Solutions for exercises
Performance by Design

A.J. Bonnema
abonnema@xs4all.nl

March 24, 2008

Abstract

Solution to exercises from "Performance by Design" by Menascé, Almeida and Dowdy ([2]).

Mark, that neither the publisher of the book nor any of the authors have in any way been involved in the creation of these solutions. Also, neither the publisher nor any of the authors are responsible for or endorse any of the material enclosed. All responsibility for these solutions remain solely with the author of this document.

DISCLAIMER. This set of solutions is presented AS-IS. No pretence of usability, correctness or other qualities are assumed or implied. Do not read, unless you are completely sane and warded off from influences toward insanity. Also, please direct any flames, complaints, annoyances and other rants at spam@devnull.com.

However, if you find the results usable and correct, or, if you want to contribute to improve the results, by reporting errors, corrections, and giving other constructive remarks, please contact the author at his email address.

1 Chapter 1

1.1 exercise 1.1

In figure 1 the results of executing iostat with a delay of 120 seconds is printed. On the machine used, `sda` and `sdb` are harddisks and `hda` is the dvd-reader, which happens to contain a cdrom that is not mounted.

The first entry in the output is printed immediately and contains the averages of all actions from the previous call to iostat. We will ignore these figure and use the last two to determine the average throughput.

During the first two minutes a few PDF files were opened to be read and reading continued into the second two minutes. As you can probably see from the output, the PDF-files being read were on the second disk, `sdb`. Also, from the two measurements, you can see that reading the files does not induce any further I/O's.

During the first two minutes 9060 KB is read (column kB_read) for `sdb` or 75.5 KB/s on average (column Kb_read/s).

```
abonnema@athene:/work/werk/performance-analysis$ iostat -k 120
Linux 2.6.22-14-generic (athene)        03/09/2008

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
          14.02    0.06    2.57    0.37    0.00   82.99

Device:            tps    kB_read/s    kB_wrtn/s    kB_read    kB_wrtn
sda               2.94        26.04         5.39    2087388     432172
sdb               2.96        20.20        16.26    1619431    1303296
hda               0.00         0.00         0.00         76          0

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
          28.86    3.38    4.03    0.66    0.00   63.07

Device:            tps    kB_read/s    kB_wrtn/s    kB_read    kB_wrtn
sda               3.98        53.53       156.47       6424      18776
sdb               6.82        75.50       158.47       9060      19016
hda               0.00         0.00         0.00          0          0

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
          22.73    0.00    2.49    0.01    0.00   74.77

Device:            tps    kB_read/s    kB_wrtn/s    kB_read    kB_wrtn
sda               0.25         0.10        26.57         12       3188
sdb               0.74         0.07        13.30          8       1596
hda               0.00         0.00         0.00          0          0
```

Figure 1: **Exercise 1.1** Running iostat from linux with a 120 second interval. The first figures are printed immediately.

The exercise also mentions that iostat shows the service time per transaction. This appears not to be the case, so after re-reading the manual page for iostat, the experiment is repeated with different parameters.

In figure 2 the statement is adjusted and limited to the second harddisk. The

first set of figures shows the result of opening a set of LaTeX files for editting. The statistics show an average throughput of 2 read requests per second and 0.9 write requests per second. The average service time i.e. `await` is equal to 5.73 milliseconds, while the service time without the wait is about 1.57 milliseconds.

The utilization given as 0.46% is the percentage of CPU time during which I/O requests were being issued to the device. The utilization of the harddisk will be the total busy time divided by the total observation time. The total busy time is equal to the average service time (`svctm`) times 120 seconds. Therefore the utilization is equal to the service time i.e. 0.00157 or 0.2%.

```
abonnema@athene:/work/werk/performance-analysis$ iostat -x sdb -k 120
Linux 2.6.22-14-generic (athene)        03/09/2008

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
          14.03    0.06    2.55    0.37    0.00   83.00

Device:   rrqm/s   wrqm/s     r/s     w/s    rkB/s    wkB/s avgrq-sz avgqu-sz   await  svctm  %util
sdb         1.87     3.22    2.04    0.90    20.03    16.45    24.84     0.02    5.73   1.57   0.46
```

Figure 2: **Exercise 1.1** Running iostat with parameters to show the service time too.

1.2 exercise 1.2

From the performance tool in windows XP, after adding the counter %Idle for the object PhysicalDisk measurements were started. Then start opening and reading as many PDF files as possible (books).

As you can see in figure 3, the disk was not impressed. Probably, to get an idle figure of significantly less than 95%, we need a program that is really IO bound.

1.3 exercise 1.3

Given See table 1.2 on page 31.

Requested The availability of the site in the two days.

Solution

$$\text{Availability} = 1 - \frac{\sum \text{downtime}}{\text{total time}} =$$

$$1 - \frac{34}{2 \times 24 \times 60} =$$

$$1 - \frac{34}{2880} = \frac{2846}{2880} = 98.8\%$$

$$\boxed{\text{Avail} = 98.8\%}$$

1.4 exercise 1.4

Given See table 1.3 on page 31.

Requested Availibility and compare results with results of exercise 1.3.

```
"(PDH-TSV 4.0) (W. Europe Standard Time)(-60)" "\\U035223\PhysicalDisk(0 C: D: E:)\% Idle Time"
"PBD-ex-1.2 (Menasce)"
"02/27/2008 11:25:53.375" "9.0356420950221309e-005" "PBD-ex-1.2 (Menasce)"
"02/27/2008 11:25:58.390" "99.706654704049839" "PBD-ex-1.2 (Menasce)"
"02/27/2008 11:26:03.406" "99.95076984423676" "PBD-ex-1.2 (Menasce)"
"02/27/2008 11:26:08.421" "99.340576697819316" "PBD-ex-1.2 (Menasce)"
"02/27/2008 11:26:13.421" "96.919553999999991" "PBD-ex-1.2 (Menasce)"
"02/27/2008 11:26:18.421" "99.409019999999998" "PBD-ex-1.2 (Menasce)"
"02/27/2008 11:26:23.421" "99.344324" "PBD-ex-1.2 (Menasce)"
"02/27/2008 11:26:28.421" "98.494188000000008" "PBD-ex-1.2 (Menasce)"
"02/27/2008 11:26:33.421" "96.361058" "PBD-ex-1.2 (Menasce)"
"02/27/2008 11:26:38.421" "99.162134000000009" "PBD-ex-1.2 (Menasce)"
"02/27/2008 11:26:43.421" "99.926068000000001" "PBD-ex-1.2 (Menasce)"
"02/27/2008 11:26:48.421" "99.964337999999998" "PBD-ex-1.2 (Menasce)"
"02/27/2008 11:26:53.421" "98.545289999999994" "PBD-ex-1.2 (Menasce)"
"02/27/2008 11:26:58.421" "99.964390000000009" "PBD-ex-1.2 (Menasce)"
"02/27/2008 11:27:03.421" "99.969042000000002" "PBD-ex-1.2 (Menasce)"
"02/27/2008 11:27:08.421" "99.962848000000008" "PBD-ex-1.2 (Menasce)"
"02/27/2008 11:27:13.421" "99.176852000000011" "PBD-ex-1.2 (Menasce)"
"02/27/2008 11:27:18.421" "99.989642000000003" "PBD-ex-1.2 (Menasce)"
"02/27/2008 11:27:23.421" "98.055081999999999" "PBD-ex-1.2 (Menasce)"
"02/27/2008 11:27:28.421" "97.917982000000009" "PBD-ex-1.2 (Menasce)"
"02/27/2008 11:27:33.421" "98.130861999999993" "PBD-ex-1.2 (Menasce)"
"02/27/2008 11:27:38.421" "99.238333999999995" "PBD-ex-1.2 (Menasce)"
"02/27/2008 11:27:43.421" "97.375553999999994" "PBD-ex-1.2 (Menasce)"
"02/27/2008 11:27:48.421" "98.588346000000001" "PBD-ex-1.2 (Menasce)"
"02/27/2008 11:27:53.421" "95.010300000000001" "PBD-ex-1.2 (Menasce)"
"02/27/2008 11:27:58.421" "97.551407999999995" "PBD-ex-1.2 (Menasce)"
```

Figure 3: **Exercise 1.1** Performance tool monitoring PhysicalDisk - %Idle while reading PDF files.

Solution

$$\text{Availability} = 1 - \frac{\sum \text{downtime}}{\text{total time}} = 1 - \frac{34}{2 \times 24 \times 60} = 1 - \frac{34}{2880} = \frac{2846}{2880} = 98.8\%$$

$\boxed{\text{Avail} = 98.8\%}$

The total availibility in 2 days is equal to 98.8 % in both exercises. Assuming day trading hours of 9:00 AM to 5:00 PM, exercise 1.3 presents a much better case than exercise 1.4.

In case 1.3 only 6 minutes of downtime occurred during general trading hours. In case 1.4 all 34 minutes of downtime are during trading hours.

1.5 exercise 1.5

Most projects I have been in had few, if any, non-functional requirements.

Performance requirements On a project, where the system was designed, built and used within the same company, performance requirements were noted in the set of requirements. However, the requirements were not checked for being realistic in a cost versus profit sense. Also, when the time came to test, the only criterion really used, was whether the accepting party could be persuaded

to accept the system, because time and money were no longer available. This meant: "Either accept, or have no system at all". The users were persuaded to accept. Measurements were done by timing with a wristwatch on the most business sensitive transactions.

Later on, during maintenance of the system, performance was never an issue. The business performance i.e. number of finished transactions per day, was more than adequate for the job.

Availability requirements The same project also contained an availability requirement of a number of days per year and a maximum of one week non-availability in one stretch. No measurements were made to insure that this requirement was met, because the business had a manual fallback procedure that was slower, but for a short period, adequate.

2 Chapter 2

2.1 exercise 2.1

Given A 56 Kbps line transporting packets of 1500 bytes.

Requested

- What is the service demand of each packet?
- Does the service demand change with traffic?

Solution Service demand = $\frac{1500 \times 8bits}{56 \times 1024bits} = \frac{12000}{57344} = 0.209$ sec.

$\boxed{D_i = 0.209 \text{ sec}}$

If more traffic arrives, the service demand remains the same, but the packet may have to wait before being transferred, if other packets precede it.

Also, the chance of failure and resubmission may increase and add to the waiting time of the packet.

2.2 exercise 2.2

Solution Each CPU represents 1 queue, hence $K = 4$. As the programs in the benchmark are equivalent in their CPU use, there is one class: $R = 1$. As the benchmark consists of serveral programs running concurrently, the number of customers is bounded. Thus the best way to model this network is a closed QN. The CPU's are load independant resources: so type = LI for each of the CPU's.

2.3 exercise 2.3

Given A transaction workload of 10 tps and 50 client workstations submitting requests.

Requested The type of QN model to be used (open, closed, mixed).

Solution The transaction workload has a fixed arrival rate, and an unspecified number of customers. This indicates an open QN class.

The 50 workstations submitting requests has a fixed number of customers and no specified arrival rate. This indicates a closed QN class.

In conclusion, the system contains a closed customer class and an open customer class and can best be modeled using a mixed QN.

2.4 exercise 2.4

Given $D_1 = 0.100$ sec, $D_2 = 0.150$ sec.

Requested The change in service demand in the following cases:

1. disk 1 increases speed by 40%
2. hit rate on database server's cache is 30%
3. A log on disk 2 for all update transactions. 30% of all transactions are update transactions. Recording a log takes 0.015 sec.

	Class			
	TT	MT	CT	UT
Arrival rate (tps)	0.20	0.30	0.20	0.10
Service demand (sec)				
CPU	0.15	0.30	0.45	0.70
Disk 1	0.20	0.35	0.55	0.30
Disk 2	0.10	0.30	0.60	0.20

Figure 4: **Exercise 2.5**: given, see table 2.6 on page 58.

Solution

1. Only 60% of the original service demand will be necessary in the new situation: $D_1 = 0.100 * 0.6 = 0.06$ sec

   $\boxed{D_1 = 0.06 \text{ sec.}}$

2. In 30% of the transactions no demand for service is made. Otherwise, the service demand does not change. So on average a service demand of 0.07 seconds is necessary for disk 1 and 0.105 seconds for disk 2:

   $$D_1 = 0.3 \times 0 + 0.7 \times 0.100 = 0.07$$

   $$D_2 = 0.3 \times 0 + 0.7 \times 0.150 = 0.105$$

3. - Service demand in case of an update = $D_1 = 0.1$ sec and $D_2 = 0.150 + 0.015 = 0.165$ sec. This occurs in 30% of the cases.
   - Service demand in case of a retrieval remains $D_1 = 0.1$ sec and $D_2 = 0.150$. This occurs in 70% of the cases.

   Thus, $D_1 = 0.1$ in all cases. However $D_2 = 0.30 \times 0.165 + 0.70 \times 0.150 = 0.1545$ on average for disk2.

   $\boxed{D_2 = 0.1545}$

2.5 exercise 2.5

Given See figure 4.

Requested How to aggregate MT and CT into a single class? Determine arrival rate and service demands of the aggregated class.

Solution The arrivals of MT and CT accumulate over time. So the arrival rates can be cumulated into a new arrival rate $\lambda_{MCT} = \lambda_{MT} + \lambda_{CT} = 0.3 + 0.2 = 0.5$.

The service demands of the aggregated class MCT depend on the service demands of both MT and CT classes. Also it depends on the respective arrival rates.

The aggregated class MCT will, on average, need $\lambda_{MCT} \times D_{i,MCT}$ seconds of resource i per second. Analogously, MT will need $\lambda_{MT} \times D_{i,MT}$ seconds and CT will need $\lambda_{CT} \times D_{i,CT}$ seconds of resource i per second. The sum of MT and CT average service demands per second must equal the average service demands of the aggregated class MCT. Thus:

$$\begin{aligned} \lambda_{MCT} \times D_{i,MCT} &= \lambda_{MT} \times D_{i,MT} + \lambda_{CT} \times D_{i,CT} \Leftrightarrow \\ D_{i,MCT} &= \frac{\lambda_{MT} \times D_{i,MT} + \lambda_{CT} \times D_{i,CT}}{\lambda_{MCT}} \end{aligned}$$

Depending on the arrival rates, the end result will be between both service demands and closest to the one with the largest arrival rate, in this case MT.

Using this formula generates the following service times for the aggregated class MCT:

$$\begin{aligned} D_{cpu,MCT} &= \frac{\lambda_{MT} \times D_{CPU,MT} + \lambda_{CT} \times D_{CPU,CT}}{\lambda_{MCT}} \\ &= \frac{0.3 \times 0.3 + 0.2 \times 0.45}{0.5} \\ &= \frac{0.09 + 0.09}{0.5} = 0.36 \text{ sec} \end{aligned}$$

$D_{cpu,MCT} = 0.36$ sec.

$$\begin{aligned} D_{d_1,MCT} &= \frac{\lambda_{MT} \times D_{Disk1,MT} + \lambda_{CT} \times D_{Disk1,CT}}{\lambda_{MCT}} \\ &= \frac{0.3 \times 0.35 + 0.2 \times 0.55}{0.5} \\ &= \frac{0.105 + 0.11}{0.5} = 0.43 \text{ sec} \end{aligned}$$

$D_{d_1,MCT} = 0.43$ sec.

$$\begin{aligned} D_{d_2,MCT} &= \frac{\lambda_{MT} \times D_{Disk2,MT} + \lambda_{CT} \times D_{Disk2,CT}}{\lambda_{MCT}} \\ &= \frac{0.3 \times 0.30 + 0.2 \times 0.60}{0.5} \\ &= \frac{0.09 + 0.12}{0.5} = 0.42 \text{ sec} \end{aligned}$$

$D_{d_2,MCT} = 0.42$ sec.

2.6 exercise 2.6

Given Consider the customer (delay resource) as a special case of a load dependant resource. A closed QN with n customers and a "service time" of S per customer.

Requested The service rate $\mu(n)$.

Solution A load-dependent resource can be viewed as a single queue with m resources as in figure PBD-2.1 on page 39. A delay resource can be viewed as a load-dependent resource with precise n processors if n customers arrive.

The response time for each customer is thus always equal to S. For one customer the service rate is equal to $\frac{1}{S}$. For n customers the service rate is equal to $n \times \frac{1}{S}$.

$\mu(n) = \frac{n}{S}$

example Suppose the service time for the delay resource is 5 seconds. The service rate for one customer is thus $\frac{1}{5}$ tps. If 5 customers arrive per second, the service rate will be 1 tps. If 10 customers arrive per second the service rate will be 2 tps. And so on.

3 Chapter 3

3.1 exercise 3.1

Given $T = 3600$ sec, $C_0 = 7200, M = 5$

Requested Average time spent in a system by a job: R.

Solution According to Little's law the average number of jobs in the system $R = \frac{M}{X_0}$. By definition the throughput is equal to $X_0 = \frac{C_0}{T} = \frac{7200}{3600} = 2$ tps. Thus, the average number of jobs in the system is equal to $R = \frac{5}{2} = 2.5$ sec.

$$\boxed{\begin{array}{l} R \\ = 2.5 \text{ sec} \end{array}}$$

3.2 exercise 3.2

Given

$U_{CPU} = 0.25, U_{disk1} = 0.35, U_{disk2} = 0.30$
$T = 3600$ sec, $C_0 = 21600$ requests
$QN =$ open

Requested

$$D_{CPU}, D_{disk1}, D_{disk2}, X_0^{max}, R_0$$

Solution

$$\left.\begin{array}{lll} D_{CPU} & = \frac{U_{CPU}}{X_0} & \\ X_0 & = \frac{C_0}{T} = \frac{21600}{3600} & = 6 \end{array}\right\} \Rightarrow$$

$$D_{CPU} = \frac{0.25}{6} = 0.042 \qquad \boxed{D_{CPU} = 0.042}$$

$$D_{disk1} = \frac{U_{disk1}}{X_0} = \frac{0.35}{6} = 0.058 \qquad \boxed{D_{disk1} = 0.058}$$

$$D_{disk2} = \frac{U_{disk2}}{X_0} = \frac{0.30}{6} = 0.050 \qquad \boxed{D_{disk2} = 0.050}$$

$$X_0^{max} = min[\tfrac{1}{max(D_i)}, \tfrac{N}{\sum_{i=1}^{3} D_i}] \qquad \boxed{X_0^{max} = 17.2}$$

$$X_0^{max} = min[\tfrac{1}{0.058}, n.a.] = min[17.2, n.a.] = 17.2$$

the variable N is not defined in an open system. $\boxed{X_0^{\max} = 17.2}$

Estimating the response time

$$R_i = \begin{cases} S_i & \text{if there is no waiting time} \\ S_i + \underbrace{Q \times S_i}_{\text{using forced flow law}} & \text{if the queue has length Q} \end{cases}$$

$\Leftrightarrow R_i = S_i + R_i X_i S_i$ where $Q = R_i \times X_i$

$\Leftrightarrow R_i(1 - X_i S_i) = S_i$ now get $R_i$alone on the left.

$\Leftrightarrow R_i = \frac{S_i}{1 - X_i S_i}$ using utilization law $U_i = S_i \times X_i$.

$\Leftrightarrow R_i = \frac{S_i}{1 - U_i}$ and now the question is: what is S_i?

Note that $\lambda_i = X_i$, so the formula would be: $\frac{S_i}{1-\lambda S_i}$. We still need V_i to determine S_i. Assuming $V_i = 1$ gives $X_1 = 1 \times X_0$ so that $S_i = \frac{U_i}{X_i} = D_i$. That gives $R_i = \frac{D_i}{1-X_0 D_i}$ Or:

$$R_1 = \frac{0{,}042}{1-0.25\times 0{,}042} = 0.042$$
$$R_2 = \frac{0.058}{1-0.35\times 0.058} = 0.059$$
$$R_3 = \frac{0.05}{1-0.30\times 0.05} = 0.05$$

An estimation for R_0^{max} would then be $\sum_{i=1}^{K} D_i = 0.042 + 0.059 + 0.05 = 0.151$ sec/tx.

$\boxed{R_0 \approx 0.151}$

3.3 exercise 3.3

Given See exercise 3.2

Requested The web server's throughput in a graph.

Solution According to Little's Law:

$$N_i = R_i \times X_i \Leftrightarrow$$
$$X_i = \frac{N_i}{R_i}$$

Maximizing throughput using upper bound formula (3.3.21) gives the upper bound:

$$\left.\begin{array}{l} X_0 = min[\frac{1}{max\{D_i\}}, \frac{N}{\sum_{i=1}^{K} D_i}] \\ max\{D_i\} = 0.058 \Rightarrow \frac{1}{0.058} \approx 17.2 \\ \sum_{i=1}^{K} D_i = 0.151 \end{array}\right\} \Rightarrow X_0 = min[17.2, \tfrac{N}{0.151}] \Leftrightarrow$$

$$\boxed{X_0 = min[17.2, 6.58N]}$$

For $N = 1, 2, 3$ these values are: $6.58, 13, 16, 17.2$ and from then on the upper limit remains 17.2. In other words, the graph should have values that remain on or below these values.

After solving the model using `ClosedQN.xls` the set of values was calculated (figure 5).

From these values one can easily see that the upper bounds mentioned above are respected and the upper limit of $16.\not{9}$ is approximated by the higher values of N. In figure 6 a plot of the first twenty values is printed.

3.4 exercise 3.4

Given

$T = 1800$ sec, $K = 1 : R_1 = disk, QN =$ open

$C_0 = 5400$ txs

$C_1 = 18900$ I/O's

$U_1 = 0.40$

N	X_0	N	X_0
1	$6,67$	11	$16,12$
2	$9,94$	12	$16,27$
3	$11,87$	13	$16,39$
4	$13,12$	14	$16,49$
5	$13,98$	15	$16,58$
6	$14,61$	16	$16,65$
7	$15,07$	17	$16,71$
8	$15,43$	18	$16,76$
9	$15,71$	19	$16,81$
10	$15,94$	20	$16,85$

Figure 5: **Exercise 3.3**: resulting throughput X_0 for values of N.

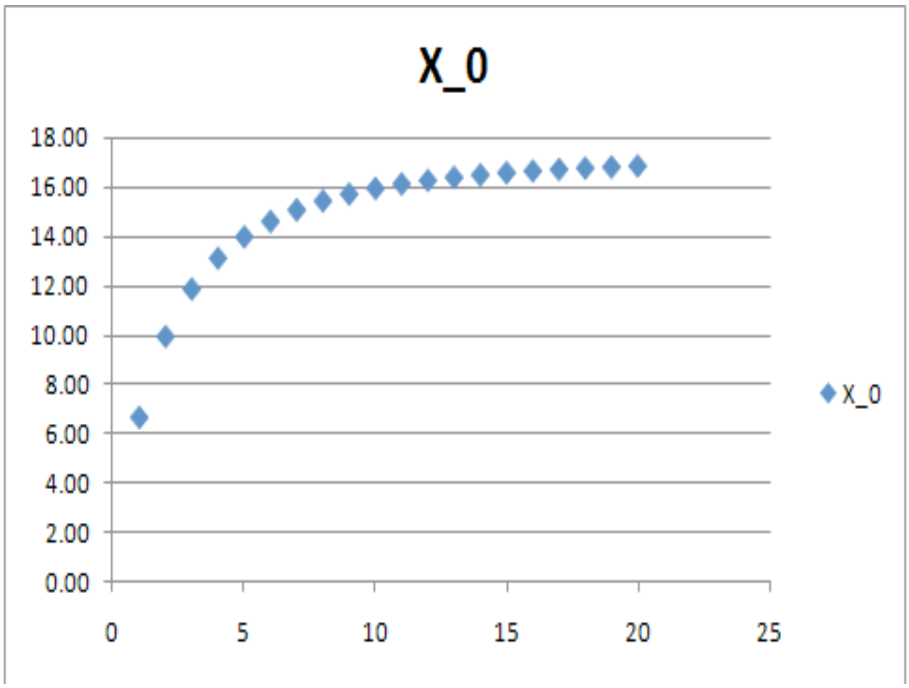


Figure 6: **Exercise 3.3**: throughput for the first twenty values of N

Requested

1. Average number of I/O's per tx
2. Average server time per tx

Solution By definition the average number of visits per transaction is equal to:

$$V_1 = \frac{C_1}{C_0} = \frac{18900}{5400} = 3.5 \text{ I/O's per sec} \qquad \boxed{V_1 = 3.5}$$

Using the utilization law we can derive the average server time per transaction:

$$\left.\begin{array}{l} U_i = S_i \times X_i \Rightarrow \\ S_i = \frac{U_i}{X_i} = \frac{0.40}{X_i} \\ X_i = \frac{C_i}{T} \Leftrightarrow X_1 = \frac{18900}{1800} = 10.5 \end{array}\right\} \Rightarrow S_1 = \frac{0.4}{10.5} \approx 0.038 \text{ sec} \qquad \boxed{S_0 = 0.13 \text{ sec}}$$

3.5 exercise 3.5

Given

$$T = 3600, C_0 = 5400 \text{ tx}, S_1 = 0.03, V_1 = 3, K = 1$$

Requested Utilization of disk

Solution

$$\left.\begin{array}{l} U_1 = S_1 \times X_1 \\ X_0 = \frac{C_0}{T} = \frac{5400}{3600} = 1.5 \text{ tps.} \\ X_1 = V_1 \times X_0 = 3 \times 1.5 = 4.5 \end{array}\right\} \Rightarrow U_1 = 0.03 \times 4.5 = 0.135 \qquad \boxed{U_1 = 0.135}$$

3.6 exercise 3.6

Given

$$S = 0.1 \text{ sec}, \; X = \lambda = 128 \text{ packets/sec.}$$

Requested Average number of concurrent packets in transit at any time

Solution

$$N = S \times X \qquad \text{Little's Law}$$

$$N = 0.1 \times 128 = 12.8 \text{ packets} \qquad \boxed{N = 12.8}$$

3.7 exercise 3.7

Given

$$T = 3600 \text{ sec}, C_0 = 7200, U_1 = 0.30, S_1 = 0.03 \text{ sec}$$

Requested Average number of accesses per file request

Solution

$$\left.\begin{array}{l} X_i = V_i \times X_0 \Leftrightarrow V_i = \frac{X_i}{X_0} \\ X_0 = \frac{C_0}{T} = \frac{7200}{3600} = 2 \text{ tps} \\ U_1 = S_1 \times X_1 \Leftrightarrow X_1 = \frac{U_1}{S_1} = \frac{0.3}{0.03} = 10 \end{array}\right\} \Rightarrow V_1 = \frac{X_1}{X_0} = \frac{10}{2} = 5 \text{ accesses} \qquad \boxed{V_1 = 5}$$

3.8 exercise 3.8

Given PBD-example 3.2 i.e. 3.4 (see remark)

Requested What are the values of system throughput and response time, and of utilization of the CPU and the disks, when there are 5 concurrent transactions in execution? (Use **ClosedQN.xls**).

Solution Remark: PBD-example 3.2 has no information on CPU activity. For that reason it was complemented with the contents of PBD-example 3.4.

Using the closed QN-model to solve for 5 concurrent transactions gives a system throughput of $X_0 = 5,64$ tps and a response time of $0,886$ seconds. The utilizations are:

	utilization
CPU	0,519
disk 1	0,446
disk 2	0,610
disk 3	0,801

3.9 exercise 3.9

Given

$$T = 3600 \text{ sec}, K = 3, R_1 = CPU, R_2 = disk_1, R_3 = disk_2$$
$$U_1 = 0.32, \quad U_2 = 0.6$$
$$V_2 = 5 \qquad V_3 = 8$$
$$S_2 = 0.03 \text{ sec}, \quad S_3 = 0.025 \text{ sec}$$

Requested

X_0, U_3, D_1, D_2, D_3

Solve `ClosedQN.XLS` (multi $= 0 \ldots 4$)

Give an approximation for average degree of multiprogramming

Solution

$$D_i = \frac{U_i}{X_0} = V_i \times S_i \qquad \text{(Service demand law)}$$
$$\Rightarrow X_0 = \frac{U_i}{V_i S_i} = \frac{U_2}{V_2 S_2} = \frac{0.6}{5x0.03} = 4 \qquad \boxed{X_0 = 4}$$

$$U_3 = S_3 X_3 = S_3(V_3 X_0) = 0.025 \times 8 \times 4 = 0.8 \qquad \boxed{U_3 = 0.80}$$

$D_1 = \frac{U_1}{X_0} = \frac{0.32}{4} = 0.08$ sec $\boxed{D_1 = 0.08}$

$D_2 = \frac{U_2}{X_0} = \frac{0.6}{4} = 0.15$ sec $\boxed{D_2 = 0.15}$

$D_3 = \frac{U_3}{X_0} = \frac{0.8}{4} = 0.2$ sec $\boxed{D_3 = 0.20}$

Resolution The values for N=0 can not be calculated using `ClosedQN.XLS`. But it can be explained[1] as follows. Using p as the probability that a process is not active, the probability that no process is active at multiprogramming level N is equal to p^N. The utilization of the resources will be $1 - p^N$. Now, if N is equal to zero, this indicates a utilization of $1 - p^0 = 1 - 1 = 0$ for all devices.

Using `ClosedQN.XLS` the values for N=1...4 were calculated as follows:

N	Class	X	R	Q
1	CPU	2.326	0.080	0.186
	disk 1	2.326	0.150	0.349
	disk 2	2.326	0.200	0.465
2	CPU	3.364	0.092	0.311
	disk 1	3.364	0.201	0.675
	disk 2	3.364	0.301	1.014
3	CPU	3.916	0.101	0.396
	disk 1	3.916	0.247	0.966
	disk 2	3.916	0.418	1.638
4	CPU	4.240	0.107	0.455
	disk 1	4.240	0.287	1.217
	disk 2	4.240	0.549	2.329

Approximate degree of multiprogramming From the figures in the previous item, one might take the throughput as a rough estimation of the multiprogramming level. Otherwise, two less obvious correlations are $Q(\text{disk}_1) * 10/3$ and $R(\text{disk}_2) * 10 - 1$.

3.10 exercise 3.10

Given

$$QN = \text{ Closed}, N = 50, Z = 5$$

$$u_1 = 0.60, S_1 = 0.03 \text{ sec}, V_1 = 4 \text{ visits}$$

Requested Response time R.

Solution

Response time Using the INTERACTIVE RESPONSE TIME LAW $R = \frac{M}{X_0} - Z$, and the SERVICE DEMAND LAW $U_i = D_i \times X_0$ we can derive that $X_0 = \frac{U_i}{D_i} = \frac{U_i}{V_i \times S_i}$.

$$\left.\begin{array}{l} R = \frac{M}{X_0} - Z = \frac{50}{X_0} - 5 \\ X_0 = \frac{U_i}{V_i S_i} = \frac{U_1}{V_1 S_1} = \frac{0.6}{4x0.03} = 5 \end{array}\right\} \Rightarrow R = \frac{50}{5} - 5 = 10 - 5 = 5 \text{ sec} \quad \boxed{R = 5}$$

[1] Reference: http://cs.gmu.edu/cne/modules/vm/green/multip.html. This site has other interesting modules, but often the links are broken. Sometimes they can be mended by inserting "cne/modules" before the module name.

3.11 exercise 3.11

Given

sd0			cpu			
kps	tps	serv	us	sy	wt	id
25	3	6	19	3	0	78
32	4	7	13	4	0	83
28	2	7	20	3	0	77
18	2	8	24	2	0	74
29	3	9	18	5	0	77
33	4	12	23	3	0	74
35	4	8	25	5	0	70
25	4	10	32	4	0	64
26	3	11	28	4	0	68
34	4	12	22	6	0	72

Requested Compute the disk and CPU utilizations.

Solution sd_0 From the SERVICE DEMAND LAW: $D_i = \frac{U_i}{X_0} = S_i \times V_i$ and the FORCED FLOW LAW: $X_i = V_i \times X_0$ follows $U_i = X_i \times S_i$ after rearranging the variables in the equations. In the given table this means X_i is column 'kps' and S_i is column 'serv'. In table 7 these values are calculated per line.

The average utilization for sd_0 is equal to $\frac{\sum^1 {}_{0_{j=1}} U_i[j]}{10} = \frac{2.595}{10} = 0.2595$ or a utilization U_{sd_0} of about 26%.

sd0			
kps	tps	serv	$U_i = X_i \times S_i$
25	3	6	0.150
32	4	7	0.224
28	2	7	0.196
18	2	8	0.144
29	3	9	0.261
33	4	12	0.396
35	4	8	0.280
25	4	10	0.250
26	3	11	0.286
34	4	12	0.408

Figure 7: **Exercise 3.11**: utilization for sd_0 calculated per line.

Solution CPU From the given table we can calculate the utilization as follows. Per line the U_{cpu} is equal to one minus the idle time, minus the wait: $1 - id - wt$ or $us + sy$. The average U_{cpu} amounts to $\frac{2.63}{10} = 0.263$ or 26.3%.

3.12 exercise 3.12

Given $Z = 5$ sec, $X_0 = 20$ req/sec, $R = 2$

Requested M

Solution

$$R = \frac{M}{X_0} - Z \Leftrightarrow M = X_0(R + Z)$$

$$\Rightarrow M = 20(5 + 2) = 20 \times 7 = 140 \text{ users}$$

$\boxed{M = 140 \text{ users}}$

It is remarkable, that just from input data like throughput and thinktime we are able to determine the required number of users to test with. This should come in handy during regular performance tests.

4 Chapter 4

4.1 exercise 4.1

Given Motivating example section 4.3.

Requested The performance requirements.

Solution A list of possible performance requirements:

- All functions should have a response time of less than one second for $x\%$ of the calls.
- The system should be operational for 24 hours a day, every day with a maximum maintenance time of x hours a month.
- The system should be able to handle 1000 calls per hour during peak hours while preserving the response time requirement.

4.2 exercise 4.2

Given The transition from design phase to development phase.

Requested A Workload model for both stages indicating additions during the development phase.

Solution See PBD-figure 4.3 on page 95. During the development phase the call center system will be detailed as follows:

- it should become clear which servers will be used. For instance: an application server, a database server and maybe a web server.
- Requirements for the servers and thus for the components within the server will become clear. For instance: requirements for CPU and hard disks of the database server.
- The different classes of input transactions should become clear. For instance: the batch jobs to be run and the number of transactions per hour for the online system.
- depending on the requirements these additions will be more or less detailed. For instance: for some calculations it may be necessary to know disk rotation speed and latency.

4.3 exercise 4.3

Given Motivating example plus data out of example 4.4.

Requested The maximum throughput of the database server.

Solution The maximum throughput X_0^{max} is equal to $min[\frac{1}{maxD_i}, \frac{N}{\sum_i D_i}]$. While the database server in this example represents an open class queing system, the second part of the equation is unspecified. Therefore $X_0^{max} = \frac{1}{maxD_i} = \frac{1}{0.050} = \frac{1000}{50} = 20$.

$\boxed{X_0^{max} = 20}$

4.4 exercise 4.4

Given In this exercise the application server is being viewed as the customer domain, with each active process being the customer and an average think time equals to the processing time between database requests[2].

$$\begin{aligned} N_{AS} &= 150, \\ Z_{AS} &= 550 \text{ msec } = 0.55 \text{ sec.} \\ X_{AS} &= 120 \text{ req/sec} \end{aligned}$$

Requested Response time of the database server R_{DB}.

$\boxed{R_{DB} = 0.7 \text{ sec}}$

Solution

$$R_{DB} = \frac{N_{DB}}{X_{DB}} - Z = \frac{150}{120} - 0.55 = \frac{5}{4} - \frac{55}{100} = \frac{125-55}{100} = \frac{70}{100} = 0.7 \text{ sec}$$

4.5 exercise 4.5

Given The motivating example plus the data of example 4.5 [3]

Requested

1. Aggregate the two classes into one
2. Which component is the bottleneck of the server?
3. What is the maximum throughput?

Solution The two classes are united into one class "Queries" by adding the arrival rate for both classes and by averaging the service demands. See table 8.

$$\begin{aligned} D_{CPU} &= \tfrac{0.050\times 16 + 0.150}{17} \approx 0.056 \\ D_{disk1} &= \tfrac{0.032\times 16 + 0.200}{17} \approx 0.042 \\ D_{disk2} &= \tfrac{0.016\times 16 + 0.100}{17} \approx 0.021 \end{aligned}$$

	Queries
Arrival Rate (tps)	17
Service Demands (sec)	
CPU	0.056
Disk 1	0.042
Disk 2	0.021

Figure 8: **Exercise 4.5**: Solution for the combined class

The bottleneck of the system appears to be the CPU because the service demand is highest. The maximum throughput will be:

$$X_0^{max} = \frac{1}{max\{D_i\}} = \frac{1}{0.056} \approx 18$$

[2] you need to refrase the question in order to answer the question.

[3] The text appears to contain an error: example 4.4 does not contain 2 classes, while example 4.5 does.

5 Chapter 5

5.1 Exercise 5.1

Given Information about monitoring tools on for example the Internet, books etc.

Requested A comparison between two tools for use in a performance analysis project.

Solution Tools found: nmon, sar.

Nmon The following description was copied from the web at address: `http://www.ibm.com/developerworks/aix/library/au-analyze_aix/index.html` on the 10^{th} of March 2008. The text was only adjusted to print headings and lists using LaTeX. No contents was changed.

```
Nigel Griffiths (nag@uk.ibm.com), pSeries Technical Support, IBM
04 Nov 2003
Updated 27 Feb 2006
```

This free tool gives you a huge amount of information all on one screen. Even though IBM doesn't officially support the tool and you must use it at your own risk, you can get a wealth of performance statistics.
Why use five or six tools when one free tool can give you everything you need?

Usage notes This nmon tool is NOT OFFICIALLY SUPPORTED. No warrantee is given or implied, and you cannot obtain help with it from IBM. If you have a question on nmon, please go on the Performance Tools Forum site (see Resources) so that others can find and benefit from the answers. To protect your email address from junk mail, you need to create a USER ID first (takes 20 seconds at most).
The nmon tool runs on:

- AIX 4.1.5, 4.2.0 , 4.3.2, and 4.3.3 (nmon Version 9a: This version is functionally established and will not be developed further.)
- AIX 5.1, 5.2, and 5.3 (nmon Version 10: This version now supports AIX 5.3 and POWER5 processor-based machines, with SMT and shared CPU micro-partitions.)
- Linux SUSE SLES 9, Red Hat EL 3 and 4, Debian on pSeries p5, and OpenPower
- Linux SUSE, Red Hat, and many recent distributions on x86 (Intel and AMD in 32-bit mode)
- Linux SUSE and Red Hat on zSeries or mainframe

The nmon tool is updated roughly every six months, or when new operating system releases are available. To place your name on the e-mail list for updates, contact Nigel Griffiths.
Use this tool together with nmon analyser (see Resources), which loads the nmon output file and automatically creates dozens of graphs.

Introduction The nmon tool is designed for AIX and Linux performance specialists to use for monitoring and analyzing performance data, including:

- CPU utilization
- Memory use
- Kernel statistics and run queue information
- Disks I/O rates, transfers, and read/write ratios
- Free space on file systems
- Disk adapters
- Network I/O rates, transfers, and read/write ratios
- Paging space and paging rates
- CPU and AIX specification
- Top processors
- IBM HTTP Web cache
- User-defined disk groups
- Machine details and resources
- Asynchronous I/O – AIX only
- Workload Manager (WLM) – AIX only
- IBM TotalStorage Enterprise Storage Server (ESS) disks – AIX only
- Network File System (NFS)
- Dynamic LPAR (DLPAR) changes – only pSeries p5 and OpenPower for either AIX or Linux

Also included is a new tool to generate graphs from the nmon output and create .gif files that can be displayed on a Web site.

Sar unix performance utility Another tool from the unix domain is the sar system utility. Sar extracts indicated performance measures using command line switches and options. Default is the CPU utilization. When indicated one can also track memory, disk, page-faults and network traffic. On request SAR will also write to a file or interpret result from a file. Sar measures all internal performance counters, except the wait time for remote access (to files or otherwise).
The following is part of the man page of SAR from a unix system, with most of the performance counters.

```
sar Command


 Purpose

 Collects, reports, or saves system activity information.
```

Syntax

```
/usr/sbin/sar [ {  -A | [  -a ] [  -b ] [  -c ] [  -k ] [  -m ]
[  -q ] [  -r ] [  -u ] [  -v ] [  -w ] [  -y ] } ] [  -e
hh [ :mm [ :
ss ] ] ] [  -fFile ] [  -iSeconds ] [  -oFile ]
[  -shh [ :mm [ :ss ] ] ] [ Interval [ Number ] ]
```

Description

The sar command writes to standard output the contents of selected cumulative activity counters in the operating system. The accounting system, based on the values in the Number and Interval parameters, writes information the specified number of times spaced at the specified intervals in seconds. The default sampling interval for the Number parameter is 1 second. The collected data can also be saved in the file specified by the -o File flag.

The sar command extracts and writes to standard output records previously saved in a file. This file can be either the one specified by the -f flag or, by default, the standard system activity daily data file, the /var/adm/sa/sadd file, where the dd parameter indicates the current day.

You can select information about specific system activities using flags. Not specifying any flags selects only system unit activity. Specifying the -A flag selects all activities.

The default version of the sar command (CPU utilization report) might be one of the first facilities the user runs to begin system activity investigation, because it monitors major system resources. If CPU utilization is near 100 percent (user + system), the workload sampled is CPU-bound. If a considerable percentage of time is spent in I/O wait, it impliesthat CPU execution is blocked waiting for disk I/O. The I/O may be required file accesses or it may be I/O associated with paging due to a lack of sufficient memory.

Note: The time the system spends waiting for remote file access is not accumulated in the I/O wait time. If CPU utilization and I/O wait time for a task are relatively low, and the response time is not satisfactory, consider investigating how much time is being spent waiting for remote I/O. Since no high-level command provides statistics on remote I/O wait, trace data may be useful in observing this.

If multiple samples and multiple reports are desired, it is convenient to specify an output file for the sar command. Direct the standard output data from the sar command to /dev/null and run the sar command as a background process. The syntax for this

is:
<snip a portion of the description>

dirblk/s Number of 512-byte blocks read by the directory search routine to locate a directory entry for a specific file.

iget/s Calls to any of several i-node lookup routines that support multiple file system types. The iget routines return a pointer to the i-node structure of a file or device.

lookuppn/s Calls to the directory search routine that finds the address of a v-node given a path name.

-b Reports buffer activity for transfers, accesses, and cache (kernel block buffer cache) hit ratios per second. Access to most files in AIX bypasses kernel block buffering, and therefore does not generate these statistics. However, if a program opens a block device or a raw character device for I/O, traditional access mechanisms are used making the generated statistics meaningful. The following values are accepted:

bread/s, bwrit/s Reports the number of block I/O operations. These I/Os are generally performed by the kernel to manage the block buffer cache area, as discussed in the description of the lread/s value.

lread/s, lwrit/s Reports the number of logical I/O requests. When a logical read or write to a block device is performed, a logical transfer size of less than a full block size may be requested. The system accesses the physical device units of complete blocks and buffers these blocks in the kernel buffers that have been set aside for this purpose (the block I/O cache area). This cache area is managed by the kernel, so that multiple logical reads and writes to the block device can access previously buffered data from the cache and require no real I/O to the device. Application read and write requests to the block device are reported statistically as logical reads and writes. The block I/O performed by the kernel to the block device in management of the cache area is reported as block reads and block writes.

pread/s, pwrit/s Reports the number of I/O operations on raw devices. Requested I/O to raw character devices is not buffered as it is for block devices. The I/O is performed to the device directly.

\%rcache, \%wcache Reports caching effectiveness (cache hit percentage). This percentage is calculated as [(100)x(lreads - breads)/(lreads)].

-c Reports system calls. The following values are accepted:

exec/s, fork/s Reports the total number of fork and exec system

calls.

sread/s, swrit/s Reports the total number of read/write system calls.

rchar/s, wchar/s Reports the total number of characters transferred by read/write system calls.

scall/s Reports the total number of system calls.

Note: The sar command itself can generate a considerable number of reads and writes depending on the interval at which it is run. Run the sar statistics without the workload to understand the sar command's contribution to your total statistics.

-e hh[:mm[:ss]] Sets the ending time of the report. The default ending time is 18:00.

-f File Extracts records from File (created by -o File flag). The default value of the File parameter is the current daily data file, the /var/adm/sa/sadd file.

-i Seconds Selects data records at seconds as close as possible to the number specified by the Seconds parameter. Otherwise, the sar command reports all seconds found in the data file.

-k Reports kernel process activity. The following values are accepted:

kexit/s Reports the number of kernel processes terminating per second.

kproc-ov/s Reports the number of times kernel processes could not be created because of enforcement of process threshold limit.

ksched/s Reports the number of kernel processes assigned to tasks per second.

-m Reports message and semaphore activities per second. The following values are accepted:

msg/s Reports the number of IPC message primitives.

sema/s Reports the number of IPC semaphore primitives.

-o File Saves the readings in the file in binary form. Each reading is in a separate record and each record contains a tag identifying the time of the reading.

-q Reports queue statistics. The following values are accepted:

runq-sz Reports the average number of processes in the run

queue.

\%runocc Reports the percentage of the time the run queue is occupied.

swpq-sz Reports the average number of processes waiting to be paged in.

\%swpocc Reports the percentage of the time the swap queue is occupied.

Note: A blank value in any column indicates that the associated queue is empty.

-r Reports paging statistics. The following values are accepted:

cycle/s Reports the number of page replacement cycles per second.

fault/s Reports the number of page faults per second. This is not a count of page faults that generate I/O, because some page faults can be resolved without I/O.

slots Reports the number of free pages on the paging spaces.

odio/s Reports the number of nonpaging disk I/Os per second.

-s hh[:mm[:ss]] Sets the starting time of the data, causing the sar command to extract records time-tagged at, or following, the time specified. The default starting time is 08:00.

-u Reports system unit activity. The following values are accepted:

%idle Reports the percentage of time the system unit was idle with no outstanding disk I/O requests.

%sys Reports the percentage of time the system unit spent in execution at the system (or kernel) level.

%usr Reports the percentage of time the system unit spent in execution at the user (or application) level.

%wio Reports the percentage of time the system unit was idle waiting for disk I/O to complete.

Note: The sar command reports system unit activity if no other specific content options are requested.

-v Reports status of the process, i-node, and file tables. The following values are accepted:

file-ov, proc-ov Reports the number of times per second the file

```
or process table overflowed.

file-sz, inod-sz, proc-sz  Reports the number of entries in use
for each table.

Note: The I-node System Tables are dynamically allocated when
inod-sz is reaching maximum, so there is never a possibility of
an overflow.

-w Reports system switching activity.  The following value is ac-
cepted:

pswch/s Process switches per second. The following values are ac-
cepted:

-y Reports tty device activity per second.

canch/s Reports tty canonical input queue characters.

mdmin/s Reports tty modem interrupts.

outch/s Reports tty output queue characters.

rawch/s Reports tty raw input queue characters.

revin/s Reports tty receive interrupts.

xmtin/s Reports tty transmit interrupts.
```

5.2 Exercise 5.2

Given $n = 100, T = 60$ sec

Requested The average number of transactions concurrently executed.

Solution

$$\overline{N} = \frac{\sum_{i=1}^{n} e_i}{T} \approx \frac{382}{60} \approx 6.4$$

5.3 Exercise 5.3

Given The spreadsheet `Ch05-Ex-BW.xls` contains data of 76 CPU and IO timings.

Requested

1. Calculate basic statistics
2. Draw a box and whiskers diagram
3. Visually determine clusters and characterize centroids

Solution *Calculate basic statistics*

	CPU time	No I/Os
Mean	29.9	37.0
Std dev	33.5	28.0
Var	1119.8	782.6
CV	1.118	0.754
	CPU time	No I/Os
Minimum	1.56	7
Q1	7.42	17
Q2	12.61	23
Q3	28.27	45.25
Maximum	101.18	99
Range	99.62	92
Largest	101.18	99
Smallest	1.56	7
Sum	2275.7	2818

Draw a box and whikers diagram
Using the data the package "`R`" drew a boxplot (figure 9). The points outside of the box and whiskers are outliers, i.e. are more than $\frac{3}{2}$ outside of the box drawn between Q_1 and Q_3. The amount of outliers is a good indication, that more than one class is concerned.

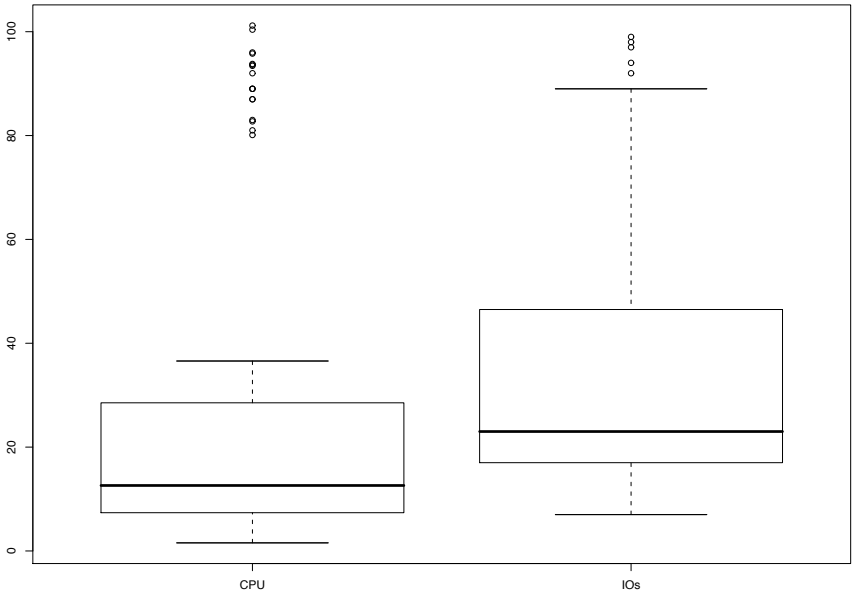


Figure 9: **exercise 5.3**:Boxplot as calculated by `R (boxplot)` and plotted by `R (plot)`

Determine clusters

```
K-means clustering with 5 clusters of sizes 8, 7, 37, 10, 14

Cluster means:
       CPU      IOs
1  9.92375 89.37500
2 93.03714 16.42857
3 14.06568 17.81081
4 88.27900 78.30000
5 10.12786 39.00000

Clustering vector:
 [1] 3 1 3 5 3 5 3 1 3 3 5 3 1 1 4 5 5 3 3 3 5 3 5 3 3 3 3 3 3 5 3 2 2 3 3 3 3 2
[39] 3 3 3 4 2 3 3 4 3 1 1 5 5 3 3 3 2 2 3 3 4 2 4 4 4 5 4 4 4 5 3 1 1 5 3 3 3 5

Within cluster sum of squares by cluster:
[1]  445.0972  279.8432 3849.0538 1745.6235  795.0892

Available components:
[1] "cluster"  "centers"  "withinss" "size"
```

Figure 10: **exercise 5.3**: output of cluster analysis when printing the cluster

The packages `R (K-means)` calculated 5 clusters and `R (plot)` plotted the result. See figure 11[4].

5.4 Exercise 5.4

Given Same data as for exercise 5.3 plus: $T = 15, U_{CPU} = 0.35, U_{disk} = 0.80$.

Requested

1. Overall throughput.
2. Throughput per classes.
3. Service demands for CPU and disk per class.

Solution

Overall throughput As there are 76 observations running for 15 seconds the throughput X_0 is:

$$X_0 = \frac{C_0}{T} = \frac{76}{15} \approx 5.0\emptyset \text{ tps}$$

[4] A later version of *R* could not reconstruct this sequence of events or even the same set of clusters. It appears that the algorithm for finding clusters was altered. Because the sum of squares was larger, the result is worse. I sent both results to the R-project for examination.

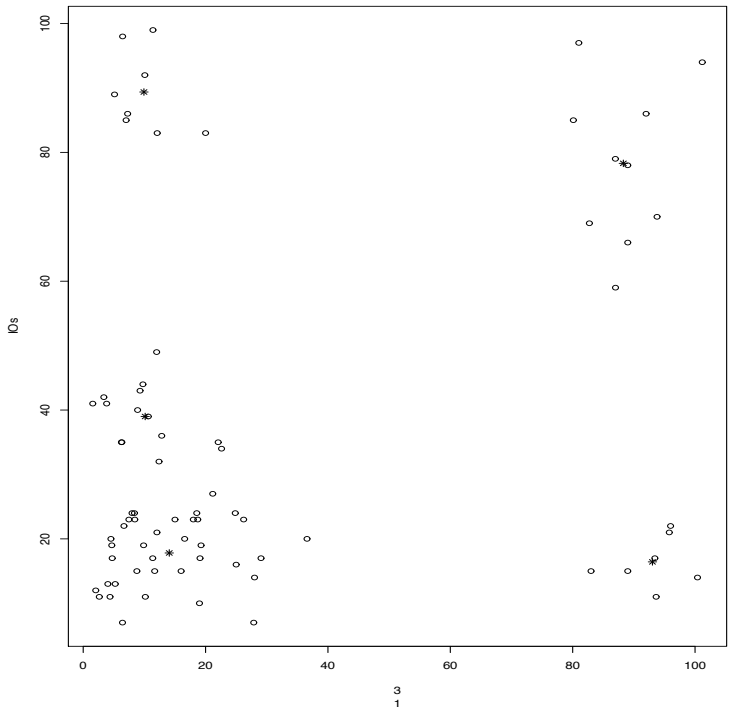


Figure 11: **exercise 5.3**:Cluster as calculated by `K-means` and plotted by `R`.

Throughput per class Through calculation with `R (K-means)` the clusters were determined with 5 clusters of sizes 8, 7, 37, 10, 14. Each class is taken to be equal to a cluster, because the CPU-IO characteristics are comparable, so we get $X_{i,r}$ with $r \in 1, 2, 3, 4, 5$. The throughput per class $X_{CPU,r}$ is equal to the number of class members as found by `R (K-Means)` divided by the observation time (15 seconds).

$$\begin{aligned} X_{0,1} &= \tfrac{8}{15} \approx 0.53 \\ X_{0,2} &= \tfrac{7}{15} \approx 0.46 \\ X_{0,3} &= \tfrac{37}{15} \approx 2.46 \\ X_{0,4} &= \tfrac{10}{15} \approx 0.66 \\ X_{0,5} &= \tfrac{14}{15} \approx 0.93 \end{aligned}$$

Service demands for CPU and disk per class The definition of service demand per resource is :

$$D_{i,r} = \frac{U_{i,r}}{X_{0,r}}$$

For the CPU this is

$$D_{CPU,r} = \frac{U_{CPU,r}}{X_{0,r}}$$

Estimating the utilization per class $U_{CPU,r}$ can be done by calculating the fraction of the total utilization that is used by the class as follows:

$$U_{CPU,r} = \frac{t_{CPU,r}}{t_{CPU}} \times U_{CPU}$$

$$U_{CPU,1} = \frac{79,39}{2275,73} \times 0.35 \approx 0.012 = 1.2\%$$
$$U_{CPU,2} = \frac{651,26}{2275,73} \times 0.35 \approx 0.100 = 10.0\%$$
$$U_{CPU,3} = \frac{520,48}{2275,73} \times 0.35 \approx 0.080 = 8.0\%$$
$$U_{CPU,4} = \frac{882,80}{2275,73} \times 0.35 \approx 0.136 = 13.6\%$$
$$U_{CPU,5} = \frac{141,81}{2275,73} \times 0.35 \approx 0.022 = 2.2\%$$

cluster i	CPU_r	N_r	$X_{0,r}$	$U_{CPU,r}$	$D_{CPU,r} = \frac{U_{CPU,r}}{X_{0,r}}$
1	79.39	8	$\frac{8}{15}$	0.012	0.0225
2	651.26	7	$\frac{7}{15}$	0.100	0.2143
3	520.48	37	$\frac{37}{15}$	0.080	0.0324
4	882.79	10	$\frac{10}{15}$	0.136	0.2040
5	141.81	14	$\frac{14}{15}$	0.022	0.0236
total	2275.73	76	$\frac{76}{15}$	0.350	0.0691

Figure 12: **Exercise 5.4**: Service demands per class for the CPU including interim results.

where U_{CPU} is given: 35%, $t_{CPU,r}$ is the CPU time for the class r and t_{CPU} is the total CPU time for all classes. Using the resulting $U_{CPU,r}$ and the service demand law we can calculate the service demands per class for the CPU (figure 12).
In the same manner as for CPU, using the number of I/O's for each cluster we can calculate the $U_{disk,r}$ (figure 13).

5.5 Exercise 5.5

Given The calculated and given data from exercise 5.3 and 5.4

Requested Solve the model using `OpenQN.xls`.

Solution The model was input into JMT[5] (figure 14) and solved in JMT (figure 15).
The utilization values per class and the arrival rates from solving the model coincide with the calculated values[6].

[5]The Java Modeling Tool, a GPL-ed application by M.Bertoli, G.Casale, G.Serazzi. Java Modelling Tools: an Open Source Suite for Queueing Network Modelling and Workload Analysis. Proceedings of QEST 2006 Conference, Riverside, US, Sep 2006, 119-120, IEEE Press.

[6]this exercise is a good check of the calculations in exercises 5.3 and 5.4.

$$U_{disk,1} = \frac{715}{2818} \times 0.80 \approx 0.203$$
$$U_{disk,2} = \frac{115}{2818} \times 0.80 \approx 0.033$$
$$U_{disk,3} = \frac{659}{2818} \times 0.80 \approx 0.187$$
$$U_{disk,4} = \frac{783}{2818} \times 0.80 \approx 0.222$$
$$U_{disk,5} = \frac{546}{2818} \times 0.80 \approx 0.155$$

cluster i	IO_r	N_r	$X_{disk,r}$	$U_{disk,r}$	$D_{disk,r} = \frac{U_{disk,r}}{X_{0,r}}$
1	715	8	$\frac{8}{15}$	0.203	0,381
2	115	7	$\frac{7}{15}$	0.033	0,071
3	659	37	$\frac{37}{15}$	0.187	0,076
4	783	10	$\frac{10}{15}$	0.222	0,333
5	546	14	$\frac{14}{15}$	0.155	0,166
total	2818	76	$\frac{76}{15}$	0.800	0,158

Figure 13: **Exercise 5.4**: Calculation of the service demands per disk including interim results.

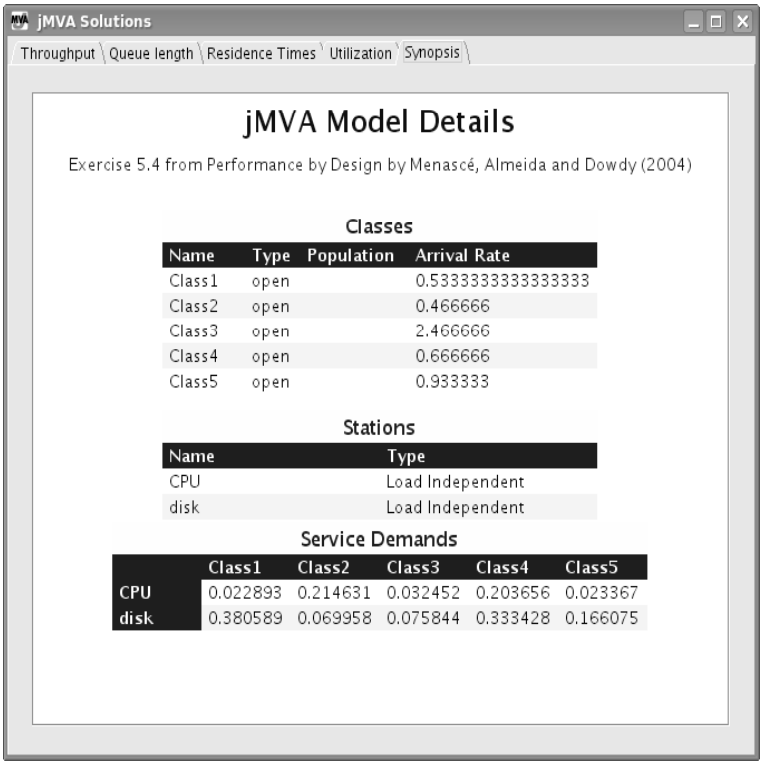


Figure 14: **Exercise 5.5**: the input parameters for the model.

5.6 Exercise 5.6

Given Resources: CPU, disk1 (D_1), disk2 (D_2), disk3 (D_3).

$$
\begin{aligned}
&T = 1 \text{ hour} = 3600 \text{ sec} \\
&\text{customer classes A, B and C} \\
&U^t_{CPU} = 0.82 \quad C_{0,A} = 2200 \\
&U^t_{D_1} = 0.28 \quad C_{0,B} = 4000 \\
&U^t_{D_2} = 0.20 \quad C_{0,C} = 1000 \\
&U^t_{D_3} = 0.35
\end{aligned}
$$

Requested Determine the service demands for this workload and comment. Justify assumptions by giving pros and cons.

Solution

Pros and cons The usage characteristics of the transaction classes are not defined. Therefore, the only way to apportion the utilization per resource among the classes is by number of transactions per class.
The disadvantage of this approach is that the reliability of the utilization depends on class characteristics that may differ per class. Each class may have very different CPU and IO characteristics in which case this choice of apportionment would not be appropriate.
For the given data, however, this can not be recognized.

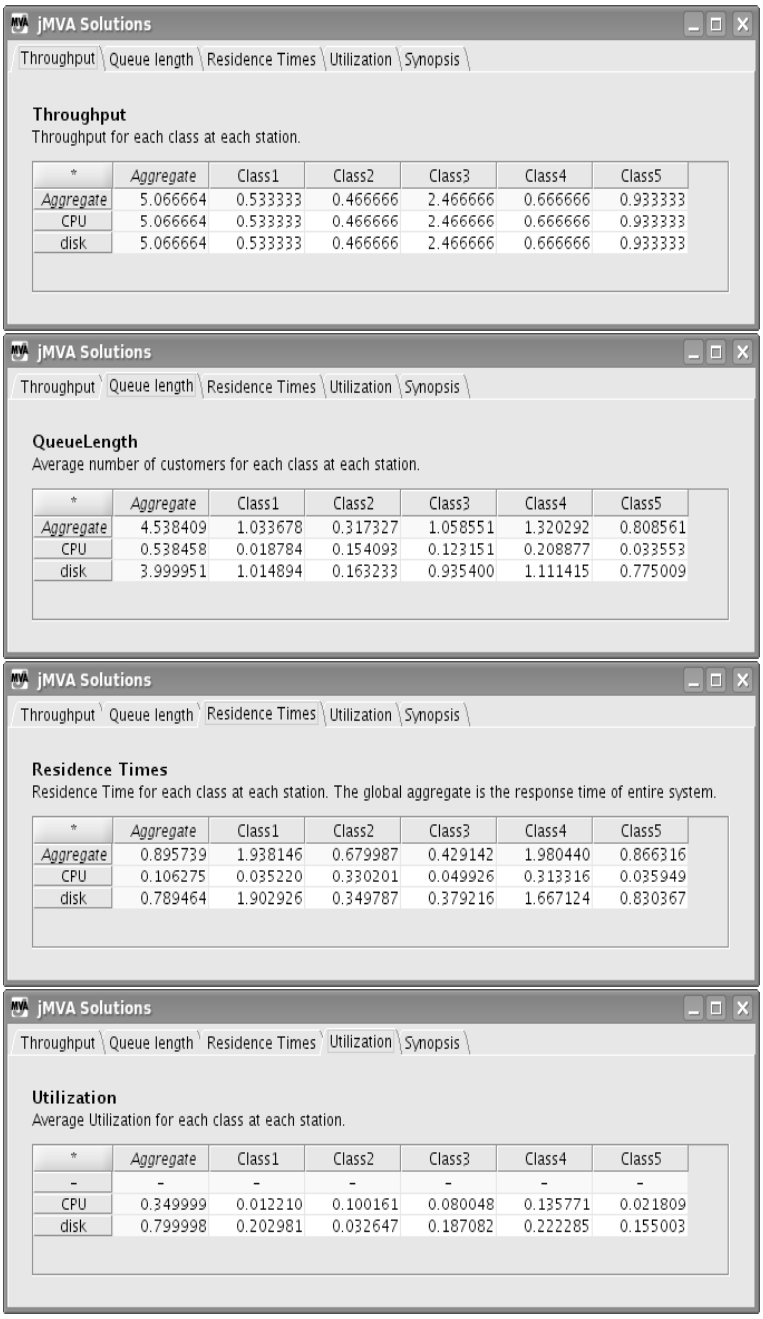


Figure 15: **Exercise 5.5**: the output for the model.

$$f_A = \frac{2200}{7200} \approx 0.30\not{5}$$
$$f_B = \frac{4000}{7200} \approx 0.55\not{5}$$
$$f_C = \frac{1000}{7200} \approx 0.13\not{8}$$

	U_r	$U_{i,A}$	$U_{i,B}$	$U_{i,C}$
f_i		$0.30\not{5}$	$0.55\not{5}$	$0.13\not{8}$
CPU	0.82	0.25056	0.45556	0.11389
D_1	0.28	0.08556	0.15556	0.03889
D_2	0.20	0.06111	0.11111	0.02778
D_3	0.35	0.10694	0.19444	0.04861

Figure 16: **Exercise 5.6**: Utilization per resource and per class A, B and C

determining service demands In order to calculate the utilization per class $U_{CPU,i}$, $U_{D_1,i}$, $U_{D_2,i}$ and $U_{D_3,i}$ we calculate a factor per class f_A, f_B and f_C using the number of transactions per class and the total number of transactions. These factors are then used to apportion resources to the classes (figure 16). We use the service demand law[7]: $D_{i,r} = \frac{U_i}{X_{0,r}}$ and basic definition of throughput $X_{0,r} = \frac{C_{i,r}}{T}$ to obtain service demands (figure 17).

	$D_{i,A}$	$D_{i,B}$	$D_{i,C}$
$X_{0,r}$	$0.6111\not{1}$	$1.1111\not{1}$	$0.2777\not{7}$
CPU	0.41000	0.41000	0.41000
D_1	0.14000	0.14000	0.14000
D_2	0.10000	0.10000	0.10000
D_3	0.17500	0.17500	0.17500

Figure 17: **Exercise 5.6**: Service demands per resource and per class A, B and C

Comments on service demands The service demands for all classes are the same for each resource! This leaves only two of the three reasons for a multiclass QN, i.e. different types of workload like batch versus online, or different service level objectives. If none of these two factors apply, then the model should be changed to a single class QN model, as there is not justification for a multiclass QN model[8].

[7] formula 3.2.14 on page 75 of the book

[8] See par. 2.4 page 41 in the book of Menascé for the justification of a multiclass QN.

5.7 Exercise 5.7

Given

transactions: phone orders from customers
resources: CPU, disk A, disk B
disk A: contains only customer data
disk B: paging device
class (1): transactions (order entry)
class (2): system overhead
$T = 900$ sec
$U_{cpu} = 0.65$
$U_{disk_A} = 0.20$
$U_{disk_B} = 0.35$
$C_{0,trans} = 1800$
$C_{pagein} = 22500$
$C_{pageout} = 10000$
CPU time user transactions: 567 sec
Cpu time for 1 page-in: 0.0015 sec
Cpu time for 1 page-out: 0.0042 sec

Requested Find the input parameters for the model.

Solution See page 56 for examples of QN model input parameters in tables 2.4 and 2.5.

CPU utilization The utilization for the CPU can be apportioned using the CPU time for the transactions (567 seconds) and the total user CPU time: $U_{cpu} \times T = 0.65 \times 900 = 585$. This leaves $585 - 567 = 18$ seconds overhead. The factors to divide CPU utilization are:

$f_{trans} = 0.9692$

$$f_{trans} = \frac{567}{585} \approx 0.9692$$
$$f_{overhead} = \frac{18}{585} \approx 0.0308$$

$f_{overhead} = 0.0308$

These factors can be used to apportion $U_{i,trans}$ and $U_{i,overhead}$.
Disk A is used for customer data. The utilization is thus fully attributable to the user transaction. Disk B is used for paging in and out. The activity of page-in is part of the transaction workload while the page-out is part of the system overhead. Thus an apportionment of utilization between transaction and overhead must be determined. The two determining factors are the number of pages in and out and the speed difference between reading and writing. As the speed factors are unknown, I will disregard the difference and assume they are equal. This leaves only the number of page-in and page-out to be considered. Using these numbers, the following factors can be calculated:

$g_{trans} = 0.69231$

$$g_{trans} = \frac{22500}{32500} \approx 0.69231$$
$$g_{overhead} = \frac{10000}{32500} \approx 0.30769$$

$g_{overhead} = 0.30769$

We calculate the service demands per class per resource using the service demand law[9] $D_{i,r} = \dfrac{U_{i,r} \times T}{C_{0,r}}$ (figure 19).
Finally, the QN model is represented in figure 20.

[9] formula 3.2.3 on page 67 in the Menascé book

	factor	U_i	$U_{i,trans}$	$U_{i,overhead}$
U_{cpu}	f_r	0.65	0.63000	0.02000
U_{disk_A}	-	0.20	0.20000	0
U_{disk_B}	g_r	0.35	0.24231	0.10769

Figure 18: **Exercise 5.7**: Apportioning utilization per resource to the classes

$$
\begin{aligned}
D_{cpu,trans} &= \tfrac{0.63\times 900}{1800} = \tfrac{0.63}{2} = 0.315 \text{ sec} \\
D_{disk_A,trans} &= \tfrac{0.20\times 900}{1800} = \tfrac{0.20}{2} = 0.10 \text{ sec} \\
D_{disk_B,trans} &= \tfrac{0.24231\times 900}{1800} = \tfrac{0.24231}{2} = 0.12155 \text{ sec}
\end{aligned}
$$

$$
\begin{aligned}
D_{cpu,overhead} &= \tfrac{0.02\times 900}{1800} = \tfrac{0.02}{2} = 0.01 \text{ sec} \\
D_{disk_A,overhead} &= 0 \text{ sec} \\
D_{disk_B,overhead} &= \tfrac{0.10769\times 900}{1800} = \tfrac{0.10769}{2} = 0.053845 \text{ sec}
\end{aligned}
$$

Figure 19: **Exercise 5.7**: Calculating $D_{i,\text{trans}}$ and $D_{i,\text{overhead}}$.

5.8 Exercise 5.8

A program must be written to make this exercise: todo.

5.9 Exercise 5.9

Given A database server with classes trivial, medium and complex. Resources are one CPU and one disk.

$$T = 3600 \text{ sec}, U_{cpu} = 0.55, U_{disk} = 0.85$$

Requested

- Find the service demands for CPU and disk for each class.
- Assume $\lambda_{medium} = 1.6$ tps and $\lambda_{complex} = 0.6$ tps are fixed. Make a graph with response times curves (3x) as a function of $\lambda_{trivial}$.
- repeat the above question for adding a new disk with balanced usage of both disks.

K = 2, R = 3 Open QN				Queue type/Sched. Discipline		
				LI/PS	LI/FCFS	LI/FCFS
Class(r)	Type	λ_r (tps)	N_r	$D_{cpu,r}$ (sec)	$D_{disk_A,r}$ (sec)	$D_{disk_B,r}$ (sec)
1 transactions	open	2	-	0.315	0.10	0.12155
2 system overhead	open	2	-	0.01	0	0.053845

Figure 20: **Exercise 5.7**: Input parameters for QN model

Table 5.10	Trivial	Medium	Complex	total
Physical I/Os	10	20	40	70
CPU seconds	840	560	252	1652
#transactions executed	10500	5600	2100	18200

Figure 21: **Exercise 5.9**: data measured

Solution Question 1 and 2 To calculate the service demand per class we use the service demand law:

$$D_{i,r} = \frac{U_{i,r}}{X_{0,r}} \tag{5.9.1}$$

The two main variables are throughput and utilization per class. Both are to be determined. The throughput per class is equal to $\frac{C_{0,r}}{T}$ (figure 22).

Throughput	**Trivial**	**Medium**	**Complex**
$X_{0,r}$	$\frac{10500}{18200} = 2.91\not{0}$	$\frac{5600}{18200} = 1.5\not{7}$	$\frac{2100}{18200} = 0.58\not{3}$

Figure 22: **Exercise 5.9, Q1**: Throughput per class

The CPU seconds measured per class lead to an apportionment of CPU utilization per class. The physical IO's measured per class lead to the apportionment of disk utilization. The factor $f_{i,r}$ can be calculated using the formula's: $f_{CPU,r} = \frac{\text{CPU seconds for class r}}{\text{total CPU seconds for all classes}}$ and $f_{disk,r} = \frac{\text{Physical IO's for class r}}{\text{total physical IO's for all classes}}$ (table 23).

$$\begin{array}{ll} f_{CPU,trivial} & = \frac{840}{1652} \approx 0.508475 \\ f_{CPU,medium} & = \frac{560}{1652} \approx 0.338983 \\ f_{CPU,complex} & = \frac{252}{1652} \approx 0.152542 \\ f_{disk,trivial} & = \frac{10}{70} \approx 0.142857 \\ f_{disk,medium} & = \frac{20}{70} \approx 0.285714 \\ f_{disk,complex} & = \frac{40}{70} \approx 0.571429 \end{array}$$

utilization factors	**Trivial**	**Medium**	**Complex**
CPU	0.508475	0.338983	0.152542
IO	0.142857	0.285714	0.571429

Figure 23: **Exercise 5.9, Q1**: Utilization factor per class

Using the factors $f_{i,r}$ we can estimate the utilization per class: $U_{i,r} = U_i \times f_{i,r}$ (figure 24).
Now with both throughput values per class and utilization per class calculated, we can estimate the service demands using equation 5.9.1 (figure 25).
The response time graph can be approximated using the calculated residence times per class (Figure 26).

Solution Question 3 Using an extra disk with the same capabilities and balancing the accesses to both disks, will on average cut the service times for IO's per disk in half, leaving the CPU times unaltered (figure 27). The graph for

$$
\begin{aligned}
U_{CPU,trivial} &\approx 0.55 \times 0.508475 \approx 0.279661 \\
U_{CPU,medium} &\approx 0.55 \times 0.338983 \approx 0.338983 \\
U_{CPU,complex} &\approx 0.55 \times 0.152542 \approx 0.152542 \\
U_{disk,trivial} &\approx 0.85 \times 0.142857 \approx 0.121429 \\
U_{disk,medium} &\approx 0.85 \times 0.285714 \approx 0.242857 \\
U_{disk,complex} &\approx 0.85 \times 0.571429 \approx 0.485714
\end{aligned}
$$

Utilization	**Trivial**	**Medium**	**Complex**
CPU	0.279661	0.338983	0.152542
disk	0.121429	0.242857	0.485714

Figure 24: **Exercise 5.9, Q1**: Utilization per class

$$
\begin{aligned}
D_{CPU,trivial} &\approx \frac{0.279661}{2.916\not{6}} \approx 0.095884 \\
D_{CPU,medium} &\approx \frac{0.338983}{1.55\not{9}} \approx 0.119855 \\
D_{CPU,complex} &\approx \frac{0.152542}{0.583\not{3}} \approx 0.143826 \\
D_{disk,trivial} &\approx \frac{0.121429}{2.916\not{6}} \approx 0.041633 \\
D_{disk,medium} &\approx \frac{0.242857}{1.55\not{9}} \approx 0.156122 \\
D_{disk,complex} &\approx \frac{0.485714}{0.583\not{3}} \approx 0.832653
\end{aligned}
$$

Service demands	**Trivial**	**Medium**	**Complex**
CPU	0.095884	0.119855	0.143826
IO	0.041633	0.156122	0.832653

Figure 25: **Exercise 5.9, Q1**: Service demands

residence per class is in figure 28. As you can see, the response time for the complex transactions increase more moderately than in the case of 1 disk.

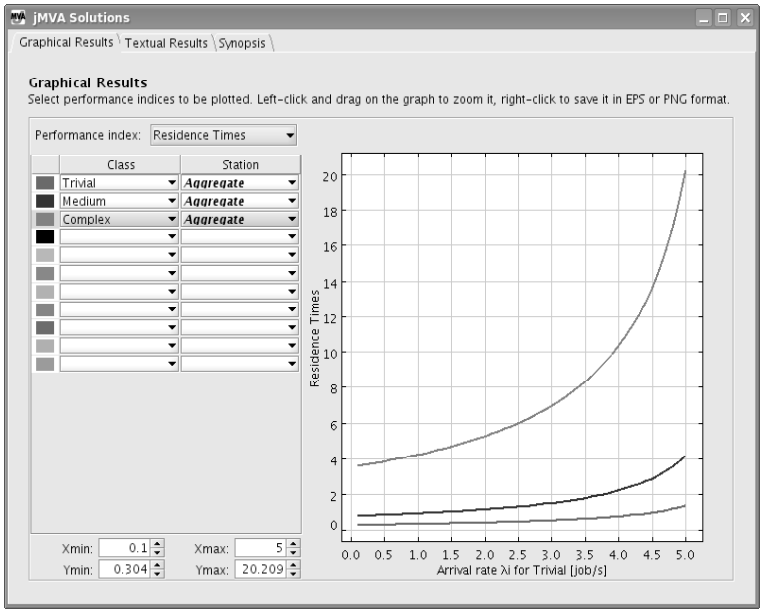


Figure 26: **Exercise 5.9, Q2**: draw the response times curves as a function of trivial arrival times

Service demands	**Trivial**	**Medium**	**Complex**
CPU	0.095884	0.119855	0.143826
disk 1	0, 020816	0, 078061	0, 416327
disk 2	0, 020816	0, 078061	0, 416327

Figure 27: **Exercise 5.9, Q3**: Service demands

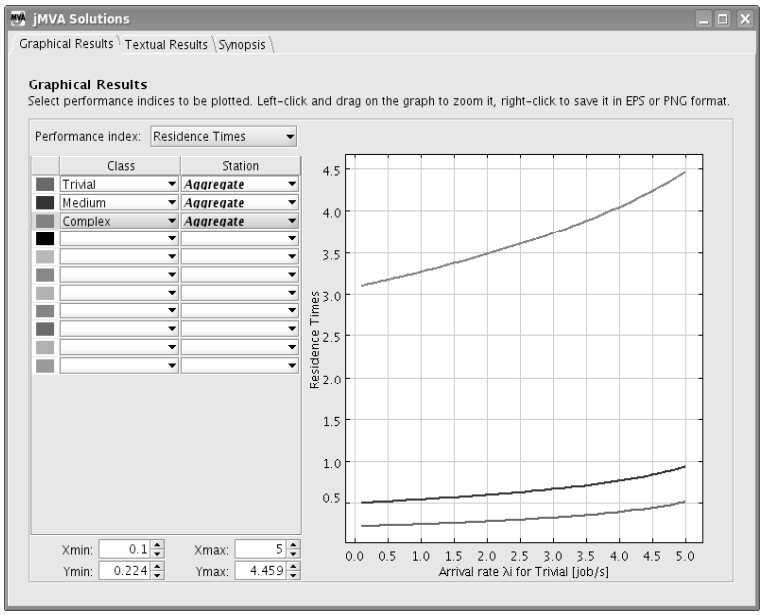


Figure 28: **Exercise 5.9, Q3**: The response times as a function of trivial arrival times, using 2 balanced disks

6 Chapter 6

6.1 Exercise 6.1

Given Data in WSData.XLS workbook

Requested basic statistics for the set of all files downloaded (PDF and ZIP), plus the coefficient of variation.

Solution

statistic	result
Mean	836
Median	1049
Standard deviation	389.4
Sample variance	151773.7
Range	999
Minimum	300
Maximum	1300
Sum	835833
Count	1000
1/2 95% Confidence interval	24.13
Coefficient of variance	0.466

Given a coefficient of variance of about 0.47. This indicates a standard deviation that is quite large in comparison to the mean. As 0.47 is considerably more than 0.25[10], one could guess that there probably more clusters are involved. Through cluster analysis one could find the clusters defined originally (PDF and ZIP).

6.2 Exercise 6.2

Given Use the results in the previous problem.

Requested

1. Construct a single class model
2. Calculate new service demands
3. Solve the new model
4. Plot throughput and average download times
5. Assess the 'error' made by assuming a single class model in a multiclass system
6. What would be an appropriate SLA?

Solution The concurrency level for the sample as a single class is calculated first[11]

$$\overline{N} = \frac{\sum_{i=1}^{1001} e_i}{T} = \frac{731.5 + 3207.7}{200} = \frac{3939.2}{200} = 19.696 \approx 20$$

$$\boxed{\overline{N} = 20}$$

[10] page 167 of the book, note in second paragraph
[11] using Eq (5.7.27) on page 152 of the book

This confirms the original calculation using two classes with respectively 4 and 16 files being processes concurrently.

Balanced case The service demands per class are equally devided among the disks. Using the same measurements as the book[12] the service demands to be calculated will be equal to the weighted average of the PDF and ZIP service demands:

$$D_{i,PDF} \times n_{PDF} + D_{i,ZIP} \times n_{ZIP} = D_{i,PDF} \times 411 + D_{i,ZIP} \times 589$$

for each resource i.
The averaged service demand is then equal to

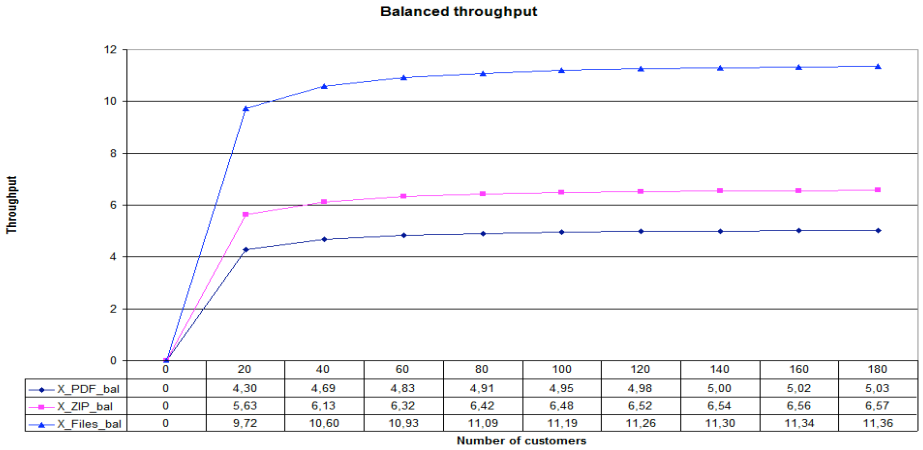
$$D_i = \frac{D_{i,PDF} \times n_{PDF} + D_{i,ZIP} \times n_{ZIP}}{n}$$

where $n = 1000$. See figure 29 for the service demands.

Resource	$D_{i,PDF}$	$D_{i,ZIP}$	D_i
	msec	msec	msec
CPU	39.4	120.8	87.3
disk 1	38.6	117.9	85.3
disk 2	38.6	117.9	85.3
disk 3	38.6	117.9	85.3
disk 4	38.6	117.9	85.3

Figure 29: **Exercise 6.2**: service demands in the balanced case

After solving the model and calculating the relevant values, the plots for the balanced case were generated (figures 30 and 31).



	0	20	40	60	80	100	120	140	160	180
X_PDF_bal	0	4,30	4,69	4,83	4,91	4,95	4,98	5,00	5,02	5,03
X_ZIP_bal	0	5,63	6,13	6,32	6,42	6,48	6,52	6,54	6,56	6,57
X_Files_bal	0	9,72	10,60	10,93	11,09	11,19	11,26	11,30	11,34	11,36

Figure 30: **Exercise 6.2** Balanced throughput for PDF, ZIP and files (=PDF+ZIP)

[12] on page 171

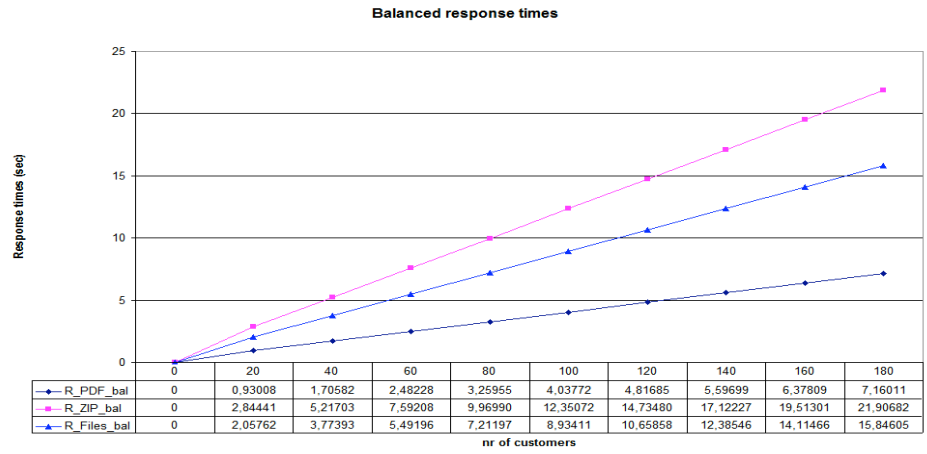


Figure 31: **Exercise 6.2** Balanced download times for PDF, ZIP and files (=PDF+ZIP)

Unbalanced case The service demands per class are *not* equally devided among the disks. Any calculation that does not consider this fact, will have additional error in the results. The reason in this case is, that the average of the total service demands for disk 1 and 2 are accumulated by 411 files and averaged over 1000 files. For disk 3 and 4 the ratio is 589 files against 1000 files.
The error caused by using one class will result in a deviation in the calculations
In this particular case, the service demands when averaged in the same way as in the balanced case leave each disk with roughly half[13] of its original service demand (figure 32).
These service demands when solved in a closed QN model containing one class in stead of two, lead to the results as shown in figures 33 and 34.

Resource	$D_{i,PDF}$	$D_{i,ZIP}$	D_i
	msec	msec	msec
CPU	39.4	120.8	87.3
disk 1	77.1	0	31.9
disk 2	77.1	0	31.9
disk 3	0	235.8	138.9
disk 4	0	235.8	138.9

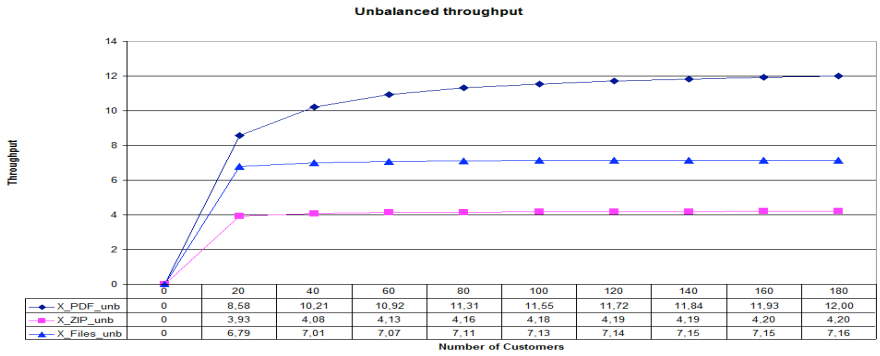
Figure 32: **Exercise 6.2**: service demands in the unbalanced case

Error assessment balanced case

Throughput The throughput appears to be about the sum of the original throughput as you can see in the graph (figure 30).

Download times The response time in the graph is almost exactly equal to the weighted average of original response times, which is suprising because of the difference in calculation compared to a weighted average.

[13] i.e. $\frac{411}{1000}$ or $\frac{589}{1000}$



	0	20	40	60	80	100	120	140	160	180
X_PDF_unb	0	8,58	10,21	10,92	11,31	11,55	11,72	11,84	11,93	12,00
X_ZIP_unb	0	3,93	4,08	4,13	4,16	4,18	4,19	4,19	4,20	4,20
X_Files_unb	0	6,79	7,01	7,07	7,11	7,13	7,14	7,15	7,15	7,16

Figure 33: **Exercise 6.2** Unbalanced throughput for PDF, ZIP and files (=PDF+ZIP)

Error assessment unbalanced case

Throughput The throughput is not the sum of the original throughput like it was in the balanced case and like you might expect, but is an average of the original throughputs. This means that the error in the unbalanced case is quite large. The explanation for this phenomenon .....[to be filled in after discussion] .....

Download times The response time in the graph is somewhere in the middle of the original response times as can be expected. However, unlike the balanced case, the deviation from the weighted average of response times is quite a bit larger. One would assess the error as being equal to the original error plus the weighted averaging error.

commenting the error in the Service Demand D_i In the second paragraph we calculated the new D_i using the formula $D_i = \frac{D_{i,PDF} \times n_{PDF} + D_{i,ZIP} \times n_{ZIP}}{n}$. Any error present, will be in the service demands $D_{i,PDF}$ and $D_{i,ZIP}$ as the numbers n_{PDF}, n_{ZIP} and n are exact. Naming the errors in the original service demands e_1 for PDF and e_2 for ZIP, the error in the combination could be expressed as $e_3 = \frac{411e_1 + 598e_2}{1000}$.
Conclusion would be, that, provided the classes use all resources in roughly the same way, the error does not need to be big compared to the original errors e_1 and e_2.
However, if one of the resources is used mainly by only one of the classes in the multiclass QN system, then this resource will contribute more error to the total error due to the imbalance.

response time SLA A sample that satisfies the original requirements of 7 and 20 seconds, but only just, will satisfy any requirements over 20 seconds, but not less. So, 20 seconds should be the requirements for the maximum response time. The response time requirements of 7 seconds for 411 PDF files and 20 seconds for 589 ZIP files can be translated into a weighted average response time for 1000 files: $\frac{7 \times 411 + 20 \times 589}{1000} = 14.657$.

max R = 20

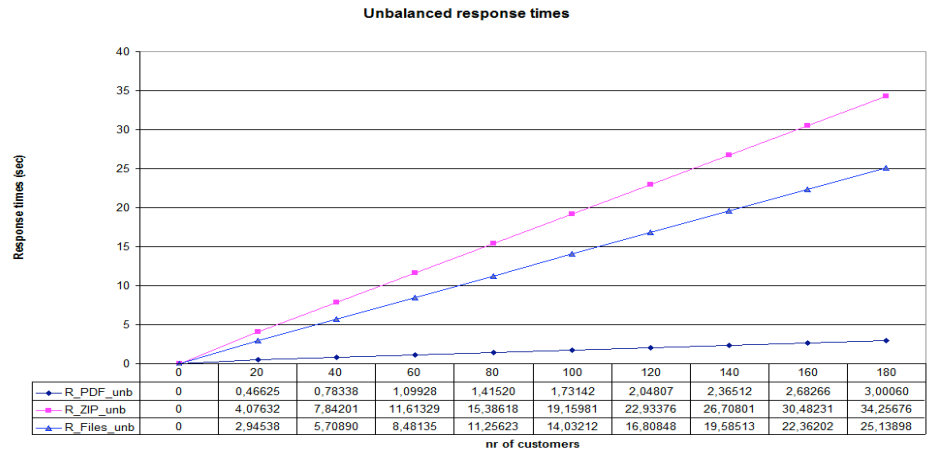


Figure 34: **Exercise 6.2** Unbalanced download times for PDF, ZIP and files (=PDF+ZIP)

This implies a maximum for the average of 15 seconds. max avg R = 15
In conclusion, any configuration that satisfies the original requirements, will also satisfy the combination of requirements[14]:

- a maximum of 20 seconds for all transactions
- a maximum of the average for all transactions of 15 seconds in a period of f.e. one day

6.3 Exercise 6.3

Given **WSData.xls**.

Requested The 25^{th} and 75^{th} percentile for the ZIP and PDF files. Also a boxplot for the ZIP and PDF files.

Solution Using excel and the command (for PDF)

```
PERCENTILE(B$2:B$412;0,25)
```

and

```
PERCENTILE(B$2:B$412;0,75)
```

are calculated (figure 35). Using the statistical package **R** both of the box and whisker diagrams are plotted (figures 36 and 37).

6.4 Exercise 6.4

Given **WSData.xls**.

Requested Compute a 90% confidence interval for the size of PDF and ZIP files.

[14]The other way around is more difficult to ascertain, because the one class QN model does not recognize PDF files. Therefore it can not determine the 7 second requirement for PDF files.

Percentile	PDF	ZIP
25^{th}	341	1083
75^{th}	416	1229

Figure 35: **Exercise 6.3**: The 25^{th} and 75^{th} percentiles for PDF and ZIP

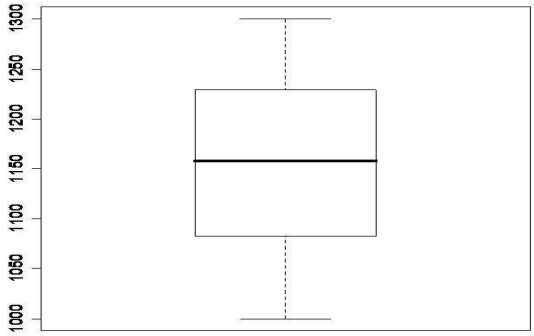


Figure 36: **Exercise 6.3** boxplot for the ZIP files

Solution Using the excel descriptive statistics of the Data Analysis package, the 90% confidence level was calculated for PDF (figure 38) and for ZIP files (figure 39). The intervals for 90% are a little shorter than the intervals for 95% for both PDF and ZIP[15]. The deviation is not too big, given the absolute values of the measurements.

6.5 Exercise 6.5

Given number of customers n and throughput $X_0(n)$.

Requested Show that response time grows linearly with the number of customers when the throughput saturates.

Solution The case under consideration is equal to considering a server including the waiting line. According to Little's Law $N = R \times X_0$ where N is the number of customers and R is the is response time. In our case this will be $n = R(n) \times X_0(n)$.
Because the throughput $X_0(n)$ is saturated, this means that it will hardly increase. The number of customers however will increase linearly. As the throughput remains the same, from the formula $n = R(n) \times X_0(n)$ it follows that $R(n)$ must also increase linearly[16].

6.6 Exercise 6.6

Given The data from chapter 6 on the unbalanced and the balanced disks.

[15] which makes sense as reliability is smaller.
[16] Well almost, because the throughput will keep increasing in ever smaller increments.

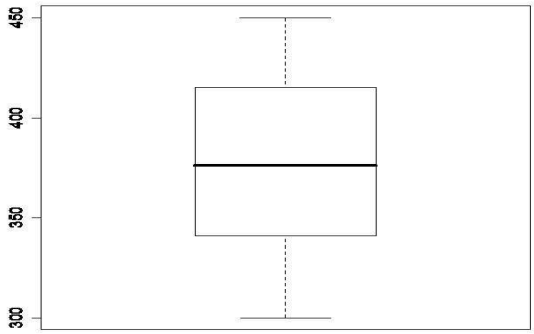


Figure 37: **Exercise 6.3** boxplot for the PDF files

PDF FILES		
Confidence coefficient	**90%**	**95%**
Mean Confidence level	3.50	4,18
Confidence interval		
interval start	374.07	373,39
interval end	381.08	381,75
interval length	7.01	8,36

Figure 38: **Exercise 6.4**: Confidence intervals 90% and 95% for PDF

Requested Explain why the balanced configuration is better for ZIP files but worse for PDF files when compared to the original configuration.

Solution The graph of figure 6.5 in the Menascé book[17] shows why this is the case. The (saturated) throughput for the ZIP files increases from 4.2 to 6.6 files per second, while the throughput for PDF files decreases dramatically from 12 to 5 files per second. So it follows, that the balanced configuration is better for the ZIP files than it is for the PDF files.

6.7 Exercise 6.7

Given The three security options of §6.6.

Requested Calculate how much faster the CPU must be for each security option, in order to withstand the load of 164 users, using the spreadsheet **ClosedQN-Secure.xls**.

Remark The book uses the unusual number of 164, where the previous exercises and examples used multiples of 20. To keep the ratio of $1 : 4$ this would imply 32.8 and 131.2 as number of customers. The spreadsheet does not accept a

[17] *and* Almeida and Dowdy

ZIP FILES		
Confidence coefficient	**90%**	**95%**
Mean Confidence level	5,81	6,94
Confidence interval		
interval start	1149,79	1148,67
interval end	1161,42	1162,54
interval length	11,63	13,88

Figure 39: **Exercise 6.4**: Confidence intervals 90% and 95% for ZIP

partial number of customers. For that reason, in the calculations the number of multiprogramming was set to 160 in stead of 164.

Solution using excel sheet ClosedQN-secure-ch06.xls For each of the security options the disk access service demands remain unaltered, while the CPU speed increases. The CPU service demand will be devided by the increase in speed. Suppose the CPU is replaced by a CPU that has twice the speed, then the service demands will be half the original service demands. If the new CPU is factor f faster than the old CPU, then the service demands for the CPU will be $\frac{\text{old service demands}}{f}$.
The new service demands are listed in figure 40. Using these service demands the new response times and throughput values can be solved using the spreadsheet **ClosedQN-Secure.xls** (figure 41).
In conclusion, the speed factors do not contribute enough to downloading the files, in order to get the response times for ZIP below 20 seconds and for PDF below 7 seconds.

Solution using jmt Using the same input data, but input into JMT, provides a different set of results. These results are to be added later. See figure 42 for the low security option results as calculated by JMT-JMVA. From the results of JMT one can infer that a speed increase of just over 2 is enough to satisfy the requirements in the low security option case.
This contradicts the results from using the spreadsheet and need further investigation[18].

6.8 Exercise 6.8

Given §6.7 Experimental comparison of two servers

Requested At what confidence level are the servers considered to be not significantly different?

Solution At the confidence level of 95% the interval has a certain length, that does not include zero. For the interval to become larger and maybe include zero, the confidence level must rise. Using the spreadsheet **servercomparison.xls**,

[18]Until proven otherwise both algorithms are assumed to be correct. I assume an input error in either the spreadsheet or JMT.

CPU Service Demands						
	Low Security		**Med Security**		**High Security**	
speed factor	PDF	ZIP	PDF	ZIP	PDF	ZIP
1	0.0889	0.5120	0.1644	0.4541	0.3175	0.8727
2	0.0445	0.1256	0.0822	0.2271	0.1588	0.4364
3	0.0296	0.0837	0.0548	0.1514	0.1058	0.2909
4	0.0222	0.0628	0.0411	0.1135	0.0794	0.2182
5	0.0178	0.0502	0.0329	0.0908	0.0635	0.1745
6	0.0148	0.0419	0.0274	0.0757	0.0529	0.1455
7	0.0127	0.0359	0.0235	0.0649	0.0454	0.1247
8	0.0111	0.0314	0.0205	0.0568	0.0397	0.1091
9	0.0099	0.0279	0.0183	0.0505	0.0353	0.0970
10	0.0089	0.0251	0.0164	0.0454	0.0318	0.0873
11	0.0081	0.0228	0.0149	0.0413	0.0289	0.0793

Figure 40: **Exercise 6.7**: CPU service demands per speed factor

one can calculate the results for confidence levels from 95% to 99.9999999% as far as the program's accuracy permits (figure 43)[19].
The result implies, that there is no confidence level where zero is including in the interval. This indicates, that the difference between the two servers is too big to have any doubt about the performance of the new server for processing PDF and ZIP files.

[19] Any confidence levels smaller than 95% will also have a smaller interval as you can verify by calculating several lower confidence levels.

Low Security				
Speed factor	X_{PDF}	X_{ZIP}	R_{PDF}	R_{ZIP}
1	2.25	3.18	16.0	45.2
2	4.56	6.33	7.9	22.7
3	5.09	6.67	7.1	21.6
4	5.10	6.67	7.1	21.6
5	5.10	6.67	7.1	21.6
6	5.10	6.67	7.1	21.6
Medium Security				
Speed factor	X_{PDF}	X_{ZIP}	R_{PDF}	R_{ZIP}
1	1.22	1.76	29.6	81.8
2	2.44	3.52	14.8	40.9
3	3.67	5.28	9.8	27.3
4	5.03	6.67	7.2	21.6
5	5.09	6.67	7.1	21.6
6	5.10	6.67	7.1	21.6
High Security				
Speed factor	X_{PDF}	X_{ZIP}	R_{PDF}	R_{ZIP}
1	0.63	0.92	57.1	157.1
2	1.26	1.83	28.6	78.6
3	1.89	2.75	19.0	52.4
4	2.52	3.66	14.3	39.3
5	3.16	4.58	11.4	31.5
6	3.80	5.49	9.5	26.2
7	4.50	6.37	8.0	22.6
8	5.06	6.67	7.1	21.6
9	5.09	6.67	7.1	21.6

Figure 41: **Exercise 6.7**: Results of solving per speed factor for each security option

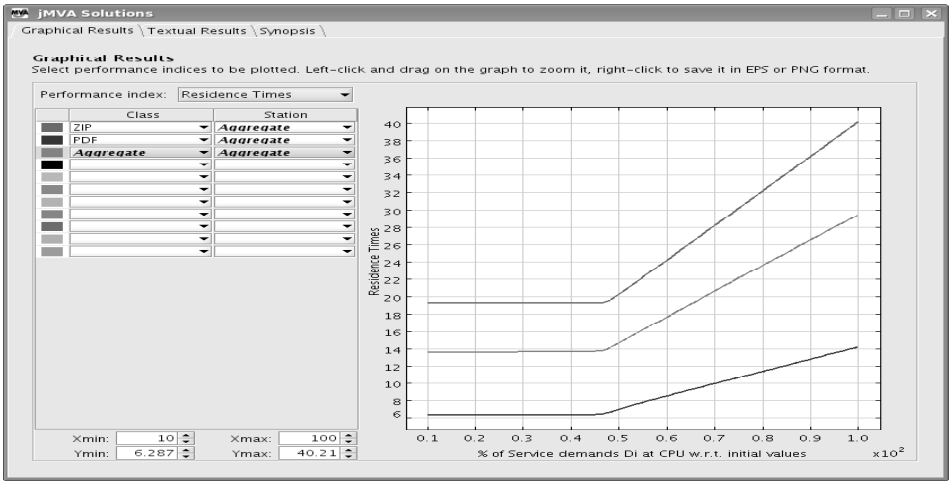


Figure 42: **Exercise 6.7**: JMT-Results of solving per speed factor for low security option

Confidence Level PDF	$\frac{1}{2}$Confidence interval	Mean lower bound	Mean upper bound
95%	0.00227	-0.0380	-0.0334
96%	0.00238	-0.0381	-0.0333
97%	0.00251	-0.0382	-0.0332
98%	0.00269	-0.0384	-0.0330
99%	0.00298	-0.0387	-0.0327
99.99%	0.00451	-0.0402	-0.0312
99.9999%	0.00567	-0.0414	-0.0300
99.999999%	0.00664	-0.0424	-0.0291
99.99999999%	0.00749	-0.0432	-0.0282
99.9999999999%	0.00826	-0.0440	-0.0275
99.999999999999%	0.00896	-0.0447	-0.0267
99.9999999999999%	0.00930	-0.0450	-0.0264

Confidence Level ZIP	$\frac{1}{2}$Confidence interval	Mean lower bound	Mean upper bound
95%	0.00227	-0.1160	-0.1058
96%	0.00238	-0.1163	-0.1056
97%	0.00251	-0.1166	-0.1053
98%	0.00269	-0.1170	-0.1049
99%	0.00298	-0.1176	-0.1042
99.99%	0.00451	-0.1210	-0.1008
99.9999%	0.00567	-0.1236	-0.0982
99.999999%	0.00664	-0.1258	-0.0960
99.99999999%	0.00749	-0.1277	-0.0941
99.9999999999%	0.00826	-0.1295	-0.0924
99.999999999999%	0.00896	-0.1311	-0.0908
99.9999999999999%	0.00930	-0.1318	-0.0901

Figure 43: **Exercise 6.8**: Confidence intervals around the mean do not include zero.

7 Chapter 7

7.1 Exercise 7.1

Given The Generalized Birth Death theorem:

$$p_k = p_0 \prod_{i=0}^{k-1} \frac{\lambda_i}{\mu_{i+1}} \tag{7.1.1}$$

Requested Prove that the solution 7.3.4 and 7.3.5 are the solutions to the Markov chain of section 7.3.

Solution The proof for equation 7.3.4 falls into two categories, one for $k \leq N$ and one for $k > N$.

k ≤ N

$$\begin{aligned}
p_k &= p_0 \left(\frac{\lambda_0}{\mu_1} \times \frac{\lambda_1}{\mu_2} \times \ldots \times \frac{\lambda_{k-1}}{\mu_k} \right) \\
&\text{using } \lambda_i = (M-i)\lambda \Rightarrow \\
p_k &= p_0 \left(\frac{M\lambda}{\mu} \times \frac{(M-1)\lambda}{2\mu} \times \ldots \times \frac{(M-k+1)\lambda}{k\mu} \right) \Leftrightarrow \\
&= p_0 \left(\frac{\lambda^k}{\mu^k} \right) \left(\frac{M(M-1)\ldots(M-k+1)}{k!} \right) \\
&= p_0 \left(\frac{\lambda}{\mu} \right)^k \left(\frac{M!}{k!(M-k)!} \right) \\
&= p_0 \left(\frac{\lambda}{\mu} \right)^k \binom{M}{k}
\end{aligned}$$

N < k ≤ M

$$\begin{aligned}
p_k &= p_0 \left(\frac{\lambda_0}{\mu_1} \times \frac{\lambda_1}{\mu_2} \times \ldots \times \frac{\lambda_{k-1}}{\mu_k} \right) \\
&\left. \begin{array}{ll} \text{using } \lambda_i & = (M-i)\lambda \\ \text{and } \mu_i & = \left\{ \begin{array}{ll} k\mu & \text{if } k \leq N \\ N\mu & \text{otherwise} \end{array} \right. \end{array} \right\} \Rightarrow \\
p_k &= p_0 \left(\frac{M\lambda}{\mu} \times \frac{(M-1)\lambda}{2\mu} \times \cdots \right. \\
&\left. \cdots \times \frac{(M-N+1)\lambda}{N\mu} \times \cdots \times \frac{(M-k+1)\lambda}{N\mu} \right) \Leftrightarrow \\
&= p_0 \left(\frac{\lambda}{\mu} \right)^k \left(M(M-1)\ldots(M-k+1) \frac{1}{N!} \frac{1}{N^{k-N}} \right) \\
&= p_0 \left(\frac{\lambda}{\mu} \right)^k \left(\frac{M(M-1)\ldots(M-k+1)k!}{k!} \frac{1}{N!N^{k-N}} \right) \\
&= p_0 \left(\frac{\lambda}{\mu} \right)^k \left(\frac{M!}{k!(M-k)!} \frac{k!}{N!N^{k-N}} \right) \\
&= p_0 \left(\frac{\lambda}{\mu} \right)^k \binom{M}{k} \frac{k!N^{N-k}}{N!}
\end{aligned}$$

Remark that in formula 7.3.4 of the book for $k > N$ the factor N^{N-k} yields a negative power parameter i.e. $\frac{1}{N^{k-N}}$.
The proof for 7.3.5 is simply a consequence of the requirement that the sum of probabilities is one.

$$\sum_{k=1}^{M} p_k = 1 \Leftrightarrow$$
$$p_0 \left(\frac{\lambda}{\mu}\right)^k \binom{M}{k} + p_0 \left(\frac{\lambda}{\mu}\right)^k \binom{M}{k} \frac{k!N^{N-k}}{N!} = 1 \Leftrightarrow$$
$$p_0 = \left[\left(\frac{\lambda}{\mu}\right)^k \binom{M}{k} + \left(\frac{\lambda}{\mu}\right)^k \binom{M}{k} \frac{k!N^{N-k}}{N!}\right]^{-1}$$

7.2 Exercise 7.2

Given A loadbalancer distributing load among the operational fraction j of M machines. The arrival rate of requests is equal to γ, per operational machine $\frac{\gamma}{j}$. Assume the average response time per operational machine is equal to $R_m = \frac{S}{1-\frac{\gamma}{j}S}$.

Requested Give an expression for the average response time R_D of a request at the system.

Solution Figure 44 shows the datacenter receiving input from its customers. Using R_d as response time for the datacenter and S_d as average service time for each request, where the arrival rate for the system is equal to γ and the Service time is equal to R_m, we define the response time as

$$R_d = \frac{S_d}{1 - \lambda_d S_d} = \frac{R_m}{1 - \gamma R_m} \tag{7.2.1}$$

Filling in the expression for R_m as defined,

$$R_d = \frac{\frac{S}{1-\frac{\gamma}{j}S}}{1 - \gamma \frac{S}{1-\frac{\gamma}{j}S}} \tag{7.2.2}$$
$$= \frac{S}{1 - \frac{\gamma}{j}S - \gamma S} \tag{7.2.3}$$
$$= \frac{S}{1 - (j-1)\frac{\gamma}{j}S} \tag{7.2.4}$$

Equation 7.2.4 shows the expression requested.

7.3 Exercise 7.3

Given The spreadsheet **Chap7-MarkovModel.xls**

Requested Draw the graphs of the probability P_j that exactly $j = 0, \ldots, M$ machines are operational as a function of $\frac{\lambda}{\mu}$ using $M = 120$. Draw the graphs for $N = 2$, $N = 5$ and $N = 10^{20}$.

[20] Slightly different than in the book where the formulation seems not exactly right.

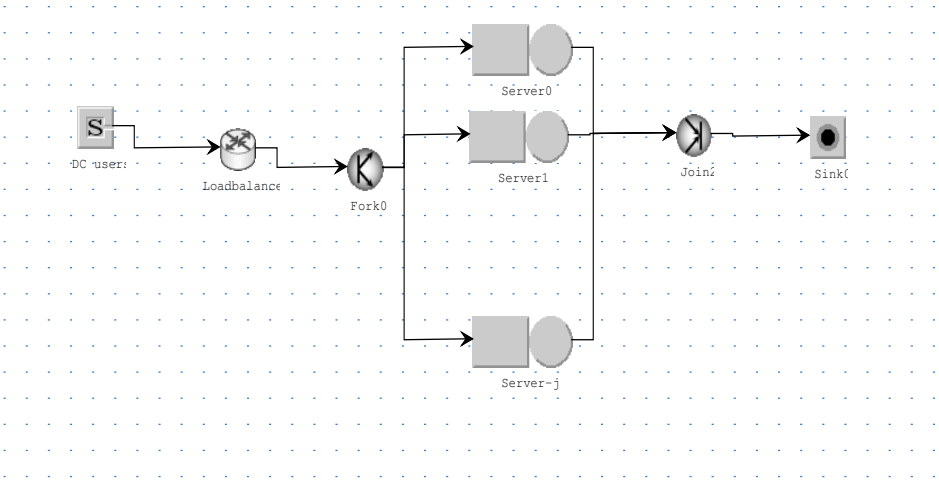


Figure 44: **Exercise 7.2** Datacenter receiving requests at arrival rate γ

Solution Using the spreadsheet the graphs were generated for $N = 2$, $N = 5$ and $N = 10$. The ratio λ/μ from $0.01, 0.02, \ldots, 0.10$ was used to create the graphs.
First verification is that the same ratio with differing repair time and MTTF result in the same probability values, which turns out to be true (no figure included). This is easy to verify with 500 MTTF and 20 minutes repair time, versus 1000 MTTF and 40 minutes repair time. Even though the models solve differently, the probabilities for the systems are identical.
After this verification, the repair time was left at 20 minutes, changing the MTTF to accomodate the ratio.
See figures 45, 46 and 47.

Comment The ratio λ/μ represents the length of time it takes for a machine to fail against the time it takes the repair crew to repair the failed machines. From all graphs, it is apparant, that only a relatively long MTTF results in a high number of machines operational with relatively high reliability. The ratio of $\frac{1}{10}$ results in very high availability with a high probability (0.3, 0.36 and 0.36 respectively) for all values of N used.
The differences between the graphs are much more apparant when the ratio is smaller than $\frac{1}{10}$. As N increases, the number of machines that remain operable increases dramatically. For $N = 10$ the optimal number of operable machines remains above 100 for all calculated ratio's. For $N = 2$ the optimal numbers are more spread out.

Query: importance of distribution? The basic figures in the book are MTTF and MTTR. One could wonder whether the *distribution* of MTTF and MTTR is of any influence in the results of the model.
For the machines it appears obvious, that the failures will not be exactly every MTTF minutes. Also, repairing machines will vary due to uncontrollable factors, like feeling OK, or having a bad day, accidents during the repair, etc.
Food for discussion.

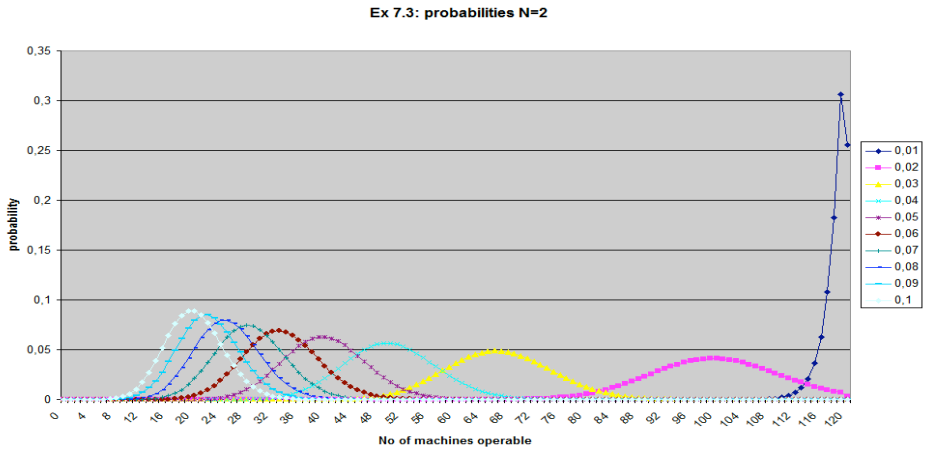


Figure 45: **Exercise 7.3** Graph of probabilities that exactly j machines are operation for N=2 and ratios of λ/μ

7.4 Exercise 7.4

Given The spreadsheet **Chap7-MarkovModel.xls**

Requested A table of MTTR for various values of λ/μ with $M = 120$ and $N = 5$.

Solution See figure 48 for the table.

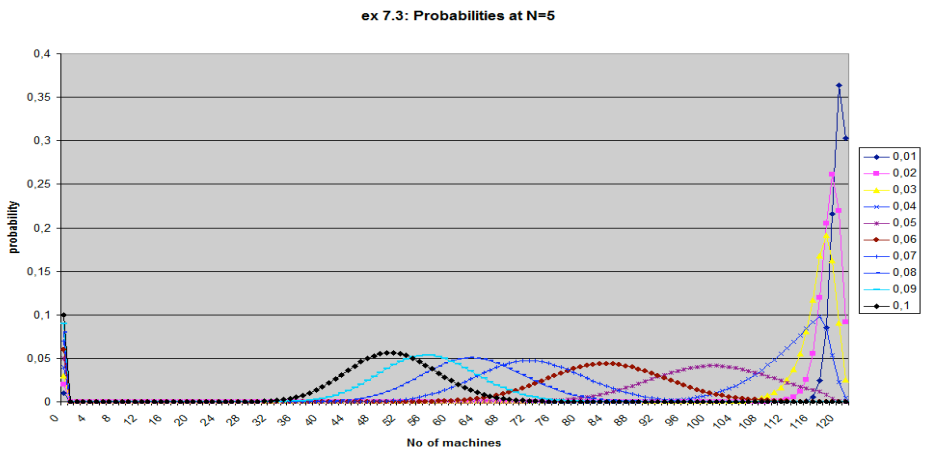


Figure 46: **Exercise 7.3** Graph of probabilities that exactly j machines are operation for N=5 and ratios of λ/μ

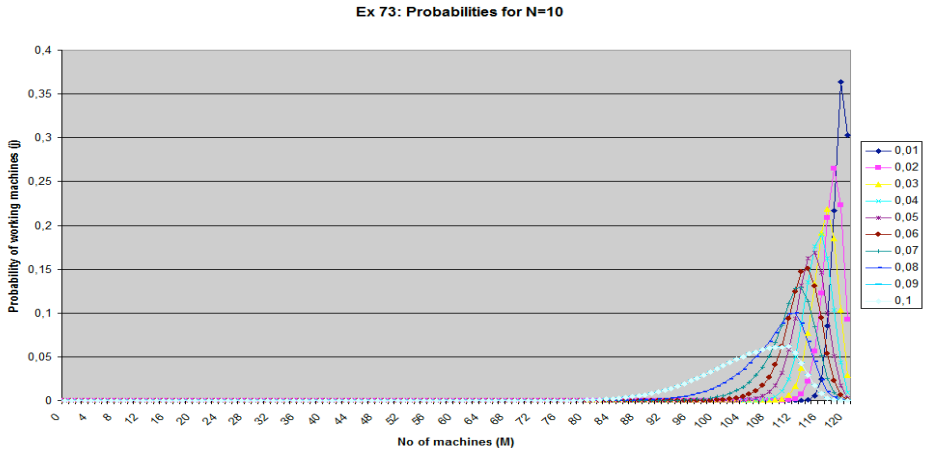


Figure 47: **Exercise 7.3** Graph of probabilities that exactly j machines are operation for N=10 and ratios of λ/μ

$N = 5$										
λ/μ	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.10
MTTR	20.04	20.72	24.18	38.14	84.40	146.70	194.29	230.00	257.78	280.00

Figure 48: **Exercise 7.4**: values of MTTR for λ/μ ratio's.

8 Chapter 8

8.1 Exercise 8.1

Given Assume the data from the book and spreadsheets **Chap8-CBMG.xls** and **Chap8-OpenQN.xls** plus the factors $f_A = 0.6$ and $f_B = 0.4$.

Requested Use **Chap8-CBMG.xls** and **Chap8-OpenQN.xls** to adjust and solve the performance model.

Solution Using the same formula's as in the spreadsheet **Chap8-CBMG.xls** and the new factors, results in new arrival rates per class (figure 49). Using these arrival rates in the model of the spreadsheet and solving it for each value of γ produces results that are plotted in figure 50. Due to utilization of 100% in device no 6 (DS-disk), no more than an arrival rate of 10.5 could be plotted. The calculations and the plot are in the spreadsheet `Chap8-OpenQN-ex-8.1.xls`.

Arrival of requests (requests/sec)						
	AR-8	AR-9	AR-10	AR-10.5	AR-10.8	AR-11
Home (h)	8.000	9.000	10.000	10.500	10.800	11.000
Search (s)	9.673	10.882	12.091	12.695	13.058	13.300
View bids (v)	1.731	1.947	2.164	2.272	2.337	2.380
Login (g)	3.020	3.397	3.775	3.963	4.077	4.152
Create Auction (c)	0.906	1.019	1.132	1.189	1.223	1.246
Place Bids (b)	1.763	1.984	2.204	2.314	2.380	2.424

Figure 49: **Exercise 8.1**: Arrival rates per class depending on the general arrival rate.

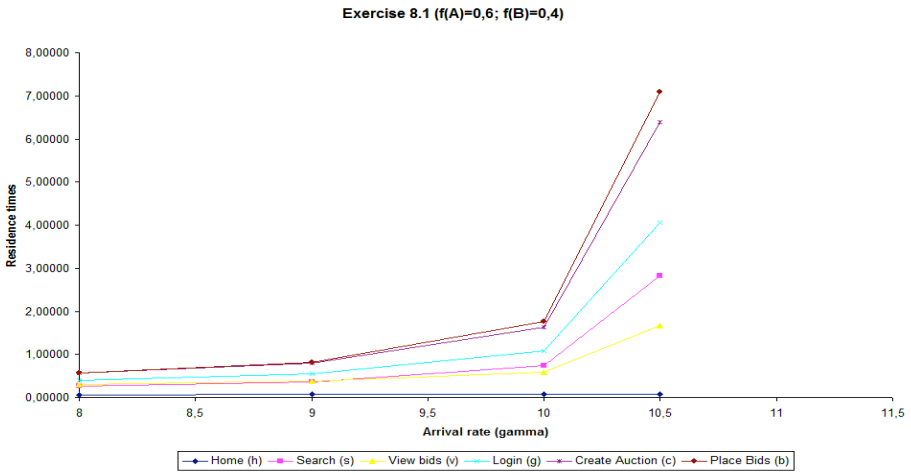


Figure 50: **Exercise 8.1**: The residence times per arrival rate between 8 and 11.

8.2 Exercise 8.2

Given The equations (8.4.9) - (8.4.14) and (8.5.16).

Requested

1. Provide an expression for the maximum value of γ as a function of visit ratio, service demands and fraction of sessions for each type of workload.
2. Calculate the maximum γ for the auction site.

Solution Abbreviate terms in $\lambda_r = \gamma(f_A \times V_h^A + f_B \times V_h^B)$ as $\lambda_r = \gamma F_h$ for equations (8.4.9) - (8.4.14).
The expression then follows from the requirement that $U_i \leq 1$ and eq. (8.5.16), where R is the number of classes:

$$\begin{aligned} U_i = \sum_{r=1}^{R} \lambda_r \times D_{i,r} \leq 1 \Leftrightarrow \\ \sum_{r=1}^{R} \gamma F_r D_{i,r} \leq 1 \Leftrightarrow \\ \gamma \sum_{r=1}^{R} F_r D_{i,r} \leq 1 \Leftrightarrow \\ \gamma \leq \left[\sum_{r=1}^{R} F_r D_{i,r} \right]^{-1} \Leftrightarrow \\ \gamma \leq \left[\sum_{r=1}^{R} (f_A \times V_r^A + f_B \times V_r^B) D_{i,r} \right]^{-1} \quad \square \end{aligned} \tag{8.2.1}$$

γ can not exceed the expression on the right side of equation 8.2.1 for *any* of the resources. Or in other words, γ_{max} is equal to the minimum of these expressions:

$$\gamma_{max} = \text{minimum} \left[\sum_{r=1}^{R} D_{i,r}(f_A \times V_r^A + f_B \times V_r^B) \right]^{-1} \quad \forall_i \tag{8.2.2}$$

Calculation for auction site The spreadsheet `Chap8-CBMG-ex-8.2.xls` contains the calculation for these expressions per resource. Figure 51 holds the results. As you can see, the database server disk has the maximum sum[21] and thus the minimum of equation 8.2.2.
So, the value for γ_{max} is determined by the DS-disk and the maximum value is $\gamma = \frac{1}{0.0890} \approx 11.23$.

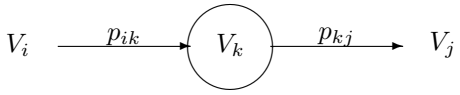
8.3 Exercise 8.3

Given A general CBMG (Customer Behaviour Model Graph). All transition probabilities p_{ij} from state i to j are known for all possible values of i and j.

Requested A system of linear equations whose solution provides the visit ratios for a general CBMG.

$1/\gamma$	Home (h)	Search (s)	View (v)	Login (g)	Create Auction (c)	Place Bids (b)	
WS-CPU	0.008	0.0112	0.0022	0.0201	0.0012	0.0028	0.0375
WS-disk	0.03	0.0125	0.0020	0.0033	0.0010	0.0019	0.0207
AS-CPU		0.0374	0.0071	0.0084	0.0045	0.0076	0.0649
AS-disk		0.0100	0.0161	0.0030	0.0011	0.0023	0.0324
DS-CPU		0.0125	0.0018	0.0050	0.0070	0.0085	0.0348
DS-disk		0.0436	0.0036	0.0167	0.0080	0.0170	**0.0890**

Figure 51: **Exercise 8.2**: The sum from the denominator of γ_{max} expression.



Solution Let V_k be an arbitrary node in the CBMG graph (figure 8.3). Then, each node V_i has as a known probability p_{ik} for the transition from node V_i to V_k and V_k has a known probability p_{kj} for the transition to any node V_j, where n is the number of states (including begin and exit states) and $i, j \in \{0, \ldots, n\}$ and $p_{ij} \in [0, 1]$).
Then the number of visits to any node V_k for $k \in \{0, \ldots, n\}$ can be expressed as:

$$V_k = \sum_{i=1}^{n} V_{i,k} \times p_{ik} \forall_{k \in \{0,\ldots,n\}} \tag{8.3.1}$$

This set of equations represents all transitions, including the ones with probability zero.

8.4 Exercise 8.4

Given Use table 8.3[22]. Assume that type A has 40% of the sessions and type B has 60% of the sessions. Assume 11 sessions per second are started.

Requested What is the average number of auctions created per hour?

Solution The number of auctions are determined by the number of visits to the node V_c CREATE AUCTION. The arrival rate for CREATE AUCTION per hour is equal to $\lambda_c \times 3600$. To calculate λ_c we need the fractions for type A and B plus the visit counts (table 8.3). This leads to the following calculation.

[21] in the analysis the DS-disk is also the bottleneck, so this makes sense
[22] PBD, §8.3 , page 210

$$\begin{aligned}\lambda_c &= \gamma(f_A \times V_c^A + f_B \times V_c^B) \\ &= 11(0.4 \times 0.128 + 0.6 \times 0.091) \\ &= 11((0.0512 + 0.00546) = 11 \times 0.1058 \\ &= 1.1638 \text{ auctions per second} \\ &= 1.1638 \times 3600 \approx 4190 \text{ auctions per hour}\end{aligned} \tag{8.4.1}$$

8.5 Exercise 8.5

Given

- 2% of the auction items are sold i.e. are a winner.
- Winner get 50 bids on average.
- The average price is $50.
- 2% commission is for the auction site.

Requested Find the maximum possible revenue throughput i.e. revenue per second generated by the auction site.

Solution The maximum throughput of revenue is generated when the number of visits to the node V_c is maximal, i.e. λ_c is maximal. λ_c is expressed as:

$$\gamma(f_A \times V_c^A + f_B V_c^B) = \gamma(0.25 \times 0.128 + 0.75 \times 0.091) = 0.10025\gamma$$

This expression reaches its maximum when γ reaches its maximum. In exercise 8.2 of the book[23] γ_{max} is calculated to be equal to approximately 11.23.
Using γ_{max} to calculate the maximum λ_c as described in the previous paragraph yields $\lambda_c = 0.10025\gamma = 0.10025 \times 11.23 = 1.1268$.
Finally, the revenue generated by the auction site is equal to 2% of the arrival rate of CREATE AUCTION (i.e. the winners) times 50 dollars minus a 2% commission, or the site generates revenues of on average $1.10 per second:

rev=$1.10

$$\begin{aligned}0.02 \times \lambda_c \times \$50 \times 0.98 \text{ commission} \\ 0.02 \times 1.1269 \times 50 \times 0.98 \approx 1.10 \text{ dollars/sec}\end{aligned}$$

8.6 Exercise 8.6

Given The same data as the previous exercise, with the two disk solution in stead of the one disk solution.

Requested The same request as the previous exercise, but for the two disk solution.

[23] and assuming that these results are correct ⌣.

Solution To find the revenue throughput we need a new value of γ_{max}, because the service demands for the two disk solution differ from the previous exercise. Once the value for γ_{max} is found, the rest of the calculations are done in the same was as in the previous exercise.
Again using the service demands and the V-table of the two disk solution, γ_{max} can be determined. This time the database server has two disks, causing the sum to be split in half among both disks. As a consequence, the CPU of the application server now has the largest sum and determines the value of γ_{max}. Calculating as in exercise 8.6 (see `Chap8-CBMG-ex-8.5.xls`) the value for γ_{max} is equal to approximately 15.41 resulting in a maximum revenue throughput of \$1.52 per second on average , in stead of the original \$1.10[24].

rev=\$1.52

8.7 Exercise 8.7

Given

- Assume the data in table 8.3[25].
- Assume the service demands in table 8.4[26].
- Assume a mix of type A = 45% and type B = 55%.
- Assume there is only one web server and one application server.

Requested What is the minimum number of database servers required to support a session start rate of 15 sessions/sec?

Preliminary From exercise 8.6 we already learned, that 2 disks yields a γ_{max} of more than 15 sessions per second. This already indicates how many servers will be necessary to support 15 session starts a second. In this case however, the load of type A transactions and type B transactions was altered.

Solution First of all, in order to get a calculation method in the spreadsheet that will work correctly, in spreadsheet `Chap8-OpenQN-ex-8.7-AS-probe.xls` the three webserver solution was built and verified using the results in the book. Subsequently the calculation for the number of application servers was built and executed. Most added calculations are on tab "Residence Times", service demands are on the tab "Utilizations". After executing the calculations for 1, 2, 3, 4 and more servers (see spreadsheet), the result is, that at minimum 2 application servers are necessary to serve 15 sessions/sec.
The answer to the question therefore is at minimum 2 application servers to be able to process 15 sessions starts per second.

2 DS servers

Additional information In the spreadsheet the queue length is specified using the formula $\frac{U_i}{1-U_i}$, which is derived from the response time formula:

[24] Performance analysis and modeling *really does* pay off!
[25] PBD, §8.3 , page 210
[26] PBD, §8.4 page 212.

$$
\begin{aligned}
R_{i,r} &= \frac{D_{i,r}}{1-U_i} \Leftrightarrow (\text{multiply by } X_i) \\
X_i \times R_{i,r} &= \frac{X_i D_{i,r}}{1-U_i} \Leftrightarrow (\text{Little's Law, Utilization Law}) \\
N_i &= \frac{U_i}{1-U_i}
\end{aligned} \tag{8.7.1}
$$

As a result the following table is generated for 1 webserver, 1 application server and 2 database servers.

$2 \times DS$				
	Utilization		Queue length	
	CPU	**disk**	**CPU**	**disk**
WS	70.9%	76.6%	2.4	3.3
AS	99.3%	50.3%	142.9	1.0
DS	27.3%	68.9%	0.4	2.2

Figure 52: **Exercise 8.7**: Utilization and quelength for each resource in the servers with 2 database servers.

As the table shows, as soon as the second database server is implemented, the application server will be the bottleneck, specifically the CPU. No matter how many DS servers are installed, the response time will not improve significantly. However, installing a second application server will solve the problem as the following table shows:

$2 \times DS$ and $2 \times AS$				
	Utilization		Queue length	
	CPU	**disk**	**CPU**	**disk**
WS	70.9%	76.6%	2.4	3.3
AS	49.7%	25.1%	1.0	0.3
DS	27.3%	68.9%	0.4	2.2

Figure 53: **Exercise 8.7**: Utilization and quelength for each resource in the servers when both database and application servers are increased to 2.

This table was generated using the spreadsheet with 2 database servers *and* 2 application servers. The table now shows, that the next bottleneck will be the web server, which already has queueing.

Verification of spreadsheet calculations The calculations in a spreadsheet can be confusing and difficult to check visually. therefore and for completeness, this sections checks the results wiht manual calculations.
Using table 8.4[27] the utilization for CPU and disk of the application are calculated. The λ values and the service demands are presented in figure 54. Using these figures and the formula $\frac{\lambda_r}{2} \times D_{i,r}$ we calculate the total utilizations of DS-CPU and DS-disk for 2 DS servers (figure 55). The last column shows the

[27] PBD, §8.4 page 212.

total utilization for CPU and disk. This value is used for calculating the residence times using equation (8.6.18) from the book (figure 56).
These values and the tab "Utilizations" in `Chap8-OpenQN-ex-8.7-AS-probe.xls` show the same results for service demands[28].
The response times for the webserver and application server have not changed. Thus the total response time calculated in the sheet is correct.
The conclusion is, that a minimum of 2 database servers is necessary to support 15 session starts per seconds.

8.8 Exercise 8.8

Given The CBMG in figure 8.6 in the book[29]. One server with a single CPU and disk. The service demands in table 8.5 in the book[30]. A start rate of 10 sessions per second.

Requested

- the average number of visits per second for each business function
- arrival rate of requests for each business function
- total utilization of CPU and disk
- residence times at CPU and disk for each business function
- response time for each business function

Solution To solve this CBMG the following steps are to be taken:

Visits per function Using the probabilities and set of equations represented in 8.8.1 determine the number of visits per business function (see exercise 8.3).

$$V_k = \sum_{i=1}^{n} V_{i,k} \times p_{ik} \quad \forall_{k \in \{0,\ldots,n\}} \tag{8.8.1}$$

Also check the probabilities: all outgoing vertices should add up to 1. Remember to include the begin and exit node.

request arrival rates Using the visits calculate the arrival rate of requests per function:

$$\lambda_r = \gamma \times V_r$$

Utilization Use the arrival rates, the service demands and equation (8.5.15) from the book[31] to calculate the utilization per resource.

$$U_i = \sum_{r=1}^{R} \frac{\lambda_r}{s} \times D_{i,r}$$

where s is the number of servers for resource i.

[28] apart from accuracy: the spreadsheet is more accurate
[29] PBD, page 220.
[30] PBD, page 219.
[31] PBD, page 212

$$
\begin{aligned}
\lambda_r &= \gamma(f_A \times V_r^A + f_B \times V_r^B) \\
\lambda_h &= 15(0.45 \times 1 + 0.55 \times 1) = 15 \\
\lambda_s &= 15(0.45 \times 1.167 + 0.55 \times 1.273) = 18.3795 \\
\lambda_v &= 15(0.45 \times 0.233 + 0.55 \times 0.191) = 3.1485 \\
\lambda_g &= 15(0.45 \times 0.427 + 0.55 \times 0.304) = 5.3903 \\
\lambda_c &= 15(0.45 \times 0.128 + 0.55 \times 0.091) = 1.61475 \\
\lambda_b &= 15(0.45 \times 0.256 + 0.55 \times 0.167) = 3.10575
\end{aligned}
\tag{8.7.2}
$$

Values of λ						
	h	s	v	g	c	b
λ_r	15.000	18.3795	3.1485	5.3903	1.61475	3.10575
Service Demands						
$D_{\text{DS-CPU}}$	0.000	0.010	0.009	0.015	0.070	0.045
$D_{\text{DS-disk}}$	0.000	0.035	0.018	0.050	0.080	0.090

Figure 54: **Exercise 8.7**: The values of λ_r from $\lambda = 15$.

$$
\begin{aligned}
U_i &= \sum_{r=1}^{R} \lambda_r \times D_{i,r} \\
U_{CPU} &= \frac{15.000}{2} \times 0.000 + \frac{18.3795}{2} \times 0.010 + \\
&\quad \frac{3.1485}{2} \times 0.009 + \frac{5.3903}{2} \times 0.015 + \\
&\quad \frac{1.61475}{2} \times 0.070 + \frac{3.10575}{2} \times 0.045 \\
&\approx 0.2729 \\
U_{disk} &= \frac{15.000}{2} \times 0.000 + \frac{18.3795}{2} \times 0.035 + \\
&\quad \frac{3.1485}{2} \times 0.018 + \frac{5.3903}{2} \times 0.050 + \\
&\quad \frac{1.61475}{2} \times 0.080 + \frac{3.10575}{2} \times 0.090 \\
&\approx 0.6891
\end{aligned}
\tag{8.7.3}
$$

Utilizations							
	h	s	v	g	c	b	total
DS-CPU	0.000	0.0919	0.0142	0.0404	0.0565	0.0699	0.2729
DS-disk	0.000	0.3216	0.0283	0.1348	0.0646	0.1398	0.6891

Figure 55: **Exercise 8.7**: Calculating the utilizations for 2 DS-servers.

$$R_r = \sum_{i=1}^{K} \frac{D_{i,r}}{1 - U_i}$$

$$U_{CPU} = 0.2729 \Rightarrow \frac{1}{1 - U_{CPU}} \approx 1.3753$$

$$U_{disk} = 0.6891 \Rightarrow \frac{1}{1 - U_{disk}} \approx 3.2165$$

$$\Rightarrow R_{CPU,r} \approx 1.3753 \times D_{CPU,r} R_{disk,r} = 3.2165 \times D_{disk,r}$$

$$\begin{aligned}
R_{CPU,h} &= 1.3753 \times 0.000 = 0.0000 & R_{disk,h} &= 3.2165 \times 0.000 = 0.0000 \\
R_{CPU,s} &= 1.3753 \times 0.010 \approx 0.0138 & R_{disk,s} &= 3.2165 \times 0.035 \approx 0.1126 \\
R_{CPU,v} &= 1.3753 \times 0.009 \approx 0.0124 & R_{disk,v} &= 3.2165 \times 0.018 \approx 0.0579 \\
R_{CPU,g} &= 1.3753 \times 0.015 \approx 0.0206 & R_{disk,g} &= 3.2165 \times 0.050 \approx 0.1608 \\
R_{CPU,c} &= 1.3753 \times 0.070 \approx 0.0963 & R_{disk,c} &= 3.2165 \times 0.080 \approx 0.2573 \\
R_{CPU,b} &= 1.3753 \times 0.045 \approx 0.0619 & R_{disk,b} &= 3.2165 \times 0.090 \approx 0.2895
\end{aligned} \tag{8.7.4}$$

Residence times web server						
	h	s	v	g	c	b
WS-CPU	0.0275	0.0309	0.0378	0.2063	0.0413	0.0516
WS-disk	0.1284	0.0428	0.0428	0.0428	0.0428	0.0428
Residence times application server						
AS-CPU	0.0000	4.3181	5.0378	3.5984	6.4772	5.7575
AS-disk	0.0000	0.0161	0.1608	0.0181	0.0221	0.0241
Residence times database server						
DS-CPU	0.000	0.0138	0.0124	0.0206	0.0963	0.0619
DS-disk	0.000	0.1126	0.0579	0.1608	0.2573	0.2895
Response times per transaction						
DS-disk	0.15588	4.5342	5.3494	4.0469	6.9367	6.2271

Figure 56: **Exercise 8.7**: Calculating the residence times for 2 DS-servers.

Residence times and response times Use the utilization and service demands to calculate the response time per business function as in equation (8.5.15)[32]:

$$R_r = \sum_{i=1}^{K} \frac{D_{i,r}}{1 - U_i}$$

These calculations can be done by hand or using a spreadsheet. In this exercise the calculations were done using a calculator.

Number of visits

check probabilities

$$\begin{aligned} V_h &: p_{hs} + p_{ha} + p_{hx} = 0.7 + 0.2 + 0.1 = 1,0 \surd \\ V_s &: p_{ss} + p_{sa} + p_{sx} = 0.6 + 0.3 + 0.1 = 1.0 \surd \\ V_a &: p_{ap} + p_{ax} = 0.4 + 0.6 = 1.0 \surd \\ V_p &: p_{px} = 1.0 \surd \end{aligned} \tag{8.8.2}$$

Visit equations

$$\begin{aligned} V_h &= V_e \times p_{eh} = 1 \times 1.0 = 1 \\ V_s &= V_h \times p_{hs} + V_s \times p_{ss} \Leftrightarrow \\ &= \frac{V_h \times p_{hs}}{1 - p_{ss}} = \frac{1 \times 0.7}{1 - 0.6} \\ &= \frac{0.7}{0.4} = \frac{7}{4} = 1.75 \\ V_a &= V_s \times p_{sa} + V_h \times p_{ha} = 0.2 \times 1 + 1.75 \times 0.3 \\ &= 0.2 + 0.525 = 0.725 \\ V_p &= V_a \times p_{ap} = 0.725 \times 0.4 = 0.29 \end{aligned} \tag{8.8.3}$$

arrival rates per second $\lambda_r = \gamma V_r$:

$$\begin{aligned} \lambda_r &= 10 \times 1 = 10 \\ \lambda_h &= 10 \times 1 = 10 \\ \lambda_s &= 10 \times 1.75 = 17.5 \\ \lambda_a &= 10 \times 0.725 = 7.25 \\ \lambda_p &= 10 \times 0.29 = 2.9 \end{aligned} \tag{8.8.4}$$

[32] PBD, page 212

Utilization per resource

$$U_i = \sum_{r=1}^{R} \frac{\lambda_r}{2} \times D_{i,r}$$

filled in for CPU and disk:

$$\begin{aligned}
U_{CPU} &= \lambda_h \times D_{CPU,h} + \lambda_s \times D_{CPU,s} + \lambda_a \times D_{CPU,a} + \lambda_p \times D_{CPU,p} \\
&= 10 \times 0.010 + 17.5 \times 0.015 + 7.25 \times 0.010 + 2.9 \times 0.020 \\
&= 0.1 + 0.2625 + 0.0725 + 0.058 = 0.493 \\
U_{disk} &= \lambda_h \times D_{disk,h} + \lambda_s \times D_{disk,s} + \lambda_a \times D_{disk,a} + \lambda_p \times D_{disk,p} \\
&= 10 \times 0.015 + 17.5 \times 0.025 + 7.25 \times 0.015 + 2.9 \times 0.010 \\
&= 0.150 + 0.4375 + 0.10875 = 0.72525
\end{aligned} \tag{8.8.5}$$

Response times

$$\begin{aligned}
R_h &= \frac{D_{CPU,h}}{1 - U_{CPU}} + \frac{D_{disk,h}}{1 - U_{disk}} \\
&= \frac{0.010}{1 - 0.493} + \frac{0.015}{1 - 0.72525} \approx 0.0743 \text{ seconds.} \\
R_s &= \frac{D_{CPU,s}}{1 - U_{CPU}} + \frac{D_{disk,s}}{1 - U_{disk}} \\
&= \frac{0.015}{1 - 0.493} + \frac{0.025}{1 - 0.72525} \approx 0.01206 \text{ seconds.} \\
R_a &= \frac{D_{CPU,a}}{1 - U_{CPU}} + \frac{D_{disk,a}}{1 - U_{disk}} \\
&= \frac{0.010}{1 - 0.493} + \frac{0.015}{1 - 0.72525} \approx 0.0743 \text{ seconds.} \\
R_p &= \frac{D_{CPU,p}}{1 - U_{CPU}} + \frac{D_{disk,p}}{1 - U_{disk}} \\
&= \frac{0.020}{1 - 0.493} + \frac{0.010}{1 - 0.72525} \approx 0.0758 \text{ seconds.}
\end{aligned} \tag{8.8.6}$$

9 Chapter 9

9.1 Exercise 9.1

Given Consider the flow of fig 9.10 of the book[33]. Each statement $s \in a, b, c, d, e$ has a known service demand D_i^s at resource i.

Requested Give an expression for the service demand D_i at resource i.

Solution From table 9.10 in the book, the number of executions per statement are derived and the service demand from this statement on resource i (figure 57). This leads to the following expression for the service demands D_i at resource i:

$$\begin{aligned} D_i &= npD_i^a + npmD_i^b + npmD_i^c + n(1-p)D_i^d + nD_i^e \\ &= n(p(D_i^a + m(D_i^b + D_i^c)) + (1-p)D_i^d + D_i^e) \end{aligned} \tag{9.1.1}$$

stmt	n_s	p_s	m	nr of execs	D_i^s
a	n	p		np	npD_i^a
b	n	p	m	npm	$npmD_i^b$
c	n	p	m	npm	$npmD_i^c$
d	n	$1-p$		$n(1-p)$	$n(1-p)D_i^d$
e	n			n	nD_i^e

Figure 57: **Exercise 9.1**: The number of executions per statement

9.2 Exercise 9.2

Given

1. application to check balance or the transfer funds.
2. customer dialog with system (see book)
3. Assumptions
   (a) 1 CPU and 2 identical diskdrives
   (b) Average service time on disk is 8 msec per IO
   (c) CPU time per IO is 0.5 msec
   (d) files:
      i. CUSTOMER (pk = ID)
      ii. CHK_ACCOUNTS (pk = ID)
      iii. SAV_ACCOUNTS (pk = ID)
      iv. HISTORY (pk = transaction date)
   (e) Each request for CUSTOMER, CHK_ACCOUNTS or SAV_ACCOUNTS requires 2 physical IO's.
   (f) Customer must enter ID and pincode in order to do a transaction

[33] PBD page 246

(g) Three types of session:

Type A Request balance (60%)

Type B Request balance and transfer funds (30%)

Type C Transfer funds (10%)

Requested The following answers are requested:

- Estimate the service demands
- Plot the response time as a function of the arrival rate
- For which value of arrival rate does the response time exceed 1 second?
- Make and justify additional assumptions (file distributions on 2 disks)

Additional assumptions The description for customer dialog and the assumptions leaves a few issues open:

file distr. Each transaction will either reference the saving account or the checking account, but not both. The customer and history files will always be referenced. As far as I can see there is no way to distribute the files over the disks, so that both disks are utilized equally. Therefore the following distribution of files among disks is proposed:

disk 1 Contains the CUSTOMERS file and the HISTORY file.

disk 2 Containt the CHK_ACCOUNTS and the SAV_ACCOUNTS files.

history access Assumed is only one IO for creating a history record.

history update Only the balance request and transfer request are considered to be transactions for which a history log is recorded. The login transaction is "entered before any transaction can take place" (2^{nd} assumption in the exercise).

pincode The pincode is probably processed locally or integrated into the CUSTOMERS file. Assumption is that no extra IO is required.

update IOs The specification mentions that retrieving the customer, check account or savings account takes two physical IOs. It does not mention the update. The assumption is, that the update will take the same amount of IOs[34].

update vs select In theory, an update should be possible without a select preceding it. In the actual application an update is probably preceded by a select, if only to check the balance whether the update can take place. However, the specification does not mention this, so the assumption is to ignore this for the exercise[35]

[34] In reality, there would have been a difference between retrieving and updating.

[35] In reality we would have to ask the systems analyst.

Solution If λ is the start rate of a session, then the types of session implies the following arrival rates:

$$\begin{aligned} \lambda_{login} &= \lambda \\ \lambda_{balance} &= (0.6 + 0.3)\lambda = 0.9\lambda \\ \lambda_{transfer} &= (0.1 + 0.3)\lambda = 0.4\lambda \end{aligned} \tag{9.2.1}$$

In figures 58, 59 and 60, the transaction logic for login, balance request and transfer request are coded, based on the assumptions and customer dialog described.

```
01 select from customer where customer
02 if #ProbValidCustomer
03 login-ok
```

Figure 58: **Exercise 9.2**: Transaction logic for the login action

```
01 if #ProbBalCheckAccRequested
02  select from chk_account where customer
03 else
04  select from sav_account where customer
05 update history num_rows = 1
```

Figure 59: **Exercise 9.2**: Transaction logic for balance request action

```
01 if #ProbTransferCheckAccRequested
02  update chk_account num_rows = 1
03 else
04  update sav_account num_rows = 1
05 update history num_rows = 1
```

Figure 60: **Exercise 9.2**: Transaction logic for transfer request action

Using these statements for login, balance requests and transfer requests the service demands per relevant statement are determined. In order to determine the service demand for CPU and disk the following equation should be used:

$$D_{i,r} = \sum_{s \in S_{i,r}} n_s \times p_s \times D^s_{i,r} \tag{9.2.2}$$

(see equation (9.4.7) in the book page 244.

Login						
stmt	n_s	p_s	IOs	$D^s_{CPU,r}$	$D^s_{disk_1,r}$	$D^s_{disk_2,r}$
01	1	1	2	0.001	0.016	
Balance						
stmt	n_s	p_s	IOs	$D^s_{CPU,r}$	$D^s_{disk_1,r}$	$D^s_{disk_2,r}$
02	0.9	p_{balchk}	2	0.001		0.016
04	0.9	p_{balsav}	2	0.001		0.016
05	0.9	1	1	0.0005	0.008	
Transfer funds						
stmt	n_s	p_s	IOs	$D^s_{CPU,r}$	$D^s_{disk_1,r}$	$D^s_{disk_2,r}$
02	0.4	p_{trfchk}	2	0.001		0.016
04	0.4	p_{trfsav}	2	0.001		0.016
05	0.4	1	1	0.0005	0.008	

Estimated service demands The service demand D_i can be calculated using equation 9.2.2 (figure 61).

login transaction

$$
\begin{aligned}
D_{CPU,log} &= 1 \times 1 \times 0.001 = 0.001 \\
D_{disk_1,log} &= 1 \times 1 \times 0.016 = 0.016 \\
D_{disk_2,log} &= 1 \times 1 \times 0.000 = 0.000
\end{aligned}
$$

balance transaction

$$
\begin{aligned}
D_{CPU,\text{bal}} &= (0.9p_{\text{balchk}} + 0.9p_{\text{balsav}}) \times 0.001 + 0.9 \times 0.0005 \\
&= 0.9(p_{\text{balchk}} + p_{\text{balsav}}) \times 0.001 + 0.9 \times 0.0005 \\
&(p_{\text{balchk}} + p_{\text{balsav}} = 1) \\
&= 0.9 \times 0.0015 = 0.00135 \\
D_{disk_1,\text{bal}} &= 0.9 \times 0.008 = 0.0072 \\
D_{disk_2,\text{bal}} &= (0.9p_{\text{balchk}} + 0.9p_{\text{balsav}}) \times 0.016 \\
&= 0.9(p_{\text{balchk}} + p_{\text{balsav}}) \times 0.016 \\
&= 0.9 \times 0.016 = 0.0144
\end{aligned}
$$

transfer funds transaction

$$
\begin{aligned}
D_{CPU,\text{trf}} &= (0.4p_{\text{trfchk}} + 0.4p_{\text{trfsav}}) \times 0.001 + 0.4 \times 0.0005 \\
&= 0.4(p_{\text{trfchk}} + p_{\text{trfsav}}) \times 0.001 + 0.9 \times 0.0005 \\
&(p_{\text{trfchk}} + p_{\text{trfsav}} = 1) \\
&= 0.4 \times 0.0015 = 0.0006 \\
D_{disk_1,\text{trf}} &= 0.4 \times 0.008 = 0.0032 \\
D_{disk_2,\text{trf}} &= (0.4p_{\text{trfchk}} + 0.9p_{\text{trfsav}}) \times 0.016 \\
&= 0.4(p_{\text{trfchk}} + p_{\text{trfsav}}) \times 0.016 \\
&= 0.4 \times 0.016 = 0.0064
\end{aligned}
\tag{9.2.3}
$$

Estimated response time The utilization for each device is necessary to compute the estimated response time. Utilization is equal to

Estimated service demands			
	$D_{r,log}$	$D_{r,bal}$	$D_{r,trf}$
CPU	0.001	0.00135	0.0006
$disk_1$	0.016	0.0072	0.0032
$disk_2$	0.000	0.0144	0.0064

Figure 61: **Exercise 9.2**: The estimated service demands

$$
\begin{aligned}
U_i &= \sum_{i=1}^{R} \lambda_r D_{i,r} \\
&. \\
U_{CPU} &= \lambda_{login} D_{CPU,login} + \lambda_{bal} D_{CPU,bal} + \lambda_{trf} D_{CPU,trf} \\
&= \lambda D_{CPU,login} + 0.9\lambda D_{CPU,bal} + 0.4\lambda D_{CPU,trf} \\
&= 0.001\lambda + 0.9 \times 0.00135\lambda + 0.4 \times 0.0006\lambda \\
&= 0.002455\lambda \\
U_{disk_1} &= \lambda_{login} D_{disk,login} + \lambda_{bal} D_{disk,bal} + \lambda_{trf} D_{disk,trf} \\
&= \lambda D_{disk,login} + 0.9\lambda D_{disk,bal} + 0.4\lambda D_{disk,trf} \\
&= (0.016 + 0.9 \times 0.0072 + 0.4 \times 0.0032)\lambda \\
&= 0.02376\lambda \\
U_{disk_2} &= \lambda_{login} D_{disk,login} + \lambda_{bal} D_{disk,bal} + \lambda_{trf} D_{disk,trf} \\
&= \lambda D_{disk,login} + 0.9\lambda D_{disk,bal} + 0.4\lambda D_{disk,trf} \\
&= (0.000 + 0.9 \times 0.0144 + 0.4 \times 0.0064)\lambda \\
&= 0.01552\lambda
\end{aligned}
\tag{9.2.4}
$$

Estimated service demands				Utilization
	$D_{r,log}$	$D_{r,bal}$	$D_{r,trf}$	U_i
CPU	0.001	0.00135	0.0006	0.002455λ
$disk_1$	0.016	0.0072	0.0032	0.02376λ
$disk_2$	0.000	0.0144	0.0064	0.01552λ

Figure 62: **Exercise 9.2**: The estimated service demands and utilization

$$
\begin{aligned}
R_r &= \sum_{i=1}^{K} \frac{D_{i,r}}{1 - U_i} \\
&. \\
R_{login} &= \frac{0.001}{1 - 0.002455\lambda} + \frac{0.016}{1 - 0.02376\lambda} + \frac{0.000}{1 - 0.01552\lambda} \\
R_{balance} &= \frac{0.00135}{1 - 0.002455\lambda} + \frac{0.0072}{1 - 0.02376\lambda} + \frac{0.0144}{1 - 0.01552\lambda} \\
R_{transfer} &= \frac{0.0006}{1 - 0.002455\lambda} + \frac{0.0032}{1 - 0.02376\lambda} + \frac{0.0064}{1 - 0.01552\lambda}
\end{aligned}
\tag{9.2.5}
$$

Using these values for utilization, we can also determine the maximum λ that this system can handle.

Using the results in 9.2.5 the following plots are presented for R_{login}, R_{balance} and R_{transfer} as a function of λ[36]. The following values were used:

```
> plot(lambda,R_log,"b");
> plot(lambda,R_bal,"b");
> plot(lambda,R_trf,"b");
> lambda
 [1]  1.0  2.0  3.0  4.0  5.0  6.0  7.0  8.0  9.0 10.0 11.0 12.0 13.0 14.0 15.0
[16] 16.0 17.0 18.0 19.0 20.0 21.0 22.0 23.0 24.0 25.0 26.0 27.0 28.0 29.0 30.0
[31] 31.0 32.0 33.0 34.0 35.0 36.0 37.0 38.0 39.0 40.0 40.1 40.2 40.3 40.4 40.5
[46] 40.6 40.7 40.8 40.9 41.0 41.1 41.2 41.3 41.4 41.5 41.6 41.7 41.8 41.9 42.0
> R_log
 [1] 0.01739187 0.01780319 0.01823543 0.01869026 0.01916949 0.01967515
 [7] 0.02020950 0.02077507 0.02137469 0.02201153 0.02268919 0.02341173
[13] 0.02418379 0.02501066 0.02589839 0.02685400 0.02788559 0.02900262
[19] 0.03021620 0.03153944 0.03298794 0.03458039 0.03633943 0.03829269
[25] 0.04047426 0.04292670 0.04570389 0.04887496 0.05253022 0.05678981
[31] 0.06181727 0.06784093 0.07518968 0.08435502 0.09610588 0.11171642
[37] 0.13346258 0.16584753 0.21920839 0.32368954 0.33991997 0.35787011
[43] 0.37782858 0.40015240 0.42528856 0.45380423 0.48643011 0.52412387
[49] 0.56816577 0.62030697 0.68300789 0.75983787 0.85617910 0.98054507
[55] 1.14724525 1.38232921 1.73873351 2.34303473 3.59177893 7.69342266
> R_bal
 [1] 0.02335557 0.02377717 0.02421583 0.02467266 0.02514890 0.02564589
 [7] 0.02616510 0.02670817 0.02727690 0.02787329 0.02849957 0.02915823
[13] 0.02985207 0.03058421 0.03135824 0.03217819 0.03304868 0.03397503
[19] 0.03496338 0.03602089 0.03715594 0.03837845 0.03970024 0.04113553
[25] 0.04270168 0.04442001 0.04631720 0.04842706 0.05079321 0.05347308
[31] 0.05654399 0.06011291 0.06433257 0.06942953 0.07575589 0.08389198
[37] 0.09486995 0.11072539 0.13612063 0.18463298 0.19209301 0.20032819
[43] 0.20946843 0.21967440 0.23114725 0.24414225 0.25898820 0.27611609
[49] 0.29610205 0.31973411 0.34811950 0.38286442 0.42639088 0.48252998
[55] 0.55772099 0.66368624 0.82424720 1.09636335 1.65848045 3.50440396
> R_trf
 [1] 0.01038025 0.01056763 0.01076259 0.01096563 0.01117729 0.01139817
 [7] 0.01162893 0.01187030 0.01212307 0.01238813 0.01266648 0.01295921
[13] 0.01326758 0.01359298 0.01393700 0.01430142 0.01468830 0.01510001
[19] 0.01553928 0.01600928 0.01651375 0.01705709 0.01764455 0.01828246
[25] 0.01897852 0.01974223 0.02058542 0.02152314 0.02257476 0.02376581
[31] 0.02513066 0.02671685 0.02859225 0.03085757 0.03366928 0.03728533
[37] 0.04216442 0.04921129 0.06049806 0.08205910 0.08537467 0.08903475
[43] 0.09309708 0.09763307 0.10273211 0.10850766 0.11510587 0.12271826
[49] 0.13160091 0.14210405 0.15471978 0.17016196 0.18950706 0.21445777
[55] 0.24787599 0.29497166 0.36633209 0.48727260 0.73710242 1.55751287
>
```

[36] For λ a sequence of 1 to almost λ_{max} was used.

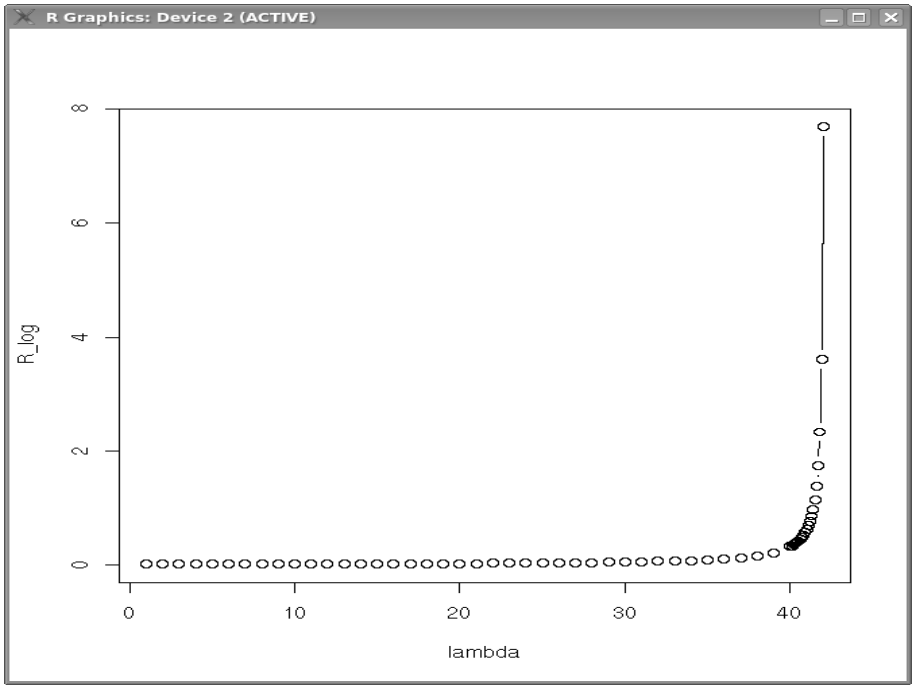


Figure 63: **Exercise 9.2**: R_{login} as a function of λ.

$$
\begin{aligned}
U_{CPU} &\leq 1 \Leftrightarrow \\
0.002455\lambda &\leq 1 \Leftrightarrow \\
\lambda &\leq \frac{1}{0.002455} \approx 407.3 \\
U_{disk_1} &\leq 1 \Leftrightarrow \\
0.02376\lambda &\leq 1 \Leftrightarrow \\
\lambda &\leq \frac{1}{0.02376} \approx 42.0 \\
U_{disk_2} &\leq 1 \Leftrightarrow \\
0.01552\lambda &\leq 1 \Leftrightarrow \\
\lambda &\leq \frac{1}{0.01552} \approx 64.4 \\
\lambda_{max} &\approx \text{minimum}(407.3, 42.0, 64.4) \\
\lambda_{max} &\approx 42.0
\end{aligned}
\tag{9.2.6}
$$

$\boxed{\lambda_{\max} \approx 42}$

1 second response time The calculated response times for balance requests appear to be the largest and are thus used for calculating the 1 second barrier.

$$
\begin{aligned}
1 &= R_{\text{balance}} \Leftrightarrow \\
1 &= \frac{0.00135}{1 - 0.002455\lambda} + \frac{0.0072}{1 - 0.02376\lambda} + \frac{0.0144}{1 - 0.01552\lambda} \\
\Rightarrow &\lambda \approx 41.77 \text{ (sessions per second)}
\end{aligned}
\tag{9.2.7}
$$

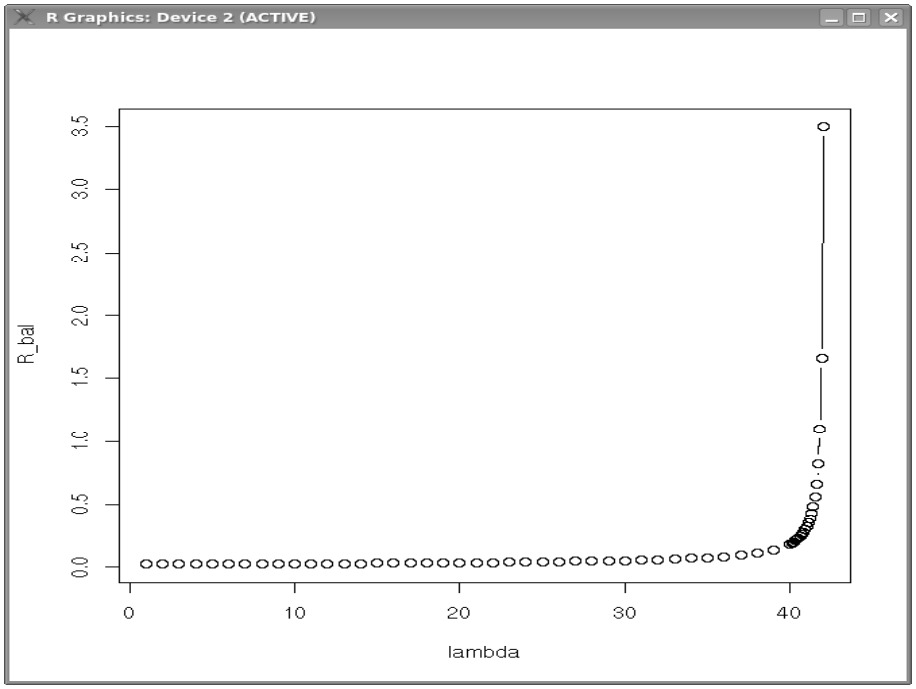


Figure 64: **Exercise 9.2**: R_{balance} as a function of λ.

9.3 Exercise 9.3

Given

1. Student-course registry system for a university.
2. 25000 students.
3. online for 24 hours per day.
4. each student enrols in 5 courses on average.
5. the same hardware and assumptions as with exercise 9.2.
6. Two alternatives are being considered:

   minimal consistency check Check only validity of courses. Registration request is filed online, but processed offline. No registration confirmation is issued.

   complete registration processing Complete checks involving the validity of the courses requested, whether the student has completed all prerequisites and whether the classes are all open. If any class is not open, the student is placed on a waiting list. A registration confirmation is returned to the student.

Requested Conduct an application sizing to compare the two alternatives. Determine the average response time for each of the two cases.

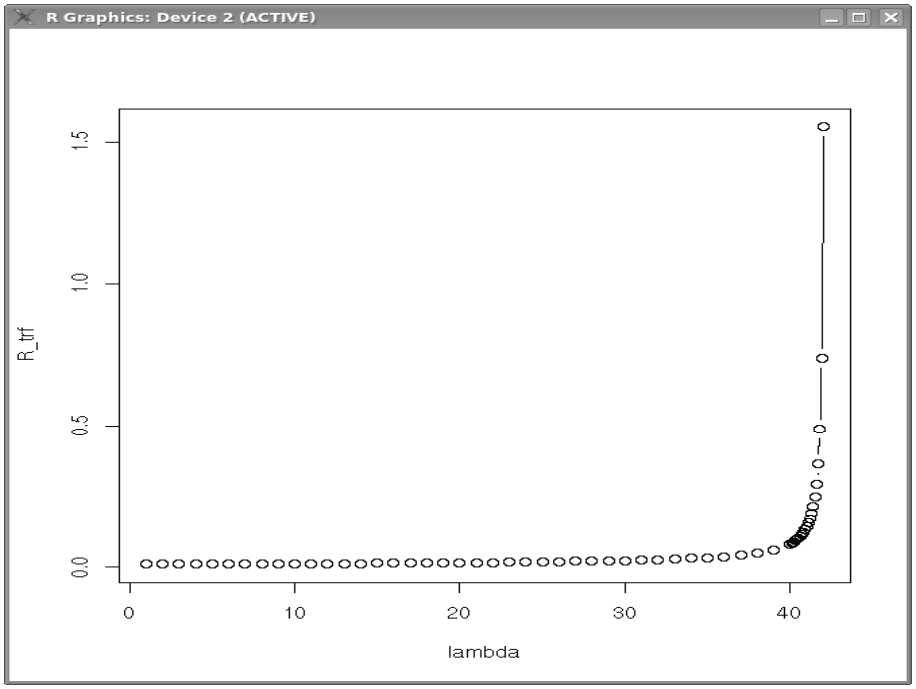


Figure 65: **Exercise 9.2**: R_{transfer} as a function of λ.

Solution

counter	quantity
#students	25000
time	24 hours for 2 weeks
#nrCoursesStudent	5
#ProbCourseClosed	10%
#PrereqCourse	2 on average per course

The arrival rate λ of requests, based on one request per student, is equal to $\frac{A_0}{T} = \frac{25000}{24*3600*14} = 0.020668$ requests per second.

$\boxed{\lambda \approx 0.020668}$

Case 1: Minimal consistency check The transaction logic for the students requests is contained in figure 66.

```
01 ! Enter student ID and code of courses
02 select from student where student
03 ! check courses
04 loop #nrCoursesStudent
05  select from course where course
06 update trans num_rows = 1
```

Figure 66: **Exercise 9.2**: Transaction logic for the student requests

From the transaction logic the files STUDENT, COURSE and TRANS can be assumed present. Following the assumptions in the previous exercise, each request to STUDENT and COURSE leads to 2 IOs, with 0.008 ms service time per IO for the disk and 0.0005 ms service time per IO for the CPU. Updating the TRANS file will only lead to one IO.

The number of records or tables in the files are 25000 for STUDENT, at least 125000 for COURSE (5 courses per student on average) and 25000 occurrences in TRANS (i.e. one request per student).

stmt	file/index	n_s	p_s	D^s_{CPU}	$D^s_{disk_1}$	$D^s_{disk_2}$
02	STUDENT	1	1	0.001	0.016	
05	COURSE	5	1	0.001		0.016
06	TRANS	1	1	0.0005	0.008	
				D_{CPU}	D_{disk_1}	D_{disk_2}
				0.0065	0.024	0.080

Using the formula (9.4.7) from the book[37] with r left out as there is but one transaction:

$$D_{CPU} = 1 \times 1 \times 0.001 + 5 \times 1 \times 0.001 + 1 \times 1 \times 0.0005 = 0.0065 \text{ seconds}$$
$$D_{disk_1} = 1 \times 1 \times 0.016 + 1 \times 1 \times 0.008 = 0.024 \text{ seconds}$$
$$D_{disk_2} = 5 \times 1 \times 0.016 = 0.080 \text{ seconds}$$

The utilization can be calculated using equation (8.5.16) on page 213, leaving out the r as there is but one transaction:

$$U_{CPU} = \lambda \times D_{CPU} \approx 0.020668 \times 0.0065 \approx 0.000134$$
$$U_{disk_1} = \lambda \times D_{disk_1} \approx 0.020668 \times 0.024 \approx 0.000496$$
$$U_{disk_2} = \lambda \times D_{disk_2} \approx 0.020668 \times 0.080 \approx 0.001653$$

The response time can be calculated using equation (8.5.15) on page 212 of the book:

$$R = \sum_{i=1}^{K} \frac{D_i}{1 - U_i} = \frac{0.0065}{1 - 0.000134} + \frac{0.024}{1 - 0.000496} + \frac{0.080}{0.001653}$$
$$R \approx 0.0065 + 0.024 + 0.080 = 0.11 \text{ seconds.}$$

Case 2: Complete registration processing The transaction logic for the student requests is contained in figure 67.
From the transaction logic the files STUDENT, COURSE, PREREQ, FOLLOW, WAITING AND TRANS can be assumed present. Following the assumptions in the previous exercise, each request to STUDENT. COURSE and PREREQ leads to 2 IOs, with 0.008 ms service time per IO for the disk and 0.0005 ms service time per IO for the CPU. Updating the TRANS, FOLLOW and WAITING file will only lead to one IO.
The number of records or tables in the files are 25000 for STUDENT, at least 125000 for COURSE (5 courses per student on average) and 25000 occurrences in TRANS (i.e. one request per student). The number of entries for FOLLOW will be about 125000 (5 courses per student) with a 10 %portion (about 12500) waiting.

[37] PBD page 244

```
01 ! Enter student ID and code of courses
02 select from student where student
03 ! check courses
04 loop #nrCoursesStudent
05     select from course where course
06     loop #PrereqCourse
07         select from prereq where course
08     if #ProbCourseClosed
09         update waiting num_rows = 1
10     else
11         update follows num_rows = 1
12 update trans num_rows = 1
```

Figure 67: **Exercise 9.2**: Transaction logic for the student requests

stmt	file/index	n_s	p_s	D^s_{CPU}	$D^s_{disk_1}$	$D^s_{disk_2}$
02	STUDENT	1	1	0.001	0.016	
05	COURSE	5	1	0.001		0.016
07	PREREQ	10	1	0.001	0.016	
09	WAITING	5	0.1	0.001		0.016
11	FOLLOW	5	0.9	0.001		0.016
12	TRANS	1	1	0.0005	0.008	
				D_{CPU}	D_{disk_1}	D_{disk_2}
				0.0215	0.184	0.160

Using the formula (9.4.7) from the book[38] with r left out as there is but one transaction:

$$\begin{aligned} D_{CPU} &= 1 \times 1 \times 0.001 + 5 \times 1 \times 0.001 \\ &\quad + 10 \times 1 \times 0.001 + 5 \times (0.1 + 0.9) \times 0.016 \\ &\quad + 1 \times 1 \times 0.0005 = 0.0065 \text{ seconds} \\ D_{CPU} &= 21.5 \times 0.001 = 0.0215 \text{ seconds} \\ D_{disk_1} &= 1 \times 1 \times 0.016 + 10 \times 1 \times 0.016 + 1 \times 1 \times 0.008 \\ &= 11 \times 0.016 + 0.008 = 0.184 \text{ seconds} \\ D_{disk_2} &= 5 \times 1 \times 0.016 + 5 \times (0.1 + 0.9) \times 0.016 \\ &= 0.160 \text{ seconds} \end{aligned}$$

The utilization can be calculated using equation (8.5.16) on page 213, leaving out the r as there is but one transaction:

$$\begin{aligned} U_{CPU} &= \lambda \times D_{CPU} \approx 0.020668 \times 0.0215 \approx 0.000444 \\ U_{disk_1} &= \lambda \times D_{disk_1} \approx 0.020668 \times 0.184 \approx 0.003803 \\ U_{disk_2} &= \lambda \times D_{disk_2} \approx 0.020668 \times 0.160 \approx 0.003306 \end{aligned}$$

The response time can be calculated using equation (8.5.15) on page 212 of the book:

[38] PBD page 244

$$R = \sum_{i=1}^{K} \frac{D_i}{1 - U_i} = \frac{0.0215}{1 - 0.000444} + \frac{0.184}{1 - 0.003803} + \frac{0.160}{0.003306}$$
$$R \approx 0.0215 + 0.185 + 0.161 = 0.37 \text{ seconds.}$$

Comparing case 1 and case 2 In both cases the hardware appears more than adequate to handle the expected average number of student requests. Response time in both cases is well below one second and thus for the students and the administration very acceptable.

Maximum throughput The question that remains unanswered is: what is the peak the system can handle. From case 2, it appears that disk 1 has the largest service demand. Using the formula for utilization plus the maximum attainable value of 100%, the maximum value for λ that can be processes appears to be:

Case 1 Minimal consistency check
$$\lambda_{max} \times D_{disk_2} = 1 \Leftrightarrow$$
$$\lambda_{max} = \frac{1}{D_{disk_2}} \approx \frac{1}{0.080} = 12.5 \text{ requests per second.}$$
Case 2 Complete registration processing
$$\lambda_{max} \times D_{disk_1} = 1 \Leftrightarrow$$
$$\lambda_{max} = \frac{1}{D_{disk_1}} \approx \frac{1}{0.184} = 5.4 \text{ requests per second.}$$

Subsecond response times Another point is at what request rate the response time will exceed one second. Using the formula for response time (as used before) this can be restated as:

Case 1 Minimal consistency check (9.3.1)

$$R = \sum_{i=1}^{K} \frac{D_i}{1 - U_i} = \frac{0.0065}{1 - 0.0065\lambda} + \frac{0.024}{1 - 0.024\lambda} + \frac{0.080}{1 - 0.080\lambda} = 1 \quad (9.3.2)$$

$$\lambda \approx 11.4 \text{ requests/second} \quad (9.3.3)$$

Case 2 Complete registration processing (9.3.4)

$$R = \sum_{i=1}^{K} \frac{D_i}{1 - U_i} = \sum_{i=1}^{K} \frac{D_i}{1 - \lambda D_i} = 1 \Leftrightarrow \quad (9.3.5)$$

$$\frac{0.0215}{1 - 0.0215\lambda} + \frac{0.184}{1 - 0.184\lambda} + \frac{0.160}{1 - 0.160\lambda} = 1 \quad (9.3.6)$$

$$\lambda \approx 3.7 \text{ requests/second} \quad (9.3.7)$$

(9.3.8)

If the peak requesting rate stays below about 4 requests/second, then case 2 might be viable with the current configuration. Otherwise, the case 2 configuration quickly becomes unable to process the requests. Up to 11 requests per second, using case 1 the response time will stay below one second. Neither case can process much more than 12 requests per second.

Given, that the peak requesting rate is currently unknown, it would be the safest bet to go for case 1 i.e. minimal consistency check.

9.4 Exercise 9.4

Given The spreadsheet `Ch09-Data.xls` and the chapter 9 case.

Requested Recompute the number of IOs generated by status viewing if an index on EmployeeId is not available for the TicketEmployee table.

Solution See also spreadsheet `Ch09-Data-ex-9.4.xls`.

Stmt no.	Index	No. of Exec.	Prob. Exec.	Index I/Os	Data I/Os	Total I/Os
2		1.0	1.0		1907928	1907928
5	TicketId	80.3	1.0	3	1.0	321.2
					Total	1908249.2

Figure 68: **Exercise 9.4**: Recalculated nr of I/O's without EmployeeId index

10 Chapter 10

10.1 Exercise 10.1

Given

- The "random walk through England".
- Two sons in stead of one with identical behaviour in the same period.
- When both are using a kayak on the same day, they share a kayak.

Requested

Father's question What percentage of days is neither son drinking in Leeds?

Lake districts relative's question How much time is there between the end of a visit and the start of a next visit to the Lake district?

Policeman's question How much days each month is one or both of the sons driving to London after drinking in Leeds?

Kayak renters' question How many days each month do they need to have a kayak available?

Solution The number of states for each son is 4, making 16 combinations of states possible. The state diagram will be complex, having a lot more state transitions than the original diagram. Finding the state transitions and their probabilities will be a lot of work.
However, we can easily calculate the probabilities for combinations of states of the two sons. From the original calculations we know the probabilities for state 1 to 4 are $p_1 = 0.2644, p_2 = 0.1586, p_3 = 0.2308, p_4 = 0.3462$[39].
Using a vector to represent the probabilities for each son to be in one of the 16 states, the resulting probabilities are the result of multiplying the vectors as follows[40]:

$$\begin{pmatrix} 0.2644 \\ 0.1586 \\ 0.2308 \\ 0.3462 \end{pmatrix} \begin{pmatrix} 0.2644 & 0.1586 & 0.2308 & 0.3462 \end{pmatrix} =$$

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 0.06990736 & 0.04193384 & 0.06102352 & 0.09153528 \\ 0.04193384 & 0.02515396 & 0.03660488 & 0.05490732 \\ 0.06102352 & 0.03660488 & 0.05326864 & 0.07990296 \\ 0.09153528 & 0.05490732 & 0.07990296 & 0.11985444 \end{pmatrix} \end{array} \tag{10.1.1}$$

In this matrix (a_{ij}) each value a_{ij} represents the state of son number 1 being in state i and son number 2 being in state j. Remark that the sum of each row i is equal to p_i. Also, the sum of each column j is equal to p_j, where i and j are in $\{1, 2, 3, 4\}$. Thus for instance the sum of row 1 is 0.2644. Finally, each item a_{ii} is equal to p_i^2.

[39] PBD, page 271

[40] the matrix was calculated using `R`, an open source freely available math package running both a Linux, Mac and a Windows version.

Father's question The percentage of days that neither of the sons is drinking in Leeds is equal to one minus the percentage of days that one or both sons are drinking in Leeds or

$$1 - (\sum_{j=1}^{4} a_{1j} + \sum_{i=1}^{4} a_{i1} - a_{11}) = 1 - (2 \times 0.2644 - 0.2644^2) \approx 0.541$$

or 54.1%.

Lake districts relative's question The probability that one or both of the sons are using a kayak in the Lake district is equal to

$$\sum_{j=1}^{4} a_{3j} + \sum_{i=1}^{4} a_{i3} - a_{33} = 2 \times 0.2308 - 0.2308^2 \approx 0.408$$

The cycle time[41] is the inverse of 0.408 or approximately 2.45 days. After a day of kayaking 1.45 days remain until one of the sons returns for a kayak.

Policeman's question One or both of the sons are in a pub in Leeds in the case of a_{1j} or a_{i1} for $i, j \in \{1, 2, 3, 4\}$. Thus, assuming 30 days a month, the number of days that one or both of the sons will be driving from Leeds to London after drinking in the pub will be equal to:

$$30 \times (0.6 \times (\sum_{j=1}^{4} a_{1j} + \sum_{i=1}^{4} a_{i1} - a_{11}) - 0.6 \times 0.6 \times a_{11})$$

$$= 30 \times (0.6 \times 2 \times 0.2644 - 0.6^2 \times 0.2644^2) \approx 8.76 \text{ days each month}$$

Remark that the case of both sons driving after a drink should be counted only once, hence the subtraction of $0.6^2 \times a_{11}$.

Kayak renters' question The probability that one or both of the sons are in the Lake district using a kayak is equal to

$$\sum_{j=1}^{4} a_{3j} + \sum_{i=1}^{4} a_{i3} - a_{33}$$

$$= 2 \times 0.2308 - 0.2308^2 \approx 0.408$$

.

Assuming 30 days in a month, this means $0.408 \times 30 \approx 12.24$ days a month.

10.2 Exercise 10.2

Given The original model and the answer to the company president's question in §10.6.

Requested Provide all the details to verify the answer.

Solution

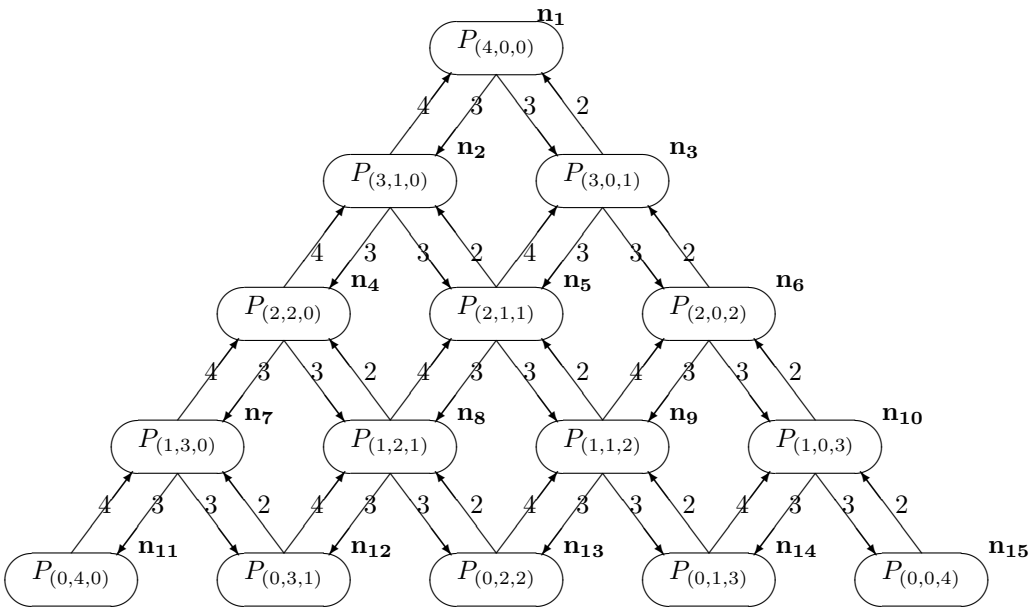


Figure 69: **Exercise 10.2** Markov diagram with 15 nodes ($n_1 \dots n_{15}$) for 4 users.

Specifying the model The modified state diagram (figure 69) shows the states and the transition weights.

Solving the model After renumbering the nodes $n_1 \dots n_{15}$ we can enter the weights into a matrix.
This matrix is $A = (a_{ij})$ where each element a_{ij} contains the incoming weights into node n_i from n_j for $i \neq j$, and a_{ii} contains the outgoing weights (negative). The values represent the balance equations $\sum arrows_{in} - \sum arrows_{out} = 0$. The fifteenth equation is deleted for being overspecified[42]. The equation $\sum_{i=1}^{n} p_i = 1$, where n is the number of nodes, is added as the last line.
The solution of $Ax = b$, where A is the weight matrix and b is the vector $(0, \dots, 0, 1)$ returns the probabilities of the solved Markov chain. Using `R` the result is determined as follows:

[41] mean time between each visit
[42] see PBD page 270 about the set of "in=out" equations being under-determined

$$A = \begin{pmatrix}
-6 & 4 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & -10 & 0 & 4 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & 0 & -8 & 0 & 4 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 3 & 0 & -10 & 0 & 0 & 4 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 3 & 3 & 0 & -12 & 0 & 0 & 4 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 3 & 0 & 0 & -8 & 0 & 0 & 4 & 2 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 3 & 0 & 0 & -10 & 0 & 0 & 0 & 4 & 2 & 0 & 0 & 0 \\
0 & 0 & 0 & 3 & 3 & 0 & 0 & -12 & 0 & 0 & 0 & 4 & 2 & 0 & 0 \\
0 & 0 & 0 & 0 & 3 & 3 & 0 & 0 & -12 & 0 & 0 & 0 & 4 & 2 & 0 \\
0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & -8 & 0 & 0 & 0 & 4 & 2 \\
0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & -4 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 3 & 3 & 0 & 0 & 0 & -6 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 3 & 0 & 0 & 0 & -6 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 3 & 0 & 0 & 0 & -6 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{pmatrix}$$

$$A \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \\ p_7 \\ p_8 \\ p_9 \\ p_{10} \\ p_{11} \\ p_{12} \\ p_{13} \\ p_{14} \\ p_{15} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \Leftrightarrow \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \\ p_7 \\ p_8 \\ p_9 \\ p_{10} \\ p_{11} \\ p_{12} \\ p_{13} \\ p_{14} \\ p_{15} \end{pmatrix} = \begin{pmatrix} 0.04287389 \\ 0.03215542 \\ 0.06431084 \\ 0.02411656 \\ 0.04823313 \\ 0.09646625 \\ 0.01808742 \\ 0.03617485 \\ 0.07234969 \\ 0.14469938 \\ 0.01356557 \\ 0.02713113 \\ 0.05426227 \\ 0.10852454 \\ 0.21704907 \end{pmatrix}$$

Interpretation Now the model is solved, the question of the company president can be answered[43].
Again the response time can be found using the interactive response time law, using a thinktime of 0: $R = \frac{M}{X_0}$. $M = 4$ is given and the throughput X_0 is to be calculated.
Just like in the original case, the throughput can be measured at the CPU. The number of states where the CPU is being used are all states in the first four levels (see diagram). This means the nodes $n_1 \dots n_{10}$. The CPU utilization is equal to the sum of the probabilities from $n_1 \dots n_{10}$ or equivalantly

$$1 - (p_{11} + p_{12} + p_{13} + p_{14} + p_{15}) = 1 - 0.4205326 = 0.5794674 \approx 57.95\%$$

$\boxed{U_{\text{CPU}} = 57.95\%}$

The service rate of the CPU is 6 transactions per minute. Therefore the throughput measured at the CPU is equal to $0.5795 \times 6 = 3.4768$ transactions

$\boxed{X_0 = 3.4768 \text{ tps}}$

[43] which is the combination of the user's question and the system administrator's question with 4 users in stead of 2.

per minute. The average response time is then $\frac{4}{3.4768} \approx 1.1505$ minutes per transaction or $1.1505 \times 60 = 69.03$ seconds per transaction. This concludes the verification of the answer to the company president's question.

$\boxed{R = 69.03 \text{ sec}}$

10.3 Exercise 10.3

Given None.

Requested Is the mean recurrence time for a periodic state always equal to its period?

Solution No, it is not necessarily equal to its period. According to the definition, the system *can* return to the periodic state in $p, 2p, 3p, \ldots$ steps, where $p > 1$. This implies a non-zero probability, but not a certainty.
If a periodic state is part of two cycles with different periods, then transition to one or the other state depends on a probability. Thus, the mean recurrence time for this state will not be a multiple of one of the periods, but a weighted average of both periods.

10.4 Exercise 10.4

Given Figure 10.7 in the book (page 281) with the probabilities:

$$P_{DE} = \tfrac{1}{3}, P_{DA} = \tfrac{2}{3}$$
$$P_{ED} = \tfrac{3}{4}, P_{EF} = \tfrac{1}{4}$$
$$P_{HH} = \tfrac{1}{2}, P_{HI} = \tfrac{1}{2}$$

and all other probabilities are 1.

Requested Give the mean recurrence for each of the recurrent states.

Solution The mean recurrence time for A, B and C is equal to the period, because once the system reaches A, B or C, it is guaranteed to recurr every 3 visits.
The mean recurrence time for F, G, H and I is more complex to calculate. Each round may contain zero, one or more rounds of $H - H$. Let r_k be the probability for a round of $F - G - H - H^k - I$ where H^k represents the k rounds $H - H$. The probability r_0 for the shortest round $F - G - H - I$ (without recurring in H) is equal to $1 \times 1 \times \frac{1}{2} \times 1 = \frac{1}{2}$. The probability r_1 for a round of $F - G - H - H^1 - I$ (with one recurrence of H) is equal to $1 \times 1 \times \frac{1}{2} \times \frac{1}{2} \times 1 = \frac{1}{4}$. In general for a round $F - G - H - (H^k) - F$ with k the number of recurrences of H, is equal to

$$r_k = \frac{1}{2^{k+1}} \tag{10.4.1}$$

First, remark that the sum of all probabilities within the cycle $F - G - H(-H^k) - I$ is equal to 1:

$$\sum_{i=0}^{\infty} \frac{1}{2^{k+1}} = \sum_{i=1}^{\infty} \frac{1}{2^k} = 1 \qquad \text{(see appendix A.1)}$$

Using equation 10.4.1 the recurrence time for a certain value of k is equal to $4+k$ times the probability for k recurrences of H, with $k \in \{0, 1, 2, \ldots\}$ i.e. $(4+k)r_k = \frac{4+k}{2^{k+1}}$. The mean recurrence time for F, G, and I is thus equal to

$$\lim_{n\to\infty} \sum_{i=0}^{n} \frac{4+i}{2^{i+1}} = \lim_{n\to\infty} \sum_{i=0}^{n} \left(\frac{4}{2^{i+1}} + \frac{i}{2^{i+1}} \right) \tag{10.4.2}$$

$$4 \times \lim_{n\to\infty} \sum_{i=0}^{n} \frac{1}{2^{i+1}} + \lim_{n\to\infty} \sum_{i=0}^{n} \frac{i}{2^{i+1}} \tag{10.4.3}$$

Using the result of appendix A.1 the first term is equal to

$$4 \times \lim_{n\to\infty} \sum_{i=1}^{n} \frac{1}{2^{i}} = 4 \times 1 = 4 \tag{10.4.4}$$

Using the result of appendix A.3 the second term is equal to

$$\frac{1}{4} \lim_{n\to\infty} \sum_{i=0}^{n} \frac{i}{2^{i-1}} = \frac{1}{4} \times \frac{1}{(1-\frac{1}{2})^2} = \frac{1}{4} \times 4 = 1 \tag{10.4.5}$$

Bringing both terms together the mean recurrence time is equal to $4 + 1 = 5$.
For node H the mean recurrence time is determined by a slightly different formula due to the recurrence $H - H^k$. The probability for a tour of 4 nodes is equal to $\frac{1}{2}$. The probability for a tour of one node is equal to $\frac{1}{2}$ too.
The mean recurrence time for node H is thus equal to:

$$4 \times \tfrac{1}{2} + 1 \times \tfrac{1}{2} = 2.5$$

	mean
	recurrence
A	3
B	3
C	3
F	5
G	5
H	2.5
I	5

Figure 70: **Exercise 10.4**: mean recurrence times

10.5 Exercise 10.5

Given None.

Requested Give a two state Markov model where the mean recurrence time for state 1 is 1.75.

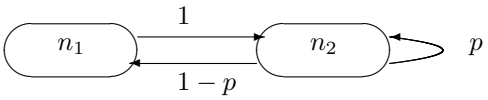


Figure 71: **Exercise 10.5** Markov diagram with 2 nodes

Solution Consider the model drawn in figure 71.
A mean recurrence time of 1.75 for state n_1 will hold if the following relationship is true:

$$\sum_{i=0}^{\infty} i(1-p)p^i = 1.75$$

This relationship holds for a value of $p = \frac{7}{11}$. This can be derived from the results of appendix A.3 as follows:

$$\sum_{i=0}^{\infty} i(1-p)p^i =$$

$$p(1-p)\sum_{i=0}^{\infty} ip^{i-1} =$$

using [**A.3**]

$$p(1-p)\frac{1}{(1-p)^2} = \frac{p}{1-p}$$

Using equation 10.5 as equal to 1.75 resolves into the value for $p = \frac{7}{11}$. A common sense check of the result can be done with `R` and the following statements:

$\boxed{p = \frac{7}{11}}$

```
% (see file ex-10-5-result.txt)
> p <- 7/11
> x <- 1:100000
> f <- x * (1-p) * p^x
> sum(f)
[1] 1.75
>
```

The last line shows the result: 1.75.

explanation R statements The first statement generates a sequence from 1 to 100,000 and assigns these to x. This approximates $0 \ldots \infty$ close enough for this recurrence expression. The second statement assigns the probability value in the model to p. The original mean recurrence relation is assigned to h, giving h a list of values concurring with each x. The required mean recurrence time is then the sum of alle elements in h, which results in the value 1.75, as you can check if you install and run `R`.
The resulting Markov model is depicted in figure 72.

10.6 Exercise 10.6

Given A small store with one worker that can contain a maximum of 2 customers. On average a service time of 5 minutes. The acceptable service level is 7.5 minutes in the store.

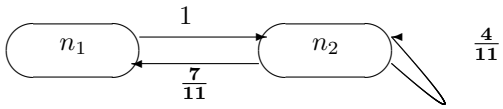


Figure 72: **Exercise 10.5** Resulting Markov diagram with 2 nodes.

.

Requested What is the maximum allowable arrival rate of customers to the store to guarantee the service level?

Solution The problem is a single server question with service time $D = 5$ minutes and R has the maximum value of 7.5 minutes. Using the response formula we get a maximum arrival rate of approximately $0.26\not{0}$ arrivals per minute or 16 arrivals per hour[44].

$$\left.\begin{array}{r} R = \dfrac{M}{X_0} - Z \Leftrightarrow \\ X_0 = \frac{M}{R+Z} = \lambda \\ Z = 0, R = 7.5, M = 2 \end{array}\right\} \Rightarrow$$

$$\begin{aligned} \Rightarrow X_0 = \lambda = \frac{2}{7.5} &= 0.26\not{0} \text{ arrivals per minute} \\ &= 0.26\not{0} \times 60 = 16 \text{ arrivals per hour} \end{aligned}$$

10.7 Exercise 10.7

Given None.

Requested Give a 2 state Markov diagram, where the sum of the mean recurrence time is as small as possible.

Solution The mean recurrence time for any node, is determined by the number of nodes in a period and the probability of taking this particular cycle. The smallest recurrence cycle attainable is by using the self-recurrent relation. I am not sure whether Dr. Menascé meant this degenerate case, when he specified this exercise. Working from the general case, the result is the degenerate case.
The general diagram has two states and four state transitions. For each state transition from state i to state j a probability p_{ij} is specified.

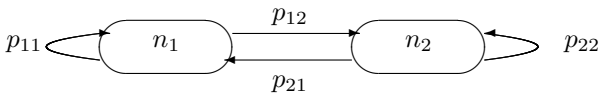


Figure 73: **Exercise 10.7** General recurrence.

The sum of the recurrence times is equal to

[44]The arrival rate is deemed equal to the throughput, as the system is in a steady state

$$\sum_{\substack{i=1\\j=1}}^{2} n \times \frac{1}{p_{ij}} \qquad \text{where n = nr of nodes in the period}$$

where p_{ij} is the probability of the transition from node i to node j. For this sum to be as small as possible, n must be as small as possible, with zero (self-recurrent) the absolute minimum, and p_{ij} must be as large as possible. It is thus clear, that the minimum sum of mean recurrence time is reached with $p_{12} = p_{21} = 0$ and $p_{11} = p_{22} = 1$. This results in the following degenerate case:

Figure 74: **Exercise 10.7** Trivial recurrence.

10.8 Exercise 10.8

Given

- A workload consisting of one completely CPU-bound job and one completely disk-bound job.
- A model considering only one class with 2 identical jobs spending half of their time at the CPU and half of their time at the disk.

Requested How much error is made in making this assumption? Choose from system throughput, device utilization, response time or some other important performance metric.

Solution Let j_1 be job 1, which is CPU-bound. Let j_2 be job2, which is disk-bound. Also, let the service times have the following values:

	j_1	j_2
CPU	$D_{\text{CPU},1} > 0 \rightarrow d_1$	$D_{\text{CPU},1} = 0$
disk	$D_{\text{CPU},1} = 0$	$D_{\text{CPU},1} > 0 \rightarrow d_2$

Further, let the arrival rates be λ_1 and λ_2, where λ_i is the arrival rate of j_i. Using these definitions, we can calculate utilization and response times:

$$\begin{aligned} U^A_{\text{CPU}} &= \lambda_1 d_1 \quad \wedge \quad U^A_{\text{disk}} = \lambda_2 d_2 \\ R^A_1 &= \frac{d_1}{1-\lambda_1 d_1} \quad \wedge \quad R^A_2 = \frac{d_2}{1-\lambda_2 d_2} \end{aligned}$$

In the model of the analyst, the arrival rate will be equal to the sum of original arrival rates, i.e. $\lambda = \lambda_1 + \lambda_2$. Also, the service demands are $\frac{d_1}{2}$ for the CPU and $\frac{d_2}{2}$ for the disk. Using these assumptions the calculations for utilization and response time become:

$$U^B_{\text{CPU}} = \lambda \frac{d_1}{2} = \frac{1}{2}\lambda d_1 = \frac{1}{2}(\lambda_1 + \lambda_2) d_1$$

$$U^B_{\text{disk}} = \lambda \frac{d_2}{2} = \frac{1}{2}\lambda d_2 = \frac{1}{2}(\lambda_1 + \lambda_2)d_2$$

$$\begin{aligned} R^B &= \frac{\frac{d_1}{2}}{1 - \frac{1}{2}(\lambda_1 + \lambda_2)d_1} + \frac{\frac{d_2}{2}}{1 - \frac{1}{2}(\lambda_1 + \lambda_2)d_2} \\ &= \frac{d_1}{2 - (\lambda_1 + \lambda_2)d_1} + \frac{d_2}{2 - (\lambda_1 + \lambda_2)d_2} \end{aligned}$$

Using these values we can derive the error by subtracting the real value from the modeled value. For the utilization:

$$\begin{aligned} U^B_{\text{CPU}} - U^A_{\text{CPU}} &= \tfrac{1}{2}\lambda_1 d_1 + \tfrac{1}{2}\lambda_2 d_1 - \lambda_1 d_1 \\ &= \tfrac{1}{2}(\lambda_2 - \lambda_1)d_1 \\ U^B_{\text{disk}} - U^A_{\text{disk}} &= \tfrac{1}{2}\lambda_1 d_2 + \tfrac{1}{2}\lambda_2 d_2 - \lambda_2 d_2 \\ &= \tfrac{1}{2}(\lambda_1 - \lambda_2)d_2 \end{aligned}$$

From this result one can see that size of the service demand and the difference of sizes in arrival rates determine the error for utilization. If the arrival rates are different, then the size of the error depends on both. If the arrival rates are the same, then the size of the service demands are irrelevant, as the error will be zero! The error for the response time is relative to the 2 jobs. Both errors will have analogue errors, so we study just the error relative to j_1:

$$R^B - R^A_1 = \frac{d_1}{2 - (\lambda_1 + \lambda_2)d_1} + \frac{d_2}{2 - (\lambda_1 + \lambda_2)d_2} - \frac{d_1}{1 - \lambda_1 d_1}$$

In this case the error compared to R^A_1 is not just determined by the error in response time due to job 1, but also due to job2. Secondly, the error is less easy to relate to service demands and arrival rates. Still, assuming $\lambda_1 = \lambda_2$ and filling this in in the error, gives

$$R^B - R^A_1 = \frac{d_1}{2 - d_1\lambda_1} + \frac{d_2}{2 - d_2\lambda_1} - \frac{d_1}{1 - d_1\lambda_1}$$

Subsequently, assuming the service demands are the same or nearly the same gives:

$$\begin{aligned} R^B - R^A_1 &= \frac{d_1}{2 - d_1\lambda_1} + \frac{d_1}{2 - d_1\lambda_1} - \frac{d_1}{1 - d_1\lambda_1} \\ &= \frac{2d_1}{2 - d_1\lambda_1} - \frac{d_1}{1 - d_1\lambda_1} \\ &= \frac{-d_1\lambda_1}{2 - 3d_1\lambda_1 + d_1^2\lambda_1^2} \end{aligned}$$

This shows, that response time is very sensitive to ignoring the two classes, even if arrival rates and service demands are reasonably equal. Utilization, however, given the same arrival rates, shows no error, independant of service demand.

10.9 Exercise 10.9

Given A fitness room for students with one treadmill and two bikes. The students arrive at regular interval of approximately 10 minutes (exponentially distributed). The first device the students use is the treadmill followed by one of the bikes. Each device is used for 15 minutes. A student that finds the desired device not available returns home (frustrated). If a student is able to finish the tour, he returns home (happy).

Requested

- Draw a labeled Markov model for this situation.
- Calculate the steady state probabilities for each of the states.
- Give the mean number of students using the stationary bikes.
- What percentage of students return home (frustrated), because the treadmill is taken?
- What is the throughput of students that return home (happy)?

Solution

Creating the model The Markov model has 2 main variables: the treadmill and the bikes. The model uses the notation (i, j) where i is the number of students on the treadmill, either zero or one, and j is the number of students on the bikes, zero, 1 or 2. This leads to 6 states: (0,0), (1,0), (0,1), (1,1), (0,2), (1,2). The Markov model is drawn in figure 75.

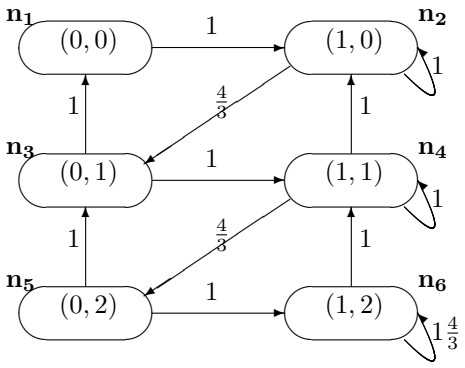


Figure 75: **Exercise 10.9** the Markov model for students and fitness devices.

The interarrival time of 10 minutes leads to an arrival rate of 6 students per hour. The usage of devices leads to 4 students per hour finishing a device, once the device is taken. Thus the weights on the graph are calculated as follows:

- The first assumption is that the arrivals are evenly distributed among the 6 nodes of the model. This is important for distributing the weights among the vertices, as they are now also distributed evenly. As there are 6 states where a student can arrive, each vertex symbolizing an arrival gets a weight of one. This leaves a total weight of 3 for students that get in and 3 for students that go home frustrated, after finding someone on the treadmill.

- The transfer from the treadmill to the bike can take place from 3 states: (1,0), (1,1) and (1,2). As the treadmill takes 15 minutes, 4 students per hour switch to a bike. Assuming the probabilities for these state transitions are evenly distributed, this makes for a weight of $\frac{4}{3}$ for each transition.
- The transfer from the bike home (happy), happens 4 times: (0,1), (1, 1), (0, 2) and (1, 2). As the bikes also take 15 minutes, 4 students per hour may go home (happy). Thus the weight assigned is equal to 1 in all cases.

Solving the model From this model using the "flow in = flow out" set of equations, we get the following equations:

$$\begin{aligned}
1: \quad & P_{(0,1)} && = P_{(0,0)} \\
2: \quad & P_{(0,0)} + P_{(1,1)} && = \frac{4}{3} P_{(1,0)} \\
3: \quad & P_{(0,2)} + \tfrac{4}{3} P_{(1,0)} && = P_{(0,1)} \\
4: \quad & P_{(0,1)} + P_{(1,2)} && = 1\frac{4}{3} P_{(1,1)} \\
5: \quad & \tfrac{4}{3} P_{(1,1)} && = P_{(0,2)} \\
6: \quad & P_{(0,2)} && = P_{(0,2)}
\end{aligned}$$

We relabel the variables as indicated in the margin and in the diagram (with n_i), we create a matrix and at the same time get rid of the fractions by multiplicating across the equations involved. The last row in the matrix is replaced by the equation $\sum_{i=1}^{6} p_i = 1$ and the resulting matrix is fed into the package R together with the vector $b = (0, 0, 0, 0, 0, 1)$ and the request to solve the equation $Ap = b$ for the vector p. The result is:

$$\begin{pmatrix} -1 & 0 & 1 & 0 & 0 & 0 \\ 3 & -4 & 0 & 3 & 0 & 0 \\ 0 & 4 & -6 & 0 & 3 & 0 \\ 0 & 0 & 3 & -7 & 0 & 1 \\ 0 & 0 & 0 & 4 & -6 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

$$\Leftrightarrow \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \end{pmatrix} = \begin{pmatrix} 0.21739130 \\ 0.26086957 \\ 0.21739130 \\ 0.13043478 \\ 0.08695652 \\ 0.08695652 \end{pmatrix}$$

question: mean number of students on the bikes The states $3 \ldots 6$ have one or more students on the bikes. For each state, the number of students on the bike is equal to $b_i \times P_i$, where b_i is the number of students on the bike in state i. Thus the average number of students the bikes is equal to

$$\sum_{n=1}^{6} b_i P_i = 0 \times P_1 + 0 \times P_2 + 1 \times P_3 + 1 \times P_4 + 2 \times P_5 + 2 \times P_6$$
$$= P_3 + P_4 + 2(P_5 + P_6) \approx 0.696$$

This means that on average one bike is taken, about 70% of the time.

Frustration percentage The treadmill is taken in the states 2, 4 and 6. As the students arrive according to an exponential distribution[45], the percentage of students that arrive only to return home frustrated is proportional to the amount of time that the states 2, 4 and 6 are active. Thus, the percentage of arriving students that will find the treadmill occupied and return home frustrated will be equal to $P_2 + P_4 + P_6 \approx 0.478$.

Happy throughput As calculated before, on average 0.696 students are using the bikes. Thus the throughput of happy students is equal to 4 students per hour per bike times 0.696 or $2 \times 4 \times 0.696 \approx 5.6$ students per hour.

10.10 Exercise 10.10

Given The data of exercise 10.9 is given with some modifications:

- A rowingmachine is added.
- The treadmill is followed by the bike with 0.3 probability or the rowingmachine with 0.7 probability.
- After using the bike, the student always uses the rowing machine.
- After using the rowing machine, the student always goes to the treadmill
- Each piece of equipement is used for 3 minutes on average.
- There are always 2 students in the weight room.
- The student waits if the required equipment is not available i.e. taken.
- Each student visits the rowing machine 3 times on average.

Requested

- Draw the Markov model of this system. Clearly mark all the arcs.
- How many times does the typical student use the treadmill?
- How manyu times does the typical student use the rowing machine?
- What are the steady state probabilities of each state?
- What is the utilization of each piece of equipment?
- How many students leave the weight room per hour?
- How much time does the typical student stay in the weight room (i.e. what is the response time per student)?

[45] actually, negative exponential distribution

Solution

A Markov diagram for one student is drawn in figure 76. In table 77 all possible states are encoded and labeled with a name. The states are named according to the place of the students on the devices:

TT One student on the treadmill and one student waiting.

TR One student on the treadmill and one student on the rowing machine

TB One student on the treadmill and one student on one of the bikes

RR One student on the rowing machine and one student waiting

RB One student on the rowing machine and one student on one of the bikes

BB Both students on the bikes

Determining the weights should be done carefully. To avoid mistakes a check is included that the total of the weight is 2 in all outgoing transitions for one state (figure 78).
Using the transition table as input the Markov model for two students can now be constructed (figure 79).

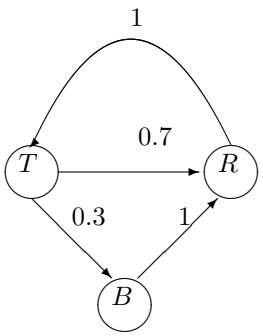


Figure 76: **Exercise 10.10** The Markov model for one student.

name	$Wait_T$	T	$Wait_R$	R	B
TT	1	1	0	0	0
TR	0	1	0	1	0
TB	0	1	0	0	1
RR	0	0	1	1	0
RB	0	0	0	1	1
BB	0	0	0	0	2

Figure 77: **Exercise 10.10**: A state matrix encoding

Using the model as drawn in figure 79, the transition matrix can be constructed with the incoming weights in the row item a_{ij} where $i \neq j$, and the sum of the outgoing weights in the items a_{ii} (figure 80).
Using transition matrix with the TT row replaced by the P row, and solving the equation $Ap = b$ in `R` where p is the vector to be solved (the probabilities) and b is the last column the problem is solved using `R`.
Thus, the solution of the Markov chain is:

transition	probability
$T \to R$	0.7
$\to B$	0.3
$R \to T$	1
$B \to R$	1

state	$\mathbf{S_1(s)}$	T	R	B	check
RR	S_1(R)	1			
	S_2(R)	1			
	weight	**2 TR**			2
TT	S_1(T)		0.7	0.3	
	S_2(T)		0.7	0.3	
	weight		**1.4 TR**	**0.6 TB**	2
TR	S_1(T)		0.7	0.3	
	S_2(R)	1			
	weight	**1 TT**	**0.7 RR**	**0.3 RB**	2
RB	S_1(R)	1			
	S_2(B)		1		
	weight	**1 TB**	**1 RR**		2
TB	S_1(T)		**0.7 RB**	0.3	
	S_2(B)		**1 TR**		
	weight		1.7	**0.3 BB**	2
BB	S_1(B)		1		
	S_2(B)		1		
	weight		**2 RB**		2

Figure 78: **Exercise 10.10**: single and dual transition tables

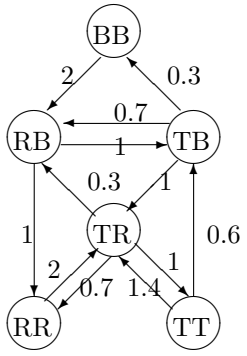


Figure 79: **Exercise 10.10** The Markov model for two students.

$$P = \begin{pmatrix} P_{BB} \\ P_{RB} \\ P_{TB} \\ P_{TR} \\ P_{RR} \\ P_{TT} \end{pmatrix} = \begin{pmatrix} 0.01701323 \\ 0.11342155 \\ 0.11342155 \\ 0.37807183 \\ 0.18903592 \\ 0.18903592 \end{pmatrix}$$

	BB	RB	TB	TR	RR	TT	b
BB	-2		0.3				0
RB	2	-2	0.7	0.3			0
TB		1	-2			0.6	0
TR			1	-2	2	1.4	0
RR		1		0.7	-2		0
TT				1		-2	0
P	1	1	1	1	1	1	1

Figure 80: **Exercise 10.10**: Transition matrix

```
> A <- read.table("ex10.10.txt")
> A
   BB RB   TB   TR RR  TT
BB -2  0  0.3  0.0  0 0.0
RB  2 -2  0.7  0.3  0 0.0
TB  0  1 -2.0  0.0  0 0.6
TR  0  0  1.0 -2.0  2 1.4
RR  0  1  0.0  0.7 -2 0.0
P   1  1  1.0  1.0  1 1.0
> b <- c(0,0,0,0,0,1)
> x <- solve(A,b)
> sum(x)
[1] 1
> x
[1] 0.01701323 0.11342155 0.11342155 0.37807183 0.18903592 0.18903592
```

Figure 81: **Exercise 10.10**: The results from `R`

Answers to the questions

How many times does the typical student use the treadmill? From the fact that students wait their turn in stead of leaving, we can deduce, that each student follows the same path as he/she would in the case of one student in the room. So in determining the number of times the student uses the treadmill, we can use the one student model.
From the one student model one can conclude, that a typical student will visit the treadmill the same number of times as he/she visits the rowing machine, because each visit to the rowing machine is followed by a visit to the treadmill, unless the student leaves and a different student enters and follows to the treadmill. Further, each visit to the treadmill is certainly followed by a visit to the rowing machine — sometimes via a bike — unless the student leaves and the next student follows to the rowing machine.
So the typical student visits the treadmill on average 3 times. The calculations in the next two paragraphs support this reasoning.

System throughput X_0 The throughput for the rowing device is equal to the utilization times the service rate:

$$X_R = U_R \times \mu_R$$

We already know that the service rate is equal to $\frac{1}{3}$ as the average service time is 3 minutes per visit for each device. The utilization is the sum of probabilities that a student is working on the device. In the case of the rowing machine, there is only one device, so each probability represents the utilization of that device. For the rowing machine:

$$U_R = P_{TR} + P_{RR} + P_{RB} \approx 0.3781 + 0.1890 + 0.1134 = 0.6805$$

$$X_R = U_R \times \mu_R \approx \frac{0.6805}{3} \approx 0.2268$$

So the throughput of the rowing machine is equal to about 0.2268 people per minute. Knowing that a typical student will visit the rowing machine 3 times we can calculate the throughput:

$$X_R = V_R \times X_0 \Leftrightarrow$$

$$X_0 = \frac{X_R}{V_R} = \frac{0.2268}{3} = 0.0756 \text{ students per minute}$$

Throughput of treadmill X_T The throughput for the treadmill can be calculated as the product of utilization for the treadmill and the service rate, which is $\frac{1}{3}$. The utilization for the treadmill is equal to:

$$U_T = P_{TB} + P_{TR} + P_{TT} \approx 0.1134 + 0.3781 + 0.1890 = 0.6805$$

$$X_T = U_T \times \mu_T \approx \frac{0.6805}{3} \approx 0.2268$$

Using the forced flow law and the system throughput calculated earlier, we can calculate the number of visits for the treadmill, as follows:

$$X_T = V_T \times X_0 \Leftrightarrow V_T = \frac{X_T}{X_0} = \frac{0.2268}{0.0756} = 3$$

This calculation supports the above reasoning, that the number of visits for the treadmill must be equal to the number of visits to the rowing machine, on average.

How many times does a typical student use the rowing machine? The typical student visits the rowing machine 3 separate times, is a given fact. There is no way to derive this from the other data that I know of or could think of. In other words, the given is necessary to calculate the system throughput and response time.

What are the steady state probabilities of each state? The steady state probabilities were calculated earlier, and the result is:

$$P = \begin{pmatrix} P_{BB} \\ P_{RB} \\ P_{TB} \\ P_{TR} \\ P_{RR} \\ P_{TT} \end{pmatrix} = \begin{pmatrix} 0.01701323 \\ 0.11342155 \\ 0.11342155 \\ 0.37807183 \\ 0.18903592 \\ 0.18903592 \end{pmatrix}$$

What is the utilization of each pieace of equipment? The utilization for the treadmill and the rowing machine were calculated before and both were equal to 0.2268 or 22.68%.
The utilization of the bikes is equal to the utilization percentages as solved in the model, devided by two where only one device is taken:

$$U_B = P_{BB} + \frac{P_{RB} + P_{TB}}{2} \approx 0.0170 + \frac{0.1134 + 0.1134}{2} \approx 0.1304 \text{ per bike}$$

So each bike has a utilization of about 13%.

How many students leave the weight room per hour? As the throughput for the system X_0 is equal to 0.0756, this means that approximately $0.0756 \times 60 = 4.5$ students leave the room per hour.

How much time does the typical student stay in the weight room? Using the formula for response time in a closed model, because there are 2 students in the room at all times, and considering a thinktime of zero, the response time is equal to:

$$R = \frac{M}{X_0} = \frac{2}{0.0756} \approx 26.5 \text{ minutes}$$

Each student stays in the weight room for an average of 26.5 minutes.

10.11 Exercise 10.11

Given An ISP with m dial-up ports. The arrival rate of connection requests is λ requests per second. The completion rate is μ requests per second and the session duration is $1/\mu$ seconds.

Requested

- The state transition diagram with arrival and departure rates.
- The probability $p_k(k = 0, \ldots m)$ that k connections are active as a function of λ, μ and m.
- Find an expression for the probability of p_{busy} that a customer finds all dial-up ports busy as a function of the state probabilities p_k.

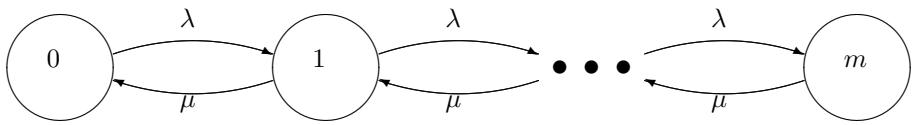


Figure 82: **Exercise 10.11**: State transition diagram

Solution The state transition diagram (figure 82 shows the adheres to a general Birth-Death model, with the addition that the arrival and completion rates are not state dependant, which in the general case, they are.

Using the general solution of the GBD theorem the probabilities p_k are equal to:

$$P_k = \left[\sum_{k=0}^{\infty}\prod_{i=0}^{k-1}\lambda_i/\mu_{i+1}\right]^{-1}\prod_{i=0}^{k-1}\lambda_i/\mu_{i+1}$$

Because the arrival and completion rates are state independant, this can be altered to

$$P_k = \left[\sum_{k=0}^{m}\lambda^k/\mu^k\right]^{-1}\lambda^k/\mu^k \Leftrightarrow$$

$$P_k = \left[1+\lambda/\mu+\cdots+\lambda^m/\mu^m\right]^{-1}\lambda^k/\mu^k \Leftrightarrow$$

$$P_k = \frac{\lambda^k/\mu^k}{1+\lambda/\mu+\cdots+\lambda^m/\mu^m} \Leftrightarrow$$

$$P_k = \frac{\lambda^k\mu^{m-k}}{\mu^m+\lambda\mu^{m-1}+\cdots+\lambda^m} \tag{10.11.1}$$

In state m all dial-up ports are in use. The probability for this to happen is equal to p_m. Using equation 10.11.1:

$$P_{\text{busy}} = P_m = \frac{\lambda^m\mu^{m-m}}{\mu^m+\lambda\mu^{m-1}+\cdots+\lambda^m} \Leftrightarrow$$

$$P_{\text{busy}} = \frac{\lambda^m}{\mu^m+\lambda\mu^{m-1}+\cdots+\lambda^m}$$

10.12 Exercise 10.12

Given A webserver is subjected to 10 load tests. Test n has a fixed number of n requests executing at the server. For each test the average response time R is measured (see table 10.1 on page 288 of PDB).

Requested

- Find the throughput of the webserver for each of the 10 tests.
- Assume the above throughput characteristics and an arrival rate of 2 requests per second. Determine the fraction f of rejected requests, if the server is servicing a number of requests equal to W, for $W \in \{0, 1, \ldots, 10\}$.
- Plot a graph of f versus W

Solution X0 The system can be viewed as a closed QN model, because the number of customers is fixed and known for each test. As the number customers is held constant, the thinktime between transactions is zero. Therefore according to the Interactive Response Time Law, the response time R is equal to

$$R = \frac{M}{X_0} \Leftrightarrow$$

$$X_0 = \frac{M}{R} \tag{10.12.1}$$

Using equation 10.12.1 the tabel is amended to include the throughput.

n	R (in sec)	X_0
1	1.400	0.714
2	2.114	0.946
3	2.838	1.057
4	3.570	1.120
5	4.311	1.160
6	5.060	1.186
7	5.816	1.204
8	6.578	1.216
9	7.346	1.225
10	8.119	1.232

Solution graph f versus W The exercise is very much like the previous exercise (10.11), where the m dial-up ports are replaced by the W maximum number of concurrent processes. The Markov model that is usable in this case should be alike.
The arrival rate is 2 requests per second. The completion rate is equal to the throughput. Because the number of processes is kept at an exact number of W, the throughput obtained in the previous experiments, can be used as completion rate.
The system can be solved using the GBD-theorem where arrival and completion rates are independent of the states and where W is the maximum state attainable. The probability of rejection in this case is equal to (see the result of exercise 10.11 for p_{busy})

$$p_{\text{busy}} = \frac{\lambda^W}{X_0^W + \lambda X_0^{W-1} + \cdots + \lambda^W} \tag{10.12.2}$$

The fraction f of rejections is equal to the probability of being busy times the arrival rate, i.e.

$$f(W) = \lambda \times p_{\text{busy}} \Leftrightarrow$$
$$f(W) = \frac{\lambda^{W+1}}{X_0^W + \lambda X_0^{W-1} + \cdots + \lambda^W}$$

Remark that the throughput also depends on W, and we will use the values calculated in the previous item. The first 3 entries for $f(W)$ are calculated explicitely. The results are in table 83 and in the graph (figure 84).

$$f(1) = \frac{2^2}{0.714^1 + 2} = \frac{4}{2.714} \approx 1.4738$$

$$f(2) = \frac{2^3}{0.946^2 + 0.946 \times 2 + 2^2} \approx \frac{8}{6.7869} \approx 1.1787$$

$$f(3) = \frac{2^4}{1.057^3 + 1.057^2 \times 2 + 1.057 \times 2^2 + 2^3} \approx \frac{16}{15.6434} \approx 1.0228$$

n	R (in sec)	X_0	fraction
1	1.400	0.714	1.4737
2	2.114	0.946	1.1787
3	2.838	1.057	1.0227
4	3.570	1.120	0.9309
5	4.311	1.160	0.8734
6	5.060	1.186	0.8358
7	5.816	1.204	0.8104
8	6.578	1.216	0.7928
9	7.346	1.225	0.7807
10	8.119	1.232	0.7721

Figure 83: **Exercise 10.12**: fraction results

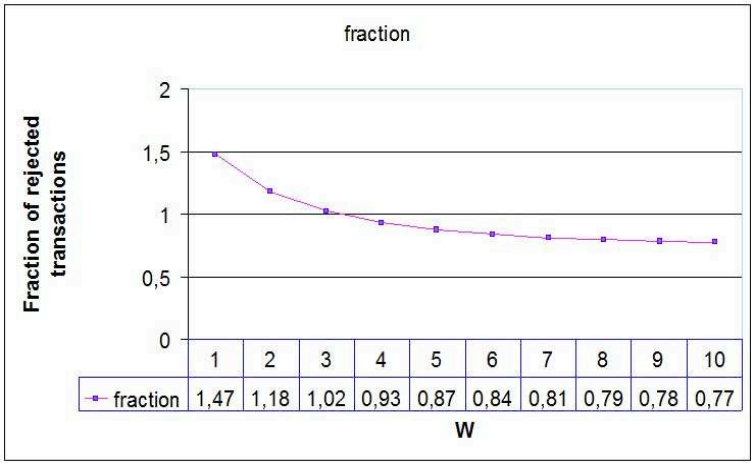


Figure 84: **Exercise 10.12**: Graph of fraction f versus W

11 Chapter 11

11.1 Exercise 11.1

Requested Show that in a G/G/1 queue, the average number of customers at the server is equal to the utilization of the server.

11.1.1 Solution

First, the server can only serve one customer in any period of time. This already indicates, that the average number of customers at the server will never exceed one. Second, remark that if no customer is at the server this indicates that the server is idle.
Let p be the probability that a customer is in the system. The average utilization of the system is then equal to p. The number of customers in the system is then equal to

$$0 \times (1-p) + 1 \times p = p$$

In conclusion, the average number of customers in the server is equal to the utilization of the server.

11.2 Exercise 11.2

Given The GBD-theorem.

Requested Derive equations (11.3.9) and (11.3.11) from the book[46].

Solution 11.3.9 Equation (11.3.9) $p_k = (1-\rho)\rho^k \quad k = 0, 1, \cdots$ can be derived as follows.
The GBD specificies p_k as

$$p_k = \left[\sum_{k=0}^{\infty} \prod_{i=0}^{k-1} \lambda_i/\mu_{i+1}\right]^{-1} \prod_{i=0}^{k-1} \lambda_i/\mu_{i+1} \quad \text{for } k = 0, 1, 2, \ldots \tag{11.2.1}$$

The first term in equation 11.2.1 is equal to p_0. The GBD-theorem also specifies that the utilization is equal to $1 - p_0$. So we can rewrite equation 11.2.1 to

$$p_k = p_0 \times \prod_{i=0}^{k-1} \lambda_i/\mu_{i+1} \quad \text{for } k = 0, 1, 2, \ldots$$

$$p_k = (1-\rho) \times \prod_{i=0}^{k-1} \lambda_i/\mu_{i+1} \quad \text{for } k = 0, 1, 2, \ldots$$

Because μ_i is equal to μ and λ_i is equal to λ for all $i \in \{1, 2, \cdots\}$, and because $1/\mu$ is equal to $E[S]$ we can rewrite the last equation to

$$p_k = (1-\rho) \times \left(\frac{\lambda}{\mu}\right)^k$$

$$p_k = (1-\rho) \times (\lambda \times E[S])^k$$

And because $\rho = \lambda E[S]$ we get the final equation.

$$p_k = (1-\rho) \times \rho^k$$

[46] PBD, page 295

Solution 11.3.11 Equation (11.3.11)

$$T = N/\lambda = \frac{E[S]}{1-\rho}$$

can be derived as follows. The GBD-theorem specifies the response time as being equal to

$$\text{response time } = T = \frac{\sum_{k=1}^{\infty} kP_k}{\sum_{k=1}^{\infty} \mu_k P_k} \tag{11.2.2}$$

Using the value $\lambda_i = \lambda$ and $\mu_i = \mu$, and using the result of the previous part of this exercise, we can rewrite equation 11.2.2 to

$$T = \frac{\sum_{k=1}^{\infty} k(1-\rho)\rho^k}{\sum_{k=1}^{\infty} \mu(1-\rho)\rho^k}$$

Removing $(1-\rho)$ from the sum and provided $\rho \neq 1$:

$$T = \frac{(1-\rho)\rho\sum_{k=1}^{\infty} k\rho^{k-1}}{(1-\rho)\mu\sum_{k=1}^{\infty} \rho^k} = \frac{\rho\sum_{k=1}^{\infty} k\rho^{k-1}}{\mu\sum_{k=1}^{\infty} \rho^k} \tag{11.2.3}$$

We know that $0 \leq \rho < 1$, so we can use appendix A.3 for the numerator:

$$\sum_{k=1}^{\infty} k\rho^{k-1} = \frac{1}{(1-\rho)^2}$$

Again using of appendix A.3 for the denominator:

$$\mu\sum_{k=1}^{\infty} \rho^k = \mu\frac{\rho}{1-\rho}$$

The equation 11.2.3 can now be rewriten as:

$$T = \frac{\rho\dfrac{1}{(1-\rho)^2}}{\mu\dfrac{\rho}{1-\rho}} = \frac{\rho}{(1-\rho)^2} \times \frac{1-\rho}{\mu\rho} = \frac{1}{\mu(1-\rho)}$$

Now use $\frac{1}{\mu} = E[S]$:

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

and the equation (11.3.11) of the book appears.

11.3 Exercise 11.3

Requested Derive the average waiting time for M/M/1 from Eq. (11.7.30) of the book[47].

[47] PBD, page 307

Solution The average waiting time $W_{M/M/c}$ uses a factor $C(\rho, c)$ which we will calculate first. Knowing $c = 1$ and filling that into equation (11.7.29) in the book we get

$$C(\rho, 1) = \frac{(1 \times \rho)^1/1!}{(1-\rho)\sum_{n=0}^{1-1}(1 \times \rho)^1/1! + (1 \times \rho)^1/1!}$$

$$C(\rho, 1) = \frac{\rho}{(1-\rho)\sum_{n=0}^{0}\rho^n/n! + \rho}$$

$$C(\rho, 1) = \frac{\rho}{(1-\rho) \times 1 + \rho} = \rho$$

Using this result and filling it into equation (11.7.30) from the book gives

$$W_{M/M/1} = \frac{\rho}{1 \times (1-\rho)/E[S]} = \frac{\rho E[S]}{1-\rho}$$

And the formula of (11.3.12) of page 295 of the book appears.

11.4 Exercise 11.4

Given Two clusters of webservers A and B. A has n servers and B has m servers ($m > n$). Both have an average arrival rate of λ for which no distribution is mentioned.
A load balancer in front of each cluster divides the arrivals evenly among the servers.
The average service time is equal to x for the A cluster and kx for the B cluster ($k > 1$). Because the symbol k is used, I will assume that $k \in N$. The service time in either cluster has an arbitrary distribution.

Requested Derive an expression for m such that the average response time of a request in cluster A is the same as in cluster B.

guess Anyone who has to guess the value of m, will probably guess $m = nk$. The reason is, that $\rho_B = \frac{\lambda kx}{m}$ and defining $m = nk$ makes ρ_B equal to $\rho_A = \frac{\lambda x}{n}$.

Solution Because the loadbalancer divides the load evenly among the servers, the system is effectively equal to a multiqueue, multiserver system, where each server has its own queue. The service time, when being serviced will remain equal to E[S]. But the service time of the customers in front of the waiting process, will effectively be equal to $\frac{E[S]}{n}$, given n servers. The response time T will then be equal to the service time $E[S]$ plus the effective service time of the customers in front of the waiting process:

$$T = E[S] + \frac{E[S]}{n}N \tag{11.4.1}$$

where N is the number of customers in the queue.
According to Little's Law, $N = XR$ or in the current notation $N = XT$. In a steady state system, $X = \lambda$, so this is $N = \lambda T$.
Thus the equation 11.4.1 becomes:

$$T = E[S] + \frac{\lambda E[S]}{n}T \Leftrightarrow$$

Because the utilization of the system is equal to the utilization of one server divided by n we get that $\frac{\lambda E[S]}{n} = \frac{U}{n} = \rho$ (see also equation(11.7.27) on page 306 of PBD), so the equation becomes:

$$T = \frac{E[S]}{1-\rho} \tag{11.4.2}$$

This is exactly the M/M/1 equation for response time[48]. The equation for a vanilla M/G/1 queue[49] shows the same equation as for M/M/1 plus a correctional factor of $\frac{1+C_s^2}{2}$ (see the solution for exercise 11.6, equation 11.6.1). As this factor depends on the coefficient of variance C_s^2 for the service times of A and B, there is no way to compensate for these factors. However, making the assumption, that both service time distributions will be alike, for instance because the machines are only different in speed, not in any other factors, we can use equation 11.4.2, because $T_A \times \frac{1+C_s^2}{2} = T_B \times \frac{1+C_s^2}{2} \Leftrightarrow T_A = T_B$. Using this for T_A and T_B we get:

$$T_A = \frac{E[S_A]}{1-\rho_A} = \frac{x}{1-\frac{\lambda x}{n}} = \frac{nx}{n-\lambda x}$$

$$T_B = \frac{E[S_B]}{1-\rho_B} = \frac{kx}{1-\frac{\lambda kx}{m}} = \frac{nkx}{m-\lambda kx}$$

Using $T_A = T_B$ we can derive the expression for m:

$$\begin{aligned}
\frac{nx}{n-\lambda x} &= \frac{nkx}{m-\lambda kx} \Leftrightarrow \\
(m-\lambda kx)nx &= (n-\lambda x)nkx \Leftrightarrow \\
mnx &= (n-\lambda x)nkx + \lambda knx^2 \Leftrightarrow \\
m &= \frac{(n-\lambda x)nkx + \lambda knx^2}{nx} \Leftrightarrow \\
m &= (n-\lambda x)k + \lambda kx \Leftrightarrow \\
m &= nk - \lambda kx + \lambda kx \Leftrightarrow \\
m &= nk
\end{aligned} \tag{11.4.3}$$

In conclusion, using the assumption that both distributions of service times for A and for B are alike, we can use the result $m = kn$ to get comparable response times in A and B.

$C_{s_A} \neq C_{s_B}$ In the case that the service time distribution of A and B are not alike, we have the following equation:

$$T_A C_A = T_B C_B \qquad \text{where } C_i = 1 + {C_{s_i}}^2$$

and where also is used

$$\frac{C_B}{C_A} = \frac{1+C_{s_B}^2}{1+C_{s_A}^2} = \frac{1+C_{s_A}^2 - C_{s_A}^2 + C_{s_B}^2}{1+C_{s_A}^2} = 1 + \frac{C_{s_B}^2 - C_{s_A}^2}{1+C_{s_A}^2}$$

[48] PBD, page 295, equation (11.3.10).
[49] PDB, page 296, equation (11.4.15)

$$\frac{nx}{n-\lambda x}C_A = \frac{nkx}{m-\lambda kx}C_B \Leftrightarrow$$
$$(m-\lambda kx)C_A nx = (n-\lambda x)nC_B kx \Leftrightarrow$$
$$mC_A nx = (n-\lambda x)nC_B kx + \lambda knC_A x^2 \Leftrightarrow$$
$$m = \frac{(n-\lambda x)nC_B kx + \lambda kC_A nx^2}{C_A nx} \Leftrightarrow$$
$$m = (n-\lambda x)\frac{C_B}{C_A}k + \lambda kx \Leftrightarrow$$
$$m = \frac{C_B}{C_A}nk - \lambda\frac{C_B}{C_A}kx + \lambda kx \Leftrightarrow$$
$$m = \frac{C_B}{C_A}nk + (1-\frac{C_B}{C_A})\lambda kx \Leftrightarrow$$

$$m = \frac{1+C_{s_B}^2}{1+C_{s_A}^2}nk + \frac{C_{s_B}^2 - C_{s_A}^2}{1+C_{s_A}^2}\lambda kx \tag{11.4.4}$$

Remark, that using $C_{S_A}^2 = C_{S_B}^2$ reduces equation 11.4.4 to equation 11.4.3.

11.5 Exercise 11.5

Given

- Arrival rate of 10 requests per second (Poisson process)
- 30% of type a and 70% of type b.
- Values given for service time

$$E[S_a] = 0.1 \text{ sec}$$
$$C_s^a = 1.5$$
$$E[S_b] = 0.08 \text{ sec}$$
$$C_s^b = 1.2$$

Requested Compute the average response time for each of the following cases:

- Request type a and b have the same priority
- Request type a has non-preemptive priority over b
- Request type b has non-preemptive priority over a
- Request type a has preemptive priority over b
- Request type b has preemptive priority over a

Solution The case describes a M/G/1 queue with the following formula's:

- **General case** formula's (11.4.14 - 11.4.17) on page 296 of PBD.
- **non-preemptive priority** formula's (11.6.20 - 11.6.23) on page 303 of PBD.
- **preemptive priority** formula's (11.6.24) on page 304 of PBD.

No priority The arrival rate of a is equal to $\lambda_a = 10 \times 0.3 = 3$ requests/second. The arrival rate of b is equal to $\lambda_b = 10 \times 0.7 = 7$ requests/second. The utilization of the server is equal to

$\lambda_a = 3$
$\lambda_b = 7$

$$\rho = \lambda_a \times E[S_a] + \lambda_b \times E[S_b] = 3 \times 0.1 + 7 \times 0.08 = 0.3 + 0.56 = 0.86$$

The waiting time W_a for process a is equal to

$\rho = 0.86$

$$W_a = \frac{\rho E[S_a](1 + (C_s^a)^2)}{2(1-\rho)} = \frac{0.86 \times 0.1 \times (1 + 1.5^2)}{2 \times 0.34} \approx 0.411 \text{ seconds}$$

The response time is the sum of $E[S]$ and W:

$$T_a = 0.1 + 0.411 = 0.511 \text{ seconds}$$

The waiting time W_b for process b is equal to

$$W_b = \frac{\rho E[S_b](1 + (C_s^b)^2)}{2(1-\rho)} = \frac{0.86 \times 0.08 \times (1 + 1.2^2)}{2 \times 0.34} \approx 0.247 \text{ seconds}$$

The response time is the sum of $E[S]$ and W:

$$T_b = 0.08 + 0.247 = 0.327 \text{ seconds}$$

Non-preemptive priority $a > b$

$$\begin{aligned}
\sigma_a = C_s^a.E[S_a] = 1.5 \times 0.1 = 0.15 \\
E[S_a^2] = \sigma_a^2 + (E[S_a])^2 = 0.0225 + 0.01 = 0.0325 \\
\sigma_b = C_s^b.E[S_b] = 1.2 \times 0.08 = 0.096 \\
E[S_b^2] = \sigma_b^2 + (E[S_b])^2 = 0.009216 + 0.0064 = 0.015616 \\
W_0 = \frac{1}{2}\sum_{j=1}^{P} \lambda_j E[S_j^2] = \frac{1}{2}(3 \times 0.0325 + 7 \times 0.015616) \approx 0.1034 \\
\pi_a = \lambda_a E[S_a] = 3 \times 0.1 = 0.3 \\
\pi_b = \pi_a + \lambda_b E[S_b] = 0.3 + 7 \times 0.08 = 0.86 \\
W_a = \frac{W_0}{1 - \pi_a} = \frac{0.1034}{0.7} \approx 0.148 \\
W_b = \frac{W_0}{(1-\pi_a)(1-\pi_b)} = \frac{0.1034}{0.7 \times 0.14} \approx 1.055 \text{ seconds} \\
T_a = E[S_a] + W_a \approx 0.1 + 0.148 = 0.248 \text{ seconds} \\
T_b = E[S_b] + W_b \approx 1.055 + 0.08 = 1.135 \text{ seconds}
\end{aligned}$$

Non-preemptive priority $b > a$ The values for σ_a and σ_b do not change and can be used for further calculations. The value of W_0 can be used for this

item too.

$$\pi_a = \lambda_a E[S_a] + \pi_b = 3 \times 0.1 + 7 \times 0.08 = 0.86$$
$$\pi_b = \lambda_b E[S_b] = 7 \times 0.08 = 0.56$$
$$W_a = \frac{W_0}{(1-\pi_a)(1-\pi_b)} = \frac{0.1034}{0.14 \times 0.44} \approx 1.679 \text{ seconds}$$
$$W_b = \frac{W_0}{1-\pi_b} = \frac{0.1034}{0.44} \approx 0.235 \text{ seconds}$$
$$T_a = W_a + E[S_a] \approx 1.679 + 0.1 = 1.779 \text{ seconds}$$
$$T_b = W_b + E[S_b] \approx 0.235 + 0.08 = 0.315 \text{ seconds}$$

Apparantly, the increase in performance by giving priority to transaction b is not nearly as significant as for transaction a. The probable cause is, that transaction a, once running, will still take the main portion of the available resources, because the service time of a is a lot more than that of b: 0.1 versus 0.08 seconds.

Preemptive priority $a > b$ The formula for calculating the response time is (11.6.24) on page 304 of PBD. This equation uses the values of $E[S_a^2] = 0.0325$ and $E[S_b^2] = 0.0156$ that we calculated earlier.

$$\begin{aligned} \pi_a &= \lambda_a E[S_a] = 3 \times 0.1 = 0.3 \\ T_a &= \frac{E[S_a](1-\pi_a) + \lambda_a E[S_a^2]/2}{(1-\pi_a)} \\ &= \frac{0.1 \times 0.7 + 3 \times 0.0325/2}{0.7} \approx 0.170 \end{aligned}$$

$$\begin{aligned} \pi_b &= \lambda_a E[S_a] + \lambda_b E[S_b] = 3 \times 0.1 + 7 \times 0.08 = 0.86 \\ T_b &= \frac{E[S_b](1-\pi_b) + \lambda_a E[S_a^2]/2 + \lambda_b E[S_b^2]/2}{(1-\pi_a)(1-\pi_b)} \\ &= \frac{0.08 \times 0.14 + 3 \times 0.0325/2 + 7 \times 0.0156/2}{0.7 \times 0.14} \approx 1.169 \end{aligned}$$

Preemptive priority $b > a$

$$\begin{aligned} \pi_a &= \lambda_a E[S_a] + \lambda_b E[S_b] = 3 \times 0.1 + 7 \times 0.08 = 0.86 \\ \pi_b &= \lambda_b E[S_b] = 7 \times 0.08 = 0.56 \\ T_a &= \frac{E[S_a](1-\pi_a) + \lambda_a E[S_a^2]/2 + \lambda_b E[S_b^2]/2}{(1-\pi_a)(1-\pi_b)} \\ &= \frac{0.1 \times 0.14 + 3 \times 0.0325/2 + 7 \times 0.0156/2}{0.14 \times 0.44} \approx 1.905 \text{ seconds} \end{aligned}$$

$$\begin{aligned} T_b &= \frac{E[S_b](1-\pi_b) + \lambda_b E[S_b^2]/2}{(1-\pi_b)} \\ &= \frac{0.08 \times 0.44 + 7 \times 0.0156/2}{0.44} \approx 0.204 \text{ seconds} \end{aligned}$$

This time the performance gain for priority b transactions is comparable to the gains of transaction a, when they are priority, although a is still a winner.

The following table shows the results per case. All Δ-values are with respect to the "no priority" case.

	transaction type a		**transaction type** b	
scenarios	T_a	Δ_a	T_b	Δ_b
No priority	0.511		0.327	
Non-preemptive priority $a > b$	0.248	-51%	1.135	+247%
Non-preemptive priority $b > a$	1.779	+248%	0.315	-4%
Preemptive priority $a > b$	0.170	-67%	1.169	+257%
Preemptive priority $b > a$	1.905	+272%	0.204	-38%

11.6 Exercise 11.6

Given Example 11.7 from §11.6.2 of PBD (page 304), specifically the statement that requests of the highest priority are not hindered by lower priority requests.

Requested Prove the above statement by computing the wait time for class 3 requests using vanilla M/G/1 results (§11.4).

Solution The vanilla M/G/1 queue has a wait time of

$$W_{M/G/1} = \frac{\rho E[S](1 + C_s^2)}{2(1 - \rho)}$$

Remark the structure of the formule, if one rewrites it as follows

$$W_{M/G/1} = \underbrace{\underbrace{\frac{\rho}{1-\rho}}_{N_{M/M/1}} \times E[S]}_{W_{M/M/1}} \underbrace{\times \frac{1 + C_s^2}{2}}_{\text{Correction } M/G/1} \tag{11.6.1}$$

In order to calculate $W_{M/G/1}$ we need ρ_3 and C_s^2 for class 3 transactions.

$$\sigma_3^2 = E[S_3^2] - (E[S_3])^2 = 0.180 - 0.09 = 0.172$$
$$\Rightarrow C_s^2 = \frac{0.172}{0.09} = 1.91$$
$$\rho_3 = \lambda_3 E[S_3] = 0.24 \times 0.3 = 0.072$$

$$W_{M/G/1} = \frac{0.072}{1 - 0.072} \times 0.3 \times \frac{1 + 1.911}{2} = 0.0338 \tag{11.6.2}$$

The response time $T = 0.3 + 0.0338 \approx 0.334$.
For the M/G/1 queue with preemptive priorities only the reponse time formula is given.

$$T_{\text{M/G/1-prio-preemptive}} = \frac{E[S_p](1 - \pi_p) + \sum_{i=p}^{P} \lambda_i E[S_i^2]/2}{(1 - \pi_p)(1 - \pi_{p+1})}$$

If p is the highest priority, then the sum (the second term in the numerator) reduces to $\lambda_p E[S_p^2]/2$. Also, $\pi_p = \lambda_p E[S_p] = \rho$ for the same reason. And finally,

the π_{p+1} does not exist as p is the highest priority. Thus the equation becomes

$$T_{\text{M/G/1-prio-preemptive}} = \frac{E[S_3](1-\rho_3) + \lambda_3 E[S_3^2]/2}{1-\rho_3}$$

$$T_{\text{M/G/1-prio-preemptive}} = E[S_3] + \underbrace{\frac{\lambda_3 E[S_3^2]}{1-\rho_3} \times \underbrace{\frac{1+0}{2}}_{Correction}}_{W_{\text{M/G/1 preempt}}}$$

$$T_{\text{M/G/1-prio-preemptive}} = 0.3 + \frac{0.24 \times 0.18}{1-0.072} \times \frac{1}{2}$$

$$T_{\text{M/G/1-prio-preemptive}} = 0.3 + 0.023 = 0.323$$

So the waiting time from the vanilla M/G/1 equation is equal to 0.0338, while the preemptive priority equation returns 0.023. The layout of both formula's show the differences between the formulas:

- In stead of using $E[S]$ in the waiting time[50], the priority queue uses $E[S^2]$, which is a lot smaller, because it is the second moment of processing time.
- In stead of using a correction of $\frac{1+C_s^2}{2}$, the priority queue uses a correction of $\frac{1}{2}$, which is as if $C_s = 0$, making the result even smaller.

In conclusion, it seems very likely, that the response time from T_3 in the priority queue is not decreased by the presence of the lower class of transactions.

I wonder why Using the values $E[S^2]$ and $C_s = 0$, makes you wonder, whether the priority case is not overly optimistic. After all, the highest priority runs a M/G/1 queue as if the machines are empty, so in fact the formula for the highest priority should resolve to the same formula as for a vanilla M/G/1 queue.
So I wonder why, the formula for the highest priority gives a faster response than the same transaction in an empty machine using M/G/1.

[50] the formula uses ρ, which is equal to $\lambda_3 E[S_3]$.

12 Chapter 12

12.1 Exercise 12.1

Given Balance equations in table 12.2 (PBD, page 319).

Requested

- Give details to verify the results in table 12.3 (PBD, page 319).
- Give details to verify the results in table 12.4 (PBD, page 320).

Solution part a The nodes of the Markov model are renumbered in figure 85.

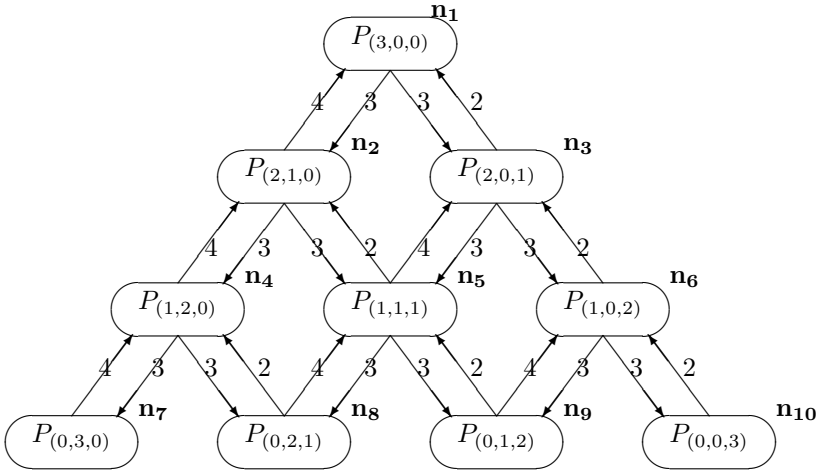


Figure 85: **Exercise 12.1** Markov diagram with 10 nodes ($n_1 \ldots n_{10}$) for 3 users.

Reworking the balance equations from table 12.2 of the book, so that the right side of the equation is equal to a constant (zero or one), while leaving out the equation for the last node (n_{10}) and replacing it by the last equation in table 12.2 $\sum_{i=1}^{10} p_i = 1$, we get the following set of equations

$$
\begin{array}{c}
\\ \mathbf{1} \\ \mathbf{2} \\ \mathbf{3} \\ \mathbf{4} \\ \mathbf{5} \\ \mathbf{6} \\ \mathbf{7} \\ \mathbf{8} \\ \mathbf{9} \\ \mathbf{10}
\end{array}
\begin{array}{c}
\begin{array}{cccccccccc}
\mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} & \mathbf{5} & \mathbf{6} & \mathbf{7} & \mathbf{8} & \mathbf{9} & \mathbf{10}
\end{array} \\
\begin{pmatrix}
-6 & 4 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & -10 & 0 & 4 & 2 & 0 & 0 & 0 & 0 & 0 \\
3 & 0 & -8 & 0 & 4 & 2 & 0 & 0 & 0 & 0 \\
0 & 3 & 0 & -10 & 0 & 0 & 4 & 2 & 0 & 0 \\
0 & 3 & 3 & 0 & -12 & 0 & 0 & 4 & 2 & 0 \\
0 & 0 & 3 & 0 & 0 & -8 & 0 & 0 & 4 & 2 \\
0 & 0 & 0 & 3 & 0 & 0 & -4 & 0 & 0 & 0 \\
0 & 0 & 0 & 3 & 3 & 0 & 0 & -6 & 0 & 0 \\
0 & 0 & 0 & 0 & 3 & 3 & 0 & 0 & -6 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{pmatrix}
\end{array}
\begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \\ p_7 \\ p_8 \\ p_9 \\ p_{10} \end{pmatrix}
=
\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}
\tag{12.1.1}
$$

This equation is solved using the package `R` (see figure 86).

```
> A <- read.table("ex12.1-matrix.txt")
> A
   X1  X2 X3  X4  X5 X6 X7 X8 X9 X10
1  -6   4  2   0   0  0  0  0  0   0
2   3 -10  0   4   2  0  0  0  0   0
3   3   0 -8   0   4  2  0  0  0   0
4   0   3  0 -10   0  0  4  2  0   0
5   0   3  3   0 -12  0  0  4  2   0
6   0   0  3   0   0 -8  0  0  4   2
7   0   0  0   3   0  0 -4  0  0   0
8   0   0  0   3   3  0  0 -6  0   0
9   0   0  0   0   3  3  0  0 -6   0
10  1   1  1   1   1  1  1  1  1   1
> b <- c(0,0,0,0,0,0,0,0,0,1)
> x<-solve(A,b)
> x
 [1] 0.07398844 0.05549133 0.11098266 0.04161850 0.08323699 0.16647399
 [7] 0.03121387 0.06242775 0.12485549 0.24971098
>
```

Figure 86: **Exercise 12.1**: Solving the set of equations from table 12.2 (PBD).

The resulting vector is thus equal to

$$
\begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \\ p_7 \\ p_8 \\ p_9 \\ p_{10} \end{pmatrix}
=
\begin{pmatrix} 0.07398844 \\ 0.05549133 \\ 0.11098266 \\ 0.04161850 \\ 0.08323699 \\ 0.16647399 \\ 0.03121387 \\ 0.06242775 \\ 0.12485549 \\ 0.24971098 \end{pmatrix}
$$

$\overline{\mathbf{p}}$	$\mathbf{p_i}$	**CPU**	**fd**	**sd**
p_1	0.0740	3	0	0
p_2	0.0555	2	1	0
p_3	0.1110	2	0	1
p_4	0.0416	1	2	0
p_5	0.0832	1	1	1
p_6	0.1665	1	0	2
p_7	0.0312	0	3	0
p_8	0.0624	0	2	1
p_9	0.1249	0	1	2
p_{10}	0.2497	0	0	3

Solution part b Using the above results we can calculate and check the requested performance metrics.

$$N_i = \sum_{i=1}^{K} n_{i,node} \times p_{i,node}$$

$$\begin{aligned} N_{cpu} &= 3 \times 0,0740 + 2 \times 0,0555 + 2 \times 0,1110 + 1 \times 0,0416 + 1 \times 0,0832 + 1 \times 0,1665 = 0.8463 \quad \surd \\ N_{fd} &= 1 \times 0,0555 + 2 \times 0,0416 + 1 \times 0,0832 + 3 \times 0,0312 + 2 \times 0,0624 + 1 \times 0,1249 = 0.5652 \quad \text{⁀} \\ N_{sd} &= 1 \times 0,1110 + 1 \times 0,0832 + 2 \times 0,1665 + 1 \times 0,0624 + 2 \times 0,1249 + 3 \times 0,2497 = 1.5885 \quad \surd \end{aligned}$$

$$U_i = \sum_{i=1}^{K} p_{i,node}$$

$$\begin{aligned} U_{cpu} &= 0.0740 + 0.0555 + 0.1110 + 0.0416 + 0.0832 + 0.1665 = 0.5318 \rightarrow 53.18\% \quad \surd \\ U_{fd} &= 0,0555 + 0,0416 + 0,0832 + 0,0312 + 0,0624 + 0,1249 = 0.3988 \rightarrow 39.88\% \quad \surd \\ U_{sd} &= 0,1110 + 0,0832 + 0,1665 + 0,0624 + 0,1249 + 0,2497 = 0.7977 \rightarrow 79.77\% \quad \surd \end{aligned}$$

$$X_i = \frac{U_i}{S_i}$$

$$\begin{aligned} X_{cpu} &= \frac{0.5318}{10} = 0.05318 \text{ tps} = 3.1908 \text{ tpm} \quad \surd \\ X_{fd} &= \frac{0.3988}{15} = 0.02658\not{0} \text{ tps} = 1.5952 \text{ tpm} \quad \text{⁀} \\ X_{sd} &= \frac{0.7977}{30} = 0.02659 \text{ tps} = 1.5954 \text{ tpm} \quad \surd \\ X_0 &= X_{cpu} = 0.05318 \text{ tps} = 3.1908 \text{ tpm} \quad \surd \end{aligned}$$

$$R_i = \frac{N_i}{X_0}$$

$$\begin{aligned} R_{cpu} &= \frac{0.8463}{0.05318} = 15.91 \text{ seconds} \quad \surd \\ R_{fd} &= \frac{0.5652}{0.05318} = 10.63 \text{ seconds} \quad \surd \\ R_{sd} &= \frac{1.5885}{0.05318} = 29.87 \text{ seconds} \quad \surd \end{aligned}$$

$$R = 15.91 + 10.63 + 29.87 = 56.41 \text{ seconds} \quad \surd$$

So, the results in the book are mainly correct.

12.2 Exercise 12.2

Given System description:

- A weightroom contains 1 treadmill (T), 1 stationary bike (B) and 1 rowing machine (R).

- There are always 8 students in the weightroom
- The students always take the route: $T \rightarrow B \rightarrow R$ for each cycle and a typical student stays for two cycles.
- $S_T = 5$ minutes, $S_B = 8$ minutes and $S_R = 4$ minutes.
- If a device is taken, the student waits until it is free.

Requested

1. Use MVA to determine the average number of students leaving the room per hour and the average amount of time each student stays in the weightroom.
2. Plot the average amount of time each student stays in the weightroom as a function of the number of students allowed in the weightroom.
3. If the maximum utilization is 80%, what is the maximum number of students that should be allowed in the weight room?
4. Which device is the system bottleneck?
5. If a percentage of students were allowed to bypass this device, what percentage should be allowed to bypass the device, so that it is no longer a bottleneck?
6. If one extra stationary bike were purchased, with how much would the average stay time be reduced?

Solution MVA Using the MVA algorithm as described in PBD (page 323), the model is solved for utilization, throughput, response time and number of students in the weight room. The calculations were performed using a spreadsheet `ex-12-2-mva.xls`. In figure 87, page 114, the results are printed.
The solution indicates that 0.0613 students leave the room per minute, or about 3.7 students per hour. Each student stays about 130.55 minutes or 2 hours and 10 minutes in the weight room. In figure 88 the response time versus number of students in the room is plotted.

Maintenance cap of 80% Assuming that each device should be used for a maximum of 80%, then no more than 2 students can use the weight room, as with 3 students the utilization of the bike is already 81%.

Bottleneck As figure 87 clearly shows, the bike is the system bottleneck.

Bypassing the bottleneck If a certain percentage of students were allowed to bypass the bike, then in order for the bike not to be the system bottleneck, this percentage must be 37.5%.
This follows from the following sequence of calculations:

$$\left.\begin{array}{r} U_i(n) = S_i \times X_i(n) \\ X_i(n) = V_i \times X_0(n) \end{array}\right\} \Rightarrow$$

$$\left.\begin{array}{c} U_i(n) = S_i \times V_i \times X_0(n) = D_i \times X_0(n) \\ X_0(n) = \frac{n}{R_0(n)} = \frac{n}{\sum_{i=1}^{K} R_i'(n)} = \frac{n}{\sum_{i=1}^{K} V_i \times S_i \times (1+\overline{n}_i(n-1))} \end{array}\right\} \Rightarrow$$

$$U_i(n) = \frac{n \times D_i}{\sum_{i=1}^{K} V_i \times S_i \times (1+\overline{n}_i(n-1))} = \frac{n \times D_i}{\sum_{i=1}^{K} D_i \times (1+\overline{n}_i(n-1))}$$

i	n	$R'_T(n)$	$R_0(n)$	$X_0(n)$	$X_i(n)$	$U_i(n)$	$\overline{n}_i(n)$
T	0						0.0000
B							0.0000
R							0.0000
T	1	10.00	34.00	0.0294	0.0588	0.2941	0.2941
B		16.00			0.0588	0.4706	0.4706
R		8.00			0.0588	0.2353	0.2353
T	2	12.94	46.35	0.0431	0.0863	0.4315	0.5584
B		23.53			0.0863	0.6904	1.0152
R		9.88			0.0863	0.3452	0.4264
T	3	15.58	59.24	0.0506	0.1013	0.5064	0.7892
B		32.24			0.1013	0.8103	1.6329
R		11.41			0.1013	0.4051	0.5779
T	4	17.89	72.64	0.0551	0.1101	0.5506	0.9852
B		42.13			0.1101	0.8810	2.3197
R		12.62			0.1101	0.4405	0.6951
T	5	19.85	86.53	0.0578	0.1156	0.5778	1.1472
B		53.11			0.1156	0.9246	3.0692
R		13.56			0.1156	0.4623	0.7836
T	6	21.47	100.85	0.0595	0.1190	0.5950	1.2775
B		65.11			0.1190	0.9519	3.8736
R		14.27			0.1190	0.4760	0.8489
T	7	22.77	115.54	0.0606	0.1212	0.6058	1.3798
B		77.98			0.1212	0.9693	4.7241
R		14.79			0.1212	0.4847	0.8961
T	8	23.80	130.55	0.0613	0.1226	0.6128	1.4583
B		91.59			0.1226	0.9804	5.6122
R		15.17			0.1226	0.4902	0.9295

Figure 87: **Exercise 12.2**: MVA solution, results from `ex-12-2-mva.xls`

Using the value of $n = 1$ we can calculate the basic values for the utilization per device

$$U_i(1) = \frac{1 \times D_i}{\sum_{i=1}^{K} D_i \times (1 + \overline{n}_i(0))} = \frac{1 \times D_i}{\sum_{i=1}^{K} D_i \times (1 + 0)}$$

$$U_i(1) = \frac{D_i}{D_T + D_B + D_R} \tag{12.2.1}$$

The highest utilized device is the system bottleneck. The current system bottleneck is the bike (B), which has a utilization of 47% for $n = 1$, and is higher than the utilization of the other devices for each n. In order for the bike not to be the system bottleneck any more, the utilization can not be any higher than the utilization for the treadmill (T) or the rowing machine (R) for any n. Starting at the utilization for $n = 1$ (i.e. single student utilization), this leads to the equation $U_B \leq max(U_T, U_R)$ or

$$\frac{D_B}{D_T + D_B + D_R} \leq \frac{max(D_T, D_R)}{D_T + D_B + D_R} \Leftrightarrow$$

$$D_B \leq max(D_T, D_R) \Leftrightarrow$$

$$D_B \leq 10 \tag{12.2.2}$$

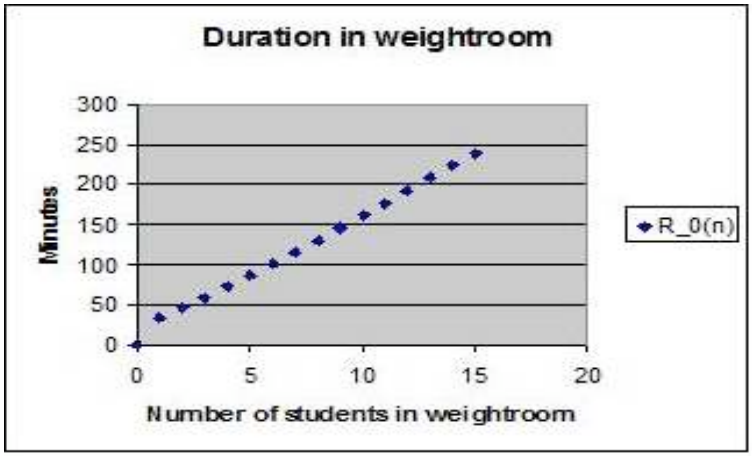


Figure 88: **Exercise 12.2**: Response time versus number of students in the room.

Using the equality of $D_i = V_i \times S_i$

$$V_B \times 8 \leq 10 \Leftrightarrow$$
$$V_B \leq \frac{5}{4}$$

This means a reduction of visits of 37.5% ($\frac{2-1.25}{2}$). So, if 37.5% of the students were allowed to bypass the bike, then the bike would not be the system bottleneck anymore. Using the spreadsheet `ex-12-2-mva-2.xls` you can check this reasoning by entering a p_i of 62.5%[51]. This decreases the visits to the bike from 2 to 1.25. The utilization for the bike will then be equal to the utilization of the treadmill (35.71%).

Extra stationary bike When using an extra stationary bike, the average time that a student spends in the weight room decreases from 130 to 95 minutes, a decrease of about 27% (see figure 89, page 123). The model used introduces a second bike and reduces the average visit to each bike to 1.00 in stead of 2.00. Thus, the new system now has 4 devices and is solved in the spreadsheet `ex-12-2-mva-3.xls`.

12.3 Exercise 12.3

Given The database example in §12.4 and balancing the system by upgrading the speed of the CPU and the slow disk.

Requested Answer the following questions:

- By how much would the speed of the CPU and slow disk need to be improved to achieve a balanced system?
- How much would these upgrades improve overall system performance (i.e. throughput and response time)?

[51] Remember to use a comma in the spreadsheet (Dutch format)

Solution To make utilization of CPU and slow disk equal to the utilization of the fast disk, we must decrease the service time to same amount, that it is for the fast disk. This way, utilization, which is equal to $X_0 \times D_i$[52] will be equal for all devices.
The service demand for the fast disk is equal to 7.5 seconds. The CPU has a service demand of 10 and the slow disk a service demand of 15. Therefore, the CPU must increase speed by 33.3%, making the service demand 7.5[53].
Analogously, the slow disk must increase speed by a factor 2 to get the service demand down from 15 to 7.5 milliseconds.
In order to find the overall improvement, the model must be solved with the old and the new values. Chapter 12 (page 324 — 326) contains the solved values.
The spreadsheet `ex-12-3-mva.xls` solves both the old and the new model. The results from the spreadsheet are (almost) identical to the results in the book[54].
The results are shown in figures 90 — 93 on pages 123—124.
As you can see, response time improves from 56.41 seconds to 37.50 or about 34% improvement. The throughput improves from 0.0531 tps to 0.0800 tps or about 51% improvement.

12.4 Exercise 12.4

Given The database example in §12.4.

Requested Use MVA to find the optimum proportion of files among the fast disk and the slow disk for 2 users.

Solution Using spreadsheet `ex-12-4-mva.xls` the proportion of files that should be stored on the fast disk to maximize the troughput, was found to be about 0.90 for 2 customers. This spreadsheet also shows the values requested for exercise 12.5.

12.5 Exercise 12.5

Given The database example in §12.4.

Requested

- Use MVA to find the optimum proportion of files among the fast disk and the slow disk for 1, 2, 3, 5, 10 and 15 users. Plot the results.
- Provide a hypothesis why the optimal proportion changes as a function of the number of customers in the system.
- Provide a hypothesis of what the optimal proportion would be if the number of customers in the system grows toward infinity. Justify your hypothesis.

[52] Service Demand Law, PBD, chapter 3
[53] D_{cpu} goes down from 10 to 7.5, forcing the speed to increase by $\frac{10}{7.5}$
[54] There are a few miscalculations in the book, like $\overline{n}_{fd}$ and $\overline{n}_{sd}$

Solution MVA Using spreadsheet `ex-12-4-mva.xls` varying the proportion of "fd" from 0.0 to 1.0 by 0.1, shows all values in a table (see figure 94 on page 117). The plots are in figures 95 and 96 on pages 125 and 126.

n	0.0000	0.1000	0.2000	0.3000	0.4000	0.5000	0.6000	0.7000	0.8000	0.9000	1,0000
1	0.0250	0.0260	0.0270	0.0282	0.0294	0.0308	0.0323	0.0339	0.0357	0.0377	**0.0400**
2	0.0308	0.0333	0.0360	0.0390	0.0421	0.0452	0.0482	0.0508	0.0526	**0.0533**	0.0526
3	0.0325	0.0357	0.0394	0.0436	0.0483	0.0532	0.0576	0.0609	**0.0621**	0.0612	0.0585
5	0.0332	0.0369	0.0413	0.0467	0.0534	0.0609	0.0681	**0.0724**	0.0720	0.0682	0.0635
10	0.0333	0.0370	0.0417	0.0476	0.0554	0.0659	0.0780	**0.0840**	0.0800	0.0730	0.0663
15	0.0333	0.0370	0.0417	0.0476	0.0555	0.0666	0.0812	**0.0886**	0.0821	0.0738	0.0666

Figure 94: **Exercise 12.5**: the resulting table from spreadsheet `ex-12-4-mva.xls`

Optimal proportion A hypothesis why the optimal proportion of files changes as a function of the number of customers in the system, is that the amount of queuing determines the optimal proportion of files.
If only one customer is in the system, then the service time will be the shortest possible if all files are on the fast disk. As more customers join the system, more queuing will result and having all files on the fast disk is now suboptimal, because, in stead of waiting on the fast disk, some of the customers could use the slow disk.
In the extreme case of a number of customers tending towards infinity, the throughput will be highest if the average service demand for each disk is equal, i.e. if the disks are balanced. Therefore, the optimal case is when $\frac{2}{3}^{rd}$ of the files are stored on the fast disk and $\frac{1}{3}^{rd}$ is stored on the slow disk[55].
This makes the average service demand for each disk equal to 10 seconds on average. This also explains, why in the figures and the graph (figures 94 and 95), the line tends towards $\frac{2}{3}^{rd}$ as the number of customers increase.

12.6 Exercise 12.6

Given

- One CPU and one disk
- T = 3600 seconds, $U_{\text{CPU}} = 30\%$, $C_0 = 10800$ http requests.
- $V_{\text{disk}} = 3$ IO's, $S_{\text{disk}} = 20$ msec.

[55] See §12.4 of PBD, page 328

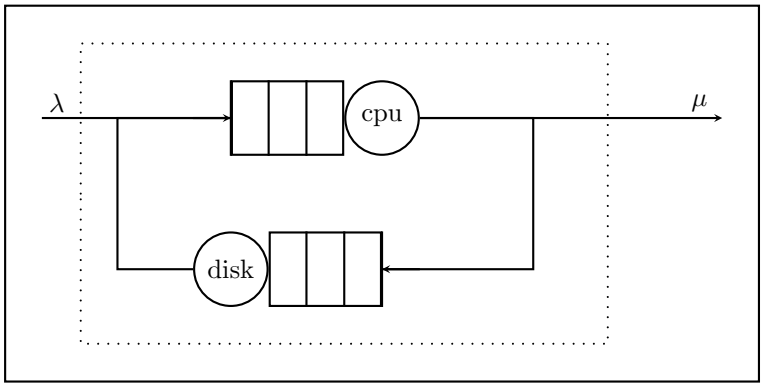


Figure 97: **Exercise 12.6**: Model for one cpu, one disk

Requested

- What are the service demands for CPU and disk?
- Find $X_0(n)$ where n is the number of concurrently executing http-requests, and $n = 0, 1, 2, 3$.
- Assume $\lambda = 5$ requests per second. Assume only 3 concurrent http requests can be processed. Assume the waiting queue for processing is infinite. Requested is the average response time for an http-request for the complete system, including the infinite waiting queue. [Hint: use the GBD-theorem and the results from the previous item]

Solution service demands

$$D_{\text{disk}} = V_{\text{disk}} \times S_{\text{disk}} = 3 \times 20 = 60 \text{ msec} \quad (12.6.1)$$

$$D_{\text{CPU}} = \frac{U_{\text{CPU}}}{X_0} = \frac{U_{\text{CPU}}}{\frac{C_0}{T}} = \frac{0.3}{\frac{10800}{3600}} = \frac{0.3}{3} = 0.1 \text{ seconds} = 100 \text{ msec} \quad (12.6.2)$$

Solution X_0 as a function of n In order to find the throughput as a function of n, the MVA algorithm should be used. The spreadsheet `ex-12-6-mva.xls` contains the MVA solution, the result is copied into figure 98, which shows the throughput for $n = 0, 1, 2, 3$. For $n = 3$ the throughput is equal to 0.009 request/msec or 9 requests/second.

i	**n**	$\mathbf{R'_T(n)}$	$\mathbf{R_0(n)}$	$\mathbf{X_0(n)}$	$\mathbf{X_i(n)}$	$\mathbf{U_i(n)}$	$\mathbf{\overline{n}_i(n)}$
CPU	0						0.0000
disk							0.0000
CPU	1	100.00	160.00	0.0063	0.0063	0.6250	0.6250
disk	1	60.00			0.0188	0.3750	0.3750
CPU	2	162.50	245.00	0.0082	0.0082	0.8163	1.3265
disk	2	82.50			0.0245	0.4898	0.6735
CPU	3	232.65	333.06	0.0090	0.0090	0.9007	2.0956
disk	3	100.41			0.0270	0.5404	0.9044

Figure 98: **Exercise 12.6**: The solved model for the 3 concurrent http requests.

Solution new system The system described, uses the above original system as a black box. So, on a higher level, the described system can be seen as a single load independent server with an infinite waiting queue as in figure 99.

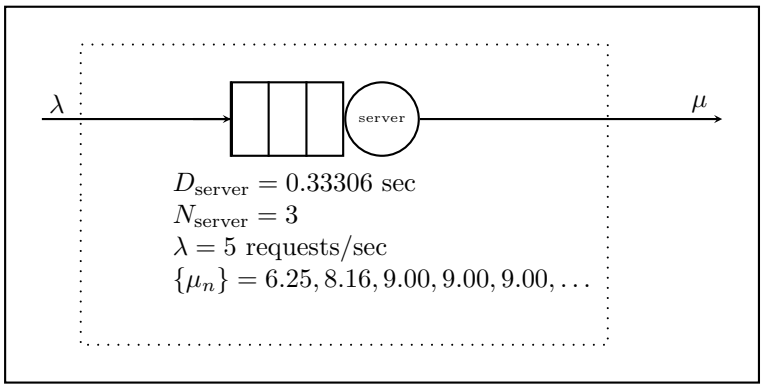


Figure 99: **Exercise 12.6**: A single server. The server contains the original system.

The situation can be modeled as a Markov chain, with the number of customers (concurrent http-requests + waiting line) describing the states as in figure 100. In this model, the λ-values are not state dependant, i.e. $\lambda_i = \lambda \quad \forall i \in N$. The μ-values are slightly state dependant: $\mu_1 \approx 6.25$, $\mu_2 \approx 8.16$, $\mu_3 \approx 9.00$ and $\mu_i = \mu_3 \quad (\forall i \in \{4, 5, \ldots\})$.

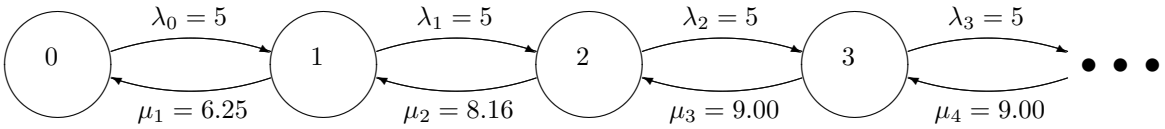


Figure 100: **Exercise 12.6**: The Markov chain for the system with an infinite waiting line.

The GBD theorem[56] gives the response time as:

$$\text{response time} = \frac{\text{queue length}}{\text{throughput}} = \frac{\sum_{k=1}^{\infty} kP_k}{\sum_{k=1}^{\infty} \mu_k P_k} \tag{12.6.3}$$

with

$$P_k = P_0 \prod_{i=0}^{k-1} \lambda_i/\mu_{i+1} \text{ for } k = 0, 1, 2, \ldots \tag{12.6.4}$$

and

$$P_0 = \left[\sum_{k=0}^{\infty} \prod_{i=0}^{k-1} \lambda_i/\mu_{i+1} \right]^{-1} \tag{12.6.5}$$

In order to calculate the response time, we need to solve the infinite sum of terms of both the numerator and the denominator of equation 12.6.3. Because each

[56]PBD, pages 279,280

term in both the numerator and the denominator contains P_k and thus P_0, we can eliminate P_0 from the equation:

$$R_0 = \frac{\sum_{k=1}^{\infty} k \prod_{i=0}^{k-1} \lambda_i/\mu_{i+1}}{\sum_{k=1}^{\infty} \mu_k \prod_{i=0}^{k-1} \lambda_i/\mu_{i+1}} \tag{12.6.6}$$

We start out solving the numerator

$$\begin{aligned}
R_0^{num} &= \textstyle\sum_{k=1}^{\infty} k \prod_{i=0}^{k-1} \lambda_i/\mu_{i+1} \Leftrightarrow \\
R_0^{num} &= 1 \times \tfrac{\lambda}{\mu_1} + 2 \times \tfrac{\lambda^2}{\mu_1\mu_2} + 3 \times \tfrac{\lambda^3}{\mu_1\mu_2\mu_3} + 4 \times \tfrac{\lambda^4}{\mu_1\mu_2\mu_3^2} + \cdots + k \times \tfrac{\lambda^k}{\mu_1\mu_2\mu_3^{k-2}} + \cdots \\
R_0^{num} &= \tfrac{\lambda}{\mu_1} + \tfrac{\lambda\mu_3}{\mu_2\mu_1}\left(2 \times \tfrac{\lambda}{\mu_3} + 3 \times \tfrac{\lambda^2}{\mu_3^2} + 4 \times \tfrac{\lambda^3}{\mu_3^3} + \cdots + k\tfrac{\lambda^{k-1}}{\mu_3^{k-1}} + \cdots\right) \\
R_0^{num} &= \tfrac{\lambda}{\mu_1} + \tfrac{\lambda\mu_3}{\mu_2\mu_1}\left(2q^1 + 3q^2 + 4q^3 + \cdots + kq^{k-1} + \cdots\right) \qquad (\text{where } q = \tfrac{\lambda}{\mu_3})
\end{aligned}$$

Now add $1 \times q^0$ to the infinite sum inside the product and subtract it outside the product

$$\begin{aligned}
R_0^{num} &= \tfrac{\lambda}{\mu_1} + \tfrac{\lambda\mu_3}{\mu_1\mu_2}\left(-1 + 1q^0 + 2q^1 + 3q^2 + 4q^3 + \cdots + kq^{k-1} + \cdots\right) \\
R_0^{num} &= \tfrac{\lambda}{\mu_1} + \tfrac{\lambda\mu_3}{\mu_1\mu_2}\left(\textstyle\sum_{k=1}^{\infty} kq^{k-1} - 1\right) \\
&\text{Using appendix A.2} \\
R_0^{num} &= \tfrac{\lambda}{\mu_1} + \tfrac{\lambda\mu_3}{\mu_1\mu_2}\left(\frac{1}{(1-q)^2} - 1\right)
\end{aligned}$$

$$R_0^{num} = \frac{\lambda}{\mu_1}\left(1 + \frac{\mu_3}{\mu_2}\left(\frac{1}{(1-\frac{\lambda}{\mu_3})^2} - 1\right)\right) \tag{12.6.7}$$

$$R_0^{num} \approx 4.3769 \tag{12.6.8}$$

The next step is solving the denominator

$$R_0^{den} = \sum_{k=1}^{\infty} \mu_k \prod_{i=0}^{k-1} \lambda_i/\mu_{i+1} \Leftrightarrow$$

$$\begin{aligned}
R_0^{den} &= \mu_1\frac{\lambda}{\mu_1} + \mu_2\frac{\lambda^2}{\mu_1\mu_2} + \mu_3\frac{\lambda^3}{\mu_1\mu_2\mu_3} + \cdots + \mu_3\frac{\lambda^k}{\mu_1\mu_2\mu_3^{k-2}} + \cdots \\
R_0^{den} &= \lambda + \frac{\lambda^2}{\mu_1} + \frac{\lambda^3}{\mu_1\mu_2} + \frac{\lambda^4}{\mu_1\mu_2\mu_3} + \cdots + \frac{\lambda^k}{\mu_1\mu_2\mu_3^{k-3}} + \cdots \\
R_0^{den} &= \lambda + \frac{\lambda^2}{\mu_1} + \frac{1}{\mu_1\mu_2}\left(\lambda^3 + \frac{\lambda^4}{\mu_3} + \cdots + \frac{\lambda^k}{\mu_3^{k-3}} + \cdots\right) \\
R_0^{den} &= \lambda + \frac{\lambda^2}{\mu_1} + \frac{\mu_3^3}{\mu_1\mu_2}\left(\frac{\lambda^3}{\mu_3^3} + \frac{\lambda^4}{\mu_3^4} + \cdots + \frac{\lambda^k}{\mu_3^k} + \cdots\right)
\end{aligned}$$

Now add the first three missing terms to the sum within the parentheses and subtract the same amounts. which we keep within parentheses

$$R_0^{den} = \lambda + \frac{\lambda^2}{\mu_1} \underbrace{- \frac{\lambda}{\mu_3}\frac{\mu_3^3}{\mu_1\mu_2} - \frac{\lambda^2}{\mu_3^2}\frac{\mu_3^3}{\mu_1\mu_2}}_{added} + \frac{\mu_3^3}{\mu_1\mu_2}\left(\overbrace{\underbrace{-1 + 1 + \frac{\lambda}{\mu_3} + \frac{\lambda^2}{\mu_3^2}}_{added} + \frac{\lambda^3}{\mu_3^3} + \cdots + \frac{\lambda^k}{\mu_3^k} + \cdots}^{-1+(\sum_{k=0}^{\infty} q^k)}\right)$$

$$R_0^{den} = \lambda + \frac{\lambda^2}{\mu_1} - \frac{\lambda\mu_3^2}{\mu_1\mu_2} - \frac{\lambda^2\mu_3}{\mu_1\mu_2} + \frac{\mu_3^3}{\mu_1\mu_2}\left(\sum_{k=0}^{\infty} q^k - 1\right) \qquad (\text{where } q = \frac{\lambda}{\mu_3})$$

Using appendix A.2

$$R_0^{den} = \lambda + \frac{\lambda^2}{\mu_1} - \frac{\lambda\mu_3^2}{\mu_1\mu_2} - \frac{\lambda^2\mu_3}{\mu_1\mu_2} + \frac{\mu_3^3}{\mu_1\mu_2}\left(\frac{1}{1-q} - 1\right)$$

$$R_0^{den} = \lambda + \frac{\lambda}{\mu_1}\left(\lambda - \frac{\mu_3}{\mu_2}(\mu_3 - \lambda)\right) + \frac{\mu_3^3}{\mu_1\mu_2}\left(\frac{1}{1-\frac{\lambda}{\mu_3}} - 1\right)$$

$$R_0^{den} = \lambda + \frac{\lambda}{\mu_1}\left(\lambda - \frac{\mu_3}{\mu_2}(\mu_3 - \lambda)\right) + \frac{\mu_3^3}{\mu_1\mu_2}\left(\frac{\mu_3}{\mu_3 - \lambda} - 1\right)$$

$$R_0^{den} = \lambda + \frac{\lambda}{\mu_1}\left(\lambda - \frac{\mu_3}{\mu_2}(\mu_3 - \lambda)\right) + \frac{\mu_3^3}{\mu_1\mu_2} \times \frac{\lambda}{\mu_3 - \lambda} \tag{12.6.9}$$

$$R_0^{den} \approx 23.3341 \tag{12.6.10}$$

Using the results from equations 12.6.8 and 12.6.10, the average response time for the http-requests is

$$R_0 = \frac{R_0^{num}}{R_0^{den}} \approx \frac{4.3769}{23.3341} \approx 0.1876 \text{ seconds} \tag{12.6.11}$$

$\boxed{R_0 = 0.1876 \text{ seconds}}$

The average response time is 0.1876 seconds per http-request. How probable is this? Lets have a look at the utilization. For this we need the P_0 factor (see equation 12.6.5).

$$P_0^{-1} = \sum_{k=0}^{\infty}\prod_{i=0}^{k-1}\lambda_i/\mu_{i+1} \tag{12.6.12}$$

$$p_0^{-1} = 1 + \frac{\lambda}{\mu_1} + \frac{\lambda^2}{\mu_1\mu_2} + \frac{\lambda^3}{\mu_1\mu_2\mu_3} + \frac{\lambda^4}{\mu_1\mu_2\mu_3^2} + \cdots + \frac{\lambda^k}{\mu_1\mu_2\mu_3^{k-2}} + \cdots +$$

Separate terms, so that all terms in the infinite sum have at least μ_1 and μ_2 in the denominator and λ^2 in the numerator

$$p_0^{-1} = 1 + \frac{\lambda}{\mu_1} + \frac{\lambda^2}{\mu_1\mu_2}\left(1 + \frac{\lambda}{\mu_3} + \frac{\lambda^2}{\mu_3^2} + \cdots + \frac{\lambda^k}{\mu_3^k} + \cdots\right)$$

$$p_0^{-1} = 1 + \frac{\lambda}{\mu_1} + \frac{\lambda^2}{\mu_1\mu_2} \times \sum_{k=0}^{\infty}\left(\frac{\lambda}{\mu_3}\right)^k$$

Now use appendix A.3.

$$p_0^{-1} = 1 + \frac{\lambda}{\mu_1} + \frac{\lambda^2}{\mu_1\mu_2} \times \frac{1}{1-\frac{\lambda}{\mu_3}}$$

$$p_0^{-1} = 1 + \frac{\lambda}{\mu_1} + \frac{\lambda^2}{\mu_1\mu_2} \times \frac{\mu_3}{\mu_3 - \lambda}$$

The results of the calculations are summarized in figure 101.

variable	value
μ_1	6.2500
μ_2	8.1633
μ_3	9.0074
λ	5.0000

(a) MVA output

i	P_i	$\sum_i(P_i)$
1	0.2757	0.2757
2	0.1689	0.4446
3	0.0937	0.5384
4	0.0520	0.5904
5	0.0289	0.6193
6	0.0160	0.6353
7	0.0089	0.6442
8	0.0049	0.6492
9	0.0027	0.6519
10	0.0015	0.6534
11	0.0008	0.6543
12	0.0005	0.6548
13	0.0003	0.6550
14	0.0001	0.6552
15	0.0001	0.6552
16	0.0000	0.6553
17	0.0000	0.6553

(b) probabilities

variable	**value**
Queuelength	4,3769 txns
Throughput	23,3341 txns/sec
Responsetime	0,1876 sec/txn
Utilization	65,53%
$\mathbf{1/p_0}$	2.9014
$\mathbf{p_0}$	0.3447

(c) Markov output

Figure 101: The intput and output variables to calculate the Markov chain probabilities, (a) Basic variables (output from MVA), (b) probabilities P_i for $i \in \{1, 2, \ldots, 17\}$ (c) Calculated variables from Markov chain

i	n	$R'_T(n)$	$R_0(n)$	$X_0(n)$	$X_i(n)$	$U_i(n)$	$\overline{n}_i(n)$
T	0						0.0000
B0							0.0000
B1							0.0000
R							0.0000
T	1	10.00	34.00	0.0294	0.0588	0.2941	0.2941
B0		8.00			0.0294	0.2353	0.2353
B1		8.00			0.0294	0.2353	0.2353
R		8.00			0.0588	0.2353	0.2353
T	2	12.94	42.59	0.0470	0.0939	0.4696	0.6077
B0		9.88			0.0470	0.3757	0.4641
B1		9.88			0.0470	0.3757	0.4641
R		9.88			0.0939	0.3757	0.4641
T	3	16.08	51.22	0.0586	0.1172	0.5858	0.9417
B0		11.71			0.0586	0.4686	0.6861
B1		11.71			0.0586	0.4686	0.6861
R		11.71			0.1172	0.4686	0.6861
T	4	19.42	59.88	0.0668	0.1336	0.6680	1.2970
B0		13.49			0.0668	0.5344	0.9010
B1		13.49			0.0668	0.5344	0.9010
R		13.49			0.1336	0.5344	0.9010
T	5	22.97	68.59	0.0729	0.1458	0.7289	1.6744
B0		15.21			0.0729	0.5831	1.1085
B1		15.21			0.0729	0.5831	1.1085
R		15.21			0.1458	0.5831	1.1085
T	6	26.74	77.35	0.0776	0.1551	0.7757	2.0745
B0		16.87			0.0776	0.6206	1.3085
B1		16.87			0.0776	0.6206	1.3085
R		16.87			0.1551	0.6206	1.3085
T	7	30.75	86.15	0.0813	0.1625	0.8125	2.4982
B0		18.47			0.0813	0.6500	1.5006
B1		18.47			0.0813	0.6500	1.5006
R		18.47			0.1625	0.6500	1.5006
T	8	34.98	95.00	0.0842	0.1684	0.8421	2.9460
B0		20.00			0.0842	0.6737	1.6847
B1		20.00			0.0842	0.6737	1.6847
R		20.00			0.1684	0.6737	1.6847

Figure 89: **Exercise 12.2**: MVA solution for model with extra bike, results from `ex-12-2-mva-3.xls`

	V_i	S_i	D_i
CPU	1.00	10.00	10.0000
fd	0.50	15.00	7.5000
sd	0.50	30.00	15.0000

Figure 90: **Exercise 12.3**: The original input parameters for the database model.

	V_i	S_i	D_i
CPU	1.00	7.50	7.5000
fd	0.50	15.00	7.5000
sd	0.50	15.00	7.5000

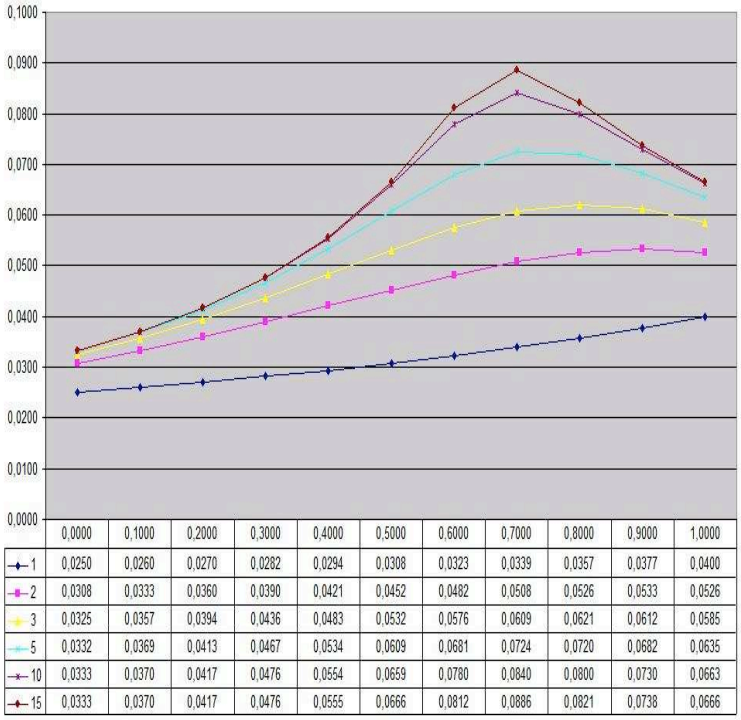
Figure 91: **Exercise 12.3**: The input parameters with upgraded CPU and slow disk for the balanced system.

i	**n**	$\mathbf{R'_T(n)}$	$\mathbf{R_0(n)}$	$\mathbf{X_0(n)}$	$\mathbf{X_i(n)}$	$\mathbf{U_i(n)}$	$\mathbf{\overline{n}_i(n)}$
CPU	0						0.0000
fd							0.0000
sd							0.0000
CPU	1	10.00	32.50	0.0308	0.0308	0.3077	0.3077
fd		7.50			0.0154	0.2308	0.2308
sd		15.00			0.0154	0.4615	0.4615
CPU	2	13.08	44.23	0.0452	0.0452	0.4522	0.5913
fd		9.23			0.0226	0.3391	0.4174
sd		21.92			0.0226	0.6783	0.9913
CPU	3	15.91	56.41	0.0532	0.0532	0.5318	0.8462
fd		10.63			0.0266	0.3988	0.5653
sd		29.87			0.0266	0.7977	1.5884

Figure 92: **Exercise 12.3**: The solved model for the original system.

i	**n**	$\mathbf{R'_T(n)}$	$\mathbf{R_0(n)}$	$\mathbf{X_0(n)}$	$\mathbf{X_i(n)}$	$\mathbf{U_i(n)}$	$\mathbf{\overline{n}_i(n)}$
CPU	0						0.0000
fd							0.0000
sd							0.0000
CPU	1	7,50	22,50	0,0444	0,0444	0,3333	0,3333
fd		7,50			0,0222	0,3333	0,3333
sd		7,50			0,0222	0,3333	0,3333
CPU	2	10,00	30,00	0,0667	0,0667	0,5000	0,6667
fd		10,00			0,0333	0,5000	0,6667
sd		10,00			0,0333	0,5000	0,6667
CPU	3	12,50	37,50	0,0800	0,0800	0,6000	1,0000
fd		12,50			0,0400	0,6000	1,0000
sd		12,50			0,0400	0,6000	1,0000

Figure 93: **Exercise 12.3**: The solved model for the balanced system.



	0,0000	0,1000	0,2000	0,3000	0,4000	0,5000	0,6000	0,7000	0,8000	0,9000	1,0000
1	0,0250	0,0260	0,0270	0,0282	0,0294	0,0308	0,0323	0,0339	0,0357	0,0377	0,0400
2	0,0308	0,0333	0,0360	0,0390	0,0421	0,0452	0,0482	0,0508	0,0526	0,0533	0,0526
3	0,0325	0,0357	0,0394	0,0436	0,0483	0,0532	0,0576	0,0609	0,0621	0,0612	0,0585
5	0,0332	0,0369	0,0413	0,0467	0,0534	0,0609	0,0681	0,0724	0,0720	0,0682	0,0635
10	0,0333	0,0370	0,0417	0,0476	0,0554	0,0659	0,0780	0,0840	0,0800	0,0730	0,0663
15	0,0333	0,0370	0,0417	0,0476	0,0555	0,0666	0,0812	0,0886	0,0821	0,0738	0,0666

Figure 95: **Exercise 12.5**: Results of varying proportion of files on the fast disk from 0.0 (no files) to 1.0 (all files)

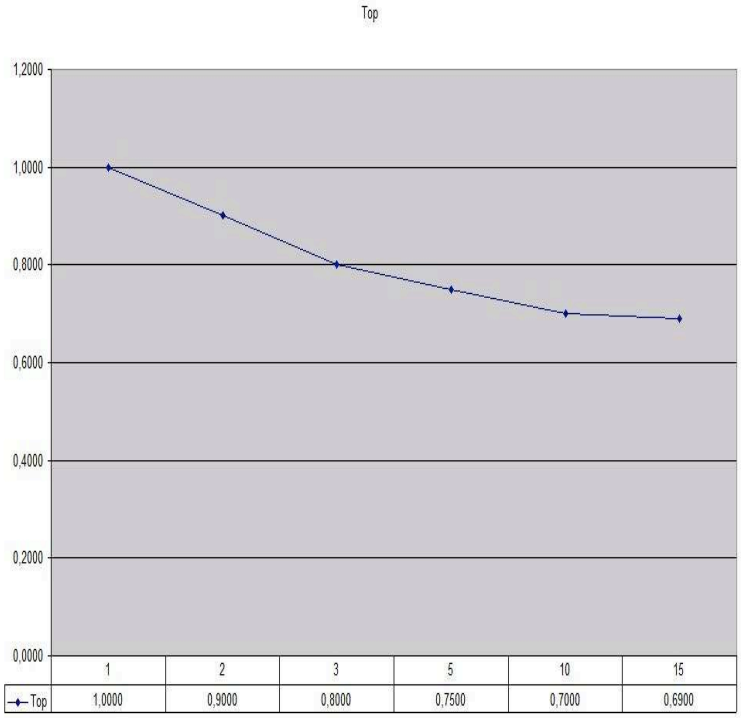


Figure 96: **Exercise 12.5**: The optimal proportion of files on the fast disk as a function of number of customers

13 Chapter 13

13.1 Exercise 13.1

Given The system specified in PBD, table 13.1 (page 343).

Requested

1. Solve the performance model using Bard's approximation
2. Compare the results to the results obtained by Schweitzer's approximation
3. Comments?

Solution Firstly, remark that the BCMP assumption that the number of class r customers at each service centre increases proportionally to the number of class r customers in the network, apply to both query and update transactions. For query transactions the second disk is not included in this proportion.
To use Bard's approximation, one can use the algorithm described on PBD, page 362 in Figure 13.5, replacing the approximation for $t = r$ with Bard's approximation.
The solution is calculated in spreadsheet `Ex13.1-Bard`. The results are summarized in figure 102.

	Throughput		Response Time	
Method	Query	Update	Query	Update
Exact MVA	4.093	0.409	0.733	2.445
Bard	3.879	0.669	0.773	1.494
Relative error (|%|	5.23	63.67	5.52	38.88
Schweitzer	3.995	0.407	0.751	2.457
Bard	3.879	0.669	0.773	1.494
Relative error (|%|	3.05	64.47	3.26	39.18

Figure 102: **Exercise 13.1** Comparison of Bard's approximation to exact and Schweitzer's approximation.

The results show a big error for the update transactions. The error in query transactions is visible in the response times per device. On the level of the global response time the error is virtually canceled out.

13.1.1 Comments

The approximated service demands according to the LCP algorithm[57] contain large errors. The basic assumption of Bard's approximation is that with large number there is little difference between $\bar{n}_{i,r}(\vec{N})$ and $\bar{n}_{i,r}(\vec{N} - \vec{(1)}_r)$.
The current case is one with small numbers, voiding the assumption and apparantly creating larger errors.

13.2 Exercise 13.2

Given The transaction system of PBD, §13.3.

[57] Bard's Large Customer Population algorithm

Requested

- Calculate the model using the approximate MVA algorithm (Schweitzer's approximation) with a maximum difference of 0.001 for successive values of $n_{i,r}$.
- Suppose that the number of update transactions in the system is tripled:
  - Recalculate the model's result with the approximate MVA
  - Recalculate the model's result with the exact MVA
  - Compare the computational effort required by the two algorithms

Solution The approximate MVA calculations for (1,3) are shown in figure 103. The approximate and exact MVA calculations for (3, 3) are shown in the figures 104 and resp. 105.
All calculations are part of the spreadsheet `Ex13.2-Schweitzer-1`.

	Query		Update		Query Queue length (calc.)		Update Queue length (calc.)			Maximum
	RT		RT							
iter	Total	TPUT	Total	TPUT	CPU	Disk 1	CPU	Disk 1	Disk 2	Difference
1					1.500	1.500	0.333	0.333	0.333	
2	0.665	4.511	2.378	0.421	1.105	1.895	0.395	0.505	0.101	2.297
3	0.722	4.155	2.419	0.413	0.931	2.069	0.326	0.574	0.099	0.212
4	0.736	4.076	2.437	0.410	0.832	2.168	0.297	0.604	0.098	0.098
5	0.743	4.038	2.448	0.408	0.783	2.217	0.280	0.621	0.098	0.061
6	0.747	4.016	2.453	0.408	0.759	2.241	0.273	0.630	0.098	0.026
7	0.749	4.005	2.456	0.407	0.749	2.251	0.269	0.633	0.098	0.015
8	0.750	4.000	2.456	0.407	0.744	2.256	0.267	0.635	0.098	0.007
9	0.750	4.000	2.457	0.407	0.740	2.260	0.266	0.636	0.098	0.004
10	0.751	3.995	2.458	0.407	0.739	2.261	0.266	0.637	0.098	0.002
11	0.751	3.995	2.457	0.407	0.739	2.261	0.265	0.637	0.098	0.004
12	0.751	3.995	2.457	0.407	0.739	2.261	0.265	0.637	0.098	

Figure 103: **Exercise 13.2** Approximate MVA for (1,3) configuration.

Configuration $(3, 1)$ The approximated response times for query and update are 0.751 and 2.457 versus 0.733 and 2.444 for the exact algorithm (PBD, table 13.3). The throughput for query and update are 3.995 and 0.407 versus 4.093 and 0.409 for the exact algorithm.

Configuration $(3, 3)$ The response times for query and update are 1.054 and 3.338 as calculated using the approximated MVA algorithm versus 1.021 and 3.310 as calculated using the exact MVA algorithm. The errors are respectively 3.2% and 0.9%.
The throughput of query and update are 2.846 and 0.899 as calculated by the approximated MVA algorithm and 2.938 and 0.906. The errors are respectively 3.1% and 0.8%.

iter	Query Queue length (calculated) CPU	Disk 1	Disk 2	Update Queue length (calculated) CPU	Disk 1	Disk 2	Max Diff	RT Total	TPUT	RT Total	TPUT
1	1.500	1.500		1.000	1.000	1.000					
2	1.105	1.895		1.146	1.467	0.386	1.591	0.855	3.509	3.108	0.965
3	0.933	2.067		0.997	1.723	0.280	0.379	0.974	3.080	3.237	0.927
4	0.815	2.186		0.890	1.850	0.260	0.145	1.013	2.962	3.283	0.914
5	0.743	2.258		0.819	1.924	0.256	0.097	1.030	2.913	3.306	0.907
6	0.701	2.299		0.776	1.971	0.254	0.060	1.040	2.885	3.319	0.904
7	0.676	2.324		0.750	1.997	0.253	0.037	1.047	2.865	3.327	0.902
8	0.661	2.339		0.735	2.013	0.253	0.023	1.049	2.860	3.331	0.901
9	0.651	2.349		0.726	2.022	0.253	0.015	1.051	2.854	3.335	0.900
10	0.647	2.353		0.720	2.027	0.253	0.008	1.053	2.849	3.337	0.899
11	0.644	2.356		0.717	2.030	0.253	0.005	1.053	2.849	3.337	0.899
12	0.641	2.359		0.716	2.032	0.253	0.005	1.053	2.849	3.337	0.899
13	0.640	2.359		0.714	2.034	0.253	0.003	1.054	2.846	3.338	0.899
14	0.640	2.359		0.714	2.034	0.253		1.054	2.846	3.338	0.899
15	0.640	2.359		0.714	2.034	0.253		1.054	2.846	3.338	0.899

Figure 104: **Exercise 13.2** Approximate MVA for (3,3) configuration

Computational effort The Bard-Schweitzer's algorithm uses 13 iterations to get to the final answer. The exact MVA algorithm uses 15 columns of calculation. The number of multiplications and additions required for the exact algorithm is proportional to

$$KR\prod_{r=1}^{R}(1+N_r) = 5 \times 2(4 \times 4) = 160$$

For the Bard-Schweitzer algorithm this is proportional to:

$$K \times R \times iterations = 5 \times 2 \times 13 = 130$$

The difference is a decrease in computational effort of about 29%[58].

13.3 Exercise 13.3

Given The transaction system of PBD, §13.3.

Requested What is the effect on performance if the query transactions get balanced on both disk 1 and disk 2? Compare with the current situation.

Solution The service demand of query transactions for Disk 1 in the original situation (i.e. 180 msec) on average will be split evenly accross both disks when balanced accross both disks. The resulting service demands are listed in table 106 The calculations are performed in spreadsheet `eX13.3.xls`. The responsetimes and throughputs obtained are displayed in figure 107 plus a comparison with the original situation. Both are calculated using the exact MVA algorithm.
The results show an improvement of about 10% for response time and throughput of the update transactions and an improvement of 20 - 30% for the query transactions.

[58]The effort of getting this exercise right with the exact algorithm, is about 5% easier in the exact MVA algorithms' case.

Class	Variable	Population (Update, Query)							
		(0,0)	(0,1)	(0,2)	(0,3)	(1,0)	(1,1)	(1,2)	(1,3)
Query	$R_{1,q}$		0.105	0.144	0.174		0.141	0.177	0.204
	$R_{2,q}$		0.180	0.294	0.422		0.259	0.388	0.529
	$R_{3,q}$		0.000	0.000	0.000		0.000	0.000	0.000
	$X_{0,q}$		3.509	4.566	5.034		2.500	3.540	4.093
	$n_{1,q}$	0	0.368	0.658	0.876		0.353	0.627	0.835
	$n_{2,q}$	0	0.632	1.342	2.124		0.648	1.374	2.165
	$n_{3,q}$	0	0.000	0.000	0.000		0.000	0.000	0.000
Update	$R_{1,u}$					0.375	0.513	0.622	0.704
	$R_{2,u}$					0.480	0.783	1.124	1.500
	$R_{3,u}$					0.240	0.240	0.240	0.240
	$X_{0,u}$					0.913	0.651	0.504	0.409
	$n_{1,u}$	0				0.342	0.334	0.313	0.288
	$n_{2,u}$	0				0.438	0.510	0.566	0.614
	$n_{3,u}$	0				0.219	0.156	0.121	0.098
Class	Variable	(2,0)	(2,1)	(2,2)	(2,3)	(3,0)	(3,1)	(3,2)	(3,3)
Query	$R_{1,q}$		0.176	0.209	0.231		0.210	0.238	0.256
	$R_{2,q}$		0.347	0.491	0.644		0.445	0.602	0.765
	$R_{3,q}$		0.000	0.000	0.000		0.000	0.000	0.000
	$X_{0,q}$		1.912	2.857	3.429		1.527	2.381	2.938
	$n_{1,q}$		0.337	0.597	0.792		0.321	0.567	0.752
	$n_{2,q}$		0.663	1.403	2.208		0.680	1.433	2.248
	$n_{3,q}$		0.000	0.000	0.000		0.000	0.000	0.000
Update	$R_{1,u}$	0.503	0.633	0.728	0.796	0.629	0.746	0.826	0.880
	$R_{2,u}$	0.690	1.036	1.411	1.814	0.926	1.309	1.716	2.146
	$R_{3,u}$	0.293	0.277	0.269	0.264	0.335	0.308	0.294	0.284
	$X_{0,u}$	1.346	1.028	0.831	0.696	1.587	1.270	1.058	0.906
	$n_{1,u}$	0.677	0.651	0.605	0.554	0.998	0.947	0.874	0.797
	$n_{2,u}$	0.929	1.065	1.173	1.263	1.470	1.662	1.816	1.944
	$n_{3,u}$	0.394	0.285	0.224	0.184	0.532	0.391	0.311	0.257

Figure 105: **Exercise 13.2** Exact MVA calculations for (3, 3) configuration

13.4 Exercise 13.4

Given One database server with 2 disks, D1 and D2. Three workload classes: Query (Q), update (U) transactions and interactive (I) users. Input parameters as in PBD, table 13.12.

Requested Use the QN solvers with chapter 13 of PBD to answer the following questions:

- What is the average response time for each class?
- What is the impact on response time if the arrival rate of query transactions increases by 95%?
- In the case of an increased arrival rate of query transactions consider the following hardware upgrades:
    1. Replace disk D1 by one twice as fast.

Class	Service Demand (sec.)		
	Processor	Disk 1	Disk 2
Query	0.105	0.090	0.090
Update	0.375	0.480	0.240

Figure 106: **Exercise 13.3** Service demands

	response		throughput	
	query	update	query	update
original	0.733	2.444	4.093	0.409
balanced	0.572	2.192	5.245	0.456
improvement	22%	10%	28%	12%

Figure 107: **Exercise 13.3** Results and comparison from spreadsheet `eX13.3.xls`

2. Replace the CPU by one twice as fast.

Compare the performance improvements in each case.

- With the increased arrival rate of query transactions and the CPU (twice as fast), draw a graph of response time versus number of simultaneous clients when this number varies from 50 to 250. What is the maximum number of simultaneous clients that can be supported with a response time below 1.5 seconds?

Solution

Part 1. The average response time for each class In order to calculate the response times, we need to solve the model of the described model. The description shows a mixed model consisting of two open classes and one closed class. Using the algorithm from PBD, table 13.9 page 372 we perform the following steps:

1. Solve the open classes using `OpenQN.xls`: see spreadsheet `Ex-13.4-OpenQN.xls`.
2. Elongate the service demands of the closed class:

$$D^e_{i,I} = \frac{D_{i,I}}{1 - U_{i,open}} \forall r \in CPU, D1, D2$$

3. Compute the performance results for the closed model
4. Determine $n_{imckised}(\vec{C}) = \sum_{r=1}^{C}(\vec{C})$
5. Compute the average residence time for the open submodel.

Step 1. Solve open classes In the spreadsheet `Ex13.4-OpenQN.xls` the first step is performed. The results are shown in figure 108.

Class	Utilizations		
	U_{cpu}	U_{D1}	U_{D2}
Q	0.18000	0.09000	0.18000
U	0.15000	0.04500	0.13500
Total	0.33000	0.13500	0.31500

Class	Queue length		
	Q_{cpu}	Q_{D1}	Q_{D2}
Q	0.26866	0.10405	0.26277
U	0.22388	0.05202	0.19708
Total	0.49254	0.15607	0.45985

Class	Residence times			
	R_{cpu}	R_{D1}	R_{D2}	R
Q	0.08955	0.03468	0.08759	0.21183
U	0.14925	0.03468	0.13139	0.31532

Figure 108: **Exercise 13.4** Step 1 : Solve open class performance model and obtain utilizations.

Step 2. Determine open utilization.

$$U_{i,open} = \sum_{r=1}^{O} U_{i,r}$$

Using the results from step 1 the utilization per device for the open classes are:

$$U_{cpu,open} = 0.18 + 0.15 = 0.33$$

$$U_{D1,open} = 0.09 + 0.045 = 0.0135$$

$$U_{D2,open} = 0.18 + 0.135 = 0.315$$

Step 3. Elongate the service demands of the closed classes.

$$D^e_{i,r} = \frac{D_{i,r}}{1 - U_{i,open}} \forall r \in \{interactive(I)\}$$

Again using the results of the previous step, we get:

$$D^e_{cpu,I} = \frac{D_{cpu,I}}{1 - U_{cpu,open}} = \frac{0.09}{1 - 0.33} \approx 0.1343$$

$$D^e_{D1,I} = \frac{D_{D1,I}}{1 - U_{D1,open}} = \frac{0.045}{1 - 0.0135} \approx 0.0456$$

$$D^e_{D2,I} = \frac{D_{D2,I}}{1 - U_{D2,open}} = \frac{0.000}{1 - 0.315} = 0.0000$$

Step 4. Compute the performance results for the closed model. The performance results are obtained through the use of spreadsheet `Ex13.4-step4-ClosedQN.xls` and displayed in figure 109.

	Residence times			
Class	R_{cpu}	R_{D1}	R_{D2}	R
I	6.64801	0.06835	0.00000	6.71636
	Queue length			
Class	$\overline{n}_{\text{cpu}}$	$\overline{n}_{\text{D1}}$	$\overline{n}_{\text{D2}}$	$X_{0,r}$
I	49.491	0.509	0.000	7.44

Figure 109: **Exercise 13.4** Step 4. Solve performance results for closed model.

Step 5. Determine $n_{i,closed}$

$$\overline{n}_{i,closed} = \sum_{r=1}^{C} \overline{n}_{i,r}(\vec{C})$$

As there is only one closed class the queue length is easily determined:

$$\overline{n}_{cpu,closed} = 49.491$$

$$\overline{n}_{D1,closed} = 0.509$$

$$\overline{n}_{D2,closed} = 0.000$$

Step 6. Compute the average residence time for the open submodel.

$$R'_{i,r}(\vec{O}) = \frac{D_{i,r}[1 + \overline{n}_{i,closed}(\vec{C})]}{1 - U_{i,open}(\vec{O})}$$

$$\overline{n}_{i,open} = \lambda_r R'_{i,r}(\vec{O})$$

$$\begin{aligned} R'_{cpu,Q} &= \tfrac{D_{cpu,Q}[1+\overline{n}_{cpu,closed}(\vec{C})]}{1-U_{cpu,open}(\vec{O})} \\ &= \tfrac{0.06[1+49.491]}{1-0.33} \approx 4.522 seconds \\ R'_{D1,Q} &= \tfrac{D_{D1,Q}[1+\overline{n}_{D1,closed}(\vec{C})]}{1-U_{D1,open}(\vec{O})} \\ &= \tfrac{0.03[1+0.509]}{1-0.0135} \approx 0.0459 seconds \\ R'_{D2,Q} &= \tfrac{D_{D2,Q}[1+\overline{n}_{D2,closed}(\vec{C})]}{1-U_{D2,open}(\vec{O})} \\ &= \tfrac{0.06[1+0]}{1-0.315} \approx 0.0876 seconds \end{aligned}$$

$$\begin{aligned} R'_{cpu,U} &= \tfrac{D_{cpu,U}[1+\overline{n}_{cpu,closed}(\vec{C})]}{1-U_{cpu,open}(\vec{O})} \\ &= \tfrac{0.10[1+49.491]}{1-0.33} \approx 7.5360 seconds \\ R'_{D1,U} &= \tfrac{D_{D1,U}[1+\overline{n}_{D1,closed}(\vec{C})]}{1-U_{D1,open}(\vec{O})} \\ &= \tfrac{0.03[1+0.509]}{1-0.0135} \approx 0.0459 seconds \\ R'_{D2,U} &= \tfrac{D_{D2,U}[1+\overline{n}_{D2,closed}(\vec{C})]}{1-U_{D2,open}(\vec{O})} \\ &= \tfrac{0.09[1+0]}{1-0.315} \approx 0.1314 seconds \end{aligned}$$

$$
\begin{aligned}
\overline{n}_{cpu,open} &= \sum_{r=1}^{R} \lambda R'_{cpu,r}(\vec{O}) \\
&= 3.0 \times 4.522 + 1.5 \times 7.536 \\
&= 13.566 + 11.307 = 24.870 \\
\overline{n}_{D1,open} &= \sum_{r=1}^{R} \lambda R'_{D1,r}(\vec{O}) \\
&= 3.0 \times 0.0459 + 1.5 \times 0.0459 \\
&= 0.1377 + 0.06885 = 0.20655 \approx 0.21 \\
\overline{n}_{D2,open} &= \sum_{r=1}^{R} \lambda R'_{D2,r}(\vec{O}) \\
&= 3.0 \times 0.0876 + 1.5 \times 0.1314 \\
&= 0.2628 + 0.1971 = 0.4599 \approx 0.46
\end{aligned}
$$

The average response time for the query transactions is about 4.7 seconds and for the update transactions is about 7.7 seconds. The average response times for interactive transactions is about 6.7 seconds.
In order to check the results, the same model is run through JMT, where calculating a mixed model is automatic (a feature of the JMT/JMVA module). The jmva file is store in `Ex13.4-part1.jmva`.
The results of both the JMT tool and the spreadsheet calculations are printed in figure 110.
Note, that the algorithm that PBD presents does not include utilizations of the closed classes.
In conclusion, from the calculations it is clear that the CPU is the bottleneck for the system and especially for the interactive transactions.

Part 2. Impact on response time of increasing query transactions by 95% With JMT one can do a what-if on the arrival rate of one or more classes. The model of part 1 is used to create a what-if simulation on the arrival rate of the query transactions, see input file `Ex13.4-part2.jmva`. The results are shown in figure 111.

Part 3. Compare two improvements with the case of part 2. With the arrival rate for query transactions at 5.85, two improvements need to be compared:

1. replace disk D1 by one twice as fast.
2. replace the CPU by one twice as fast.

If the disk is twice as fast, the service demands will be half of their original value. In order to know the effect of this change, the model must be solved again with the new service demands. The new service demands of disk D1 are 0.015 for class Q and U, and 0.0225 for class I.
For the CPU the same reasoning goes as for the disk, so the new service demands are 0.03 for class Q, 0.05 for class U and 0.045 for class I.
Using these new service demands we recalculate the model. The JMT input is in the files `ex13.4-part3-D1.jmva` and `ex13.4-part3-CPU.jmva`. The results of the original, the improved disk and the improved CPU variants are in figure 112. In the comparison it is clear, that the improved disk does not improve the system response time or throughput. As the utilization of the CPU was already at 100% this should be no supprise.

Throughput						
	JMT output			Calculations		
Class	Q	U	I	Q	U	I
Aggregate	3.0000	1.5000	7.4444	3.0	1.5	7.44
CPU	3.0000	1.5000	7.4444	3.0	1.5	7.44
Disk 1	3.0000	1.5000	7.4444	3.0	1.5	7.44
Disk 2	3.0000	1.5000	0.0000	3.0	1.5	0.00

Queue length						
	JMT output			Calculations		
Class	Q	U	I	Q	U	I
Aggregate	13.9643	11.5584	50.0000	-	-	50.000
CPU	13.5317	11.2764	49.3679	13.566	11.304	49.491
Disk 1	0.1698	0.0849	0.6321	0.1377	0.0689	0.509
Disk 2	0.2628	0.1971	0.0000	0.2628	0.1971	0.000

Residense times						
	JMT output			Calculations		
Class	Q	U	I	Q	U	I
Aggregate	4.6548	7.7056	6.7164	4.656	7.713	6.716
CPU	4.5106	7.5176	6.6315	4.522	7.536	6.648
Disk 1	0.0566	0.0566	0.0849	0.046	0.046	0.068
Disk 2	0.0876	0.1314	0.0000	0.088	0.131	0.000

Utilization							
	JMT output				Calculations		
Class	Aggregate	Q	U	I	Q	U	I
CPU	1.0000	0.1800	0.1500	0.6700	0.18	0.15	-
Disk 1	0.4700	0.0900	0.0450	0.3350	0.09	0.045	-
Disk 2	0.3150	0.1800	0.1350	0.0000	0.18	0.135	-

Figure 110: **Exercise 13.4** Results from JMT for the original case, rounded to 4 decimals.

The faster CPU however, does improve the response time and the throughput, but total utilization is still very near 100%. The situation for the CPU gives more balance as both the CPU and disk1 are nearing 100% with the faster CPU.

Part 4. Graph 50 to 250 users. The requested graph is painted in figure 113. Alas, there is no value that permits a response time below 1.5 seconds.

Response times			
Class	Q	U	I
Aggregate	6.2490	10.3578	9.0180
CPU	6.0757	10.1261	8.9331
Disk 1	0.0566	0.0566	0.0849
Disk 2	0.1167	0.1751	0.0000

Figure 111: **Exercise 13.4** Part 2. Results from JMT , rounded to 4 decimals.

Response times									
	Original			D1 twice as fast			CPU twice as fast		
Class	Q	U	I	Q	U	I	Q	U	I
Aggregate	6.2490	10.3578	9.0180	6.2490	10.3825	9.0180	2.1704	3.1356	3.0209
CPU	6.0757	10.1261	8.9331	6.1127	10.1878	8.9886	1.3603	2.2672	1.9929
Disk 1	0.0566	0.0566	0.0849	0.01961	0.0196	0.0294	0.6933	0.6933	1.0280
Disk 2	0.1167	0.1751	0.0000	0.11673	0.1751	0.0000	0.1167	0.1751	0.000

Throughput									
	Original			D1 twice as fast			CPU twice as fast		
Class	Q	U	I	Q	U	I	Q	U	I
Aggregate	5.85	1.5000	5.5444	5.85	1.5	5.5444	5.85	1.5	16.5514
CPU	5.85	1.5000	5.5444	5.85	1.5	5.5444	5.85	1.5	16.5514
Disk 1	5.85	1.5000	5.5444	5.85	1.5	5.5444	5.85	1.5	16.5514
Disk 2	5.85	1.5000	0.0000	5.85	1.5	0.0000	5.85	1.5	0.0000

Utilization									
	Original			D1 twice as fast			CPU twice as fast		
Class	Q	U	I	Q	U	I	Q	U	I
CPU	0.3510	0.1500	0.4990	0.3510	0.1500	0.4990	0.1755	0.0750	0.7448
Disk 1	0.1755	0.0450	0.2495	0.0878	0.0225	0.1248	0.1755	0.0450	0.7448
Disk 2	0.3510	0.1350	0.0000	0.3510	0.1350	0.0000	0.3510	0.1350	0.0000

Figure 112: **Exercise 13.4** Part 3: comparing D1 and CPU improvement.

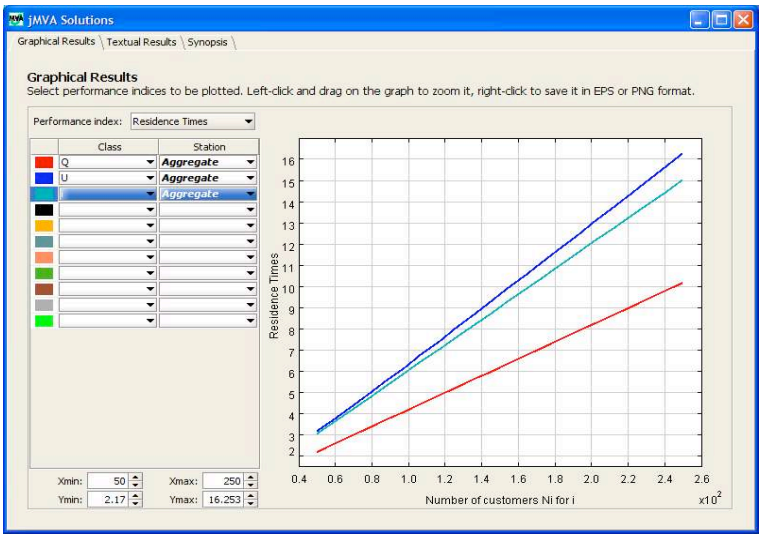


Figure 113: **Exercise 13.4** Part4. Graph 50 - 250 users

13.5 Exercise 13.5

Given System with one CPU and disk. Workload consisting of trivial, complex and batch. Trivial is an open class with $\lambda_t = 10$ tps. Complex is an open class with $\lambda_c = 0.1$ tps. Batch is a closed class with 1 customer and 15 minutes thinktime between jobs.
Furthermore, each IO demands 0.015 msec of CPU and 9 msec of disk service time. Table 13.13 in PBD page 376 contains workload data.

Requested The following items are requested:

- Average response time and throughput for each to the workloads.
- Utilization of CPU and disk
- The residence times at the CPU and the disk for each of the workloads.

Solution Answer to these questions requires solving the model, which is a mixed class QN model. The service demands can be derived from the IO characteristics of the workloads in table 13.13 of PBD.
The trivial transactions need 30.08 msec of CPU service and 47.25 msec of disk service time:

$$D_{t,cpu} = D_{t,cpu}^{\text{other}} + D_{t,cpu}^{\text{IO}} \tag{13.5.1}$$
$$D_{t,cpu}^{\text{IO}} = 0.015 \text{ msec} \times (3.5 \times 5.0 \times (1 - 0.7) \text{ IO's}) \tag{13.5.2}$$
$$= 0.015 \text{ msec} \times 5.25 \text{ IO's} = 0.07875 \text{ msec} \approx 0.08 \text{ msec} \tag{13.5.3}$$
$$D_{t,cpu} = 30 + 0.08 = 30.08 \text{ msec} \tag{13.5.4}$$

$$D_{t,disk} = 9 \text{ msec} \times 5.25 \text{ IO's}) \tag{13.5.5}$$
$$= 47.25 \text{ msec} \tag{13.5.6}$$

The complex transactions need 180.9 msec of CPU service and 540 msec of disk service:

$$D_{c,cpu} = D_{c,cpu}^{\text{other}} + D_{c,cpu}^{\text{IO}} \tag{13.5.7}$$
$$D_{c,cpu}^{IO} = 0.015 \text{ msec} \times (20.0 \times 15.0 \times (1 - 0.8)\text{IO's}) \tag{13.5.8}$$
$$= 0.015 \text{ msec} \times 60 \text{ IO's} \tag{13.5.9}$$
$$= 0.9 \text{ msec} \tag{13.5.10}$$
$$D_{c,cpu} = 180.0 + 0.9 = 180.9 \text{ msec} \tag{13.5.11}$$

$$D_{c,disk} = 60 \text{ IO's} \times 9 \text{ msec} = 540 \text{ msec} \tag{13.5.12}$$

The report transactions need 1300.4 msec of CPU service and 30240 msec of disk service:

$$D_{r,cpu} = D_{r,cpu}^{\text{other}} + D_{r,cpu}^{\text{IO}} \tag{13.5.13}$$
$$D_{r,cpu}^{IO} = 0.015 \text{ msec} \times (120.0 \times 40.0 \times (1 - 0.3)\text{IO's}) \tag{13.5.14}$$
$$= 0.015 \text{ msec} \times 3360 \text{ IO's} \tag{13.5.15}$$
$$= 50.4 \text{ msec} \tag{13.5.16}$$
$$D_{r,cpu} = 1250.0 + 50.4 = 1300.4 \text{ msec} \tag{13.5.17}$$

$$D_{r,disk} = 3360 \text{ IO's} \times 9 \text{ msec} = 30240 \text{ msec} \tag{13.5.18}$$

So the workload parameters are:

Class	Service Demands		
	Trivial msec	Complex msec	Report msec
CPU	30.8	180.9	1300.4
disk	47.25	540.0	30240.0
Param	workload parameters		
	Trivial tps	Complex tps	Report
λ_i	10	0.1	
N_i			1

Now that we have the workload parameters, we can use the algorithm for solving a mixed model to answer the questions.

step 1. solve the open submodel. The open submodel is solved in spreadsheet `ex13.5-step1-OpenQN.xls`. The results are printed in figure 114.

Step 1. Solve open classes See figure 114.

Class	Utilizations		
	Trivial	*Complex*	*Total*
CPU	0.30800	0.01809	0.32609
DISK	0.47250	0.05400	0.52650
Total	0.33000	0.13500	0.31500

Class	Queue length		
	Trivial	*Complex*	*Total*
CPU	0.45703	0.02684	0.48388
Disk	0.99789	0.11404	1.11193

Class	Residence times	
	Trivial	*Complex*
CPU	0.04570	0.26843
Disk	0.09979	1.14044
Total	0.14549	1.40888

Figure 114: **Exercise 13.5** Step 1 : Solve open class performance model and obtain utilizations.

Step 2. Determine open utilization.

$$U_{i,open} = \sum_{r=1}^{O} U_{i,r}$$

Using the results from step 1 the utilization per device for the open classes are:

$$U_{cpu,open} = 0.32609$$

$$U_{D,open} = 0.52650$$

Step 3. Elongate the service demands of the closed classes.

$$D^e_{i,r} = \frac{D_{i,r}}{1 - U_{i,open}} \forall r \in \{report\}$$

Again using the results of the previous step, we get:

$$D^e_{cpu,report} = \frac{D_{cpu,report}}{1 - U_{cpu,open}} = \frac{1.3004}{1 - 0.32609} \approx 1.9296$$

$$D^e_{D,report} = \frac{D_{D,report}}{1 - U_{D,open}} = \frac{30.240}{1 - 0.52650} \approx 63.865$$

Step 4. Compute the performance results for the closed model. The performance results are obtained through the use of spreadsheet `Ex13.5-step4-ClosedQN.xls` and displayed in figure 115.

	Residence times	
Class	Report	
CPU	1.9296	
Disk	63.8650	
Total	65.7946	
	Queue length	
Class	Report	$X_{0,r}$
$\overline{n}_{\text{cpu}}$	0.029	0.02
$\overline{n}_{\text{D}}$	0.971	

Figure 115: **Exercise 13.5** Step 4. Solve performance results for closed model.

Step 5. Determine $n_{i,closed}$

$$\overline{n}_{i,closed} = \sum_{r=1}^{C} \overline{n}_{i,r}(\vec{C})$$

As there is only one closed class the queue length is easily determined:

$$\overline{n}_{cpu,closed} = 0.029$$

$$\overline{n}_{D,closed} = 0.971$$

Step 6. Compute the average residence time for the open submodel.

$$R'_{i,r}(\vec{O}) = \frac{D_{i,r}[1 + \overline{n}_{i,closed}(\vec{C})]}{1 - U_{i,open}(\vec{O})}$$

$$\overline{n}_{i,open} = \lambda_r R'_{i,r}(\vec{O})$$

$$\begin{aligned} R'_{cpu,Trivial} &= \tfrac{D_{cpu,Trivial}[1+\overline{n}_{cpu,closed}(\vec{C})]}{1-U_{cpu,open}(\vec{O})} \\ &= \tfrac{0.0308[1+0.029]}{1-0.32609} \approx 0.0471 seconds \\ R'_{D,Trivial} &= \tfrac{D_{D,Trivial}[1+\overline{n}_{D,closed}(\vec{C})]}{1-U_{D1,open}(\vec{O})} \\ &= \tfrac{0.04725[1+0.971]}{1-0.52650} \approx 0.1967 seconds \\ R_{Trivial} &= 0.0471 + 0.1967 = 0.2438 \text{ seconds} \end{aligned}$$

$$
\begin{aligned}
R'_{cpu,Complex} &= \frac{D_{cpu,Complex}[1+\overline{n}_{cpu,closed}(\vec{C})]}{1-U_{cpu,open}(\vec{O})} \\
&= \frac{0.1809[1+0.029]}{1-0.32609} \approx 0.2766 seconds \\
R'_{D,Complex} &= \frac{D_{D,Complex}[1+\overline{n}_{D,closed}(\vec{C})]}{1-U_{D,open}(\vec{O})} \\
&= \frac{0.54[1+0.971]}{1-0.52650} \approx 2.2478 seconds \\
R_{Complex} &= 0.2766 + 2.2478 = 2.5244 \text{ seconds}
\end{aligned}
$$

$$
\begin{aligned}
\overline{n}_{cpu,open} &= \sum_{r=1}^{R} \lambda R'_{cpu,r}(\vec{O}) \\
&= 10.0 \times 0.0471 + 0.1 \times 0.2766 \\
&= 0.471 + 0.02766 = 0.49866 \\
\overline{n}_{D,open} &= \sum_{r=1}^{R} \lambda R'_{D1,r}(\vec{O}) \\
&= 10 \times 0.1967 + 0.1 \times 2.2478 \\
&= 1.967 + 0.22478 = 2.19178 \approx 2.19
\end{aligned}
$$

Average reponse time and throughput The response times and residence times are shown in figure 116. Remains the throughput for the report class to be calculated. The thinktime for this class is 15 minutes or 900 seconds.

$$
X_{0,r}(\vec{N}) = \frac{N_r}{Z_r + \sum_{i=1}^{K} R'_{i,r}(\vec{N})}
$$

$$
X_{0,Report} = \frac{1}{900 + 65.7946} \approx 0.0010354 \text{ tps or } 3.7 \text{ tph}
$$

Responsetimes			
Class	Trivial	Complex	Report
CPU	0.0471	0.2766	1.9296
Disk	0.1967	2.2478	63.8650
Total	0.2438	2.5244	65.7946

Throughput			
Class	Trivial tps	Complex tps	Report tph
Total	10.0	0.1	3.7

Figure 116: **Exercise 13.5** Average response times and throughput.

What is $U_{i,closed}$? Finally, the utilization per device must be determined. From the $U_{i,open}$ we know that $U_{cpu,open} = 0.32609$ and $U_{disk,open} = 0.52650$. But we have not calculated the $U_{i,closed}$.
From the utilization law we know that $U_{i,r} = X_{0,r}D_{i,r}$ so that

$$
\begin{aligned}
U_{cpu,\text{report}} &= X_{0,\text{report}} D_{cpu,\text{report}} && (13.5.19) \\
&= 0.0010354 * 1.9296 \approx 0.001998 && (13.5.20)
\end{aligned}
$$

$$
\begin{aligned}
U_{disk,\text{report}} &= X_{0,\text{report}} D_{disk,\text{report}} && (13.5.21) \\
&= 0.0010354 * 63.8650 \approx 0.06613 && (13.5.22)
\end{aligned}
$$

From these values the total utilization can be calculated:

$$U_{cpu} \approx 0.32609 + 0.001998 = 0.3281 = 32.81\% \tag{13.5.23}$$

$$U_{disk} \approx 0.52650 + 0.06613 = 0.59263 \approx 59.26\% \tag{13.5.24}$$

14 Chapter 14

14.1 Exercise 14.1

Given Example 14.2 (CS-model) with a 10 Mbps Ethernet in stead of a 100 Mbps, a slot duration of 51.2 μsec (i.e. 10^{-6} seconds), and an $L_p = 1518$ bytes including header and payload.

Requested What would be the impact of using a 10 Mbps Ethernet in stead of the original 100 Mbps Ethernet?

Solution Consider that the model as described in §14.2 is essientially unchanged, so that PBD figure 14.4, page 384 and PBD table 14.1, page 386 are both valid, apart from the network for which the bandwidth changes from 100 Mbps to 10 Mbps.

Recalculating 100 Mbps results On page 391, the PBD specifies the results to be a throughput of 82.17 SQL requests per second and a response time of 0.265 seconds.
The specified service demand $D^{\text{sv}}_{\text{processor}} = 0.12$ has a maximum throughput of 8.33, which is incompatible with the specified result on page 391. The assumption is, that the service demand for the CPU should be about 0.012 seconds.
In spreadsheet `Example14.2.xls` the original case for 100 Mbps is calculated. The throughput and response time[59] are identical to the specified result when calculated at a processor service demand of 0.01217 seconds. The network time is calculated to be about 0.00100 at 100 Mbps and a network service demand of $NP_{sql} \times avgL_p/B \approx 0.00097152$.
Conclusion is, that the specified service demand of 0.12 seconds is in error and should be 0.012. When using 0.01217 seconds the throughput and response time are equal to PBD.

Calculating the network with 10 Mbps Using the above model with a service demand of 0.01217 seconds, we can recalculate the results using a 10 Mbps model. This results in 0.0195713 seconds network time at a service demand of $NP_{sql} \times avgL_p/B \approx 0.0097152$ seconds[60]. The total network time increases, but does not touch the total response time.
At 30 customers the processes are mostly waiting for response from the processor, leaving the network time to be irrelevant to the total response time. An increase in network time just leads to a decrease in waiting time for the processor as the calculations in the spreadsheet clearly show.

14.2 Exercise 14.2

Given Example 14.4.

Requested What would the response times of each class be if the LAN is replaced by a LAN of 100 Mbps?

[59] remember to subtract the delay time from R_0
[60] remark: one zero less, or a factor 10 larger than at 100 Mbps

Solution The attribution of the LAN to the response time of each class in the original situation of a 10 Mbps LAN is equal to 0.00017 to 0.167, 0.00044 to 0.310 and 0.00134 to 0.552. This represents contributions of approximately 0.1%, 0.14% and 0.24%.
The replacement LAN will have 10 times the original bandwidth. It is safe to say, that the LAN contribution to the response times will be less than in the original case of 10 Mbps i.e. have less impact on the total response time. Thus, the total response times will not change significantly.

14.3 Exercise 14.3

Given Example 14.4 with twice the number of running processes, and the same proportion of trivial, average and complex transactions.

Requested Compute the new values of response time and throughput for each class. Make and justify recommendations to guarantee that the average transaction response time does not exceed 2 seconds.

Solution The algorithm described in PBD, table 14.7 is implemented in a set of PYTHON programs in subdirectory `Closed-LD-multiple-class`. Using the program `example14_4.py` one can execute the example 14.4 as specified starting on page 394 of the PDB. The results are summarized in figure 117. This verifies the algorithm, because the results are mostly the same as in PDB, table 14.5, apart from rounding errors.
When creating the model the specified thinktime is modeled as a delay device called client. The total response time in figure 117 includes the delay for the client, i.e. think time. The field `R -/- Delay` contains the response time excuding the client delay, or the response time as specified in PBD.

Example 14.4 Result in 16 iterations							
Req	Residence time (sec)				R	R -/- Delay	X_0
Type	cpu	disk	lan	client	(sec)	(sec)	(req/s)
trivial	0.00342	0.16416	0.00017	0.10000	0.268	0.168	37.35
average	0.02131	0.28795	0.00043	0.20000	0.510	0.310	39.24
complex	0.03552	0.51480	0.00135	0.40000	0.952	0.552	5.25

Figure 117: **Exercise 14.3** Results for example 14.4, the differences with table PBD-14.5 are underlined.

Now that the algorithm is verified through the results, we use this algorithm to calculate the response time for doubling or tripling the number of processes, keeping the proportions equal to those in the example. Figure 118 shows the results for doubling and tripling the number of processes.
As you can see, the average response time for complex transactions nears 2 seconds for a double load and exceeds 2 seconds when the load is tripled. This response time is mainly due to disk accesses (2.2469 of 2.286 seconds).
The recommendation would be to install a faster disk, or double the disk capacity and balance the accesses accross the disks.
In figure 119 the results for 2 disks with doubled and tripled load are shown. The service demands for both disks are taken to be half of the original service demands, on the assumption that the disk accesses are properly balanced. As you can see, the response time for all transactions is now well below 2 seconds.

Example 14.4 Doubled nr of processes (16 iterations)							
Req	Residence time (sec)				R	R -/- Delay	X_0
Type	cpu	disk	lan	client	(sec)	(sec)	(req/s)
trivial	0.00358	0.43978	0.00017	0.10000	0.544	0.444	36.80
average	0.02245	0.77009	0.00044	0.20000	0.993	0.793	40.28
complex	0.03737	1.37559	0.00135	0.40000	1.814	1.414	5.51

Example 14.4 Tripled nr of processes (16 iterations)							
Req	Residence time (sec)				R	R -/- Delay	X_0
Type	cpu	disk	lan	client	(sec)	(sec)	(req/s)
trivial	0.00363	0.71871	0.00017	0.10000	0.823	0.723	36.47
average	0.02281	1.25808	0.00044	0.20000	1.481	1.281	40.50
complex	0.03796	2.24690	0.00135	0.40000	2.686	2.286	5.58

Figure 118: **Exercise 14.3** The input of ex. 14.4 with double and triple processes

Example 14.4 with 2 disks and doubled load (44 iterations)								
Req	Residence time (sec)					R	R -/- Delay	X_0
Type	cpu	disk 1	disk 2	lan	client	(sec)	(sec)	(req/s)
trivial	0.03634	0.04784	0.04784	0.00018	0.10000	0.232	0.132	86.13
average	0.22710	0.08418	0.08418	0.00046	0.20000	0.596	0.396	67.12
complex	0.37828	0.15035	0.15035	0.00142	0.40000	1.080	0.680	9.26

Example 14.4 with 2 disks and tripled load (54 iterations)								
Req	Residence time (sec)					R	R -/- Delay	X_0
Type	cpu	disk 1	disk 2	lan	client	(sec)	(sec)	(req/s)
trivial	0.06589	0.07917	0.07917	0.00018	0.10000	0.324	0.224	92.47
average	0.41330	0.13917	0.13917	0.00046	0.20000	0.892	0.692	67.26
complex	0.68796	0.24851	0.24851	0.00142	0.40000	1.586	1.186	9.46

Figure 119: **Exercise 14.3** Example 14.4 with 2 disks and double and triple processes

14.4 Exercise 14.4

Given An interactive system with M terminals, one cpu and two disks. The measurement results are printed in PBD, table 14.7 on page 408. Due to memory constraints only 5 processes are allowed in memory at the same time. In the new situation the system will be redesigned such that only 60% of the old thinktime is necessary.

Requested this exercise should produce the following results:

1. Determine the number of terminals $M_{\max}$ where the response time does not exceed 3 seconds.
2. Plot the response times versus the number of terminals
3. How much time is spent in the system and waiting for memory when $M_{\max}$ terminals are active?
4. Compute processor and disk utilizations at $M_{\max}$.

5. Recommend and justify a system upgrade to handle 1.2 $M_{\max}$ with an average response time that does not exceed 3 seconds.

To take memory queuing into account use the load-dependent version of MVA.

Solution From the exercise description the following considerations apply:

- The data makes no distinction in transaction types, so there is but one transaction class.
- The description shows a system best described using a closed model with currently 100 users and 5 processes concurrent in memory
- In order to model thinktime, the client can be modeled as a delay device with the service demand equal to thinktime.
- Given the description, we must model memory indirectly through the maximum number of processes that run concurrently.
- We make the simplifying assumption, that processes enter memory directly without going through the processor first. The processor time used for memory access is part of the specified cpu service demand and can not be distinguished from processor time used for other activities.

Thus, we model the system as in figure 120 on page 153[61].
For this model the following assumptions apply:

- When a process arrives, if memory is available, the process enters the system to run.
- If memory is not available, the process enters a waiting queue.
- When a process finishes using the cpu and disks it leaves memory to join the client for thinktime.
- Because of the number of customers, we assume that more than 5 processes are waiting to be executed. I.e. as soon as one process finishes, another process takes its place in memory. So, for the subsystem of cpu and disks a thinktime of zero is assumed.

To calculate the new model, we need to derive service demands, throughput and think time from the current measurements. Using the definition (see PBD, chapter 3) we can derive throughput: $X_0 = \frac{C_i}{T} = \frac{11808}{3600} = 3.28$.
Using the utilization law we can now derive service demands per device:

$$D_{\text{cpu}} = \frac{U_{\text{cpu}}}{X_0} = \frac{0.26}{3.28} \approx 0.079 \tag{14.4.1}$$

$$D_{\text{disk1}} = \frac{U_{\text{disk1}}}{X_0} = \frac{0.41}{3.28} \approx 0.125 \tag{14.4.2}$$

$$D_{\text{disk2}} = \frac{U_{\text{disk2}}}{X_0} = \frac{0.33}{3.28} \approx 0.101 \tag{14.4.3}$$

From the operational law for interactive response time $X_0 = \frac{M}{Z+R}$ we can also derive the original think time:

$$Z_{\text{old}} = \frac{M}{X_0} - R = \frac{100}{3.28} - 0.47 \approx 30.0 \text{ seconds} \tag{14.4.4}$$

[61] this model resembles the model described in figure 2.9 on page 50 of the PBD.

The new think time will be $0.6 \times Z_{\text{old}} = 18$ seconds. The leads to the following input parameters for the model:

service demands	D_i
CPU	0.079
disk1	0.125
disk2	0.101
Z_{sys}	18 sec
Z_{subsys}	0 sec

The model now consists of a submodel containing a maximum of 5 processes with one cpu and two disks, and the total model containing this submodel with M customers and a queue for the submodel as shown in figure 120.
The subdivision into two systems as in figure 120 is necessary to answer the questions about memory and waiting times for memory. Calculating this model involves two systems: the total closed system with 100 customers and the subsystem of cpu and disks and 5 processes. Because the memory queue and waitingtimes are of primary concern, we short the client-and-memory partition and calculate the cpu-disks-system as a subsystem first.
We then replace the subsystem by a flow equivalent server (FES) in the total system: figure 121. In the FES, the service rates for each number of processes in the submodel are replaced by the equivalent throughputs as calculated in the separate submodel.
In summary we perform the following steps:

1. Solve the reduced network containing the cpu and the 2 disks to determine the throughput when there are n customers in the network, $n = 0, 1, 2, \ldots 5$.

2. In order to simulate memory expansion we also calculate values of n larger than 5.

3. Replace the reduced network by a load dependent "Flow equivalent server" (FES) whose load dependent service rate, $\mu(n)$ is equal to the throughput of the shorted network with n customers, $n = 0, 1, 2, \ldots 5$. $\mu(n) = \mu(5)$ for $n > 5$. In case of memory expansion, the throughput must be recalculated for the higer value of n. Remark that the service demand for the FES is equal to the reciproke of the service rate (throughput in the submodel).

4. recalculate the complete system using the exact single system load dependent algorithm.

Solving the submodel Using a copy of the `ClosedQN` from chapter 13 the submodel is solved in spreadsheet `Exercise-14.4-ClosedQN-submodel.xls`.
The model is solved for 5 processes, and also for more than 5 processes, to simulate to increase of memory in the server. Initially, we are only concerned with 5 processes.
The results are printed in figure 122.
The second column shows the throughput that will be used as service rate in the flow equivalent server (FES). The third column ($1/X_0$) represents the service demand of the FES. These service demands show a drop in duration from 5 concurrent processes to 20 concurrent processes. This means, that the more processes can run concurrently, the shorter each process should take.
However, as the first disk has a utilization of more than 98% at 20 concurrent processes, no significant improvement should be expected from increasing memory further with the current server.

Solving the model Inserting the flow equivalent server into the model in accordance with figure 121 on page 154, we can solve the model using the exact single class load-dependent algorithm (Figure PBD-14.5 on page 389). The service rate is made equal to the throughput as calculated in the submodel (see figure 122.
The algorithm was implemented in `exercise_14_4.py`. The output of the program is written to the file [pbd-exercise-14.4-exact-n-procs.txt] where n contains the number of processes used in the submodel. The main part of the output for 5 processes is shown in figure 123.
As you can see, the value of $M_{\max}$ is equal to 139, the transaction spends about 0.1484 seconds in the computer system (service demand for 5 customers) and about $2.90 - 0.15 = 2.75$ seconds waiting in the queue.
Using the submodel results, the utilizations for CPU and disks will be equal to the values at 5 processes, because no more processes are allowed to run concurrently. Therefore the utilizations for CPU, disk1 and disk2 are equal to 53.2%, 84.2% and 68.0% as you can see in figure 122.
JMT was also used to solve the model with 5 internal processes with comparable results.

Graph: response time versus number of customers The graphs for response time versus number of terminals is printed in figure 124 on page 156. The first graph is derived from the home-made PYTHON-implementation, the second graph is from JMT. It appears that the JMT graph is in error[62], because the response time *decreases* in stead of increases.

Supporting $1.2 \times M_{\max} = 167$ customers In order to process $1.2 \times M_{\max}$ and maintain a response time of 3 seconds or less, the system must be altered. The options are:

1. Increase memory to a point where 167 users can work and receive a response time of 3 seconds or less.
2. Decrease service demand for the FES.
3. A combination of both

Increasing memory As cpu and disks are not yet saturated at 5 concurrent processes, increasing memory seems to be the most logical action.
We adapted the program to simulate increasing the memory until 20 concurrent processes (figure 122 After running the simulation with 5 to 20 processes running concurrently, it turns out, that at 20 processes only 162 terminals can be active for a maximum response time of 3.0 seconds.
Conclusion must be, that increase of memory is not enough. The first option therefore is not sufficient for the problem of the customer.

Decrease the service demand for the FES Disk 1 in the subsystem appears to be the bottleneck. Possible solutions to remove this bottleneck are:

- speeding up the application
- replacing the disk by a faster disk

[62] A question is submittted to the JMT help forum to solve this issue.

- expanding the number of disks and balancing between these disks
- introducing a form of caching if this could decrease the amount of disk hits on disk 1

For this experiments let us assume that either the application is changed to use the disk more efficiently (i.e. need less service from disk 1) or we replace disk1 by a faster disk. Let us also assume, that the service demand of disk1 in the new configuration is equal to the service demand of disk2.
The parameters for the subsystem will now be:

service demands	D_i
CPU	0.079
disk1	0.101
disk2	0.101
Z_{sys}	18 sec
Z_{subsys}	0 sec

Resolving the submodel produces the results as printed in figure 125 on page 157. As figure 125 shows the throughput at 20 processes is higher than with the original disk. Also the utilization for disk1 is lower, while the utilizations for disk2 and cpu are higher i.e. better used.

Solving the submodel with a faster disk1 Using the figures from fig. 125 on page 157 in a recalculation of the total system, the result is, that 167 terminals with a maximum response time of 3 seconds can be supported using a faster disk as specified and increasing the memory to allow 7 processes to run concurrently. See figure 126.

Checking the results The intention to re-use the approximate multiclass load-dependent algorithm to check the results is thwarted by 2 factors:

1. In order to find the same result, the algorithm must be run repetitively for each number of customers between 100 and 300 i.e. 200 runs.
2. Also, the algorithm does not seem to converge for 300 customers.

In stead of using the multiclass algorithm, JMT ran the same model[63] for 5 processes and came to the same results as can be seen in the jmt file: `exercise-14.4.jmva`. All is well except for the response time graph which strangely deviates from ClosedQN.xls, as yet unexplained. We suspect an error in the JMT-algorithm implementation.

14.5 Exercise 14.5

Given Equations PBD-14.4.14, PBD-14.4.15 and PBD-14.4.16

Requested Show that eq. PBD-14.4.15 and PBD-14.4.16 are the solutions to equation PBD-14.4.14.

Solution equation PBD-14.4.15 We start with equation PBD-14.4.15 and prove it by induction. First we prove that eq. PBD-14.4.15 is correct for $j = 1$. Then we prove, that if eq. PBD-14.4.15 is correct for any $j \in \{1, \ldots, |\vec{N}| - 1\}$, then eq. 14.4.15 is also correct for $j + 1$.

[63] using the service demands in stead the service rate

Equation PBD-14.4.15 induction step 1. According to equation PBD-14.4.14 with $j = 1$:

$$P_i(1|\vec{N}) = \frac{\sum_{r=1}^{R} D_{i,r} X_{0,r}(\vec{N})}{\alpha(1)} P_i(0|\vec{N}) \tag{14.5.1}$$

This equation is also equal to equation PBD-14.4.15. So, for $j = 1$ the equation is correct.

Equation PBD-14.4.15 induction step 2. Assume, that equation PBD-14.4.15 is correct for some $j-1$ for which $j \in \{2, \ldots, |\vec{N}|\}$. According to eq PBD-14.4.15 the following holds (induction assumption):

$$P_i(j-1|\vec{N}) = P_i(0|\vec{N}) \prod_{k=1}^{j-1} \frac{\sum_{r=1}^{R} D_{i,r} X_{0,r}(\vec{N})}{\alpha_i(j-1)} \tag{14.5.2}$$

By equation PBD-14.4.14 the following also holds:

$$P_i(j|\vec{N}) = \frac{\sum_{r=1}^{R} D_{i,r} X_{0,r}(\vec{N})}{\alpha_i(j)} P_i(j-1|\vec{N}) \tag{14.5.3}$$

Thus using the induction assumption (equation 14.5.2) and filling in $P_i(j-1|\vec{N})$ we get:

$$P_i(j|\vec{N}) = \frac{\sum_{r=1}^{R} D_{i,r} X_{0,r}(\vec{N})}{\alpha_i(j)} P_i(0|\vec{N}) \prod_{k=1}^{j-1} \frac{\sum_{r=1}^{R} D_{i,r} X_{0,r}(\vec{N})}{\alpha_i(j-1)} \tag{14.5.4}$$

$$P_i(j|\vec{N}) = P_i(0|\vec{N}) \prod_{k=1}^{j} \frac{\sum_{r=1}^{R} D_{i,r} X_{0,r}(\vec{N})}{\alpha_i(j)} \tag{14.5.5}$$

By equation 14.5.5 equation PBD-14.4.15 is valid for j if the equation is valid for $j-1$ and $j \in \{2, \ldots, |\vec{N}|\}$. Because eq. PBD-14.4.15 holds for $j = 1$, it must also hold for all values of $j \in \{1, \ldots, |\vec{N}|\}$. □

Solution equation PBD-14.4.16 All probabilities should sum to 1, so that the following must hold:

$$\sum_{j=0}^{|\vec{N}|} P_i(j|\vec{N}) = 1 \tag{14.5.6}$$

Filling in equation PBD-14.4.15, we get:

$$P_i(0|\vec{N}) + P_i(0|\vec{N}) \left\{ \prod_{k=1}^{1} \frac{\sum_{r=1}^{R} D_{i,r} X_{0,r}(\vec{N})}{\alpha_i(1)} + \cdots + \prod_{k=1}^{j} \frac{\sum_{r=1}^{R} D_{i,r} X_{0,r}(\vec{N})}{\alpha_i(k)} + \cdots + \prod_{k=1}^{|\vec{N}|} \frac{\sum_{r=1}^{R} D_{i,r} X_{0,r}(\vec{N})}{\alpha_i(|\vec{N}|)} \right\} = 1$$

$$P_i(0|\vec{N}) + P_i(0|\vec{N}) \left\{ \sum_{j=1}^{|\vec{N}|} \prod_{k=1}^{j} \frac{\sum_{r=1}^{R} D_{i,r} X_{0,r}(\vec{N})}{\alpha_i(k)} \right\} = 1 \tag{14.5.7}$$

$$P_i(0|\vec{N}) = \left[1 + \sum_{j=1}^{|\vec{N}|} \prod_{k=1}^{j} \frac{\sum_{r=1}^{R} D_{i,r} X_{0,r}(\vec{N})}{\alpha_i(k)} \right]^{-1} \qquad \square \tag{14.5.8}$$

14.6 Exercise 14.6

Given Equations PBD-14.5.24 and PBD-14.5.25

Requested Prove that both equations reduce to $P_i(j|\vec{L}) = \frac{U_i^j}{1-U_i}$ for $j \geq 0$.

Solution If device i is load independent, this means that $\alpha(j) = 1 \quad \forall j$. This also implies that $\beta(j) = 1 \quad \forall j$. If we start with PBD-14.5.25 and fill in these values we get:

$$P_i(0|\vec{L}) = \left[\sum_{j=0}^{\infty} U_i^j\right]^{-1} = \frac{1}{\sum_{j=0}^{\infty} U_i^j} \tag{14.6.1}$$

Replacing the sum in the denominator by $\sum_{j=0}^{\infty} x^j$ with $|x| \leq 1$ and $x = U_i$ and realizing that this is equal to the generating function of $\frac{1}{1-x}$, we can rewrite the above as:

$$P_i(0|\vec{L}) = \frac{1}{\frac{1}{1-U_i}} = (1 - U_i) \tag{14.6.2}$$

Now, we can fill in all known values into PBD-14.5.24 and get:

$$P_i(j|\vec{L}) = P_i(0|\vec{L})\frac{U_i^j}{\beta(j)} \qquad \forall j \geq 1 \tag{14.6.3}$$

$$P_i(j|\vec{L}) = (1 - U_i)\frac{U_i^j}{1} = (1 - U_i)U_i^j \qquad \forall j \geq 1 \tag{14.6.4}$$

This formula now also fits $P_i(0|\vec{L})$. because $U_i^0 = 1$ for any value of U_i. So we get:

$$P_i(j|\vec{L}) = (1 - U_i)U_i^j \qquad \forall j \geq 0 \tag{14.6.5}$$

Which is what we needed to prove. □

14.7 Exercise 14.7

Given Equations PBD-14.5.26, PBD-14.5.27, PBD-14.5.28.

Required Assuming $\alpha_i(j) = \alpha_i(w_i)$ for $j \geq w_i$ prove the mentioned equations.

Solution

proving PBD-14.5.26 The definition of $P_i(j|\vec{L})$ is split between $j = 1, \ldots, w_i$ and $j > w_i$. Because the first part of the definition is (by definition) equal to equation PBD-14.5.24, this needs no proof. We know that $\alpha_i(j) = \alpha_i(w_i)$ for $j > w_i$, so that every $\beta_i(j)$ for $j > w_i$ can be written as $[\alpha_i(w_i)]^{j-w_i}\alpha_i(w_i)\alpha_i(w_i - 1)\ldots\alpha_i(1)$. Filling this in in equation PBD-14.5.24 gives the second part of the definition of PBD-14.5.26 directly.

proving PBD-14.5.27

$$P_i(0|\vec{L}) = \left[\sum_{j=0}^{\infty} \frac{U_i^j}{\beta_i(j)}\right]^{-1}$$

$$P_i(0|\vec{L}) = \left[\sum_{j=0}^{w_i} \frac{U_i^j}{\beta_i(j)} + \sum_{j=w_i+1}^{\infty} \frac{U_i^j}{\beta_i(j)}\right]^{-1}$$

The $\beta_i(j)$ in the second term can be rewritten as $[\alpha_i(w_i)]^{j-w_i} \times \alpha_i(w_i)\alpha_i(w_i-1)\ldots\alpha_i(1)$ leading to

$$P_i(0|\vec{L}) = \left[\sum_{j=0}^{w_i} \frac{U_i^j}{\beta_i(j)} + \sum_{j=w_i+1}^{\infty} \frac{U_i^j}{[\alpha_i(w_i)]^{j-w_i} \times \alpha_i(w_i)\alpha_i(w_i-1)\ldots\alpha_i(1)}\right]^{-1}$$

$$P_i(0|\vec{L}) = \left[\sum_{j=0}^{w_i} \frac{U_i^j}{\beta_i(j)} + \frac{1}{\alpha_i(w_i)\alpha_i(w_i-1)\ldots\alpha_i(1)} \sum_{j=w_i+1}^{\infty} \frac{U_i^j}{[\alpha_i(w_i)]^{j-w_i}}\right]^{-1}$$

By multiplying each term in the second sum by $[\alpha_i(w_i)]_i^w$ the powers of U_i and $\alpha_i(w_i)$ both become j. By replacing the denomitor before the second sum with the relevant β-expression, we get:

$$P_i(0|\vec{L}) = \left[\sum_{j=0}^{w_i} \frac{U_i^j}{\beta_i(j)} + \frac{[\alpha_i(w_i)]_i^w}{\beta_i(w_i)} \sum_{j=w_i+1}^{\infty} \left(\frac{U_i}{\alpha_i(w_i)}\right)^j\right]^{-1}$$

Now the second partial infinite sum can be replaced by a generating function as follows (think $\frac{U_i}{\alpha}$ in place of ρ):

$$\sum_{j=a}^{\infty} \rho^j = \rho^a \sum_{j=0}^{\infty} \rho^j = \frac{\rho^a}{1-\rho}$$

using this fact we get:

$$P_i(0|\vec{L}) = \left[\sum_{j=0}^{w_i} \frac{U_i^j}{\beta_i(j)} + \frac{[\alpha_i(w_i)]_i^w}{\beta_i(w_i)} \frac{\left[\frac{U_i}{\alpha_i(w_i)}\right]^{w_i+1}}{1-\frac{U_i}{\alpha_i(w_i)}}\right]^{-1}$$

This proves equation PBD-14.5.27

proving PBD-14.5.28 Starting from equation PBD-14.5.23 we have:

$$\overline{N}_i(\vec{L}) = \sum_{j=1}^{\infty} jP_i(j|\vec{L})$$

Using equation PBD-14.5.26 we get:

$$\overline{N}_i(\vec{L}) = \sum_{j=1}^{w_i} jP_i(0|\vec{L})\frac{U_i^j}{\beta_i(j)} + \sum_{j=w_i+1}^{\infty} jP_i(0|\vec{L})\frac{U_i^j}{\beta_i(w_i)[\alpha_i(w_i)]^{j-w_i}}$$

$$\overline{N}_i(\vec{L}) = P_i(0|\vec{L}) \left\{ \sum_{j=1}^{w_i} j \frac{U_i^j}{\beta_i(j)} + \sum_{j=w_i+1}^{\infty} j \frac{U_i^j}{\beta_i(w_i)[\alpha_i(w_i)]^j} [\alpha_i(w_i)]^{w_i} \right\}$$

$$\overline{N}_i(\vec{L}) = P_i(0|\vec{L}) \left\{ \sum_{j=1}^{w_i} j \frac{U_i^j}{\beta_i(j)} + \frac{[\alpha_i(w_i)]^{w_i}}{\beta_i(w_i)} \sum_{j=w_i+1}^{\infty} j \left(\frac{U_i}{\alpha_i(w_i)} \right)^j \right\}$$

Using the hinted formula with $\rho = \frac{U_i}{\alpha_i(w_i)}$ yields

$$\overline{N}_i(\vec{L}) = P_i(0|\vec{L}) \left\{ \sum_{j=1}^{w_i} j \frac{U_i^j}{\beta_i(j)} + \frac{[\alpha_i(w_i)]^{w_i}}{\beta_i(w_i)} \left(\frac{U_i}{\alpha_i(w_i)} \right)^{w_i+1} \frac{\frac{U_i}{\alpha_i(w_i)} + (w_i+1)(1 - \frac{U_i}{\alpha_i(w_i)})}{(1 - \frac{U_i}{\alpha_i(w_i)})^2} \right\}$$

$$\overline{N}_i(\vec{L}) = P_i(0|\vec{L}) \left\{ \sum_{j=1}^{w_i} j \frac{U_i^j}{\beta_i(j)} + \frac{U_i^{w_i+1}}{\beta_i(w_i)\alpha_i(w_i)} \frac{\frac{U_i}{\alpha_i(w_i)} + (w_i+1)(1 - \frac{U_i}{\alpha_i(w_i)})}{(1 - \frac{U_i}{\alpha_i(w_i)})^2} \right\}$$

This concludes the proofs for exercise 14.7. □

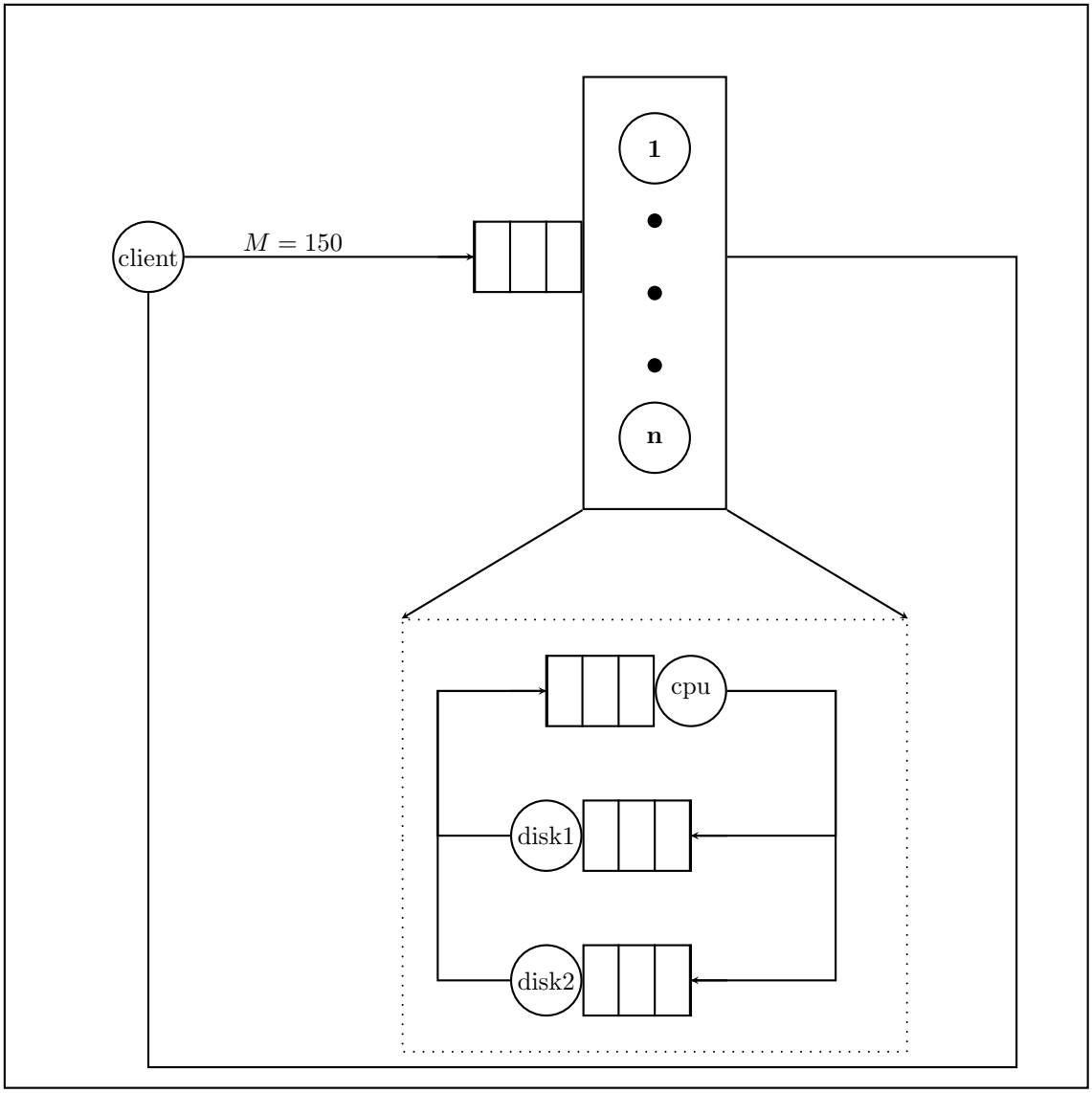


Figure 120: **Exercise 14.4** QN model for the new situation.

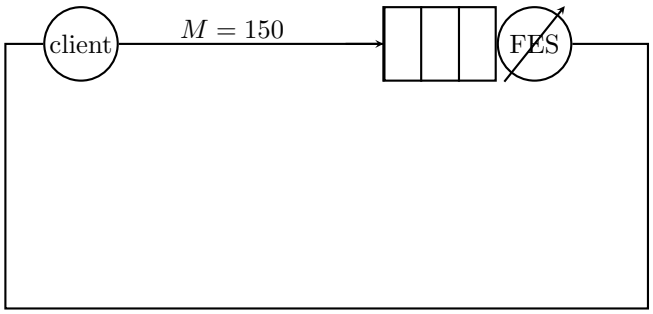


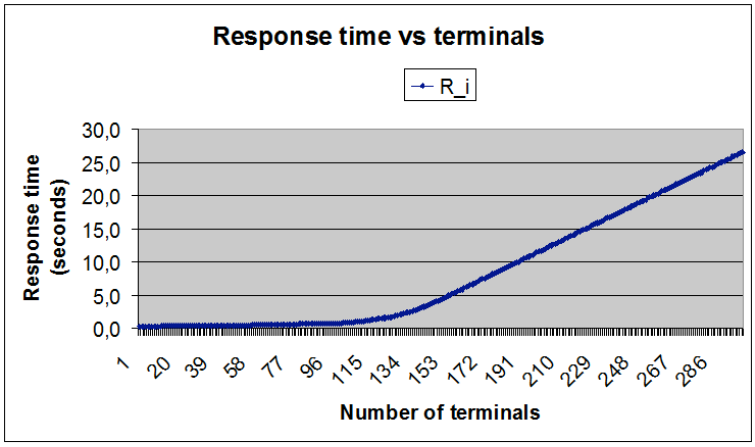
Figure 121: **Exercise 14.4** QN model for the new situation.

cust	X_0	$1/X_0$	U_{cpu}	U_{disk1}	U_{disk2}	Q_{cpu}	Q_{disk1}	Q_{disk2}
0								
1	3.279	0.3050	25.9%	41.0%	33.1%	0.3	0.4	0.3
2	4.863	0.2056	38.4%	60.8%	49.1%	0.5	0.9	0.7
3	5.773	0.1732	45.6%	72.2%	58.3%	0.7	1.4	1.0
4	6.349	0.1575	50.2%	79.4%	64.1%	0.8	2.0	1.2
5	6.737	0.1484	53.2%	84.2%	68.0%	0.9	2.6	1.5
6	7.009	0.1427	55.4%	87.6%	70.8%	1.0	3.2	1.7
7	7.207	0.1388	56.9%	90.1%	72.8%	1.1	4.0	1.9
8	7.354	0.1360	58.1%	91.9%	74.3%	1.2	4.7	2.1
9	7.466	0.1339	59.0%	93.3%	75.4%	1.2	5.5	2.3
10	7.553	0.1324	59.7%	94.4%	76.3%	1.3	6.3	2.4
11	7.621	0.1312	60.2%	95.3%	77.0%	1.3	7.1	2.6
12	7.676	0.1303	60.6%	95.9%	77.5%	1.4	8.0	2.7
13	7.720	0.1295	61.0%	96.5%	78.0%	1.4	8.8	2.8
14	7.756	0.1289	61.3%	96.9%	78.3%	1.4	9.7	2.9
15	7.785	0.1285	61.5%	97.3%	78.6%	1.4	10.6	3.0
16	7.810	0.1280	61.7%	97.6%	78.9%	1.5	11.5	3.0
17	7.831	0.1277	61.9%	97.9%	79.1%	1.5	12.4	3.1
18	7.849	0.1274	62.0%	98.1%	79.3%	1.5	13.3	3.2
19	7.864	0.1272	62.1%	98.3%	79.4%	1.5	14.3	3.2
20	7.877	0.1270	62.2%	98.5%	79.6%	1.5	15.2	3.3

Figure 122: **Exercise 14.4** Results from solving the submodel with 5 processes.

```
Results from solving MVA model using multiclass load dependent algorithm :
--------------------------------------------------------------------------------
    PBD, Exercise 14.4 with 5 processes (exact singleclass) Input parameters
--------------------------------------------------------------------------------
     rname     rtype
    client         D
       FES        LD

  customer     ctype       N_r
       txn    closed       300

 serv demand    client       FES
         txn  18.00000       sd(n)

serv demand[FES]
           0           1           2           3           4           5
     0.00000     0.30497     0.20563     0.17322     0.15751     0.14843

[ld device parameters]
service rate for 1 to 6 are: [0, 3.2789999999999999, 4.8630000000000004,
5.7729999999999997, 6.3490000000000002, 6.7370000000000001]


[Error parameters]
epsilon = 0.00001
--------------------------------------------------------------------------------
                            Prepending output by caller
--------------------------------------------------------------------------------
The maximum number of customers within the response time of 3.0 seconds is 139.
The response time for 139 customers is    2.89879 seconds.
--------------------------------------------------------------------------------
                             Result for 300 customers
--------------------------------------------------------------------------------
      Req Residence time (sec)  Response R-/-Delay      Tput-----Utilization----
      Type    client       FES     (sec)     (sec)   (req/s)    client       FES
       txn  18.00000  26.53021    44.530    26.530      6.74     0.404     0.411
```

Figure 123: **Exercise 14.4** Results from exact single class load dependent algorithm.



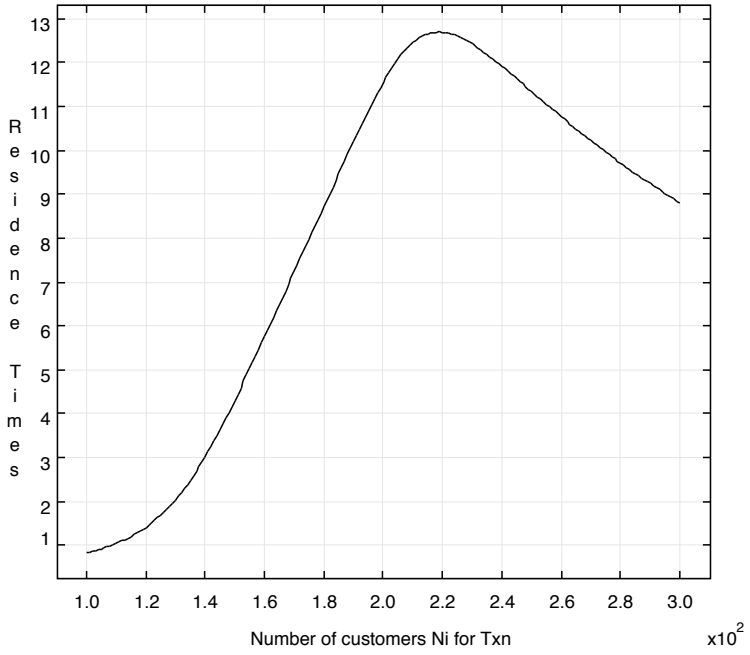


Figure 124: **Exercise 14.4** A graph of response time versus number of users. Remark: the JMT graph appears to be incorrect.

cust	X_0	$1/X_0$	U_{cpu}	U_{disk1}	U_{disk2}	Q_{cpu}	Q_{disk1}	Q_{disk2}
0								
1	3.559	0.2810	28.1%	35.9%	35.9%	0.3	0.4	0.4
2	5.317	0.1881	42.0%	53.7%	53.7%	0.5	0.7	0.7
3	6.357	0.1573	50.2%	64.2%	64.2%	0.8	1.1	1.1
4	7.039	0.1421	55.6%	71.1%	71.1%	1.0	1.5	1.5
5	7.518	0.1330	59.4%	75.9%	75.9%	1.1	1.9	1.9
6	7.871	0.1271	62.2%	79.5%	79.5%	1.3	2.4	2.4
7	8.140	0.1229	64.3%	82.2%	82.2%	1.4	2.8	2.8
8	8.351	0.1197	66.0%	84.3%	84.3%	1.6	3.2	3.2
9	8.521	0.1174	67.3%	86.1%	86.1%	1.7	3.7	3.7
10	8.660	0.1155	68.4%	87.5%	87.5%	1.8	4.1	4.1
11	8.776	0.1140	69.3%	88.6%	88.6%	1.9	4.6	4.6
12	8.873	0.1127	70.1%	89.6%	89.6%	2.0	5.0	5.0
13	8.956	0.1117	70.8%	90.5%	90.5%	2.0	5.5	5.5
14	9.027	0.1108	71.3%	91.2%	91.2%	2.1	5.9	5.9
15	9.090	0.1100	71.8%	91.8%	91.8%	2.2	6.4	6.4
16	9.144	0.1094	72.2%	92.4%	92.4%	2.2	6.9	6.9
17	9.192	0.1088	72.6%	92.8%	92.8%	2.3	7.4	7.4
18	9.235	0.1083	73.0%	93.3%	93.3%	2.3	7.8	7.8
19	9.273	0.1078	73.3%	93.7%	93.7%	2.4	8.3	8.3
20	9.307	0.1074	73.5%	94.0%	94.0%	2.4	8.8	8.8

Figure 125: **Exercise 14.4** Resolving the submodel with a faster disk1

```
 Results from solving MVA model using multiclass load dependent algorithm :
--------------------------------------------------------------------------------
PBD, Exercise 14.4 (faster disk) with 7 processes (exact singleclass) Input parameters
--------------------------------------------------------------------------------
     rname     rtype
    client         D
       FES        LD

  customer     ctype       N_r
       txn    closed       300

 serv demand    client       FES
         txn  18.00000       sd(n)

serv demand[FES]
           0           1           2           3           4           5           6
     0.00000     0.28098     0.18808     0.15731     0.14207     0.13301     0.12705     0.12

[ld device parameters]
service rate for 1 to 8 are:        [0, 3.5590000000000002, 5.3170000000000002, 6.35700000000

[Error parameters]
epsilon = 0.00001
--------------------------------------------------------------------------------
                           Prepending output by caller
--------------------------------------------------------------------------------
The maximum number of customers within the response time of 3.0 seconds is 169.
The response time for 169 customers is    2.94148 seconds.
--------------------------------------------------------------------------------
                            Result for 300 customers
--------------------------------------------------------------------------------
      Req Residence time (sec)  Response R-/-Delay      Tput-----Utilization----
      Type    client       FES     (sec)     (sec)   (req/s)    client       FES
       txn  18.00000  18.85504    36.855    18.855      8.14     0.488     0.327
--------------------------------------------------------------------------------
```

Figure 126: **Exercise 14.4** Result using a faster disk1 and an increase in memory to allow for 7 concurrent processes.

15 Chapter 15

15.1 Exercise 15.1

Given Equations PBD-15.2.4 and PBD-15.2.5

Requested Derive these equations (hint: $\sigma^2 = \overline{x}^2 - (\overline{x})^2$ where $\overline{x}^2$ is the second moment of a random variable and $\overline{x}$ its mean.

Solution The value of S_Γ is equal to the expected value of the service time based on the probabilities for short and long. So

$$S_\Gamma = \sum_{x \in \{S,L\}} x_i P[X = x_i]$$

$$S_\Gamma = p_L S_L + p_S S_S$$

So this shows the derivation of equation PBD-15.2.4. Because S_Γ is also the first moment or mean, we can use S_Γ in calculating the value of CV_Γ. First we need the value of σ^2:

$$\sigma^2 = \overline{x}^2 - (\overline{x})^2 \qquad \text{see hint}$$

$$\sigma^2 = \sum_{i \in \{S,L\}} x_i^2 P[X = x_i] - \left(\sum_{i \in \{S,L\}} x_i P[X = x_i] \right)^2$$

$$\sigma^2 = S_L^2 p_L + S_S^2 p_S - (p_L S_L + p_S S_S)^2$$

Now we can calculate the coefficient of variation as follows:

$$CV_\Gamma^2 = \left(\frac{\sigma}{\mu} \right)^2 = \frac{\sigma^2}{\mu^2}$$

$$CV_\Gamma^2 = \frac{S_L^2 p_L + S_S^2 p_S - (p_L S_L + p_S S_S)^2}{(p_L S_L + p_S S_S)^2}$$

$$CV_\Gamma^2 = \frac{S_L^2 p_L + S_S^2 p_S}{(p_L S_L + p_S S_S)^2} - 1$$

Which is **not right** and does **not** concur with the text of PBD. ⍰

15.2 Exercise 15.2

Given Example PBD-15.3.2

requested

- Use the open multiclass model of Chapter 13 to solve the unconstrained model
- Compte the values of line 1 of Table PBD-15.2

Solution

Queues	Classes		
	Query	Update	Total
CPU	0.91116	0.15496	1.06612
disk1	5.10811	0.64865	5.75676
disk2	0.00000	0.05042	0.05042

Figure 127: **Exercise 15.2** The queue lengths as calculated using the unconstrained classes in `OpenQN.xls`.

Initialization step Using `OpenQN.xls` we solve the queue lengths for the unconstrained classes. The results are depicted in figure 127.
From step 1 of the algorithm proposed by Lazowska and Zahorjan PBD-page 424 ([1]):

$$\overline{n}_c = minimum(J_c, \text{Q-length in unconstrained model})$$

$$\overline{n}_q = minimum(4, 6.0) = 4.0$$

$$\overline{n}_u = minimum(1, 0.85) = 0.85$$

Thus $\overline{n}_c = (4.0, 0.85)$. Using these values in the spreadsheet `ex15.2-ClosedQN-step2.xls`, we resolve the throughputs from step 2 and 3 of the algorithm. The results are shown in figure 128.

Throughput line 1			
n_q	n_u	$X_q(n)$	$X_u(n)$
1	0,85	2,546	
2	0,85	3,581	
3	0,85	4,118	
4	0,85	4,437	
4	1		0,342

Figure 128: **Exercise 15.2** Calculating $X_Q(n_Q)$ and $X_u(n_u)$ from steps 2 and 3 of Lazowska's algorithm

Step 3b to 3d. Following steps 3b and 3c we get a FESC[64] as detailed in figure 129. We use the open class model, because both update and query refer to transactions.

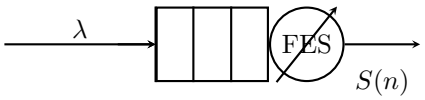


Figure 129: **Exercise 15.2** The reduced model to be used for update and query transactions in step 3c

The algorithm of section PBD-15.3.1 specifies the use of the GBD theorem from Chapter 10 for open classes. So, for both the query and the update class we will use the GBD theorem.

[64] Flow Equivalent Server Center

From the previous step we know the troughputs per number of customers, which we will now insert as the service rate i.e. $\mu(k)$ where k represents the number of customers. The arrival rate is not dependant on the number customers and is known to be 4.2 for query transactions and 0.2 for update transactions.

Solving FESC for Query class From the GBD-theorem we also know the probabilities expressed in $\mu(k)$ and λ:

$$P_k = \left[\sum_{k=0}^{\infty}\prod_{i=0}^{k-1}\frac{\lambda_i}{\mu_{i+1}}\right]^{-1}\prod_{i=0}^{k-1}\frac{\lambda_i}{\mu_{i+1}} \qquad \text{for } k = 0, 1, 2, \ldots$$

Using the results from figure 128 we have the throughputs $\{2.546, 3.581, 4.118, 4.437\}$ for $X(1), X(2), X(3)$ and $X(4)$ respectively. In the FESC model for the query transactions the service rate will be equal to these values with $\mu(n) = \mu(4)$ for $n > 4$. Thus we get:

$$P_0 = \left[\sum_{k=0}^{\infty}\prod_{i=0}^{k-1}\frac{\lambda_i}{\mu_{i+1}}\right]^{-1}$$

$$1/P_0 = \frac{\lambda}{\mu_1} + \frac{\lambda^2}{\mu_1\mu_2} + \frac{\lambda^3}{\mu_1\mu_2\mu_3} + \sum_{k=4}^{\infty}\frac{\lambda^3}{\mu_1\mu_2\mu_3}\left(\frac{\lambda}{\mu_4}\right)^{k-3}$$

$$1/P_0 = \frac{\lambda}{\mu_1} + \frac{\lambda^2}{\mu_1\mu_2} + \frac{\lambda^3}{\mu_1\mu_2\mu_3} + \frac{\lambda^3}{\mu_1\mu_2\mu_3}\sum_{k=4}^{\infty}\left(\frac{\lambda}{\mu_4}\right)^{k-3}$$

Replace $k-3$ by k and observe that $\left(\frac{\lambda}{\mu}\right)^0 = 1$, we get

$$1/P_0 = \frac{\lambda}{\mu_1} + \frac{\lambda^2}{\mu_1\mu_2} + \frac{\lambda^3}{\mu_1\mu_2\mu_3} + \frac{\lambda^3}{\mu_1\mu_2\mu_3}\left(\sum_{k=0}^{\infty}\left(\frac{\lambda}{\mu_4}\right)^{k} - 1\right)$$

Now we use the fact that $\sum_{k=0}^{\infty} x^k$ for $|k| < 1$ is equal to $\frac{1}{1-x}$, and get

$$1/P_0 = \frac{\lambda}{\mu_1} + \frac{\lambda^2}{\mu_1\mu_2} + \frac{\lambda^3}{\mu_1\mu_2\mu_3} + \frac{\lambda^3}{\mu_1\mu_2\mu_3}\left(\frac{1}{1-\frac{\lambda}{\mu_4}} - 1\right)$$

Now we fill in the values for λ and μ_i and get for query:

$$1/P_0 = \frac{4.2}{2.546} + \frac{4.2^2}{2.546\times 3.581} + \frac{4.2^3}{2.546\times 3.581\times 4.118} + \frac{4.2^3}{2.546\times 3.581\times 4.118}\left(\frac{1}{1-\frac{4.2}{4.437}} - 1\right)$$

The value is calculated in spreadsheet `Ch15-Ex15.2-Lazowska-step-3.xls`.

$$1/P_0 \approx 40.5281$$

Now that P_0 is known for query, we can calculate the other values for P_k for $k > 0$ and the queue length for the query transactions.
From equation PBD-15.3.15 on page 422 we have:

$$N_{\text{query}} = \sum_{k=1}^{4} kP_k + 4\times(1 - \sum_{k=0}^{4})$$

$$N_{\text{query}} \approx 3.6350$$

Solving FESC for Update class Using the GBD theorem we can again derive the queue length using a comparable reasoning as with the queue length for the Query class. In the update class only one service rate is available, as only one process will be running. This means that the probability P_0 is obtained as follows:

$$P_0 = \left[\sum_{k=0}^{\infty} \prod_{i=0}^{k-1} \frac{\lambda_i}{\mu_{i+1}} \right]^{-1}$$

$$1/P_0 = \sum_{k=0}^{\infty} \left(\frac{\lambda}{\mu}^k \right)$$

Now, once again using the generating function for $\frac{1}{1-x}$ we get

$$1/P_0 = \frac{1}{1 - \frac{\lambda}{\mu}}$$

Filling in the values $\lambda = 0.2$ and $\mu = 0.342$ we get

$$1/P_0 = \frac{1}{1 - \frac{0.2}{0.342}} - 1 \approx 2.40851$$

Now that we have P_0 we can calculate the queue length in the same way as for the query class, using the equation PBD-15.3.14 on page 422.

$$N_{\text{Update}} = \sum_{k=1}^{1} kP_k + 1 \times (1 - \sum_{k=0}^{1} P_k)$$

$$N_{\text{Update}} \approx 0.585$$

Equation PBD-15.3.14 remark When first looking at the formula, you could wonder why the second component exists, as only J processes are allowed to execute. The point of the second component is, that the probability that J processes will execute is not only dependent on the probability of exactly 1 to J processes arriving, but also on the probability of more processes arriving.

15.3 Exercise 15.3

Given Example of §15.4.1.

Requested Compute the response time if consumer requests have priority over corporate requests at the CPU.

Solution The model will be solved using the SWIC algorithm (stepwise inclusion of classes). The model used for the requested solution reverses the roles of consumer and corporate customers. The consumer will now use the original processor, while the corporate customer will use the shadow processor. Therefore the service demands of the corporate customer will be inflated to cater for the higher priority consumers activities.
The first step is to solve the model with the consumers as the only class. See spreadsheet `Ch15-ex15.3-ClosedQN-consumer-only`. The throughput calculated is about 49.16 requests/sec, leading to a CPU utilization of
$49.16 \times 0.015 \approx 0.73733$.

Inflating the service demands for the corporate customers leads to a service demand of

$$D_{cpu,c}^{\text{shw}} = \frac{D_{cpu,c}}{1 - U_{cpu,p}} = \frac{0.033}{1 - 0.73733} \approx 0.1256$$

We solve the model again using $D_{cpu,x} = 0.015$ and $D_{cpu,c} = 0.1256$ in the spreadsheet `Ch15-ex15.3-ClosedQN-consumer-plus-corp.xls`. The resulting response times are 0.24 seconds for consumers and 7.65 seconds for corporate customers. The throughputs are about 48 transactions/second for consumers and almost 8 transactions/second for corporate users.
Although the consumer class has priority in the CPU, the throughput is still lower than in the single class case, because it does not have priority on the disks. Specifically disk2 causes the throughput to drop a little bit. 0.15 seconds of the total response time of 0.24 seconds is spent in disk2, while the service demand for disk2 is equal to that for the CPU, i.e. 0.015 seconds.

15.4 Exercise 15.4

Given An iterative approach to modeling priorities as presented by Lazowska et al. This requires that one model is created with P shadow CPUs. Initial values for throughput are estimated to be zero. The algorithm is recalculated until the throughput values converge within a given tolerance.

Requested Write a program that implements this algorithm and use it to solve the example from §15.4.2.

Solution Chapter 11 of Lazowska's book "Quantative System Performance" ([1]) numbers priorities from low to high, while PBD describes the priorities from high to low. The algorithm conforms to the description of the PBD. Therefore the formula from algorithm 11.1 in Lazowska's book is replace by :

$$D_{c,CPU_j} = \begin{cases} \frac{D_{c,CPU}}{1 - \sum_{k=1}^{c-1} U_{k,CPU}} & c = j \\ 0 & c \neq j \end{cases}$$

The algorithm is implemented in program PRIO_SCHED.PY. The example being calculated is based on closed classes. The closed case was coded in `mva_closed_approx_schweitzer.py` for the approximate algorithm according to Schweitzer-Bard as described in PBD-chapter13. The results are shown in the subdirectory `output` and the file `pbd-example-15.4.txt`. A summary of the results are printed in figure 130.
As you can see from the summary results in figure 130, the figures are slightly different from the figures calculated in the SWIC algorithm as shown in figure PBD-15.6 on page 433. Especially the throughput for class 4 deviates enough to result in a different utilization and therefore different response time. The deviation are 3.34 and 4.08 sec for class 3 and 4 in the SWIC algorithm versus 3.32 and 4.29 sec in the Lazowska algorithm.
It would be interesting to know which algorithm is more accurate in general or more specifically, which conditions are more favorable for which algorithm.

```
--------------------------------------------------------------------------------
                            Result in 7 iterations
--------------------------------------------------------------------------------
      Req Residence time (sec)  Response R-/-Delay      Tput
      Type       cpu      disk     (sec)     (sec)   (req/s)
    class1   0.13321   3.89168     4.025     4.025     0.745
    class2   0.19481   2.73586     2.931     2.931     1.365
    class3   1.66787   1.65058     3.318     3.318     0.603
    class4   3.77233   0.51777     4.290     4.290     0.233


      Req +--- Util 1 (%) ---+
      Type       cpu      disk
    class1       7.5      37.3
    class2      20.5      47.8
    class3      48.2      12.1
    class4      21.0       1.4
                97.1      98.5
```

Figure 130: **Exercise 15.4** Results from solving example 15.4 with the Lazowska algorithm.

A Math

A.1 $S_n = \sum_{i=0}^{n} x^i$ for $0 \leq x < 1$

The series S_n converges for $|x| < 1$ and $n \to \infty$. For probabilities this sum often occurs in one form or another, especially when using Markov chains.
The sum is solved by:

$$S = S_\infty = \sum_{i=0}^{\infty} x^i = \frac{1}{1-x} \qquad \text{for } 0 \leq x < 1$$

The reason this is true can be found by executing the division of 1 by $1 - x$, which results in the infinite sum S_n. Another way to see this is by multiplying the infinite sum by $1 - x$, which results in 1.

$$(1-x) \times S = (1 + x + x^2 + x^3 + \cdots) - (x + x^2 + x^3 + \cdots) = 1$$

A.2 Application to $u_n = (\frac{1}{2})^n$

According to section A.1 the limit of $S_n = \sum_{i=1}^{n} u_n$ for $n \to \infty$ is equal to 1.
In this particular case one can also see this as follows:

$$\begin{aligned} S_n &= \tfrac{1}{2} + (\tfrac{1}{2})^2 + (\tfrac{1}{2})^3 + (\tfrac{1}{2})^4 + \cdots + (\tfrac{1}{2})^n && \text{(A.2.1)} \\ &= 1 - (\tfrac{1}{2})^n && \text{(A.2.2)} \end{aligned}$$

So the result is:

$$\lim_{n->\infty} S_n = 1 \qquad \square$$

A.3 $S_n = \sum_{n=0}^{\infty} nx^{n-1}$ for $0 \leq x < 1$

This series of S_n also emerges when working with Markov chains, for instance from formula PBD-10.8.5 on page 380, when determining queue length.

$$\text{queue length} = \sum_{k=1}^{\infty} kP_k$$

where P_k is a probability.
The solution to the infinite sum S_n is:

$$S_n = \sum_{k=0}^{\infty} kx^{n-1} = \frac{1}{(1-x)^2}$$

One can see the truth of this statement by determining a derivative from the formula in section A.1 on both sides of the equation. Another way is to divide 1 by $(1-x)^2$ to get the polynomium as in S_n.

A.4 $\sum_{j=a}^{\infty} j\rho^j$

In exercise PBD-14.7 on page 408, the authors provide a hint without proof: $\sum_{j=a}^{\infty} j\rho^j = \rho^a \frac{\rho + a(1-\rho)}{(1-\rho)^2}$ for $\rho < 1$. We start with the definition:

$$\sum_{j=a}^{\infty} j\rho^j = a\rho^a + (a+1)\rho^{a+1} + (a+2)\rho^{a+2} + \cdots$$

First we isolate ρ^a from the rest

$$\sum_{j=a}^{\infty} j\rho^j = \rho^a \left(a + (a+1)\rho + (a+2)\rho^2 + \cdots\right)$$

We can separate this into two infinite sums:

$$\sum_{j=a}^{\infty} j\rho^j = \rho^a \left(a\left(1 + \rho + \rho^2 + \cdots\right) + \left(\rho + 2\rho^2 + 3\rho^3 + \cdots\right)\right)$$

Now use the fact that the first infinite sum is equal to the generating function $\frac{1}{1-\rho}$, and separate the ρ from the second sum, so we get:

$$\sum_{j=a}^{\infty} j\rho^j = \rho^a \left(a\frac{1}{1-\rho} + \rho\left(1 + 2\rho + 3\rho^2 + \cdots\right)\right)$$

The only infinite sum left is now equal to the generating function $\frac{1}{(1-\rho)^2}$:

$$\sum_{j=a}^{\infty} j\rho^j = \rho^a \left(a\frac{1}{1-\rho} + \rho\frac{1}{(1-\rho)^2}\right)$$

multiplying the first term by $\frac{1-\rho}{1-\rho}$

$$\sum_{j=a}^{\infty} j\rho^j = \rho^a \left(\frac{a(1-\rho)}{(1-\rho)^2} + \frac{\rho}{(1-\rho)^2}\right)$$

$$\sum_{j=a}^{\infty} j\rho^j = \rho^a \frac{a(1-\rho)+\rho}{(1-\rho)^2}$$

and we have the expression desired. □

References

[1] Edward D. Lazowska, John Zahorjan, G. Scott Graham, and Kenneth C. Sevcik. *Quantitative System Performance*. Prentice Hall, 1984.

[2] Daniel A. Menasce, Lawrence W. Dowdy, and Virgilio A. F. Almeida. *Performance by Design: Computer Capacity Planning By Example*. Prentice Hall PTR.