Real-Time Systems II

Alan Burns

University of York, UK

Simple Process Model

- The application is assumed to consist of a fixed set of processes
- All processes are periodic, with known periods
- The processes are completely independent of each other
- All system's overheads, context-switching times and so on are ignored (i.e, assumed to have zero cost)
- All processes have a deadline equal to their period (that is, each process must complete before it is next released)
- All processes have a fixed worst-case execution time

Standard Notation

B	Worst-case blocking time
C	Worst-case computation time (WCET)
D	Deadline of the process
I	The interference time of the process
J	Release jitter of the process
N	Number of processes in the system
P	Priority assigned to the process (if applicable)
R	Worst-case response time of the process
T	Minimum time between process releases
U	The utilization of each process (equal to C/T)

Fixed-Priority Scheduling (FPS)

- This is the most widely used approach and is the main focus of this course
- Each process has a fixed, static, priority which is computer pre-run-time
- The runnable processes are executed in the order determined by their priority
- In real-time systems, the "priority" of a process is derived from its temporal requirements, not its importance to the correct functioning of the system or its integrity

FPS and Rate Monotonic Priority Assignment

- Each process is assigned a (unique) priority based on its period; the shorter the period, the higher the priority
- i.e, for two processes `i` and `j`,

$$T_i < T_j \Rightarrow P_i > P_j$$

- This assignment is optimal in the sense that if any process set can be scheduled (using pre-emptive priority-based scheduling) with a fixed-priority assignment scheme, then the given process set can also be scheduled with a rate monotonic assignment scheme

Liu and Layland

- Proved rate monotonic optimality
- Showed that the worst-case situation is when all processes are released at the same time
- Showed that the first release is the worst
- Derived the utilisation-based test

Example Priority Assignment

Process	Period, T	Priority, P
a	25	5
b	60	3
c	42	4
d	105	1
e	75	2

Utilisation-Based Analysis

- For D=T task sets only
- A simple sufficient but not necessary schedulability test exists

$$U \equiv \sum_{i=1}^{N} \frac{C_i}{T_i} \leq N\left(2^{1/N} - 1\right)$$

$$U \leq 0.69 \text{ as } N \to \infty$$

Utilization Bounds

N	Utilization bound
1	100.0%
2	82.8%
3	78.0%
4	75.7%
5	74.3%
10	71.8%

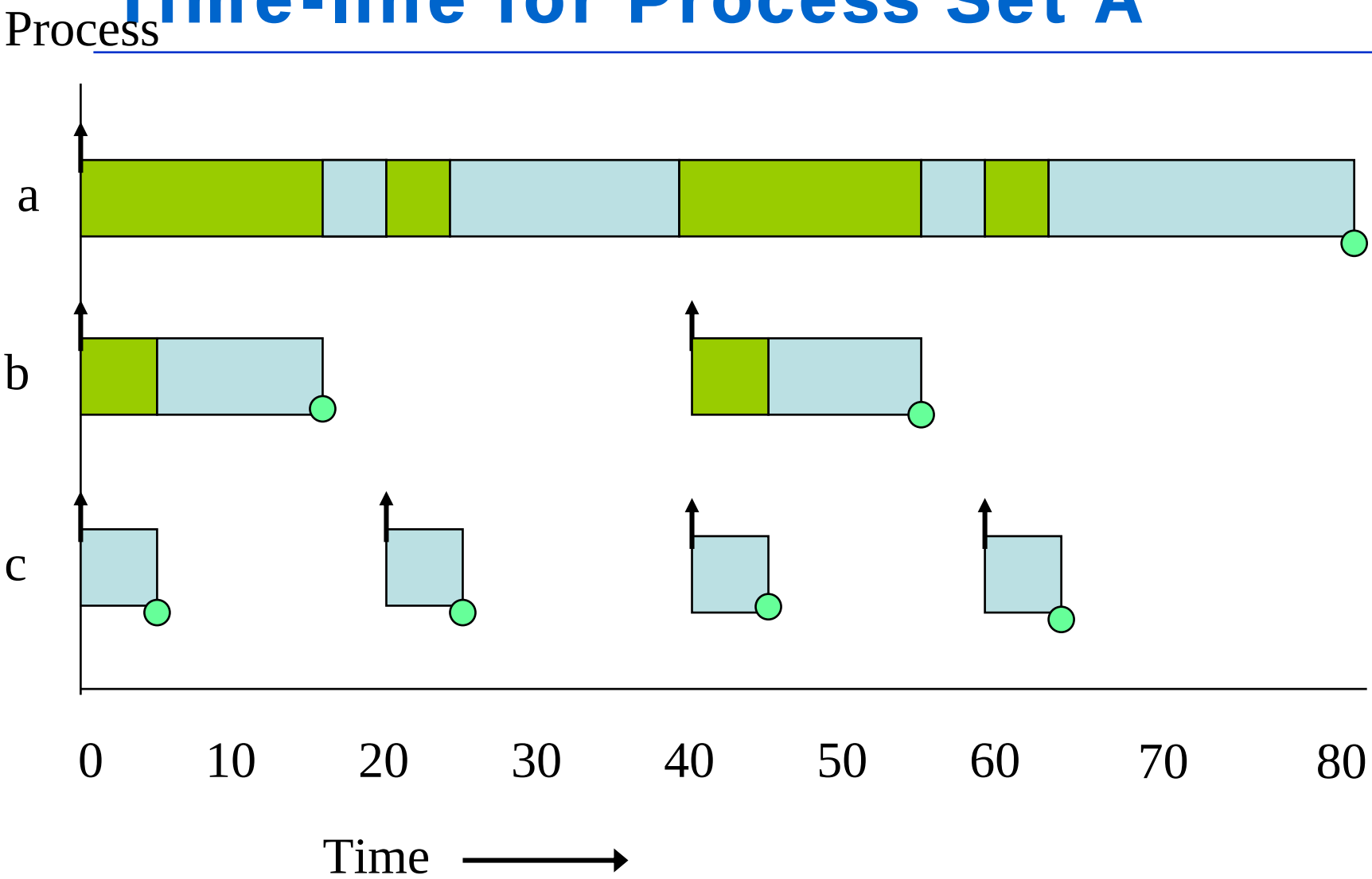
Approaches 69.3% asymptotically

Process Set A

Process	Period T	ComputationTime C	Priority P	Utilization U
a	80	40	1	0.50
b	40	10	2	0.25
c	20	5	3	0.25

- **The combined utilization is 1.0**
- **This is above the threshold for three processes (0.78) but the process set will meet all its deadlines**

Time-line for Process Set A



Criticism of Utilisation-based Tests

- Not exact
- Not general
- BUT it is O(N)

The test is said to be sufficient but not necessary

Response-Time Analysis

- **Here task *i*'s worst-case response time, *R*, is calculated first and then checked (trivially) with its deadline**

$$R_i \leq D_i$$

$$R_i = C_i + I_i$$

Where *I* is the interference from higher priority tasks

Calculating R

During R, each higher priority task j will execute a number of times:

$$\text{Number of Releases} = \left\lceil \frac{R_i}{T_j} \right\rceil$$

The ceiling function $\lceil \ \rceil$ gives the smallest integer greater than the fractional number on which it acts. So the ceiling of 1/3 is 1, of 6/5 is 2, and of 6/3 is 2.

Total interference is given by:

$$\left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

Response Time Equation

$$R_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

Where $hp(i)$ is the set of tasks with priority higher than task i

Solve by forming a recurrence relationship:

$$w_i^{n+1} = C_i + \sum_{j \in hp(i)} \left\lceil \frac{w_i^n}{T_j} \right\rceil C_j$$

The set of values $w_i^0, w_i^1, w_i^2, ..., w_i^n, ..$ is monotonically non decreasing. When $w_i^n = w_i^{n+1}$ the solution to the equation has been found; w_i^0 must not be greater that R_i (e.g. 0 or C_i)

Process Set B

Process	Period T	ComputationTime C	Priority P
a	7	3	3
b	12	3	2
c	20	5	1

$$R_a = 3$$

$$w_b^0 = 3$$

$$w_b^1 = 3 + \left\lceil \frac{3}{7} \right\rceil 3 = 6$$

$$w_b^2 = 3 + \left\lceil \frac{6}{7} \right\rceil 3 = 6$$

$$R_b = 6$$

$$w_c^0 = 5$$

$$w_c^1 = 5 + \left\lceil \frac{5}{7} \right\rceil 3 + \left\lceil \frac{5}{12} \right\rceil 3 = 11$$

$$w_c^2 = 5 + \left\lceil \frac{11}{7} \right\rceil 3 + \left\lceil \frac{11}{12} \right\rceil 3 = 14$$

$$w_c^3 = 5 + \left\lceil \frac{14}{7} \right\rceil 3 + \left\lceil \frac{14}{12} \right\rceil 3 = 17$$

$$w_c^4 = 5 + \left\lceil \frac{17}{7} \right\rceil 3 + \left\lceil \frac{17}{12} \right\rceil 3 = 20$$

$$w_c^5 = 5 + \left\lceil \frac{20}{7} \right\rceil 3 + \left\lceil \frac{20}{12} \right\rceil 3 = 20$$

$$R_c = 20$$

Revisit: Process Set A

Process	Period T	Computation Time C	Priority P	Response time R
a	80	40	1	80
b	40	10	2	15
c	20	5	3	5

- **The combined utilization is 1.0**
- **This was above the ulilization threshold for three processes (0.78), therefore it failed the test**
- **The response time analysis shows that the process set will meet all its deadlines**

Response Time Analysis

- Is sufficient and necessary
- If the process set passes the test they will meet all their deadlines; if they fail the test then, at run-time, a process will miss its deadline (unless the computation time estimations themselves turn out to be pessimistic)

Process Interactions / Blocking

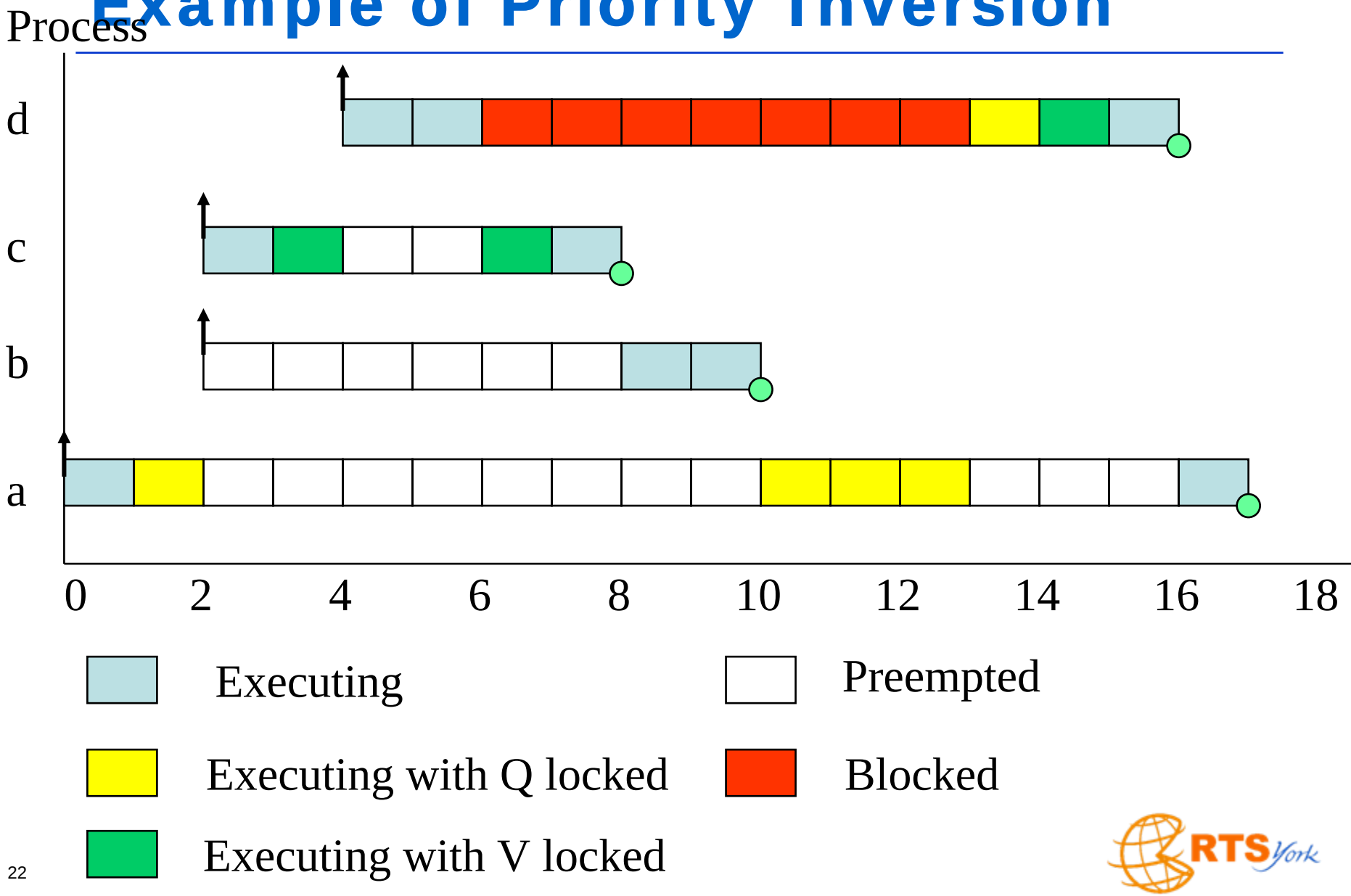
- If a process is suspended waiting for a lower-priority process to complete some required computation then the priority model is, in some sense, being undermined
- It is said to suffer **priority inversion**
- If a process is waiting for a lower-priority process, it is said to be **blocked**

Priority Inversion

- To illustrate an extreme example of priority inversion, consider the executions of four periodic processes: a, b, c and d; and two resources: Q and V

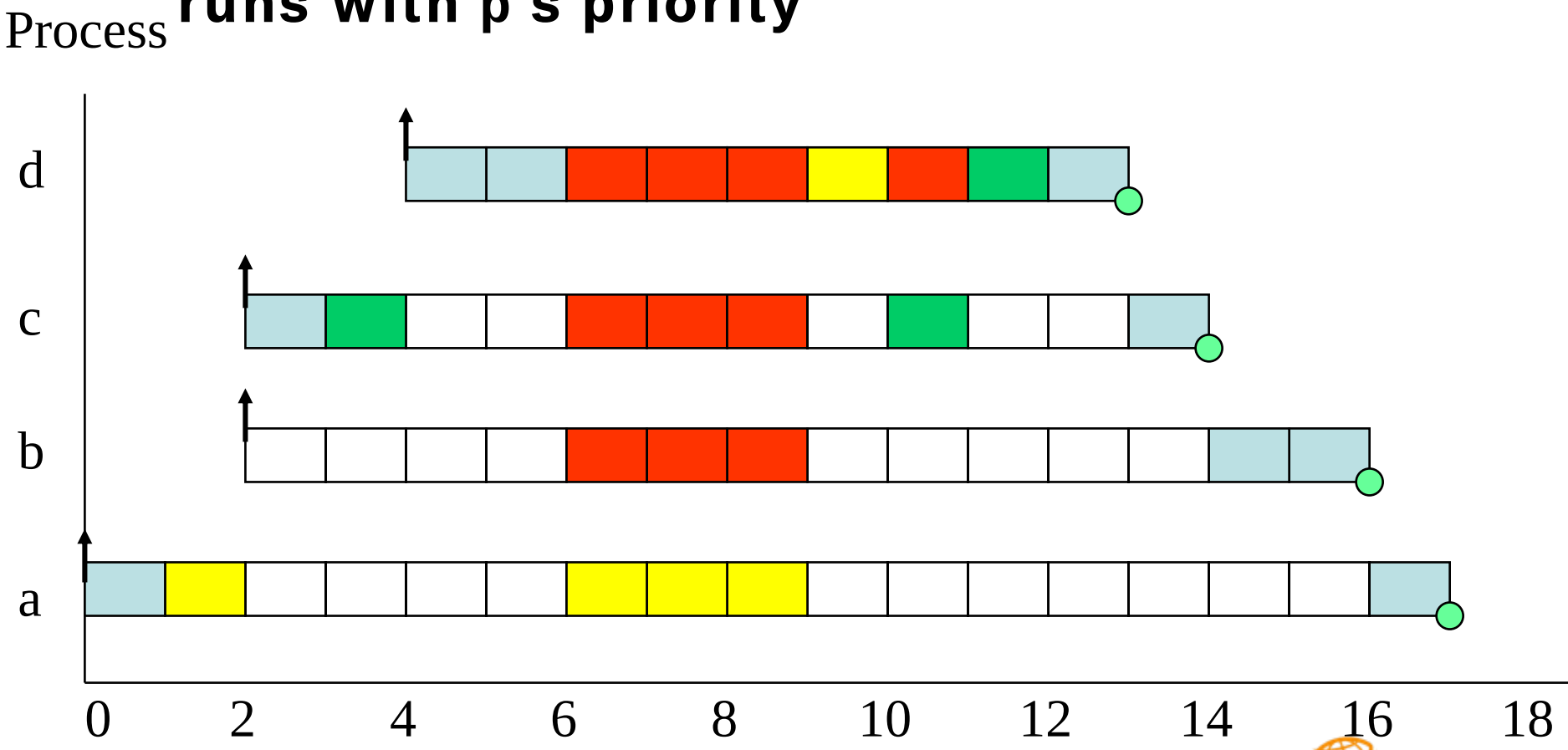
Process	Priority	Execution Sequence	Release Time
a	1	EQQQQE	0
b	2	EE	2
c	3	EVVE	2
d	4	EEQVE	4

Example of Priority Inversion



Priority Inheritance

- If process p is blocking process q, then q runs with p's priority



Mars Path-Finder

- A problem due to priority inversion nearly caused the lose of the Mars Path-finder mission
- As a shared bus got heavily loaded critical data was not been transferred
- Time-out on this data was used as an indication of failure and lead to re-boot
- Solution was a patch that turned on priority inheritance, this solved the problem

Response Time and Blocking

$$R_i = C_i + B_i + I_i$$

$$R_i = C_i + B_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

$$w_i^{n+1} = C_i + B_i + \sum_{j \in hp(i)} \left\lceil \frac{w_i^n}{T_j} \right\rceil C_j$$

Priority Ceiling Protocols

Two forms

- **Original ceiling priority protocol**
- **Immediate ceiling priority protocol**

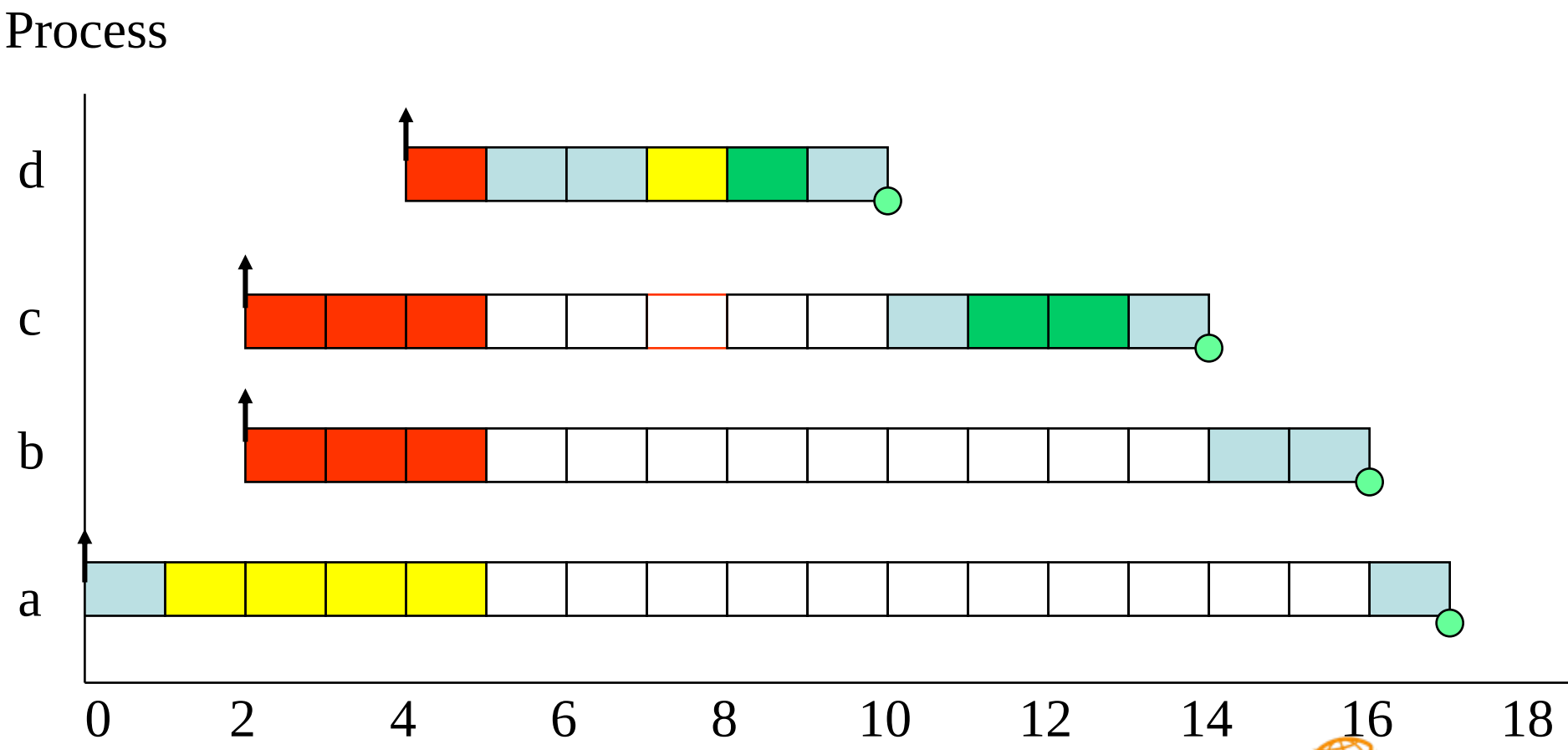
On a Single Processor

- A high-priority process can be blocked at most once during its execution by lower-priority processes
- Deadlocks are prevented
- Transitive blocking is prevented
- Mutual exclusive access to resources is ensured (by the protocol itself

ICPP

- Each process has a static default priority assigned (perhaps by the deadline monotonic scheme)
- Each resource has a static ceiling value defined, this is the maximum priority of the processes that use it
- A process has a dynamic priority that is the maximum of its own static priority and the ceiling values of any resources it has locked
- As a consequence, a process will only suffer a block at the very beginning of its execution
- Once the process starts actually executing, all the resources it needs must be free; if they were not, then some process would have an equal or higher priority and the process's execution would be postponed

ICPP Inheritance



Summary

- Rate monotonic analysis
- Response Time analysis
- Priority inversion and inheritance
- Priority ceiling protocols