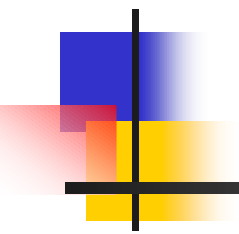


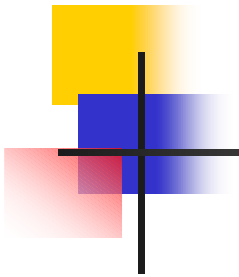
An Optimal Service Ordering for a World Wide Web Server

Amy Csizmar Dalal
Hewlett-Packard Laboratories
Palo Alto, CA
email: amy_dalal@hp.com

Scott Jordan
University of California
Irvine, CA
email: sjordan@uci.edu

PAWS-2001

June 17, 2001



Overview

- Motivation
- System model
- Derivation of an optimal policy
- Two extensions
- A counterexample
- Conclusions

Motivation

- Key motivation is the existence of an *impatient user population*
  - processor sharing is not effective if many users time out
  - is fairness really a good idea?
- Key metric is *user-perceived performance*, measured as a function of user response time at the web server

The model

- Web server modeled as a single-server queue
  - service times ~ exp(μ), i.i.d, proportional to sizes of requested files
  - Request arrivals ~ Poisson(λ)
- Network delays not considered here
- Switching time between requests is negligible

The model

- User-perceived performance ≡ 1 - P{user aborts the download before the file is completely received}
  - decreasing function of response time
  - ≡ ***revenue***
- Server is unaware if/when requests are aborted
- Server alternates between busy cycles, when there is at least one request in the system, and idle cycles (state *0*), when there are no requests waiting or in service. These cycles are i.i.d.

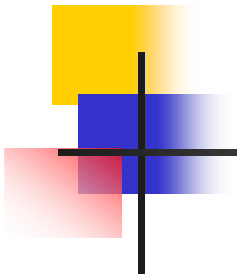
The model

- **Revenue** earned by the server under policy p from serving request i to completion (collected upon departure):

$$g_{ip}(t_{ip}) = e^{-c(t_{ip}-x_i)}$$

- **Potential revenue**: $g_{ip}(t)$
- By the Renewal Reward Theorem, the average reward earned by the server per unit time under policy p is

$$V_p(0) = \frac{\mathrm{E}\left[\sum_{i=0}^{N} g_{ip}(t_{ip})\right]}{\mathrm{E}[Z_1]}$$



Objective

The optimal policy is the service policy that maximizes average revenue earned per unit time; it satisfies

$$V = \max_{p} V_p(0)$$

Derivation of the optimal policy

The optimal policy satisfies the following:

1. non-idling
2. switches between jobs in service only at arrival/departure epochs
3. non-processor-sharing
4. Markov (independent of both past and future arrivals)

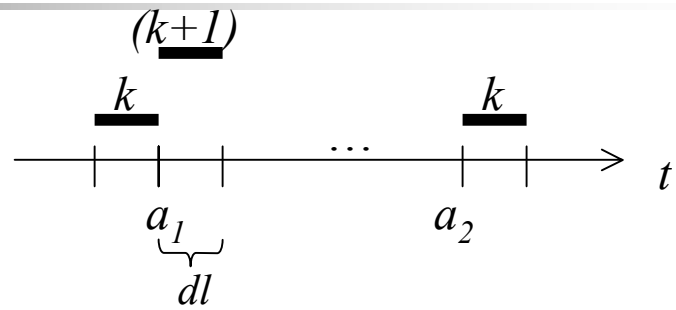
Proof of (2)

If $g_{(k+1)}(a_1) \geq g_k(a_1)$

use p'

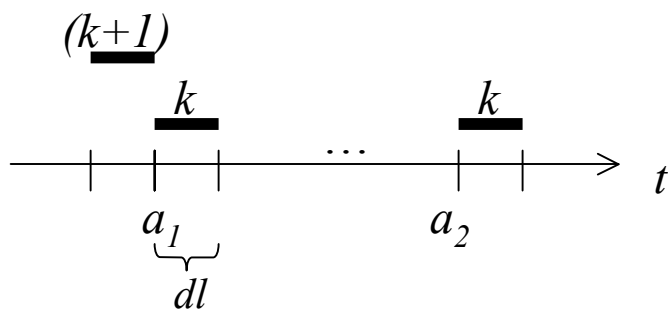
Revenue difference =

$$\mu\, d\ell\, e^{-c(a_1 - x_{(k+1)})}(1 - e^{-c\,d\ell})$$

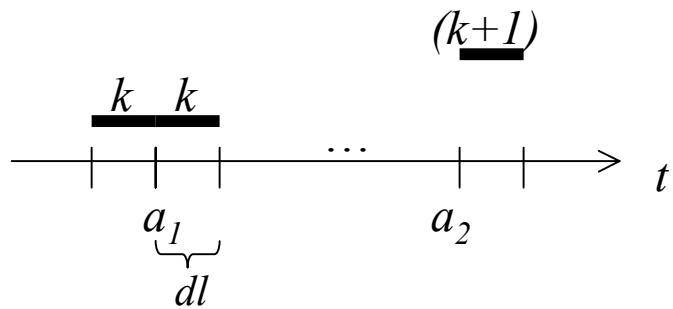


Else use p''

Revenue difference =



$$\mu\, d\ell\, e^{-c\,d\ell}(e^{-ca_1} - e^{-ca_2})(e^{cx_k} - e^{cx_{(k+1)}})$$



Proof of (3)

Consider a portion of the sample path over which the server processes two jobs to completion

Three possibilities:

- serve 1 then 2: expected revenue is $\frac{\mu}{c+\mu}c_1 + \left(\frac{\mu}{c+\mu}\right)^2 c_2$ (1)
- serve 2 then 1: expected revenue is $\frac{\mu}{c+\mu}c_2 + \left(\frac{\mu}{c+\mu}\right)^2 c_1$ (2)
- serve both until one completes, then serve the other: expected revenue is

$$q(1) + (1-q)(2)$$

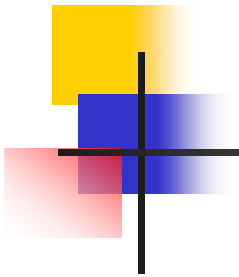
Definition of an optimal policy

State vector at time t under policy p:

$$G_{tp} = \{g_{1p}(t) \quad g_{2p}(t) \quad \ldots \quad g_{Mp}(t)\}$$

$\boldsymbol{M}$ = number of requests that have not departed prior to t

best job: $\arg\max_i g_{ip}(t), i \in 1, 2, \ldots M$



Optimal policy

The optimal policy chooses the *best job* to serve at any time t regardless of the order of service chosen for the rest of the jobs in the system (**greedy**)

⇨ **LIFO-PR**

Two extensions

- Varying initial reward
  - applicable to different “classes” of users, QoS levels, etc.
- File sizes independently drawn from a set of exponential distributions
  - applicable to hosting several types of content at a server, each described by a different exponential distribution

Extension 1

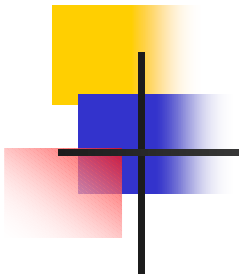
- *Initial reward* $\equiv C_i$
- Potential revenue: $g_{ip}(t) = C_i e^{-c(t-x_i)}$
- Natural extension of the original system model

Optimal policy: serve the request that has the highest potential revenue value at any time *t*

(function of the time in the system so far and the initial reward)

Extension 2

- Service time of request $i \sim \exp(\mu_i)$, independent
- Additional initial assumptions:
  - server only switches between jobs in service upon an arrival to or departure from the system
  - non-processor-sharing



Extension 2

We show that the optimal policy is

- non-idling
- Markov (independent of past and future arrivals)

Extension 2

State vector at time t under policy p:

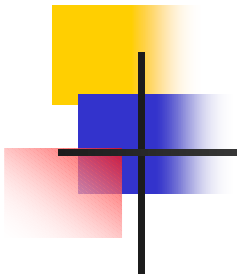
$$G_{tp} = \{(g_{1p}(t), \mu_1) \quad (g_{2p}(t), \mu_2) \ldots (g_{Mp}(t), \mu_M)\}$$

Best job satisfies

$$\arg\max_i g_{ip}(t)\,\mu_i, i \in 1, 2, \ldots M$$

Optimal policy:

Serve the request with the highest $g_{ip}(t)\ \mu_i$ product



A counterexample

Q: What happens if we remove the two additional initial assumptions in Extension 2?

- specifically, what happens if we throw processor-sharing policies back into the pool of possible policies?

A counterexample

Consider a sample path over which the server processes two jobs to completion. Job 1 arrives at time 0, job 2 arrives at time a.

Three possibilities from a until the end of the busy cycle:

- NPS1: serve 1 then 2
- NPS2: preempt 1, serve 2, serve 1
- PS: upon 2's arrival, serve both until one departs, then serve the other

A counterexample

Expected revenues for NPS1, NPS2:

$$\mathrm{E}(g_{NPS1}) = c_1 \frac{\mu_1}{c+\mu_1} + c_2 \frac{\mu_1}{c+\mu_1}\frac{\mu_2}{c+\mu_2}$$

$$\mathrm{E}(g_{NPS2}) = c_1 \frac{\mu_1}{c+\mu_1}\frac{\mu_2}{c+\mu_2} + c_2 \frac{\mu_2}{c+\mu_2}$$

A counterexample

However,

$$\mathrm{E}(g_{PS}) \neq q\mathrm{E}(g_{NPS1}) + (1-q)\mathrm{E}(g_{NPS2})$$

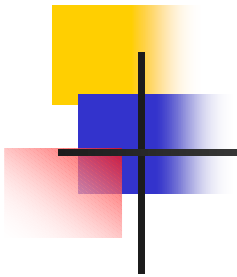
⇨ There exists at least one numerical example in which

$$\mathrm{E}(g_{PS}) > q\mathrm{E}(g_{NPS1}) + (1-q)\mathrm{E}(g_{NPS2})$$

A counterexample

A: We cannot clearly state that NPS policies are always better than PS policies. In fact, for at least one sample path, PS slightly outperforms both NPS alternatives.

(Note: We have not yet been able to reproduce this behavior in larger sample path simulations.)



Conclusions

- When network delays are ignored, impatient users are best served by a greedy policy
- This result does not always hold when file sizes are drawn from a set of exponential distributions
  - the implications of this are not yet completely understood