Chapter 10

Stallings 9e

Multiprocessor/Multicore Classifications

- Loosely Coupled/ Distributed/Cluster
  - Supercomputer
  - Beowulf
- Asymmetric Multiprocessor - Specialized processors
  - I/O Processor
  - Floating point processor
  - GPU
- Tightly coupled Processor
  - Multicore

Granularity

Table 10.1 Synchronization Granularity and Processes

Grain Size	Description	Synchronization Interval (Instructions)
Fine	Parallelism inherent in a single instruction stream	< 20
Medium	Parallel processing or multitasking within a single application	20–200
Coarse	Multiprocessing of concurrent processes in a multiprogramming environment	200–2,000
Very Coarse	Distributed processing across network nodes to form a single computing environment	2,000–1M
Independent	Multiple unrelated processes	Not applicable

Scheduling Performance

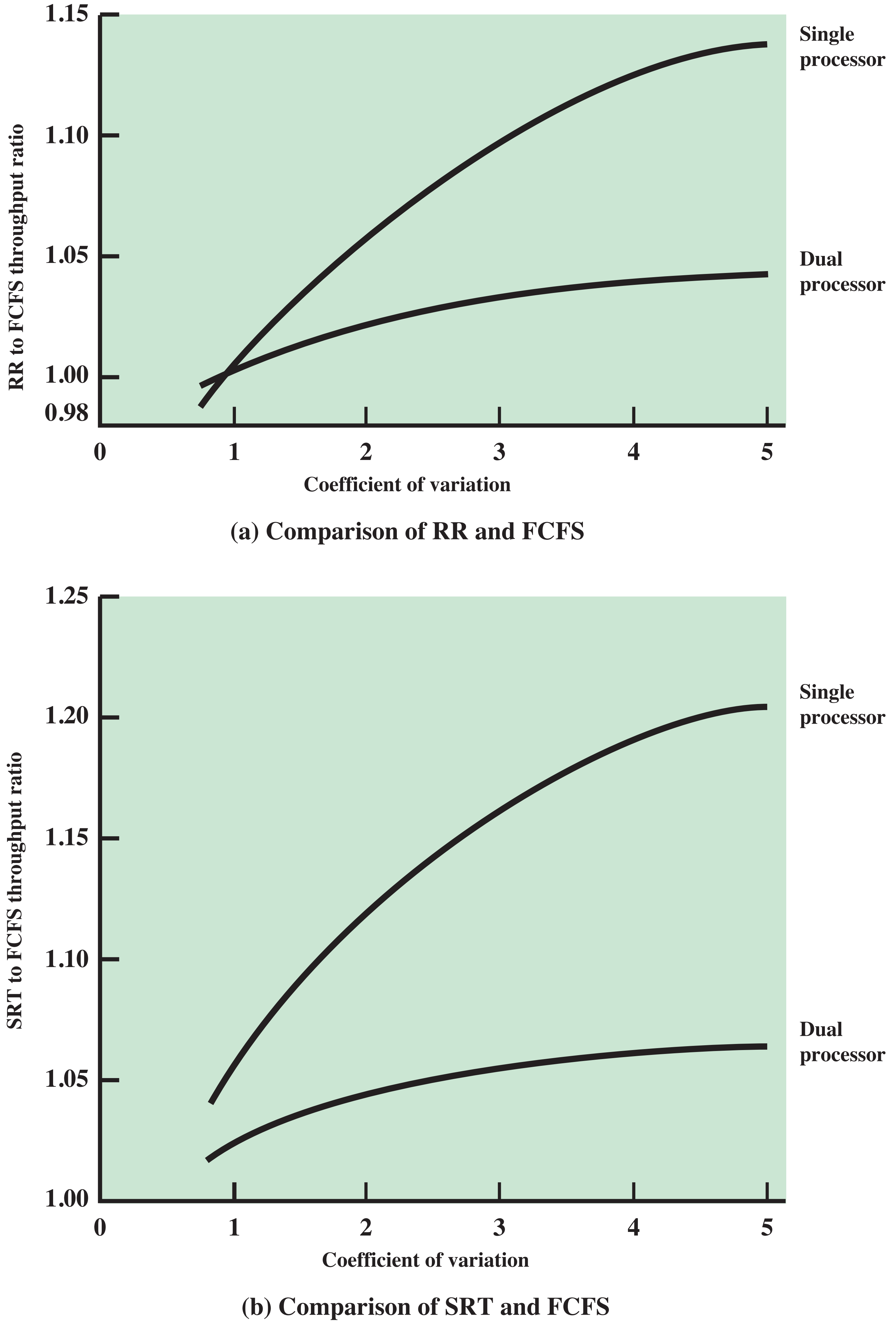


Figure 10.1 Comparison of Scheduling Performance for One and Two Processors

Thread Scheduling

- Load Sharing - CPU treated as an system resource
- Gang Scheduling - Set of threads scheduled on CPUs as a unit
- Dedicated Processor - threads are assigned to a particular CPU
  - Affinity - the ability to assign a thread to a CPU
    - Soft affinity - scheduler tries, but might not succeed
- Dynamic Scheduling - number of threads can be changed

Load Sharing - Most Common

Scheduling Policies

- First Come First Served (FCFS)
- Smallest Number of Threads First
- Preemptive smallest number of threads first

Load Sharing

Most Common, but:

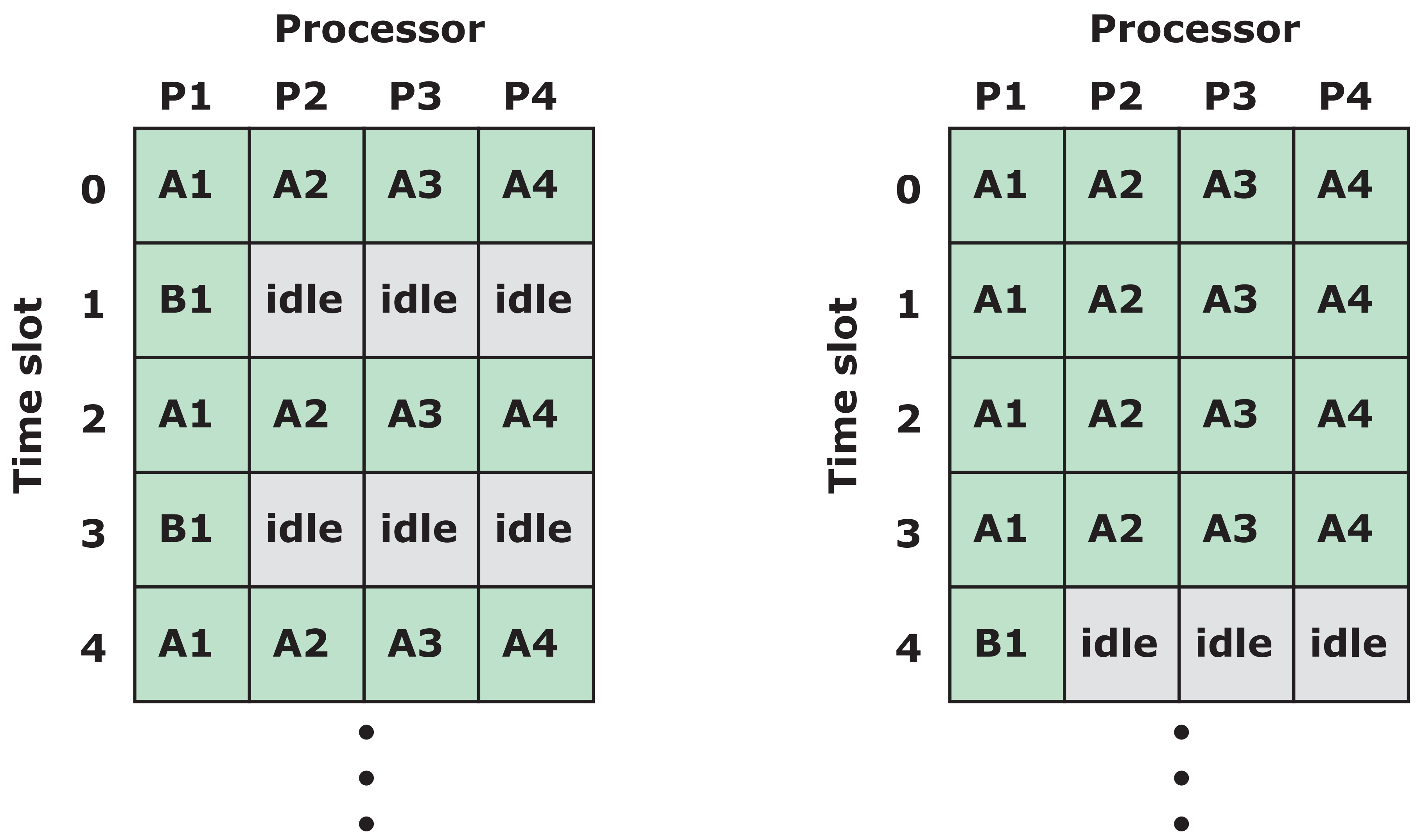
- Central queue of threads becomes a bottleneck
- Preempted threads unlikely to be assigned to same CPU again
- No guarantee that all the threads of a process scheduled at same time

Performance enhancements:

- Mach Kernel: local run queue plus shared global run queue
- Solaris: lightweight threads
- Gang Scheduling

Gang Scheduling

- All threads of a process (or other group) scheduled at once
  - Reduces context switching within threads of a process
  - Applied to fine or medium grained applications



(a) Uniform scheduling

(b) Weighted scheduling

Figure 10.2 Gang Scheduling

Dedicated Scheduling

Minimizes expensive context switching

Table 10.2 Application Speedup as a Function of Number of Threads

Number of Threads per Application	Matrix Multiplication	FFT
1	1	1
2	1.8	1.8
4	3.8	3.8
8	6.5	6.1
12	5.2	5.1
16	3.9	3.8
20	3.3	3
24	2.8	2.4

Multicore Scheduling

Typical Multicore Processor

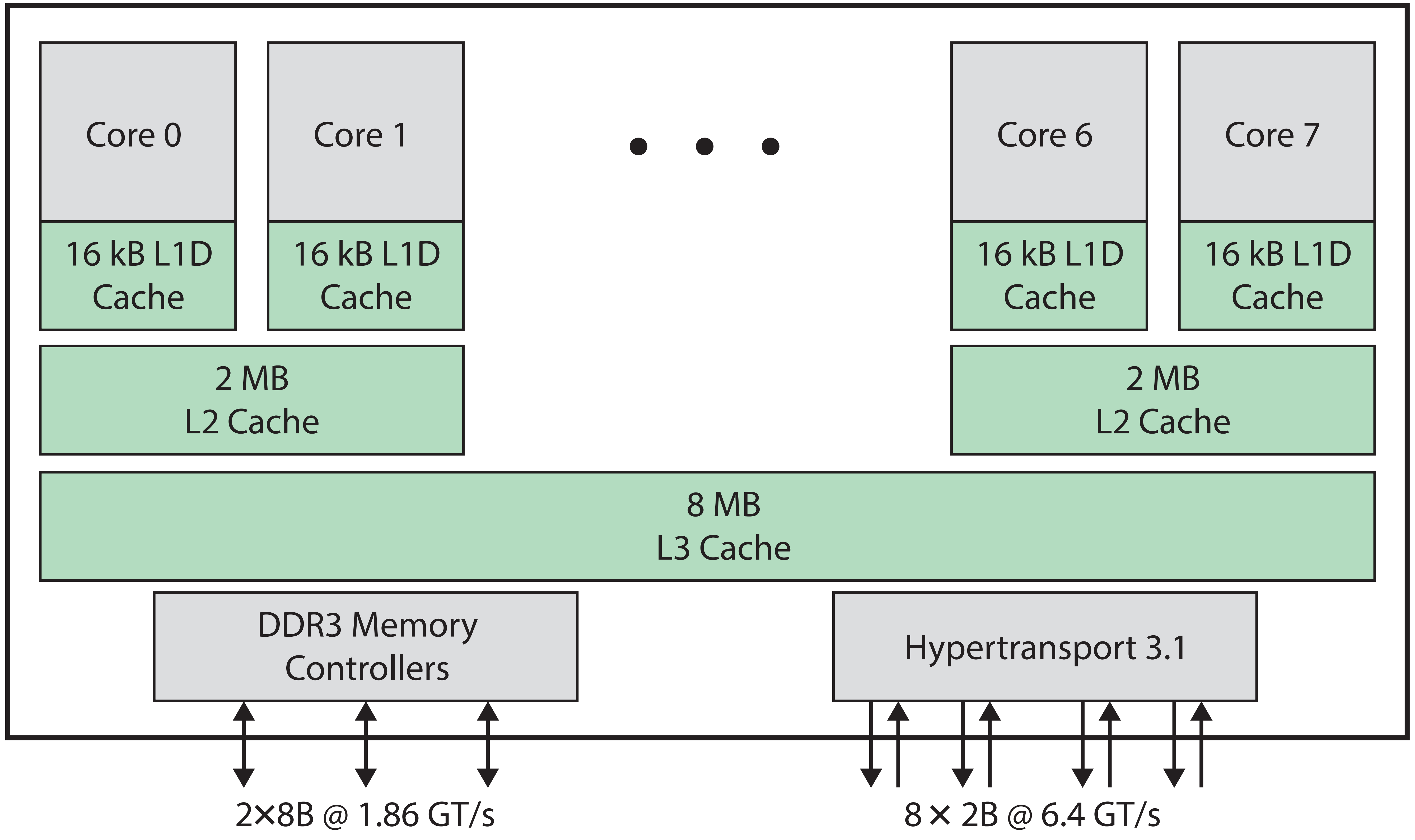


Figure 10.3 AMD Bulldozer Architecture

Real-Time Scheduling

- In addition to performing the correct computation, task must meet certain time constraints
  - Hard real time - system must meet time constraint (deadline), otherwise task will not execute properly
  - Soft real time - deadline is desirable, but not mandatory
- Type of real time task
  - Aperiodic task - has deadline by which it must finish and/or start
  - Periodic task - repetitive execution, “must run once within a second”

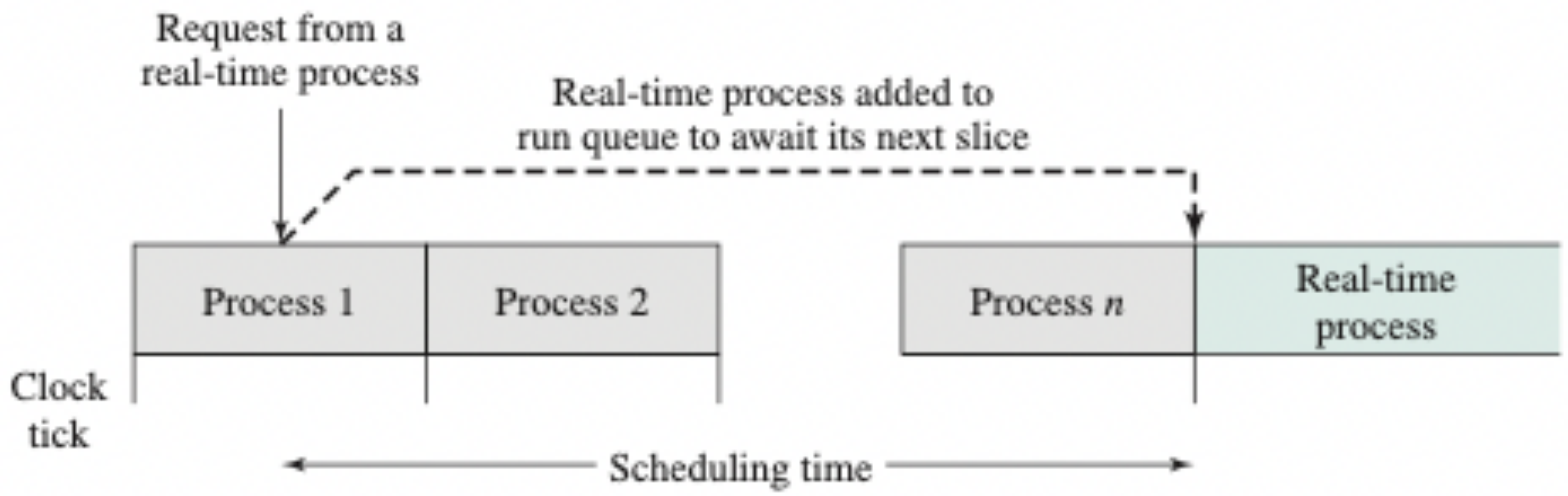
Real Time Characteristics

- Determinism - system timing is same for all input or conditions
- Responsiveness - how long does system take to respond to an event
- User control - how much can user influence operation of system
- Reliability - what is behavior of system in response to failures
- Fail-Soft Operation - in event of failure/fault, can task stop in a condition that is not detrimental to overall system

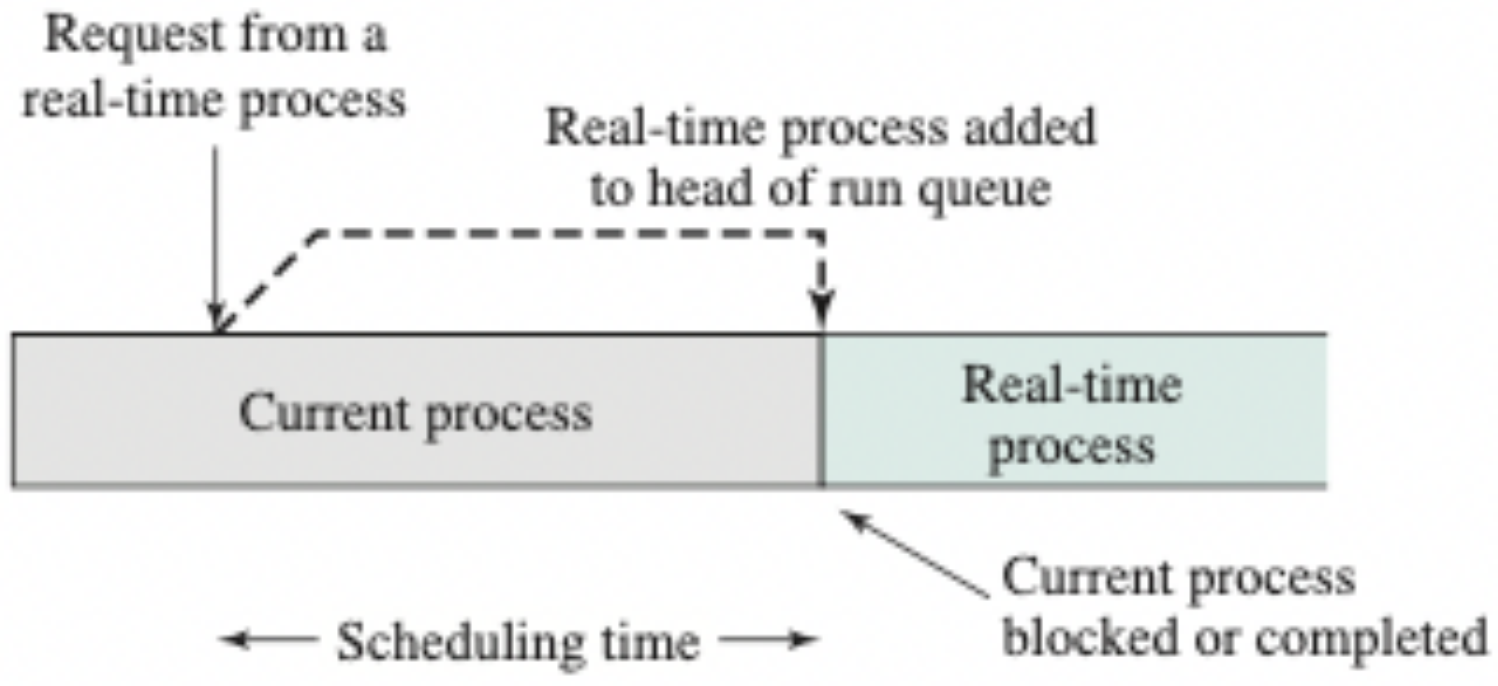
Real-Time Scheduling

- Static table-driven approaches - predetermined schedule
- Static priority-driven preemptive approaches - priorities are modified in response to conditions
- Dynamic planning-based approaches - schedule is determines at run time
- Dynamic best effort approaches - system tries to meet all deadlines, aborts/ ignores any process that doesn’t meet its deadline

Real-Time Schedules



(a) Round-robin preemptive scheduler



(b) Priority-driven nonpreemptive scheduler

Real-Time Schedules

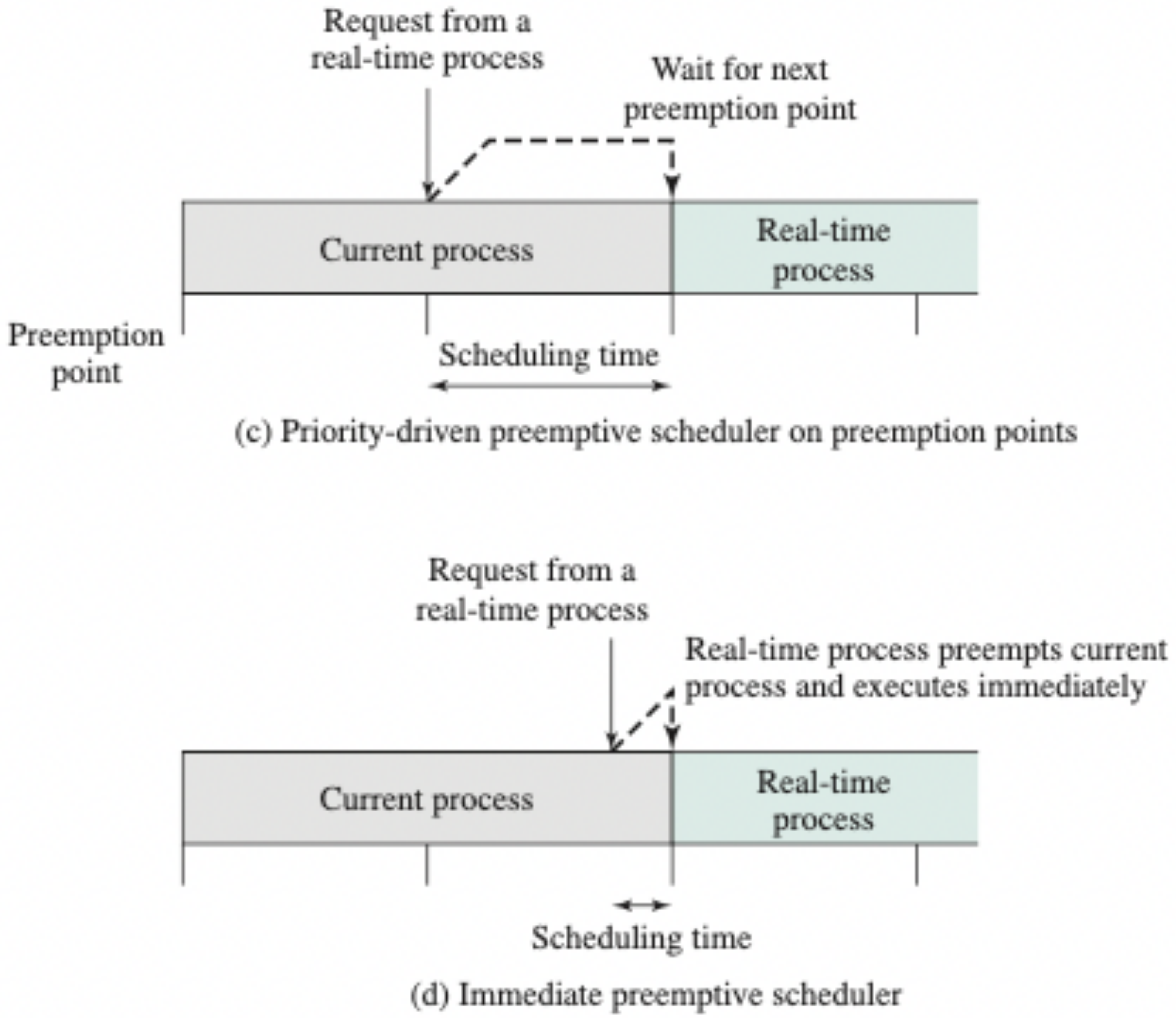


Figure 10.4 **Scheduling of Real-Time Process**

Deadline Scheduling

Information needed for scheduling

- Ready Time
- Starting Deadline
- Completion Deadline
- Processing time
- Resource Needs
- Priority
- Subtask structure

Table 10.3 Execution Profile of Two Periodic Tasks

Process	Arrival Time	Execution Time	Ending Deadline
A(1)	0	10	20
A(2)	20	10	40
A(3)	40	10	60
A(4)	60	10	80
A(5)	80	10	100
•	•	•	•
•	•	•	•
•	•	•	•
B(1)	0	25	50
B(2)	50	25	100
•	•	•	•
•	•	•	•
•	•	•	•

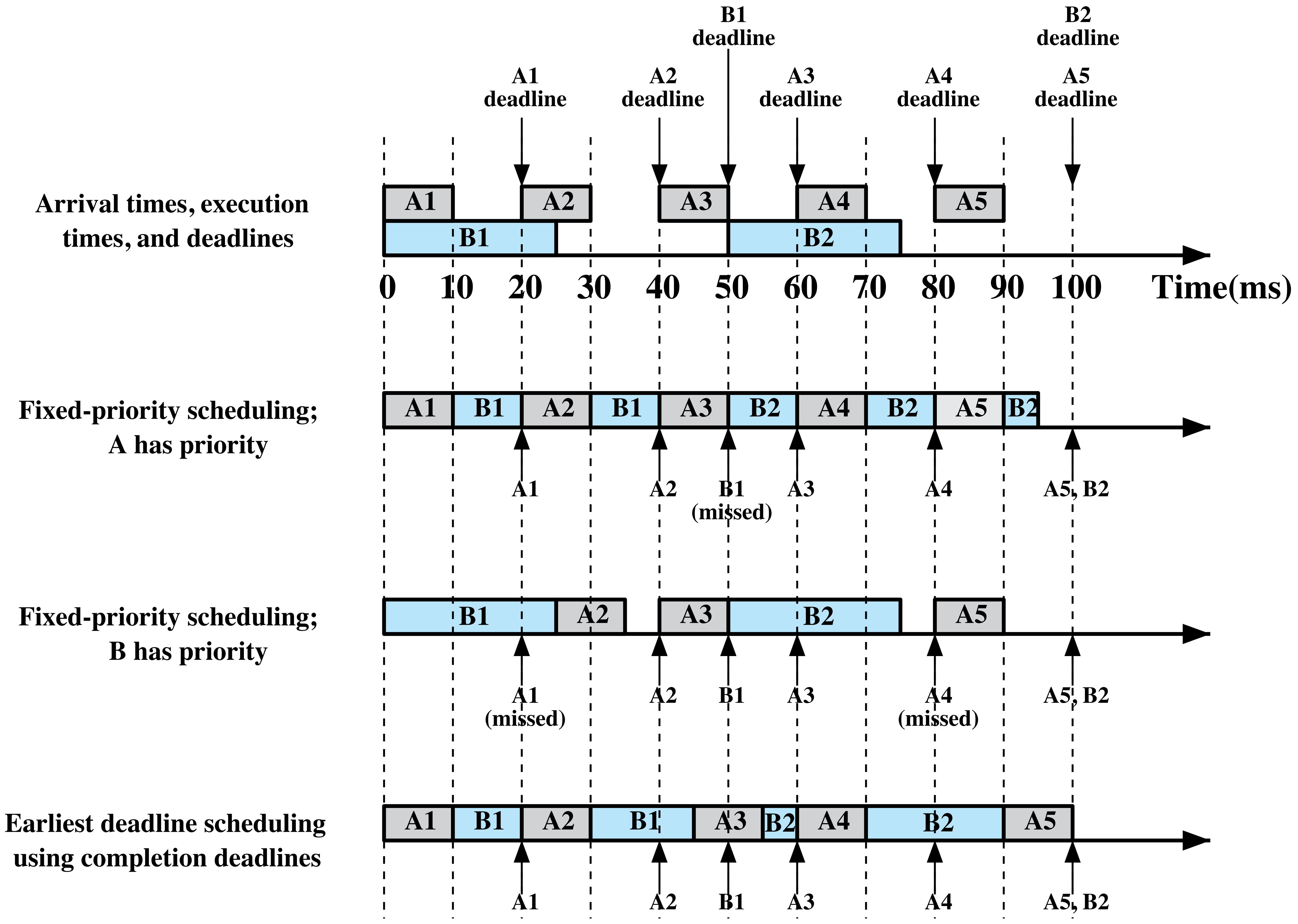


Figure 10.5 Scheduling of Periodic Real-time Tasks with Completion Deadlines (based on Table 10.2)

Rate Monotonic Scheduling

- Task priority is based on task's period
  - Highest priority task is one with shortest period
  - Next highest priority task is one with next shortest period, etc.

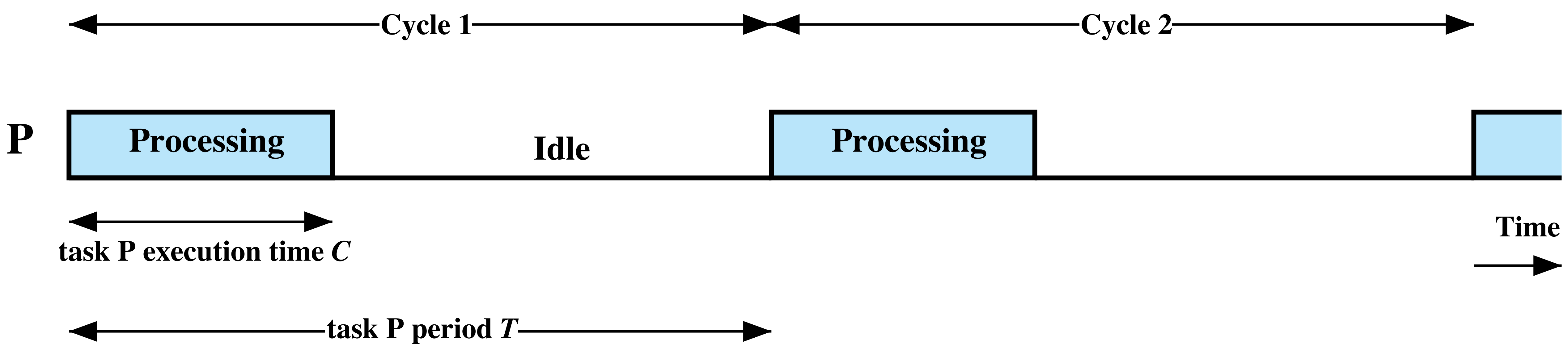


Figure 10.7 Periodic Task Timing Diagram

Linux Real-time Scheduling

- `SCHED_FIFO` - First-in-first-out real time threads
  - When executing FIFO thread is interrupted, thread placed in proper priority queue
  - When a FIFO thread becomes ready, if it has higher priority than currently executing thread is preempted and highest priority thread executes. For equal priority threads, longest waiting thread executes
  - Currently executing thread continues executing, unless a higher priority thread becomes ready, or thread is blocked for an event, or thread voluntarily gives up processor (`sched_yield`)

Linux Real-time Scheduling (Con’t)

- `SCHED_RR` - round robin real-time threads
  - Real-time threads take priority over non-real-time threads
- `SCHED_NORMAL` - other, non-real-time threads

Priorities - lower value is higher priority

- Real-time priorities assigned values 1-99
- `SCHED_NORMAL` assigned value 100-139

Windows Scheduling

- Priority-based preemptive scheduler
- Two classes ("bands")
  - Real-time priority class - all threads have fixed priority, have precedence over normal threads
  - Variable priority class - initial priority, but may be raised over task's lifetime
- Priority queues - managed round robin

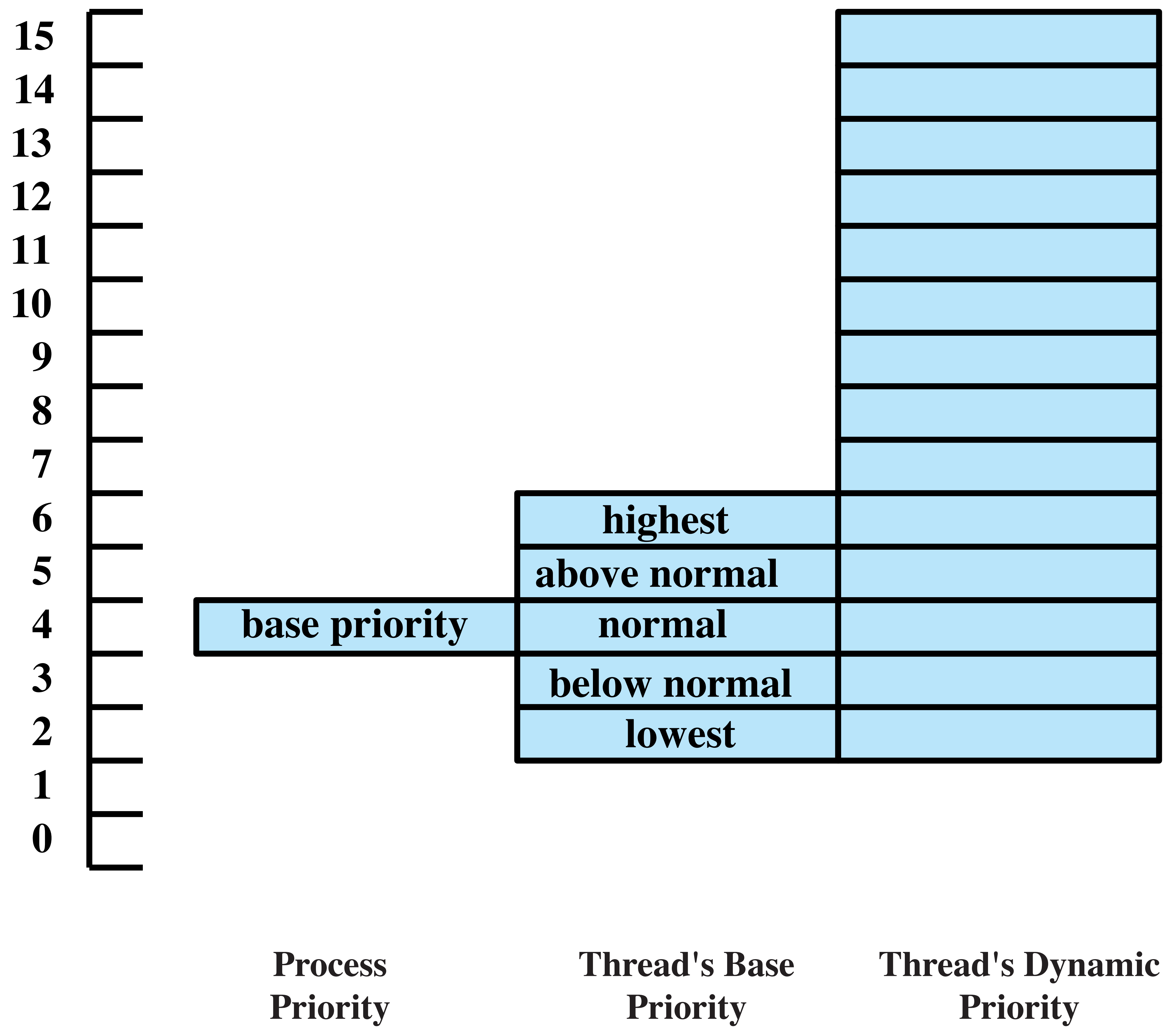


Figure 10.15 Example of Windows Priority Relationship