

Operating Systems

Exercise 3

2002-06-24

Dominique Emery, s97-602-056

Thomas Bocek, s99-706-319

Philip Iezzi, s99-714-354

Florian Caflisch, s96-920-525

Exercise 3.1

Table 1 → page 2

Table 2 → page 3

table 1:

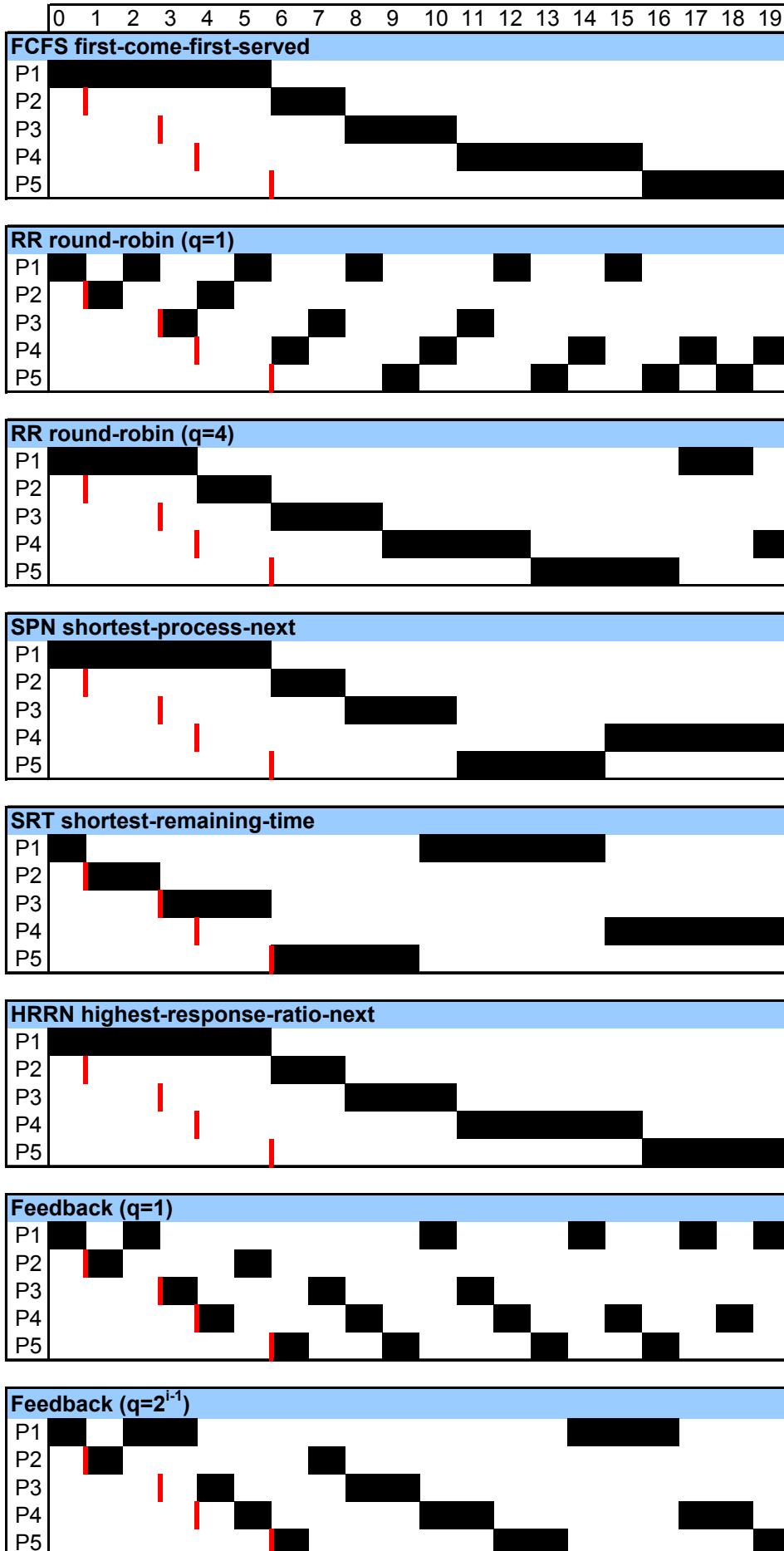


table 2:

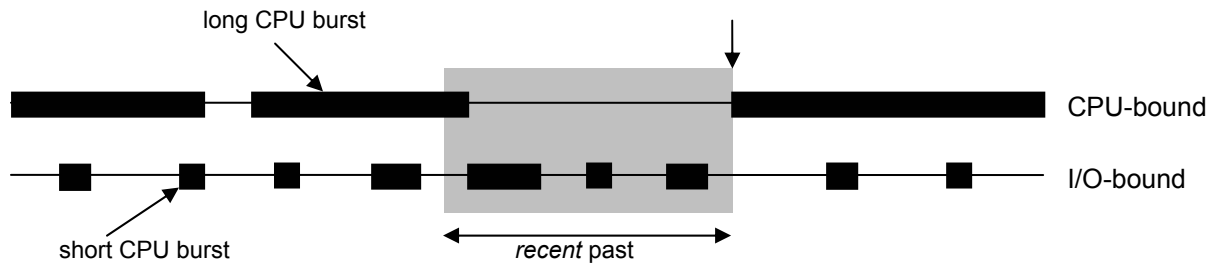
Process	1	2	3	4	5	
Arrival Time	0	1	3	4	6	
Burst Time T_s	6	2	3	5	4	

	1	2	3	4	5	Mean
FCFS first-come-first-served						
F	6	8	11	16	20	12.2
T_U	6	7	8	12	14	9.4
T_U/T_s	1	3.5	2.67	2.4	3.5	2.61
RR round-robin (q=1)						
F	16	5	12	20	19	14.4
T_U	16	4	9	16	13	11.6
T_U/T_s	2.67	2	3	3.2	3.25	2.82
RR round-robin (q=4)						
F	19	6	9	20	17	14.2
T_U	19	5	6	16	11	11.4
T_U/T_s	3.17	2.5	2	3.2	2.75	2.72
SPN shortest-process-next						
F	6	8	11	20	15	12
T_U	6	7	8	16	9	9.2
T_U/T_s	1	3.5	2.67	3.2	2.25	2.52
SRT shortest-remaining-time						
F	15	3	6	20	10	10.8
T_U	15	2	3	16	4	8
T_U/T_s	2.5	1	1	3.2	1	1.74
HRRN highest-response-ratio-next						
F	6	8	11	16	20	12.2
T_U	6	7	8	12	14	9.4
T_U/T_s	1	3.5	2.67	2.4	3.5	2.61
Feedback (q=1)						
F	20	6	12	19	17	14.8
T_U	20	5	9	15	11	12
T_U/T_s	3.33	2.5	3	3	2.75	2.92
Feedback (q=2ⁱ⁻¹)						
F	17	8	10	19	20	14.8
T_U	17	7	7	15	14	12
T_U/T_s	2.83	3.5	2.33	3	3.5	3.03

Exercise 3.2

I/O-bound processes tend to use the CPU for short burst at a time and their I/O-burst time is much longer, so they usually have used less CPU time within the recent past than other processes. So this algorithm favors I/O-bound programs.

However, the algorithm takes into account only recent history. Like this, **CPU-bound processes** will not starve permanently. If a process does not receive time quantum for a while, it becomes one of the processes that had the least CPU time in the recent past and it will get a chance to use CPU.



Exercise 3.3

informal argument:

To get the least average response time it is always the best to let the short processes run first and the long ones at the end. The short ones don't need to "wait" for the long ones and the long ones don't affect the average so much. The proportion for the long ones does not take so much into account.

formal proof:

Let's consider two processes p_1 (burst time t_1) and p_2 (burst time t_2).

Both processes got the same arrival time.

assumption: $t_1 < t_2$

p_1, p_2	p_2, p_1
average time: $avg_{12} = \frac{\frac{t_1}{t_1} + \frac{t_1 + t_2}{t_2}}{2} = \frac{4t_2 + 2t_1}{t_2}$	average time: $avg_{21} = \frac{\frac{t_2}{t_2} + \frac{t_2 + t_1}{t_1}}{2} = \frac{4t_1 + 2t_2}{t_1}$

assumption:

$$avg_{12} < avg_{21}$$

$$\frac{4t_2 + 2t_1}{t_2} < \frac{4t_1 + 2t_2}{t_1}$$

proof:

$$\frac{4t_2 + 2t_1}{t_2} < \frac{4t_1 + 2t_2}{t_1}$$

$$\Rightarrow 4t_2t_1 + 2t_1^2 < 4t_2t_1 + 2t_2^2$$

$$\Rightarrow t_1^2 < t_2^2$$

$$\Rightarrow t_1 < t_2$$

this is also valid for a process sequence with more than 2 processes:

$$(t_x < t_{x+1}) \cap (t_{x+1} < t_{x+2}) \Rightarrow t_x < t_{x+2}$$

Exercise 3.4

There's a couple of reasons where the optimal choice is p=1:

- ✓ All tasks relate on each other. Task_n has to wait for task_{n-1}; task_{n-1} has to wait for task_{n-2},...
- They need to be processed in **serial order**.
With p=1 the whole load will be on node 1 and node 2 will be relieved. Using node 2 in this case makes no sense, as anyway the tasks need to get serialized and cannot run parallel.
- ✓ **Node 2 is not reliable** (e.g. it crashed)
- ✓ Node 2 is just a **mirror** of Node 1, so distributing the tasks onto both nodes doesn't give us any advantage in performance
- ✓ **costs of Node 2** are much higher than costs of Node 1 (e.g. node 2 has been outsourced)

Exercise 3.5

a) remote accesses

		Data Resources				
		R1	R2	R3	R4	R5
	N	2	1	3	3	1
Processes	P1	1	✓		✓	✓
	P2	2		✓		✓
	P3	1	✓	✓	✓	
	P4	2			✓	✓
	P5	3		✓		✓

total: 12 remote accesses

b) optimal g / optimal f

given: $f = \{(1,1), (2,2), (3,1), (4,2), (5,3)\}$		given: $g = \{(1,2), (2,1), (3,3), (4,3), (5,1)\}$																																																																																																							
<table border="1"> <thead> <tr> <th colspan="2"></th> <th colspan="5">Data Resources</th> </tr> <tr> <th colspan="2"></th> <th>N</th> <th>R1</th> <th>R2</th> <th>R3</th> <th>R4</th> <th>R5</th> </tr> </thead> <tbody> <tr> <td rowspan="5">Processes</td> <td>P1</td> <td>1</td> <td>✓</td> <td></td> <td>✓</td> <td>✓</td> <td></td> </tr> <tr> <td>P2</td> <td>2</td> <td></td> <td>✓</td> <td></td> <td>✓</td> <td>✓</td> </tr> <tr> <td>P3</td> <td>1</td> <td>✓</td> <td>✓</td> <td>✓</td> <td></td> <td></td> </tr> <tr> <td>P4</td> <td>2</td> <td></td> <td></td> <td>✓</td> <td></td> <td>✓</td> </tr> <tr> <td>P5</td> <td>3</td> <td></td> <td>✓</td> <td></td> <td>✓</td> <td>✓</td> </tr> </tbody> </table>				Data Resources							N	R1	R2	R3	R4	R5	Processes	P1	1	✓		✓	✓		P2	2		✓		✓	✓	P3	1	✓	✓	✓			P4	2			✓		✓	P5	3		✓		✓	✓	<table border="1"> <thead> <tr> <th colspan="2"></th> <th colspan="5">Data Resources</th> </tr> <tr> <th colspan="2"></th> <th>N</th> <th>R1</th> <th>R2</th> <th>R3</th> <th>R4</th> <th>R5</th> </tr> </thead> <tbody> <tr> <td rowspan="5">Processes</td> <td>P1</td> <td>?</td> <td>✓</td> <td></td> <td>✓</td> <td>✓</td> <td></td> </tr> <tr> <td>P2</td> <td>?</td> <td></td> <td>✓</td> <td></td> <td>✓</td> <td>✓</td> </tr> <tr> <td>P3</td> <td>?</td> <td>✓</td> <td>✓</td> <td>✓</td> <td></td> <td></td> </tr> <tr> <td>P4</td> <td>?</td> <td></td> <td></td> <td>✓</td> <td></td> <td>✓</td> </tr> <tr> <td>P5</td> <td>?</td> <td></td> <td>✓</td> <td></td> <td>✓</td> <td>✓</td> </tr> </tbody> </table>				Data Resources							N	R1	R2	R3	R4	R5	Processes	P1	?	✓		✓	✓		P2	?		✓		✓	✓	P3	?	✓	✓	✓			P4	?			✓		✓	P5	?		✓		✓	✓
		Data Resources																																																																																																							
		N	R1	R2	R3	R4	R5																																																																																																		
Processes	P1	1	✓		✓	✓																																																																																																			
	P2	2		✓		✓	✓																																																																																																		
	P3	1	✓	✓	✓																																																																																																				
	P4	2			✓		✓																																																																																																		
	P5	3		✓		✓	✓																																																																																																		
		Data Resources																																																																																																							
		N	R1	R2	R3	R4	R5																																																																																																		
Processes	P1	?	✓		✓	✓																																																																																																			
	P2	?		✓		✓	✓																																																																																																		
	P3	?	✓	✓	✓																																																																																																				
	P4	?			✓		✓																																																																																																		
	P5	?		✓		✓	✓																																																																																																		
optimal g: $g = \{(1,1), (2,1 2 3), (3,1), (4,1 2 3), (5,2)\}$		optimal f: $f = \{(1,3), (2,1), (3,1 2 3), (4,1 3), (5,1)\}$																																																																																																							
number of remote accesses: <table style="margin-left: auto; margin-right: auto;"> <tr><td>R1</td><td>0</td></tr> <tr><td>R2</td><td>2</td></tr> <tr><td>R3</td><td>1</td></tr> <tr><td>R4</td><td>2</td></tr> <tr><td>R5</td><td>1</td></tr> <tr><td><u>Total</u></td><td><u>6</u></td></tr> </table>		R1	0	R2	2	R3	1	R4	2	R5	1	<u>Total</u>	<u>6</u>	number of remote accesses: <table style="margin-left: auto; margin-right: auto;"> <tr><td>P1</td><td>1</td></tr> <tr><td>P2</td><td>1</td></tr> <tr><td>P3</td><td>2</td></tr> <tr><td>P4</td><td>1</td></tr> <tr><td>P5</td><td>1</td></tr> <tr><td><u>Total</u></td><td><u>6</u></td></tr> </table>		P1	1	P2	1	P3	2	P4	1	P5	1	<u>Total</u>	<u>6</u>																																																																														
R1	0																																																																																																								
R2	2																																																																																																								
R3	1																																																																																																								
R4	2																																																																																																								
R5	1																																																																																																								
<u>Total</u>	<u>6</u>																																																																																																								
P1	1																																																																																																								
P2	1																																																																																																								
P3	2																																																																																																								
P4	1																																																																																																								
P5	1																																																																																																								
<u>Total</u>	<u>6</u>																																																																																																								
For the given process-to-node allocation (f) we chose the optimal node for each resource. We always took the maximum used node from the f-function. At R2 and R4 it doesn't matter which node we take as we cannot prevent the other two remote accesses. Like this we're able to keep remote accesses at the minimum.		For the given resource-to-node allocation (g) we chose the optimal node for each process. We always took the maximum used node from the g-function. At P3, we should use node 1, 2, or 3. At P4 we can chose between node 1 or 3 as both of them are allocated in the resource-to-node function. Like this we're able to keep remote accesses at the minimum.																																																																																																							
comment: decisions with alternatives: Any of the alternatives can be chosen without any impact on the number of remote accesses.																																																																																																									

c1) f given, optimal g (Pseudo-Code):

```
n := amount(P);
m := amount(R);
matrix := mn-requirements matrix; // all requirements are 1, else 0
// f(x) returns the node allocated to process x
// g(x) returns the node allocated to data resource x
for (i =1; i <= n; i++) {
    for (j = 1; j <= m; j++) {
        if (matrix[Pi,Rj]) {
            vote for [Rj,node f(Pi)];
        }
    }
}
```

		Data Resources				
		R1	R2	R3	R4	R5
Processes	N	max vote	max vote	max vote	max vote	max vote
	P1	1		1	1	
	P2	2		2	2	2
	P3	1	1	1		
	P4	2		2		2
	P5	3		3		3

```
for (j = 1; j <= m; j++) {
    g(j) = max_vote([Rj,node]); // chooses the node with the highest vote
}
```

c2) g given, optimal f (Pseudo-Code):

```
n := amount(P);
m := amount(R);
matrix := mn-requirements matrix; // all requirements are 1, else 0
// f(x) returns the node allocated to process x
// g(x) returns the node allocated to data resource x
for (j =1; j <= m; j++) {
    for (i = 1; i <= n; i++) {
        if (matrix[Pi,Rj]) {
            vote for [Pi,node g(Rj)];
        }
    }
}
```

		Data Resources				
		R1	R2	R3	R4	R5
Processes	N	2	1	3	3	1
	P1	max vote	2	3	3	
	P2	max vote		1	3	1
	P3	max vote	2	1	3	
	P4	max vote		3		1
	P5	max vote		1		3

```
for (i = 1; i <= n; i++) {
    f(i) = max_vote([Pi,node]); // chooses the node with the highest vote
}
```

d) optimal f and g with minimal remote accesses

If we would allocate all the processes and all the data resources to the same node, there would only be local accesses and, of course, that would be optimal. :)

Under the assumption that each node gets at least one process, we need to look for another solution...

- 1) create the P-R-requirements matrix respectively create a table and mark every field where process accesses a data resource.
- 2) to start, all processes and all resources are allocated to node 1 (default value)
- 3) traverse through the whole matrix/table and do the following for every marked cell:
Calculate the number of marked cells that would be affected if we move the current process-resource-pair to another node.
Choose the allocation with the smallest amount of affected cells
- 4) iterate through the whole matrix/table and repeat step 3)

		Data Resources				
		R1	R2	R3	R4	R5
	N	2	1	3	1	1
Processes	P1	2	✓	✓	✓	
	P2	1		✓	✓	✓
	P3	1	✓	✓		
	P4	3		✓		✓
	P5	1		✓	✓	✓

total: 5 remote accesses

$$f = \{ (1,2), (2,1), (3,1), (4,3), (5,1) \}$$

$$g = \{ (1,2), (2,1), (3,3), (4,1), (5,1) \}$$

e) optimal f and g for any given requirements-matrix and number of nodes

Follow the instructions in d) 1-4

Here we need to assure that no node gets more than x processes (x: desired amount of distribution). Iterate through the whole matrix/table until the process count of node 1 equals x. Also, we do the allocation with the smallest amount of affected cells to a node whose process count is less than x.