Selfish Round Robin (SRR)

This gives a better service to processes that have been executing for a while

Processes in the ready list are split into two lists new and accepted

New processes wait while accepted processes are serviced.

The priority of a new process will increase by at a rate a.

The priority of an accepted process will increase by a rate b.

Both a and b are parameters and can be adjusted to tune the method

When the priority of a new process reaches that of an accepted process that new process becomes accepted

Selfish Round Robin (SRR) example

Assume there are no ready processes.

Any new process will be allocated priority of 0.

Let a = 2 and b = 1

So any process in the new queue will increment by 2 (value of a) until it catches up with a process in the accepted queue and then will subsequently be shifted to that queue

After each unit time (q) a process in the accepted queue will increment by the value of the parameter b i.e. 1

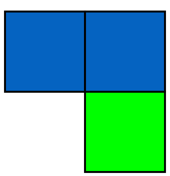
Once in the accepted queue then we follow the Round Robin scheduling mechanism let us set for our example q = 1

Process name	Arrival Time	Priorit ies	Service required	Start time	Finish time	T	W	P
A	0	0	3	0	3	3	0	1.0
B	1		5	3	9	8	3	1.6
C	3		2	7	10	7	5	3.5
D	9		5	10	15	6	1	1.2
E	12		5	15	20	8	3	1.6
Mean						6.4	2.4	1.78

Process A is added straight to the accepted list and is initialised with the value of 0

After 1 quantum unit of time A is incremented by 1 and also B arrives which is assigned to 0 and placed in the new list

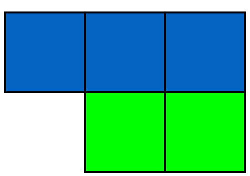
Process name	Arrival Time	Priorities	Service required	Start time	Finish time	T	W	P
A	0	1	3	0	3	3	0	1.0
B	1	0	5	3	9	8	3	1.6
C	3		2	7	10	7	5	3.5
D	9		5	10	15	6	1	1.2
E	12		5	15	20	8	3	1.6
Mean						6.4	2.4	1.78



After the 2nd quantum of time A's priority increments by 1 and B by 2.

B now moves to the back of the accepted list so A runs again for 1 quanta

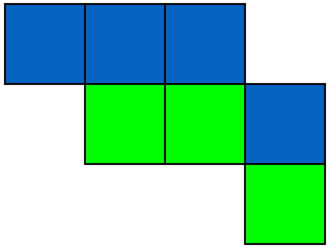
Process name	Arrival Time	Priorities	Service required	Start time	Finish time	T	W	P
A	0	2	3	0	3	3	0	1.0
B	1	2	5	3	9	8	3	1.6
C	3		2	7	10	7	5	3.5
D	9		5	10	15	6	1	1.2
E	12		5	15	20	8	3	1.6
Mean						6.4	2.4	1.78



Process C now arrives and is given the priority of 0 and placed in the new list

A completes and is removed from queue, so B is now in the front of the accepted list

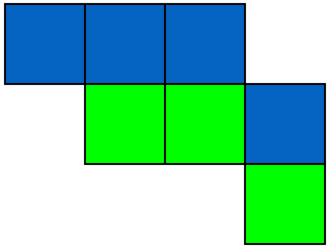
Process name	Arrival Time	Priorit ies	Service required	Start time	Finish time	T	W	P
A	0	------	3	0	3	3	0	1.0
B	1	3	5	3	9	8	3	1.6
C	3	0	2	7	10	7	5	3.5
D	9		5	10	15	6	1	1.2
E	12		5	15	20	8	3	1.6
Mean						6.4	2.4	1.78



Process C now arrives and is given the priority of 0 and placed in the new list

A completes and is removed from queue, so B is now in the front of the accepted list

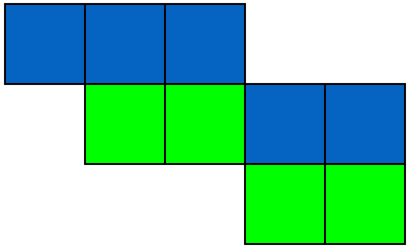
Process name	Arrival Time	Priorit ies	Service required	Start time	Finish time	T	W	P
A	0	------	3	0	3	3	0	1.0
B	1	3	5	3	9	8	3	1.6
C	3	0	2	7	10	7	5	3.5
D	9		5	10	15	6	1	1.2
E	12		5	15	20	8	3	1.6
Mean						6.4	2.4	1.78



For the next quantum unit of time B increments in the accepted list by 1 to 4 and as it's the only process will run next

C in the new list increments its priority by 2

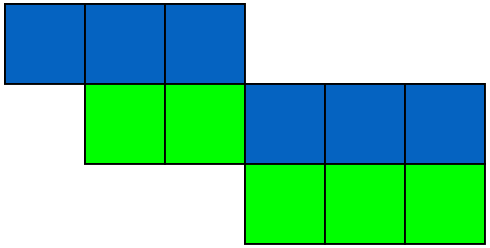
Process name	Arrival Time	Priorit ies	Service required	Start time	Finish time	T	W	P
A	0	------	3	0	3	3	0	1.0
B	1	4	5	3	9	8	3	1.6
C	3	2	2	7	10	7	5	3.5
D	9		5	10	15	6	1	1.2
E	12		5	15	20	8	3	1.6
Mean						6.4	2.4	1.78



C has a priority of 2 and B has a priority of 4

C stays in the new list and so for the next quanta B will run and increment by 1 and C will increment by 2

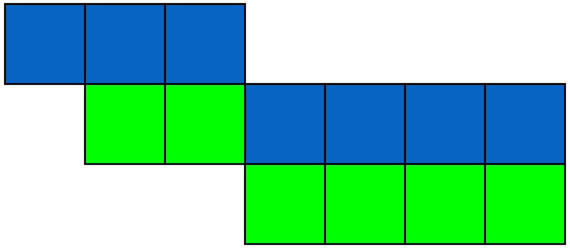
Process name	Arrival Time	Priorities	Service required	Start time	Finish time	T	W	P
A	0	------	3	0	3	3	0	1.0
B	1	5	5	3	9	8	3	1.6
C	3	4	2	7	10	7	5	3.5
D	9		5	10	15	6	1	1.2
E	12		5	15	20	8	3	1.6
Mean						6.4	2.4	1.78



C has a priority of 4 and B has a priority of 5

C stays in the new list and so for the next quanta B will run and increment by 1 and C will increment by 2

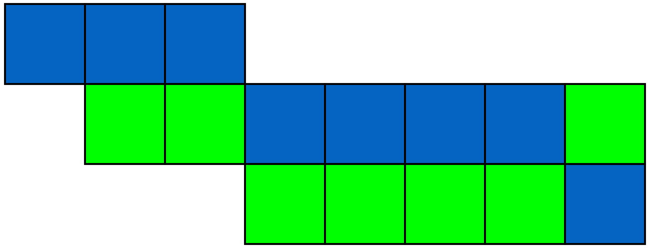
Process name	Arrival Time	Priorit ies	Service required	Start time	Finish time	T	W	P
A	0	------	3	0	3	3	0	1.0
B	1	6	5	3	9	8	3	1.6
C	3	6	2	7	10	7	5	3.5
D	9		5	10	15	6	1	1.2
E	12		5	15	20	8	3	1.6
Mean						6.4	2.4	1.78



C now has a priority value of 6 equal to that of B so moves to the accepted list.

B has just run and is placed in the back of the accepted list and so C is in the front and will run for 1 quanta

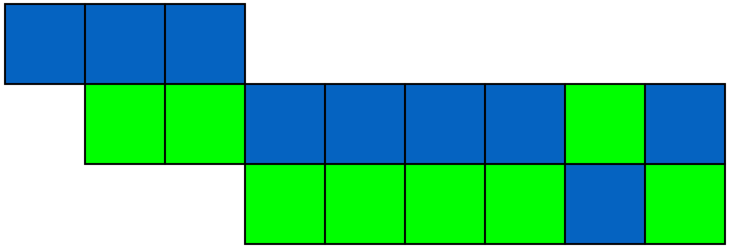
Process name	Arrival Time	Priorities	Service required	Start time	Finish time	T	W	P
A	0	------	3	0	3	3	0	1.0
B	1	7	5	3	9	8	3	1.6
C	3	7	2	7	10	7	5	3.5
D	9		5	10	15	6	1	1.2
E	12		5	15	20	8	3	1.6
Mean						6.4	2.4	1.78



Processes B and C priorities incremented by 1

Process C is now swapped after 1 quanta and placed in the back of the accepted list and B now is placed in the running state

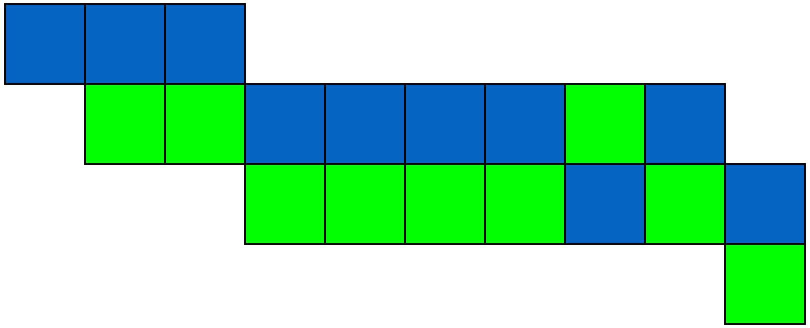
Process name	Arrival Time	Priorit ies	Service required	Start time	Finish time	T	W	P
A	0	------	3	0	3	3	0	1.0
B	1	8	5	3	9	8	3	1.6
C	3	8	2	7	10	7	5	3.5
D	9		5	10	15	6	1	1.2
E	12		5	15	20	8	3	1.6
Mean						6.4	2.4	1.78



B now completes and is removed leaving C in the accepted list so runs next

Process D arrives and is given a priority of 0 and placed in the new list

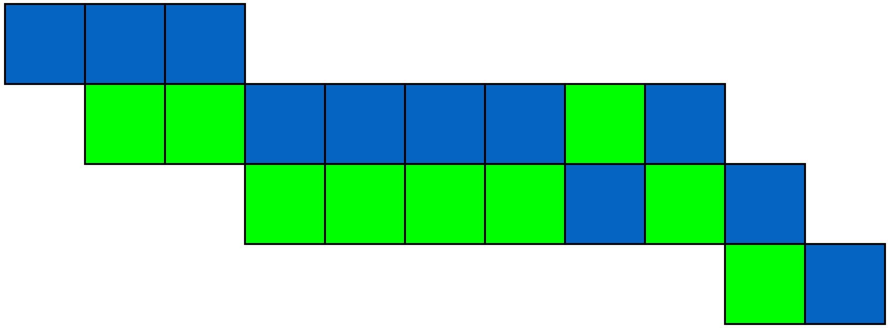
Process name	Arrival Time	Priorit ies	Service required	Start time	Finish time	T	W	P
A	0	------	3	0	3	3	0	1.0
B	1	------	5	3	9	8	3	1.6
C	3	9	2	7	10	7	5	3.5
D	9	0	5	10	15	6	1	1.2
E	12		5	15	20	8	3	1.6
Mean						6.4	2.4	1.78



C now completes and is removed from the list

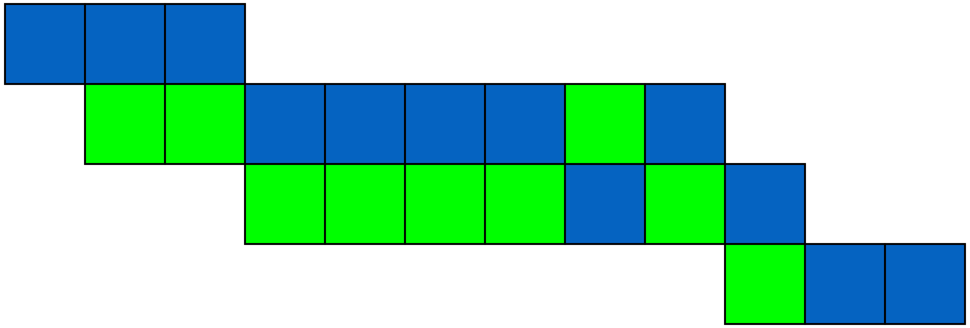
D's priority is incremented by 2 and moves to the accepted list and into the running state

Process name	Arrival Time	Priorit ies	Service required	Start time	Finish time	T	W	P
A	0	------	3	0	3	3	0	1.0
B	1	------	5	3	9	8	3	1.6
C	3	------	2	7	10	7	5	3.5
D	9	2	5	10	15	6	1	1.2
E	12		5	15	20	8	3	1.6
Mean						6.4	2.4	1.78



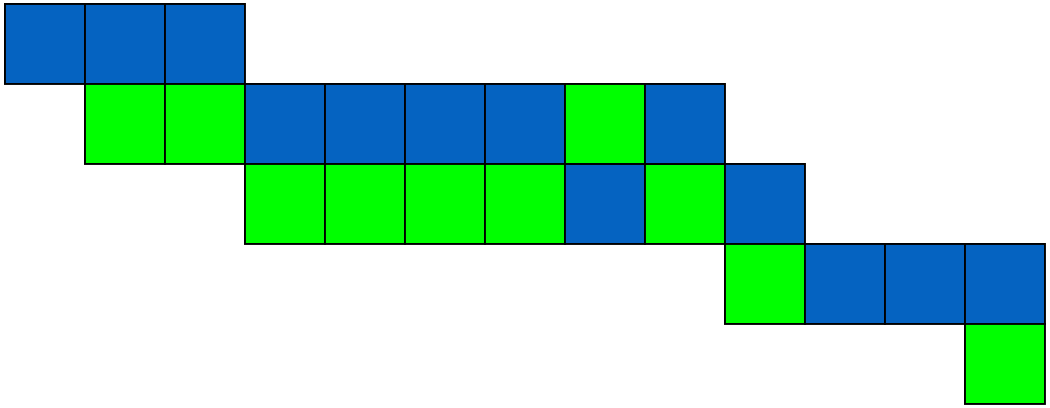
D is now incremented by 1

Process name	Arrival Time	Priorit ies	Service required	Start time	Finish time	T	W	P
A	0	------	3	0	3	3	0	1.0
B	1	------	5	3	9	8	3	1.6
C	3	------	2	7	10	7	5	3.5
D	9	3	5	10	15	6	1	1.2
E	12		5	15	20	8	3	1.6
Mean						6.4	2.4	1.78



For the next quanta D increments by 1 and as it's the only process in the accepted list runs Process E arrives and is given a priority of 0 and placed in the new list

Process name	Arrival Time	Priorit ies	Service required	Start time	Finish time	T	W	P
A	0	------	3	0	3	3	0	1.0
B	1	------	5	3	9	8	3	1.6
C	3	------	2	7	10	7	5	3.5
D	9	4	5	10	15	6	1	1.2
E	12	0	5	15	20	8	3	1.6
Mean						6.4	2.4	1.78

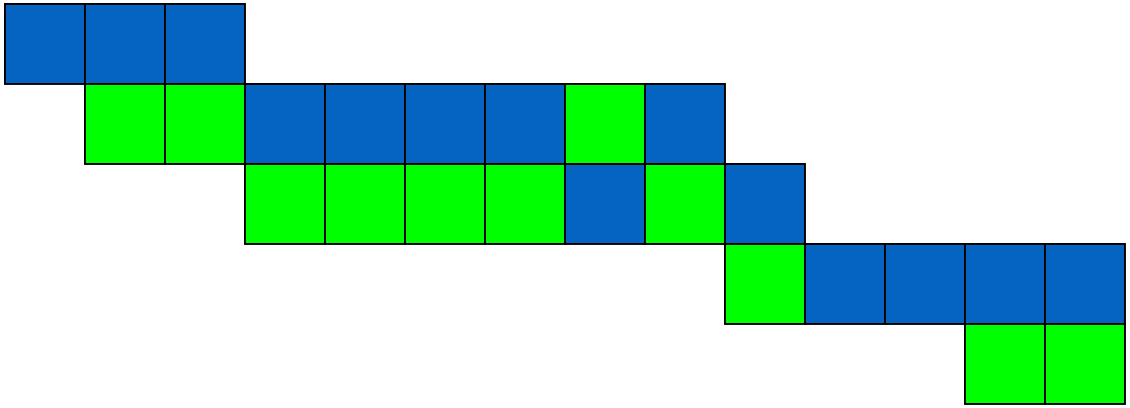


D's priority increments by 1 to 5

E's priority increments by 2 to 2

D runs

Process name	Arrival Time	Priorit ies	Service required	Start time	Finish time	T	W	P
A	0	------	3	0	3	3	0	1.0
B	1	------	5	3	9	8	3	1.6
C	3	------	2	7	10	7	5	3.5
D	9	5	5	10	15	6	1	1.2
E	12	2	5	15	20	8	3	1.6
Mean						6.4	2.4	1.78

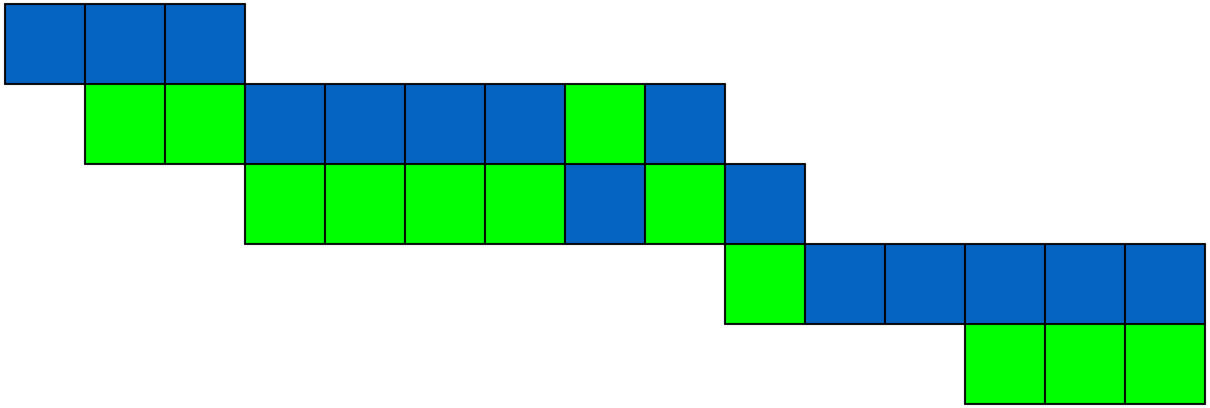


D's priority increments by 1 to 6

E's priority increments by 2 to 4

D runs

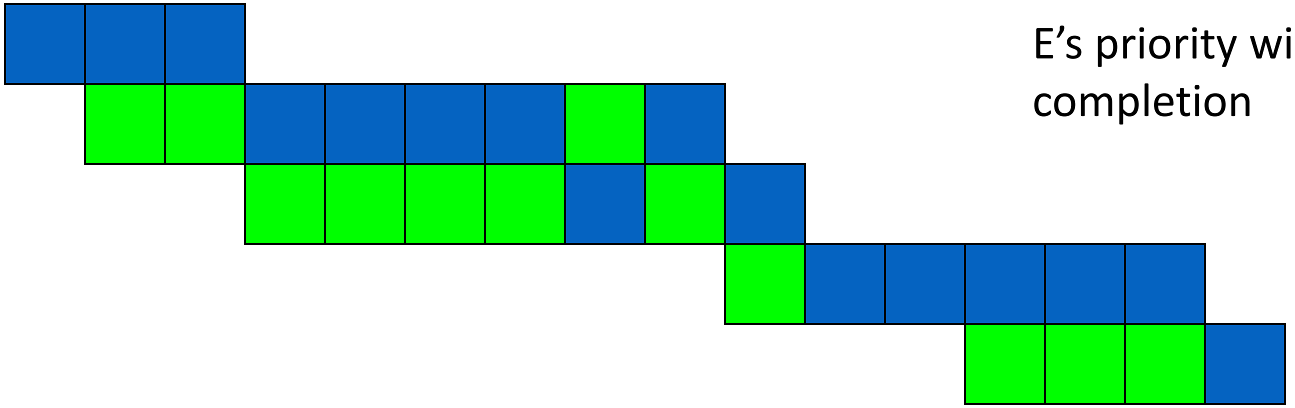
Process name	Arrival Time	Priorities	Service required	Start time	Finish time	T	W	P
A	0	------	3	0	3	3	0	1.0
B	1	------	5	3	9	8	3	1.6
C	3	------	2	7	10	7	5	3.5
D	9	6	5	10	15	6	1	1.2
E	12	4	5	15	20	8	3	1.6
Mean						6.4	2.4	1.78



D completes

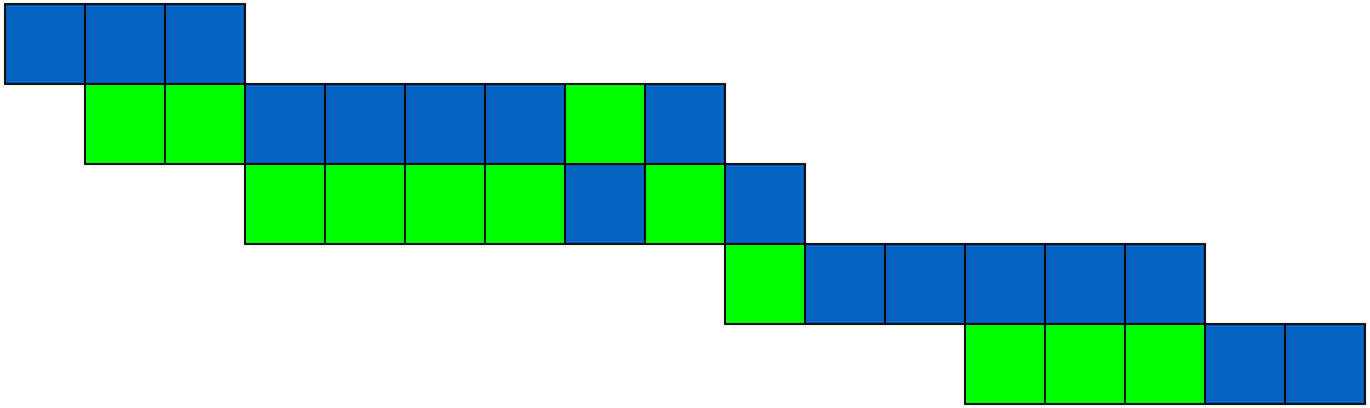
E priority increments by 2 and placed in the accepted list

Process name	Arrival Time	Priorit ies	Service required	Start time	Finish time	T	W	P
A	0	------	3	0	3	3	0	1.0
B	1	------	5	3	9	8	3	1.6
C	3	------	2	7	10	7	5	3.5
D	9	------	5	10	15	6	1	1.2
E	12	6	5	15	20	8	3	1.6
Mean						6.4	2.4	1.78

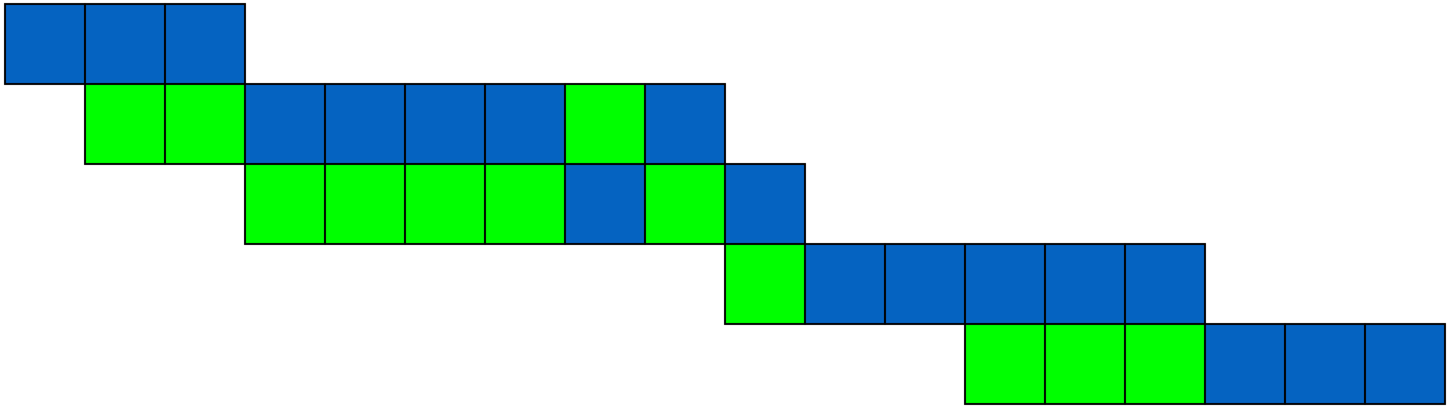


E’s priority will increment by 1 until completion

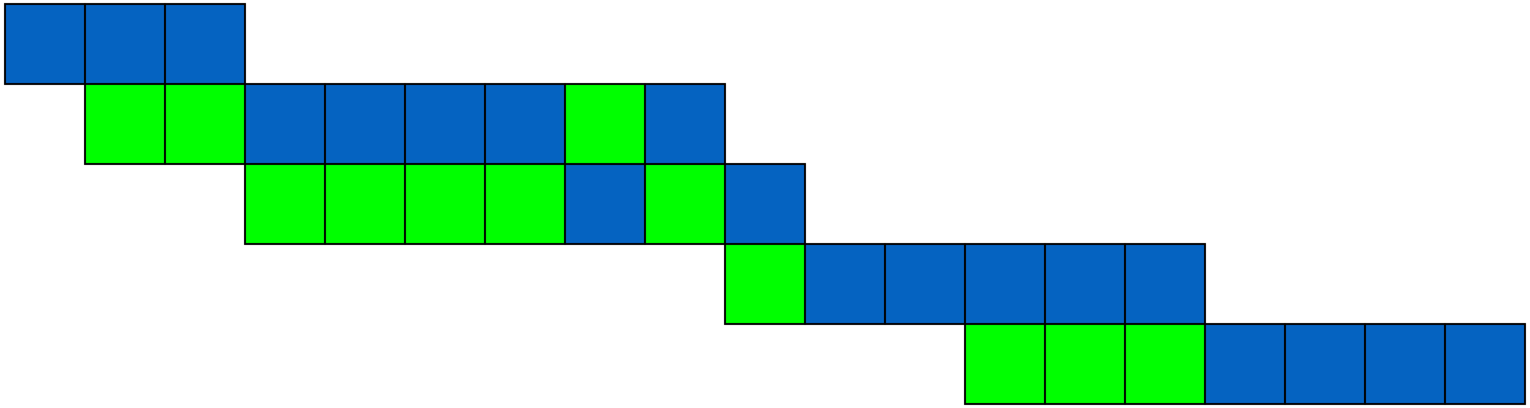
Process name	Arrival Time	Priorities	Service required	Start time	Finish time	T	W	P
A	0	------	3	0	3	3	0	1.0
B	1	------	5	3	9	8	3	1.6
C	3	------	2	7	10	7	5	3.5
D	9	------	5	10	15	6	1	1.2
E	12	7	5	15	20	8	3	1.6
Mean						6.4	2.4	1.78



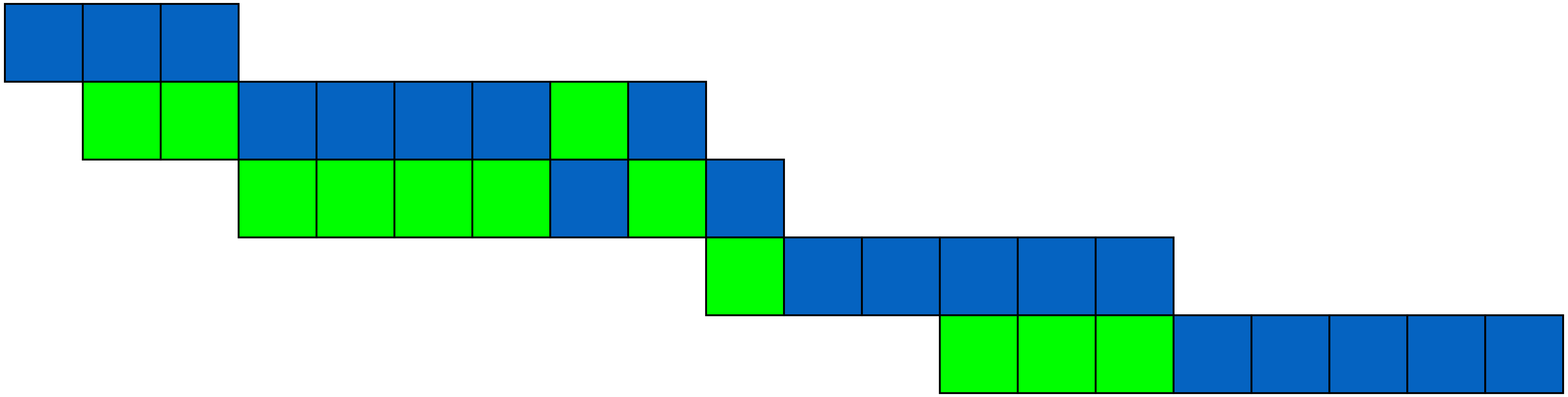
Process name	Arrival Time	Priorit ies	Service required	Start time	Finish time	T	W	P
A	0	------	3	0	3	3	0	1.0
B	1	------	5	3	9	8	3	1.6
C	3	------	2	7	10	7	5	3.5
D	9	------	5	10	15	6	1	1.2
E	12	8	5	15	20	8	3	1.6
Mean						6.4	2.4	1.78



Process name	Arrival Time	Priorit ies	Service required	Start time	Finish time	T	W	P
A	0	------	3	0	3	3	0	1.0
B	1	------	5	3	9	8	3	1.6
C	3	------	2	7	10	7	5	3.5
D	9	------	5	10	15	6	1	1.2
E	12	9	5	15	20	8	3	1.6
Mean						6.4	2.4	1.78



Process name	Arrival Time	Priorities	Service required	Start time	Finish time	T	W	P
A	0	------	3	0	3	3	0	1.0
B	1	------	5	3	9	8	3	1.6
C	3	------	2	7	10	7	5	3.5
D	9	------	5	10	15	6	1	1.2
E	12	10	5	15	20	8	3	1.6
Mean						6.4	2.4	1.78



In summary

Adjusting the relative values of a and b will greatly affect the behaviour of SRR

e.g. if $b/a >= 1$ then a new process will never be accepted until all the existing accepted processes are completed so becomes FCFS

If $b/a = 0$ all processes are accepted immediately so becomes RR

If $0 < b/a < 1$, accepted processes are selfish, but not completely

Multiple-Processor Scheduling

- CPU scheduling more complex when multiple CPUs available
- *Homogeneous processors*
- *Symmetric multiproccessing* – load sharing
- *Asymmetric multiprocessing* – only one processor accesses system data structures, no need for data sharing

Which to choose?

Depends on

System workload

Hardware support for dispatcher

Relative weighting of performance criteria
response time, CPU utilisation,
throughput