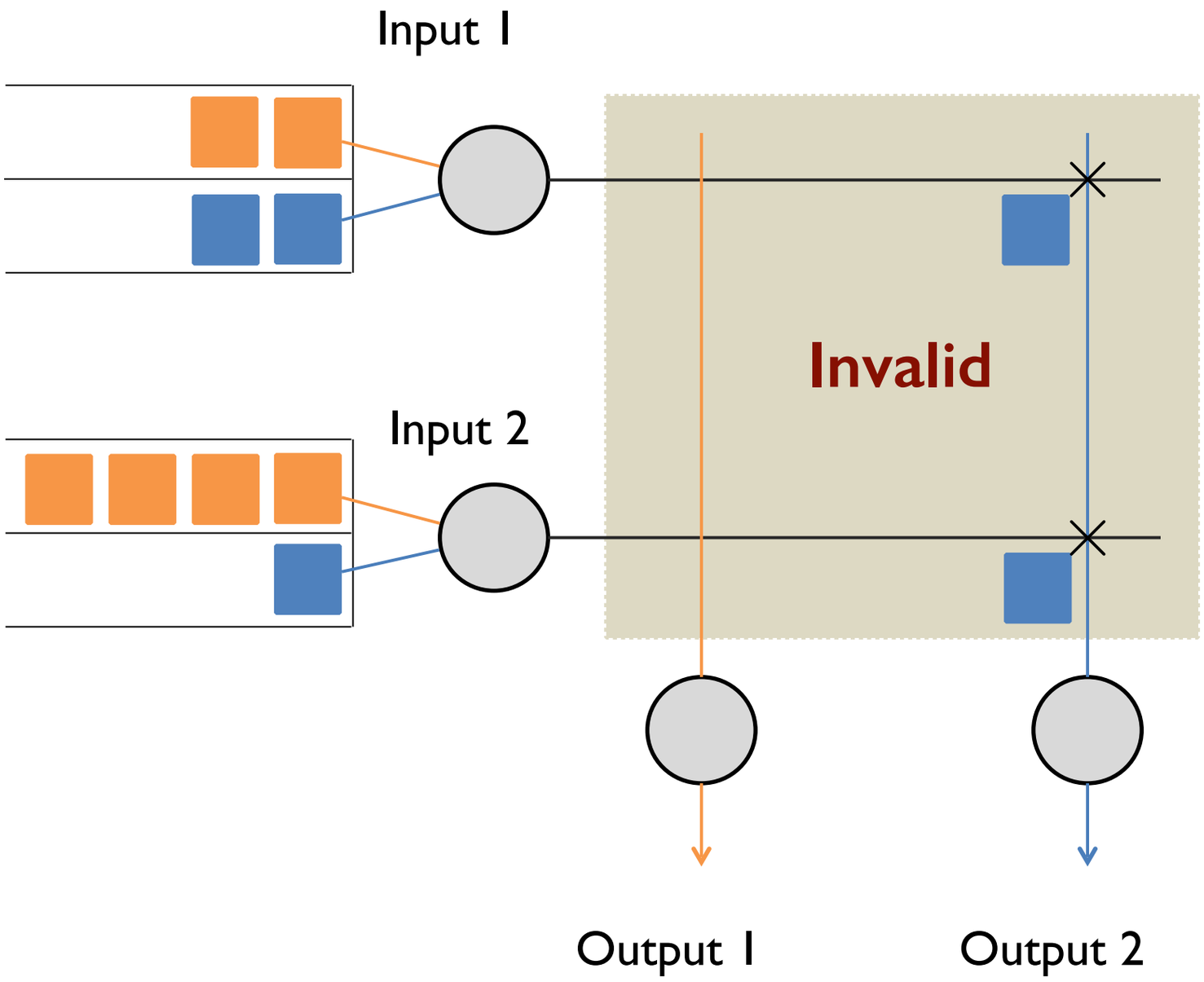


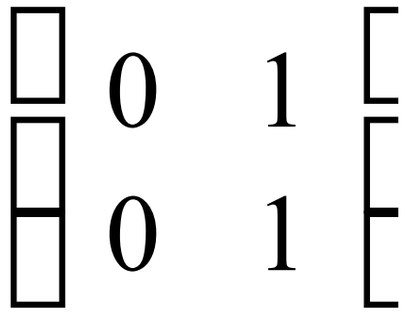
Each integral time instance

each input can send at most one packet

each output can receive at most one packet

Switched Networks



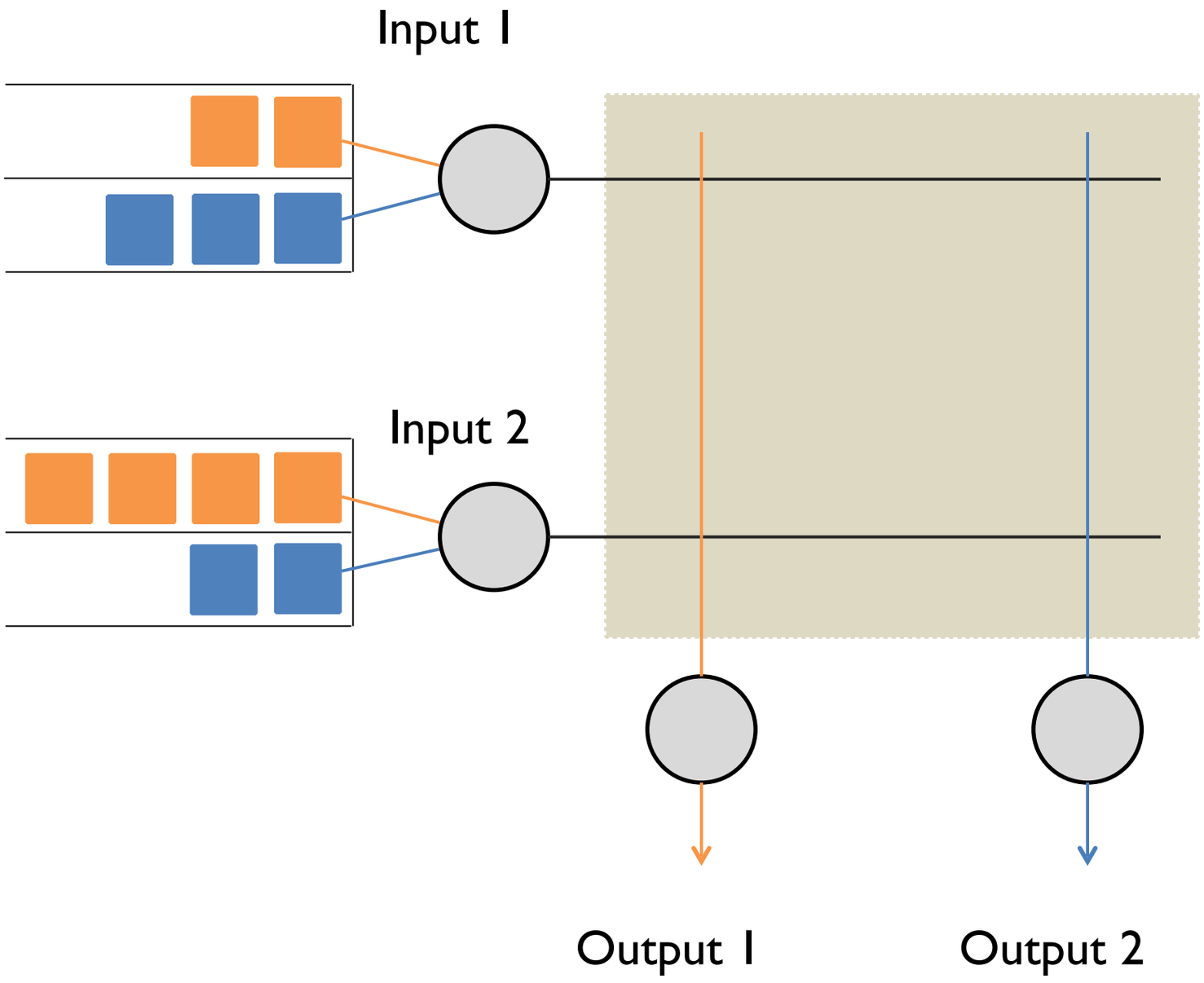


Each integral time instance

each input can send at most one packet

each output can receive at most one packet

Switched Networks

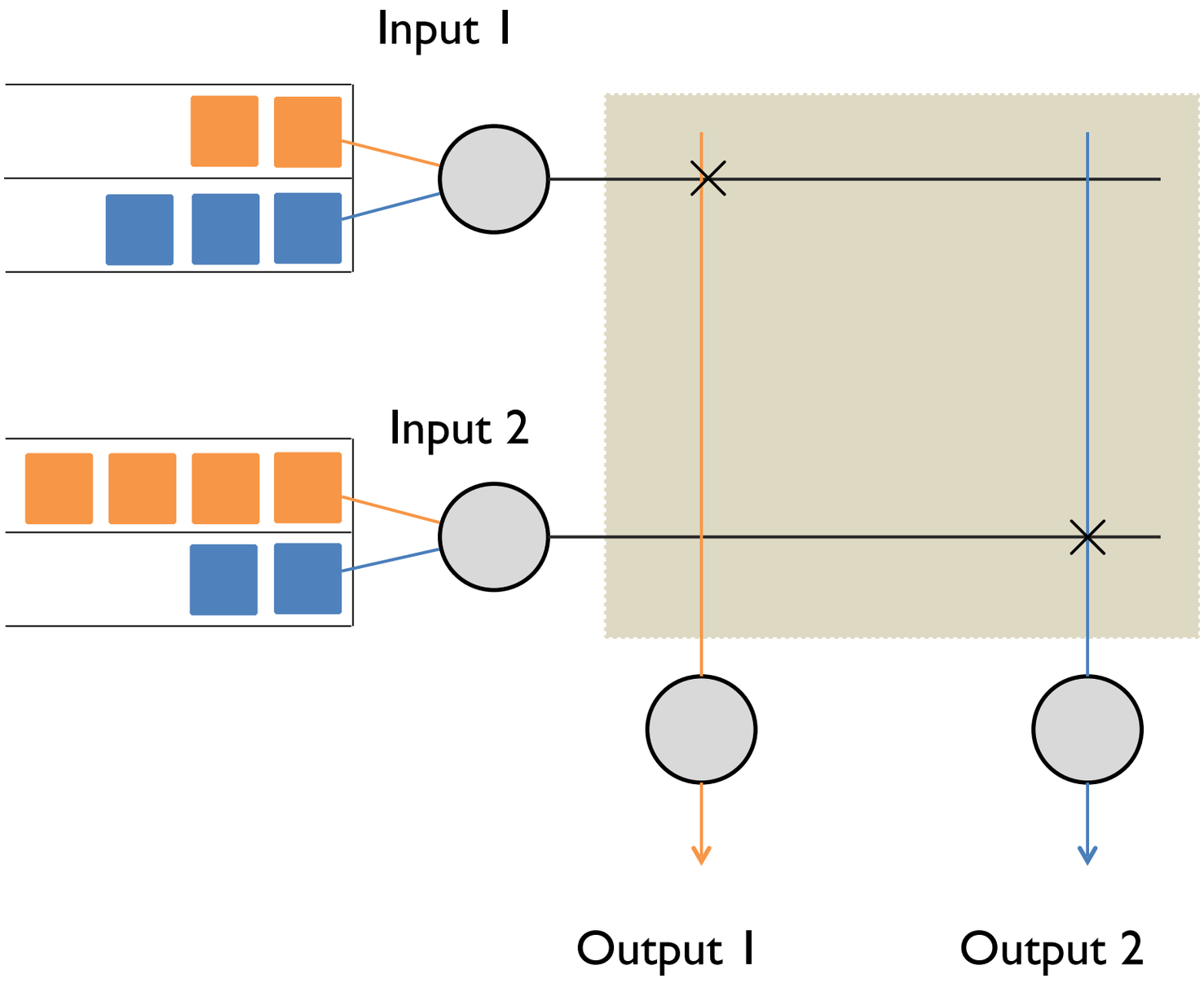


Each integral time instance

each input can send at most one packet

each output can receive at most one packet

Switched Networks

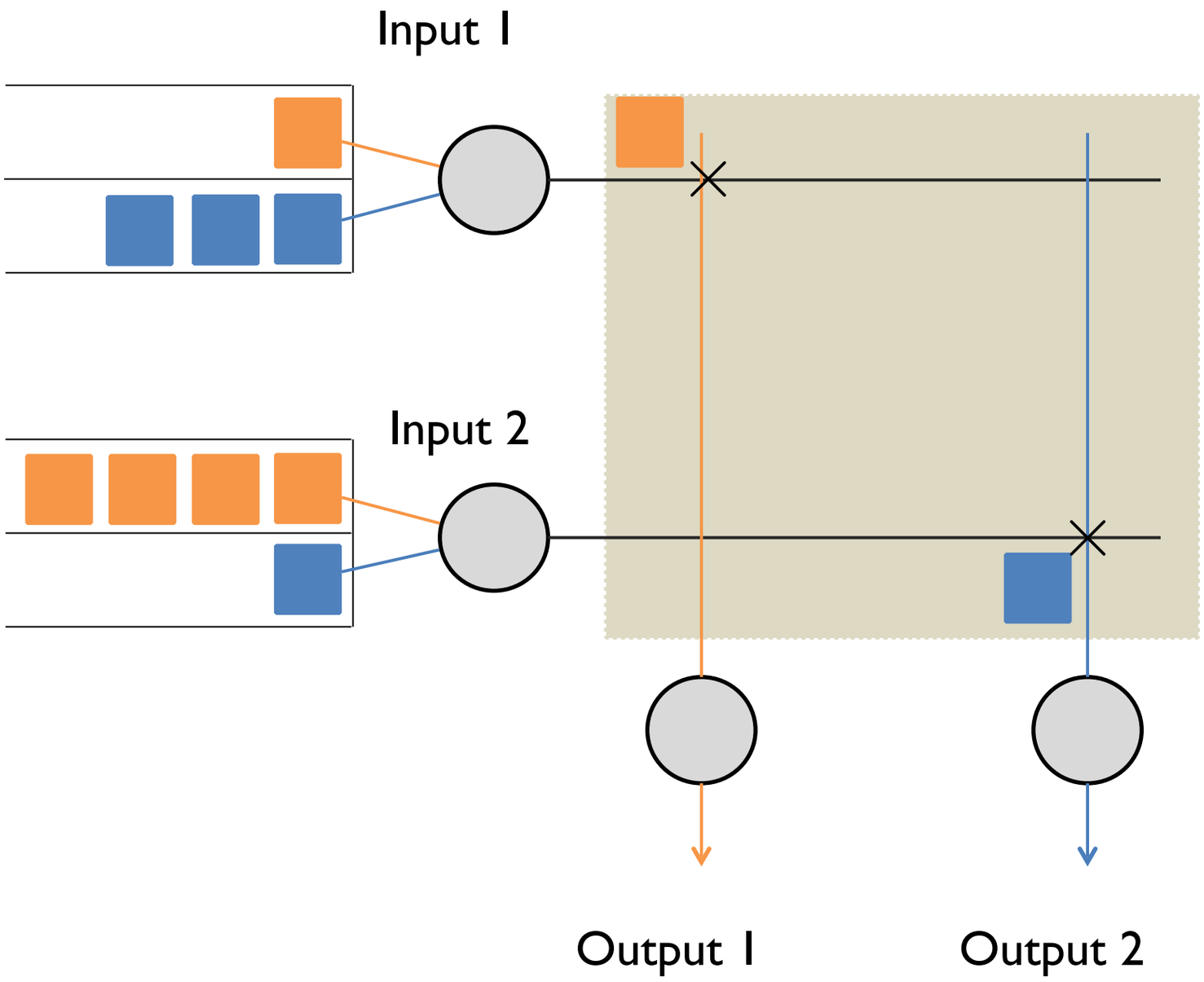


Each integral time instance

each input can send at most one packet

each output can receive at most one packet

Switched Networks

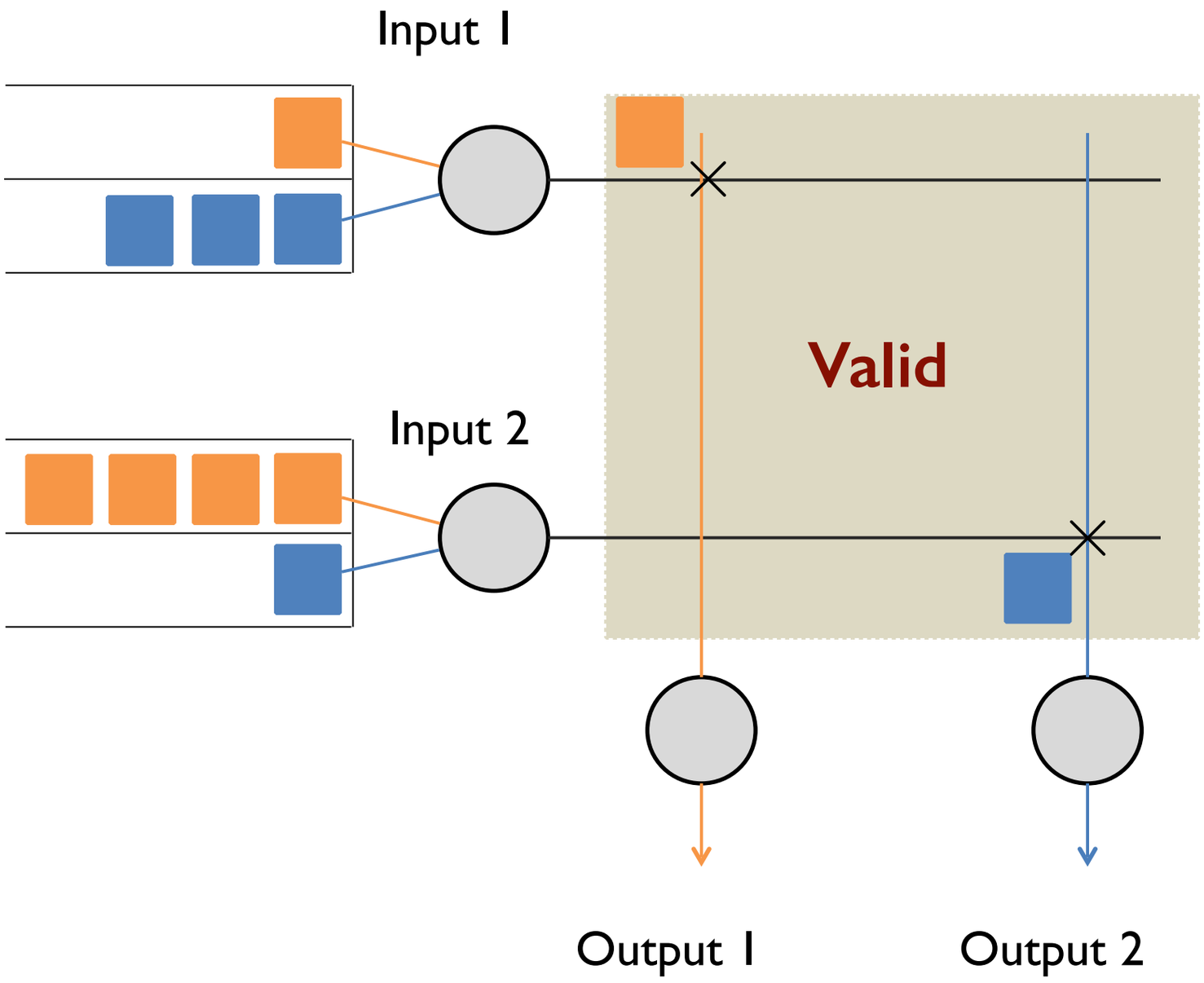


Each integral time instance

each input can send at most one packet

each output can receive at most one packet

Switched Networks



$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Each integral time instance

each input can send at most one packet

each output can receive at most one packet

Switched Networks

The set of feasible rates

convex combination of possible schedules

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

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

$$\text{s.t. } \alpha + \beta \leq 1, \quad \alpha, \ \beta \geq 0.$$

Load of λ : the smallest possible $\alpha + \beta$ satisfying above

Switched Networks

The set of feasible rates

convex combination of possible schedules

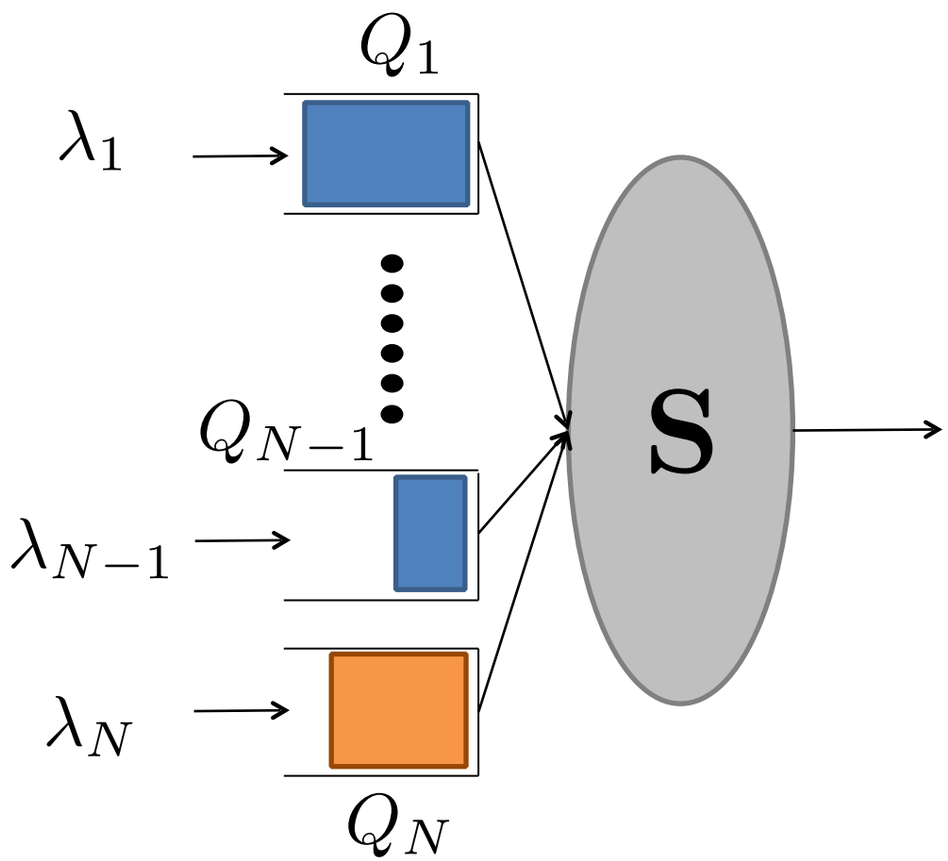
In a 2-port switch, $\lambda = [\lambda_{ij}]$ is feasible iff

$$\lambda_{11} + \lambda_{12} \leq 1 \qquad \lambda_{21} + \lambda_{22} \leq 1$$

$$\lambda_{11} + \lambda_{21} \leq 1 \qquad \lambda_{12} + \lambda_{22} \leq 1$$

Load of λ : maximum of the left-hand-side of the above four terms

Switched Networks



Scheduling constraints

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

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

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

Switched Networks

Capacity region

convex hull of $\mathbb{S}$

can be represented as

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

given a feasible λ, it's load is defined as

$$\rho(\lambda) = \min\{\rho \leq 1 : A\lambda \leq \rho b\}$$

An Open Question

Consider input-queued switch with n inputs / outputs

there exists a policy such that

achieves maximal capacity

computationally efficient (poly-time)

average queue-sizes

$$\mathbb{E}\Big[\sum_{i,j} Q_{ij}\Big] \leq c\frac{n}{1-\rho(\lambda)}$$

where c is a universal constant

Informally around since 2000, formally noted down in Shah-Tsitsiklis-Zhong (2010)

Switched Networks

We describe policy so that switched network becomes

equivalent to "product-form" queueing network

Four step approach

Identify "relaxed" network with feasible actions being convex hull of $\mathbb{S}$

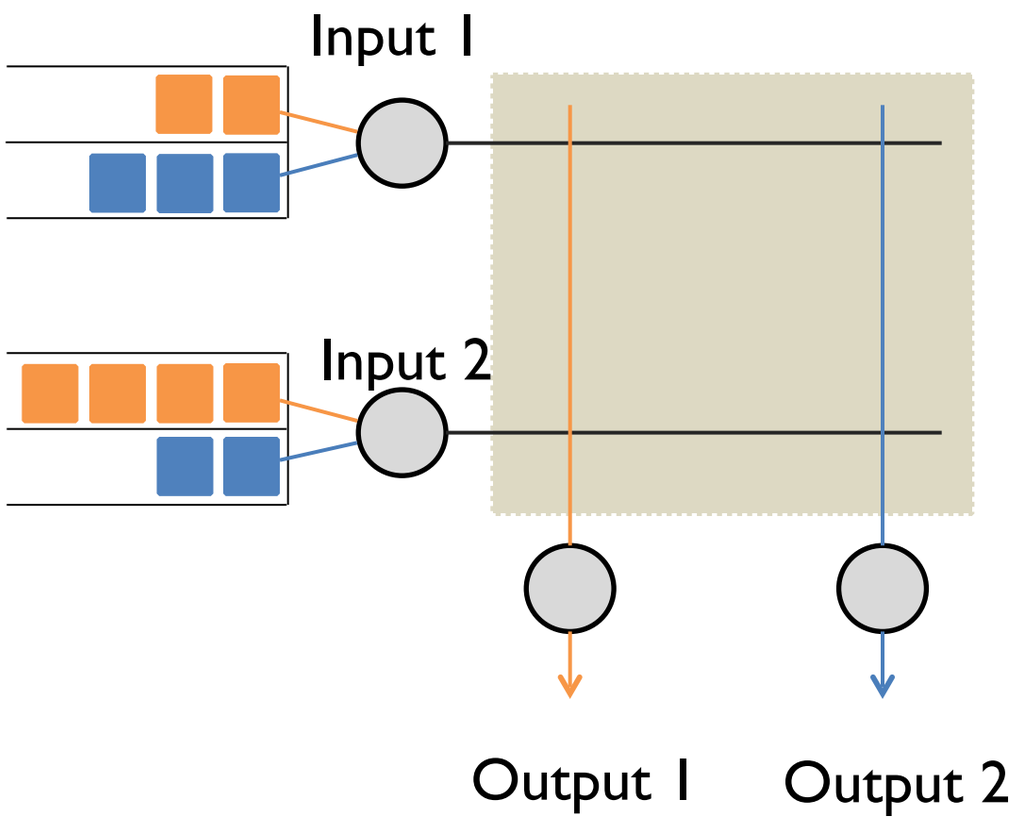
Identify multi-class "product-form" (ala Kelly) network

Design policy for "relaxed" network which makes it equivalent to this multi-class "product-form" network

Design emulation policy for switched network that emulates behavior of any policy in "relaxed" network

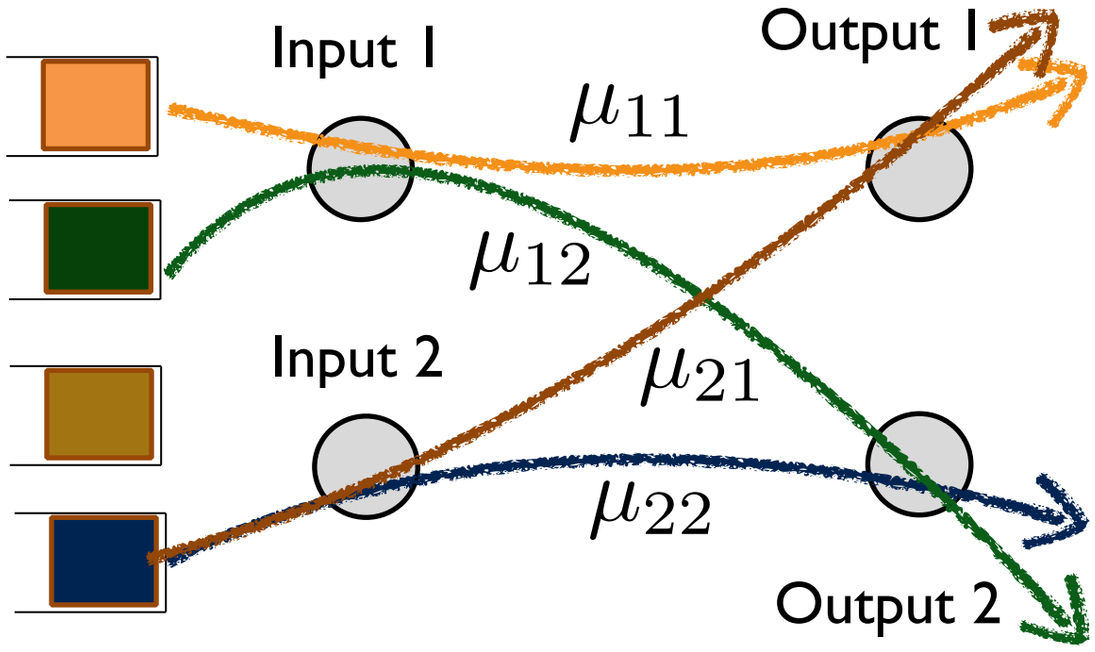
(a good analogy = rounding or relaxation in combinatorial optimization)

Relaxed Switched Networks



Switched network

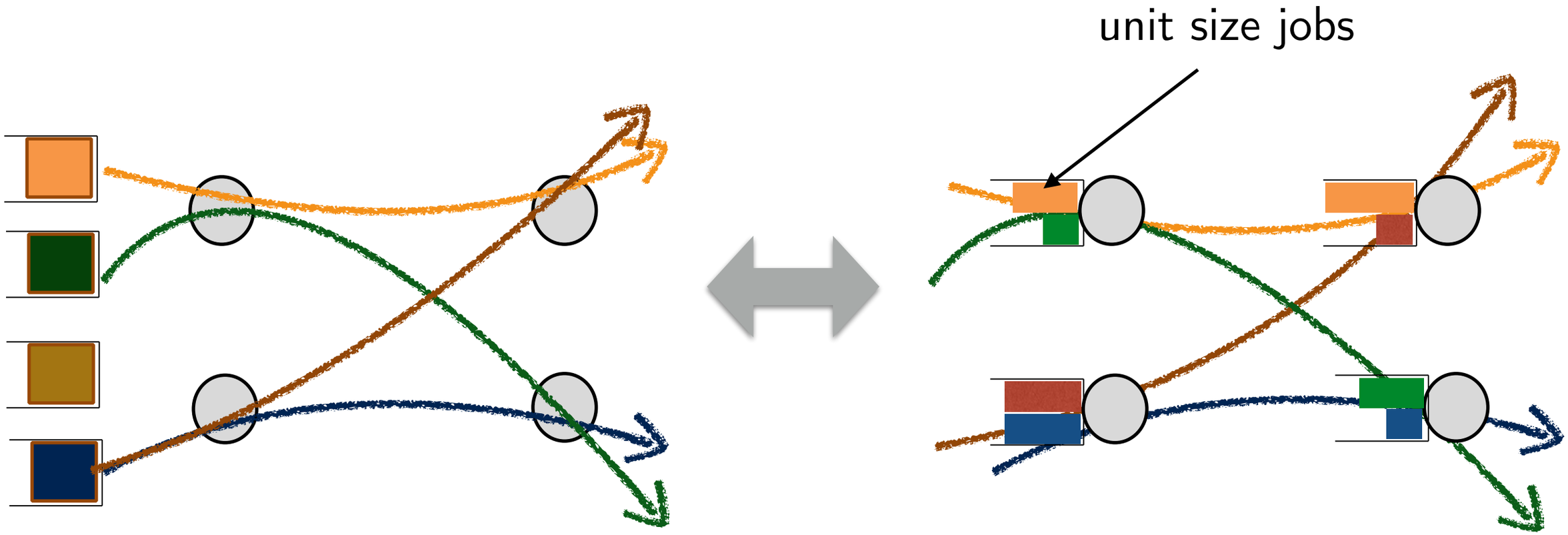
$$\mu_{11} + \mu_{12} \leq 1 \quad \mu_{11} + \mu_{21} \leq 1$$



$$\mu_{21} + \mu_{22} \leq 1 \quad \mu_{12} + \mu_{22} \leq 1$$

Flow-level network
(bandwidth sharing)

Multi-class "Product-form" Network



Flow-level network
(bandwidth sharing)

Multi-class Multi-hop Network
(processor sharing nodes)

Policy for Relaxed Switched Network

Store and Forward Policy by Massoulie, Bonald-Proutiere (2003)

Each flow is allocated rate

as a function of total number of flows queued

the function corresponds to ratio of two normalization constants

with and without one less flow of the specific type

the normalization constant arises from stationary distribution of

associated Multi-class "product-form" network

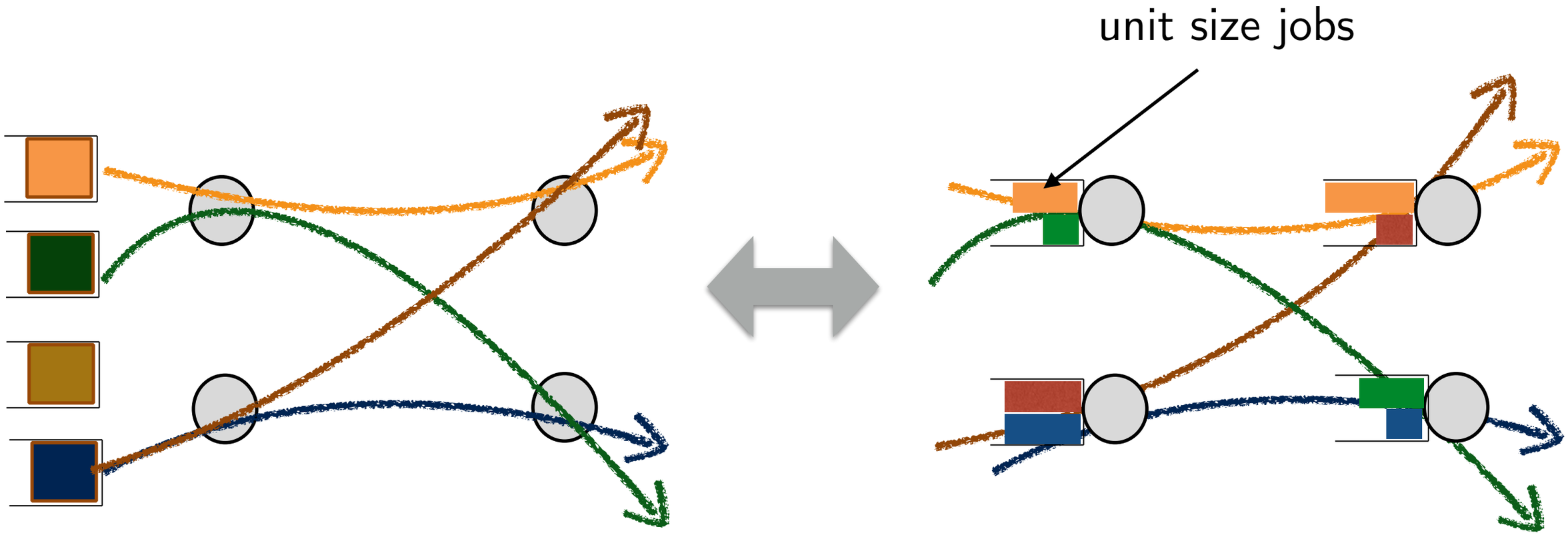
Such choice makes sure that under stationary distribution

total jobs in flow network = total jobs in multi-class network

Insensitivity established by Zachary (2008)

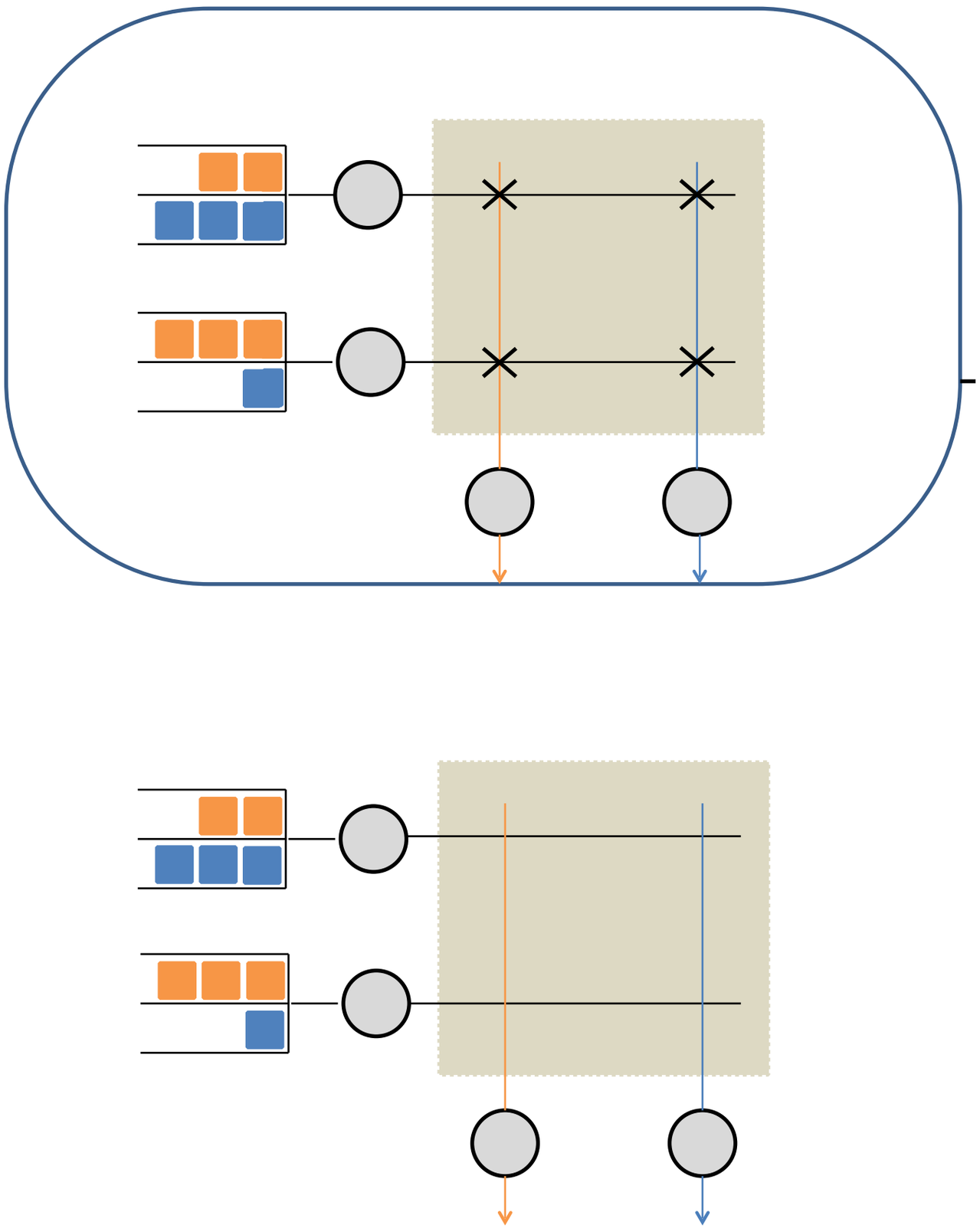
Policy for Relaxed Switched Network

Store and Forward Policy by Massoulie, Bonald-Proutiere (2003)

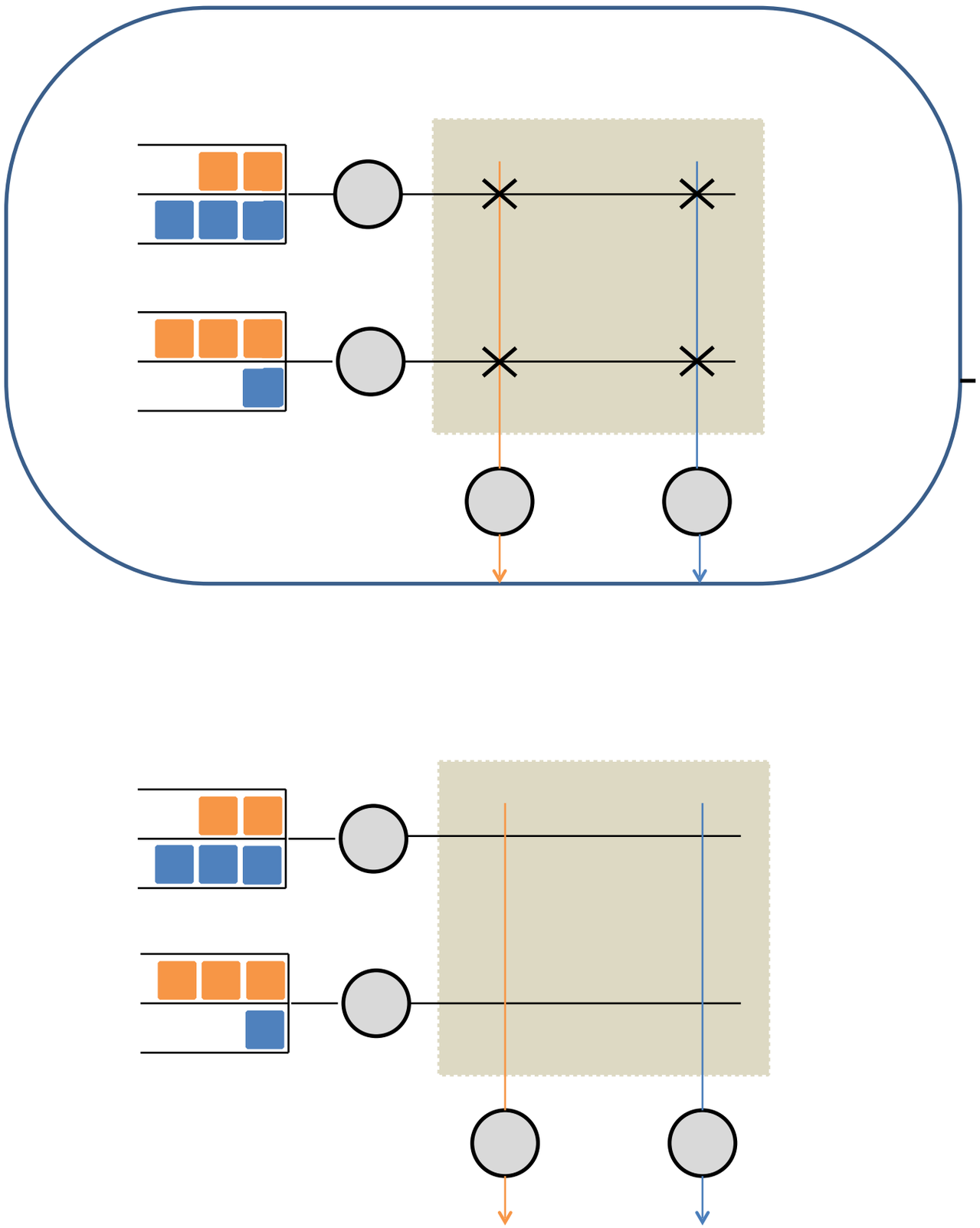


$$\text{Average number of jobs in the system in steady-state} \sim \frac{\text{number of nodes in multi-class net.}}{1-\rho}$$

Emulating Relaxed Switched Networks



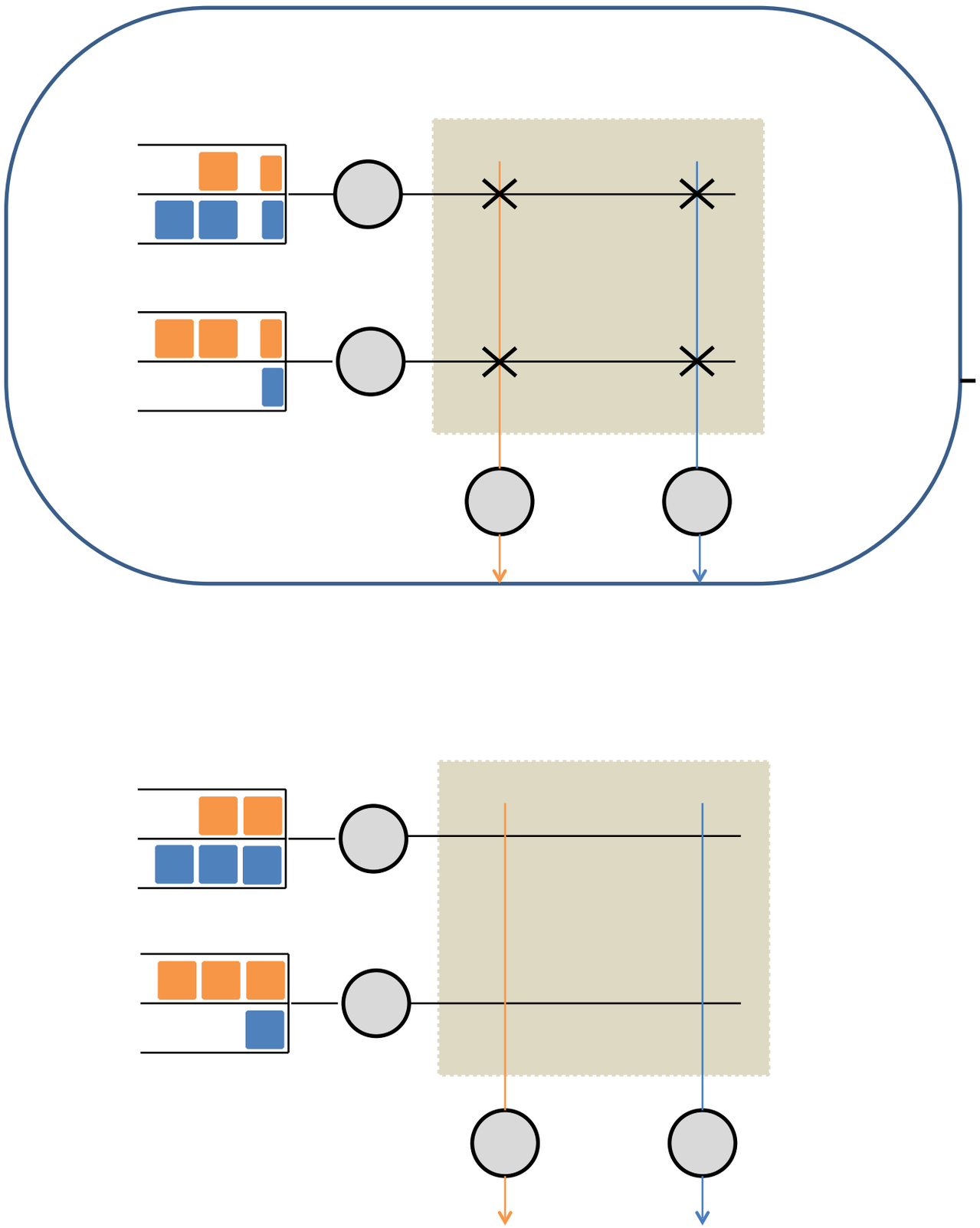
Emulating Relaxed Switched Networks



Time 1

$$\begin{bmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{bmatrix}$$

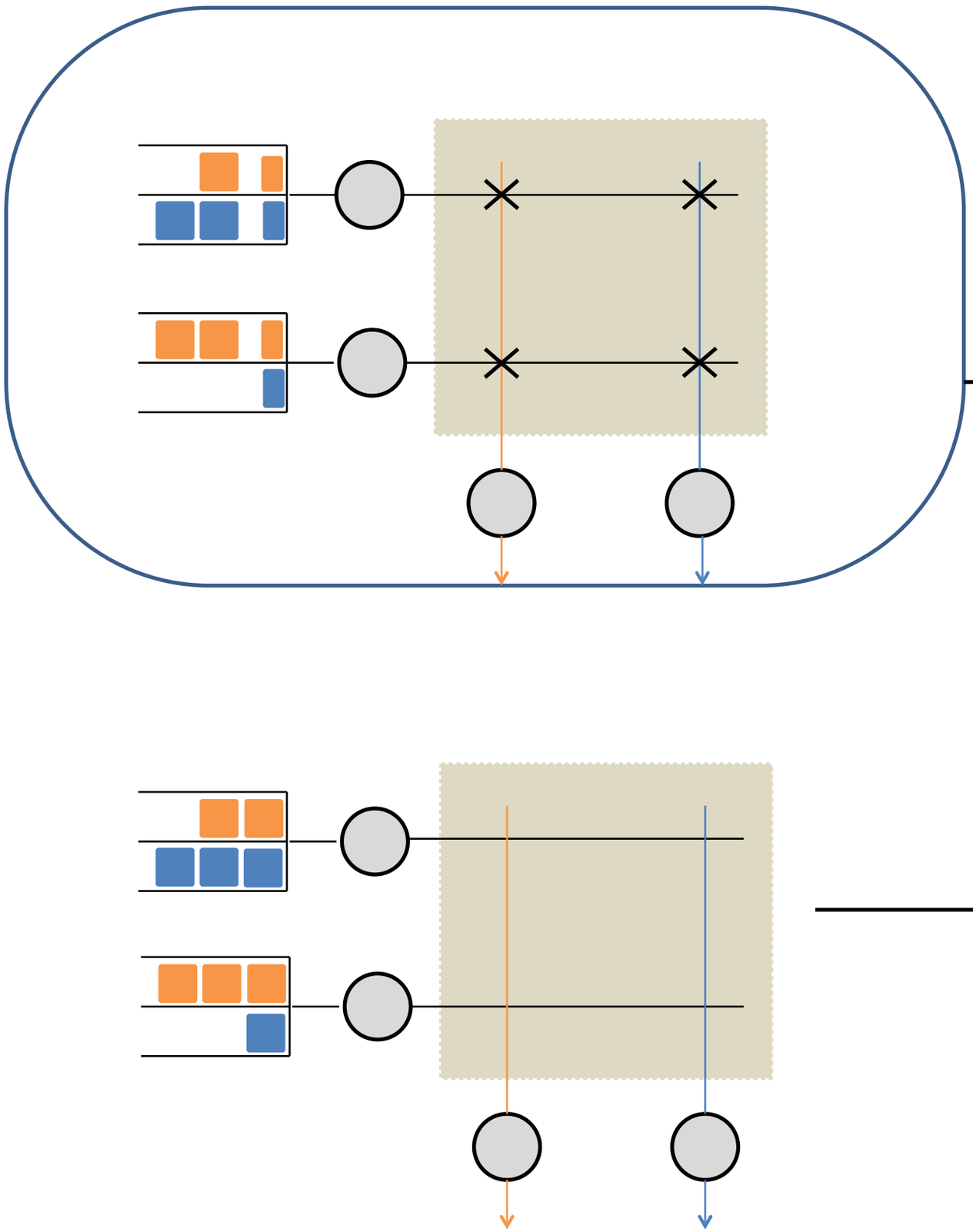
Emulating Relaxed Switched Networks



Time 1

$$\begin{bmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{bmatrix}$$

Emulating Relaxed Switched Networks

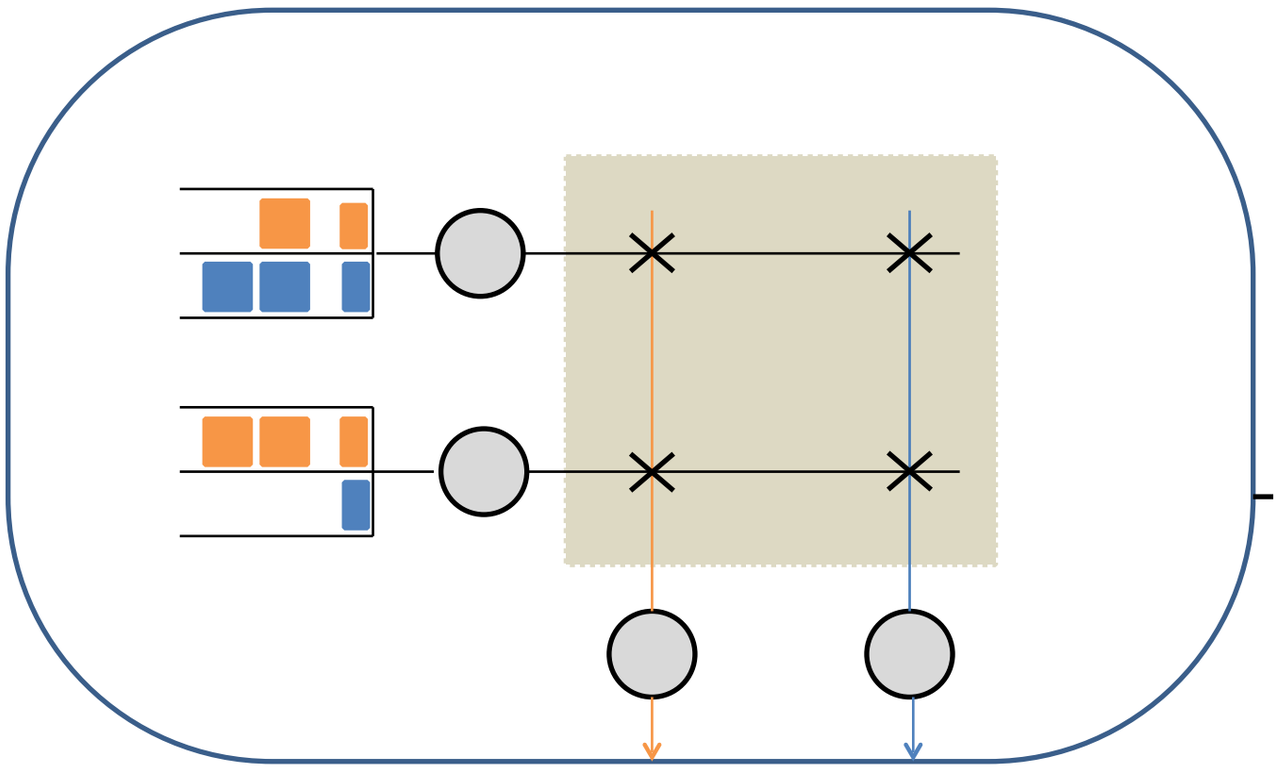


Time 1

$$\begin{bmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

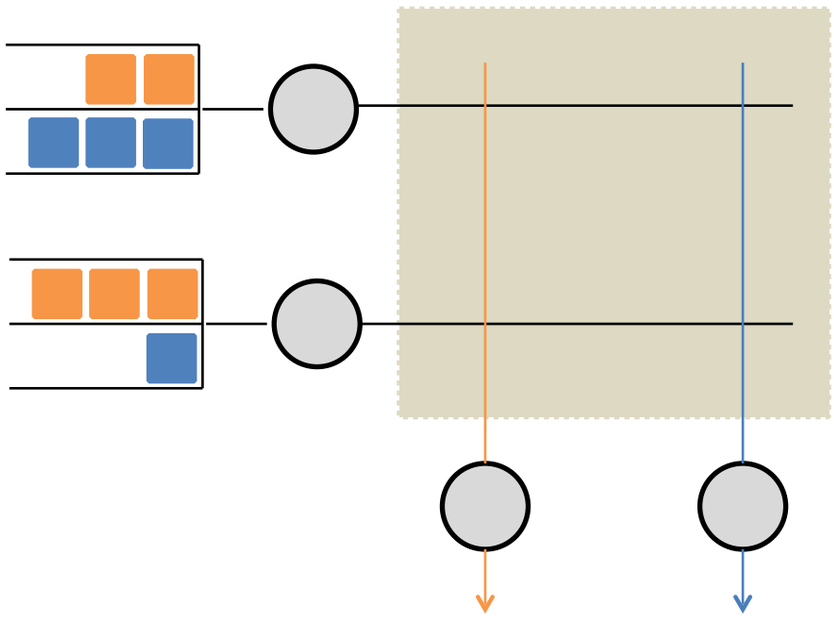
Emulating Relaxed Switched Networks



Time 1

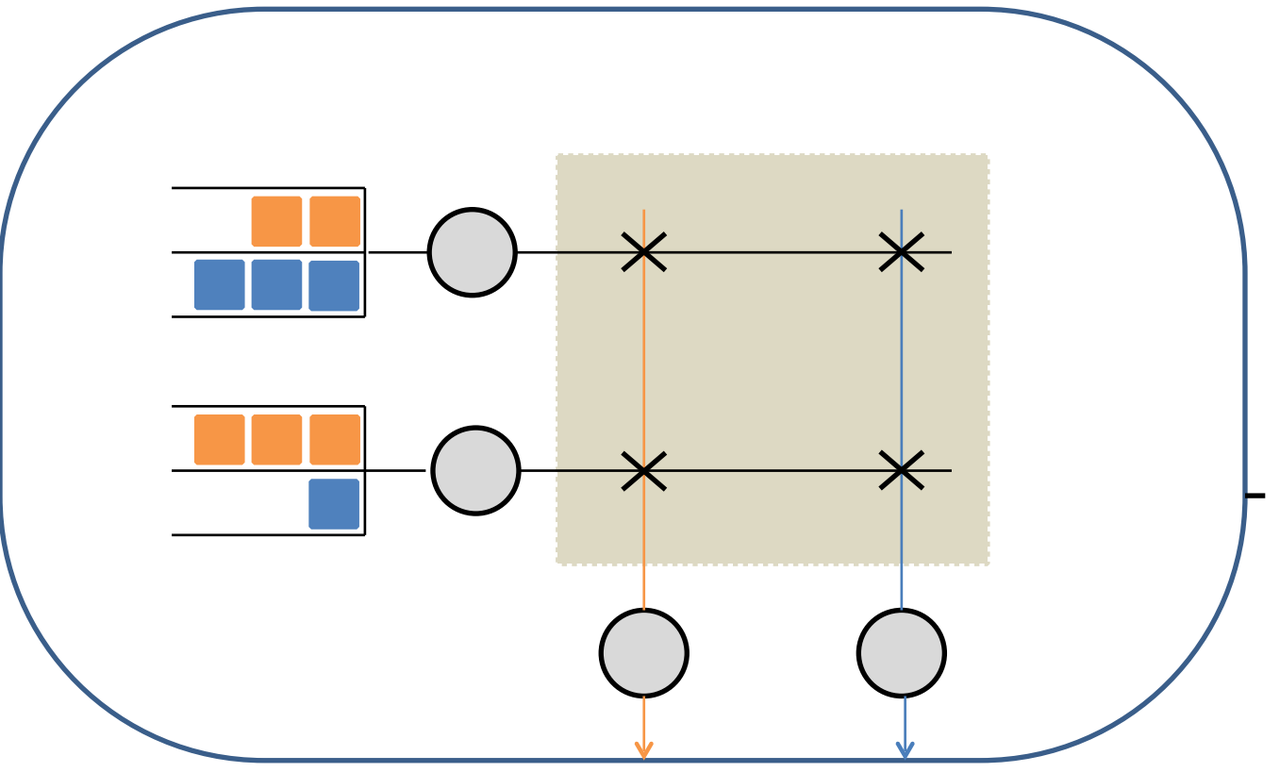
$$\begin{bmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{bmatrix}$$

$$\begin{bmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{bmatrix} \longleftarrow \text{Lag}$$



$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

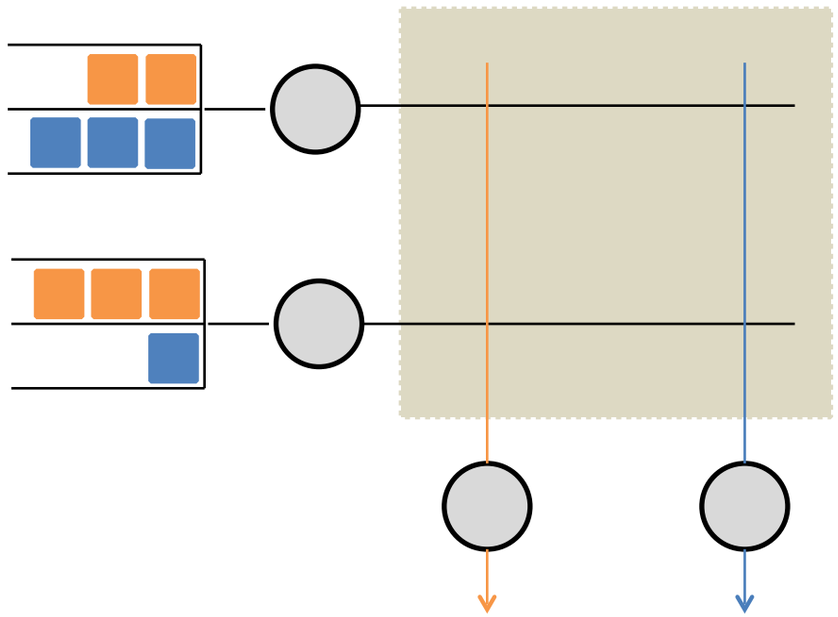
Emulating Relaxed Switched Networks



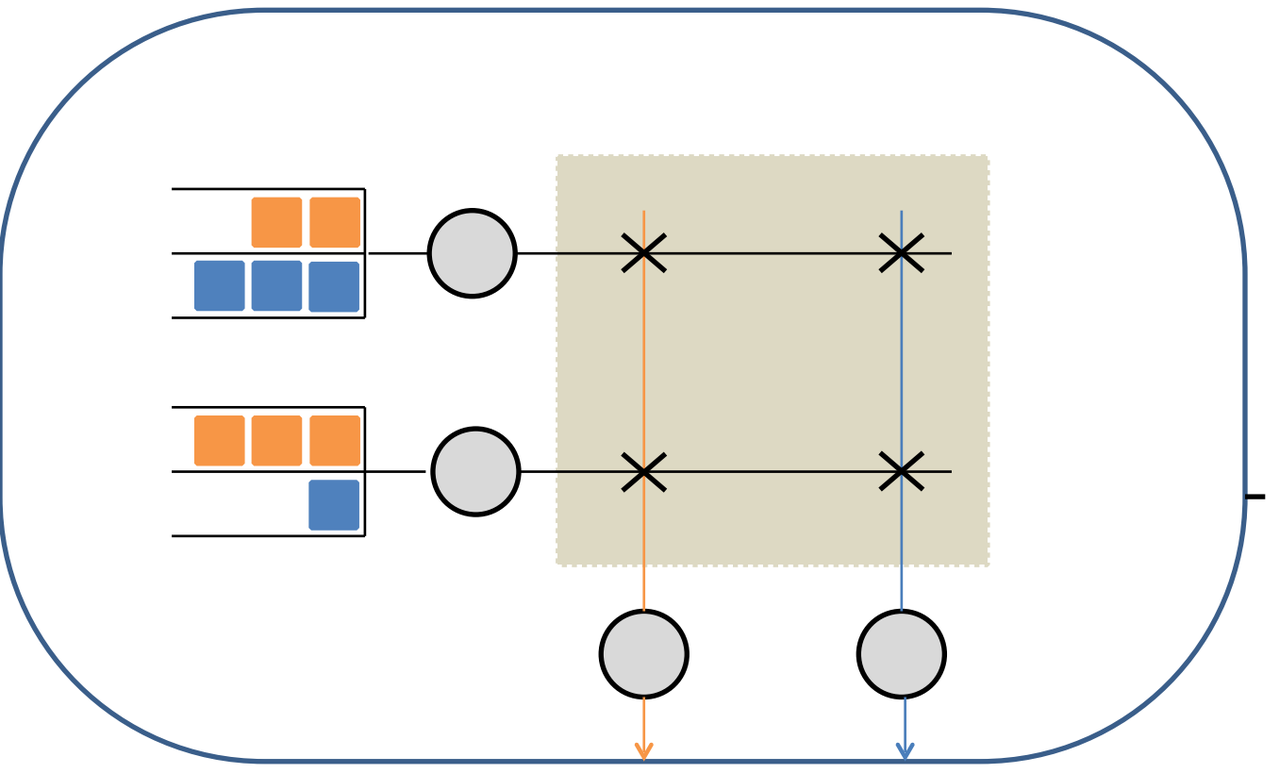
Time 2

$$\begin{bmatrix} 1/4 & 3/4 \\ 3/4 & 1/4 \end{bmatrix}$$

$$\begin{bmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{bmatrix}$$



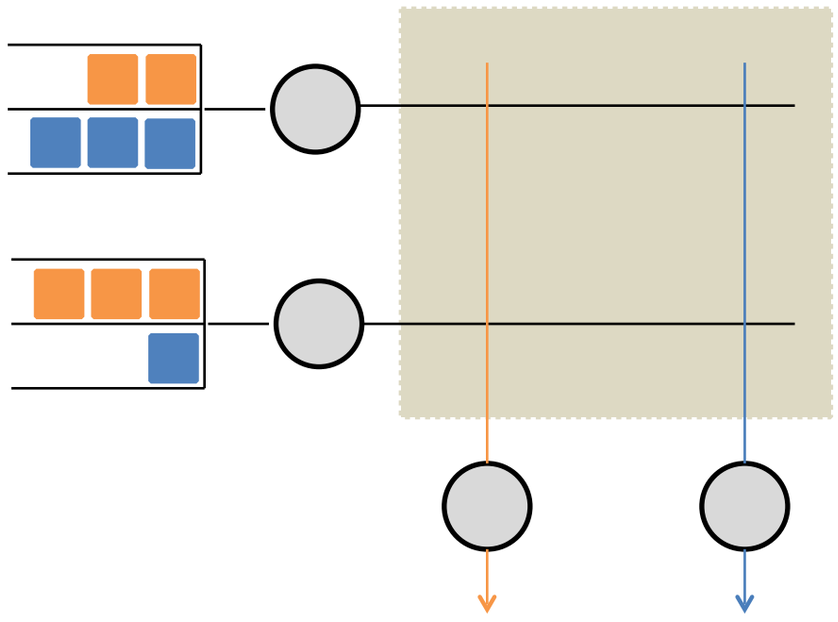
Emulating Relaxed Switched Networks



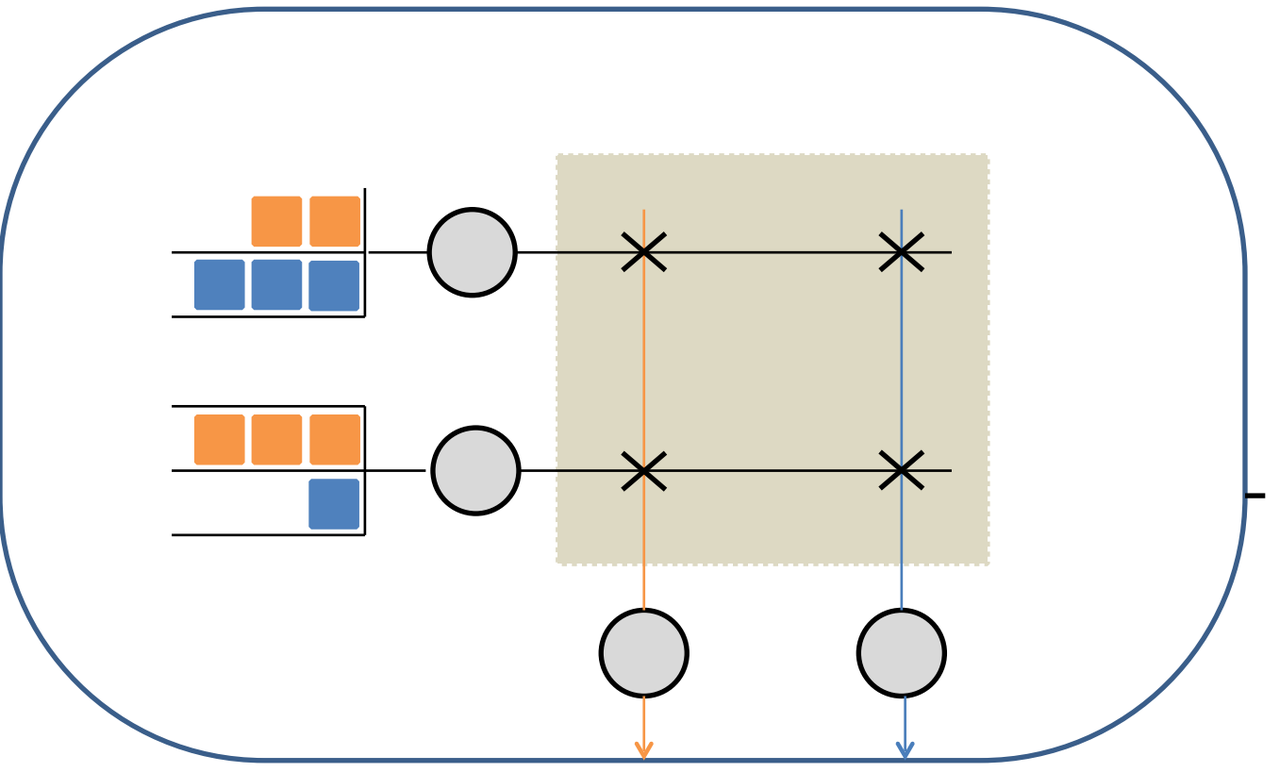
Time 2

$$\begin{bmatrix} 1/4 & 3/4 \\ 3/4 & 1/4 \end{bmatrix}$$

$$\begin{bmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{bmatrix} + \begin{bmatrix} 1/4 & 3/4 \\ 3/4 & 1/4 \end{bmatrix}$$

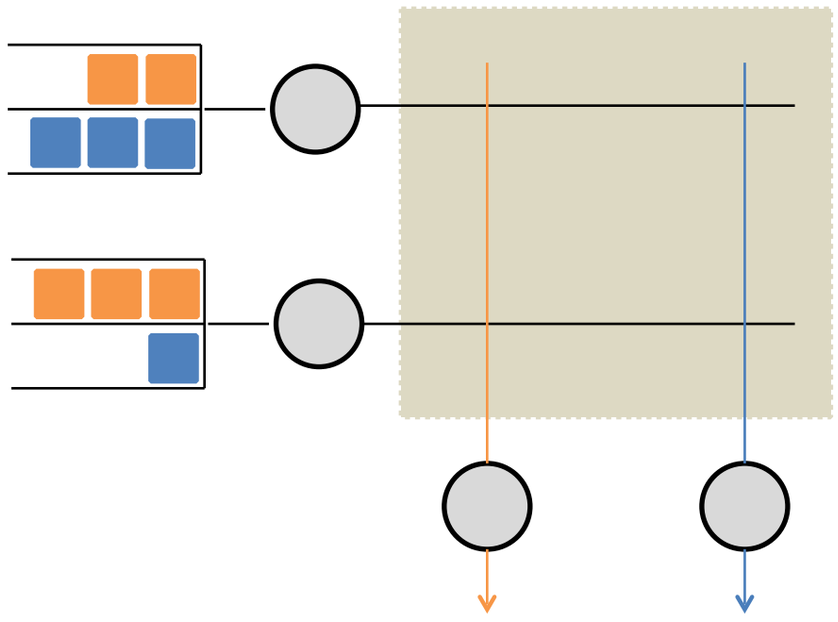


Emulating Relaxed Switched Networks

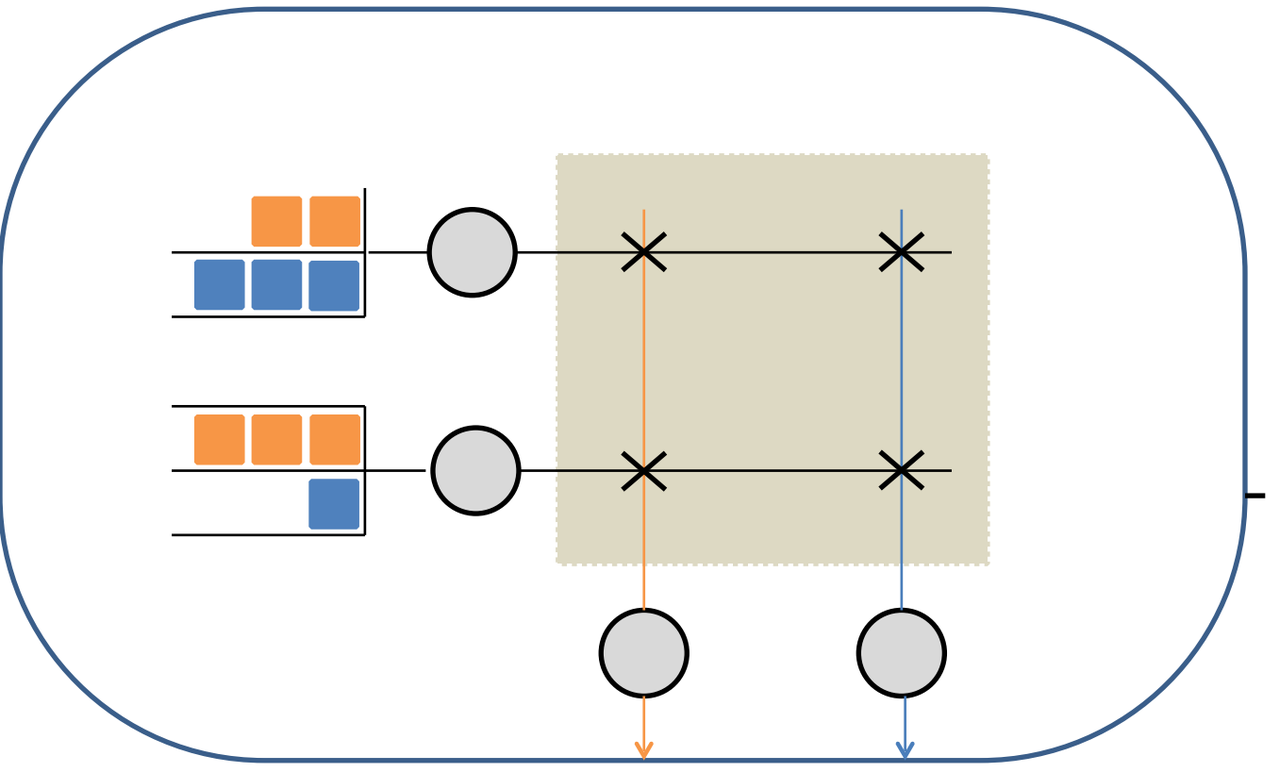


$$\begin{bmatrix} 1/4 & 3/4 \\ 3/4 & 1/4 \end{bmatrix}$$

$$\begin{bmatrix} 3/4 & 5/4 > 1 \\ 5/4 > 1 & 3/4 \end{bmatrix}$$

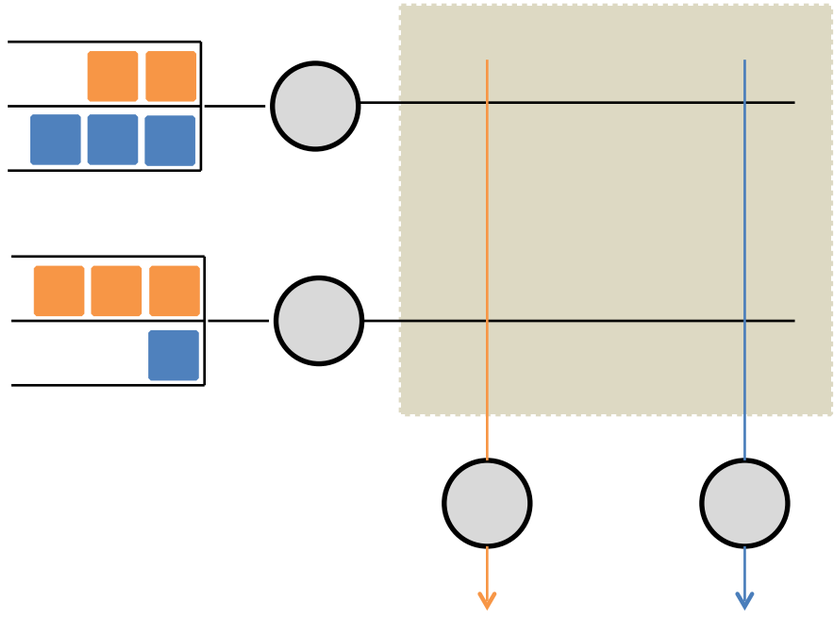


Emulating Relaxed Switched Networks

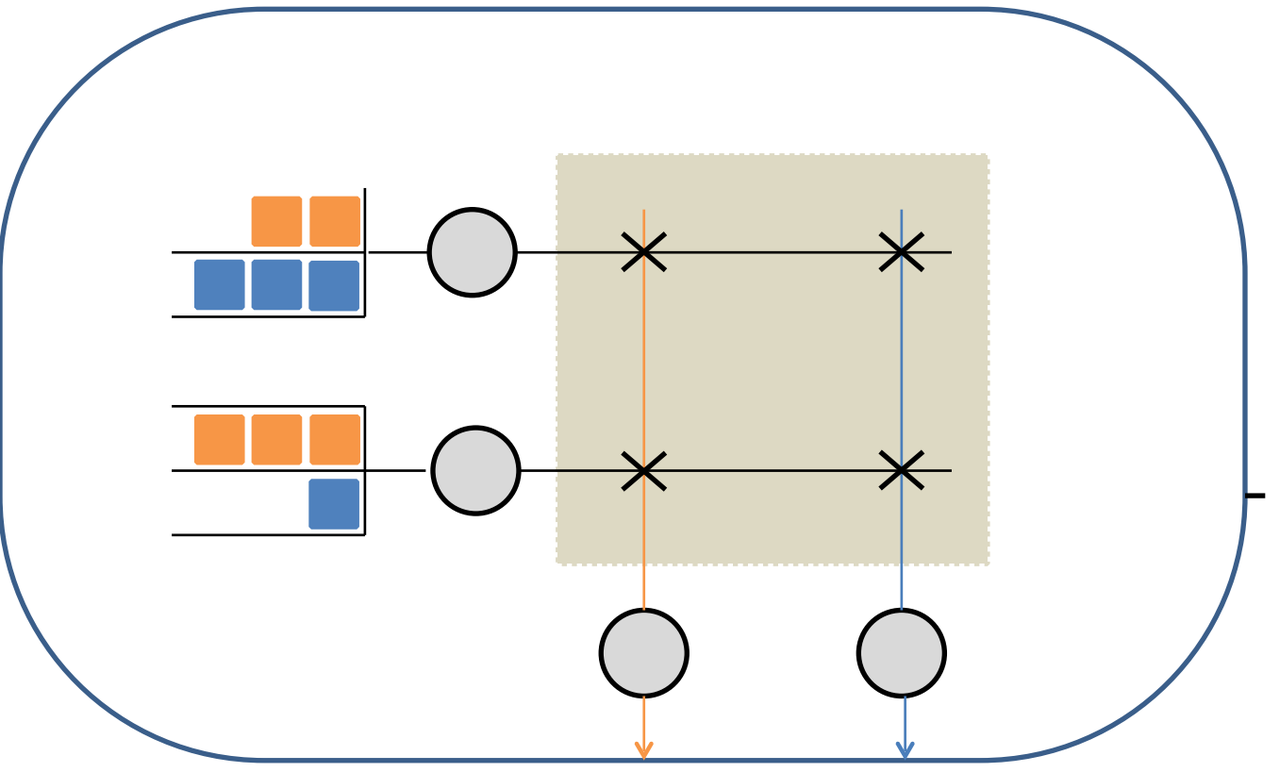


$$\begin{bmatrix} 1/4 & 3/4 \\ 3/4 & 1/4 \end{bmatrix}$$

$$\begin{bmatrix} 3/4 & 5/4 > 1 \\ 5/4 > 1 & 3/4 \end{bmatrix}$$

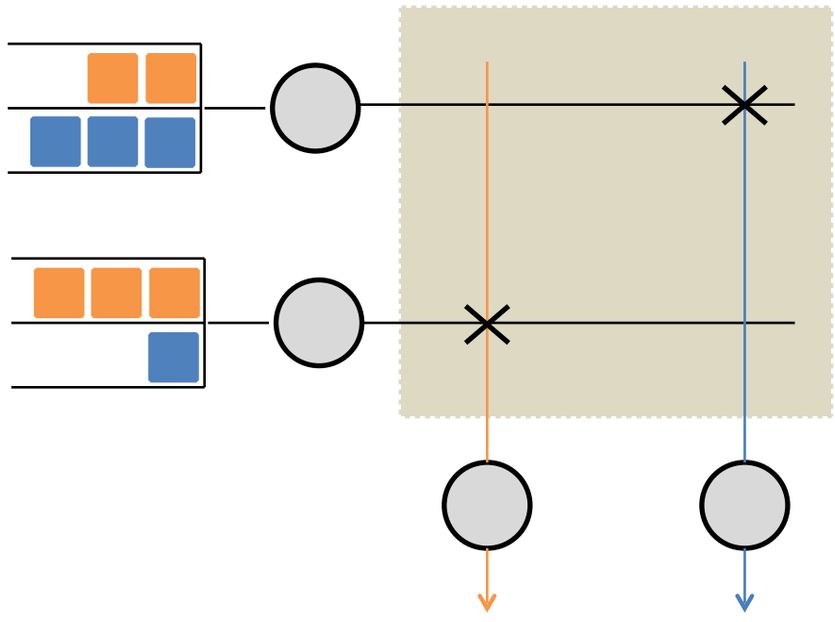


Emulating Relaxed Switched Networks



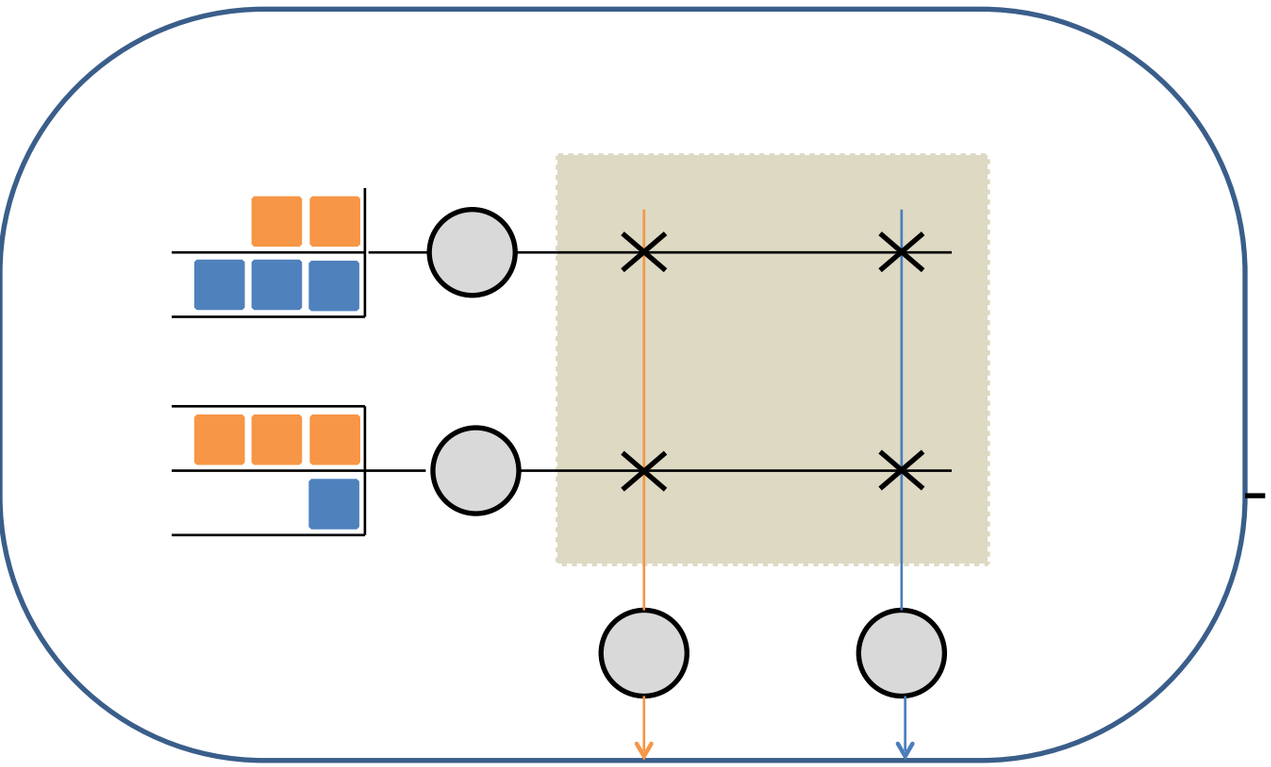
$$\begin{bmatrix} 1/4 & 3/4 \\ 3/4 & 1/4 \end{bmatrix}$$

$$\begin{bmatrix} 3/4 & 5/4 > 1 \\ 5/4 > 1 & 3/4 \end{bmatrix}$$



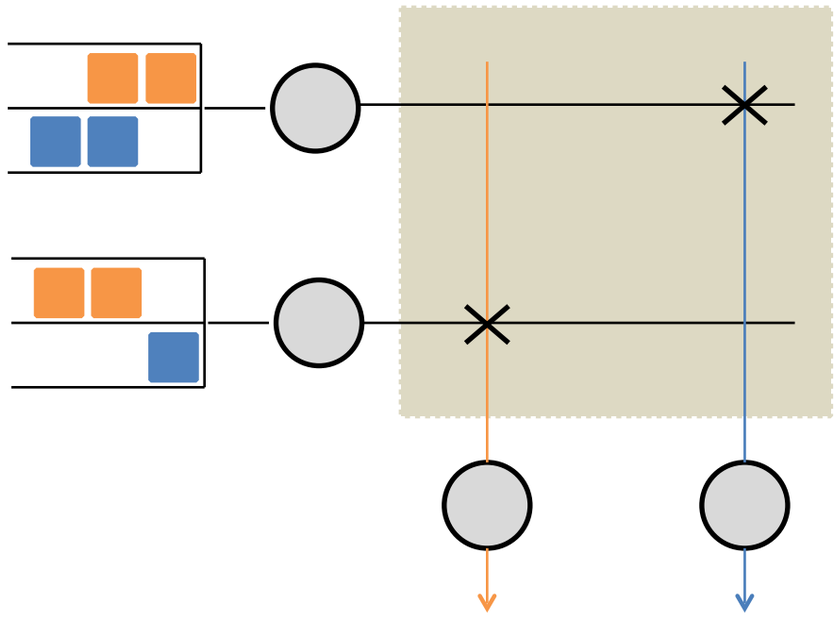
$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Emulating Relaxed Switched Networks



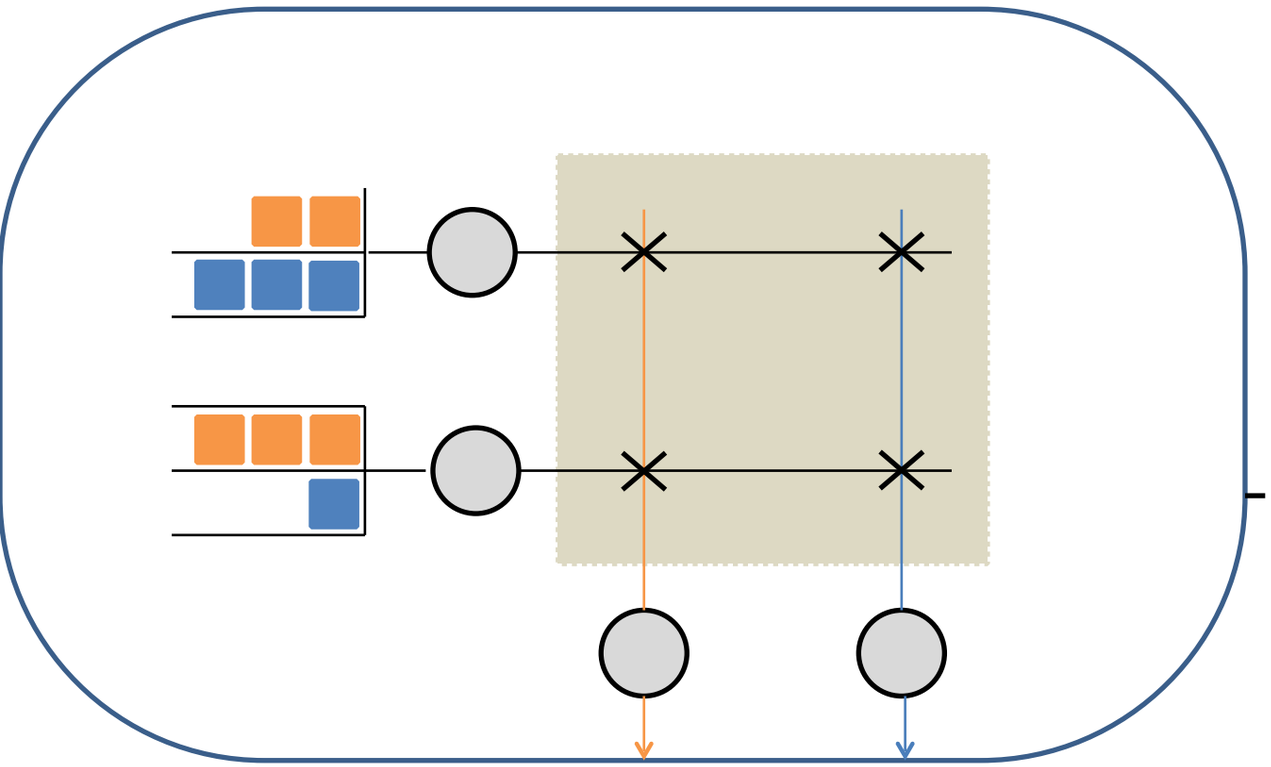
$$\begin{bmatrix} 1/4 & 3/4 \\ 3/4 & 1/4 \end{bmatrix}$$

$$\begin{bmatrix} 3/4 & 5/4 > 1 \\ 5/4 > 1 & 3/4 \end{bmatrix}$$



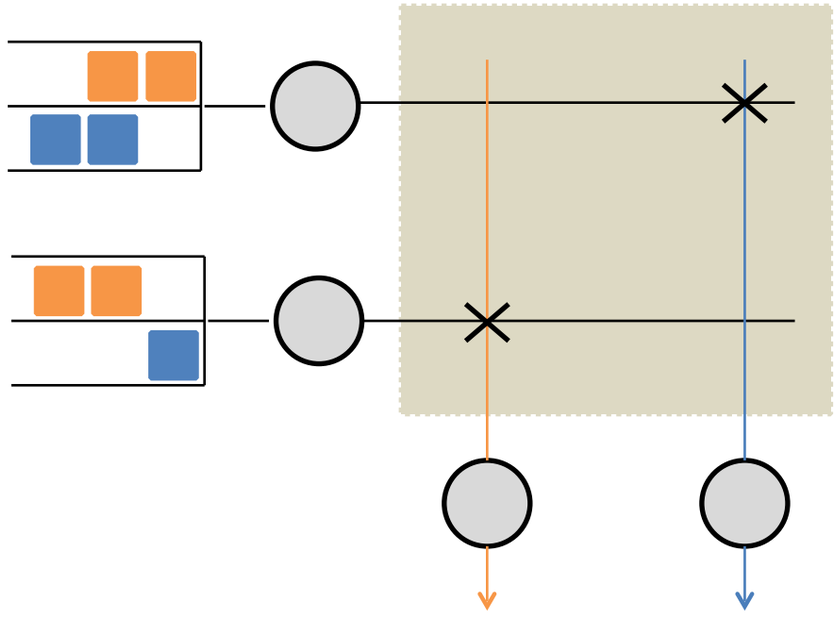
$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Emulating Relaxed Switched Networks



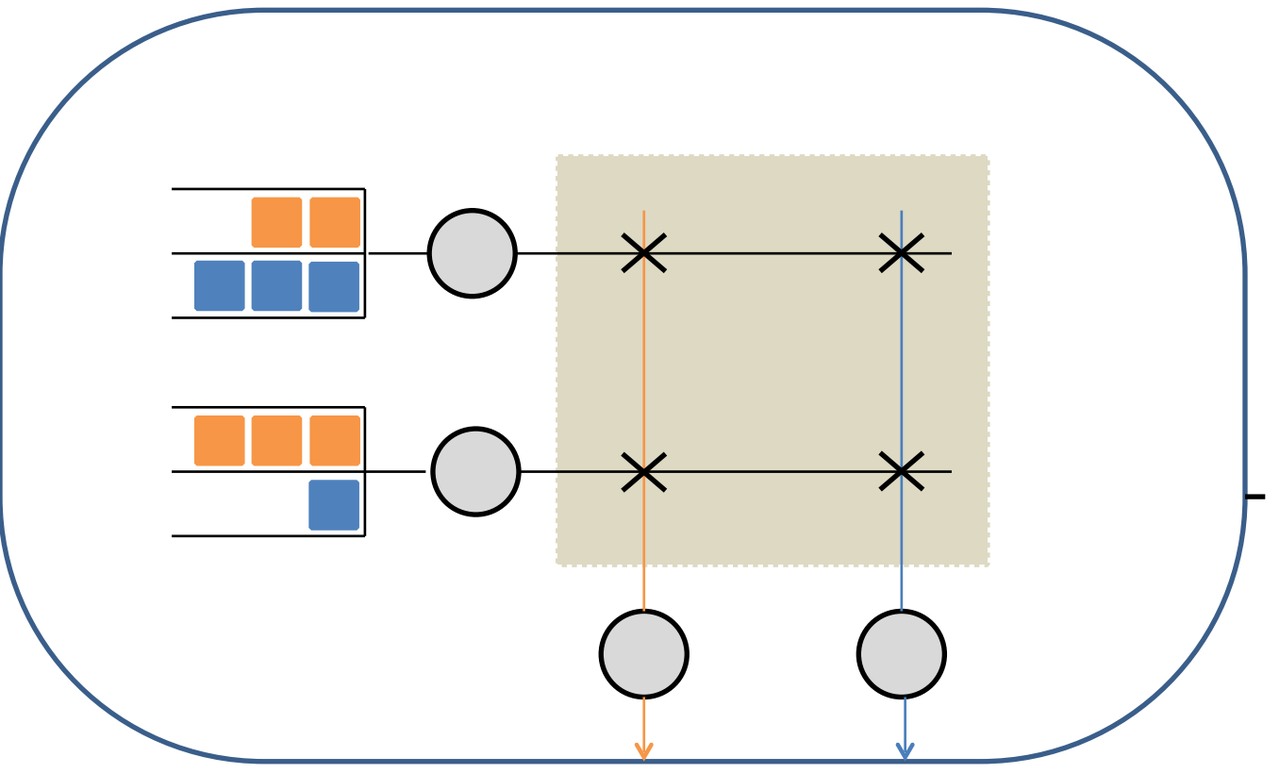
$$\begin{bmatrix} 1/4 & 3/4 \\ 3/4 & 1/4 \end{bmatrix}$$

$$\begin{bmatrix} 3/4 & 5/4 > 1 \\ 5/4 > 1 & 3/4 \end{bmatrix} - \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$



$$\rightarrow \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

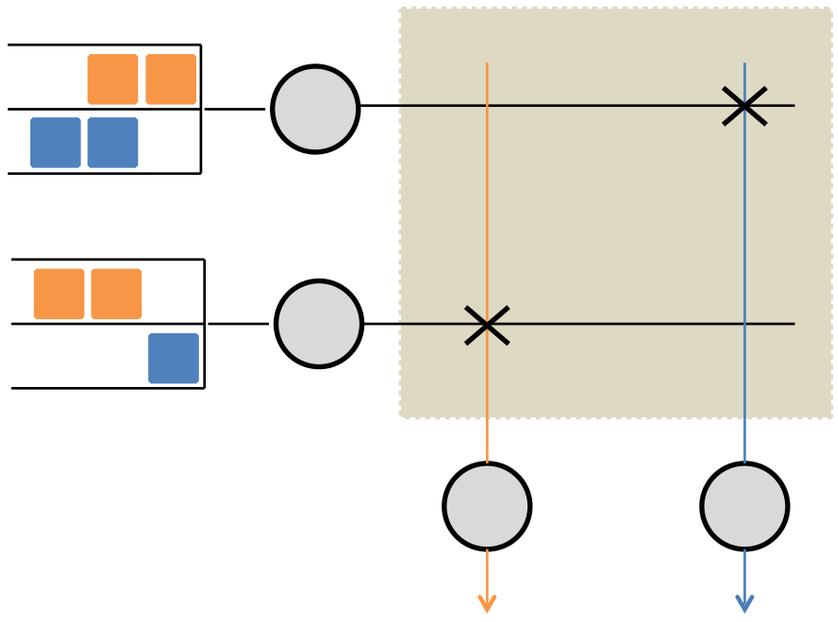
Emulating Relaxed Switched Networks



Time 2

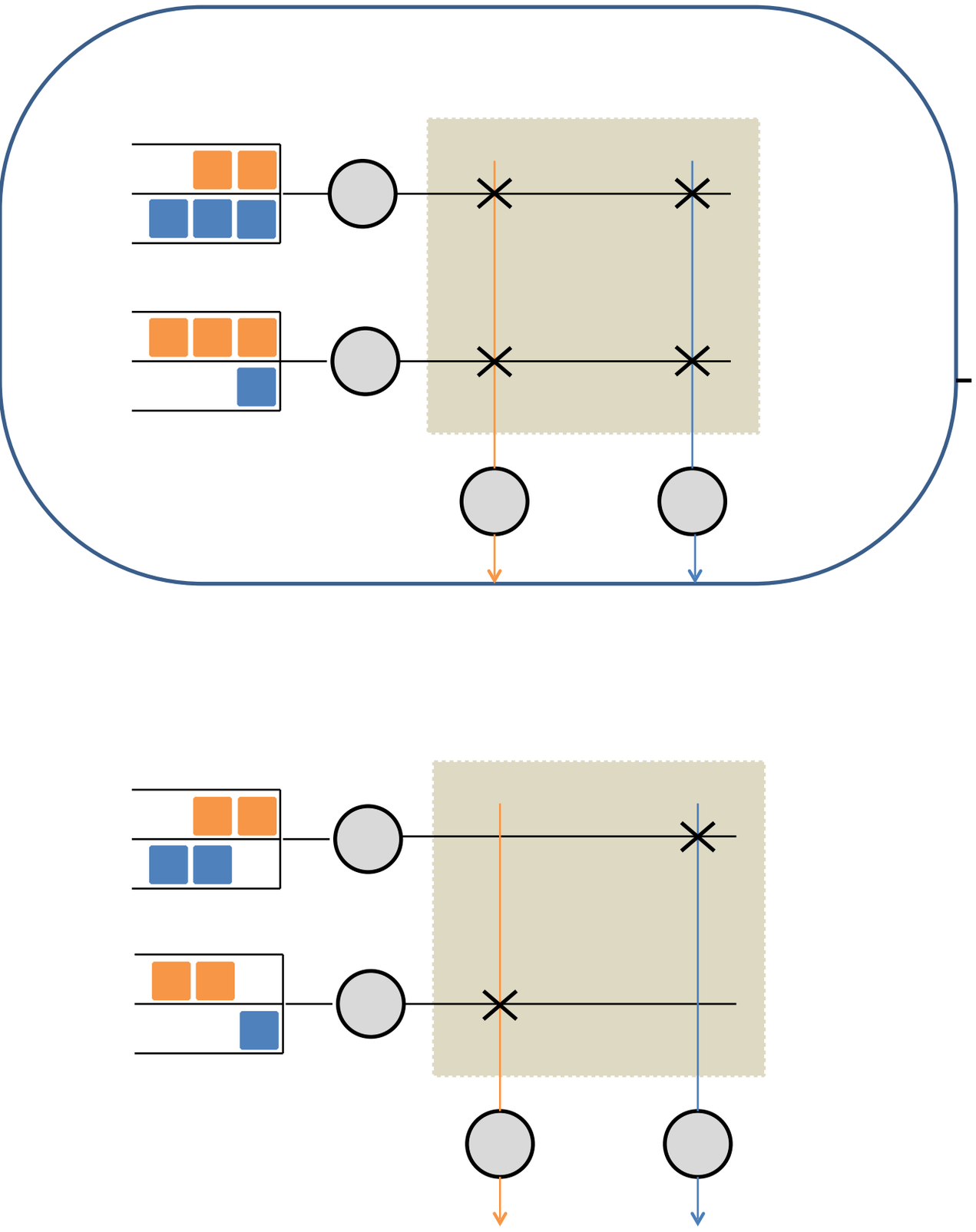
$$\begin{bmatrix} 1/4 & 3/4 \\ 3/4 & 1/4 \end{bmatrix}$$

$$\begin{bmatrix} 3/4 & 1/4 \\ 1/4 & 3/4 \end{bmatrix} \longleftarrow \text{Lag}$$



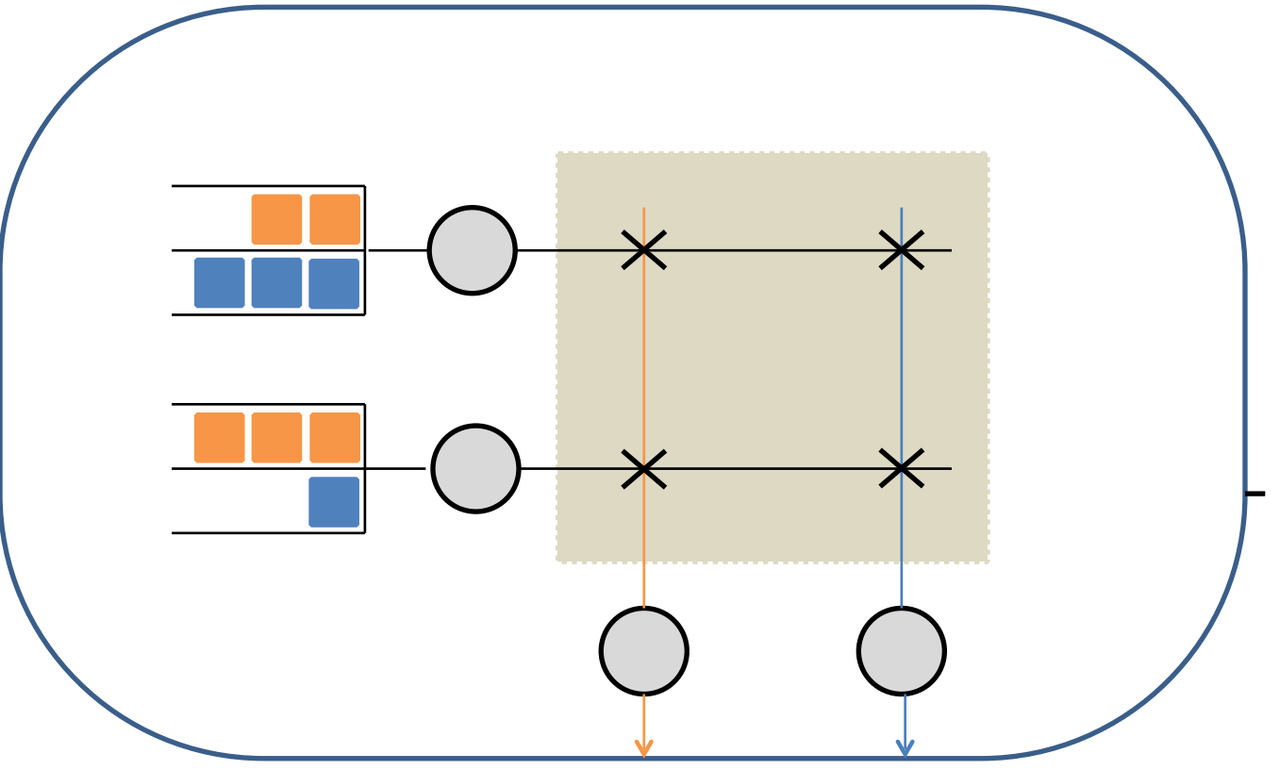
$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Emulating Relaxed Switched Networks



Time 3, etc

Emulating Relaxed Switched Networks

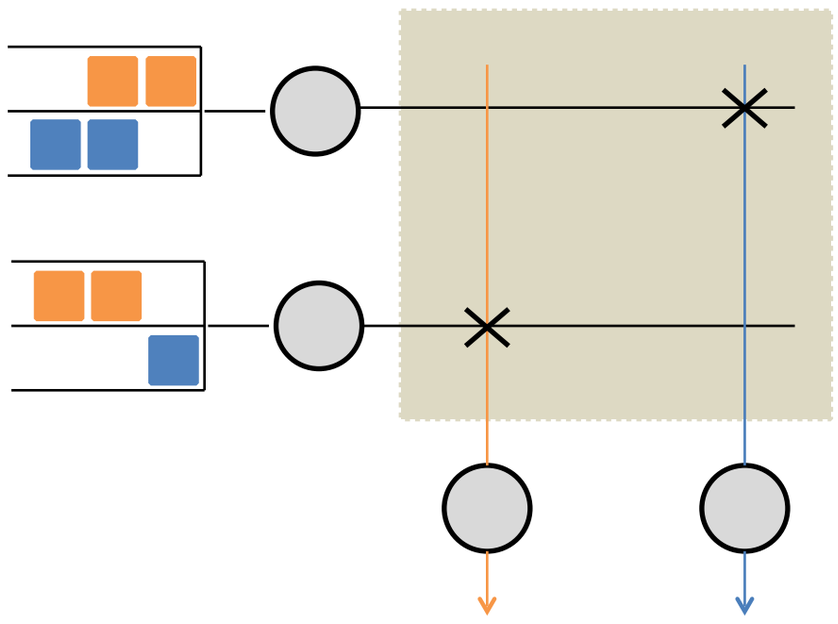


Time 3, etc

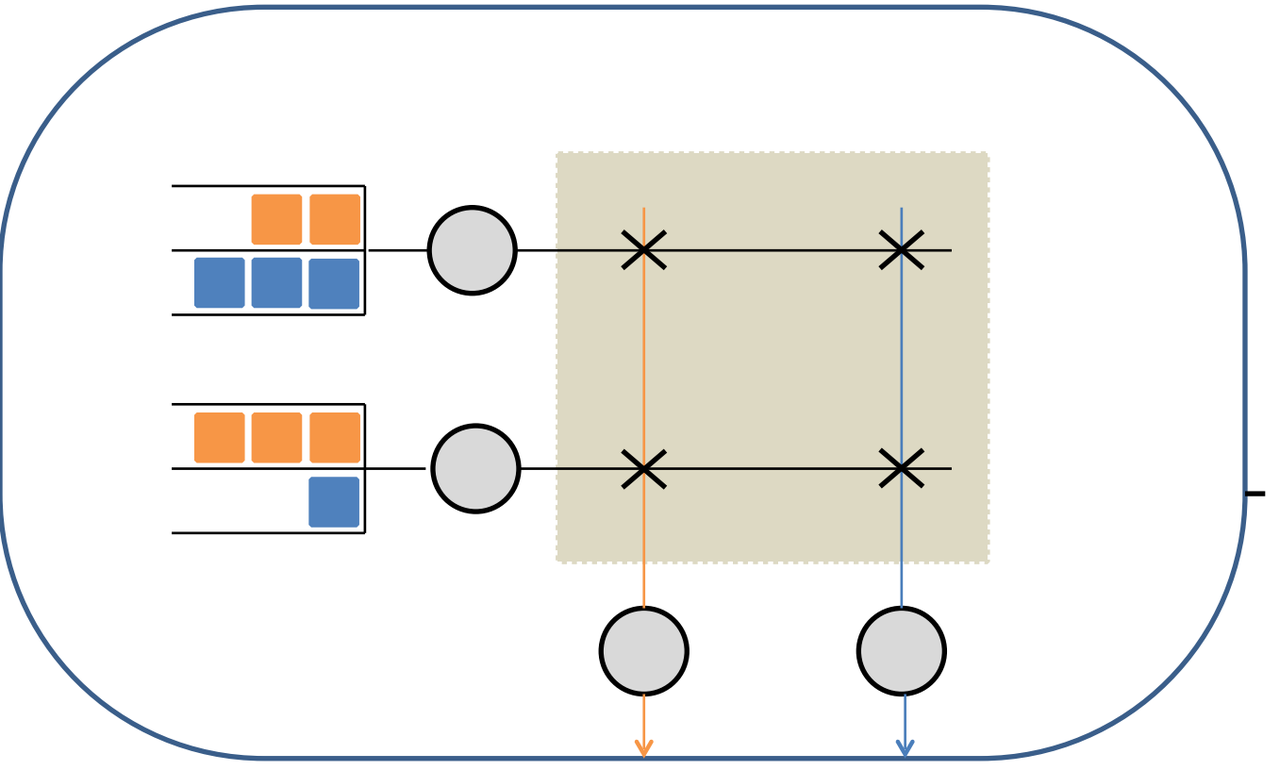
Lag

$$\begin{bmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \frac{1}{2}\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$\Rightarrow$ No matching



Emulating Relaxed Switched Networks

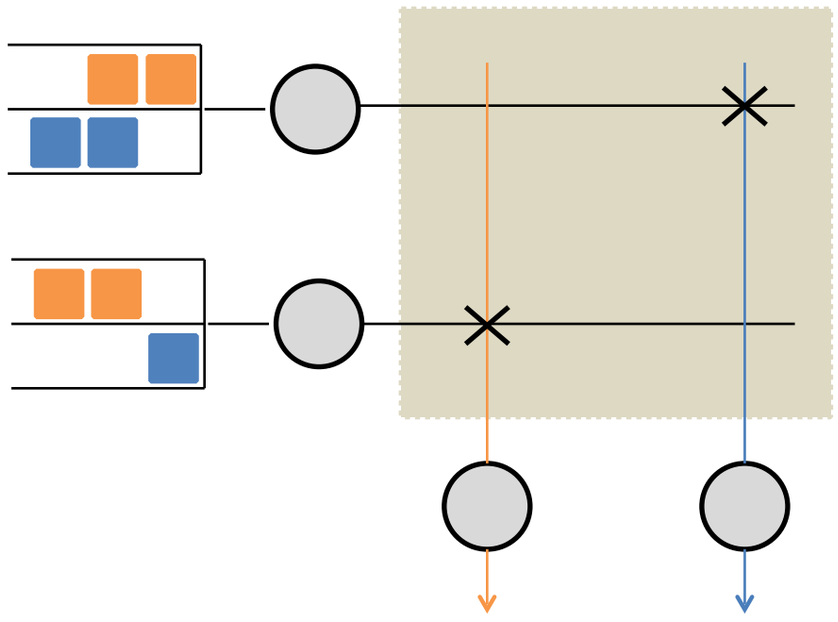


Time 3, etc

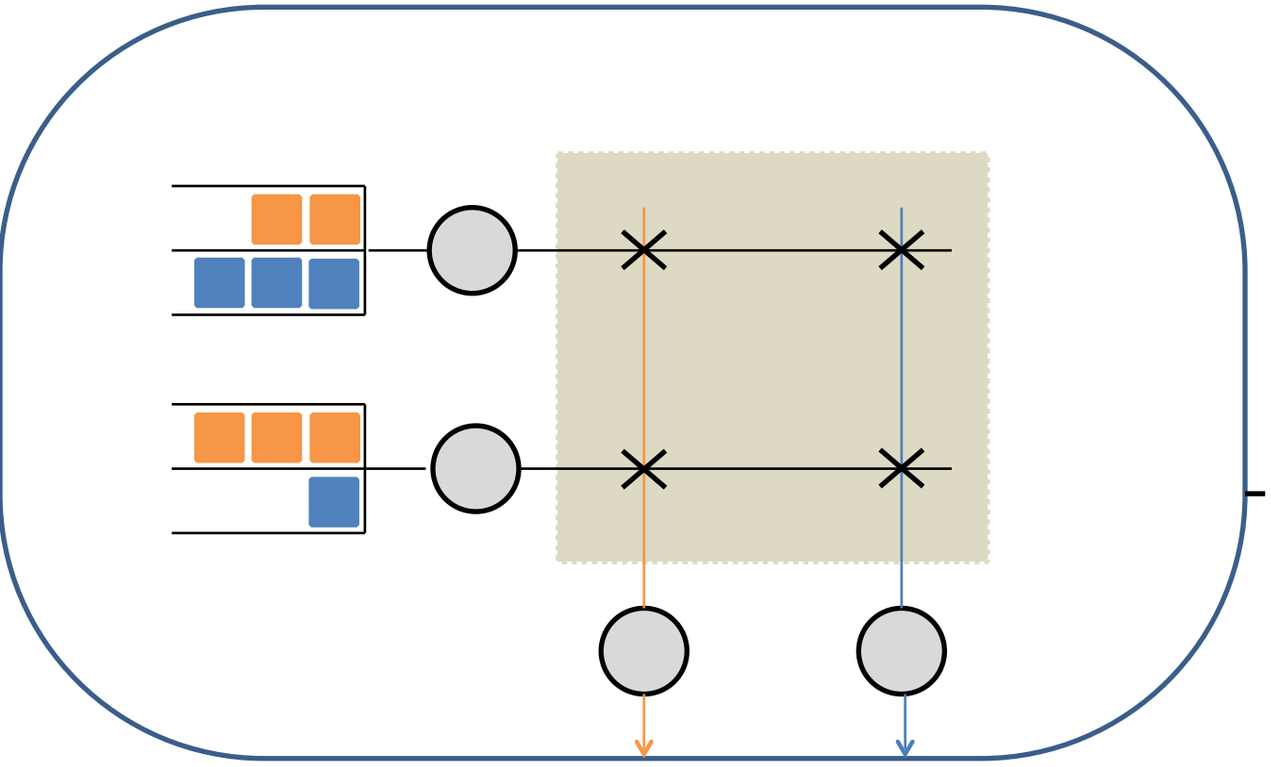
Lag

$$\begin{bmatrix} 3/4 & 5/4 \\ 5/4 & 3/4 \end{bmatrix} = \frac{3}{4}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \frac{5}{4}\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$\Rightarrow \text{pick}\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$



Emulating Relaxed Switched Networks

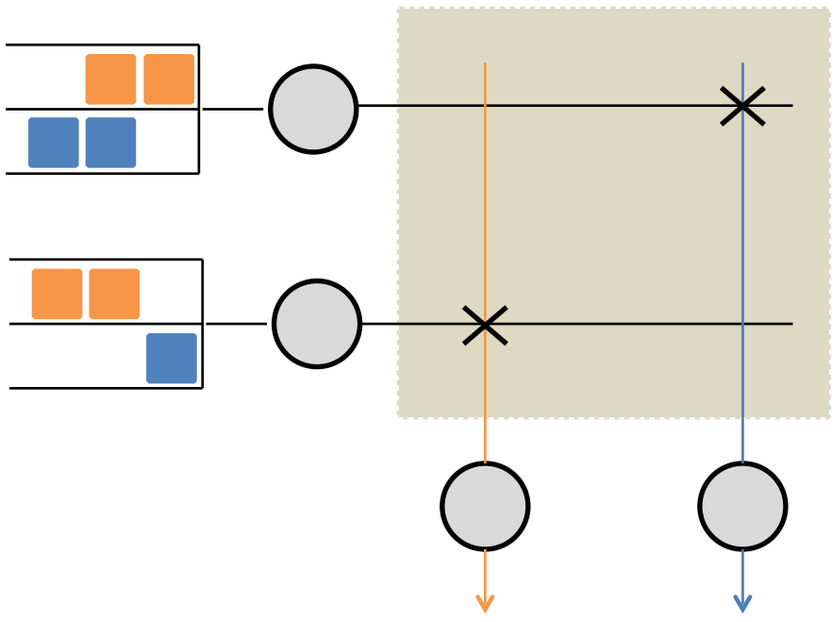


Time 3, etc

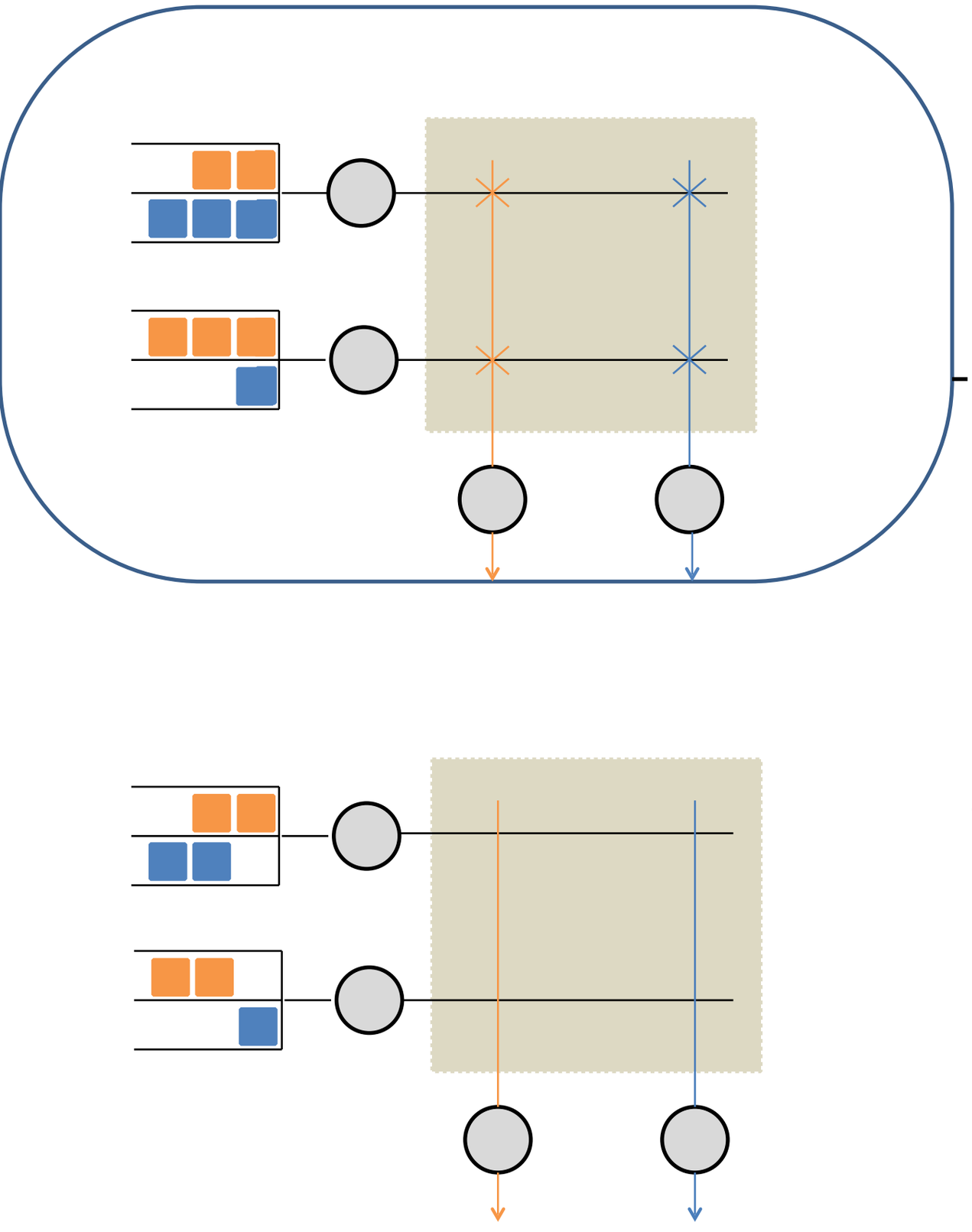
Lag

$$\begin{bmatrix} 7/4 & 5/4 \\ 5/4 & 7/4 \end{bmatrix} = \frac{7}{4}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \frac{5}{4}\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$\Rightarrow \text{pick}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



Emulating Relaxed Switched Networks



Time 3, etc

Lemma. Total difference in q-size is at most n^3.

Emulating Relaxed Switched Networks

In summary:

$$\begin{aligned}
\text{total avg q-size of switched net.} &\leq \text{total avg q-size of switched net.} + C(\mathbb{S}) \\
&= \frac{M}{1-\rho} + C(\mathbb{S})
\end{aligned}$$

Recall: Capacity region, the convex hull of feasible schedules $\mathbb{S}$

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

Emulating Relaxed Switched Networks

In summary:

$$\begin{array}{ccc} \text{total avg q-size of switched net.} & \leq & \text{total avg q-size of switched net.} \quad + \quad C(\mathbb{S}) \\ & = & \dfrac{M}{1-\rho} \quad + \quad C(\mathbb{S}) \end{array}$$

num of faces of capacity region

Recall: Capacity region, the convex hull of feasible schedules $\mathbb{S}$

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

That is, *baseline* performance is achieved cf. Harrison-Mandyam-Shah-Yang (2014)

What Happened to The Open Question

Consider input-queued switch with N inputs / outputs

there exists a policy such that

achieves maximal capacity

computationally efficient (poly-time)

average queue-sizes

$$\mathbb{E}\Big[\sum_{i,j} Q_{ij}\Big] \leq c\frac{n}{1-\rho(\lambda)}$$

where c is a universal constant

baseline performance + emulation

$$\mathbb{E}\Big[\sum_{i,j} Q_{ij}\Big] \leq 2\frac{n}{1-\rho(\lambda)} + n^3$$

Switched Flow Networks:
Multiple-hop

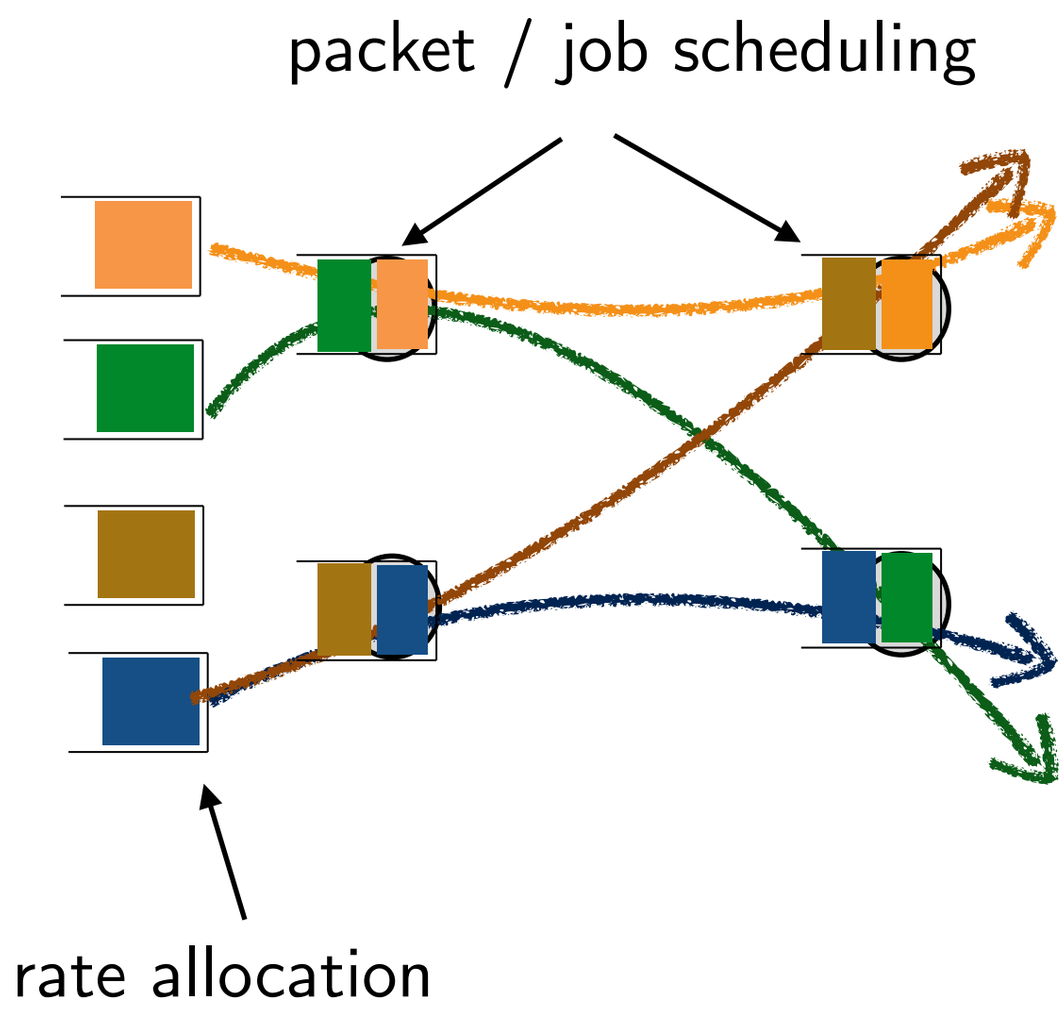
D. Shah and Q. Xie, "Centralized congestion control and scheduling in a data-center",
Preprint.

Switched Flow Networks

Flows are a collection of packets

Flows get rates assigned per which packets are injected in the network

Packets hop through network where each node decides service policy



Excellent model for data center networks cf. Perry-Balakrishnan-Shah (2014, 16)

Switched Flow Networks

Switched Flow Networks

Rate allocation

Switched Flow Networks

Rate allocation

Store-and-forward

Switched Flow Networks

Rate allocation

Store-and-forward

Packet injection in the network per Poisson process

Switched Flow Networks

Rate allocation

Store-and-forward

Packet injection in the network per Poisson process

Packet scheduling

Switched Flow Networks

Rate allocation

Store-and-forward

Packet injection in the network per Poisson process

Packet scheduling

Ideally, need to use LCFS or Processor Sharing

Switched Flow Networks

Rate allocation

Store-and-forward

Packet injection in the network per Poisson process

Packet scheduling

Ideally, need to use LCFS or Processor Sharing

But, nodes process packets in integral time-steps

Switched Flow Networks

Rate allocation

Store-and-forward

Packet injection in the network per Poisson process

Packet scheduling

Ideally, need to use LCFS or Processor Sharing

But, nodes process packets in integral time-steps

Need emulation, like before

Switched Flow Networks

Rate allocation

Store-and-forward

Packet injection in the network per Poisson process

Packet scheduling

Ideally, need to use LCFS or Processor Sharing

But, nodes process packets in integral time-steps

Need emulation, like before

But, now it needs to hold over a multi-hop network

Switched Flow Networks

Last-Come-First-Serve Emulation

Operate continuous time network per LCFS

Schedule packets in discrete-time network

using LCFS policy at integral time steps

but with respect to the arrival time of continuous network

Lemma (informally). The departure time of each packet from a queue in discrete time is within fixed constant of departure time from the corresponding ideal continuous time network as long as network has acyclic routing (allows for variable length packets).

That is, delay of both networks are within constant (independent of network size, load, etc.)

Switched Flow Networks

In summary:

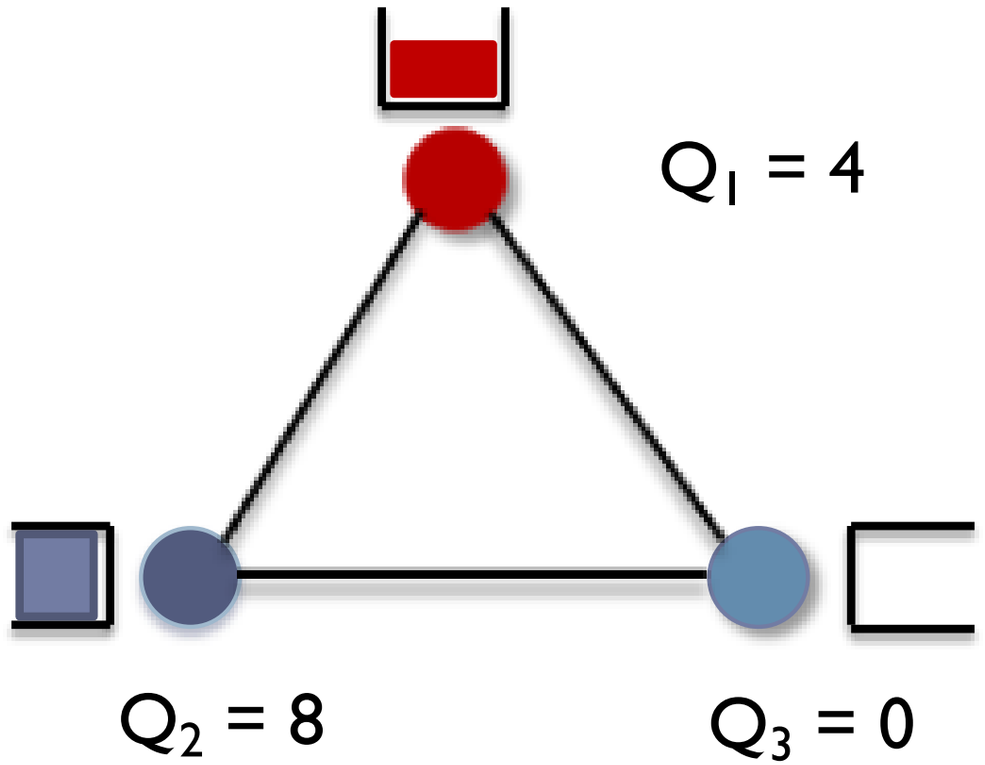
$$\text{total avg q-size of switched flow net.} = O\left(\frac{\text{number of hops}}{1-\rho}\right)$$

That is, we achieve *baseline* performance

Wireless Networks:
Distributed Implementation

D. Shah and J. Shin, "Randomized scheduling algorithm for queueing networks",
The Annals of Applied Probability, 2012.

Medium Access



Constraint

- No two neighbors can transmit at the same time
- Each node does not have knowledge of its neighbors

Medium Access As A Game

Medium Access As A Game

Rules

Medium Access As A Game

Rules

When asked

Medium Access As A Game

Rules

When asked

you may respond (raise hand) in < 100ms

Medium Access As A Game

Rules

When asked

you may respond (raise hand) in < 100ms

Winner

Medium Access As A Game

Rules

When asked

you may respond (raise hand) in < 100ms

Winner

if s/he is unique responder

Medium Access As A Game

Rules

When asked

you may respond (raise hand) in < 100ms

Winner

if s/he is unique responder

Reward

Medium Access As A Game

Rules

When asked

you may respond (raise hand) in < 100ms

Winner

if s/he is unique responder

Reward

bar of chocolate

Medium Access As A Game

Rules

When asked

you may respond (raise hand) in < 100ms

Winner

if s/he is unique responder

Reward

bar of chocolate

Goal

Medium Access As A Game

Rules

When asked

you may respond (raise hand) in < 100ms

Winner

if s/he is unique responder

Reward

bar of chocolate

Goal

Someone must win each time, and evenly across

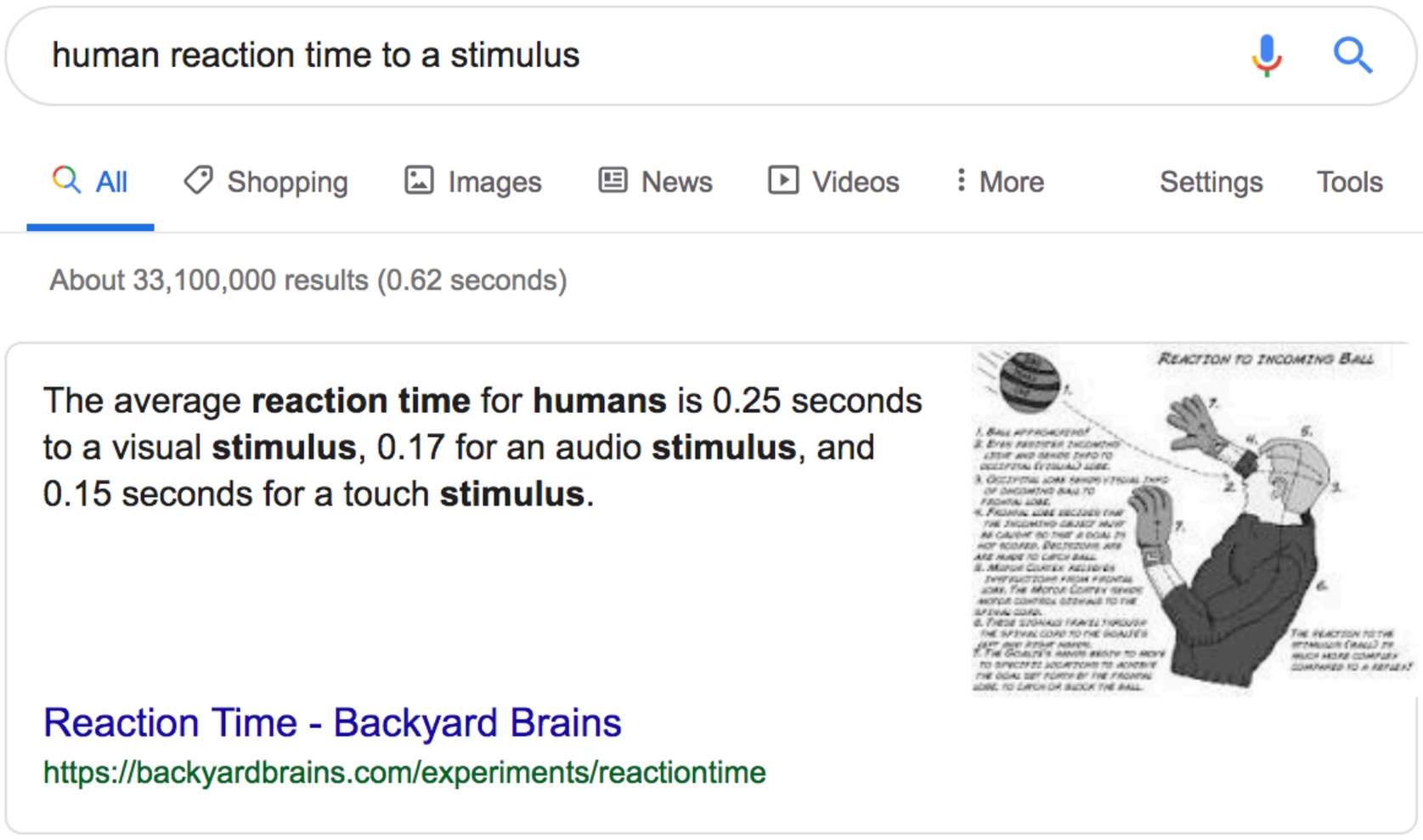
Medium Access As A Game

Rules

When asked

you may respond (raise hand) in < 100ms

V



I

Goal

S

Medium Access

Network G = (V, E)

V = {1, ..., n} being n nodes (wireless transmitters)

E edges capturing interference structure

Feasible transmission

$$\sigma = [\sigma_i] \in \{0,1\}^n$$

$$\sigma_i + \sigma_j \leq 1,\ (i,j) \in E$$

Distributed implementation

each node i need to decide $\sigma_i \in \{0,1\}$

so that most simultaneous transmissions can happen

Medium Access

Independent sets of G

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

Capacity region

$$\Lambda = \text{Convex Hull of } \mathcal{I}(G)$$

$$= \left\{ \sum_\sigma \alpha_\sigma \sigma,\ \alpha_\sigma \geq 0,\ \sum_\sigma \alpha_\sigma \leq 1 \right\}$$

Efficient Medium Access

Positive recurrent as long as arrival rate $\lambda \in \Lambda^o$

An Ideal Policy

MW policy

Queue size $Q = [Q_i]$

Choose schedule $\sigma^\star \in \mathcal{I}(G)$

where $\sigma^\star \in \arg\max_\sigma \sum_i \sigma_i f(Q_i)$

for choice of function $f : \mathbb{R} \to \mathbb{R}$ such that

$$f(0) = 0,\ f'(x) > 0,\ \forall x \geq 0$$

for example

$$f(x) = x,\ f(x) = \log(x+1),\ f(x) = \log\log(x+e)$$

cf. Andrews et al (2000), Stolyar (2004), Shah-Wischik (2006)

Distributed Implementation

How to find MW schedule in a distributed manner?

Answer: emulation!

Identify a reversible Markov chain on $\mathcal{I}(G)$

Per stationary distribution

$$\mathbb{E}\Big[\sum_i \sigma_i f(Q_i)\Big] \approx \sum_i \sigma_i^\star f(Q_i)$$

Find a distributed way to simulate the Markov chain

And, argue that changing queue-size does not cause an issue

Reversible Markov Chain

Markov chain on $\mathcal{I}(G)$

Given state $\sigma \in \mathcal{I}(G)$

Choose any i amongst n nodes uniformly at random

if $\sigma_i = 1$

then set it to 0 with prob. $\frac{1}{W_i}$

else if $\sigma_j = 0,\ \forall\ (i,j) \in E$

then set it to 1 with prob. 1

else

do nothing

Reversible Markov Chain

Markov chain on $\mathcal{I}(G)$

Stationary distribution: for $\sigma \in \mathcal{I}(G)$

$$\mathbb{P}(\sigma) \propto \exp\left(\sum_i \sigma_i \log W_i\right)$$

From Gibb's maximal principle (or variational characterization)

$$\mathbb{E}\Big[\sum_i \sigma_i \log W_i\Big] \geq \Big(\max_{\gamma \in \mathcal{I}(G)} \sum_i \gamma_i \log W_i\Big) - n$$

If queue-sizes were fixed, this would suffice to establish positive recurrence

as long as $\lambda \in \Lambda^o$

cf. Rajagopalan-Shah-Shin (2009)

Changing Queue-size Does not Matter

Roughly speaking (for W = f(Q))

Change in stationary distribution per unit time

$$\Delta\pi \approx f'(Q)$$

Reduction to stationary distribution due to step of MC

$$\Delta d(\mu,\pi) \approx \frac{1}{\text{mixing time}} \approx \frac{1}{f(Q)^n}$$

For large enough Q, we need

$$\Delta\pi \ll \Delta d(\mu,\pi) \;\Rightarrow\; f'(Q) \ll \frac{1}{f(Q)^n}$$

Changing Queue-size Does not Matter

For large enough Q, we need

$$\Delta\pi \ll \Delta d(\mu, \pi) \;\Rightarrow\; f'(Q) \ll \frac{1}{f(Q)^n}$$

For $f(x) = x$

$$f'(x) = 1 \gg x^{-n} = 1/f(x)^n$$

For $f(x) = \log(x+1)$

$$f'(x) = \frac{1}{x+1} \ll \log^{-n}(x+1) = 1/f(x)^n$$

Distributed Medium Access

In summary:

each node, say i, attempts transmission at each time t w.p. $p_i(t)$

setting $p_i(t)$

if transmission of node i was successful at time t-1, then

$$p_i(t) = 1 - \frac{1}{\log(Q_i(t)+1)}$$

if transmission of node i was failure at time t-1, then

$$p_i(t) = 0$$

if no attempt of transmission by node i

$p_i(t) = 0$, interfering neighbor attempted

$p_i(t) = 0.5$, otherwise

Baseline Performance and Discussion

J. M. Harrison, C. Mandayam, D. Shah, Y. Yang, "Resource sharing networks: Overview and an open problem". Stochastic Systems. 2014.

Where Are We?

Switched networks emulation of multi-class "product-form"

Achieves maximal capacity

Average queue-size achieves *baseline* performance

scales with number of facets of capacity polytope

Switched network emulation of reversible MC on space of schedules

Achieves maximal capacity

Distributed, computationally efficient implementation

What about getting all three: capacity, queue-size and implementation?

And, how good is *baseline* performance?

Where Are We?

What about getting all three: capacity, queue-size and implementation?

Unlikely by Shah-Tse-Tsitsiklis (2010)

There exist examples of switched networks s.t. not possible to have policy

Polynomial time computation per schedule

Polynomial size queues

Achieves at least a constant fraction of the capacity

subject to standard computational hypothesis

Where does baseline performance fit in?

Baseline performance

Baseline performance through emulation of Store-and-Forward

- Maximal capacity
- Linear queue-size in number of facets of capacity polytope
- Computation cost is not understood

Baseline performance

Baseline performance through emulation of Store-and-Forward

- Maximal capacity
- Linear queue-size in number of facets of capacity polytope
- Computation cost is not understood

There exists example of switched networks such that (cf. Xie-Shah 2019)

- Num. of facets of capacity polytope is super-polynomially in num. of ques.
- Follows from a reduction to facet complexity cf. Goos-Jain-Watson (2016)

Baseline performance

Baseline performance through emulation of Store-and-Forward

- Maximal capacity
- Linear queue-size in number of facets of capacity polytope
- Computation cost is not understood

There exists example of switched networks such that (cf. Xie-Shah 2019)

- Num. of facets of capacity polytope is super-polynomially in num. of ques.
- Follows from a reduction to facet complexity cf. Goos-Jain-Watson (2016)

Moreover, Baseline performance does not mean optimal performance

- e.g. under MW, queue-size always scales as $O(n^2/(1-\rho))$

Baseline performance

Baseline performance through emulation of Store-and-Forward

- Maximal capacity
- Linear queue-size in number of facets of capacity polytope
- Computation cost is not understood

Computation cost of SFA

- Asymptotically (in large queues) equivalent to Proportional Fair

  cf. Massoulie (2008)
- Proportional Fair is concave maximization over convex set
  - Poly number of oracle membership calls to convex set required
  - Oracle member may require poly computation in # of facets

So, Where Are We, In General?

Policy	Capacity	Queue-size	Complexity/ Implementation
Store-n-Forward Emulation	Maximal	$\frac{\lvert\text{facets}(\mathbb{S})\rvert}{1-\rho}$	poly$\left(\lvert\text{facets}(\mathbb{S})\rvert\right)$
Gibbs Emulation	Maximal	SuperPoly(n) x DoubleExp$(\frac{1}{1-\rho})$	fully distributed
Maximum Weight	Maximal	$\frac{n^2}{1-\rho}$	poly$\left(\lvert\text{facets}(\mathbb{S})\rvert\right)$
IDEAL	Maximal	$\frac{\lvert\text{facets}(\mathbb{S})\rvert}{1-\rho}$	poly$\left(n\right)$
IDEAL	Maximal	$\frac{n}{1-\rho}$	poly$\left(\lvert\text{facets}(\mathbb{S})\rvert\right)$

= conjecture

So, Where Are We, For Input-Queued Switch?

Policy	Capacity	Queue-size	Condition	Reference
Store-n-Forward Emulation	Maximal	$\mathbb{E}\Big[\sum_{i,j} Q_{ij}\Big] \le c\frac{n}{1-\rho(\lambda)}$	$\rho \ge 1 - O\Big(\frac{1}{n^2}\Big)$ any λ	Shah-Walton-Zhong (2014)
Maximum Weight	Maximal	$\mathbb{E}\Big[\sum_{i,j} Q_{ij}\Big] \le c\frac{n}{1-\rho(\lambda)}$	$\rho \ge 1 - O\big(\frac{1}{n^{4+}}\big)$ uniform $\lambda = [\rho]$	Maguluri-Srikant (2016)
Clever Batching	Maximal	$\mathbb{E}\Big[\sum_{ij} Q_{ij}\Big] \le \widetilde{O}\Big(\frac{n}{(1-\rho)^{4/3}}\Big)$	$1 - O\Big(\frac{1}{n}\Big) \ge \rho \ge 1 - O\Big(\frac{1}{n^2}\Big)$ uniform $\lambda = [\rho]$	Xu-Zhong (2019)
IDEAL	Maximal	$\mathbb{E}\Big[\sum_{i,j} Q_{ij}\Big] \le c\frac{n}{1-\rho(\lambda)}$	$\rho < 1$ any λ	?

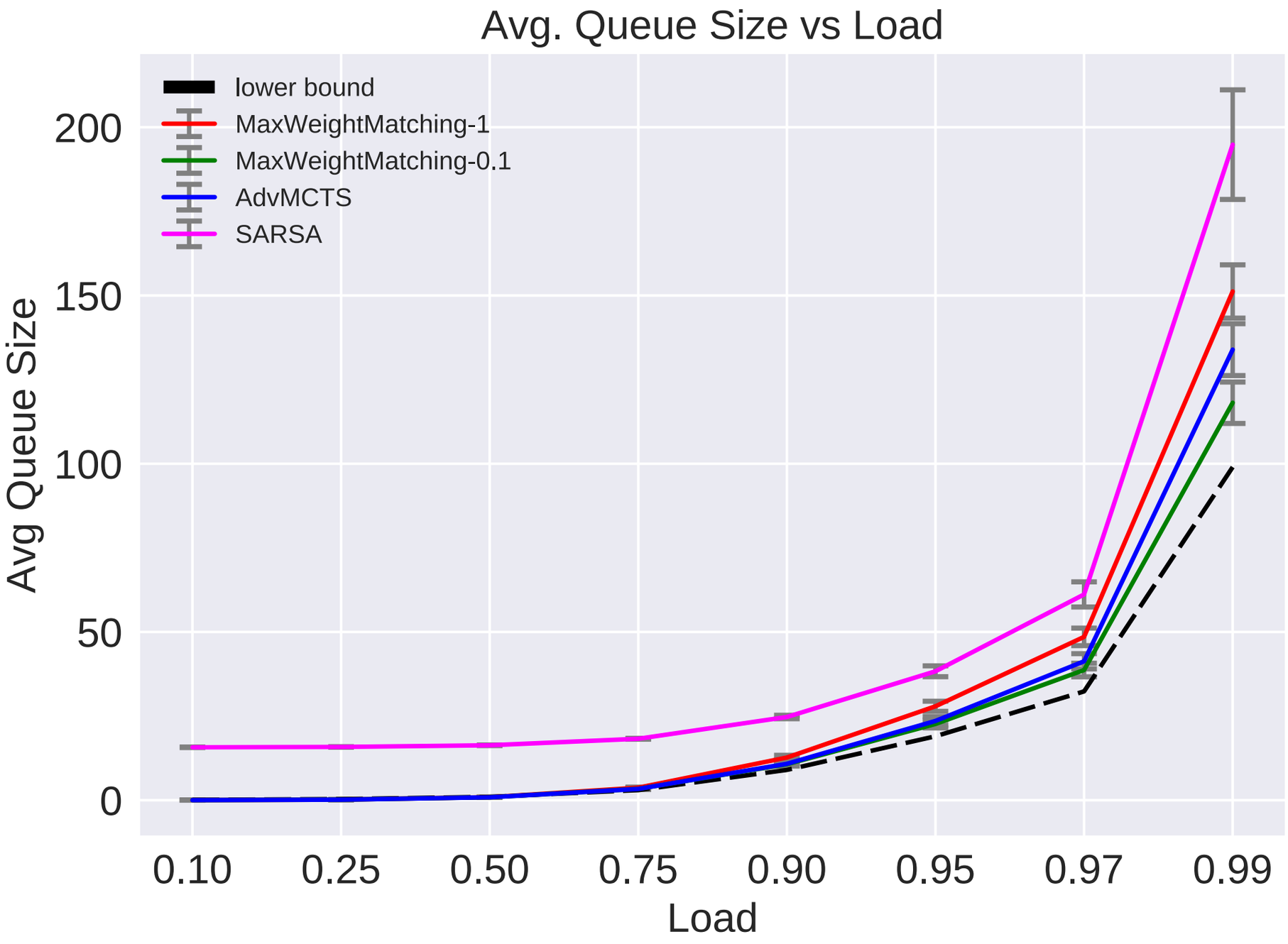
= conjecture

Reinforcement Learning and Resource Allocation

AlphaGoZero inspired Policy cf. Shah-Xie-Xu (2019)

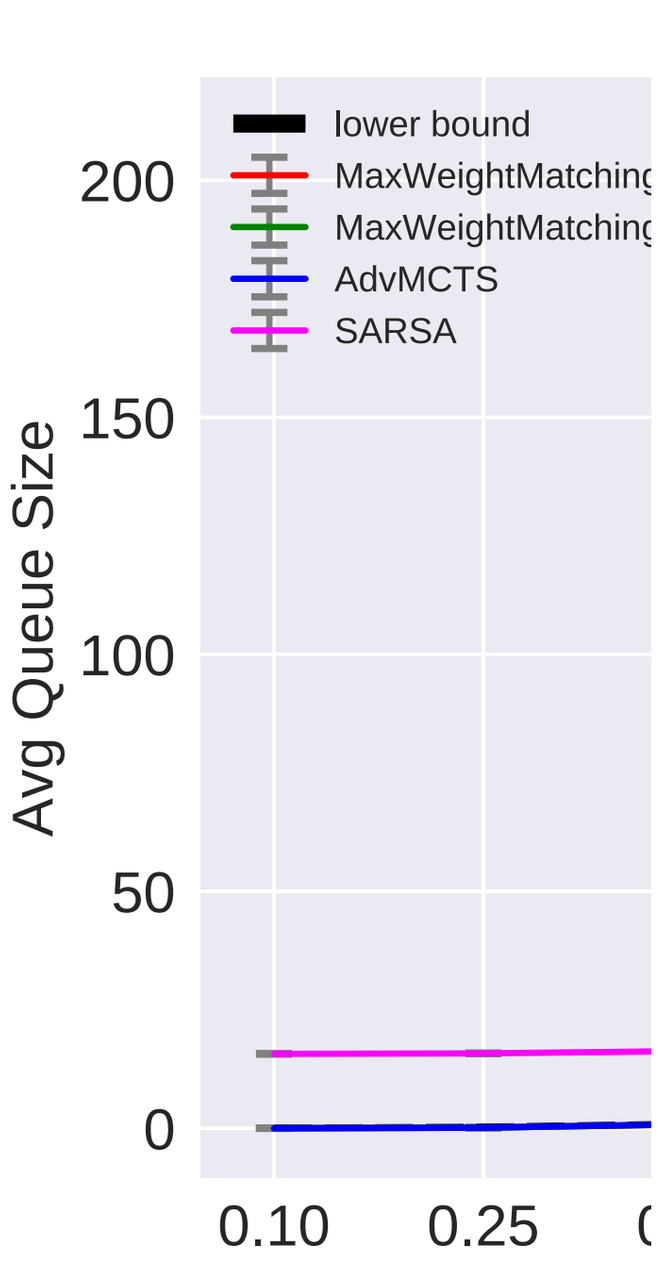
Reinforcement Learning and Resource Allocation

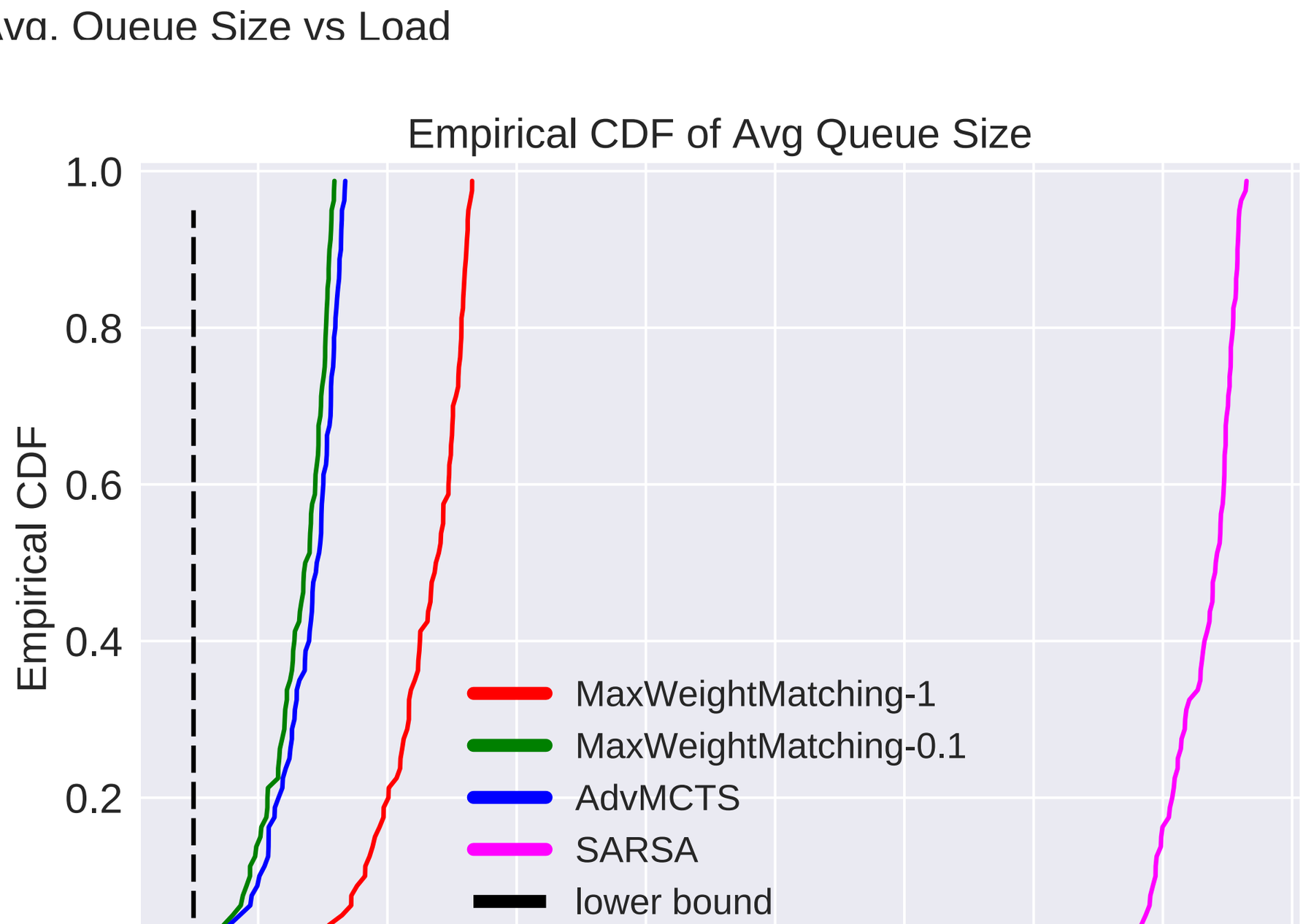
AlphaGoZero inspired Policy cf. Shah-Xie-Xu (2019)



Reinforcement Learning and Resource Allocation

AlphaGoZero inspired Policy cf. Shah-Xie-Xu (2019)





That's all, folks!

Thank you.