Controller design for flow networks of switched servers with setup times

Erjen Lefeber

Eindhoven University of Technology

AG Meeting
March 26, 2008, Eindhoven

Motivation





Problem

Problem

How to control these networks?

Decisions: When to switch, and to which job-type

Goals: Minimal number of jobs, minimal flow time

Current approach

Start from policy, analyze resulting dynamics

Kumar, Seidman (1990)

Problem

Problem

How to control these networks?

Decisions: When to switch, and to which job-type

Goals: Minimal number of jobs, minimal flow time

Current approach

Start from policy, analyze resulting dynamics

Kumar, Seidman (1990)

Problem

Problem

How to control these networks?

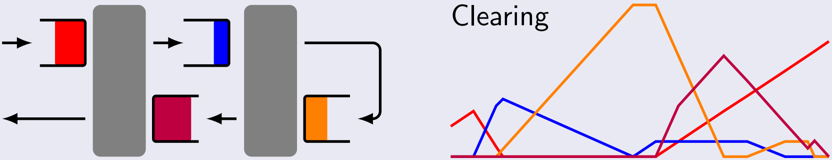
Decisions: When to switch, and to which job-type

Goals: Minimal number of jobs, minimal flow time

Current approach

Start from policy, analyze resulting dynamics

Kumar, Seidman (1990)



Problem

Current status (after two decades)

Several policies exist that guarantee stability of the network

Remark

Stability is only a prerequisite for a good policy

Open issues

- Do existing policies yield satisfactory network performance?
- How to obtain pre-specified network behavior?

Main subject of study (modest)

Fixed, deterministic flow networks (not evolving, constant inflow)

Problem

Current status (after two decades)

Several policies exist that guarantee stability of the network

Remark

Stability is only a prerequisite for a good policy

Open issues

- Do existing policies yield satisfactory network performance?
- How to obtain pre-specified network behavior?

Main subject of study (modest)

Fixed, deterministic flow networks (not evolving, constant inflow)

Problem

Current status (after two decades)

Several policies exist that guarantee stability of the network

Remark

Stability is only a prerequisite for a good policy

Open issues

- Do existing policies yield satisfactory network performance?
- How to obtain pre-specified network behavior?

Main subject of study (modest)

Fixed, deterministic flow networks (not evolving, constant inflow)

Problem

Current status (after two decades)

Several policies exist that guarantee stability of the network

Remark

Stability is only a prerequisite for a good policy

Open issues

- Do existing policies yield satisfactory network performance?
- How to obtain pre-specified network behavior?

Main subject of study (modest)

Fixed, deterministic flow networks (not evolving, constant inflow)

Approach

Notions from control theory

1. Generate feasible reference trajectory
2. Design (static) state feedback controller
3. Design observer
4. Design (dynamic) output feedback controller

Parallels with this problem

1. Determine desired system behavior
2. Derive non-distributed/centralized controller
3. Can state be reconstructed?
4. Derive distributed/decentralized controller

Approach

Notions from control theory

1. Generate feasible reference trajectory
2. Design (static) state feedback controller
3. Design observer
4. Design (dynamic) output feedback controller

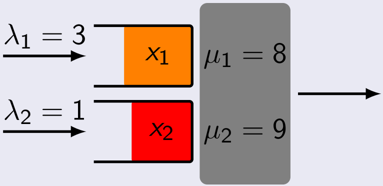
Parallels with this problem

1. Determine desired system behavior
2. Derive non-distributed/centralized controller
3. Can state be reconstructed?
4. Derive distributed/decentralized controller

Example 1: Single machine

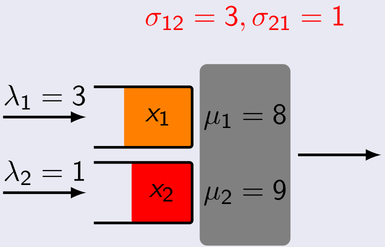
Single machine

$\sigma_{12} = 3, \sigma_{21} = 1$



Example 1: Single machine

Single machine



State

x_0	remaining setup time
x_i	buffer contents ($i = 1, 2$)
m	mode $\in \{1, 2\}$

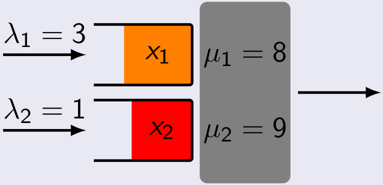
Input

u_0	activity $\in \{①, ②, ❶, ❷\}$
u_i	service rate step $i = 1, 2$

Example 1: Single machine

Single machine

$\sigma_{12} = 3, \sigma_{21} = 1$



Continuous dynamics

$$\dot{x}_0(t) = \begin{cases} -1 & \text{if } u_0 \in \{❶, ❷\} \\ 0 & \text{if } u_0 \in \{①, ②\} \end{cases}$$

$$\dot{x}_1(t) = \lambda_1 - u_1(t)$$

$$\dot{x}_2(t) = \lambda_2 - u_2(t)$$

Discrete event dynamics

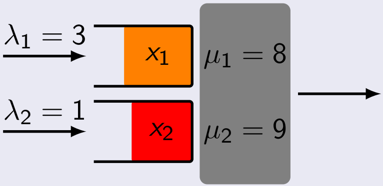
$$x_0 := \sigma_{21} \qquad m := 1 \qquad \text{if } u_0 = ❶ \text{ and } m = 2$$

$$x_0 := \sigma_{12} \qquad m := 2 \qquad \text{if } u_0 = ❷ \text{ and } m = 1$$

Example 1: Single machine

Single machine

$\sigma_{12} = 3, \sigma_{21} = 1$



Continuous dynamics

$$\dot{x}_0(t) = \begin{cases} -1 & \text{if } u_0 \in \{❶, ❷\} \\ 0 & \text{if } u_0 \in \{①, ②\} \end{cases}$$

$$\dot{x}_1(t) = \lambda_1 - u_1(t)$$

$$\dot{x}_2(t) = \lambda_2 - u_2(t)$$

Discrete event dynamics

$$x_0 := \sigma_{21} \qquad m := 1 \qquad \text{if } u_0 = ❶ \text{ and } m = 2$$

$$x_0 := \sigma_{12} \qquad m := 2 \qquad \text{if } u_0 = ❷ \text{ and } m = 1$$

Example 1: Single machine

Input contraints

$$
\begin{array}{llll}
u_0 \in \{❶, ❷\} & u_1 = 0 & u_2 = 0 & \text{if } x_0 > 0 \\
u_0 \in \{①, ❷\} & u_1 \leq \mu_1 & u_2 = 0 & \text{if } x_0 = 0,\ x_1 > 0,\ m = 1 \\
u_0 \in \{①, ❷\} & u_1 \leq \lambda_1 & u_2 = 0 & \text{if } x_0 = 0,\ x_1 = 0,\ m = 1 \\
u_0 \in \{❶, ②\} & u_1 = 0 & u_2 \leq \mu_2 & \text{if } x_0 = 0,\ x_2 > 0,\ m = 2 \\
u_0 \in \{❶, ②\} & u_1 = 0 & u_2 \leq \lambda_2 & \text{if } x_0 = 0,\ x_2 = 0,\ m = 2
\end{array}
$$

Objective

Minimize:

$$\limsup_{t\to\infty} \frac{1}{t}\int_0^t x_1(\tau) + x_2(\tau)\,\mathrm{d}\,\tau \qquad \text{or} \qquad \frac{1}{T}\int_0^T x_1(\tau) + x_2(\tau)\,\mathrm{d}\,\tau$$

Example 1: Single machine

Input contraints

$$
\begin{array}{llll}
u_0 \in \{❶, ❷\} & u_1 = 0 & u_2 = 0 & \text{if } x_0 > 0 \\
u_0 \in \{①, ❷\} & u_1 \leq \mu_1 & u_2 = 0 & \text{if } x_0 = 0,\ x_1 > 0,\ m = 1 \\
u_0 \in \{①, ❷\} & u_1 \leq \lambda_1 & u_2 = 0 & \text{if } x_0 = 0,\ x_1 = 0,\ m = 1 \\
u_0 \in \{❶, ②\} & u_1 = 0 & u_2 \leq \mu_2 & \text{if } x_0 = 0,\ x_2 > 0,\ m = 2 \\
u_0 \in \{❶, ②\} & u_1 = 0 & u_2 \leq \lambda_2 & \text{if } x_0 = 0,\ x_2 = 0,\ m = 2
\end{array}
$$

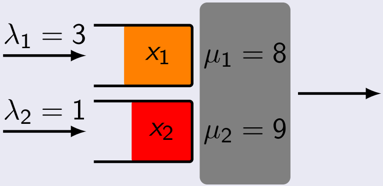
Objective

Minimize:

$$
\limsup_{t\to\infty} \frac{1}{t} \int_0^t x_1(\tau) + x_2(\tau)\,\mathrm{d}\,\tau \qquad \text{or} \qquad \frac{1}{T} \int_0^T x_1(\tau) + x_2(\tau)\,\mathrm{d}\,\tau
$$

Desired behavior

Single machine



Desired behavior

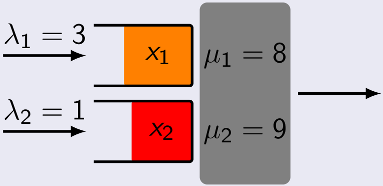
Remarks

- Many existing policies assume non-idling a-priori
- Slow-mode optimal if $(\frac{\lambda_1}{\mu_1} + \frac{\lambda_2}{\mu_2}) + (\lambda_2 - \lambda_1)(1 - \frac{\lambda_2}{\mu_2}) < 0.$
- Trade-off in wasting capacity: idle $\Leftrightarrow$ switch more often

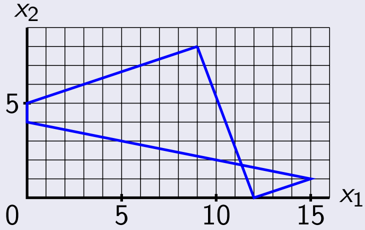
TU/e NWO

Desired behavior

Single machine



Desired behavior

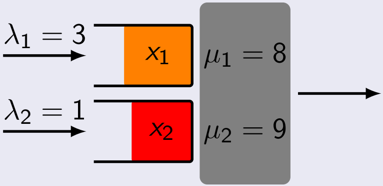


Remarks

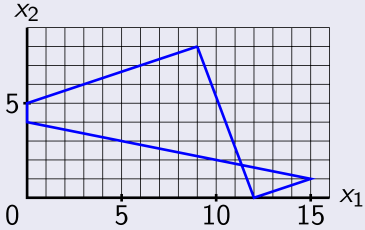
- Many existing policies assume non-idling a-priori
- Slow-mode optimal if $(\frac{\lambda_1}{\mu_1} + \frac{\lambda_2}{\mu_2}) + (\lambda_2 - \lambda_1)(1 - \frac{\lambda_2}{\mu_2}) < 0$.
- Trade-off in wasting capacity: idle $\Leftrightarrow$ switch more often

Desired behavior

Single machine



Desired behavior



Remarks

- Many existing policies assume non-idling a-priori
- Slow-mode optimal if $(\frac{\lambda_1}{\mu_1} + \frac{\lambda_2}{\mu_2}) + (\lambda_2 - \lambda_1)(1 - \frac{\lambda_2}{\mu_2}) < 0$.
- Trade-off in wasting capacity: idle $\Leftrightarrow$ switch more often

Controller design

Main idea

Lyapunov: if energy is decreasing all the time ⇒ system settles down at constant energy level

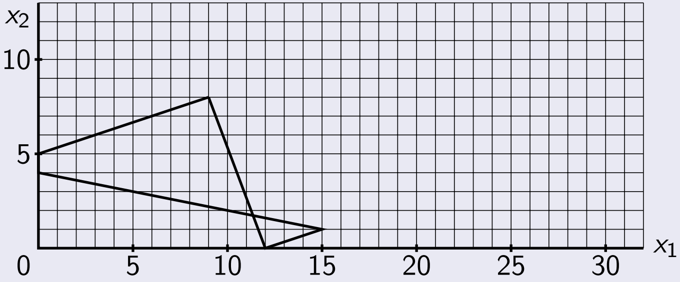
Lyapunov function candidate

Controller design

Main idea

Lyapunov: if energy is decreasing all the time ⇒ system settles down at constant energy level

Lyapunov function candidate

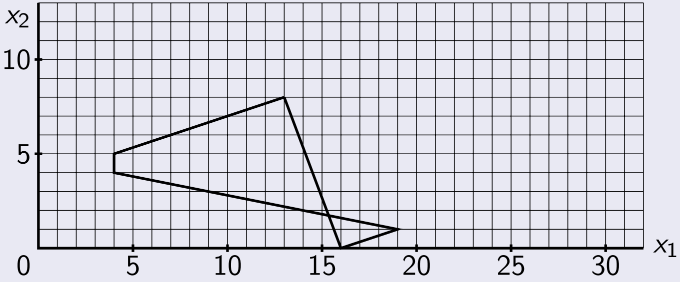


Controller design

Main idea

Lyapunov: if energy is decreasing all the time ⇒ system settles down at constant energy level

Lyapunov function candidate

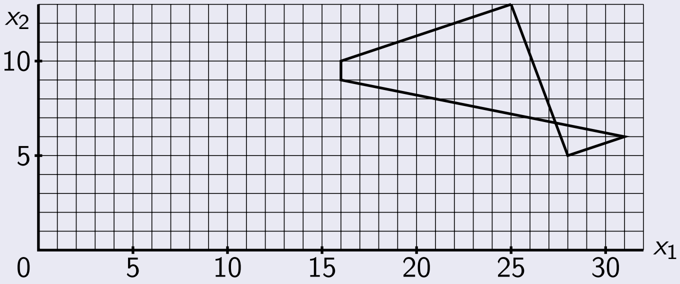


Controller design

Main idea

Lyapunov: if energy is decreasing all the time ⇒ system settles down at constant energy level

Lyapunov function candidate

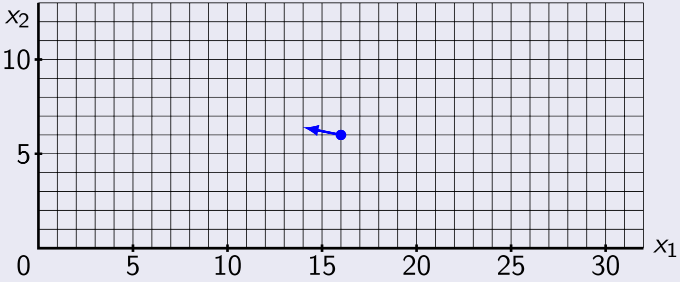


Controller design

Main idea

Lyapunov: if energy is decreasing all the time $\Rightarrow$ system settles down at constant energy level

Lyapunov function candidate

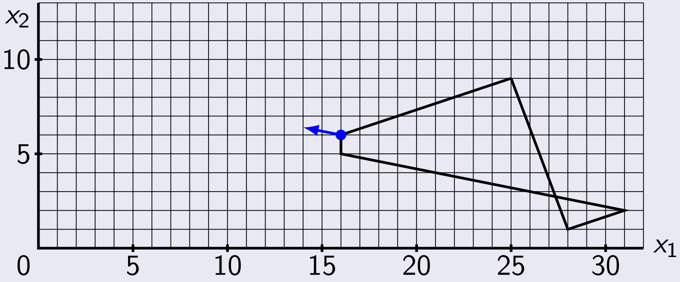


Controller design

Main idea

Lyapunov: if energy is decreasing all the time ⇒ system settles down at constant energy level

Lyapunov function candidate

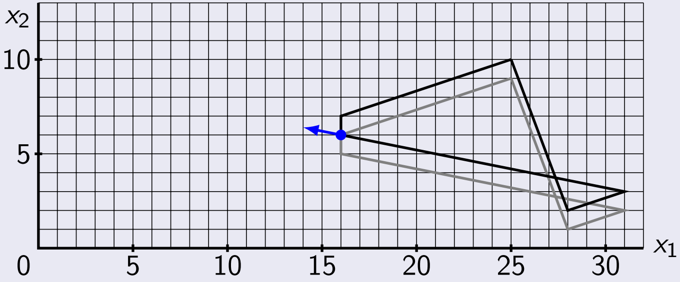


Controller design

Main idea

Lyapunov: if energy is decreasing all the time ⇒ system settles down at constant energy level

Lyapunov function candidate

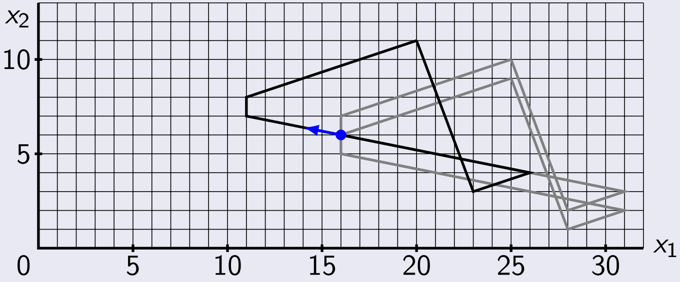


Controller design

Main idea

Lyapunov: if energy is decreasing all the time ⇒ system settles down at constant energy level

Lyapunov function candidate

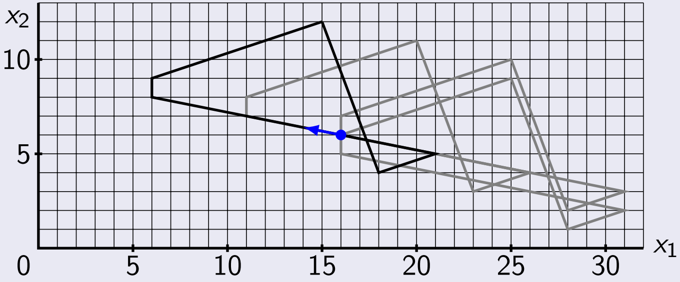


Controller design

Main idea

Lyapunov: if energy is decreasing all the time $\Rightarrow$ system settles down at constant energy level

Lyapunov function candidate

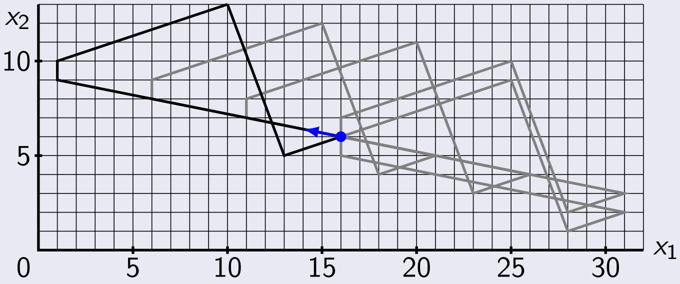


Controller design

Main idea

Lyapunov: if energy is decreasing all the time ⇒ system settles down at constant energy level

Lyapunov function candidate

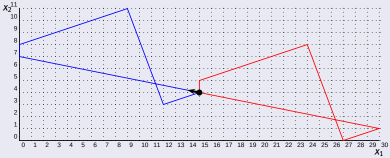


Controller design

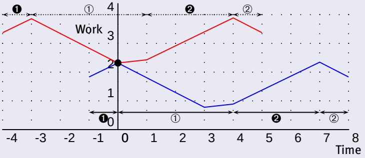
Lyapunov function candidate

The smallest additional mean amount of work from all feasible curves for state (work:$x_1/\mu_1 + x_2/\mu_2$).

Phase plane



Time evolution work



Controller design

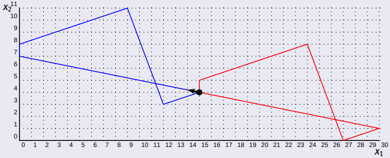
Let Lyapunov function candidate decrease as quickly as possible

Controller design

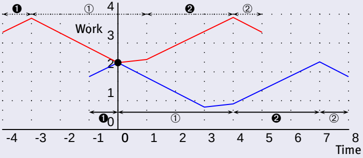
Lyapunov function candidate

The smallest additional mean amount of work from all feasible curves for state (work:$x_1/\mu_1 + x_2/\mu_2$).

Phase plane



Time evolution work

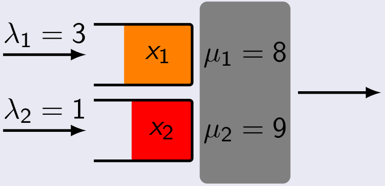


Controller design

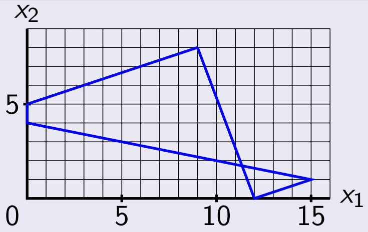
Let Lyapunov function candidate decrease as quickly as possible

Controller design (Result)

Single machine



Desired behavior

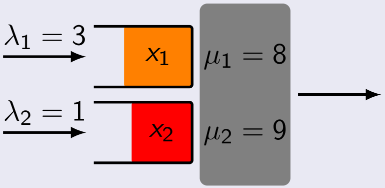


Resulting Controller, cf. [Lefeber, Rooda (2006)]

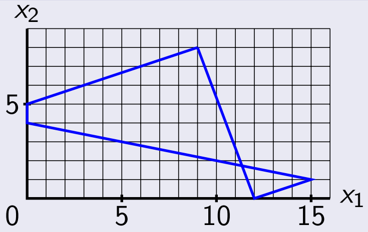
- When serving type 1:
  1. empty buffer
  2. serve until $x_2 \geq 5$
  3. switch to type 2
- When serving type 2:
  1. empty buffer
  2. serve until $x_1 \geq 12$
  3. switch to type 1

Controller design (Result)

Single machine



Desired behavior

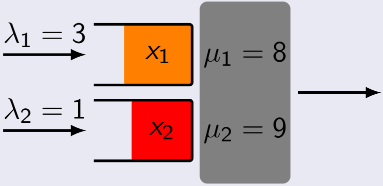


Resulting Controller, cf. [Lefeber, Rooda (2006)]

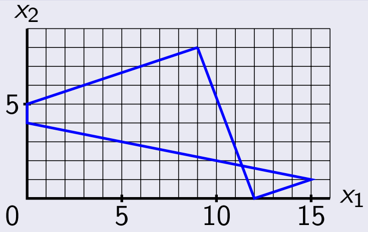
- When serving type 1:
  1. empty buffer
  2. serve until $x_2 \geq 5$
  3. switch to type 2
- When serving type 2:
  1. empty buffer
  2. serve until $x_1 \geq 12$
  3. switch to type 1

Controller design (Result)

Single machine



Desired behavior



Resulting Controller, cf. [Lefeber, Rooda (2006)]

- When serving type 1:
  1. empty buffer
  2. serve until $x_2 \geq 5$
  3. switch to type 2
- When serving type 2:
  1. empty buffer
  2. serve until $x_1 \geq 12$
  3. switch to type 1

Recap

Notions from control theory

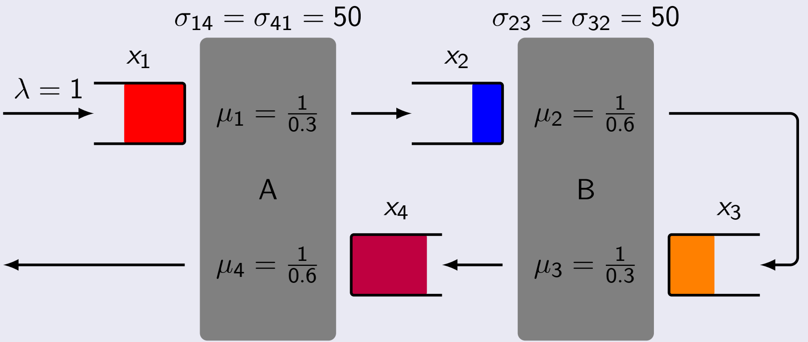
1. Generate feasible reference trajectory
2. Design (static) state feedback controller
3. Design observer
4. Design (dynamic) output feedback controller

Parallels with this problem

1. Determine desired system behavior
2. Derive non-distributed/centralized controller
3. Can state be reconstructed?
4. Derive distributed/decentralized controller

Example 2: Kumar-Seidman case

Transactions on Automatic Control, Vol 35, No 3, March 1990

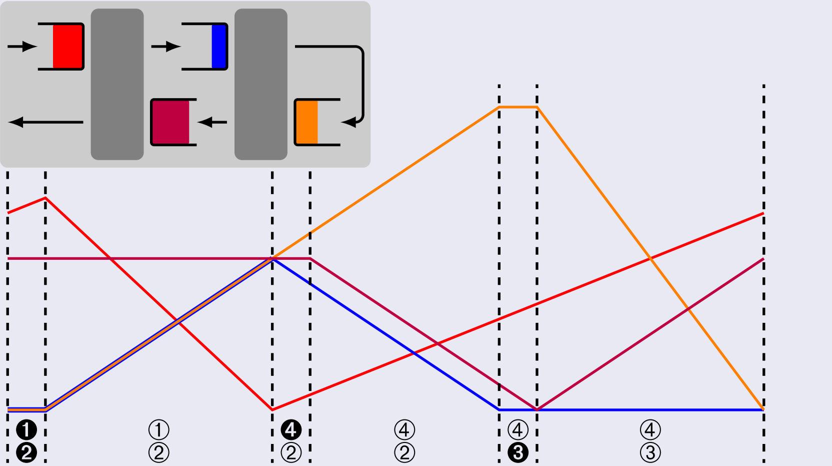


Observation

Sufficient capacity (consider period of at least 1000).

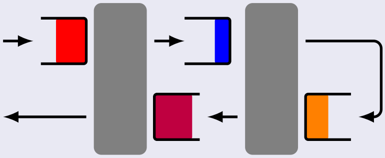
Desired behavior

Desired behavior

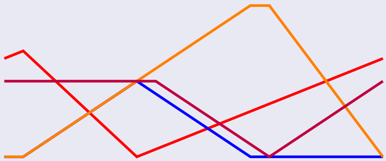


Resulting controller

Network



Desired behavior



Resulting controller

Mode (1,2): to (4,2) when both $x_1 = 0$ and $x_2 + x_3 \geq 1000$

Mode (4,2): to (4,3) when both $x_2 = 0$ and $x_4 \leq 83\frac{1}{3}$

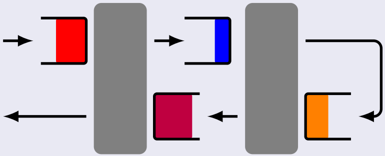
Mode (4,3): to (1,2) when $x_3 = 0$

Remark:

- Non-distributed/centralized controller

Observability

Network



Assumptions

- Clearing policy used for machine B
- At $t = t_1$: ③ starts
- At $t = t_2 > t_1$: ③ stops

System state can be reconstructed at machine A

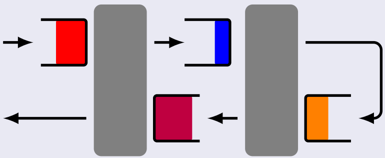
- $x_3(t_2) = 0$, and $x_3(t_1 - 50) = x_3(t_1) = (t_2 - t_1)/0.6$
- $x_2(t_1 - 50) = 0$, and $x_2(t_2) = \int_{t_1-50}^{t_2} u_1(\tau)\,\mathrm{d}\,\tau$

Observation

Observablity determined by network topology

Observability

Network



Assumptions

- Clearing policy used for machine B
- At $t = t_1$: ③ starts
- At $t = t_2 > t_1$: ③ stops

System state can be reconstructed at machine A

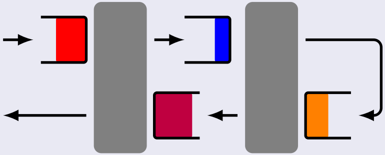
- $x_3(t_2) = 0$, and $x_3(t_1 - 50) = x_3(t_1) = (t_2 - t_1)/0.6$
- $x_2(t_1 - 50) = 0$, and $x_2(t_2) = \int_{t_1-50}^{t_2} u_1(\tau)\,\mathrm{d}\,\tau$

Observation

Observablity determined by network topology

Observability

Network



Assumptions

- Clearing policy used for machine B
- At $t = t_1$: ③ starts
- At $t = t_2 > t_1$: ③ stops

System state can be reconstructed at machine A

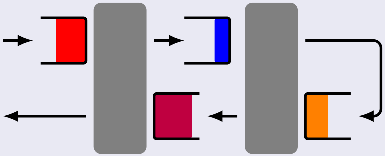
- $x_3(t_2) = 0$, and $x_3(t_1 - 50) = x_3(t_1) = (t_2 - t_1)/0.6$
- $x_2(t_1 - 50) = 0$, and $x_2(t_2) = \int_{t_1-50}^{t_2} u_1(\tau)\,\mathrm{d}\,\tau$

Observation

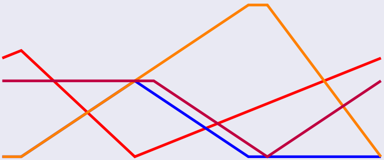
Observablity determined by network topology

Distributed controller, cf. [Lefeber, Rooda (2008)]

Network



Desired behavior



Distributed controller

Serving 1: Serve at least 1000 jobs until $x_1 = 0$, then switch. Let $\bar{x}_1$ be nr of jobs served.

Serving 2: Serve at least 1000 jobs until $x_2 = 0$, then switch.

Serving 4: Let $\bar{x}_4$ be nr of jobs in Buffer 4 after setup. Serve $\bar{x}_4 + \frac{1}{2}\bar{x}_1$ jobs, then switch.

Serving 3: Empty buffer, then switch.

Conclusions

New approach

1. Determine desired system behavior (trajectory generation)
2. Derive non-distributed/centralized controller (state feedback)
3. Derive distributed/decentralized controller (output feedback)

Advantage

All three problems can be considered separately

Centralized control

Approach can deal with

- Arbitrary networks
- Finite buffers
- Transportation delays

Decentralized control

- Observer based approach results in new, tailor-made controllers that perform better

Conclusions

New approach

1. Determine desired system behavior (trajectory generation)
2. Derive non-distributed/centralized controller (state feedback)
3. Derive distributed/decentralized controller (output feedback)

Advantage

All three problems can be considered separately

Centralized control

Approach can deal with

- Arbitrary networks
- Finite buffers
- Transportation delays

Decentralized control

- Observer based approach results in new, tailor-made controllers that perform better

Conclusions

New approach

1. Determine desired system behavior (trajectory generation)
2. Derive non-distributed/centralized controller (state feedback)
3. Derive distributed/decentralized controller (output feedback)

Advantage

All three problems can be considered separately

Centralized control

Approach can deal with

- Arbitrary networks
- Finite buffers
- Transportation delays

Decentralized control

- Observer based approach results in new, tailor-made controllers that perform better

Conclusions

New approach

1. Determine desired system behavior (trajectory generation)
2. Derive non-distributed/centralized controller (state feedback)
3. Derive distributed/decentralized controller (output feedback)

Advantage

All three problems can be considered separately

Centralized control

Approach can deal with

- Arbitrary networks
- Finite buffers
- Transportation delays

Decentralized control

- Observer based approach results in new, tailor-made controllers that perform better

TU/e NWO

Future work

Research

- Centralized control
  - Modify existing approach to overcome some shortcomings
  - Derive class of controllers (instead of only one)
  - Finite buffers: reachability of desired orbit
  - Deal with parametric uncertainty; robustness if parameters are either different or time-varying.
- Decentralized control
  - Observability (including tests)
  - Observer design
  - Stability analysis of distributed policies
- Stochastic extensions
  - Analyze performance of derived (de)centralized controllers for stochastic queueing networks

Future work

Research

- Centralized control
  - Modify existing approach to overcome some shortcomings
  - Derive class of controllers (instead of only one)
  - Finite buffers: reachability of desired orbit
  - Deal with parametric uncertainty; robustness if parameters are either different or time-varying.
- Decentralized control
  - Observability (including tests)
  - Observer design
  - Stability analysis of distributed policies
- Stochastic extensions
  - Analyze performance of derived (de)centralized controllers for stochastic queueing networks

Future work

Research

- Centralized control
  - Modify existing approach to overcome some shortcomings
  - Derive class of controllers (instead of only one)
  - Finite buffers: reachability of desired orbit
  - Deal with parametric uncertainty; robustness if parameters are either different or time-varying.
- Decentralized control
  - Observability (including tests)
  - Observer design
  - Stability analysis of distributed policies
- Stochastic extensions
  - Analyze performance of derived (de)centralized controllers for stochastic queueing networks