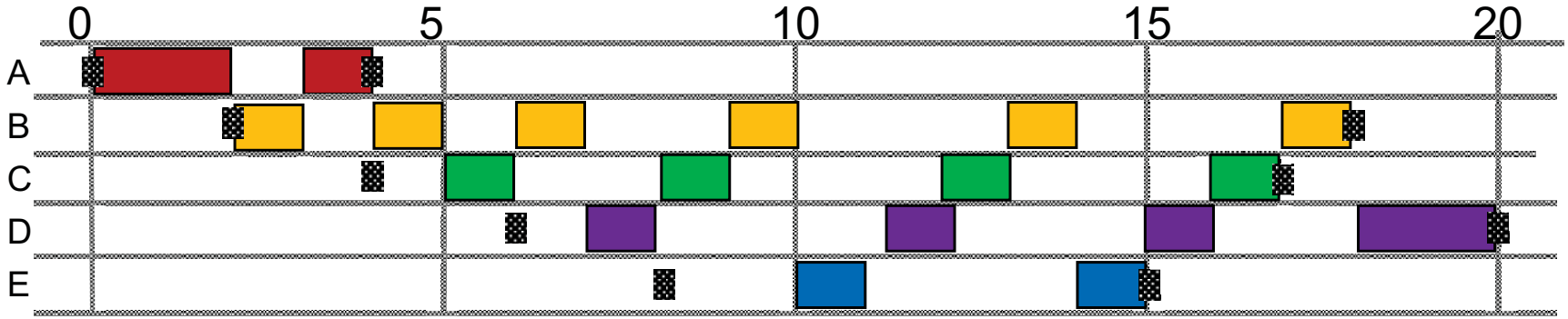
Round Robin

- Stop monopolization of length job by preemption
- In **Round Robin scheduling**, (RR) or **time slicing**, a clock regularly interrupts and pre-empts the running process.
- The next process to run is chosen by an FCFS strategy.
- The main parameter is the **length of the time slice**.
  - A short time slice ensures that brief processes move through the system rapidly,
  - but means many interrupts.
- What happens when time slice becomes infinite?

RR - Example

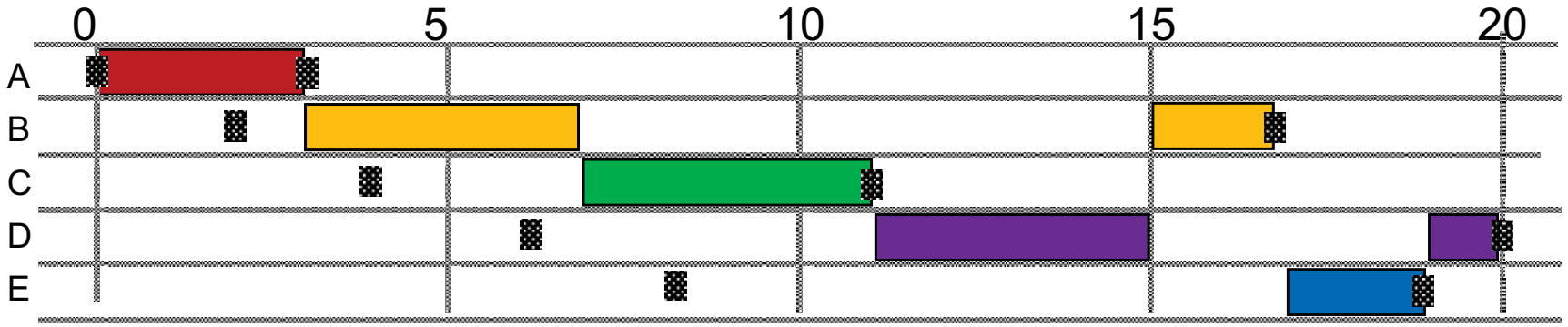
- Example for time slice of 1 unit:



Process	A	B	C	D	E	
Arrival Time	0	2	4	6	8	
Service Time (T_s)	3	6	4	5	2	Mean
RR $q = 1$						
Finish Time	4	18	17	20	15	
Turnaround Time (T_r)	4	16	13	14	7	10.80
T_r/T_s	1.33	2.67	3.25	2.80	3.50	2.71

RR - Example

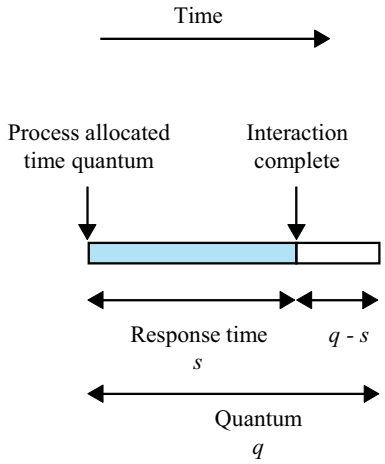
- Example with time slice of 4 units:



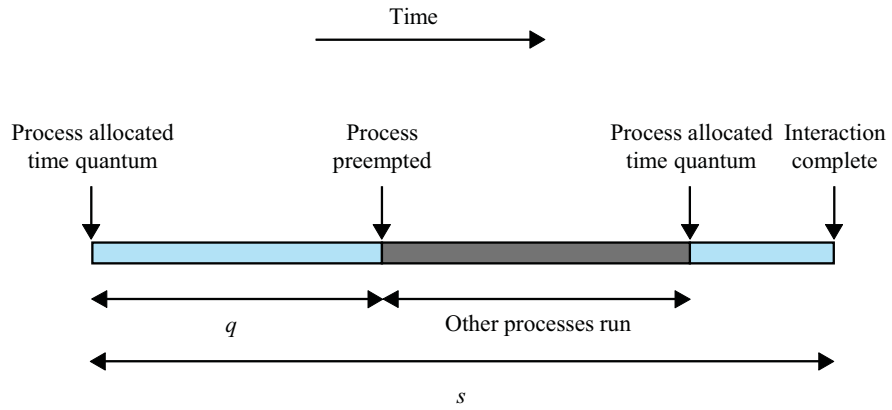
Process	A	B	C	D	E	
Arrival Time	0	2	4	6	8	
Service Time (T_S)	3	6	4	5	2	Mean
RR q = 4						
Finish Time	3	17	11	20	19	
Turnaround Time (T_r)	3	15	7	14	11	10.00
T_r/T_S	1.00	2.5	1.75	2.80	5.50	2.71

Effect of preemption time quantum

- The time slice may be chosen to be a little **larger than the time required for a typical interaction**; that is, most processes should require only one time slice.



(a) Time quantum greater than typical interaction



(b) Time quantum less than typical interaction

THE UNIVERSITY OF NEWCASTLE AUSTRALIA

Round Robin Performance

- Round Robin is effective for general purpose time-sharing systems.
- Each context switch has the OS using the CPU instead of the user process
  - give up CPU, save all info, reload w/ status of incoming process
  - Say 20 ms quantum length, 5 ms context switch
  - Waste of resources
    - 20% of CPU time (5/25) for context switch
  - If 500 ms quantum, better use of resources
    - 1% of CPU time (5/505) for context switch
    - Bad if lots of users in system – interactive users waiting for CPU
  - Balance found depends on job mix
- Still favors CPU-bound processes
  - An I/O bound process uses the CPU for a time less than the time quantum and then is blocked waiting for I/O
  - A CPU-bound process runs for its whole time slice and goes back into the ready queue (in front of the blocked processes)

Virtual Round Robin (VRR)

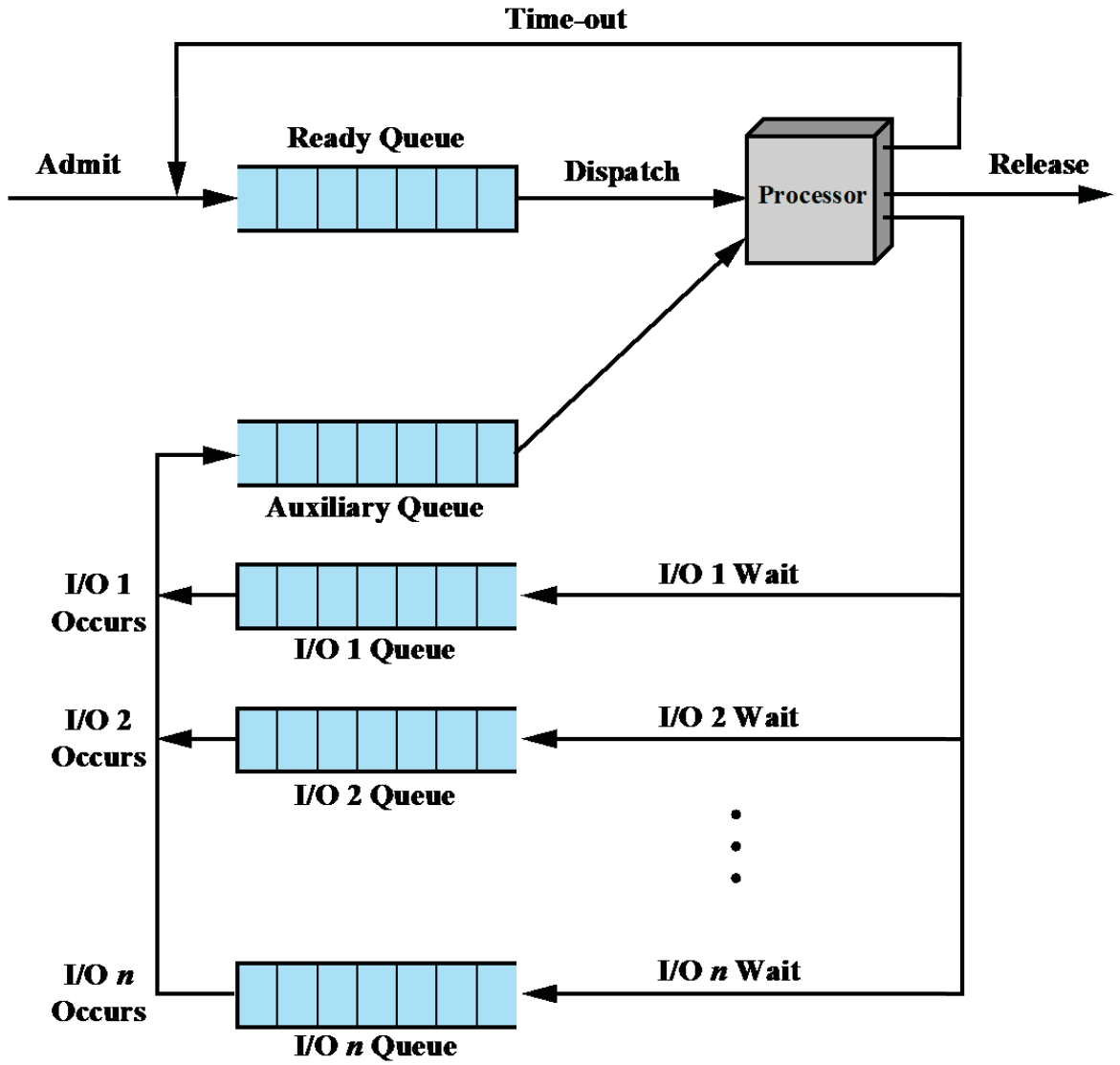


Figure 9.7 Queuing Diagram for Virtual Round-Robin Scheduler

Virtual Round Robin (VRR)

- **Virtual Round Robin** (VRR) avoids unfairness of RR.
- New processes arrive & join Ready queue - FCFS.
- Timed-out running processes rejoin Ready queue.
- I/O blocked processes join appropriate I/O queue.
  - (So far this is standard)
- NEW FEATURE: add FCFS *Auxiliary* queue.
- Processes released from I/O wait queue join *Auxiliary* queue.
- Scheduler dispatches processes from *Auxiliary* queue before processes from Ready queue. But these processes only run with up to the time remaining from their last basic time quantum (before they were blocked).