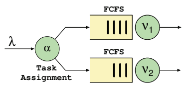
Tutorial on

Size- and Energy-aware Task Assignment in Server Farms



Esa Hyytiä

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

The 26th International Teletraffic Congress (ITC-26)

Outline

1. Introduction

Queues at supermarkets

Queues at supermarkets: static case

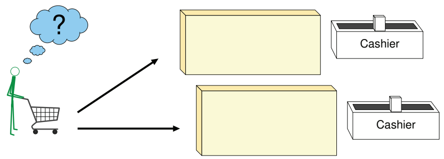
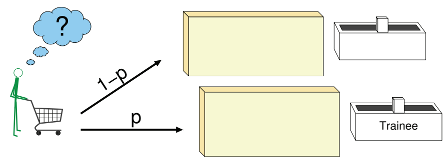


Figure 1: Non-observable system, calls for a static policy

Static policy: routing independent of the state of the system

What is the optimal choice?
- Choose a random cashier?
- Express lines when at most k items?

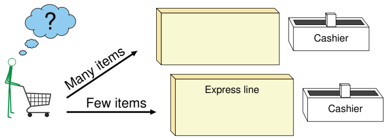
Bernoulli split (RND)



RND: *"Assign a job to server i with probability of p_i"*

Optimal when no information on
- Jobs (size, type, class, ...)
- System's state

Size-interval-Task-Assignment (SITA)



SITA: *"Assign short jobs to server 1, and long to server 2"*

More precisely:

- Size x of the current job is known
- Divide job sizes to k consecutive intervals $I_1, \dots, I_k$
- Server i receives the jobs belonging to size interval I_k

Dynamic case

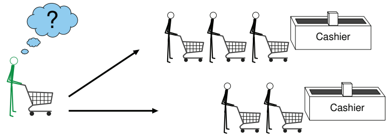


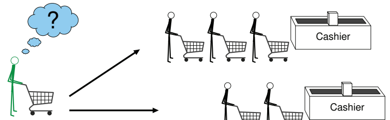
Figure 2: Number of customers can be observed

Dynamic policy: routing depends on the state of the system

What is the optimal choice given the number of customers?

- Join the shortest queue?
- Is that always a better policy than, e.g., the static SITA?
- What if some cashier is slower than another?

Join-the-shortest-queue (JSQ)



JSQ: *"Choose the queue with the least number of jobs"*

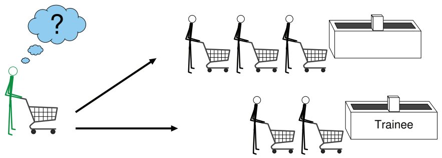
Optimal for the mean delay when: (Winston, 1977)

- Servers are identical
- Service times are exponentially distributed

However: When job sizes vary a lot, SITA outperforms JSQ!

- With JSQ short jobs get stuck behind the long jobs
- SITA avoids this by explicitly segregating the short and long jobs!

Slow server problem



- One fast server, one slow server
- JSQ is no longer optimal ...
- Neither is greedy[1] ...
- Difficult problem in general!
  - When to route a job to a slower server

[1] Individually optimal: the queue with the shortest expected delay

Size-aware case

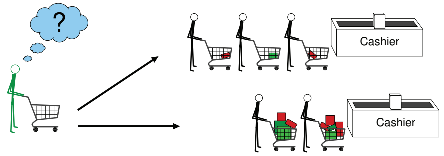


Figure 3: Actual (or expected) service times are available

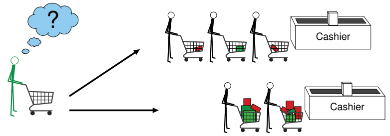
Size-aware setting:

- Exact information about the current state
- Stochastic component: later arriving jobs

What is the optimal choice?

- Choose the queue with the shortest delay?
- Even if I have MANY items in my cart?

Least-work-left (LWL)



LWL: *"Choose the queue with the shortest backlog"*

- Optimal for mean delay when
  - Identical servers
  - Constant service times

However, when job sizes vary a lot, SITA outperforms also LWL!

- With JSQ & LWL, short jobs get stuck behind the long jobs

Lesson: *Take into account also later arriving jobs!*

Summary of Scenarios

	Setting	Heuristic policies	Type
i)	State-unware	RND, SITA	Static
ii)	Number-aware	JSQ	Dynamic
iii)	Size-aware	LWL	

- Objective: minimize the mean delay
- Each policy optimal ONLY in certain scenarios
- Often homogeneous system required
- Difficulty lies with the jobs arriving in the future:
  1. Process the present jobs efficiently, but
  2. Ensure later arriving jobs do not suffer too much

Individually optimal ≠ socially optimal!

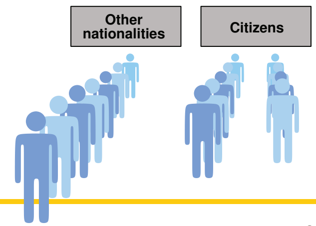
Individually optimal: greedy decisions by customers ("my delay")
Socially optimal: take into account also other customers ("mean delay")

Other scenarios

"Parallel servers with dispatching"

Other scenarios

1. Call centers / helpdesks
   - "servers with different skills" (cf. language skills)
2. Immigration lines
3. Manufacturing systems



Street tolls: Lane selection problem

- See, e.g., Conolly (1984)
- Special lanes:
  - Exact change only
  - Credit card only
  - Electronic pass only

Packet routing problem

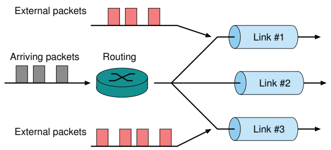


Figure 4: Choosing a link for each packet

- Two or more alternative links (paths)
- Background traffic often present
- Popular heuristics:
  - Random (Bernoulli) split
  - Round-robin: regulates the inter-arrival time

Distributed Computing

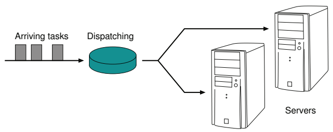


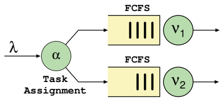
Figure 5: Choosing a server for each job

A large number of scenarios:

- Web-server farms (CDNs, Akamai, Google, Facebook)
- Super-computing (CSC)
- Cloud computing

Job sizes often available!

Dispatching Problem

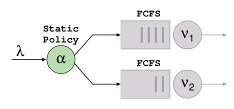
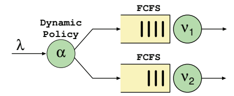
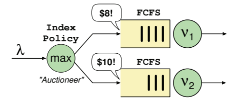


Dispatching Problem

- *k parallel heterogeneous servers with*
  - *Service rate ν_i*
  - *FCFS scheduling*
- *Jobs arrive according to a Poisson process with rate λ*
- *Jobs are dispatched upon arrival*
- *Objective is to minimize the mean delay*

$$\min \mathrm{E}[T]$$

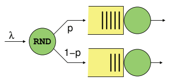
Classification of Dispatching Policies

Static policy: Decision may depend only on the job itself (size, value, class) - not on past decisions - not on current state of the queues	
Dynamic policy: Decision takes into account the new job and the states of the queues	
Index policy: Each queue computes independently *"an offer"* for the new job, and the best offer wins	

1.4 Optimality Results

Random Bernoulli splitting (RND)

"Choose the queue independently in random using probabilities p_i"



- Often easy to analyze (decomposition of Poisson process)
- The load balancing p_i are independent of the arrival rate
- Robust basic policy if no information is available

Altman et al. (2011)

RND is an optimal static policy for Poisson arrivals and PS servers (with server-specific holding costs)

- Size information of the new job does not help with PS servers (cf. SITA)

Size-Interval-Task-Assignment (SITA)

"Short jobs to one queue and the long to the other"

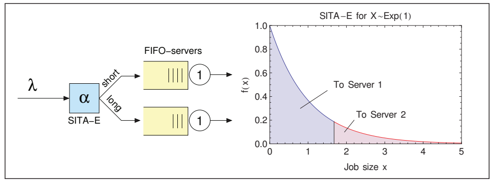


Figure 6: SITA assigns jobs of given size interval to the same server

- For n servers, thresholds $(\xi_1, \ldots, \xi_{n-1})$ define n intervals:

$$\underbrace{(0, \xi_1)}_{\text{Server 1}}, \underbrace{(\xi_1, \xi_2)}_{\text{Server 1}}, \ldots \underbrace{(\xi_{n-1}, \infty)}_{\text{Server } n}$$

SITA (cont.)

- Thresholds ξ_i can be chosen w.r.t. given objective
- Proposed in (Crovella et al., 1998; Harchol-Balter et al., 1999)
- Idea: segregate the short and long jobs from each other
  - High variance in job sizes is a problem for FCFS queues
- SITA-E uses such intervals that balance the load
  - SITA-E is a robust policy that depends only on the job size distribution (not on the arrival rate or the arrival pattern)

Feng et al. (2005)

SITA is optimal static size-aware policy for Poisson arrivals and identical FCFS servers

- SITA gives a lower mean delay than RND for FCFS servers
- SITA is static and thus scales to arbitrary number of dispatchers
- See also (Harchol-Balter et al., 2009) and (Bachmat and Sarfati, 2010)

Join-the-Shortest Queue (JSQ)

"Choose the queue with the least number of jobs"

Winston (1977)

JSQ is optimal for Poisson arrivals, identical servers, and exponential service times when the number in queue is known.

Weber (1978)

JSQ is optimal also for IFR service times.

- First analytical studies by Haight (1958)
- See also Ephremides et al. (1980), Johri (1989), Hordijk and Koole (1990), Towsley et al. (1990), Sparaggis and Towsley (1994), and Koole et al. (1999)
- Optimal also for G/M/1 queues under general assumptions (Akgun et al., 2011)

Round-robin (RR)

"Choose the queue sequentially $1, 2, \ldots, n, 1, \ldots$*"*

Ephremides et al. (1980)

Round-robin is optimal for identical FCFS servers that were initially in a same state when the dispatching history is available.

See also, e.g.,

- Hajek (1983), and Hajek (1985)
- Liu and Towsley (1994), and Liu and Righter (1998)
- Down and Wu (2006), and Wu and Down (2009)

Least-Work-Left (LWL)

"Pick the queue with the shortest backlog"

Daley (1987) (based on (Foss, 1980))

G/G/k (i.e., LWL with general inter-arrival times) stochastically minimizes both the maximum and total backlog with identical servers at an arbitrary arrival time instance

- The counterexample by Stoyan (1976) shows that pathwise RR can yield both a lower waiting time and a lower total backlog (at arrival times)

Harchol-Balter et al. (1999)

The M/G/k system with a central queue is equivalent to LWL

- Thus a server is never idle at the same time when a job is waiting in some queue (cf. work-conserving scheduling in a queue)

Least-Work-Left (LWL) (cont.)

Hyytiä et al. (2011a)

LWL is the optimal policy for Poisson arrivals and identical FCFS or PS servers with a fixed service time

- This system reduces to JSQ if the ties are resolved accordingly

Other remarks:

- LWL is the individually optimal decision for identical FCFS servers
- Can consider pre- and post-assignment backlogs if heterogeneous servers
- LWL is an *index policy*: servers can compute their offers independently

See also:

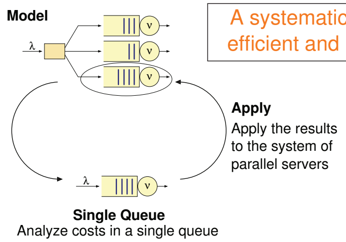
- Harchol-Balter et al. (2009) for a surprising comparison to SITA
- and Sharifnia (1997) (who refers to LWL as JSQ)

Our Approach

- All previous results are for the mean delay in specific homogeneous systems
- We are interested in general energy-aware cost structures with possibly heterogeneous servers

Our approach:

A systematic approach to compute efficient and robust control policies!



1.5 Admission costs

Admission Cost to M/G/1

Suppose we are given an M/G/1-queue

The typical performance metrics are:

1. Stability, is the system stable? *(Is $\rho < 1$?)*
2. Mean delay $E[T]$ *(e.g., Pollaczek-Khinchine)*

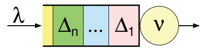
Here we are interested in another quantity:

How much the overall delay increases if a given job is added to the system?

This quantity, admission cost, depends on the state.

Admission Cost to M/G/1-FCFS

Size-aware setup:



- Single server FCFS queue
- Current state:

$$\mathbf{z} = (\Delta_1, \Delta_2, \ldots, \Delta_n),$$

  where Δ_i is the (remaining) service time of job i
- Job 1 receives currently service, job n is the last in queue

How much the total delay will increase on average if a size x job is admitted to the queue?

Components of the Admission cost:

1. What is the delay of the new job? $x + \sum_i \Delta_i$
2. Does accepting it hurt the existing n jobs? **No**
3. Does accepting it hurt jobs arriving in future? **Yes!**

Admission costs to LCFS & PS

Same setup with LCFS queue:

1. What is the delay of the new job? **Depends**
2. Does accepting it hurt the existing n jobs? **Yes!**
3. Does accepting it hurt jobs arriving in future? **No**

Same setup with PS queue:

1. What is the delay of the new job? **Depends**
2. Does accepting it hurt the existing n jobs? **Yes!**
3. Does accepting it hurt jobs arriving in future? **Yes!**

We need to assume something about the arrival process!

We will determine the admission costs in the MDP framework

2. Markov decision processes

Markov Decision Processes (MDPs)

Basic setting:

- Discrete time Markov-chain
- State transition probabilities depend on policy α,

$$p_{ij} = p_{ij}(\alpha)$$

- Some cost structure $\Rightarrow$ mean cost rate $r(\alpha)$
  - E.g., mean number of jobs in M/M/1
- Task: find the optimal policy α,

$$\operatorname*{argmin}_{\alpha} r(\alpha)$$

Example Cost Structures

1. Blocked calls in a loss system
2. Delay in a server system
   - Let $N(t)$ denote the number of jobs in the system at time t
   - Delay costs incurred during time $(0, t)$ are

$$C(t) = \int_0^t N(t)\, dt$$

   - Equivalently: Job i incurs a cost equal to its sojourn time T_i
3. Running costs:[2]
   - When server is busy it incurs costs at rate e_1
   - When server is idle it incurs costs at rate e_0
   - Generalizations, e.g., to different sleeping states

[2] See (Penttinen et al., 2011) and (Hyytiä et al., 2014a,b)

Solving MDPs

- Two standard approaches:
  1. Value iteration
  2. Policy iteration
- Both are based on the so-called value functions $v(z)$

We will utilize the policy iteration approach

... but first few words about those value functions ...

Value Function

- Let $C_{\mathbf{z}}(t)$ denote the cumulative costs incurred during $(0,t)$ from an initial state $\mathbf{z}$
- Let r denote the mean cost rate,

$$r = \lim_{t\to\infty} \frac{C_{\mathbf{z}}(t)}{t}, \qquad \forall\, \mathbf{z}$$

- Value function $v_{\mathbf{z}}$ gives the expected difference in the infinite time-horizon cumulative costs between
  a) system initially in state $\mathbf{z}$, and
  b) system initially in equilibrium,

$$v_{\mathbf{z}} \triangleq \lim_{t\to\infty} \mathrm{E}[\, C_{\mathbf{z}}(t) - r\,t\,]$$

Example: Delay in M/M/1

- For delay in M/M/1 the cost rate $C_{\mathbf{z}}(t)$ is

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

- The value function reads

$$v_{\mathbf{z}} = \lim_{t\to\infty} \left(\mathrm{E}[\int_0^t N_{\mathbf{z}}(s)\, ds] - \mathrm{E}[N]\, t \right).$$

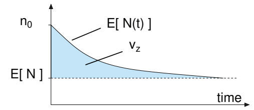


Figure 7: Value function for M/M/1 in state n_0

Example: Delay in size-aware M/G/1-FCFS

M/G/1-FCFS initially in state $\mathbf{z} = (1,3)$:

- Job with remaining size 3 currently receiving service
- Another job with size 1 is waiting
- Also later arriving jobs have to wait (FCFS)

Value function is the expected difference in the infinite time-horizon costs:

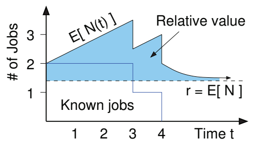


Figure 8: Value function for an size-aware M/G/1-FCFS in state $\mathbf{z}$

Comparison of States

The mean difference in costs incurred between states $(\mathbf{z}_1, \mathbf{z}_2)$ is

$$d(\mathbf{z}_1, \mathbf{z}_2) \triangleq \lim_{t\to\infty} \mathrm{E}[V_{\mathbf{z}_2}(t) - V_{\mathbf{z}_1}(t)],$$

which gives

$$d(\mathbf{z}_1, \mathbf{z}_2) = v_{\mathbf{z}_2} - v_{\mathbf{z}_1}.$$

Admission cost:

- Suppose that
  - State $\mathbf{z}_1$ is the current state
  - State $\mathbf{z}_2$ includes also a new job x, i.e., $\mathbf{z}_2 = \mathbf{z}_1 + x$
- Then $v_{\mathbf{z}_2} - v_{\mathbf{z}_1}$ gives the **admission cost** for the new job!

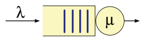
References

1. R. Bellman, *A Markovian decision process*, Indiana Univ. Math. J. 6 (1957).
2. R. A. Howard, *Dynamic Probabilistic Systems, Volume II: Semi-Markov and Decision Processes*, Wiley Interscience, 1971.
3. S. M. Ross, *Applied Probability Models with Optimization Applications*, Holden-Day Inc., 1970.
4. M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, Wiley, 2005.
5. J. Virtamo, *Lecture notes on Markov decision processes*, S-38.141 Teletraffic Theory, TKK," 2004.

3. Number-aware Systems

Delay in M/M/1

Standard M/M/1 Queue:



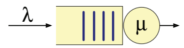
- Performance metric: mean delay $E[T]$
- Number-aware system with exponential service times
- All work-conserving scheduling disciplines are equivalent
  - This includes FCFS, LCFS, PS, ...

Mean delay in M/M/1

The mean delay in an M/M/1 queue is

$$E[T] = \frac{1}{\mu - \lambda}$$

Value Function w.r.t. Delay for M/M/1



Proposition 1

The value function for a work-conserving and a number-aware M/M/1 queue is [3]

$$\boxed{v_n = \frac{1}{2} \cdot \frac{n(n+1)}{\mu - \lambda} - \frac{\lambda\mu}{(\mu-\lambda)^3}.} \qquad (1)$$

[3] Krishnan 1987, Aalto and Virtamo (1996), Virtamo (Lecture slides, 2004)

Proof:

- Without loss of generality, we can assume LCFS
- Current state, n jobs, has no effect on jobs arriving in future
- Mean difference in costs between a system with initially n jobs and an empty system is thus

  "the expected sojourn time of the n jobs"
- Expected sojourn time of the i^{th} job in the queue is[4]

$$\frac{i}{\mu - \lambda}$$

- Therefore,

$$v_n - v_0 = \sum_{i=1}^{n} \frac{i}{\mu - \lambda} = \frac{n(n+1)}{2(\mu - \lambda)}$$

[4] The mean remaining busy period in M/G/1 with backlog u is $u/(1-\rho)$

Proof: (cont.)

- The constant term v_0 follows from the identity

$$\sum_n \pi_n v_n = 0$$

which yields (Hyytiä et al., 2012d)

$$v_n = \boxed{\frac{n(n+1)}{2(\mu-\lambda)} - \frac{\mu\lambda}{(\mu-\lambda)^3}} \qquad \square$$

Remarks:

- Result holds for all work-conserving scheduling disciplines
- The constant term is immaterial and often omitted
- For alternative proofs, see (Krishnan, 1987; Aalto and Virtamo, 1996; Virtamo, 2004; Hyytiä et al., 2012d)
- See also Whittle (1996): Section 10.3 and Section 11.5

Admission cost to M/M/1

By definition, the admission cost to M/M/1 is thus

$$c_n = v_{n+1} - v_n = \boxed{\frac{n+1}{\mu - \lambda}}$$

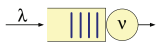
- With LCFS, this is the sum of
  1. the expected sojourn time of the new job
  2. the increase in the sojourn time of the present n jobs

  all equal to $1/(\mu - \lambda)$
- With PS, the same cost is shared among the all present jobs and jobs arriving in the near future
- With FCFS, the same cost is shared among the new job and the jobs arriving in the near future

M/M/1 with Holding Costs

Standard M/M/1:

- Poisson arrival process, rate λ
- Exponential (i.i.d.) service times with mean $1/\mu$
- FCFS scheduling



Holding cost structure:

- Jobs are associated with i.i.d. holding cost $B_i \sim B$, ...which become known upon arrival
- Job i incurs costs at rate B_i until it departs
- State $\mathbf{z} = (b_1, \ldots, b_n)$, where job 1 receives service first

Objective: $\min \mathrm{E}[BT]$

Note:

- Holding cost quantifies the importance of a job!
- For delay, each job is equally important, $B_i = 1$

M/M/1-FCFS with Holding Costs

- Mean cost per job is[5]

$$\mathrm{E}[BT] = \mathrm{E}[B] \cdot \mathrm{E}[T] = \frac{\mathrm{E}[B]}{\mu - \lambda}$$

- Recall that the value function w.r.t. delay (1)

$$v_z - v_0 = \frac{n(n+1)}{2(\mu - \lambda)}$$

corresponds to the additional delay experienced by

1. the present n jobs, and
2. jobs arriving in future

if the queue is initially in state n instead of equilibrium

[5] Holds for all work-conserving scheduling disciplines independent of B_i.

M/M/1-FCFS with Holding Costs

- The expected delay of the present n jobs is

$$d_1 = \frac{1}{\mu} \sum_{i=1}^{n} i = \frac{n(n+1)}{2\mu}$$

- Therefore, the additional delay the future jobs experience is

$$\frac{n(n+1)}{2(\mu - \lambda)} - \frac{n(n+1)}{2\mu} = \frac{\lambda\, n(n+1)}{2(\mu - \lambda)\mu}$$

- The value function w.r.t. holding costs is

$$\overbrace{\frac{1}{\mu} \sum_{i=1}^{n} i\, b_i}^{\text{present } n \text{ jobs}} + \overbrace{\frac{\lambda\, n(n+1)}{2(\mu - \lambda)\mu} \mathrm{E}[B]}^{\text{future jobs}}$$

Number-aware value functions for M/M/1 Queues

A similar analysis can be carried out also for LCFS and PS[6]:

	Present jobs		Future jobs	Number-aware value functions $v_z - v_0$	
	Costs incurred	Total delay	Delay increase	Holding costs	Delay
FCFS	$\frac{1}{\mu}\sum_i i\, b_i$	$\frac{n(n+1)}{2\mu}$	$\frac{\lambda n(n+1)}{2(\mu-\lambda)\mu}$	$\frac{1}{\mu}\sum_i i\, b_i + \frac{\lambda n(n+1)}{2(\mu-\lambda)\mu}\mathrm{E}[B]$	$\frac{n(n+1)}{2(\mu-\lambda)}$
LCFS	$\frac{1}{\mu-\lambda}\sum_i i\, b_i$	$\frac{n(n+1)}{2(\mu-\lambda)}$	-	$\frac{1}{\mu-\lambda}\sum_i i\, b_i$	-"-
PS	$\frac{n+1}{2\mu-\lambda}\sum_i b_i$	$\frac{n(n+1)}{2\mu-\lambda}$	$\frac{\lambda n(n+1)}{2(\mu-\lambda)(2\mu-\lambda)}$	$\frac{n+1}{2\mu-\lambda}\sum_i b_i + \frac{\lambda n(n+1)\mathrm{E}[B]}{2(\mu-\lambda)(2\mu-\lambda)}$	-"-

Remarks:

- With b_i=1 the value function w.r.t. holding costs reduces to the one w.r.t. delay
- With FCFS and LCFS, job 1 is currently receiving service (head of the queue)

[6] See Doroudi et al. (2014) for PS

M/M/s:

Proposition 2 (*M/M/s*)

For the value function of an M/M/s queue w.r.t. delay it holds that

$$v_{k+1} - v_k = \begin{cases} \dfrac{W}{\mathrm{Erl}(k,a)} + \dfrac{1}{\mu}, & 0 \le k \le s, \\ \dfrac{W}{\mathrm{Erl}(s,a)} + \dfrac{k-s}{s\mu(1-\rho)} + \dfrac{1}{\mu}, & k > s, \end{cases} \tag{2}$$

where $\mathrm{Erl}(k, a)$ *denotes the Erlang's blocking formula with* k *servers and the offered load of* $a = \lambda/\mu$.

Proof.

See Krishnan (1987, 1990). □

Loss systems

- In queueing systems one typically minimizes the mean delay
- In *loss systems* the performance metric is the blocked customers
- A prime example is the classical Erlang's loss system, M/M/s/s:

Erlang's loss system (M/M/s/s)

- *s system places*
- *s servers (i.e., there are no waiting places)*

Erlang's blocking formula,

$$\mathrm{Erl}(s,a) = \frac{a^s/s!}{1 + a + a^2/2! + \ldots + a^s/s!}$$

The mean number of customers is $\mathrm{E}[N] = a(1 - \mathrm{Erl}(s,a))$

M/M/s/s – Erlang's Loss System

Proposition 3 (*M/M/s/s*)

For the value function of M/M/s/s w.r.t. blocked calls it holds that

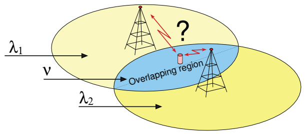
$$c_k = v_{k+1} - v_k = \frac{\mathrm{Erl}(s,a)}{\mathrm{Erl}(k,a)}, \qquad (3)$$

where $k = 0, 1, \ldots, (s-1)$ *denotes the number of jobs (calls) upon arrival, and* $\mathrm{Erl}(k,a)$ *is the Erlang's blocking formula with* $a = \lambda/\mu$.

Proof.

See Krishnan and Ott (1986). □

Example: mobile network



- Mobile users are associated with either of the two base stations[7]
- Three types of users:
  1. Those who can communicate only with base station 1
  2. Those who can communicate only with base station 2
  3. Those who can communicate with either (flexible users)

Task: Choose a base station for the flexible users so as to minimize the mean blocking probability

[7] Adapted from van Leeuwaarden et al. (2001).

M/M/s/k

Proposition 4 (*M/M/s/k*)

For the value function of M/M/s/k w.r.t. blocked calls it holds that

$$c_j = v_{j+1} - v_j = \lambda \cdot \mathrm{E}[t_j^*] \cdot B(s,k,\rho) \qquad (4)$$

where $\rho = \lambda/\mu/s$,

$$\mathrm{E}[t_j^*] = (\lambda \cdot B(\min\{j,s\}, j, \rho))^{-1}$$

and $B(s,k,\rho)$ *denotes the blocking probability of an M/M/s/k system,*

$$B(s,k,\rho) = \frac{s^s}{s!}\rho^k \cdot \left(\sum_{j=0}^{s-1} \frac{(s\rho)^j}{j!} + \frac{(s\rho)^s}{s!} \cdot \frac{1-\rho^{k-s+1}}{1-\rho} \right)^{-1}$$

See (van Leeuwaarden et al., 2001) for a proof and numerical examples.

References

1. Krishnan and Ott, *State-dependent routing for telephone traffic: Theory and results*, in IEEE Conference on Decision and Control, 1986.
2. Krishnan, *Joining the right queue: a Markov decision rule*, in the 28th Conference on Decision and Control, 1987.
3. Krishnan, *Joining the right queue: a state-dependent decision rule*, IEEE Transactions on Automatic Control, 1990.
4. Whittle, *Optimal Control: Basics and Beyond*, Wiley, 1996.
5. Aalto and Virtamo, *Basic packet routing problem*, in NTS-13, 1996.
6. van Leeuwaarden, Aalto and Virtamo, *Load Balancing in Cellular Networks Using First Policy Iteration*, Technical Report, Networking Laboratory, TKK, 2001.
7. Hyytiä, Penttinen and Sulonen, *Non-Myopic Vehicle and Route Selection in Dynamic DARP with Travel Time and Workload Objectives,* Computers & Operations Research, 2012.
8. Virtamo, *Lecture notes on Markov decision processes*, S-38.141 Teletraffic Theory, TKK," 2004.

4. Size-aware Systems

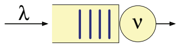
Size-aware System

Size-aware means that

- Service requirement (job size) become known upon arrival
- Scheduling discipline can utilize the size information
- Dispatcher is aware of the (remaining) service times

Common feature especially in ICT context, cf. file sizes.

Size-aware Value Functions for M/G/1



A. Elementary scheduling disciplines:
- M/G/1-FCFS
- M/G/1-LCFS

B. Size-aware scheduling disciplines:
- Size-aware scheduling
- M/G/1-SPT (shortest-processing-time)
- M/G/1-SRPT (shortert-remaining-processing-time)
- M/G/1-SPTP (shortest-processing-time-product)

C. Processor sharing (PS)
- M/D/1-PS (fixed job sizes)
- M/M/1-PS

M/G/1: Notation

Basic case:
- Poisson arrival rate λ
- Service times X_i i.i.d., $X_i \sim X$
- Offered load $\rho = \lambda\,\mathrm{E}[X]$
- Size-aware state $\mathbf{z} = (\Delta_1; ..; \Delta_n)$ with n jobs:
  - Δ_i is the remaining service time of job i
- Backlog $u_{\mathbf{z}} = \sum_i \Delta_i$

With arbitrary holding costs:
- State $\mathbf{z} = ((\Delta_1, b_1); ..; (\Delta_n, b_n))$
  - b_i is the holding cost of job i, and $B_i \sim B$
- $\mathrm{E}[B]$ is the mean holding cost (arbitrary job)

Slowdown Metric

- Size-aware scenario
  - It is natural to consider also size-based metrics
- Slowdown of a job is defined as[8]

$$\gamma \triangleq \frac{T}{X} = \frac{\text{sojourn time}}{\text{service requirement}} \qquad (5)$$

- Idea: large tasks can wait longer
- Equivalently, the (job-specific) holding cost b is inversely proportional to the (known) service requirement x

$$b = \frac{1}{x}$$

[8] Yang and de Veciana (2002) refer to (5) as the bit-transmission delay.

Holding Cost Structure

Summary

Holding cost:
- *Job i accrues costs at job-specific rate $B_i \sim B$*

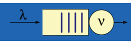
Delay with $\mathbf{b_i = 1}$:
- *Total cost rate is the number of jobs in the system, N_t*
- *Cost that job i incurs is equal to its latency*

$$b_i \cdot T_i = T_i$$

Slowdown with $\mathbf{b_i = 1/x_i}$:
- *Cost that job i incurs is equal to its slowdown*

$$b_i \cdot T_i = \frac{T_i}{x_i}$$

Size-aware M/G/1-FCFS



Notation:

- Poisson arrival process with rate λ
- Offerent load $\rho = \lambda\,\mathrm{E}[X]$
- State $\mathbf{z} = (\Delta_1; ..; \Delta_n)$, where Δ_i is the (remaining) service time of job i
- Job 1 is served first, job n is at the end of the queue
- $u_{\mathbf{z}} = \sum_i \Delta_i$ is the backlog in the queue

Proposition 5 (*Size-aware M/G/1-FCFS*)

The value function of size-aware M/G/1-FCFS w.r.t. delay satisfies[9] [10]

$$v_{(\Delta_1;..;\Delta_n)} - v_0 = \frac{\lambda\, u_{\mathbf{z}}^2}{2(1-\rho)} + \sum_{i=1}^{n} (n+1-i)\,\Delta_i \qquad (6)$$

[9] Hyytiä et al. (2012c,b)

[10] For M/M/1 see Aalto and Virtamo (1996) and Hyytiä et al. (2012d)

Proof

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

Observations:
1. Both systems behave identically once S1 becomes empty
2. $v_{\mathbf{z}} - v_0$ is equal to the additional time jobs spent in S1

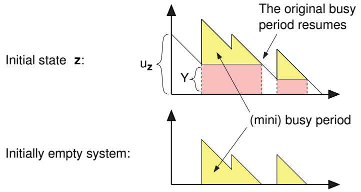
$$v_{\mathbf{z}} - v_0 = V_1 + V_2$$

V_1 = the (remaining) delay of present jobs (only in S1)
V_2 = the expected additional delay the later arrivals in S1

When job i is processed $(n+1-i)$ of the present jobs are in the system, and therefore $V_1 = \sum_{i=1}^{n}(n+1-i)\,\Delta_i$.

Proof (cont.)

- A later arriving task starts a busy period in S2, which corresponds to a mini busy period in S1



- During busy periods, arriving jobs increase the cumulative delay by an amount equal to the post arrival workload
- These jobs experience an additional delay Y in S1
- Otherwise the delay contributions are equal!

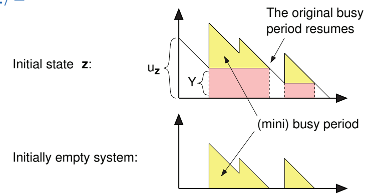
Proof (cont.)

Summing up:

- Mean number of (mini) busy periods before S1 empty: $\lambda\, u_{\mathbf{z}}$
- Mean number of jobs served during a busy period: $1/(1-\rho)$
- Mean offset $E[Y] = u_{\mathbf{z}}/2$

Therefore,

$$V_2 = \lambda\, u_{\mathbf{z}} \cdot \frac{1}{1-\rho} \cdot \frac{u_{\mathbf{z}}}{2} = \frac{\lambda\, u_{\mathbf{z}}^2}{2(1-\rho)},$$



and $V_1 + V_2 = v_{\mathbf{z}} - v_0$, which completes the proof. □

M/G/1-FCFS

Some remarks

- The proof is by a coupling argument, which is utilized also later
- The opposite numbering (job n at the head of the queue) gives

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

- Note that $v_{(\Delta_1;..;\Delta_n)}$ is insensitive to service time distribution[11]

Admission cost to M/G/1-FCFS

$$c_{\mathbf{z}}(x) = v_{(\Delta_1;..;\Delta_n;x)} - v_{(\Delta_1;..;\Delta_n)} = \frac{\lambda}{2(1-\rho)}(2u_{\mathbf{z}}x + x^2) + u_{\mathbf{z}} + x \quad (7)$$

[11] Unlike the mean delay, which depends on the second moment $E[X^2]$

M/G/1-FCFS with Holding Costs

Proposition 6 (*Size-aware M/G/1-FCFS*)

The value function of size-aware M/G/1-FCFS w.r.t. arbitrary job-specific holding costs b_i satisfies [12]

$$v_{\mathbf{z}} - v_0 = \sum_{i=1}^{n} \left(\Delta_i \sum_{j=i}^{n} b_j \right) + \frac{\lambda\, u_{\mathbf{z}}^2}{2\,(1-\rho)} E[B]. \quad (8)$$

Proof.

The result follows directly from identifying the terms in (6)

$$v_{\mathbf{z}} - v_0 = \overbrace{\sum_{i=1}^{n} (n+1-i)\, \Delta_i}^{\text{present jobs}} + \overbrace{\frac{\lambda\, u_{\mathbf{z}}^2}{2(1-\rho)}}^{\text{future jobs}}$$

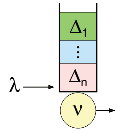
and adding the appropriate weights. □

[12] Hyytiä et al. (2012a)

Size-aware M/G/1-LCFS (preemptive)

Notation:

- Poisson arrival process with rate λ
- Offerent load $\rho = \lambda\, E[X]$
- State $\mathbf{z} = (\Delta_1; ..; \Delta_n)$, where Δ_i is the (remaining) service time of job i
- Job n is the most recent arrival currently being processed



Proposition 7 (*Size-aware M/G/1-LCFS*)

The value function of size-aware M/G/1-LCFS w.r.t. delay satisfies [13]

$$v_{(\Delta_1;..;\Delta_n)} - v_0 = \frac{1}{1-\rho} \sum_{i=1}^{n} i \cdot \Delta_i. \quad (9)$$

Note: Insensitivity to service time distribution.

[13] Hyytiä et al. (2012c)

Proof

We prove also this result by a coupling argument:

- Consider two systems under same arrivals:
  1. S1 initially in state $\mathbf{z} = (\Delta_1, .., \Delta_n)$,
  2. S2 initially empty.
- Let D_i denote the (remaining) delay of job i in S1.
- With LCFS, the current state has no effect on the future arrivals' sojourn times.
- The difference between the relative value of S1 and S2 is equal to the mean remaining delay of the n present jobs,

$$v_{(\Delta_1;..;\Delta_n)} - v_0 = \sum_{i=1}^{n} E[D_i].$$

Proof (cont.)

- Remaining delay D_n of job n is given by a random sum,

$$D_n = \Delta_n + \left(B_1 + \ldots + B_{A(\Delta_n)}\right)$$

where $A(\Delta_n)$ denotes the number of (mini) busy periods during time Δ_n, and B_i the corresponding durations,

$$\mathrm{E}[B_i] = \mathrm{E}[X]/(1-\rho)$$

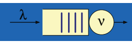
- Taking the expectation on both sides gives

$$\mathrm{E}[D_n] = \Delta_n + \mathrm{E}[A(\Delta_n)] \cdot \mathrm{E}[B] = \frac{\Delta_n}{1-\rho}$$

- Similarly,

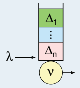
$$\mathrm{E}[D_i] = \frac{\sum_{j=i}^{n} \Delta_j}{1-\rho} \quad \Rightarrow \quad v_{\mathbf{z}} - v_0 = \sum_{i=1}^{n} \mathrm{E}[D_i] = \boxed{\frac{1}{1-\rho} \sum_{i=1}^{n} i \cdot \Delta_i} \quad \square$$

M/G/1-LCFS with Holding Costs



Proposition 8 (*Size-aware M/G/1-LCFS*)

The value function of size-aware M/G/1-LCFS w.r.t. arbitrary job-specific holding costs b_i satisfies[14]



$$\boxed{v_{\mathbf{z}} - v_0 = \frac{1}{1-\rho} \sum_{i=1}^{n} \left(\Delta_i \sum_{j=1}^{i} b_j \right)}. \qquad (10)$$

Proof.

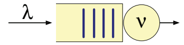
The expected sojourn time of job i is $\mathrm{E}[D_i] = (1-\rho)^{-1} \sum_{j=i}^{n} \Delta_j$.
As the future arrivals are not affected by the current state,

$$v_{\mathbf{z}} - v_0 = \sum_{i=1}^{n} b_i \, \mathrm{E}[D_i] = \frac{1}{1-\rho} \sum_{i=1}^{n} \left(b_i \sum_{j=i}^{n} \Delta_j \right)$$

which is equivalent to (10). □

[14] Hyytiä et al. (2012a)

Size-aware Value Functions for M/G/1



A. Elementary scheduling disciplines:
- M/G/1-FCFS
- M/G/1-LCFS

B. Size-aware scheduling disciplines:
- Size-aware scheduling
- M/G/1-SPT (shortest-processing-time)
- M/G/1-SRPT (shortert-remaining-processing-time)
- M/G/1-SPTP (shortest-processing-time-product)

C. Processor sharing (PS)
- M/D/1-PS (fixed job sizes)
- M/M/1-PS

Size-aware Scheduling

SPT: (shortest-processing-time)

"Assign the shortest job to server first"

- Optimal non-preemptive scheduling for delay (Schrage, 1968)

SRPT: (shortest-remaining-processing-time)

"Serve job with the shortest remaining service time"

- Optimal preemptive scheduling for delay
  - Holds for any arrival sequence (for each sample path)

SPTP: (shortest-processing-time-product)

"Serve job with the smallest product of initial and remaining service time"

- Specifically tailored for the slowdown metric [15,16]

[15] Proposed by Yang and de Veciana (2002)
[16] Wierman et al. (2005) refer to SPTP as the RS policy

Size-aware Scheduling

Index based scheduling:

Job with the smallest index ("offer") is served first

Notation:

Δ_i Remaining service time of job i
Δ_i^* Initial service time of job i

Scheduling	Index	Optimality
SPT	Δ_i^*	optimal non-preemptive / delay & slowdown
SRPT	Δ_i	optimal preemptive / delay
SPTP	$\Delta_i \cdot \Delta_i^*$	optimal preemptive / slowdown[17]

[17] Hyytiä, Aalto, Penttinen, SIGMETRICS'12.

SPTP

Optimality of SPTP (Hyytiä et al., 2012a)

SPTP is the optimal scheduling in M/G/1 w.r.t. slowdown

Remarks:

- Unlike with SRPT, this does not hold for every arrival sequence
- Proof is based on Gittins index

Gittins index, $M/G/1$ multi-class queue

The *Gittins index* for a class-k job with attained service a:

$$G_k(a) = \sup_{\delta>0} \frac{w_k\, P\{X_k - a \le \delta \mid X_k > a\}}{E[\min\{X_k - a, \delta\} \mid X_k > a]},$$

w_k: class-k holding cost
X_k: class-k service requirement

The *Gittins index policy* serves the job i^* such that

$$i^* = \arg\max_i \; G_{k_i}(a_i),$$

k_i: class of job i
a_i: attained service of job i

Proposition 9 (*Gittins*)

The Gittins index policy minimizes the mean holding costs,

$$\sum_k p_k w_k E[T_k],$$

p_k : *fraction of class-k jobs*
T_k : *sojourn time of a class-k job*

among the non-anticipating scheduling policies.

Optimality of SPTP

- Non-anticipating policies are not aware of the (remaining) service times $x_k - a_k$
- Idea: the initial service requirement = class
- That is, a deterministic service time x_k per class k

(Technical assumption in the proof: finite set of service times)

Sketch of Proof

- Single-server M/G/1-queue, load $\rho < 1$
- Associate: class $k \leftrightarrow$ service time x_k
- Gittins index is now

$$G_k(a) = \frac{w_k}{x_k - a} = \frac{\text{holding cost rate}}{\text{remaining service time}}$$

with the optimal δ equal to $x_k - a$.

- Gittins theorem: optimal policy that serves job i^* such that

$$i^* = \operatorname{argmin}_i \frac{\Delta_i}{w_{k(i)}}$$

Δ_i : remaining service time of job i
$k(i)$: class of job i

Sketch of Proof (cont.)

Gittins index policy:

$$i^* = \operatorname{argmin}_i \frac{\Delta_i}{w_{k(i)}}$$

Δ_i : remaining service time of job i
$k(i)$: class of job i

If we choose $w_k = 1/x_k$, we see that the mean slowdown is minimized by SPTP.

If we choose $w_k = 1$, for all k, we obtain the well-known optimality result of SRPT with respect to the mean sojourn time.

Optimal Single-Server Scheduling

	non-preemptive		preemptive	
	class-aware	size-aware	non-anticipating	anticipating size-aware
delay	SEPT ($c\mu$-rule)	SPT	FB, FCFS,... (depends on $f(x)$)	SRPT
slowdown	-"-	-"-	FB, FCFS, ... (depends on $f(x)$)	SPTP (M/G/1)

Multi-Server Scheduling

- Multi-server scheduling is more difficult
- Basic slow server problem:
  - One fast and one slow server, and a shared queue
  - What is the optimal *scheduling* w.r.t. the mean delay?
- The optimal scheduling is a threshold policy:

Activate the slower server only when the number in the system is greater than n^*

References:

1 (Larsen, 1981): *first studies and conjecture of the optimality of threshold policy*
2 (Agrawala et al., 1984): *optimality for exp-jobs without arrivals*
3 (Lin and Kumar, 1984), (Walrand, 1984), (Koole, 1995): *with Poisson arrivals*
4 (Viniotis and Ephremides, 1988), (Righter and Xu, 1991): *non-exponential service times*
5 (Véricourt and Zhou, 2006): *more than two servers (hard!)*
6 (Akgun et al., 2014): *with energy consumption*

Size-aware M/G/1: Additional notation

1. Jobs in state $\mathbf{z}$ are numbered so that (without new arrivals) job 1 is served first and job n last
2. $f(x)$ denotes the pdf of the service time
3. $\rho(x)$ denotes the load due to jobs shorter than x

$$\rho(x) \triangleq \lambda \int_0^x t\, f(t)\, dt$$

4. Define

$$h(x) \triangleq \frac{f(x)\, b(x)}{(1-\rho(x))^2}$$

where $b(x)$ is the mean holding cost of a job with size x

$$b(x) = \mathrm{E}[B \,|\, X = x]$$

M/G/1-SPT (Non-preemptive)

Proposition 10 (*Hyytiä et al. (2012c)*)

The size-aware relative value of state $\mathbf{z}$ with respect to arbitrary holding costs in an M/G/1-SPT queue is

$$v_{\mathbf{z}} - v_0 = \sum_{i=1}^{n} b_i \left(\Delta_i + \frac{1}{1-\rho(\Delta_i)} \left(\sum_{j=1}^{i-1} \Delta_j \right) \right) + \frac{\lambda}{2} \sum_{i=1}^{n} \left[\left(\sum_{j=i+1}^{n} \Delta_j^2 + \left(\sum_{j=1}^{i} \Delta_j \right)^2 \right) \int_{\tilde{\Delta}_i}^{\tilde{\Delta}_{i+1}} h(x)\, dx \right] \quad (11)$$

- *Job 1 is receiving service, and $\Delta_2 < \ldots < \Delta_n$ (SPT order)*
- $\tilde{\Delta}_i = \begin{cases} 0, & i = 1 \\ \Delta_i, & i = 2, \ldots, n \\ \infty & i = n+1 \end{cases}$

M/G/1-SRPT with Holding Costs

Proposition 11 (*Hyytiä et al. (2012c)*)

The size-aware value function of an M/G/1-SRPT queue w.r.t. arbitrary holding costs satisfies

$$v_{\mathbf{z}} - v_0 = \sum_{i=1}^{n} b_i \left(\frac{1}{1-\rho(\Delta_i)} \left(\sum_{j=1}^{i-1} \Delta_j \right) + \int_0^{\Delta_i} \frac{1}{1-\rho(x)}\, dx \right) + \frac{\lambda}{2} \sum_{i=0}^{n} \left[\left(\sum_{j=1}^{i} \Delta_j \right)^2 \int_{\Delta_i}^{\Delta_{i+1}} h(x)\, dx + (n-i) \int_{\Delta_i}^{\Delta_{i+1}} x^2\, h(x)\, dx \right] \quad (12)$$

- *Job 1 receives currently service and $\Delta_1 < \ldots < \Delta_n$,*
- $\Delta_0 = 0$ *and* $\Delta_{n+1} = \infty$

M/G/1-SRPT - Alternative Form

Proposition 12 (*Hyytiä et al. (2012a)*)

The size-aware value function of an M/G/1-SRPT queue w.r.t. arbitrary holding costs satisfies

$$v_{\mathbf{z}} - v_0 = \sum_{i=1}^{n} b_i \left(\frac{u_{\mathbf{z}}(\Delta_i)}{1-\rho(\Delta_i)} + \int_0^{\Delta_i} \frac{1}{1-\rho(t)}\, dt \right) + \frac{\lambda}{2} \int_0^{\infty} h(x) \left(u_{\mathbf{z}}(x)^2 + n_{\mathbf{z}}(x)\, x^2 \right) dx \quad (13)$$

The service order of the jobs is implicitly in the following:

- $u_{\mathbf{z}}(x)$ *= backlog due to jobs shorter than x in state* $\mathbf{z}$
- $n_{\mathbf{z}}(x)$ *= number of jobs longer than x in state* $\mathbf{z}$

M/G/1-SPTP

Proposition 13 (*Hyytiä et al. (2012a)*)

The size-aware value function of an M/G/1-SPTP queue w.r.t. arbitrary holding costs satisfies

$$v_{\mathbf{z}} - v_0 = \sum_{i=1}^{n} b_i \left(\frac{1}{1-\rho(\tilde{\Delta}_i)} \left(\sum_{j=1}^{i-1} \Delta_j \right) + \frac{2}{\Delta_i^*} \int_0^{\tilde{\Delta}_i} \frac{x\, dx}{1-\rho(x)} \right) + \frac{\lambda}{2} \sum_{i=0}^{n} \left[\left(\sum_{j=1}^{i} \Delta_j \right)^2 \int_{\tilde{\Delta}_i}^{\tilde{\Delta}_{i+1}} h(x)\, dx + \left(\sum_{j=i+1}^{n} (\Delta_j^*)^{-2} \right) \int_{\tilde{\Delta}_i}^{\tilde{\Delta}_{i+1}} x^4\, h(x)\, dx \right]$$

- *Job 1 receives service and $\sqrt{\Delta_1 \Delta_1^*} < \ldots < \sqrt{\Delta_n \Delta_n^*}$ (SPTP order)*
- $\tilde{\Delta}_i = \begin{cases} 0, & i = 0 \\ \sqrt{\Delta_i \Delta_i^*}, & i = 1, \ldots, n \\ \infty, & i = n+1 \end{cases}$

M/G/1-SPTP - Alternative Form

Proposition 14 (*Hyytiä et al. (2012a)*)

The size-aware value function of M/G/1-SPTP w.r.t. arbitrary holding costs satisfies

$$v_{\mathbf{z}} - v_0 = \sum_{i=1}^{n} b_i \left(\frac{\tilde{u}_{\mathbf{z}}(\tilde{\Delta}_i)}{1-\rho(\tilde{\Delta}_i)} + \frac{2}{\Delta_i^*} \int_0^{\tilde{\Delta}_i} \frac{x\, dx}{1-\rho(x)} \right) + \frac{\lambda}{2} \int_0^{\infty} h(x) \left(\tilde{u}_{\mathbf{z}}(x)^2 + g_{\mathbf{z}}(x)\, x^4 \right) dx$$

- $\tilde{\Delta}_i = \begin{cases} 0, & i = 0 \\ \sqrt{\Delta_i \Delta_i^*}, & i = 1, \ldots, n \\ \infty, & i = n+1 \end{cases}$
- $\tilde{u}_{\mathbf{z}}(x) = \sum_j \Delta_j \cdot \mathbf{1}(\tilde{\Delta}_j < x)$
- $g_{\mathbf{z}}(x) = \sum_j \frac{\mathbf{1}(\tilde{\Delta}_j > x)}{(\Delta_j^*)^2}$

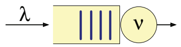
Computational Remarks

- Size-aware value functions for SPT, SRPT and SPTP appear first to be computationally difficult[18]
- However, all integrands are *independent of the state*
- Therefore it is possible to evaluate them in advance, and, e.g., tabulate the results and interpolate
- For example, for SPT in (11) we need determine offline

$$H(x) \triangleq \int_0^x h(t)\,dt$$
$$\rho(x) \triangleq \lambda \int_0^x t\,f(t)\,dt$$

[18] You do not want to evaluate integrals in on-line decision making

Value Function for M/G/1 Queues



A. Elementary scheduling disciplines:
- M/G/1-FCFS
- M/G/1-LCFS

B. Size-aware scheduling disciplines:
- Size-aware scheduling
- M/G/1-SPT (shortest-processing-time)
- M/G/1-SRPT (shortert-remaining-processing-time)
- M/G/1-SPTP (shortest-processing-time-product)

C. Processor sharing (PS)
- M/D/1-PS (fixed job sizes)
- M/M/1-PS

M/G/1-PS: (Processor sharing)

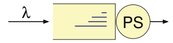


Figure 9: M/G/1 with Processor Sharing (PS)

Basics:
- PS serves the existing n jobs at equal rates $1/n$
- Mean delay in M/G/1-PS is insensitive to job size distribution

$$\mathrm{E}[T] = \frac{\mathrm{E}[X]}{1-\rho}$$

- State $\mathbf{z} = (\Delta_1; ..; \Delta_n)$ defines the remaining service times
- Ordering: $\Delta_1 \geq \ldots \geq \Delta_n$ (job n will depart first)

M/G/1-PS: Total delay

Lemma 1 (Total delay without new arrivals under PS)

Let $\Delta_1 \geq \ldots \geq \Delta_n$ denote the remaining service times. Then, the total delay in a PS queue assuming no new jobs arrive is

$$V_{\mathbf{z}} = \sum_{i=1}^{n} (2i-1)\Delta_i \qquad (14)$$

Proof.

Job n leaves the system first and job 1 last, and

$$V_{\mathbf{z}} = \Delta_n n^2 + (\Delta_{n-1} - \Delta_n)(n-1)^2 + \ldots + (\Delta_1 - \Delta_2)$$
$$= \sum_{i=1}^{n} (2i-1)\Delta_i$$

□

M/D/1-PS

Proposition 15 (*Hyytiä et al. (2011a)*)

The value function of a size-aware M/D/1-PS queue in state $\mathbf{z}$ *w.r.t. delay satisfies*

$$v_{(\Delta_1;..;\Delta_n)} - v_0 = \frac{\lambda}{1-\rho} u_{\mathbf{z}}^2 - u_{\mathbf{z}} + 2\sum_{i=1}^{n} i\,\Delta_i \qquad (15)$$

Note:
- Compact form as a new job will always depart last
- Converges to (14) when $\lambda \to 0$
- Generalization to arbitrary holding costs straightforward
- Admission cost $c_{\mathbf{z}} = v_{(d;\Delta_1;..;\Delta_n)} - v_{(\Delta_1;..;\Delta_n)}$ is

$$c_{\mathbf{z}} = \frac{2u_{\mathbf{z}} + d}{1-\rho}$$

M/M/1-PS

Proposition 16 (*Hyytiä et al. (2011b)*)

The value function of a size-aware M/M/1-PS queue in state $(m; \Delta_1, \ldots, \Delta_n)$ *satisfies*

$$v_{(m;\Delta_1,\ldots,\Delta_n)} = v_m + \frac{1}{(1-\rho)^2}\sum_{k=1}^{n}(2k-1)\Delta_k + \frac{2-\rho}{\mu(1-\rho)^2}\sum_{k=1}^{n}\left(m - \frac{k\rho}{1-\rho}\right)\left(\sum_{i=1}^{k} e^{-\mu(1-\rho)(\Delta_i-\Delta_k)}\right)\left(1-e^{-\mu(1-\rho)(\Delta_k-\Delta_{k+1})}\right)$$

- *Δ_i are n known remaining service times, $\Delta_1 > \ldots > \Delta_n$*
- $\Delta_{n+1} \triangleq 0$
- *m jobs have unknown Exp(μ) distributed service time*

M/G/1-PS: Insensitivity

Remark:

- Mean delay was insensitive to job size distribution
  - depends only on the mean E[X] and λ
- Value functions for M/D/1-PS and M/M/1-PS are different . . .

Corollary 2 (Insensitivity of M/G/1-PS)

The size-aware value function for M/G/1-PS **is not** *insensitive to job size distribution*

References

1 Hyytiä, Penttinen and Aalto, *Size- and State-Aware Dispatching Problem with Queue-Specific Job Sizes*, European Journal of Operational Research, 2012.

2 Hyytiä, Aalto, Penttinen, *Minimizing Slowdown in Heterogeneous Size-Aware Dispatching Systems*, ACM SIGMETRICS/Performance 2012.

3 Hyytiä, Aalto, Penttinen and Virtamo, *On the value function of the M/G/1 FCFS and LCFS queues*, Journal of Applied Probability, 2012.

4 Hyytiä, Virtamo, Aalto and Penttinen, *M/M/1-PS Queue and Size-Aware Task Assignment*, Performance Evaluation 2011 (IFIP PERFORMANCE'11).

5 Hyytiä, Penttinen, Aalto and Virtamo, *Dispatching problem with fixed size jobs and processor sharing discipline*, in ITC'23, 2011.

References to Value functions

Queueing systems

M/M/s	Krishnan, CDC (1987)
M/M/1	Aalto&Virtamo, NTS-13 (1996)
M/M/1-PS holding costs	Doroudi et al., Performance (2014)
M/G/1-FCFS	Sassen et al., Neerlandica (1997)
M/M/1 & M/M/1/N	Koole, CDC (1998)
M/Cox(r)/1	Bhulai, JAP (2006)

Size-aware queueing systems

M/G/1 FCFS/LCFS/SRPT	Hyytiä et al., EJOR (2012)
M/G/1 class-aware	Hyytiä et al., JAP (2012)
M/G/1 holding costs, SPTP	Hyytiä et al., Sigmetrics (2012)
M/D/1-PS	Hyytiä et al., ITC (2011)
M/M/1-PS	Hyytiä et al., Performance (2011)
Erl/G/1-FCFS	Hyytiä & Aalto, ValueTools (2013)

References to Value functions (cont.)

Setup delay and energy

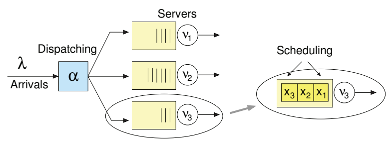
M/G/1 (w.r.t. energy)	Penttinen et al., IPCCC (2011)
M/G/1 (FCFS, setup)	Hyytiä et al., PEVA (2014)
M/G/1 (LCFS, setup)	Hyytiä et al., ITC (2014)
M/D/1 (PS, setup)	-"-

Loss systems

M/M/s/s	Krishnan, CDC (1986)
M/M/s/k	Leeuwaarden et al. (2001)

5. Size-aware Dispatching

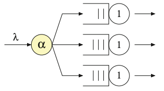
Task Assignment Problem



Task assignment (dispatching):

Route job to one of the m servers upon arrival

Size-Aware Dispatching Problem



Standard:
- Poisson arrival process, rate λ
- m parallel heterogeneous servers
- Scheduling discipline known (FCFS, LCFS, ...)
- Dispatching policy α chooses the queue upon arrival
- Objective: minimize the mean delay

Size-aware setting:
- General job size distribution
- Job sizes become known upon arrival
- Queue states (job sizes and their service order) are known

Generalization: server-specific service times (per job)

Size-Aware Dispatching Problem (cont.)

Holding costs:
- Arriving jobs have also a holding cost,

$$(X^{(1)}, B^{(1)}),\ (X^{(2)}, B^{(2)}),\ \ldots$$

$X^{(i)}$ =	Size of Job i	[bit]
$B^{(i)}$ =	Holding cost rate of Job i	[1/s]

- Job i incurs costs at rate $B^{(i)}$ until it departs

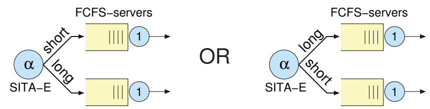
Objective: $\min\ \mathrm{E}[T \cdot B]$

Examples:

Latency (delay):	$B^{(i)} = 1$
Slowdown:	$B^{(i)} = 1/X^{(i)}$
Priorities:	$B^{(i)} =$ priority of Job i

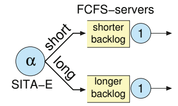
SITA with Switch

- Consider the mean delay with SITA-E and identical servers
- Roles of the servers can be exchanged anytime



- With value functions: $v_1(\mathbf{z}_1)+v_2(\mathbf{z}_2) < v_1(\mathbf{z}_2)+v_2(\mathbf{z}_1)$?
- Considering states after an arrival gives new policy:

SITA-E with Switch (SITA-Es):
"Short jobs to a shorter queue"

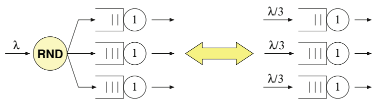


- Generalizes to $n > 2$ queues

Decomposition to M/G/1 Queues

- Deriving a value function for the whole system is difficult (e.g., for JSQ)
- Any static policy feeds servers according to a Poisson process

Decomposition by any static policy



- Static policy thus defines for each server i
  - Poisson arrival rate λ_i
  - Job size distribution X_i
  - Holding cost distribution B_i

which enables the analysis of the whole system

First policy iteration (FPI)

FPI

- *Assume a static basic policy α_0:*
  - *Defines arrival process (λ_i, X_i, B_i) for each queue*
- *Derive value functions $v_{\mathbf{z}_i}$ for the "isolated queues", and*

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

- *Carry out the FPI step*

$$\alpha(\mathbf{z}, x, b) \triangleq \operatorname*{argmin}_i \left(v_{\mathbf{z}'_i} - v_{\mathbf{z}_i}\right)$$

where $\mathbf{z}'_i$ is the new state of queue i with job (x, b) added

Note: FPI on static α_0 yields an *index policy*

First policy iteration (FPI)

For M/G/1-FCFS the costs can defined in two ways:

1. Costs are incurred at rate b during the sojourn time t
2. Job pays an immediate cost d upon arrival, $d = b \cdot t$

With Immediate Costs:
Backlog u is sufficient state information and (8) reduces to

$$v_u - v_0 = \frac{\lambda\, u^2}{2(1-\rho)} \mathrm{E}[B] \qquad (16)$$

Action *"Assign job (x, b) to queue i"*
- Immediate cost $\quad d_i = (u_i + x/\nu_i) b$
- New state $\quad u_i^* = u_i + x/\nu_i$

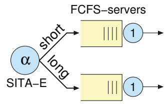
FPI policy: $\alpha(\mathbf{z}, x, b) = \operatorname*{argmin}_i\ d_i + (v^{(i)}_{u_i^*} - v^{(i)}_{u_i})$

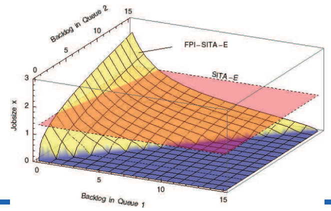
FPI-SITA-E, "Dynamic SITA-E"

- SITA-E is static ⇒ value function available ⇒ FPI-SITA-E

$$\alpha(\mathbf{z}, x) = \underset{i}{\text{argmin}} \, \frac{\lambda_i}{2(1-\rho)}(2u_i x + x^2) + u_i + x$$

  - λ_i is the arrival rate to queue i (according to SITA-E)
  - u_i is the current backlog in queue i
- Threshold with FPI-SITA-E depends on the backlogs





Numerical Examples

For Delay:

1. Two identical FCFS servers
2. Two identical SRPT servers
3. Heterogeneous PS servers

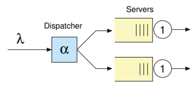
For Slowdown:

4. Three heterogeneous servers

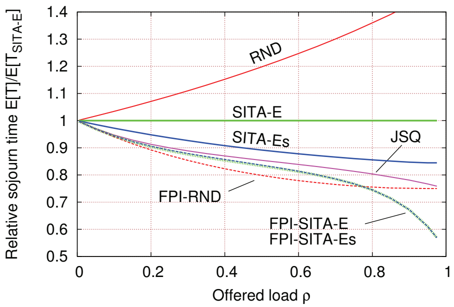
Example 1: FCFS

Example 1:

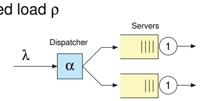
- Two identical queues with FCFS
- Job size distribution:
  1. Exponential Exp(1)
  2. Pareto(β) with $\beta = 3$: $\quad P\{X > t\} = (1+t)^{-\beta}$
- Performance metric: Relative delay to SITA-E



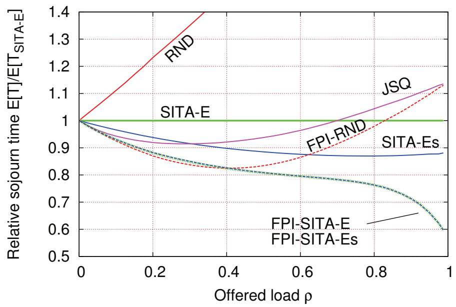
Example 1: FCFS (cont.)



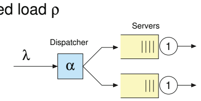
- Two identical FCFS servers
- $X \sim \text{Exp}(1)$



Example 1: FCFS (cont.)



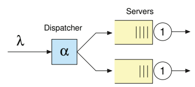
- Two identical FCFS servers
- $X \sim \text{Pareto}(1)$



Example 2: SRPT

Example 2:

- Two identical queues with SRPT
- Exponential job size distribution, Exp(1)
- Relative delay when compared to a single shared SRPT queue processed by two identical servers



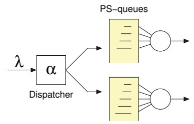
Example 2: SRPT (cont.)

- Dispatching system vs. a shared queue with SRPT (M/M/2-SRPT).
- Disadvantage due to the dispatching can be insignificant (here order of 5% with FPI-RND).

Example 3: PS Servers

Example 3:

- Poisson arrival process
- Heterogeneous PS servers
- Fixed server-specific service time $d_i = d/\nu_i$

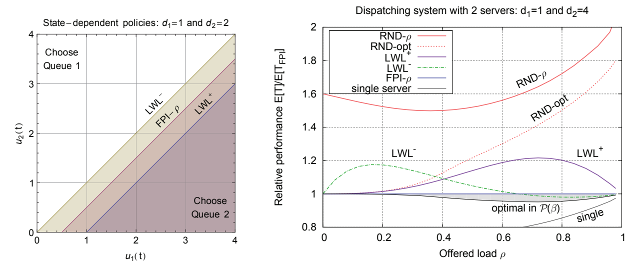


Dispatching policies:

Random Bernoulli split	RND-ρ and RND-opt	
Least-work-left (pre-assignment)	LWL$^-$:	$\underset{i}{\text{argmin}}\ u_i$
Least-work-left (post-assignment)	LWL$^+$:	$\underset{i}{\text{argmin}}\ u_i + d_i$
FPI for RND-ρ	FPI:	$\underset{i}{\text{argmin}}\ u_i + (1/2)d_i$

$\Rightarrow$ Policy family $\mathcal{P}(\beta)$ with $c_i = u_i + \beta d_i$

Example 3: PS Servers (cont.)



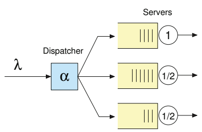
LWL$^-$, LWL$^+$ and FPI-RND illustrated for $d_1 = 1$ and $d_2 = 2$

Mean delay relative to FPI-RND: $d_1 = 1$ and $d_2 = 4$.

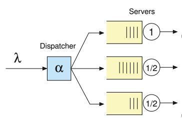
(ν_1=1 and ν_2=0.25, and single PS server has ν=1.25)

Example 4: Slowdown

- Three heterogeneous servers:
  1. Service rate $\nu_1 = 1$
  2. Service rate $\nu_2 = 1/2$
  3. Service rate $\nu_3 = 1/2$
- Bounded Pareto distributed service times
- Slowdown metric $\gamma = \frac{T}{X}$
- Scheduling disciplines: FCFS, LCFS and SPTP
- Comparison of JSQ to FPI (based on RND-opt)

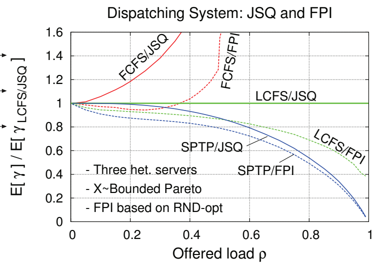


Example 4: Slowdown (cont.)



Slowdown:

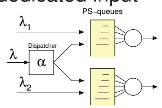
$$\gamma = \frac{T}{X}$$



- Bounded Pareto distributed service time
- Scheduling discipline: FCFS, LCFS and SPTP

Versatile Approach

1. Each server can have dedicated input



2. Basic policy can be class-specific
   - Low and high priority customers with own queues
   - When to route a low priority job to a high priority queue?
3. Service times can be server-specific
   - General purpose vs. specialized servers

Summary

- Size- and state-aware dispatching problem can be approached in the MDP framework
- Value functions $v_{\mathbf{z}}$ are required for the FPI step
- M/G/1 results sufficient for static basic policies:
  - FCFS and LCFS: $v_{\mathbf{z}}$ is insensitive to job size distribution
  - SPT, SRPT and SPTP: $v_{\mathbf{z}}$ is an integral expression
  - PS: harder to analyze (M/D/1-PS and M/M/1-PS)
- Efficient dispatching policies that take into account
  - Cost structure
  - Existing and later arriving tasks

References

1 Hyytiä, Penttinen, Aalto, *Size- and State-Aware Dispatching Problem with Queue-Specific Job Sizes*, EJOR 217(2), 2012.

2 Hyytiä, Aalto, Penttinen, *Minimizing Slowdown in Heterogeneous Size-Aware Dispatching Systems*, ACM SIGMETRICS/Performance 2012.

3 Hyytiä, Virtamo, Aalto, Penttinen, *M/M/1-PS Queue and Size-Aware Task Assignment*, Performance Evaluation 68(11), 2011 (Performance'11).

4 Hyytiä, Penttinen, Aalto and Virtamo, *Dispatching problem with fixed size jobs and processor sharing discipline*, ITC'23, 2011.

5 Hyytiä, Aalto, Penttinen and Virtamo, *On the value function of the M/G/1 FCFS and LCFS queues*, Journal of Applied Probability, 2012.

6. Lookahead approach

FPI in practice

- Value function for dynamic α_0 not available[19]
- For static α_0, system decomposes

$$v_{\mathbf{z}} = \sum_{j=1}^{k} v_{\mathbf{z}}^{(j)}$$

where $v_{\mathbf{z}}^{(j)}$ is the value function of queue j.

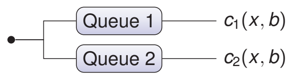
For example, for an M/G/1-FCFS queue j

$$v_{\mathbf{z}}^{(j)} - v_0 = \frac{\lambda_j \, \mathrm{E}[B_j]}{2(1-\rho_j)} (u_j)^2$$

- B_j = the mean holding cost of jobs α_0 assigns to queue j
- ρ_j = the offered load at queue j with α_0
- u_j = the current backlog in queue j

[19] The value function exists, but it is very difficult to compute.

Decision tree of FPI



Decision tree corresponding to FPI:

- New job (x, b) has arrived
- Deviate from α_0 for **one action**
- Later actions by α_0
- Terminal cost $c_i(x, b)$ according to α_0 (from value function)

$$c_i(x,b) = \left(u_i' + \frac{x}{\nu_i}\right) b + \frac{\lambda_i \, \mathrm{E}[B_i]}{2(1-\rho_i)} \left(2u_i' + \frac{x}{\nu_i}\right) \frac{x}{\nu_i}$$

where u_i' is the backlog in queue i before the arrival.

Observations of FPI

- FPI reverts back to (simple) static α_0 immediately

  "Queues are separated"
- The queues evaluate the admission cost independently;

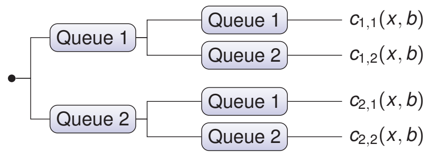
  $\Rightarrow$ FPI gives us an *index policy*!
- What if assigning job j to queue 1 means that the next job should really go to queue 2?

  Anything better than FPI?
- Idea: What if we (tentatively) fix also the next action(s)?

  "Queue 1 earns a short break in arrivals"

$\Rightarrow$ **Lookahead approach!**

Lookahead: Static second action



Decision tree for a static lookahead:

- New job (x, b) has arrived
- Deviate from α_0 for **two actions**, (i, j)
- Size, holding cost and arrival time of the next job unknown
- Terminal costs $c_{i,j}(x, b)$ by conditioning

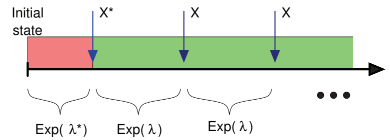
Note:

- Lookahead gives us a dynamic policy
- Evaluation involves the state of the whole system
  - $\Rightarrow$ Not an *index policy*!

Lookahead Value $L(\mathbf{z}, \lambda^*, X^*)$

Terminal costs $c_{i,j}$ can be computed from **lookahead values**:

- Let $v_{\mathbf{z}}$ denote the value function of a queue with the usual Poisson arrival process (λ, X)
- However, suppose that the **next job arrives differently**:
  - Arrival time $\tau \sim \text{Exp}(\lambda^*)$
  - Job size X^*
- After that jobs arrive as usual according to (λ, X)



The **lookahead value**, $L(\mathbf{z}, \lambda^*, X^*)$, is the expected cumulative difference in costs between the above system and the mean cost rate.

Lookahead Value $L(\mathbf{z}, \lambda^*, X^*)$

Definition 3 (Lookahead Value for M/G/1)

The lookahead value for state $\mathbf{z}$, denoted by $L(\mathbf{z}, \lambda^*, X^*)$ is the expected cost the queue incurs in comparison to mean cost rate when the next job with size X^* will arrive after time $\tau \sim \text{Exp}(\lambda^*)$, after which jobs arrive according to (λ, X)

$$L(\mathbf{z}, \lambda^*, X^*) \triangleq \text{E}[V_{\mathbf{z}}(\tau) - r \cdot \tau + v_{\mathbf{z}^* \oplus X^*}] - v_0,$$

where $V_z(\tau)$ denotes the costs incurred during time τ and $\mathbf{Z}^* \oplus X^*$ is the state with the next job X^* assigned

Convention: $X^* = 0$ means that the next job is assigned elsewhere

Remarks:

- By definition, $L(\mathbf{z}, \lambda, X) = v_{\mathbf{z}}$
- As with value functions, the constant offset is immaterial

Lookahead Value (cont.)

Terminal costs:

- Let $L_j(\cdot)$ denote the lookahead value of queue j with (λ_j, X_j) according to a static basic policy α
- Let (λ, X) denote the global arrival rate and job size, i.e.,

$$\lambda = \lambda_1 + \ldots + \lambda_n$$

- For assigning both the new and the next job to queue i

$$c_{i,i} = L_i(z_i \oplus x, \lambda, X) + \sum_{k \neq i} L_k(z_k, \lambda, 0)$$

  where $z_i \oplus x$ denotes the state of queue i with a new job x

- For assigning the new job to queue i and next to queue j $(i \neq j)$

$$c_{i,j} = L_i(z_i \oplus x, \lambda, 0) + L_j(z_j, \lambda, X) + \sum_{k \notin \{i,j\}} L_k(z_k, \lambda, 0)$$

Static Lookahead Action: FCFS

For M/G/1-FCFS w.r.t. delay:

Here it is convenient to use immediate costs upon arrival, for which the value function is

$$v_u - v_0 = \frac{\lambda u^2}{2(1-\rho)}$$

For the lookahead value we get

$$L(\mathbf{z}, \lambda^*, X^*) = \overbrace{(0 - r)\text{E}[\tau]}^{\text{until the next arrival}} + \overbrace{\text{E}[U_\tau + X^*]}^{\text{immediate cost}} + \overbrace{\text{E}[v_{U_\tau + X^*}] - v_0}^{\text{future costs after time } \tau}$$

which reduces to

$$L(\mathbf{z}, \lambda^*, X^*) = -\frac{\lambda}{\lambda^*}\left(\frac{\lambda\,\text{E}[X^2]}{2(1-\rho)} + \text{E}[X]\right) + \text{E}[U_\tau] + \text{E}[X^*] + \frac{\lambda \text{E}[(U_\tau + X^*)^2]}{2(1-\rho)}$$

Both $\text{E}[U_\tau]$ and $\text{E}[(U_\tau)^2]$ can be computed easily, and after some manipulation ...

Static Lookahead Action: FCFS

Theorem 4 (Lookahead for FCFS w.r.t. delay)

Lookahead admission cost to M/G/1-FCFS w.r.t. delay is

$$\begin{aligned} c_{i,j}(x) &= u_i + g_i \frac{x}{\nu_i}\left(2u_i - \frac{x}{\nu_i}\right) + \frac{2}{\lambda^2}\sum_k g_k(1 - \lambda u_k - e^{-\lambda u_k}) - \text{E}[T] \\ &+ \left(1 + \frac{2g_j \text{E}[X]}{\nu_j}\right)\left(u_j - \frac{1 - e^{-\lambda u_j}}{\lambda}\right) + g_j \frac{\text{E}[X^2]}{\nu_j^2} + \frac{\text{E}[X]}{\nu_j} \end{aligned}$$

where u_k are the backlogs **with the new job included in queue i**, *and g_k are the queue-specific constants*

$$g_k = \frac{\lambda_k}{2(1-\rho_k)}$$

Proof.

See Hyytiä (2013) □

Static Lookahead Action: FCFS

Theorem 5 (Lookahead for FCFS w.r.t. holding costs)

Lookahead admission cost to M/G/1-FCFS with holding costs is

$$c_{i,j}(x,b) = u_i b + g_i \frac{x}{\nu_i}\left(2u_i - \frac{x}{\nu_i}\right) + \frac{2}{\lambda^2}\sum_k g_k(1-\lambda u_k - e^{-\lambda u_k}) - \mathrm{E}[TB]$$
$$+ \left(\mathrm{E}[B] + \frac{2g_j\mathrm{E}[X]}{\nu_j}\right)\left(u_j - \frac{1-e^{-\lambda u_j}}{\lambda}\right) + g_j\frac{\mathrm{E}[X^2]}{\nu_j^2} + \frac{\mathrm{E}[XB]}{\nu_j}$$

where u_k are the backlogs **with the new job included in queue i**, *and g_k is a queue-specific constant*

$$g_k = \frac{\lambda_k\,\mathrm{E}[B_k]}{2(1-\rho_k)}$$

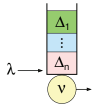
Proof.

See Hyytiä (2013) □

Static Lookahead Action: LCFS

Size-aware M/G/1-LCFS:

State $\mathbf{z} = (\Delta_1; ..; \Delta_n)$



Theorem 6

The lookahead value for M/G/1-LCFS w.r.t. delay is

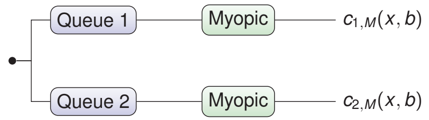
$$L(\mathbf{z}, \lambda^*, X^*) = \sum_{i=1}^{n} \frac{y_i}{1-\rho} + \frac{\left(n+1-\sum_{i=1}^{n} e^{-\lambda^* y_i}\right)(\rho^*-\rho)}{\lambda^*(1-\rho)}$$

where $y_i = \Delta_i + \ldots + \Delta_n$ and $\rho^* = \lambda^*\,\mathrm{E}[X^*]$

Proof.

See Hyytiä et al. (2014a). □

Dynamic Lookahead Action



Decision tree with dynamic second action:

- Consider all possible first actions *i*
- Second action according to a dynamic policy (e.g., Myopic)
  - Depends on the job and its arrival time
- Lookahead costs $c_{i,M}(x,b)$ by conditioning
  - Long expressions, numerical evaluation straightforward
  - Proposition 4 in Hyytiä (2013)

Summary of the Lookahead

- Upon an arrival, consider both
  1. the current job
  2. the later arriving jobs (tentatively)
- Terminal costs
  - Condition on different sample paths
  - "Tail" using a value function with a static policy
- *Deeper inspection*
  - Better evaluation of the **System's state**
  - More accurate admission costs
- Second action by a **dynamic** α_0
  - Myopic
  - LWL, ...

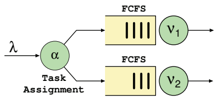
  gives an estimate for the corresponding value function

References

1. Hyytiä, *Lookahead actions in dispatching to parallel queues*, Performance Evaluation, 70(10), 2013.
2. Hyytiä, Righter and Aalto, *Task Assignment in a Heterogeneous Server Farm with Switching Delays and General Energy-Aware Cost Structure*, Performance Evaluation (2014).
3. Hyytiä, Righter and Aalto, *Energy-aware job assignment in server farms with setup delays under LCFS and PS*, ITC'26, 2014.

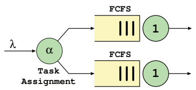
6.3 Numerical examples

Dispatching policies

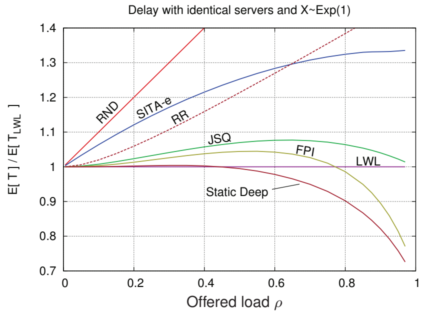


RND	Random split
RR	Round-robin
SITA-e	Size-Interval-Task-Assignment, equal loads
JSQ	Join-the-Shortest-Queue; the least number of jobs
LWL	Least-Work-Left, i.e., the shortest backlog
Myopic	Minimize the cost (delay) of the new job
FPI	FPI based on SITA-e
Deep	Lookahead strategy with α_0=SITA-e

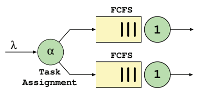
Example #1: Exp-jobs with FCFS Servers



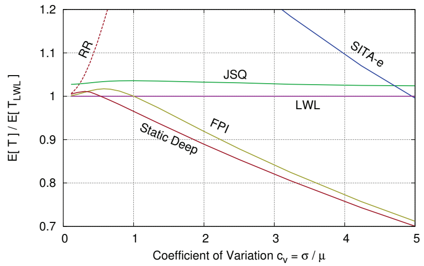
- Two identical servers
- FCFS
- $X \sim \text{Exp}(1)$
- Minimize latency
- Offered load ρ varied
- (Deep = Lookahead)



Example #2: Pareto Jobs with FCFS Servers



- Two identical servers
- FCFS
- $X \sim \text{Pareto}(k, 10^5, \alpha)$
- Minimize latency
- Set $\rho = 0.8$, vary $c_V = \sigma/\mu$
- (Deep = Lookahead)

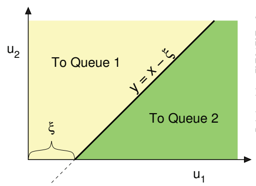


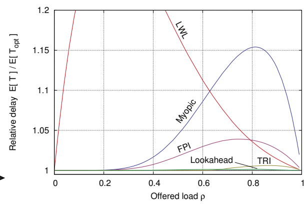
$\rho = 0.8$

Example #3: Fixed-size Jobs with FCFS

Two heterogeneous FCFS servers with fixed size jobs:

- Fast server with fixed service time $1/4$
- Slower server with fixed service time of 1





- Left: Switching curve defines the optimal policy (Hyytiä, 2014)
- Right: Gap to the optimal policy as a function of ρ
- Lookahead is practically optimal in this case!

Example #4: Two Identical LCFS Servers

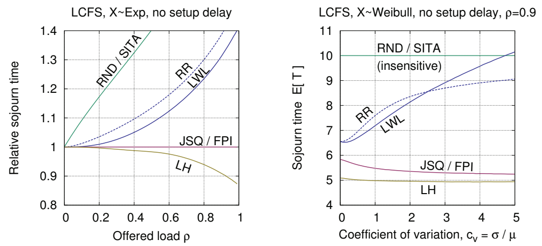


Figure 10: Left: Exponentially distributed jobs, ρ varied
Right: Weibull distributed jobs with a fixed load $\rho = 0.9$

Remark: FPI for RND yields JSQ, which ignores the service times
Lookahead does clearly a better job also here and is nearly insensitive

Summary

- FPI offers robust & adaptive cost-aware policies
- Lookahead approach builds on that:

  Consider also the next arriving job(s)!
- Explicit expressions derived for admission costs
  - Estimate for value function with dynamic policy
- Numerically, clear improvement from FPI and others
- Near-optimal? (Sometimes at least! See Example #3)

7. Energy-aware Systems

Energy-aware System Model

- Until now focus solely on performance (i.e. delay)
- Recently energy consumption has become an important design factor
- In computing, two approaches to save on energy
  1. **Speed scaling**: speed (and energy consumption) of processors can be adjusted
  2. **Switching off** currently unnecessary devices
- We focus on the latter, i.e., switching off servers

Penalty for switching off comes in the form of setup delay:

- Jobs have to wait time s before the service can begin

Related work

Related models (M/G/1):

- Removable servers, N-policy
  - (Yadin and Naor, 1963; Heyman, 1968)
  - Service starts when nth customer arrives
- Vacation models, T-policy
  - (Levy and Yechiali, 1975; Heyman, 1977)
  - Server returns periodically to check the queue
- D-policy, service starts when backlog exceeds d

Results for setup delay:

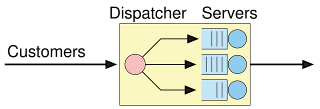
- M/G/1 with setup times (Welch, 1964)
- M/M/k approximations (Gandhi et al., 2010)
- M/M/k exact results (Gandhi et al., 2013)

No delay- and energy-*savvy* dispatching policies!

Setup Delays and Energy

Model for Server Farm

- k parallel servers
- Size-aware setting



Distinctive features here

- Energy- and Delay-aware cost structure
  - Running costs (per unit time)
  - Holding costs (per job)
    - e.g., delay (sojourn time)
- Idle servers can be switched OFF to save energy
- Setup delay postpones the start of the service

Basic Cost Structure

Basic energy-aware cost structure: [20]

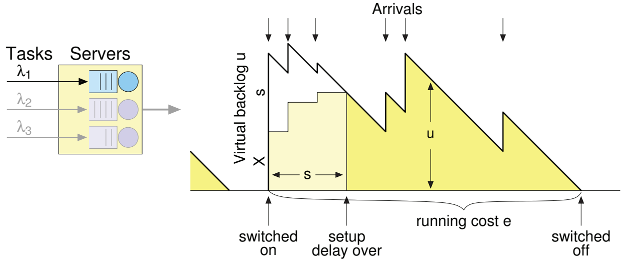
(i) **Running costs e**
  - Costs are incurred at rate e when the server is on

(ii) **Holding cost b_i for job i**
  - Job i incurs costs at rate b_i until it departs

The first is the system's cost to provide the service, and the second is the quality of service (QoS) as seen by the customers

[20] See Penttinen et al. (2011) and (Hyytiä et al., 2014a,b)

Sample Busy Period



The basic cost structure:

(i) **Running cost e** when the server is ON

(ii) **Holding cost b_i per job i** until departure (not shown, depends on scheduling)

7.2 Setup Delays in M/G/1

Mean Delay with Setup Delay

Theorem 7 (M/G/1-FCFS)

The mean delay in an M/G/1 with FCFS is

$$E[T] = \frac{\lambda E[X^2]}{2(1-\rho)} + E[X] + \frac{E[S] + (\lambda/2) E[S^2]}{1 + \lambda E[S]} \qquad (17)$$

Theorem 8 (M/G/1-LCFS)

The mean delay in an M/G/1 with preemptive LCFS is

$$E[T] = \frac{E[X]}{1-\rho} + \frac{E[S] + (\lambda/2) E[S^2]}{1 + \lambda E[S]} \qquad (18)$$

- Mean results decompose
- The extra delay term due to setup is the same for FCFS and LCFS!

Mean Delay with Setup Delay (cont.)

Theorem 9 (M/D/1-PS)

The mean delay in an M/D/1 with PS is

$$E[T] = \frac{d}{1-\rho} + (1+\rho)\frac{E[S] + (\lambda/2) E[S^2]}{1 + \lambda E[S]} \qquad (19)$$

Proof.

See Hyytiä et al. (2014a). □

Hence, the delay penalty for M/D/1-PS is $(1+\rho)p_S$, whereas with FCFS and LCFS we had only p_S, see (17) and (18).

Mean Delay with Setup Delay (cont.)

Corollary 10 (M/M/1)

The mean delay in M/M/1 with an arbitrary work-conserving scheduling discipline (e.g., FCFS, LCFS, PS) is

$$E[T] = \frac{1}{\mu - \lambda} + \frac{E[S] + (\lambda/2) E[S^2]}{1 + \lambda E[S]} \qquad (20)$$

Proof.

Substitute $X \sim \text{Exp}(\mu)$ into (17) or (18). □

Corollary 11 (Sensitivity (Hyytiä et al., 2014a)**)**

M/G/1-PS with setup delay **is not** *insensitive to job size distribution.*

Separability

Theorem 12 (Separability (Hyytiä et al., 2014a)**)**

If the mean delay in a work-conserving service system with a Poisson arrival process is additively separable, $E[T] = g_X(\lambda) + p_S(\lambda)$, *then*

$$p_S(\lambda) = \frac{E[S] + \lambda E[S^2]/2}{1 + \lambda E[S]}$$

Proof.

If $E[T]$ separates, then it holds also for the trivial case $X = 0$. In such systems, the mean delay is the remaining setup delay, $g_X(\lambda) = 0$, and

$$p_S(\lambda) = \frac{1}{\lambda} \cdot \frac{E[S + \lambda S^2/2]}{1/\lambda + E[S]} = \frac{E[S] + \lambda E[S^2]/2}{1 + \lambda E[S]}$$

□

In contrast, the mean response time for PS is not separable.

Mean running costs in M/G/1

Theorem 13 (Mean running cost in M/G/1)

$$r_R = \frac{\lambda(E[X] + E[S])}{1 + \lambda E[S]} e \qquad (21)$$

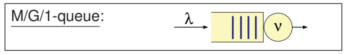
Proof.

The mean (remaining busy period) in M/G/1 is $b(u) = \frac{u}{1-\rho}$

The mean busy period with setup delay S is $E[B] = \frac{E[X]+E[S]}{1-\rho}$

The mean running cost is $r_R = \frac{E[B]}{E[B]+1/\lambda} \cdot e$, which yields (21) □

Delay and Running costs in M/G/1



- **Static switch-off policy**:
  1. `NeverOff`: keep the server always ON
  2. `InstantOff`: switch off immediately when idle
- **Mean running cost**:

$$r_R = \begin{cases} \dfrac{\lambda(\mathrm{E}[X]+\mathrm{E}[S])}{1+\lambda\,\mathrm{E}[S]}e, & \text{if InstantOff} \\ e, & \text{if NeverOff} \end{cases}$$

- **Mean delay cost**: (depends on scheduling)

$$r_T = \begin{cases} \lambda\,\mathrm{E}[T], & \text{if InstantOff} \\ \lambda\,\mathrm{E}[T \mid S \equiv 0], & \text{if NeverOff} \end{cases}$$

Example: Optimal Switch-off in M/G/1

The total cost rate under `InstantOff` in M/G/1-FCFS is

$$r_{\text{Instant}} = \overbrace{\frac{\lambda^2\,\mathrm{E}[X^2]}{2(1-\rho)} + \frac{\lambda(\mathrm{E}[S]+(\lambda/2)\,\mathrm{E}[S^2])}{1+\lambda\,\mathrm{E}[S]} + \lambda\,\mathrm{E}[X]}^{\text{Sojourn time}} + \overbrace{\frac{\lambda(\mathrm{E}[X]+s)}{1+\lambda s}e}^{\text{Running cost}}$$

and under `NeverOff`,

$$r_{\text{Never}} = \frac{\lambda^2\,\mathrm{E}[X^2]}{2(1-\rho)} + \lambda\,\mathrm{E}[X] + e$$

Studying $r_{\text{Instant}} < r_{\text{Never}}$ ⇒ `InstantOff` better if

$$\boxed{e > \frac{2\lambda\,\mathrm{E}[S] + \lambda^2\,\mathrm{E}[S^2]}{2(1-\rho)}} \qquad \left(e > \frac{\lambda s(2+\lambda s)}{2(1-\rho)}\right)$$

Note:
- Threshold depends on λ, $\mathrm{E}[X]$ and the first two moments of S
- It is the same also for LCFS and for all work-conserving M/M/1-queues
- With M/D/1-PS, the threshold gets multiplied by $(1+\rho)$

Aalto University School of Electrical Engineering
ITC-26 September, 2014, Karlskrona, Sweden E. Hyytiä

Summary

- Mean value results for M/G/1 queues often available
- Extra delay due to setup is the same for FCFS and LCFS
- In fact, this extra delay is (mean increase per job)

$$p_S(\lambda) = \frac{\mathrm{E}[S] + \lambda\,\mathrm{E}[S^2]/2}{1+\lambda\,\mathrm{E}[S]}$$

  for an arbitrary (incl. multi-server) system, if the mean delay is additively separable, $\mathrm{E}[T] = g_X(\lambda) + p_S(\lambda)$
  - This holds for FCFS and LCFS
  - … but not for PS
- Setup delay *breaks* the insensitivity property of PS

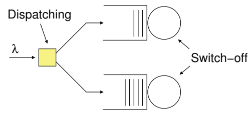
Aalto University School of Electrical Engineering
ITC-26 September, 2014, Karlskrona, Sweden E. Hyytiä

7.3 Static Dispatching

Aalto University School of Electrical Engineering
ITC-26 September, 2014, Karlskrona, Sweden E. Hyytiä

Model

System model:



- n identical parallel servers
- Jobs dispatched upon arrival
- Running costs at rate e (energy)
- Idle servers can be switched off
- Setup delay of s when switched on
- Objective:

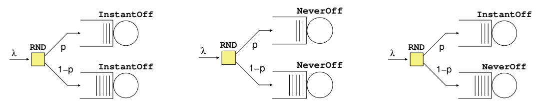
$$\boxed{\min \quad \mathrm{E}[N] + e\cdot\mathrm{E}[A]} \quad \text{or} \quad \boxed{\min \quad r_T + r_R}$$

where
- $\mathrm{E}[N]$ is the mean number in the system ($\mathrm{E}[N] = \lambda\,\mathrm{E}[T]$)
- $\mathrm{E}[A]$ is the mean number of running servers

Aalto University School of Electrical Engineering
ITC-26 September, 2014, Karlskrona, Sweden E. Hyytiä

Example: two servers

- Two identical servers:
  - Setup time $s = 2$
  - Running cost rate $e = 1$
- Poisson arrival process with rate λ
- Service times $X \sim \mathrm{Exp}(1)$ (and any work-conserving scheduling)
- RND dispatching (Bernoulli split) w.p. p
- Switch-off policies: `NeverOff` and `InstantOff`

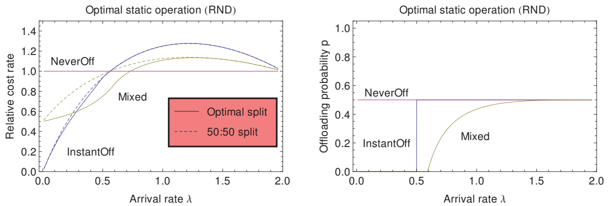


(1) `InstantOff` (2) `NeverOff` (3) `Mixed`

Aalto University School of Electrical Engineering
ITC-26 September, 2014, Karlskrona, Sweden E. Hyytiä

Static dispatching

Numerical Results



(a) Relative performance (b) Optimal split

Static dispatching

Results

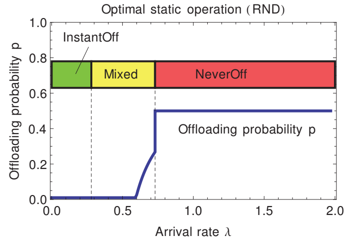


Figure 11: Optimal operation with RND.

Static Dispatching

Observations

- Optimal switch-off policy changes as the load increases
  `InstantOff` $\rightarrow$ `Mixed` $\rightarrow$ `NeverOff`
- `NeverOff` always splits the jobs uniformly
  - Running costs are fixed, $2 \times e$
  - Uniform split minimizes the mean sojourn time
- `InstantOff` and `Mixed` use
  - Only one server under a very low load
  - Uniform split under a very high load

All makes sense, but static control cannot be optimal?

7.4 Value Functions with Setup Delay

Value Function for FCFS with Setup Delay

Theorem 14 (M/G/1-FCFS)

The value function w.r.t. delay in an M/G/1-FCFS with setup delay s is

$$v_u - v_0 = \frac{\lambda u^2}{2(1-\rho)} - \frac{\lambda s(2+\lambda s)}{2(1-\rho)(1+\lambda s)} u \qquad (22)$$

Proof.

See Hyytiä et al. (2014b). □

Note: With immediate costs (or add the remaining sojourn times . . .)

Value Function for LCFS with Setup Delay

Theorem 15 (Value function)

The value function w.r.t. the response time in an M/G/1-LCFS with setup delay s is

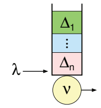
$$v_z - v_0 = \frac{n\delta + \sum_{i=1}^{n} i\Delta_i}{1-\rho} + \frac{\lambda\,\delta^2}{2(1-\rho)} - \frac{\lambda s(2+\lambda s)}{2(1-\rho)(1+\lambda s)} u \qquad (23)$$

Proof.

See Hyytiä et al. (2014a). □

Note:

- Job n is currently receiving service (head of queue)
- Linear "setup delay" term is the same as with FCFS



Value Function for M/D/1-PS

Theorem 16 (Value function)

The value function w.r.t. delay in M/D/1-PS with setup delay is

$$v_z - v_0 = q(z) - \frac{u}{1-\rho}\lambda\,\mathrm{E}[T],$$

where $u = \delta + \Delta_1 + \ldots + \Delta_n$, *and*

$$q(z) = \begin{cases} \frac{\rho(nd+\delta)}{(1-\rho)^2} + \frac{2n^2 d + (1+\rho)(2n+\lambda\delta)\delta}{2(1-\rho)}, & \delta > 0, \\ 2\sum_i i\Delta_i + \left(\lambda\frac{d+(1-\rho)u}{(1-\rho)^2} - 1\right)u, & \delta = 0 \end{cases}$$

where $\Delta_1 \geq \ldots \geq \Delta_n$ *is assumed.*

Value Function for Running Costs

Theorem 17 (Value function)

The value function w.r.t. the running costs in an M/G/1 with setup delay s is

$$v_R(u) - v_R(0) = \begin{cases} \frac{u}{1+\lambda s}e, & \text{if InstantOff} \\ 0, & \text{if NeverOff} \end{cases} \quad (24)$$

Proof.

See Hyytiä et al. (2014b) □

Note: Under `NeverOff`, all states are equal w.r.t. running costs

7.5 Dynamic Dispatching

Dynamic Dispatching

- Dynamic dispatching & switch-off decisions
  - Require state information
  - Can improve the performance
- cf. JSQ vs. RND
- Option to switch-off makes the situation more complicated
- We consider **size- and state-aware** setting

How to capitalize the state information?

Size-aware value functions for M/G/1

- Virtual backlog u includes the remaining setup time δ,

$$u = \delta + \Delta_1 + \ldots, \Delta_n.$$

- Value function w.r.t. running costs is [21]

$$v_R(u) - v_R(0) = \begin{cases} \frac{u}{1+\lambda s}e, & \text{if InstantOff} \\ 0, & \text{if NeverOff} \end{cases}$$

- Value function w.r.t. sojourn time in M/G/1-FCFS is

$$v_S(u) - v_S(0) = \begin{cases} \frac{\lambda}{2(1-\rho)}\left(u^2 - \frac{s(2+\lambda s)u}{1+\lambda s}\right) & \text{if InstantOff} \\ \frac{\lambda u^2}{2(1-\rho)} & \text{if NeverOff} \end{cases}$$

The immediate cost is equal to the resulting backlog u.

[21] See (Hyytiä et al., 2014b)

Improved dispatching

1. **First policy iteration**

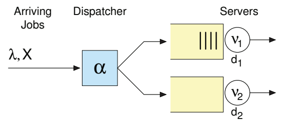
   (static policy) + (value function) $\overset{FPI}{\Rightarrow}$ new policy

   - Queues are evaluated assuming future jobs according to α_0

2. **Lookahead**
   - Evaluate decisions such as
     - This job to server i
     - Next job to server j (tentatively)
     - Later arriving jobs according to a static α_0
   - More accurate evaluation of each possible action
   - Yields typically a better policy than FPI

Example #1: Sending jobs to idle servers

Example: Routing to switched-off server

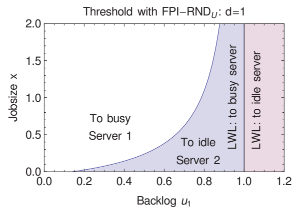


Two identical servers

	service rate	setup delay
Queue 1:	$\nu_1 = 1$,	$s_1 = 1$
Queue 2:	$\nu_2 = 1$,	$s_2 = 1$

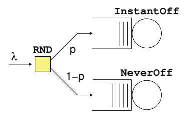
- $\lambda = 1.5$ and $E[X] = 1$
- Minimize waiting time W
- Basic policy α = RND
- Server 1 busy, $u_1 > 0$
  Server 2 idle, $u_2 = 0$

FPI sends a job to the idle Server 2 earlier than LWL



Example #2:

- Two servers
  - Server 1: `NeverOff`
    Server 2: `InstantOff`
  - Setup delay: $s = 2$
  - Running cost: $e = 1$



- Minimize $r_W + r_R$ (waiting time + running costs)
- Reference dispatching policies
  - RND: random 50:50 split
  - SITA-E: short jobs to server 1, long to server 2
  - Myopic: socially optimal if no later arrivals
  - Greedy: individually optimal choice (only delay)

Example #2: $X \sim \text{Exp}(1)$

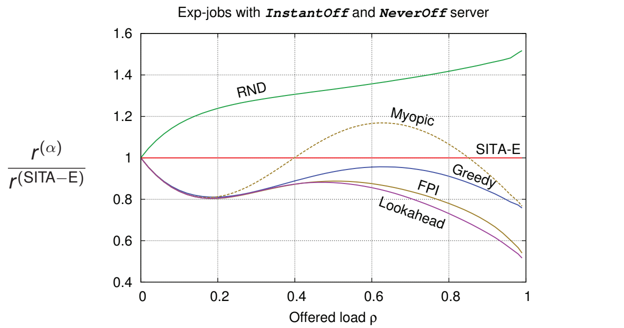


Figure 12: Relative mean cost rate with the objective of $r_W + r_R$.

Example #2: $X \sim$ Pareto (truncated)

Figure 13: Relative mean cost rate with the objective of $r_W + r_R$.

Example #3: Dispatching & Switching off

System

- 4 identical LCFS servers:
  - Service rate $\nu = 1$
  - Setup delay $s = 1$
  - Running cost $e = 1$
- Decision parameters:
  1. Dispatching decisions
  2. Switch-off policy: `InstantOff` or `NeverOff` (per server)
- Objective: $\min\ r_T + r_R$

Numerical evaluation

- We compute FPI and Lookahead policies
- . . . and compare them to RND, SITA-E, LWL and Myopic
- For each α and λ, we consider all $(\texttt{InstantOff}, \texttt{NeverOff})^4$ combinations, and choose the best among them

Example #: Results

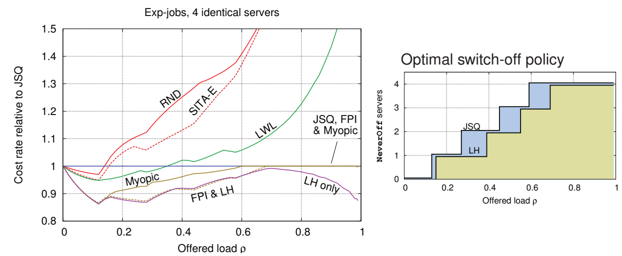


Figure 14: Performance with 4 servers when a tradeoff between the mean response time and energy consumption must be made.

The pool of always running servers increases as the load increases

Summary

- Server farm modelled as a queueing system
  - Job dispatching decisions
  - Server switch-off decisions to save energy
  - Setup delay included
- Cost structure
  1. Running costs [1/time]
  2. Delay costs T for FCFS, LCFS, and PS
- Static control straightforward
  - Mean results available
- Dynamic control is harder
  - Value functions available ⇒ FPI and Lookahead
  - Can be applied to both dispatching and switching off

References

1. Penttinen, Hyytiä and Aalto, *Energy-aware dispatching in parallel queues with on-off energy consumption*, IEEE IPCCC (2011).
2. Hyytiä, Righter and Aalto, *Task Assignment in a Heterogeneous Server Farm with Switching Delays and General Energy-Aware Cost Structure*, Performance Evaluation (2014).
3. Hyytiä, Righter and Aalto, *Energy-aware Job Assignment in Server Farms with Setup Delays under LCFS and PS*, ITC-26 (2014).

See also,

`http://www.netlab.hut.fi/u/esa/dispatching.html`

7.6 General Cost Structure

General Cost Structure

Consider the following queue-specific cost structure: [22]

(i) **Switching costs** (k_{on}, k_{off}) (per cycle)
- A cost of k_{on} when the server is switched on, and
- A cost of k_{off} when it is switched off

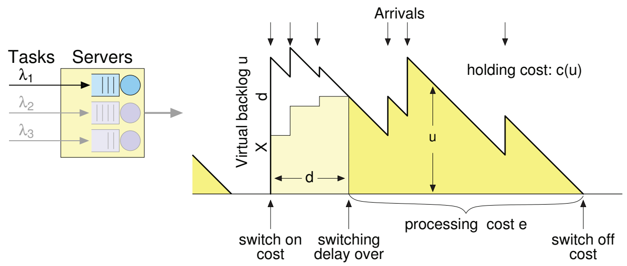
(ii) **Running costs** (e_{on}, e_{off}) (per unit time)
- Costs are incurred at rate e_{on} when the server is on
- and at rate e_{off} when it is switched off

(iii) **Holding cost** $c(u)$ (per unit time), u = virtual backlog
- Backlog based holding cost, some increasing function

Note: These costs are independent of the scheduling!

[22] See Heyman (1968) and Feinberg and Kella (2002) for normal M/G/1, Hyytiä et al. (2014b) for M/G/1 with setup delay.

General Cost Structure



The queue-specific cost structure:

(i) **switching costs** (k_{on}, k_{off}) (per cycle)

(ii) **running costs** (e_{on}, e_{off}) (per unit time)

(iii) **holding cost** $c(u)$ (per unit time), u = virtual backlog

General Cost Structure without Setup

Cost	Mean rate $\mathbf{r}_*$	Value function $\mathbf{v}_*(\mathbf{u}) - \mathbf{v}_*(\mathbf{0})$
Switching	$\lambda(1-\rho)\cdot k$	$-\lambda u \cdot k$
Running	$\rho \cdot e$	$u \cdot e$
Holding H_1	$\dfrac{\lambda \, \mathrm{E}[X^2]}{2(1-\rho)}$	$\dfrac{u^2}{2(1-\rho)}$

Holding cost H_k is a cost rate defined as $(U_t)^k$, $k = 1, 2, \ldots$

(i.e., a cost related to holding backlog in the system)

General Cost Structure with Setup Delay

Cost	Mean rate $\mathbf{r}_*$	Value function $\mathbf{v}_*(\mathbf{u}) - \mathbf{v}_*(\mathbf{0})$
Switching	$\dfrac{\lambda(1-\rho)}{1+\lambda s} \cdot k$	$-\dfrac{\lambda u}{1+\lambda s} \cdot k$
Running	$\dfrac{\rho+\lambda s}{1+\lambda s} \cdot e$	$\dfrac{u}{1+\lambda s} \cdot e$
Holding H_1	$\dfrac{\lambda\, \mathrm{E}[X^2]}{2(1-\rho)} + \dfrac{s(2\rho+\lambda s)}{2(1+\lambda s)}$	$\dfrac{u^2}{2(1-\rho)} - \dfrac{s(2\rho+\lambda s)\cdot u}{2(1-\rho)(1+\lambda s)}$

Note:

- State u = virtual backlog (incl. remaining setup time)
- Holding cost with $s = 0$ is the Pollazcek-Khinchine formula
- Setup delay shows up as an extra term in r_{H1} and $v_{H1}(u)$
  - Extra cost in $v_{H1}(u)$ due to setup delay $\propto u$
- Decomposition property (Fuhrmann & Cooper, 1985)

Quadratic Holding Costs

- Linear holding cost corresponds to metrics such as (for FCFS)
  - Latency (i.e., delay, sojourn time, waiting time)
  - Slowdown (ratio of the latency to job size, T/X)
  - . . . anything that is directly proportional to T
- Not everything is linear
  - E.g., longer waiting may cause more customer dissatisfaction $\Rightarrow$ cost rate increases!
- What about quadratic costs?

Virtual backlog,	cost rate $\propto U(t)^2$
Latency of Job i,	cost incurred $\propto (T_i)^2$

Good news: These can be computed too!

Quadratic Holding Costs

The mean holding cost rate is

$$r_{H2} = \mathrm{E}[U^2] = \frac{3\lambda^2\, \mathrm{E}[X^2]^2 + 2\lambda(1-\rho)\, \mathrm{E}[X^3]}{6(1-\rho)^2} + \underbrace{\frac{3\rho+\lambda s}{3(1+\lambda s)} s^2 + \frac{\lambda(2+\lambda s)\, \mathrm{E}[X^2]}{2(1-\rho)(1+\lambda s)} s}_{\text{setup delay}}$$

The corresponding value function is

$$v_{H2}(u) - v_{H2}(0) = \frac{1}{3(1-\rho)} u^3 + \frac{\lambda\, \mathrm{E}[X^2]}{2(1-\rho)^2} u^2 - \underbrace{\left(\frac{3\rho+\lambda s}{3(1-\rho)(1+\lambda s)} s^2 + \frac{\lambda(2+\lambda s)\, \mathrm{E}[X^2]}{2(1-\rho)^2(1+\lambda s)} s \right) u}_{\text{setup delay}}$$

- Mean cost rate (cf. PK) and value function resemble each other
- Setup delay appears as extra terms in both
- In value function, the cost of setup delay is proportional to $-u$

Waiting Time and Latency (FCFS)

- For an arbitrary cost function $c(u)$

$$c_1 \triangleq \mathrm{E}[c(W_1) + \ldots + c(W_{N_u})]$$

$$c_2 \triangleq \lambda\, \mathrm{E}[\int_0^{B_u} c(U_t)\, dt]$$

PASTA $\Rightarrow c_1 = c_2$

- For waiting time W and its square (FCFS)

Linear	$v_W(u) - v_W(0) = \lambda \left(v_{H1}(u) - v_{H1}(0) - \dfrac{du}{1+\lambda s} \right)$
Quadratic	$v_{W2}(u) - v_{W2}(0) = \lambda \left(v_{H2}(u) - v_{H2}(0) - \dfrac{s^2 u}{1+\lambda s} \right)$

- For latency, $v_T(u) - v_T(0) = v_W(u) - v_W(0)$
  Similarly, an expression for $v_{T2}(u)$ can be obtained

Summary

So what do we have?

Cost type	mean rate	value function	immediate cost	
Switching cost	✓	✓	✓	
Running cost	✓	✓		
Holding costs U^k	✓	✓		
Waiting time W	✓	✓	✓	(FCFS)
Waiting time W^2	✓	✓	✓	(FCFS)
Latency T	✓	✓	✓	(FCFS)
Latency T^2	✓	✓	✓	(FCFS)

Reference:

[1] Hyytiä, Righter and Aalto, *Task Assignment in a Heterogeneous Server Farm with Switching Delays and General Energy-Aware Cost Structure*, Performance Evaluation (2014).

8. List of References

References

Aalto, S. and Virtamo, J. (1996). Basic packet routing problem. In *The thirteenth Nordic teletraffic seminar NTS-13*, pages 85–97, Trondheim, Norway.

Agrawala, A., E.G., J. C., Garey, M., and Tripathi, S. (1984). A stochastic optimization algorithm minimizing expected flow times on uniforn processors. *IEEE Transactions on Computers*, 33(4):351–356.

Akgun, O., Righter, R., and Wolff, R. (2011). Multiple server system with flexible arrivals. *Advances in Applied Probability*, 43:985–1004.

Akgun, O. T., Down, D. G., and Righter, R. (2014). Energy-aware scheduling on heterogeneous processors. *IEEE Transactions on Automatic Control*, 59(3).

Altman, E., Ayesta, U., and Prabhu, B. (2011). Load balancing in processor sharing systems. *Telecommunication Systems*, 47(1):35–48.

Bachmat, E. and Sarfati, H. (2010). Analysis of SITA policies. *Performance Evaluation*, 67(2):102–120.

Conolly, B. W. (1984). The autostrada queueing problem. *Journal of Applied Probability*, 21(2):394–403.

Crovella, M. E., Harchol-Balter, M., and Murta, C. D. (1998). Task assignment in a distributed system: Improving performance by unbalancing load. In *Proceedings of SIGMETRICS '98*, pages 268–269, Madison, Wisconsin, USA.

Daley, D. J. (1987). Certain optimality properties of the first-come first-served discipline for G/G/s queues. *Stochastic Processes and their Applications*, 25:301–308.

Doroudi, S., Hyytiä, E., and Harchol-Balter, M. (2014). Value driven load balancing. *Performance Evaluation*, 79:306–327. (IFIP Performance'14).

Down, D. and Wu, R. (2006). Multi-layered round robin routing for parallel servers. *Queueing Systems*, 53(4):177–188.

Ephremides, A., Varaiya, P., and Walrand, J. (1980). A simple dynamic routing problem. *IEEE Transactions on Automatic Control*, 25(4):690–693.

References (cont.)

Feinberg, E. and Kella, O. (2002). Optimality of D-policies for an M/G/1 queue with a removable server. *Queueing Systems*, 42(4):355–376.

Feng, H., Misra, V., and Rubenstein, D. (2005). Optimal state-free, size-aware dispatching for heterogeneous M/G/-type systems. *Performance Evaluation*, 62(1-4):475–492.

Foss, S. G. (1980). Approximation of multichannel queueing systems. *Siberian Mathematical Journal*, 21:851–857.

Gandhi, A., Doroudi, S., Harchol-Balter, M., and Scheller-Wolf, A. (2013). Exact analysis of the M/M/k/setup class of markov chains via recursive renewal reward. In *Proc. ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*.

Gandhi, A., Harchol-Balter, M., and Adan, I. (2010). Server farms with setup costs. *Performance Evaluation*, 67(11):1123–1138.

Haight, F. A. (1958). Two queues in parallel. *Biometrika*, 45(3-4):401–410.

Hajek, B. (1983). The proof of a folk theorem on queuing delay with applications to routing in networks. *J. ACM*, 30(4):834–851.

Hajek, B. (1985). Extremal splittings of point processes. *Mathematics of Operations Research*, 10(4):543–556.

Harchol-Balter, M., Crovella, M. E., and Murta, C. D. (1999). On choosing a task assignment policy for a distributed server system. *Journal of Parallel and Distributed Computing*, 59:204–228.

Harchol-Balter, M., Scheller-Wolf, A., and Young, A. R. (2009). Surprising results on task assignment in server farms with high-variability workloads. In *Proc. of SIGMETRICS*, pages 287–298, New York, NY, USA. ACM.

Heyman, D. (1977). The T-policy for the M/G/1 queue. *Management Science*, 23(7):775–778.

Heyman, D. P. (1968). Optimal operating policies for M/G/1 queuing systems. *Operations Research*, 16(2):pp. 362–382.

Hordijk, A. and Koole, G. (1990). On the optimality of the generalised shortest queue policy. *Prob. Eng. Inf. Sci.*, 4:477–487.

References (cont.)

Hyytiä, E. (2013). Lookahead actions in dispatching to parallel queues. *Performance Evaluation*, 70(10):859–872. (IFIP Performance'13).

Hyytiä, E. (2014). Optimal routing of fixed size jobs to two parallel servers. *INFOR: Information Systems and Operational Research*. to appear.

Hyytiä, E., Aalto, S., and Penttinen, A. (2012a). Minimizing slowdown in heterogeneous size-aware dispatching systems. *ACM SIGMETRICS Performance Evaluation Review*, 40:29–40. (ACM SIGMETRICS/Performance conference).

Hyytiä, E., Aalto, S., Penttinen, A., and Virtamo, J. (2012b). On the value function of the M/G/1 FCFS and LCFS queues. *Journal of Applied Probability*, 49(4):1052–1071.

Hyytiä, E., Penttinen, A., and Aalto, S. (2012c). Size- and state-aware dispatching problem with queue-specific job sizes. *European Journal of Operational Research*, 217(2):357–370.

Hyytiä, E., Penttinen, A., Aalto, S., and Virtamo, J. (2011a). Dispatching problem with fixed size jobs and processor sharing discipline. In *23rd International Teletraffic Congress (ITC'23)*, pages 190–197, San Fransisco, USA.

Hyytiä, E., Penttinen, A., and Sulonen, R. (2012d). Non-myopic vehicle and route selection in dynamic DARP with travel time and workload objectives. *Computers & Operations Research*, 39(12):3021–3030.

Hyytiä, E., Righter, R., and Aalto, S. (2014a). Energy-aware job assignment in server farms with setup delays under LCFS and PS. In *26th International Teletraffic Congress (ITC'26)*, Karlskrona, Sweden.

Hyytiä, E., Righter, R., and Aalto, S. (2014b). Task assignment in a heterogeneous server farm with switching delays and general energy-aware cost structure. *Performance Evaluation*, 75–76(0):17–35.

Hyytiä, E., Virtamo, J., Aalto, S., and Penttinen, A. (2011b). M/M/1-PS queue and size-aware task assignment. *Performance Evaluation*, 68(11):1136–1148. (IFIP Performance'11).

Johri, P. K. (1989). Optimality of the shortest line discipline with state-dependent service rates. *European Journal of Operational Research*, 41(2):157–161.

References (cont.)

Koole, G. (1995). A simple proof of the optimality of a threshold policy in a two-server queueing system. *Systems and Control Letters*, 26(5):301–303.

Koole, G., Sparaggis, P. D., and Towsley, D. (1999). Minimizing response times and queue lengths in systems of parallel queues. *Journal of Applied Probability*, 36(4):1185–1193.

Krishnan, K. R. (1987). Joining the right queue: a Markov decision rule. In *Proc. of the 28th Conference on Decision and Control*, pages 1863–1868.

Krishnan, K. R. (1990). Joining the right queue: a state-dependent decision rule. *IEEE Transactions on Automatic Control*, 35(1):104–108.

Krishnan, K. R. and Ott, T. J. (1986). State-dependent routing for telephone traffic: Theory and results. In *IEEE Conference on Decision and Control*, volume 25, pages 2124–2128.

Larsen, R. L. (1981). *Control of multiple exponential servers with application to computer systems.* PhD thesis, University of Maryland at College Park, College Park, MD, USA.

Levy, Y. and Yechiali, U. (1975). Utilization of idle time in an M/G/1 queueing system. *Management Science*, 22(2):202–211.

Lin, W. and Kumar, P. (1984). Optimal control of a queueing system with two heterogeneous servers. *IEEE Transactions on Automatic Control*, 29(8):696–703.

Liu, Z. and Righter, R. (1998). Optimal load balancing on distributed homogeneous unreliable processors. *Operations Research*, 46(4):563–573.

Liu, Z. and Towsley, D. (1994). Optimality of the round-robin routing policy. *Journal of Applied Probability*, 31(2):466–475.

Penttinen, A., Hyytiä, E., and Aalto, S. (2011). Energy-aware dispatching in parallel queues with on-off energy consumption. In *30th IEEE International Performance Computing and Communications Conference (IPCCC)*, Orlando, FL, USA.

References (cont.)

Righter, R. and Xu, S. (1991). Scheduling jobs on nonidentical IFR processors to minimize general cost functions. *Adv. Appl. Prob.*, 23:909–924.

Schrage, L. (1968). A proof of the optimality of the shortest remaining processing time discipline. *Operations Research*, 16(3).

Sharifnia, A. (1997). Instability of the join-the-shortest-queue and FCFS policies in queuing systems and their stabilization. *Operations Research*, 45(2):309–314.

Sparaggis, P. D. and Towsley, D. (1994). Optimal routing and scheduling of customers with deadlines. *Probability in the Engineering and Informational Sciences*, 8(1):33–49.

Stoyan, D. (1976). A critical remark on a system approximation in queueing theory. *Math. Operationsforsch. Statist.*, 7:953–956.

Towsley, D., Sparaggis, P., and Cassandras, C. (1990). Stochastic ordering properties and optimal routing control for a class of finite capacity queueing systems. In *Proc. of the 29th IEEE Conference on Decision and Control*, pages 658–663.

van Leeuwaarden, J., Aalto, S., and Virtamo, J. (2001). Load balancing in cellular networks using first policy iteration. Technical report, Networking Laboratory, Helsinki University of Technology.

Véricourt, F. and Zhou, Y.-P. (2006). On the incomplete results for the heterogeneous server problem. *Queueing Syst. Theory Appl.*, 52(3):189–191.

Viniotis, I. and Ephremides, A. (1988). Extension of the optimality of the threshold policy in heterogeneous multiserver queueing systems. *IEEE Transactions on Automatic Control*, 33(1):104–109.

Virtamo, J. (2004). Lecture notes on Markov decision processes, 38.141 Teletraffic Theory, TKK.

Walrand, J. (1984). A note on 'optimal control of a queueing system with two heterogeneous servers'. *Systems and Control Letters*, 4:131–134.

Weber, R. R. (1978). On the optimal assignment of customers to parallel servers. *Journal of Applied Probability*, 15(2):406–413.

References (cont.)

Welch, P. D. (1964). On a generalized M/G/1 queuing process in which the first customer of each busy period receives exceptional service. *Operations Research*, 12(5):736–752.

Whittle, P. (1996). *Optimal Control: Basics and Beyond*. Wiley.

Wierman, A., Harchol-Balter, M., and Osogami, T. (2005). Nearly insensitive bounds on SMART scheduling. In *ACM SIGMETRICS*, pages 205–216, New York, NY, USA. ACM.

Winston, W. (1977). Optimality of the shortest line discipline. *Journal of Applied Probability*, 14:181–189.

Wu, R. and Down, D. G. (2009). Round robin scheduling of heterogeneous parallel servers in heavy traffic. *European Journal of Operational Research*, 195(2):372–380.

Yadin, M. and Naor, P. (1963). Queueing systems with a removable service station. *Operational Research Quarterly*, 14(4):393–405.

Yang, S. and de Veciana, G. (2002). Size-based adaptive bandwidth allocation: optimizing the average QoS for elastic flows. In *IEEE INFOCOM*, volume 2, pages 657–666.