Instability, Stability and Non-Stabilizability of Queueing Networks

Yoni Nazarathy,

The University of Queensland,

Based on some joint work with

Erjen Lefeber, Eindhoven University of Technology,
Leonardo Rojas-Nandayapa, The University of Queensland,
Tom Salisbury, York University,
Gideon Weiss, The University of Haifa and The University of Southern California,
Hanqin Zhang, National University of Singapore.

University of Queensland, Mathematics Colloquium,
October 8, 2012.

Contents

Queues and Queueing Networks

The Study of Queueing Networks in Applied Probability and Operations Research

Congestion, delay and resource scarcity occurs in a variety of application areas:

- Customer service systems
- Complex manufacturing lines
- Telecommunication networks and computing systems
- Transportation networks

The Study of Queueing Networks in Applied Probability and Operations Research

Congestion, delay and resource scarcity occurs
in a variety of application areas:

- Customer service systems
- Complex manufacturing lines
- Telecommunication networks and computing systems
- Transportation networks

Stochastic queueing network models often capture the essentials of such examples allowing for quantitative performance evaluation, optimization and control

The Study of Queueing Networks in Applied Probability and Operations Research

Congestion, delay and resource scarcity occurs
in a variety of application areas:

- Customer service systems
- Complex manufacturing lines
- Telecommunication networks and computing systems
- Transportation networks

Stochastic queueing network models often capture the essentials of such examples allowing for quantitative performance evaluation, optimization and control

Researchers "doing queueing" currently at UQ

Dirk Kroese, Ross McVinish, Y.N., Phil Pollett,
Leonardo Rojas-Nandayapa,...

A Single Queue

Items arrive at random times to a server, queue up, each requiring service for a random duration, then depart

A Single Queue

Items arrive at random times to a server, queue up, each requiring service for a random duration, then depart

Construction of the Queue Length Process: $Q(t)$

$A(t) \equiv$ counting process generated by a sequence of random **inter-arrival times** each with mean λ^{-1}

$S(t) \equiv$ counting process generated by a sequence of random **service times** each with mean μ^{-1}

A Single Queue

Items arrive at random times to a server, queue up, each requiring service for a random duration, then depart

Construction of the Queue Length Process: $Q(t)$

$A(t) \equiv$ counting process generated by a sequence of random **inter-arrival times** each with mean λ^{-1}

$S(t) \equiv$ counting process generated by a sequence of random **service times** each with mean μ^{-1}

$$\begin{aligned} Q(t) &= Q(0) + A(t) - S\big(T(t)\big) \\ T(t) &= \int_0^t \mathbf{1}_{\{Q(s)>0\}} ds \end{aligned}$$

A Single Queue

Items arrive at random times to a server, queue up, each requiring service for a random duration, then depart

Construction of the Queue Length Process: $Q(t)$

$A(t) \equiv$ counting process generated by a sequence of random **inter-arrival times** each with mean λ^{-1}

$S(t) \equiv$ counting process generated by a sequence of random **service times** each with mean μ^{-1}

$$\begin{aligned} Q(t) &= Q(0) + A(t) - S\big(T(t)\big) \\ T(t) &= \int_0^t \mathbf{1}_{\{Q(s)>0\}} ds \end{aligned}$$

The Load $\rho = \frac{\lambda}{\mu}$

A Single Queue

Items arrive at random times to a server, queue up, each requiring service for a random duration, then depart

Construction of the Queue Length Process: $Q(t)$

$A(t) \equiv$ counting process generated by a sequence of random **inter-arrival times** each with mean λ^{-1}

$S(t) \equiv$ counting process generated by a sequence of random **service times** each with mean μ^{-1}

$$\begin{aligned} Q(t) &= Q(0) + A(t) - S\big(T(t)\big) \\ T(t) &= \int_0^t \mathbf{1}_{\{Q(s)>0\}} ds \end{aligned}$$

The Load $\rho = \frac{\lambda}{\mu}$

- $\rho < 1$ queue is **stable**

A Single Queue

Items arrive at random times to a server, queue up, each requiring service for a random duration, then depart

Construction of the Queue Length Process: $Q(t)$

$A(t) \equiv$ counting process generated by a sequence of random **inter-arrival times** each with mean λ^{-1}

$S(t) \equiv$ counting process generated by a sequence of random **service times** each with mean μ^{-1}

$$\begin{aligned} Q(t) &= Q(0) + A(t) - S\big(T(t)\big) \\ T(t) &= \int_0^t \mathbf{1}_{\{Q(s)>0\}} ds \end{aligned}$$

The Load $\rho = \frac{\lambda}{\mu}$

- $\rho < 1$ queue is **stable**
- $\rho > 1$ queue is **unstable**

A Single Queue

Items arrive at random times to a server, queue up, each requiring service for a random duration, then depart

Construction of the Queue Length Process: $Q(t)$

$A(t) \equiv$ counting process generated by a sequence of random **inter-arrival times** each with mean λ^{-1}

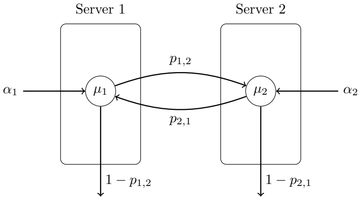
$S(t) \equiv$ counting process generated by a sequence of random **service times** each with mean μ^{-1}

$$\begin{aligned} Q(t) &= Q(0) + A(t) - S\big(T(t)\big) \\ T(t) &= \int_0^t \mathbf{1}_{\{Q(s)>0\}} ds \end{aligned}$$

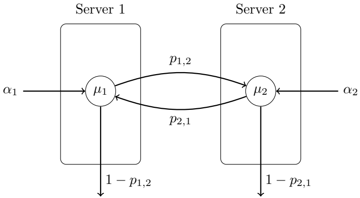
The Load $\rho = \frac{\lambda}{\mu}$

- $\rho < 1$ queue is **stable**
- $\rho > 1$ queue is **unstable**
- $\rho = 1$ queue is **critically unstable**

The Flavor of Classic Queueing Network Results

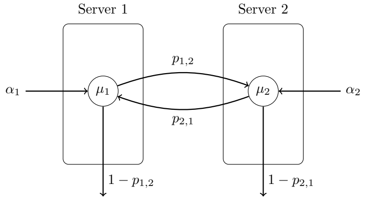


The Flavor of Classic Queueing Network Results



$$\begin{array}{rcl} \lambda_1 & = & \alpha_1 + p_{2,1}\lambda_2 \\ \lambda_2 & = & \alpha_2 + p_{1,2}\lambda_1 \end{array}, \qquad \rho_i = \frac{\lambda_i}{\mu_i}, \quad i = 1, 2.$$

The Flavor of Classic Queueing Network Results



$$\begin{aligned} \lambda_1 &= \alpha_1 + p_{2,1}\lambda_2 \\ \lambda_2 &= \alpha_2 + p_{1,2}\lambda_1 \end{aligned}, \qquad \rho_i = \frac{\lambda_i}{\mu_i}, \quad i = 1, 2.$$

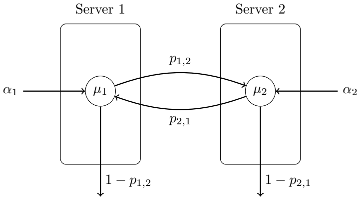
A Product Form Result

Assume Poisson arrival and service processes. If $\rho_1, \rho_2 < 1$ then,

$$\lim_{t\to\infty} P\Big(Q_1(t) = k_1,\ Q_2(t) = k_2\Big) = \prod_{i=1}^{2}(1-\rho_i)\rho_i^{k_i},$$

otherwise the network is not stable.

The Flavor of Classic Queueing Network Results



$$\begin{aligned} \lambda_1 &= \alpha_1 + p_{2,1}\lambda_2 \\ \lambda_2 &= \alpha_2 + p_{1,2}\lambda_1 \end{aligned}, \qquad \rho_i = \frac{\lambda_i}{\mu_i}, \quad i = 1, 2.$$

A Product Form Result

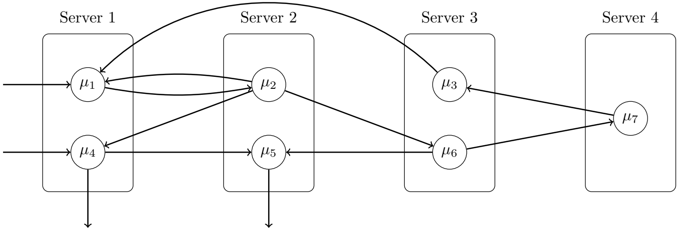
Assume Poisson arrival and service processes. If $\rho_1, \rho_2 < 1$ then,

$$\lim_{t \to \infty} P\Big(Q_1(t) = k_1,\ Q_2(t) = k_2\Big) = \prod_{i=1}^{2} (1 - \rho_i)\rho_i^{k_i},$$

otherwise the network is not stable.

Note: Without the Poisson assumptions the product form typically does not hold, yet the stability properties are the same

Multi-Class Queueing Networks

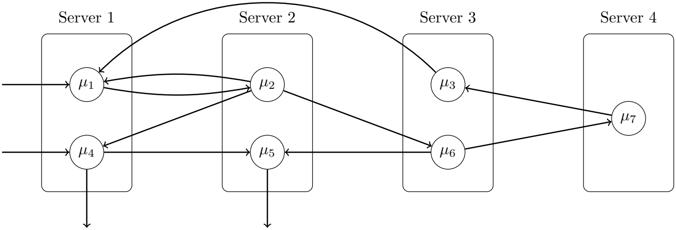


Control Policies

Now there is a choice as to how to allocate server resources:

Policy: What operation should be served by each of the servers at every time instant based on the current state

Multi-Class Queueing Networks



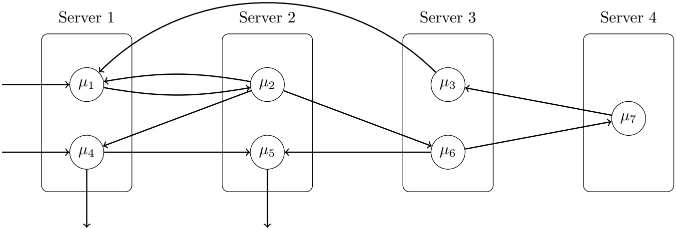
Control Policies

Now there is a choice as to how to allocate server resources:

Policy: What operation should be served by each of the servers at every time instant based on the current state

- Explicit performance analysis for a given control policy is typically intractable

Multi-Class Queueing Networks



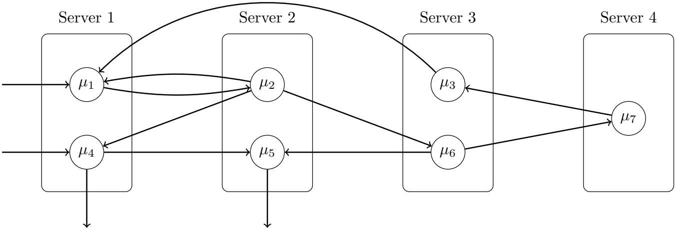
Control Policies

Now there is a choice as to how to allocate server resources:

Policy: What operation should be served by each of the servers at every time instant based on the current state

- Explicit performance analysis for a given control policy is typically intractable
- Optimal control is typically out of the question

Multi-Class Queueing Networks



Control Policies

Now there is a choice as to how to allocate server resources:

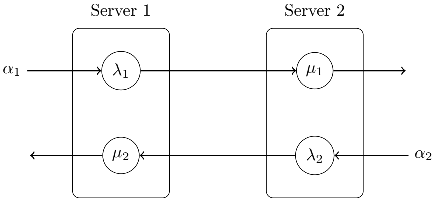
Policy: What operation should be served by each of the servers at every time instant based on the current state

- Explicit performance analysis for a given control policy is typically intractable
- Optimal control is typically out of the question

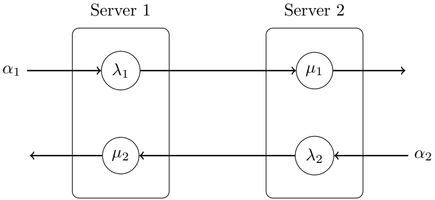
A realistic research goal: understanding stability

Instability

The Kumar-Seidman-Rybko-Stolyar Network



The Kumar-Seidman-Rybko-Stolyar Network

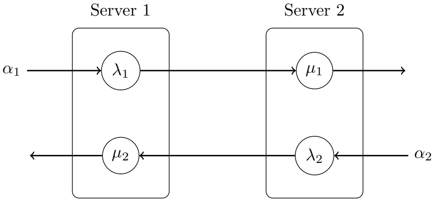


Load Conditions

Necessary condition for stability:

$$\rho_1 = \alpha_1 \frac{1}{\lambda_1} + \alpha_2 \frac{1}{\mu_2} < 1, \qquad \rho_2 = \alpha_1 \frac{1}{\mu_1} + \alpha_2 \frac{1}{\lambda_2} < 1.$$

The Kumar-Seidman-Rybko-Stolyar Network



Load Conditions

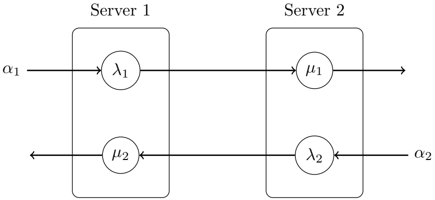
Necessary condition for stability:

$$\rho_1 = \alpha_1 \frac{1}{\lambda_1} + \alpha_2 \frac{1}{\mu_2} < 1, \qquad \rho_2 = \alpha_1 \frac{1}{\mu_1} + \alpha_2 \frac{1}{\lambda_2} < 1.$$

A Control Question

If $\rho_i < 1$, $i = 1, 2$, are all **work conserving** policies stabilizing?

The Kumar-Seidman-Rybko-Stolyar Network



Load Conditions

Necessary condition for stability:

$$\rho_1 = \alpha_1 \frac{1}{\lambda_1} + \alpha_2 \frac{1}{\mu_2} < 1, \qquad \rho_2 = \alpha_1 \frac{1}{\mu_1} + \alpha_2 \frac{1}{\lambda_2} < 1.$$

A Control Question

If $\rho_i < 1$, $i = 1, 2$, are all **work conserving** policies stabilizing?

KSRS Adversarial Idea: Try the **pull-priority** Policy

Give priority to pull operations, μ_1, μ_2, over push operations λ_1, λ_2

Dynamics of Pull-Priority KSRS

Illustrative example in which the load conditions hold:

$$\left(\alpha_i = 3,\ \lambda_i = 10,\ \mu_i = 5\right) \quad \Longrightarrow \quad \rho_i = 9/10, \quad i = 1, 2.$$

Dynamics of Pull-Priority KSRS

Illustrative example in which the load conditions hold:

$$\big(\alpha_i = 3,\ \lambda_i = 10,\ \mu_i = 5\big) \quad \Longrightarrow \quad \rho_i = 9/10, \quad i = 1, 2.$$

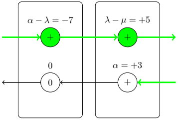
Illustration of instability by means of deterministic **fluid** dynamics:

Dynamics of Pull-Priority KSRS

Illustrative example in which the load conditions hold:

$$\left(\alpha_i = 3,\ \lambda_i = 10,\ \mu_i = 5\right) \quad \Longrightarrow \quad \rho_i = 9/10, \quad i = 1, 2.$$

Illustration of instability by means of deterministic **fluid** dynamics:

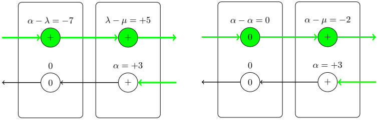


Dynamics of Pull-Priority KSRS

Illustrative example in which the load conditions hold:

$$\big(\alpha_i = 3,\ \lambda_i = 10,\ \mu_i = 5\big) \quad \Longrightarrow \quad \rho_i = 9/10, \quad i = 1, 2.$$

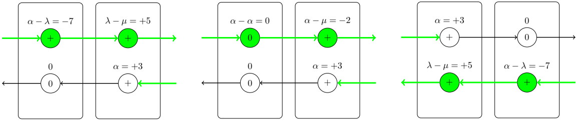
Illustration of instability by means of deterministic **fluid** dynamics:



Illustrative example in which the load conditions hold:

$$\left(\alpha_i = 3,\ \lambda_i = 10,\ \mu_i = 5\right) \quad \Longrightarrow \quad \rho_i = 9/10, \quad i = 1, 2.$$

Illustration of instability by means of deterministic **fluid** dynamics:

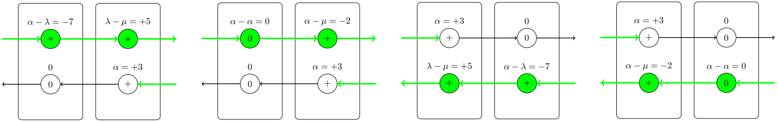


Dynamics of Pull-Priority KSRS

Illustrative example in which the load conditions hold:

$$\big(\alpha_i = 3,\ \lambda_i = 10,\ \mu_i = 5\big) \quad \Longrightarrow \quad \rho_i = 9/10, \quad i = 1, 2.$$

Illustration of instability by means of deterministic **fluid** dynamics:

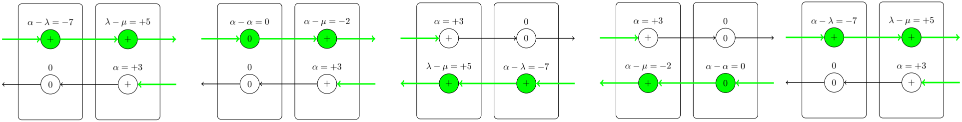


Dynamics of Pull-Priority KSRS

Illustrative example in which the load conditions hold:

$$\left(\alpha_i = 3,\ \lambda_i = 10,\ \mu_i = 5\right) \quad \Longrightarrow \quad \rho_i = 9/10, \quad i = 1, 2.$$

Illustration of instability by means of deterministic **fluid** dynamics:

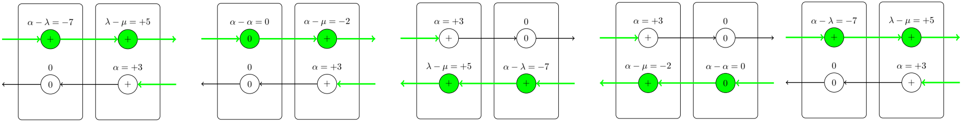


Dynamics of Pull-Priority KSRS

Illustrative example in which the load conditions hold:

$$\left(\alpha_i = 3,\ \lambda_i = 10,\ \mu_i = 5\right) \quad \Longrightarrow \quad \rho_i = 9/10, \quad i = 1, 2.$$

Illustration of instability by means of deterministic **fluid** dynamics:



A Virtual Server

Observation: Pull operations "never" occur at the same time.
Thus with this policy, an additional condition for stability is:

$$\rho_v := \alpha_1 \frac{1}{\mu_1} + \alpha_2 \frac{1}{\mu_2} < 1$$

Some Notes and Comments

Lessons Learned from KSRS

- Stability is not just a property of the network but rather of both the network and the control policy – this is in stark difference to classic queueing networks
- "Innocent looking" control policies can be very bad
- This is easy to detect (and fix) for small toy examples such as KSRS - but what about big complex manufacturing networks?
- The sub-field of *stability analysis of queueing networks* was "born" (early 90's)

Summarizing Books (including KSRS and beyond)

- *Stability of Queueing Networks*, Maury Bramson, 2008.
- *Control Techniques for Complex Networks*, Sean Meyn, 2008.
- *Fundamentals of Queueing*, Hong Chen, David Yao, 2001.

Queueing Networks with Infinite Supplies

A Different Kind of Model

Many real life systems often operate some servers at full utilization yet in previous models $\rho = 1$ implies critically unstable behavior

A Different Kind of Model

Many real life systems often operate some servers at full utilization yet in previous models $\rho = 1$ implies critically unstable behavior

Infinite Supply Models

- Assume that severs generate arrivals to the network by having some of the queues that never run out of work
- This allows full utilization of servers
- Analyze stability for

non-idling (fully-utilizing) control policies

A Different Kind of Model

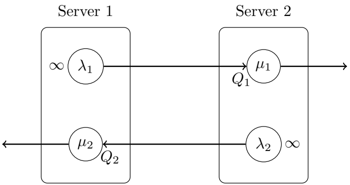
Many real life systems often operate some servers at full utilization yet in previous models $\rho = 1$ implies critically unstable behavior

Infinite Supply Models

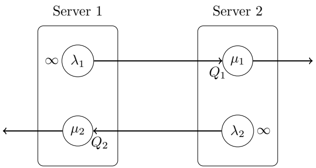
- Assume that severs generate arrivals to the network by having some of the queues that never run out of work
- This allows full utilization of servers
- Analyze stability for

non-idling (fully-utilizing) control policies

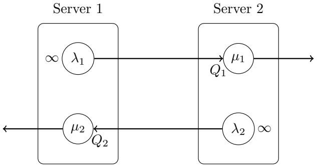
The simplest example is Gideon Weiss's **push-pull network**:



The Push-Pull Queueing Network as a Markov Chain

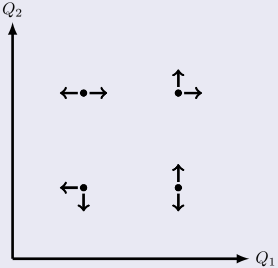


The Push-Pull Queueing Network as a Markov Chain

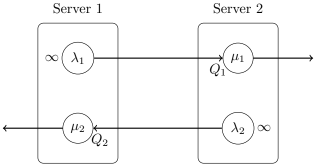


Policies and Markov Chains

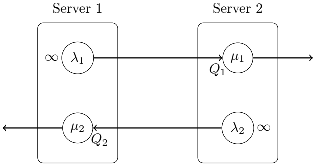
Under **Poisson assumptions**, every control policy $\mathcal{P} : \mathbb{Z}_+^2 \to \{\text{push}, \text{pull}\}^2$ (with restrictions at the axes) implies a Markov chain on $\mathbb{Z}_+^2$



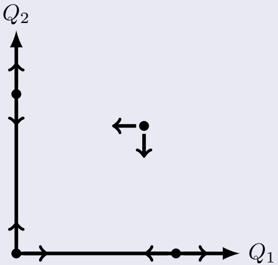
The Push-Pull Network with $\lambda_i < \mu_i$



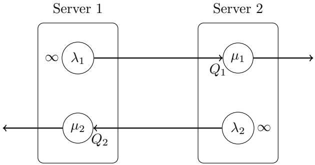
The Push-Pull Network with $\lambda_i < \mu_i$



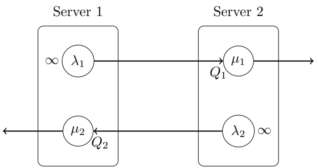
Pull-Priority Policy is Stabilizing



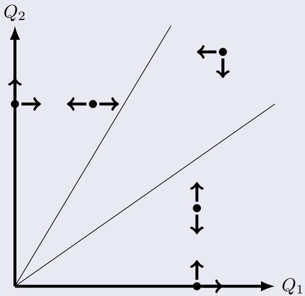
The Push-Pull Network with $\lambda_i > \mu_i$



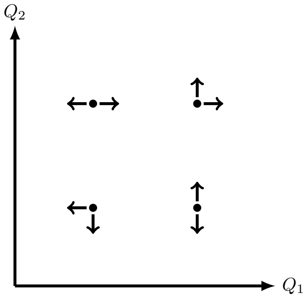
The Push-Pull Network with $\lambda_i > \mu_i$



A Threshold Policy is Stabilizing

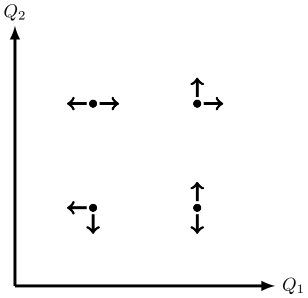


The Push-Pull Network with $\lambda_i = \mu_i$



Is there a $\mathcal{P} : \mathbb{Z}_+^2 \to \{\text{push}, \text{pull}\}^2$ (with restrictions at the axes) such that the Markov chain has a **positive recurrent** state that is reached w.p. 1?

The Push-Pull Network with $\lambda_i = \mu_i$

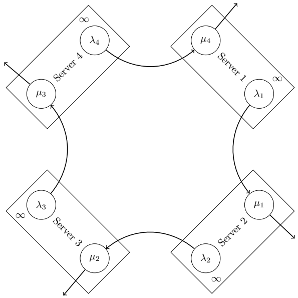


Is there a $\mathcal{P} : \mathbb{Z}_+^2 \to \{\mathsf{push}, \mathsf{pull}\}^2$ (with restrictions at the axes) such that the Markov chain has a **positive recurrent** state that is reached w.p. 1?

We'll get back to this in a few minutes....

Push-Pull Rings

Push-Pull Rings with Pull-Priority Policy



Generalizes the Push to M servers

- M queues, each with "potential load", $\gamma_i = \frac{\lambda_i}{\mu_i}$
- Pull-priority policy
- An interesting case is when $\gamma_i > 1$ but "not too large":
  It turns out that odd rings are stable yet even rings are not

Stochastic Model and Fluid Model

Stochastic Model $(Q(t), T(t))$

$$
\begin{aligned}
Q_i(t) &= Q_i(0) + S_{i,1}\big(T_{i,1}(t)\big) - S_{i,2}\big(T_{i,2}(t)\big) \\
t &= T_{i,1}(t) + T_{i-1,2}(t) \\
0 &= \int_0^t Q_i(s)\,dT_{i+1,1}(s)
\end{aligned}
$$

Associated Fluid Model $(\bar{Q}(t), \bar{T}(t))$

$$
\begin{aligned}
\bar{Q}_i(t) &= \bar{Q}_i(0) + \lambda_i \bar{T}_{i,1}(t) - \mu_i \bar{T}_{i,2}(t) \\
t &= \bar{T}_{i,1}(t) + \bar{T}_{i-1,2}(t) \\
0 &= \int_0^t \bar{Q}_i(s)\,d\bar{T}_{i+1,1}(s)
\end{aligned}
$$

Fluid Stability Framework

Thm: (Dai '95), adapted to infinite supplies

Assume minor technical assumptions on the processing time distributions. If there exists a τ such that for all solutions of the fluid model and all $t \geq \tau$, $\sum \bar{Q}_i(t) = 0$ then the (stochastic) network is stable.

- All solutions of the fluid model are Lipschitz, thus have derivatives a.e.
- **Regular time points**: Time points at which derivatives exists

Lemma

If we have a Lyapounov function: $V : \mathbb{R}^M \to \mathbb{R}$ such that for all regular time points of all solutions of the fluid model, $\frac{d}{dt} V\left(\bar{Q}(t)\right) < -\epsilon$ for some $\epsilon > 0$, then the fluid model is stable.

Stability Result: M odd, $\gamma_i > 1$

Theorem (Erjen Lefeber, Gideon Weiss, Y.N.)

The push-pull ring with M **odd**, $\gamma_i > 1$ for all i, operating under a pull-priority policy is stable if $\Delta < 0$, where

$$\Delta = \sum_{i=1}^{M} c_i \Big(\frac{M-1}{2}(\gamma_i - 1) - 1 \Big),$$

with,

$$c_i = ((\cdots(((\gamma_{i-1} - 1)\gamma_{i-2} + 1)\gamma_{i-3} - 1)\gamma_{i-4} \cdots\cdots\cdots)\gamma_{i+2} - 1)\gamma_{i+1} + 1.$$

Note: If $\gamma_i = \gamma$ for all i then the stability condition reduces to:

$$\gamma < 1 + \frac{1}{\frac{M-1}{2}}$$

Proof Overview

1. Use $V(x) = \sum_{i=1}^{M} c_i\, x_i$ as Lyapounov function for the fluid model with coefficients, c_i, designed based on the intuition that "typical" fluid trajectories eventually cycle on states of the form (e.g. $M = 5$):

   $$(+,0,+,0,+),\ \ (+,+,0,+,0),\ \ (0,+,+,0,+),\ \ (+,0,+,+,0),\ \ (0,+,0,+,+).$$

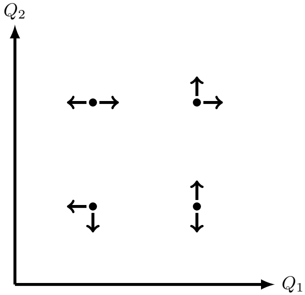
   The c_i are such that $\dot{V}(t)$ is constant during such cycles:

   $$\begin{bmatrix} -1 & 0 & \gamma_3-1 & 0 & \gamma_5-1 \\ \gamma_1-1 & -1 & 0 & \gamma_4-1 & 0 \\ 0 & \gamma_2-1 & -1 & 0 & \gamma_5-1 \\ \gamma_1-1 & 0 & \gamma_3-1 & -1 & 0 \\ 0 & \gamma_2-1 & 0 & \gamma_4-1 & -1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{bmatrix} = \begin{bmatrix} \Delta \\ \Delta \\ \Delta \\ \Delta \\ \Delta \end{bmatrix}$$

2. Characterize the regular time points and show that on these time points the Lyapounov function has negative drift
3. Now apply Jim Dai's stability framework

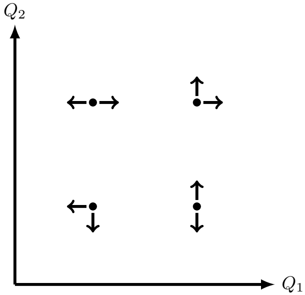
Non-Stabilizability

The Push-Pull Network with $\lambda_i = \mu_i$



Is there a $\mathcal{P} : \mathbb{Z}_+^2 \to \{\text{push}, \text{pull}\}^2$ (with restrictions at the axes) such that the Markov chain has a positive recurrent state that is reached w.p. 1?

The Push-Pull Network with $\lambda_i = \mu_i$



Is there a $\mathcal{P} : \mathbb{Z}_+^2 \to \{\text{push}, \text{pull}\}^2$ (with restrictions at the axes) such that the Markov chain has a positive recurrent state that is reached w.p. 1?

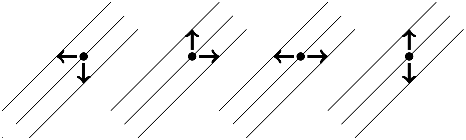
Theorem (Leonardo Rojas-Nandayapa, Tom Salisbury, Y.N.)

The push-pull network with $\lambda_i = \mu_i, i = 1, 2$ is non-stabilizable.

Non-stabilizability Proof

(This version assumes: $\lambda_1 = \mu_1 = \mu_2 = \lambda_2$) for simplicity

1. Set x_n as the embedded Markov chain of $X(t) = (Q_1(t), Q_2(t))$
2. Define $g\Big((x_1, x_2)\Big) = x_1 - x_2$ and $Z_n = g(X_n)$
   Z_n is a martingale for any $\mathcal{P}$:



3. Assume $\exists$ positive recurrent $\mathcal{B} \subset \mathbb{Z}_+^2$.
   Take $x, y \in \mathcal{B}$ with $g(x) \neq g(y)$
4. Set $X_0 = x$ and define $T = \inf\{n \geq 0 : X_n = y\}$
5. For $\mathcal{B}$ to be positive recurrent, $E[T] < \infty$ so:

$$g(x) = E[Z_0] = E[Z_T] = g(y),$$

a contradiction.

A More General Result for Non-Stabilizability

The idea of finding a linear-martingale simultaneously for all possible policies turns out to be fruitful in greater generality:

A More General Result for Non-Stabilizability

The idea of finding a linear-martingale simultaneously for all possible policies turns out to be fruitful in greater generality:

Theorem (Leonardo Rojas-Nandayapa, Tom Salisbury, Y.N.)

Consider controlled queueing networks $\{X_n, \ \ n \geq \ 0\}$ on $\mathbb{Z}_+^M$ with $L < \infty$ possible actions. Denote, by $\mathbf{D}$ the $L \times M$ matrix with rows,

$$\mathbf{\Delta}_i := E_{\text{action } i}[X_{n+1} - X_n \,|\, X_n], \quad i = 1, \ldots, L.$$

Then subject to technical non-degeneracy conditions, if

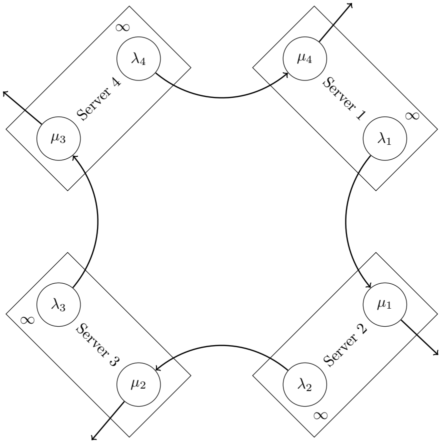
$$\text{rank}\big(\mathbf{D}\big) < M,$$

then the network is non-stabilizable.

Applications to Structured Networks

Corollary (Leonardo Rojas-Nandayapa, Tom Salisbury, Y.N.)

Push-Pull Rings with M **even** and $\lambda_i = \mu_i$ are non-stabilizable.



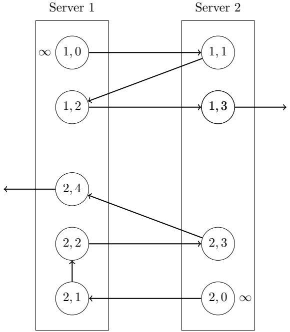
Applications to Structured Networks cont.

Corollary (Leonardo Rojas-Nandayapa, Tom Salisbury, Y.N.)

Consider an infinite supply network with 2 servers and S streams. Assume,

$$\sum_{j \in C_1(i)} \mu_{i,j}^{-1} = \sum_{j \in C_2(i)} \mu_{i,j}^{-1}, \quad i = 1, \ldots, S,$$

then the network is non-stabilizable.



Wrap-up

Current Projects Related to Stability Properties of Queueing Networks

- With Leonardo Rojas-Nandayapa and Tom Salisbury: Non-stabilizability of similar models under general processing time assumptions (can not use Martingale method as is)
- With Erjen Lefeber and Dieter Armbruster: It is known that in certain cases stability depends on the distributional assumptions. We have an illustration of this phenomenon based on deterministic dynamical (hybrid) systems
- With Gideon Weiss and Erjen Lefeber: General stability results for general queueing networks with infinite supplies
- Long term interest: Designing and understanding stabilizing adaptive control methods for complex queueing networks