Dispatching Problem with Fixed Size Jobs and Processor Sharing Discipline

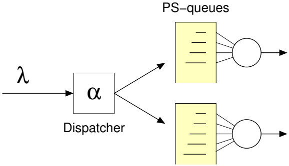
E. Hyytiä, A. Penttinen, S. Aalto and J. Virtamo

Department of Communications and Networking
Aalto University, School of Electrical Engineering, Finland

7.9.2011

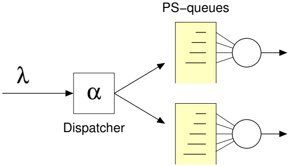
Aalto University
School of Electrical Engineering

Dispatching problem to parallel queues



- Upon arrival a job is routed to one of the *m* servers

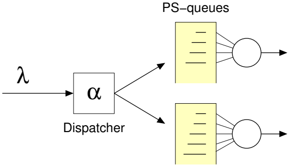
Dispatching problem to parallel queues



- Upon arrival a job is routed to one of the *m* servers
- Each server processes jobs according to a certain scheduling discipline (e.g., PS)

Aalto University
School of Electrical Engineering

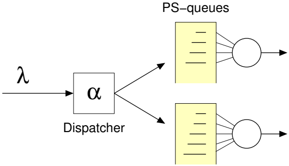
Dispatching problem to parallel queues



- Upon arrival a job is routed to one of the *m* servers
- Each server processes jobs according to a certain scheduling discipline (e.g., PS)
- Objective: minimize the mean delay (mean sojourn time)

Aalto University
School of Electrical Engineering

Dispatching problem to parallel queues



- Upon arrival a job is routed to one of the *m* servers
- Each server processes jobs according to a certain scheduling discipline (e.g., PS)
- Objective: minimize the mean delay (mean sojourn time)
- Examples:
  - job assignment in supercomputing
  - traffic routing
  - web-server farms, and
  - other distributed computing systems

Aalto University
School of Electrical Engineering

Heuristic policies

State-independent Policies:

1. **Bernoulli splitting (RND)**:
   Choose queue in random using probabilities p_i:
   i) RND-U splits the arrival stream uniformly, $p_i = 1/m$
   ii) RND-ρ balances the load, $p_i = c_i / \sum_j c_j$
   iii) RND-opt uses the p_i that minimize the mean sojourn time

Aalto University
School of Electrical Engineering

Heuristic policies

State-independent Policies:

1. **Bernoulli splitting (RND)**:
   Choose queue in random using probabilities p_i:
   i) RND-U splits the arrival stream uniformly, $p_i = 1/m$
   ii) RND-ρ balances the load, $p_i = c_i / \sum_j c_j$
   iii) RND-opt uses the p_i that minimize the mean sojourn time

State-dependent Policies:

1. **Join-the-Shortest-Queue (JSQ)**:
   Optimal when Poisson arrivals, Exponential jobs, identical servers, and only the occupancy is known (Winston, 1977).

Aalto University
School of Electrical Engineering

Heuristic policies

State-independent Policies:

1. **Bernoulli splitting (RND)**:
   Choose queue in random using probabilities p_i:
   i) RND-U splits the arrival stream uniformly, $p_i = 1/m$
   ii) RND-ρ balances the load, $p_i = c_i / \sum_j c_j$
   iii) RND-opt uses the p_i that minimize the mean sojourn time

State-dependent Policies:

1. **Join-the-Shortest-Queue (JSQ)**:
   Optimal when Poisson arrivals, Exponential jobs, identical servers, and only the occupancy is known (Winston, 1977).
2. **Round-robin (RR)**:
   Optimal with identical servers that were initially in a same state (Ephremides et. al, 1980).

Aalto University
School of Electrical Engineering

Heuristic policies

State-independent Policies:

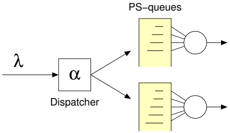
1. **Bernoulli splitting (RND)**:
   Choose queue in random using probabilities p_i:
   i) RND-U splits the arrival stream uniformly, $p_i = 1/m$
   ii) RND-ρ balances the load, $p_i = c_i / \sum_j c_j$
   iii) RND-opt uses the p_i that minimize the mean sojourn time

State-dependent Policies:

1. **Join-the-Shortest-Queue (JSQ)**:
   Optimal when Poisson arrivals, Exponential jobs, identical servers, and only the occupancy is known (Winston, 1977).
2. **Round-robin (RR)**:
   Optimal with identical servers that were initially in a same state (Ephremides et. al, 1980).
3. **Least-Work-Left (LWL)**:
   Pick the queue with the shortest backlog (Sharifnia, 1997).

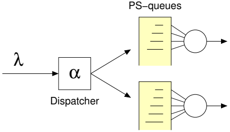
Aalto University
School of Electrical Engineering

State-aware dispatching with constant job size



- Poisson arrival process, rate λ

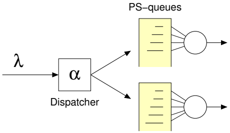
State-aware dispatching with constant job size



- Poisson arrival process, rate λ
- Fixed job size d

Aalto University
School of Electrical Engineering

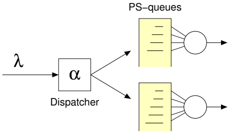
State-aware dispatching with constant job size



- Poisson arrival process, rate λ
- Fixed job size d
- m parallel heterogeneous servers:

Aalto University
School of Electrical Engineering

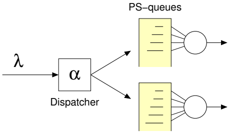
State-aware dispatching with constant job size



- Poisson arrival process, rate λ
- Fixed job size d
- m parallel heterogeneous servers:
  - Server specific processing rates c_i

Aalto University
School of Electrical Engineering

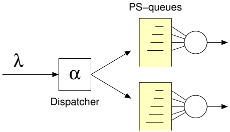
State-aware dispatching with constant job size



- Poisson arrival process, rate λ
- Fixed job size d
- m parallel heterogeneous servers:
  - Server specific processing rates c_i
  - Processor Sharing (PS) scheduling discipline

Aalto University
School of Electrical Engineering

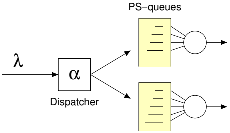
State-aware dispatching with constant job size



- Poisson arrival process, rate λ
- Fixed job size d
- m parallel heterogeneous servers:
  - Server specific processing rates c_i
  - Processor Sharing (PS) scheduling discipline
- Queue states (remaining service times) are known to the dispatcher

Aalto University
School of Electrical Engineering

State-aware dispatching with constant job size



- Poisson arrival process, rate λ
- Fixed job size d
- m parallel heterogeneous servers:
  - Server specific processing rates c_i
  - Processor Sharing (PS) scheduling discipline
- Queue states (remaining service times) are known to the dispatcher
- Objective: minimize the mean delay

Aalto University
School of Electrical Engineering

Delay costs and relative value

Delay costs are accrued at rate

$$N_{\mathbf{z}}(t) \triangleq \text{"the number of jobs in the system"},$$

where $\mathbf{z}$ denotes the initial state at time $t = 0$.

Delay costs and relative value

Delay costs are accrued at rate

$$N_{\mathbf{z}}(t) \triangleq \text{"the number of jobs in the system"},$$

where $\mathbf{z}$ denotes the initial state at time $t = 0$.

Delay costs accrued during $(0, t)$: $\quad V_{\mathbf{z}}(t) \triangleq \int_0^t N_{\mathbf{z}}(s)\, ds.$

Aalto University
School of Electrical Engineering

Delay costs and relative value

Delay costs are accrued at rate

$$N_{\mathbf{z}}(t) \triangleq \text{"the number of jobs in the system"},$$

where $\mathbf{z}$ denotes the initial state at time $t = 0$.

Delay costs accrued during $(0, t)$: $\quad V_{\mathbf{z}}(t) \triangleq \int_0^t N_{\mathbf{z}}(s)\, ds.$

Relative value: the expected difference in the cumulative costs between a system initially in state $\mathbf{z}$ and a system in equilibrium,

$$\begin{aligned} v_{\mathbf{z}} &\triangleq \lim_{t\to\infty} \mathrm{E}[V_{\mathbf{z}}(t) - r\, t] \\ &= \lim_{t\to\infty} \left(\mathrm{E}\Big[\int_0^t N_{\mathbf{z}}(s)\, ds\Big] - \mathrm{E}[N]\, t \right). \end{aligned}$$

Aalto University
School of Electrical Engineering

Approach: MDP and first policy iteration (FPI)

- Size- and state-aware setting; future arrivals not known

Aalto University
School of Electrical Engineering

Approach: MDP and first policy iteration (FPI)

- Size- and state-aware setting; future arrivals not known
- Idea: start with a reasonable basic dispatching policy, and carry out the first policy iteration (FPI) step

Aalto University
School of Electrical Engineering

Approach: MDP and first policy iteration (FPI)

- Size- and state-aware setting; future arrivals not known
- Idea: start with a reasonable basic dispatching policy, and carry out the first policy iteration (FPI) step
- Policy iteration finds the optimal policy, and the FPI step typically yields the highest improvement.

Aalto University
School of Electrical Engineering

Approach: MDP and first policy iteration (FPI)

- Size- and state-aware setting; future arrivals not known
- Idea: start with a reasonable basic dispatching policy, and carry out the first policy iteration (FPI) step
- Policy iteration finds the optimal policy, and the FPI step typically yields the highest improvement.
- Requires the relative values of states $v_{\mathbf{z}}$

Aalto University
School of Electrical Engineering

Approach: MDP and first policy iteration (FPI)

- Size- and state-aware setting; future arrivals not known
- Idea: start with a reasonable basic dispatching policy, and carry out the first policy iteration (FPI) step
- Policy iteration finds the optimal policy, and the FPI step typically yields the highest improvement.
- Requires the relative values of states $v_{\mathbf{z}}$
- However, our state-space is extremely complex (remaining service requirements at each queue)

Aalto University
School of Electrical Engineering

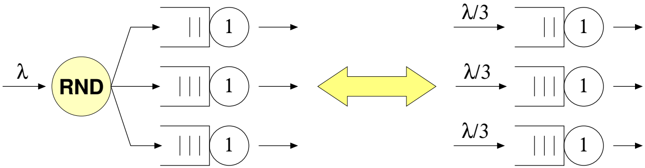
Decomposition to independent M/D/1-PS queues

- Deriving a relative value is generally a difficult task.

Aalto University
School of Electrical Engineering

Decomposition to independent M/D/1-PS queues

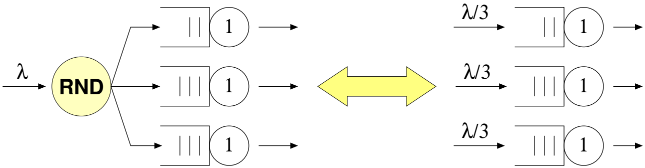
- Deriving a relative value is generally a difficult task.
- However, any state-independent policy feeds each server jobs according to a Poisson process (cf. Bernoulli split)



Aalto University
School of Electrical Engineering

Decomposition to independent M/D/1-PS queues

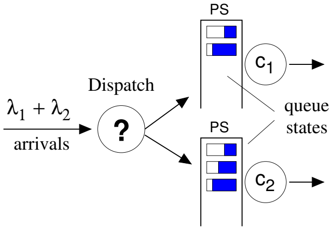
- Deriving a relative value is generally a difficult task.
- However, any state-independent policy feeds each server jobs according to a Poisson process (cf. Bernoulli split)



Analyze single M/D/1-PS queues instead?

Aalto University
School of Electrical Engineering

FPI of state-independent basic policy



Aalto University
School of Electrical Engineering

FPI of state-independent basic policy

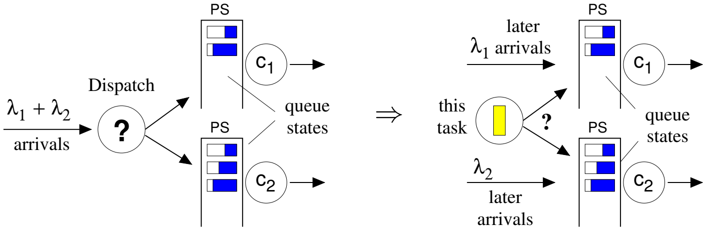


Figure: FPI considers a single decision, after which one falls back to the basic policy.

Aalto University
School of Electrical Engineering

FPI of state-independent basic policy

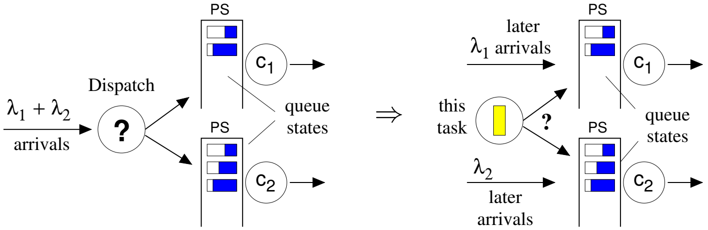


Figure: FPI considers a single decision, after which one falls back to the basic policy.

Can we solve the latter exactly?

Aalto University
School of Electrical Engineering

FPI of state-independent basic policy

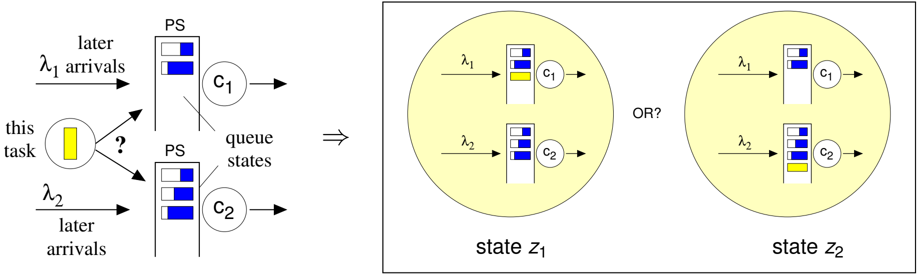


Figure: FPI of state-independent basic policy: later arrivals are dispatched according to the basic policy, isolating the queues.

Aalto University
School of Electrical
Engineering

FPI of state-independent basic policy

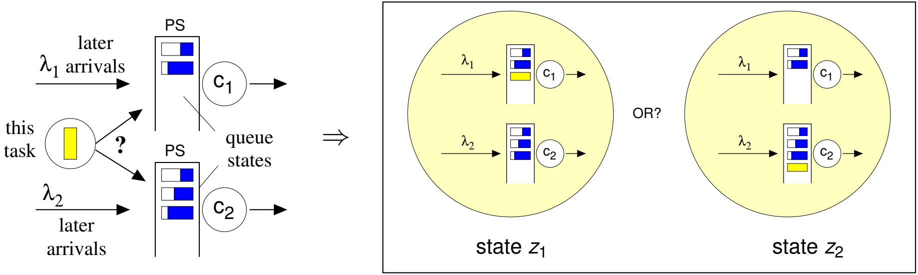


Figure: FPI of state-independent basic policy: later arrivals are dispatched according to the basic policy, isolating the queues.

The relative values v_{z_1} and v_{z_2} tell us which is the better option!

Aalto University
School of Electrical Engineering

FPI of state-independent basic policy

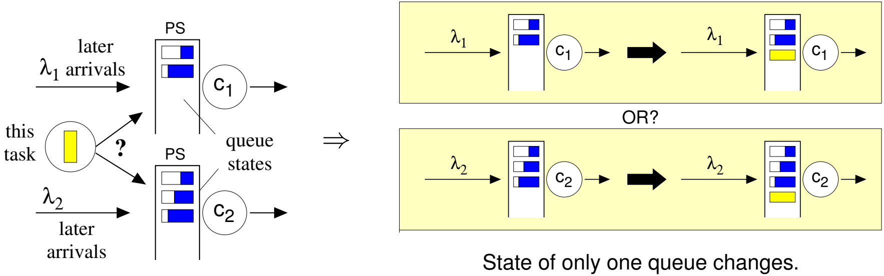


Figure: Comparison between two states in each queue.

FPI of state-independent basic policy

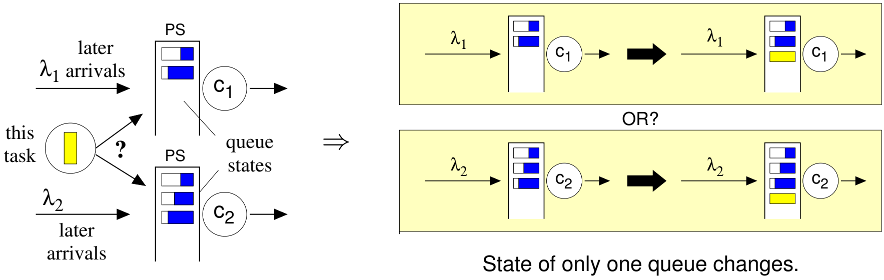


Figure: Comparison between two states in each queue.

Increments in the **queue specific relative values** $v_z^{(1)}$ and $v_z^{(2)}$ tell us which queue to choose!

Aalto University
School of Electrical Engineering

Roadmap

1. Assume a state-independent basic policy.

Aalto University
School of Electrical Engineering

Roadmap

1. Assume a state-independent basic policy.
2. Derive relative values for an “isolated queue”.

Aalto University
School of Electrical Engineering

Roadmap

1. Assume a state-independent basic policy.
2. Derive relative values for an "isolated queue".
3. Relative value of the whole system is the sum of the queue specific relative values:

$$v_{\mathbf{z}} = \sum_{i} v_{\mathbf{z}_i}.$$

Aalto University
School of Electrical Engineering

Roadmap

1. Assume a state-independent basic policy.
2. Derive relative values for an "isolated queue".
3. Relative value of the whole system is the sum of the queue specific relative values:

$$v_{\mathbf{z}} = \sum_i v_{\mathbf{z}_i}.$$

4. Carry out FPI $\Rightarrow$ new efficient dispatching policy.

Roadmap

1. Assume a state-independent basic policy.
2. Derive relative values for an "isolated queue".
3. Relative value of the whole system is the sum of the queue specific relative values:

$$v_{\mathbf{z}} = \sum_i v_{\mathbf{z}_i}.$$

4. Carry out FPI $\Rightarrow$ new efficient dispatching policy.

In practice, it is sufficient to know, e.g., $v_{\mathbf{z}} - v_0$.

Next step: Derive $v_{\mathbf{z}} - v_0$ for an M/D/1-PS queue.

Aalto University
School of Electrical Engineering

Relative value for an M/D/1-PS queue

Notation:

- λ is the Poisson arrival rate.

Aalto University
School of Electrical Engineering

Relative value for an M/D/1-PS queue

Notation:

- λ is the Poisson arrival rate.
- $\rho = \lambda\, d$ and d denotes the fixed job size.

Aalto University
School of Electrical Engineering

Relative value for an M/D/1-PS queue

Notation:

- λ is the Poisson arrival rate.
- $\rho = \lambda d$ and d denotes the fixed job size.
- $\mathbf{z} = (\Delta_1; ..; \Delta_n)$ are the remaining service times, $\Delta_i > \Delta_{i+1}$

Aalto University
School of Electrical Engineering

Relative value for an M/D/1-PS queue

Notation:

- λ is the Poisson arrival rate.
- $\rho = \lambda\, d$ and d denotes the fixed job size.
- $\mathbf{z} = (\Delta_1; ..; \Delta_n)$ are the remaining service times, $\Delta_i > \Delta_{i+1}$

Proposition: The size-aware relative value of state **z** with respect to the delay in an M/D/1-PS queue is given by

$$v_{(\Delta_1;..;\Delta_n)} - v_0 = \boxed{\frac{\lambda}{1-\rho} u_{\mathbf{z}}^2 - u_{\mathbf{z}} + 2\sum_{i=1}^{n} i\,\Delta_i.} \qquad (1)$$

where v_0 denotes the relative value of an empty system, and $u_{\mathbf{z}} = \sum_i \Delta_i$ the backlog in the queue.

Aalto University
School of Electrical Engineering

Proof sketched

- Consider two systems under the same arrivals:
  - S1 initially in state $\mathbf{z} = (\Delta_1; ..; \Delta_n)$ with $\Delta_1 \geq \ldots \geq \Delta_n$.
  - S2 initially empty.

Aalto University
School of Electrical Engineering

Proof sketched

- Consider two systems under the same arrivals:
  - S1 initially in state $\mathbf{z} = (\Delta_1; ..; \Delta_n)$ with $\Delta_1 \geq \ldots \geq \Delta_n$.
  - S2 initially empty.
- Once S1 is empty, the two systems behave equivalently

Aalto University
School of Electrical Engineering

Proof sketched

- Consider two systems under the same arrivals:
  - S1 initially in state $\mathbf{z} = (\Delta_1; ..; \Delta_n)$ with $\Delta_1 \geq \ldots \geq \Delta_n$.
  - S2 initially empty.
- Once S1 is empty, the two systems behave equivalently
- Without new arrivals, the total delay accrued in S1 is

$$\begin{aligned} \tau_{\mathbf{z}} &= \Delta_n \cdot n^2 + (\Delta_{n-1} - \Delta_n) \cdot (n-1)^2 + \ldots + (\Delta_1 - \Delta_2), \\ &= \sum_{i=1}^{n} (2i-1)\Delta_i. \end{aligned} \quad (2)$$

Aalto University
School of Electrical Engineering

Proof sketched

- Consider two systems under the same arrivals:
  - S1 initially in state $\mathbf{z} = (\Delta_1; ..; \Delta_n)$ with $\Delta_1 \geq \ldots \geq \Delta_n$.
  - S2 initially empty.
- Once S1 is empty, the two systems behave equivalently
- Without new arrivals, the total delay accrued in S1 is

$$\tau_{\mathbf{z}} = \Delta_n \cdot n^2 + (\Delta_{n-1} - \Delta_n) \cdot (n-1)^2 + \ldots + (\Delta_1 - \Delta_2), = \sum_{i=1}^{n} (2i-1)\Delta_i. \quad (2)$$

- Each arrival increases the total delay (*immediate cost*)

$$s_{\mathbf{z}} = \tau_{(d;\Delta_1;..;\Delta_n)} - \tau_{(\Delta_1;..;\Delta_n)} = \boxed{2\,u_{\mathbf{z}} + d.} \quad (3)$$

Aalto University
School of Electrical Engineering

Proof sketched

- Consider two systems under the same arrivals:
  - S1 initially in state $\mathbf{z} = (\Delta_1; ..; \Delta_n)$ with $\Delta_1 \geq \ldots \geq \Delta_n$.
  - S2 initially empty.
- Once S1 is empty, the two systems behave equivalently
- Without new arrivals, the total delay accrued in S1 is

$$\tau_{\mathbf{z}} = \Delta_n \cdot n^2 + (\Delta_{n-1} - \Delta_n) \cdot (n-1)^2 + \ldots + (\Delta_1 - \Delta_2), \\ = \sum_{i=1}^{n} (2i-1)\Delta_i. \qquad (2)$$

- Each arrival increases the total delay (*immediate cost*)

$$s_{\mathbf{z}} = \tau_{(d;\Delta_1;..;\Delta_n)} - \tau_{(\Delta_1;..;\Delta_n)} = \boxed{2\,u_{\mathbf{z}} + d.} \qquad (3)$$

- Utilize the lack of memory of Poisson arrivals.
- Virtual busy periods similar (S1 has an offset in backlog) $\Rightarrow$ the mean contribution of a busy period.

Aalto University
School of Electrical Engineering

Proof sketched

- Consider two systems under the same arrivals:
  - S1 initially in state $\mathbf{z} = (\Delta_1; ..; \Delta_n)$ with $\Delta_1 \geq \ldots \geq \Delta_n$.
  - S2 initially empty.
- Once S1 is empty, the two systems behave equivalently
- Without new arrivals, the total delay accrued in S1 is

$$\tau_{\mathbf{z}} = \Delta_n \cdot n^2 + (\Delta_{n-1} - \Delta_n) \cdot (n-1)^2 + \ldots + (\Delta_1 - \Delta_2),$$
$$= \sum_{i=1}^{n} (2i-1)\Delta_i. \quad (2)$$

- Each arrival increases the total delay (*immediate cost*)

$$s_{\mathbf{z}} = \tau_{(d;\Delta_1;..;\Delta_n)} - \tau_{(\Delta_1;..;\Delta_n)} = \boxed{2\,u_{\mathbf{z}} + d.} \quad (3)$$

- Utilize the lack of memory of Poisson arrivals.
- Virtual busy periods similar (S1 has an offset in backlog)
  $\Rightarrow$ the mean contribution of a busy period.
- Details in the paper.

Aalto University
School of Electrical Engineering

Cost of a new task in M/D/1-PS

Corollary: The expected cost due to accepting a new task to an M/D/1-PS queue at state $\mathbf{z} = (\Delta_1; ..; \Delta_n)$ is given by

$$w_{\mathbf{z}} = v_{(d;\Delta_1;..;\Delta_n)} - v_{(\Delta_1;..;\Delta_n)} = \boxed{\frac{2\,u_{\mathbf{z}} + d}{1-\rho}.} \tag{4}$$

Aalto University
School of Electrical Engineering

Cost of a new task in M/D/1-PS

Corollary: The expected cost due to accepting a new task to an M/D/1-PS queue at state $\mathbf{z} = (\Delta_1; ..; \Delta_n)$ is given by

$$w_{\mathbf{z}} = v_{(d;\Delta_1;..;\Delta_n)} - v_{(\Delta_1;..;\Delta_n)} = \boxed{\frac{2\,u_{\mathbf{z}} + d}{1-\rho}.} \qquad (4)$$

That is, the immediate cost divided by $1 - \rho$.

Aalto University
School of Electrical Engineering

Cost of a new task in M/D/1-PS

Corollary: The expected cost due to accepting a new task to an M/D/1-PS queue at state $\mathbf{z} = (\Delta_1; ..; \Delta_n)$ is given by

$$w_{\mathbf{z}} = v_{(d;\Delta_1;..;\Delta_n)} - v_{(\Delta_1;..;\Delta_n)} = \boxed{\frac{2\,u_{\mathbf{z}} + d}{1-\rho}.} \tag{4}$$

That is, the immediate cost divided by $1 - \rho$.

Preemptive M/G/1-LIFO: Immediate cost in an M/G/1-LIFO is $(n + 1)x$, where x is the size of the new task.

Aalto University
School of Electrical
Engineering

Cost of a new task in M/D/1-PS

Corollary: The expected cost due to accepting a new task to an M/D/1-PS queue at state $\mathbf{z} = (\Delta_1; ..; \Delta_n)$ is given by

$$w_{\mathbf{z}} = v_{(d;\Delta_1;..;\Delta_n)} - v_{(\Delta_1;..;\Delta_n)} = \boxed{\frac{2\,u_{\mathbf{z}} + d}{1-\rho}.} \qquad (4)$$

That is, the immediate cost divided by $1-\rho$.

Preemptive M/G/1-LIFO: Immediate cost in an M/G/1-LIFO is $(n+1)x$, where x is the size of the new task. Similarly, the expected cost due to accepting a new task with size x is

$$w_{\mathbf{z}} = \boxed{\frac{(n+1)x}{1-\rho},}$$

i.e., the immediate cost divided by $1-\rho$.

Aalto University
School of Electrical Engineering

First policy iteration (FPI) with M/D/1-PS

- Assume: relative values $v_{\mathbf{z}}$ are available for basic policy

Aalto University
School of Electrical Engineering

First policy iteration (FPI) with M/D/1-PS

- Assume: relative values $v_{\mathbf{z}}$ are available for basic policy
- Improved decision according to FPI at state $\mathbf{z}$:

$$\alpha(\mathbf{z}) \triangleq \underset{i}{\text{argmin}} \left(v_{\mathbf{z}'(i)} - v_{\mathbf{z}}\right) = \underset{i}{\text{argmin}}\, w_{\mathbf{z}(i)}$$

where $\mathbf{z}'(i)$ is the new state if the job is added to queue i.

Aalto University
School of Electrical
Engineering

First policy iteration (FPI) with M/D/1-PS

- Assume: relative values $v_{\mathbf{z}}$ are available for basic policy
- Improved decision according to FPI at state $\mathbf{z}$:

$$\alpha(\mathbf{z}) \triangleq \underset{i}{\operatorname{argmin}} \left(v_{\mathbf{z}'(i)} - v_{\mathbf{z}} \right) = \underset{i}{\operatorname{argmin}}\, w_{\mathbf{z}(i)}$$

where $\mathbf{z}'(i)$ is the new state if the job is added to queue i.

"Choose the action with the smallest expected future cost"

Aalto University
School of Electrical Engineering

First policy iteration (FPI) with M/D/1-PS

- Assume: relative values $v_{\mathbf{z}}$ are available for basic policy
- Improved decision according to FPI at state $\mathbf{z}$:

$$\alpha(\mathbf{z}) \triangleq \underset{i}{\operatorname{argmin}} \left(v_{\mathbf{z}'(i)} - v_{\mathbf{z}}\right) = \underset{i}{\operatorname{argmin}}\, w_{\mathbf{z}(i)}$$

where $\mathbf{z}'(i)$ is the new state if the job is added to queue i.

"Choose the action with the smallest expected future cost"

- Basic policy RND-ρ balances load, $\rho_i = \rho_j$, and FPI reduces to

$$\boxed{\alpha(\mathbf{z}) = \underset{i}{\operatorname{argmin}} \left(u_i(\mathbf{z}) + 0.5\, d_i\right).}$$

Aalto University
School of Electrical Engineering

Policy family $\mathcal{P}(\beta)$

Policy family $\mathcal{P}(\beta)$ with policy parameter β is defined by

$$\operatorname*{argmin}_i \; u_i(\mathbf{z}) + \beta \cdot d_i.$$

LWL^- :	$\beta = 0$	"smallest backlog before"
LWL^+ :	$\beta = 1$	"smallest backlog afterwards"
FPI-ρ :	$\beta = 0.5$	"compromise between the above"

Aalto University
School of Electrical Engineering

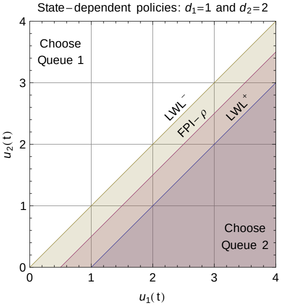
Policy family $\mathcal{P}(\beta)$

Policy family $\mathcal{P}(\beta)$ with policy parameter β is defined by

$$\underset{i}{\text{argmin}}\ u_i(\mathbf{z}) + \beta \cdot d_i.$$

LWL$^-$:	$\beta = 0$	"smallest backlog before"
LWL$^+$:	$\beta = 1$	"smallest backlog afterwards"
FPI-ρ :	$\beta = 0.5$	"compromise between the above"

State-dependent policies in $\mathcal{P}(\beta)$ are of the switch-over type:



Aalto University
School of Electrical Engineering

Numerical examples

Performance metrics:

1. Absolute mean delay (sojourn time)
2. Relative delay when compared to FPI policy

Aalto University
School of Electrical Engineering

Numerical examples

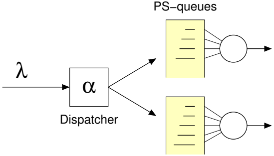
Performance metrics:

1. Absolute mean delay (sojourn time)
2. Relative delay when compared to FPI policy

Scenarios:

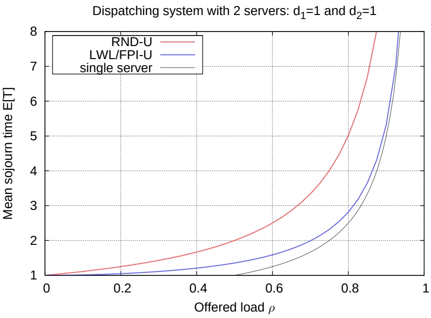
1. Symmetric case with two identical servers
2. Asymmetric case with two heterogeneous servers

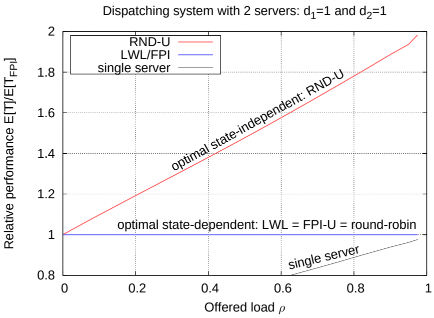
Additionally, policy optimization within $\mathcal{P}$



Aalto University
School of Electrical
Engineering

Identical servers

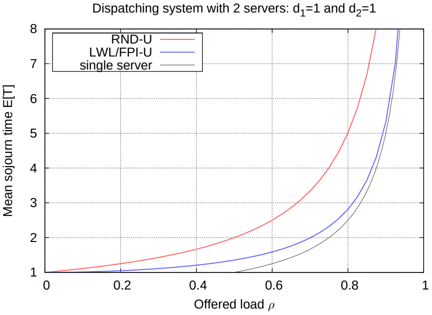


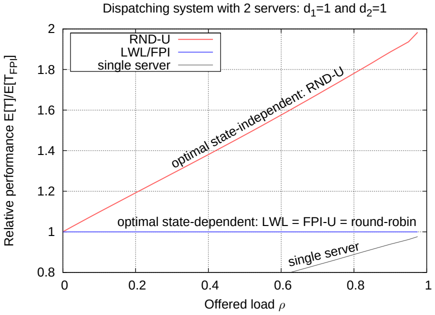


- Two policies, i) RND-U, ii) LWL/FPI/RR, and single server
- Left: resulting mean sojourn time
- Right: relative performance against the LWL

Aalto University
School of Electrical Engineering

Identical servers

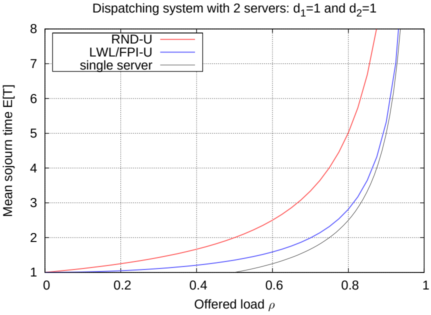


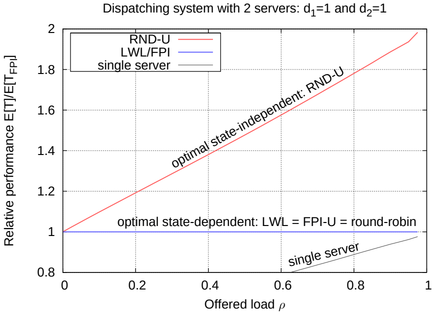


- Two policies, i) RND-U, ii) LWL/FPI/RR, and single server
- Left: resulting mean sojourn time
- Right: relative performance against the LWL
- Optimal state-independent policy: RND-U

Aalto University
School of Electrical Engineering

Identical servers



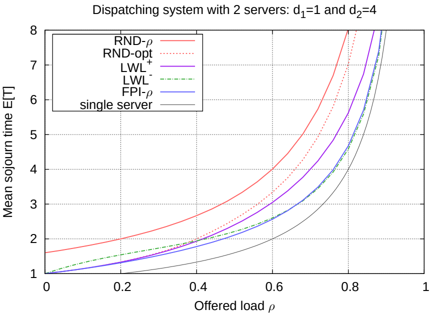


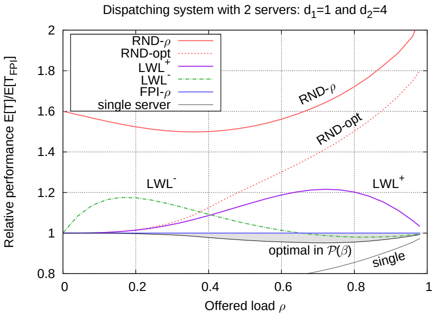
- Two policies, i) RND-U, ii) LWL/FPI/RR, and single server
- Left: resulting mean sojourn time
- Right: relative performance against the LWL
- Optimal state-independent policy: RND-U
- Optimal state-dependent policy: LWL/FPI-U/RR,

"Choose the queue with a smaller backlog"

Aalto University
School of Electrical Engineering

Asymmetric servers: $d_1 = 1$ and $d_2 = 4$

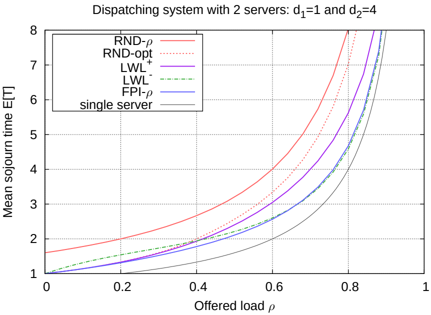


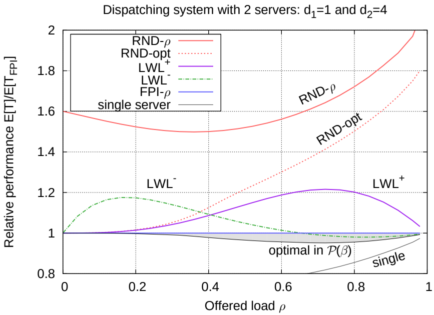


- Left: mean sojourn time
- Right: relative performance against the FPI-ρ policy

Aalto University
School of Electrical Engineering

Asymmetric servers: $d_1 = 1$ and $d_2 = 4$

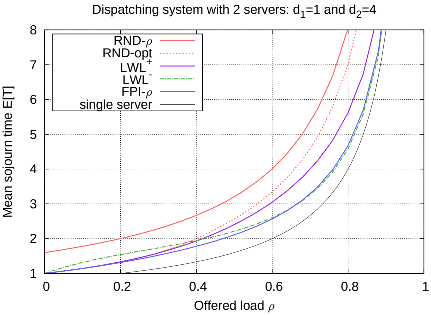


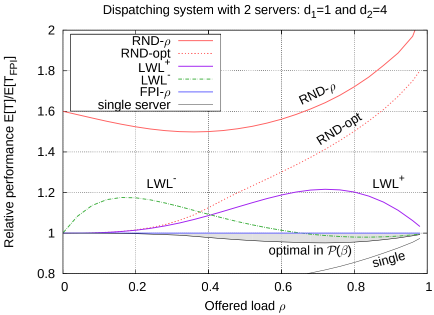


- Left: mean sojourn time
- Right: relative performance against the FPI-ρ policy
- Both LWL policies are clearly suboptimal

Aalto University
School of Electrical Engineering

Asymmetric servers: $d_1 = 1$ and $d_2 = 4$

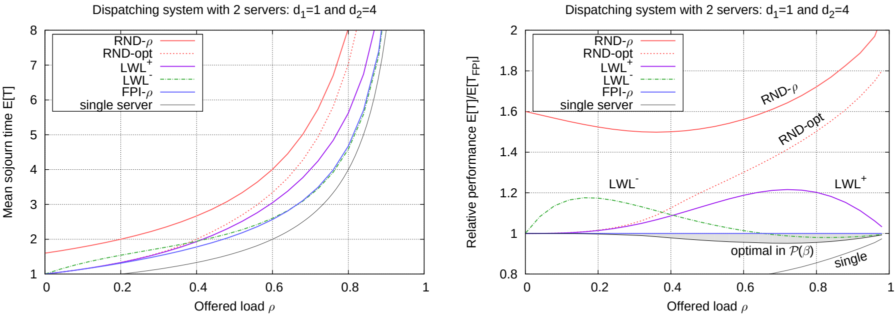




- Left: mean sojourn time
- Right: relative performance against the FPI-ρ policy
- Both LWL policies are clearly suboptimal
- FPI-ρ makes very good dispatching decisions for all ρ

Aalto University
School of Electrical Engineering

Asymmetric servers: $d_1 = 1$ and $d_2 = 4$

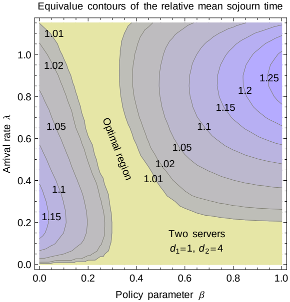


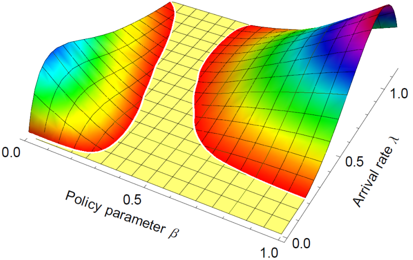
- Left: mean sojourn time
- Right: relative performance against the FPI-ρ policy
- Both LWL policies are clearly suboptimal
- FPI-ρ makes very good dispatching decisions for all ρ
- Gray area: optimal policy from $\mathcal{P}(\beta)$, defined by

$$u_i(\mathbf{z}) + \beta \cdot d_i.$$

Aalto University
School of Electrical Engineering

Policy optimization in $\mathcal{P}(\beta)$

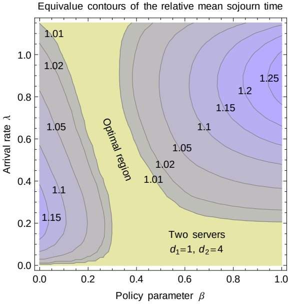


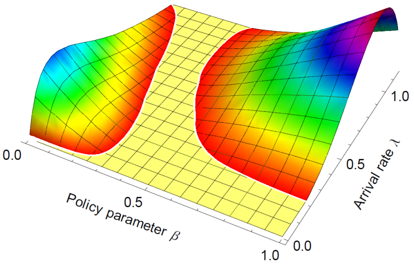


- Two servers, $d_1 = 1$ and $d_2 = 4$

Aalto University
School of Electrical Engineering

Policy optimization in $\mathcal{P}(\beta)$

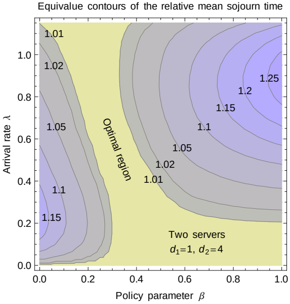


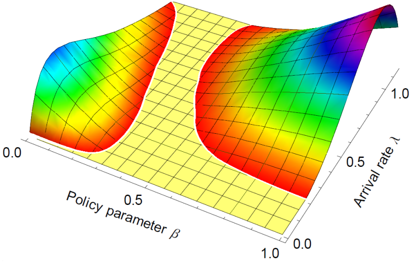


- Two servers, $d_1 = 1$ and $d_2 = 4$
- x-axis: policy parameter β
  y-axis: arrival rate λ
  z-axis: mean delay relative to the optimal at given λ

Aalto University
School of Electrical Engineering

Policy optimization in $\mathcal{P}(\beta)$

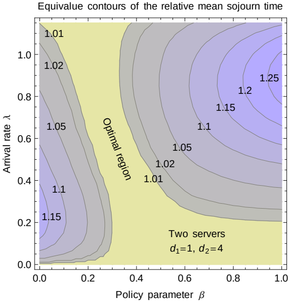


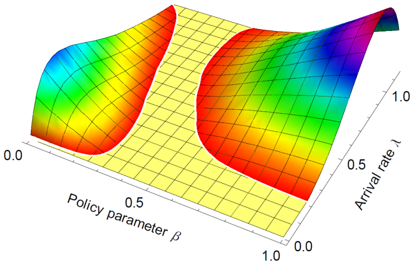


- Two servers, $d_1 = 1$ and $d_2 = 4$
- x-axis: policy parameter β
  y-axis: arrival rate λ
  z-axis: mean delay relative to the optimal at given λ
- Valley: delay is within 1% from the minimum at given λ

Aalto University
School of Electrical Engineering

Policy optimization in $\mathcal{P}(\beta)$





- Two servers, $d_1 = 1$ and $d_2 = 4$
- x-axis: policy parameter β
  y-axis: arrival rate λ
  z-axis: mean delay relative to the optimal at given λ
- Valley: delay is within 1% from the minimum at given λ
- FPI-ρ ($\beta = 0.5$) close to optimal optimal (within $\mathcal{P}$)

Aalto University
School of Electrical Engineering

Conclusions

- Size- and state-aware dispatching problem can be approached in MDP framework

Aalto University
School of Electrical Engineering

Conclusions

- Size- and state-aware dispatching problem can be approached in MDP framework
- FPI requires the relative values of the basic policy

Aalto University
School of Electrical Engineering

Conclusions

- Size- and state-aware dispatching problem can be approached in MDP framework
- FPI requires the relative values of the basic policy
- For a state-independent basic policy, sufficient to analyze M/D/1-PS queue in isolation

Aalto University
School of Electrical Engineering

Conclusions

- Size- and state-aware dispatching problem can be approached in MDP framework
- FPI requires the relative values of the basic policy
- For a state-independent basic policy, sufficient to analyze M/D/1-PS queue in isolation
- We give the relative value for a size-aware M/D/1-PS

Conclusions

- Size- and state-aware dispatching problem can be approached in MDP framework
- FPI requires the relative values of the basic policy
- For a state-independent basic policy, sufficient to analyze M/D/1-PS queue in isolation
- We give the relative value for a size-aware M/D/1-PS
- General case of M/G/1-PS seems to be difficult, however, exact result for a size-aware M/M/1-PS is also available (*Hyytiä et. al, Performance 2011*)

Aalto University
School of Electrical Engineering

Conclusions

- Size- and state-aware dispatching problem can be approached in MDP framework
- FPI requires the relative values of the basic policy
- For a state-independent basic policy, sufficient to analyze M/D/1-PS queue in isolation
- We give the relative value for a size-aware M/D/1-PS
- General case of M/G/1-PS seems to be difficult, however, exact result for a size-aware M/M/1-PS is also available (*Hyytiä et. al, Performance 2011*)
- For FCFS, LCFS, SPT and SRPT, the size-aware relative values are available for M/G/1 (submitted)

Aalto University
School of Electrical Engineering

Conclusions

- Size- and state-aware dispatching problem can be approached in MDP framework
- FPI requires the relative values of the basic policy
- For a state-independent basic policy, sufficient to analyze M/D/1-PS queue in isolation
- We give the relative value for a size-aware M/D/1-PS
- General case of M/G/1-PS seems to be difficult, however, exact result for a size-aware M/M/1-PS is also available (*Hyytiä et. al, Performance 2011*)
- For FCFS, LCFS, SPT and SRPT, the size-aware relative values are available for M/G/1 (submitted)

Thanks!

Aalto University
School of Electrical Engineering

References:

1. E. Hyytiä, A. Penttinen and S. Aalto,
Size- and State-Aware Dispatching Problem with Queue-Specific Job Sizes,
December, 2010, (submitted).
2. E. Hyytiä, A. Penttinen, S. Aalto and J. Virtamo,
Dispatching problem with fixed size jobs and processor sharing discipline,
in 23rd International Teletraffic Congress (ITC'23), September 2011, San Fransisco, USA.
3. E. Hyytiä, J. Virtamo, S. Aalto and A. Penttinen,
M/M/1-PS Queue and Size-Aware Task Assignment,
in IFIP PERFORMANCE, October 2011, Amsterdam, Netherlands, (to appear).

Aalto University
School of Electrical Engineering