# Opdracht P

Joery van den Hoff

Ik heb gekozen voor maximum Tq/Ts als criteria. Als de maximum Tq/Ts laag is dan hebben alle processen eerlijk tijd gekregen. Is het maximum hoog dan is er ten minste een proces dat niet eerlijk tijd heeft gekregen.

Wat er ontbreekt aan de Java applet is dat er geen overhead is. Als er een proces geladen moet worden, dan kost dat ook processor tijd.

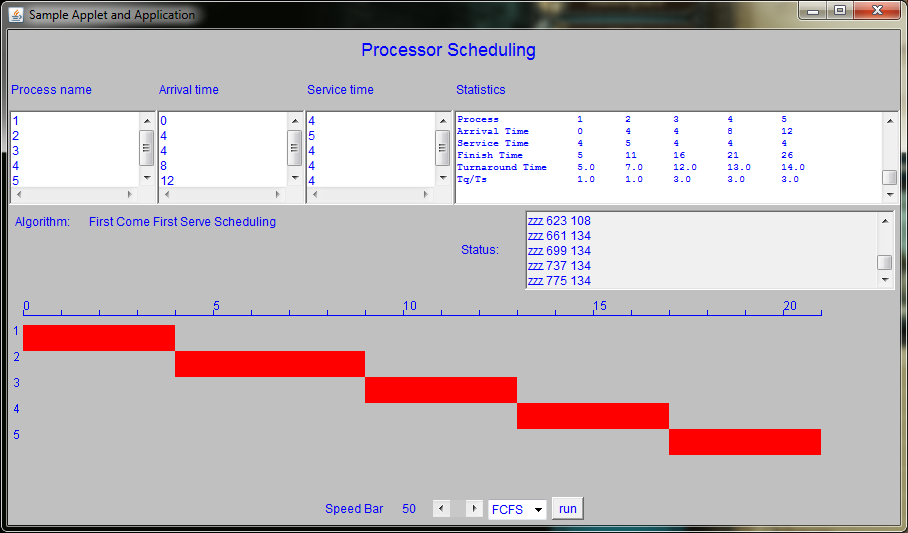
De applet is enorme spaghetti code, maar ik heb overhead toegevoegd door de volgende statements aan te roepen op de juiste plaats (als er een proces in de cpu geladen wordt):

* super.P.FinishTime++;
* super.clock++;

Helaas is het niet mogelijk om overhead in kleinere stappen te simuleren met de applet dus is de overhead 1 per switch. In de werkelijkheid is de overhead veel lager, maar dit is wel handig om verschillende algoritmes te vergelijken. Ik moet dus kiezen voor een versie van de applet met of zonder overhead per situatie om een duidelijke winnaar / verliezer te hebben. Rode blokjes betekent met overhead, blauw betekent zonder.

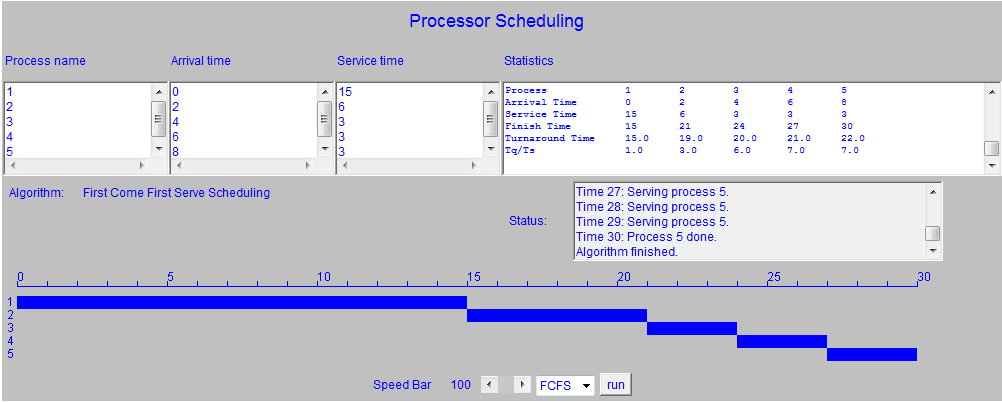
## FCFS

## Winnaar



|  |  |
| --- | --- |
| Algoritme | Maximum Tq/Ts |
| **FCFS** | **3.5** |
| RR1 | 7.5 |
| RR4 | 4.6 |
| SPN | 4.4 |
| SRT | 4.4 |

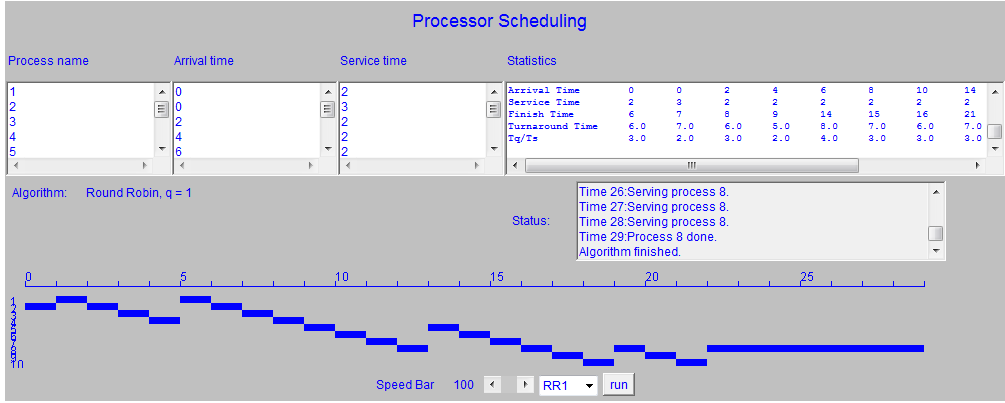
## Verliezer

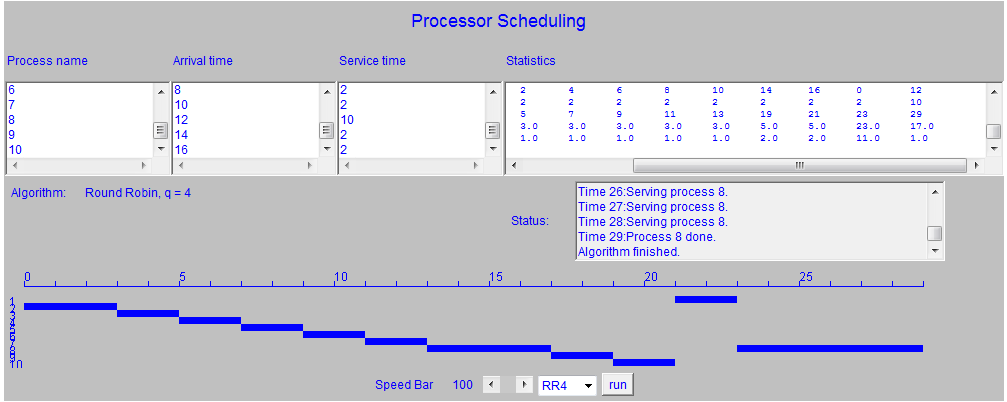


|  |  |
| --- | --- |
| Algoritme | Maximum Tq/Ts |
| **FCFS** | **7.3** |
| RR1 | 4.0 |
| RR4 | 3.5 |
| SPN | 5.3 |
| SRT | 2.5 |

## RR1

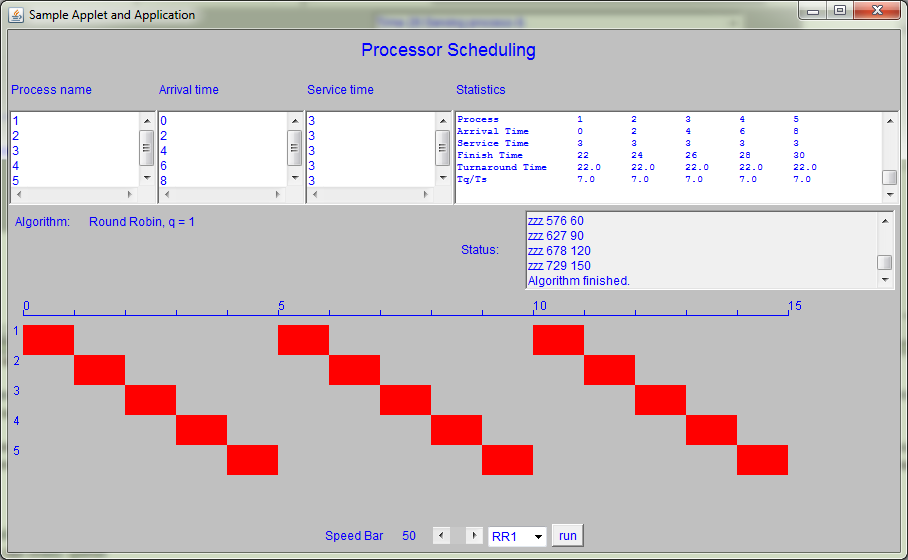
## Winnaar





|  |  |
| --- | --- |
| Algoritme | Maximum Tq/Ts |
| FCFS | 6.5 |
| **RR1** | **4.0** |
| RR4 | 11.5 |
| SPN | 5.0 |
| SRT | 5.0 |

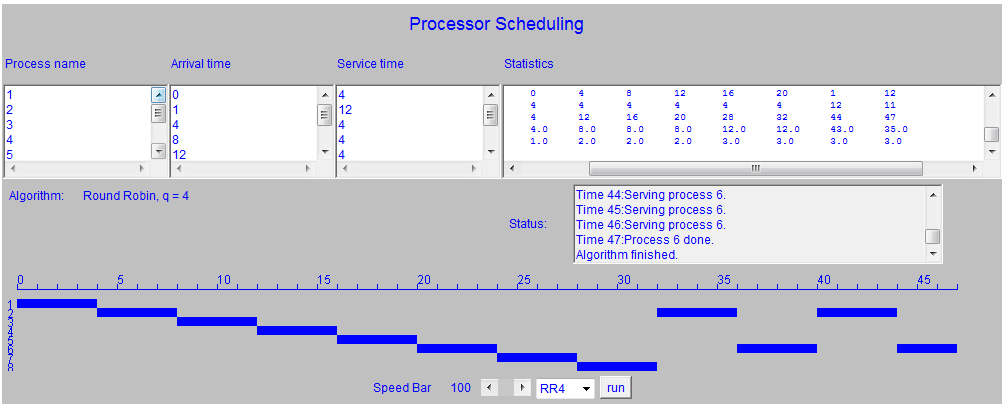
## Verliezer



|  |  |
| --- | --- |
| Algoritme | Maximum Tq/Ts |
| FCFS | 4.0 |
| **RR1** | **7.0** |
| RR4 | 4.0 |
| SPN | 4.0 |
| SRT | 4.0 |

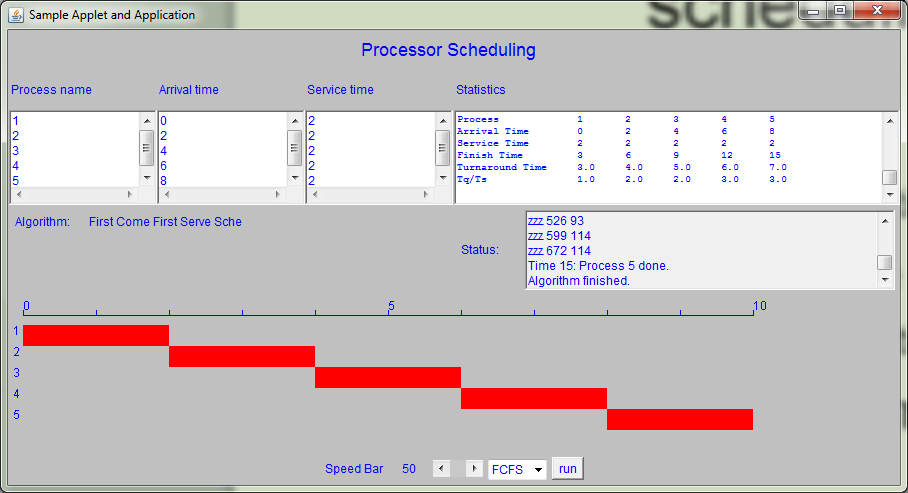
## RR4

## Winnaar



|  |  |
| --- | --- |
| Algoritme | Maximum Tq/Ts |
| FCFS | 6.8 |
| RR1 | 5.0 |
| **RR4** | **3.6** |
| SPN | 3.8 |
| SRT | 3.8 |

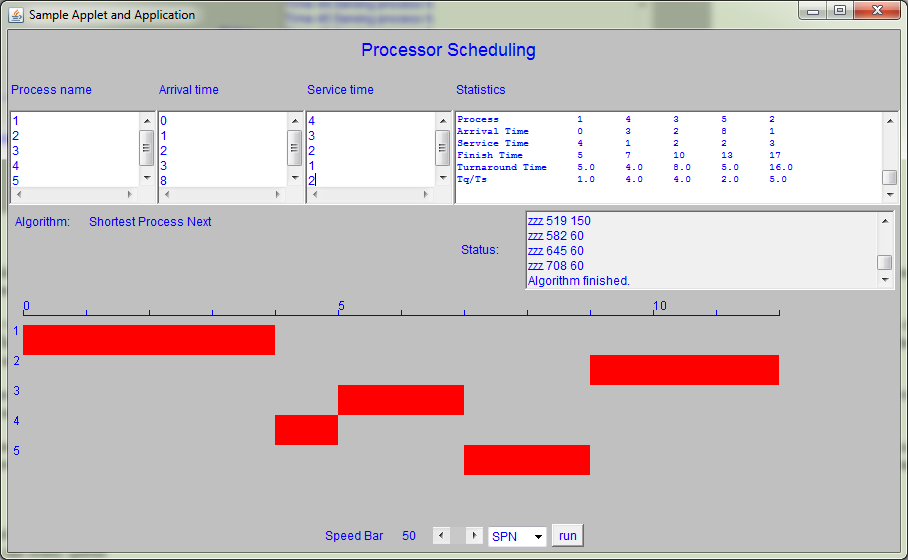
## Verliezer



|  |  |
| --- | --- |
| Algoritme | Maximum Tq/Ts |
| FCFS | 3.5 |
| RR1 | 6.0 |
| **RR4** | **7.0** |
| SPN | 3.8 |
| SRT | 4.3 |

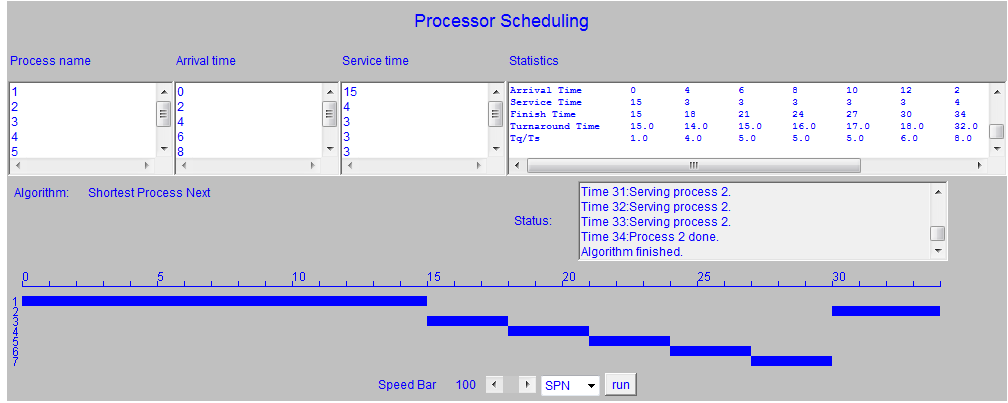
## SPN

## Winnaar



|  |  |
| --- | --- |
| Algoritme | Maximum Tq/Ts |
| FCFS | 11.0 |
| RR1 | 7.0 |
| RR4 | 11.0 |
| **SPN** | **5.3** |
| SRT | 5.7 |

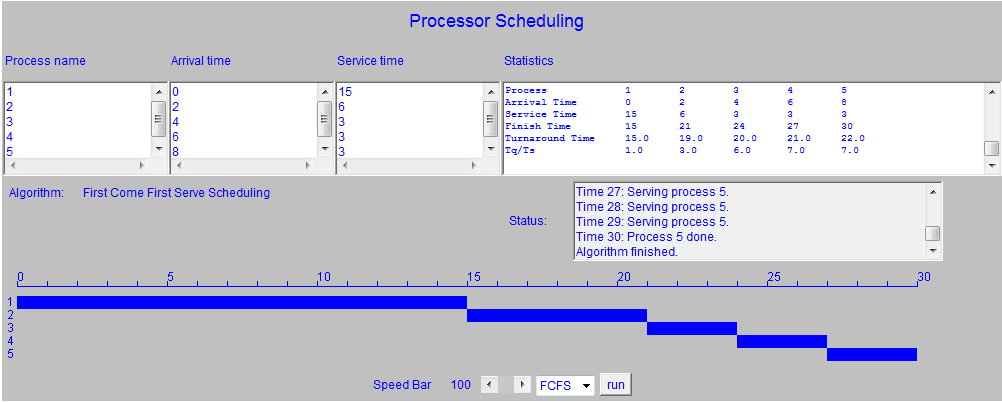
## Verliezer



|  |  |
| --- | --- |
| Algoritme | Maximum Tq/Ts |
| FCFS | 7.3 |
| RR1 | 5.0 |
| RR4 | 3.7 |
| **SPN** | **8.0** |
| SRT | 3.0 |

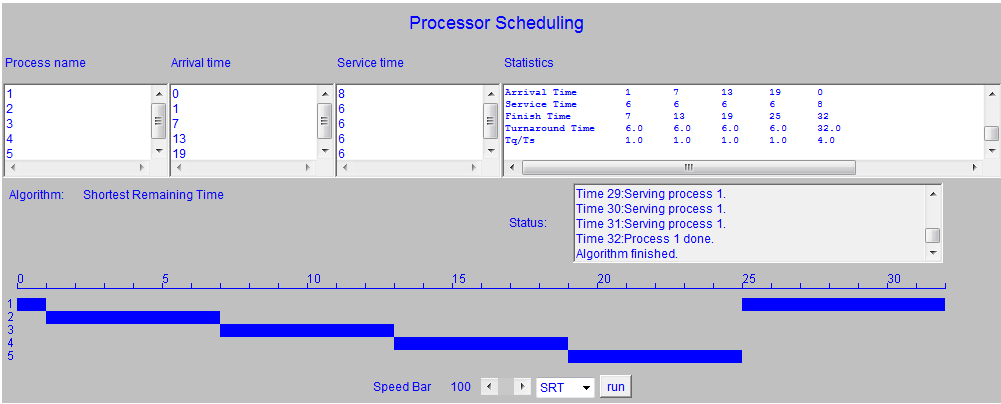
## SRT

## Winnaar



|  |  |
| --- | --- |
| Algoritme | Maximum Tq/Ts |
| FCFS | 7.3 |
| RR1 | 4.0 |
| RR4 | 3.5 |
| SPN | 5.3 |
| **SRT** | **2.5** |

## Verliezer



|  |  |
| --- | --- |
| Algoritme | Maximum Tq/Ts |
| FCFS | 2.2 |
| RR1 | 3.0 |
| RR4 | 2.8 |
| SPN | 2.2 |
| **SRT** | **4.0** |

## Samenvatting

Om het samen te vatten en aan te tonen dat ik het echt snap, hieronder een tabel met de zwakte per algoritme.

|  |  |
| --- | --- |
| Algoritme | Zwakte |
| FCFS | Korte processen moeten lang wachten (op lange) |
| RR1 | Overhead |
| RR4 | Overhead + lang wachten als je nog klein beetje moet |
| SPN | Uithongering van grotere processen |
| SRT | Uithongering van grotere processen + overhead als er veel kleine processen komen |