Struktura jądra UNIX

Wykład 1: Szeregowanie czasu rzeczywistego

Rola planisty zadań

Planista krótkoterminowy (ang. *short-term*) podejmuje decyzje ponad kilka tysięcy razy na sekundę. Uruchamiany, gdy trzeba przełączyć się na inny wątek (blokada, oczekiwanie na wej.-wyj. wywłaszczenie). Uzupełnia podstawowe statystyki o procesach.

Planista długoterminowy (ang. *long-term*) uruchamiany co najwyżej kilka razy na sekundę. Wykonuje przeliczenie priorytetów na podstawie danych historycznych (`loadavg`) i różnych heurystyk.
→ Często zintegrowany z planistą krótkoterminowym jako algorytm iteracyjny.

Dyspozytor (ang. *dispatcher*) to moduł zajmujący się przydziałem czasu procesora dla wątku wybranego przez planistę.

Jak dyspozytor wchodzi w interakcję z planistą?

Cele szeregowania zadań

Ogólne, systemowe:

- **sprawiedliwość** każdy proces traktowany tak samo
- **wymuszenie polityki** niemożność naginania zasad
- **równomierne zużycie zasobów**

Systemy interaktywne:

- **czas odpowiedzi** na zdarzenia od użytkownika
- **proporcjonalność** zużytego czasu vs. złożoność zadania

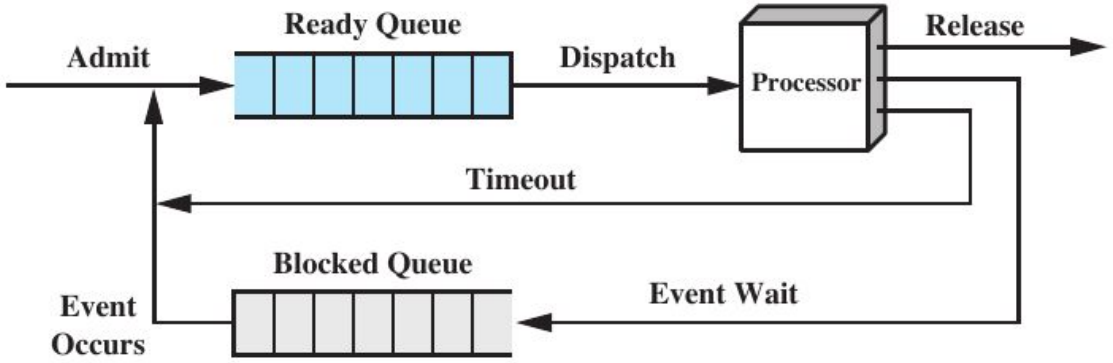
Systemy wsadowe:

- maksymalizacja **przepustowości**
- minimalizacja **czasu przebywania w systemie**
- **maksymalizacja zużycia czasu procesora**

Systemy czasu rzeczywistego:

- **dotrzymywanie terminów** (nie gubi zdarzeń i danych)
- **determinizm** (w systemach multimedialnych)

Planista i dyspozytor



Wątki w stanie `READY` na **kolejce wątków uruchamialnych** (ang. *run queue*). System bez wywłaszczania → algorytm FCFS (ang. *first-come first-served*). Z wywłaszczaniem → algorytm karuzelowy (ang. *round-robin*).

Dyspozytor (ang. *dispatcher*) odpowiada za zmianę kontekstu, aktualizuje statystyki – liczbę zmian kontekstu, czas wykonywania.

Szeregowanie zadań czasu rzeczywistego

Przekroczenie **terminu** w systemach o **ostrych ograniczeniach czasowych** (ang. *hard real-time*) może mieć katastrofalne skutki; w systemach o **łagodnych ograniczeniach** (ang. *soft real-time*) zaledwie zachodzą negatywne skutki (np. utrata przesyłanych klatek obrazu).

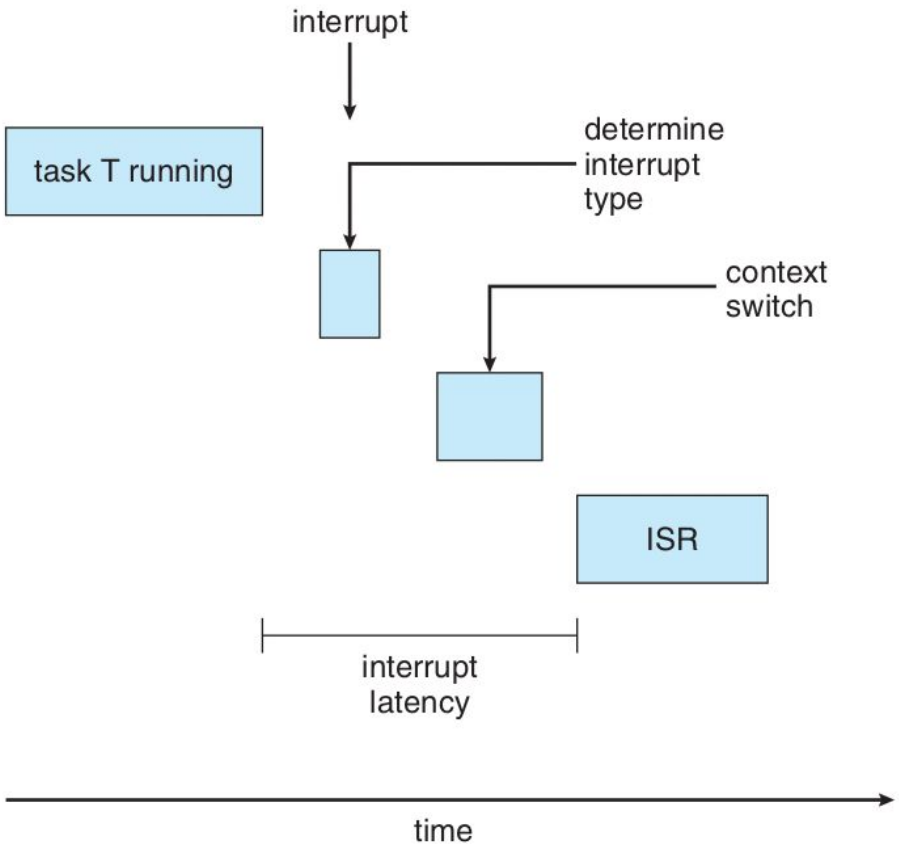
Zadania **okresowe** (ang. *periodic*) → kontrola silników; **nieokresowe** (ang. *aperiodic*) → korekcja pozycji. Minimalizujemy **czas odpowiedzi** (ang. *response interval*), na który składają się czas doręczenia przerwania, opóźnienie ekspedycji i ostatecznie przetworzenie zadania.

Opóźnienie obsługi przerwań

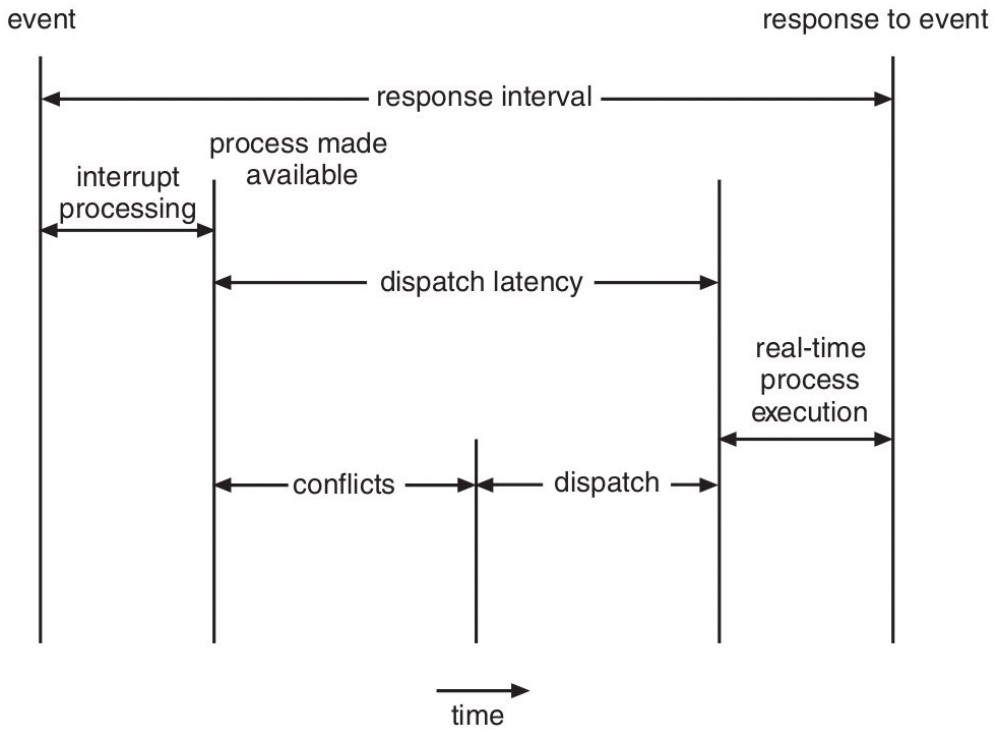
Ile czasu mija od zgłoszenia przerwania przez urządzanie do momentu jego obsłużenia?

Opóźnienie obsługi przerwań (ang. *interrupt latency*).

Jeśli przetwarzanie następuje w **procedurze obsługi przerwania** (ang. *interrupt service routine*), to jest to najkrótszy możliwy czas reakcji na zdarzenie.

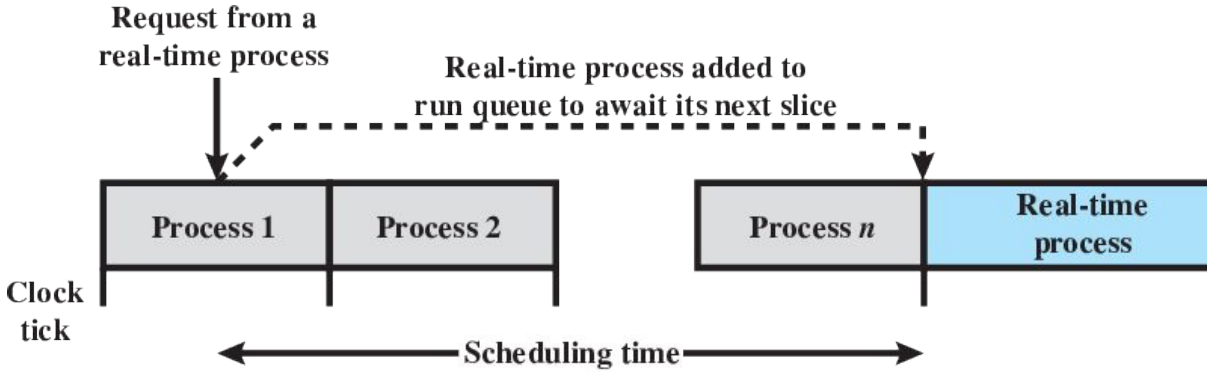


Opóźnienie obsług zdarzeń

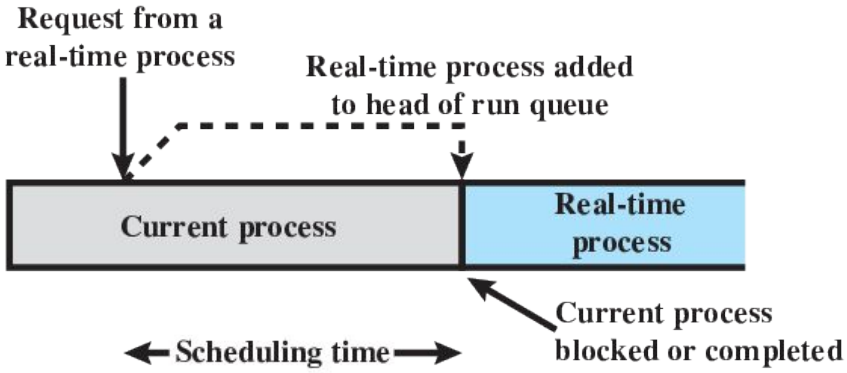


Na opóźnienie ekspedycji składa się (a) wywłaszczenie bieżącego procesu (b) odblokowania zadania do wykonania poprzez oddanie mu niezbędnych zasobów.

Czas odpowiedzi i algorytm szeregowania (1)

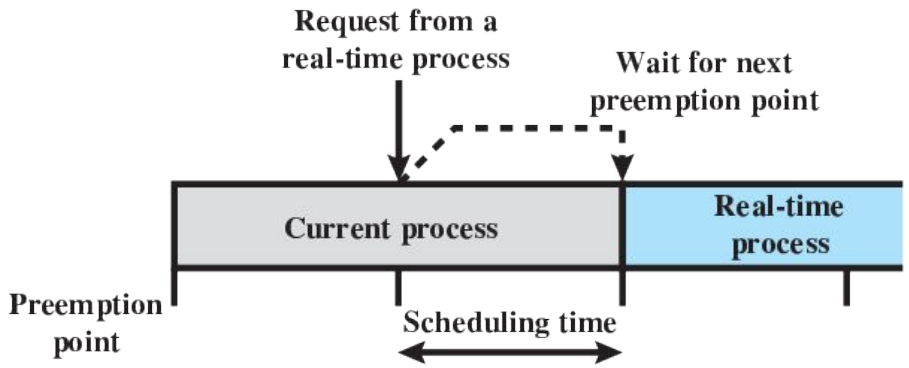


(a) Round-robin Preemptive Scheduler

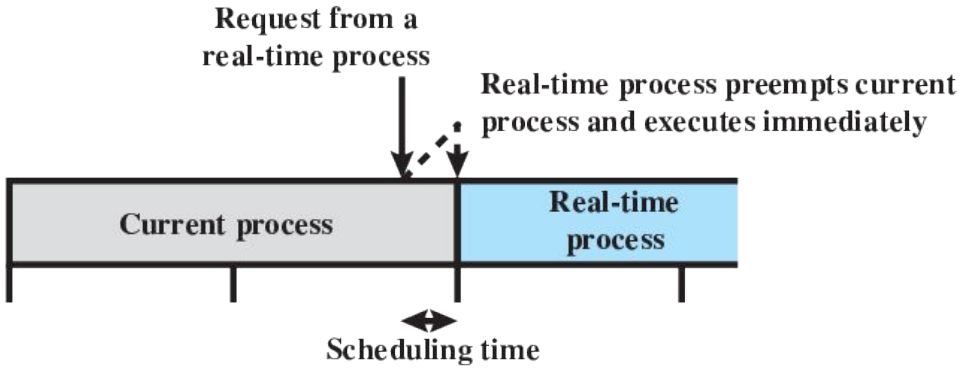


(b) Priority-Driven Nonpreemptive Scheduler

Czas odpowiedzi i algorytm szeregowania (2)

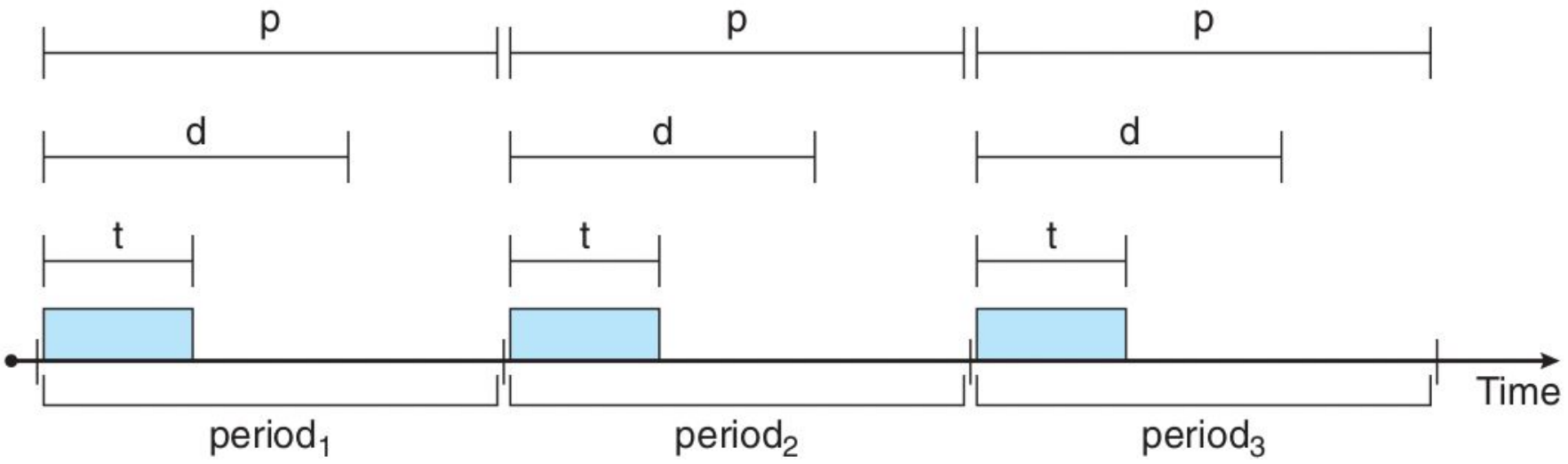


(c) Priority-Driven Preemptive Scheduler on Preemption Points



(d) Immediate Preemptive Scheduler

Szeregowanie zadań okresowych



Uruchamiamy zadanie i okresowo co p_i ms, nie możemy przekroczyć terminu d_i, a czas obsługi zadania wynosi t_i.

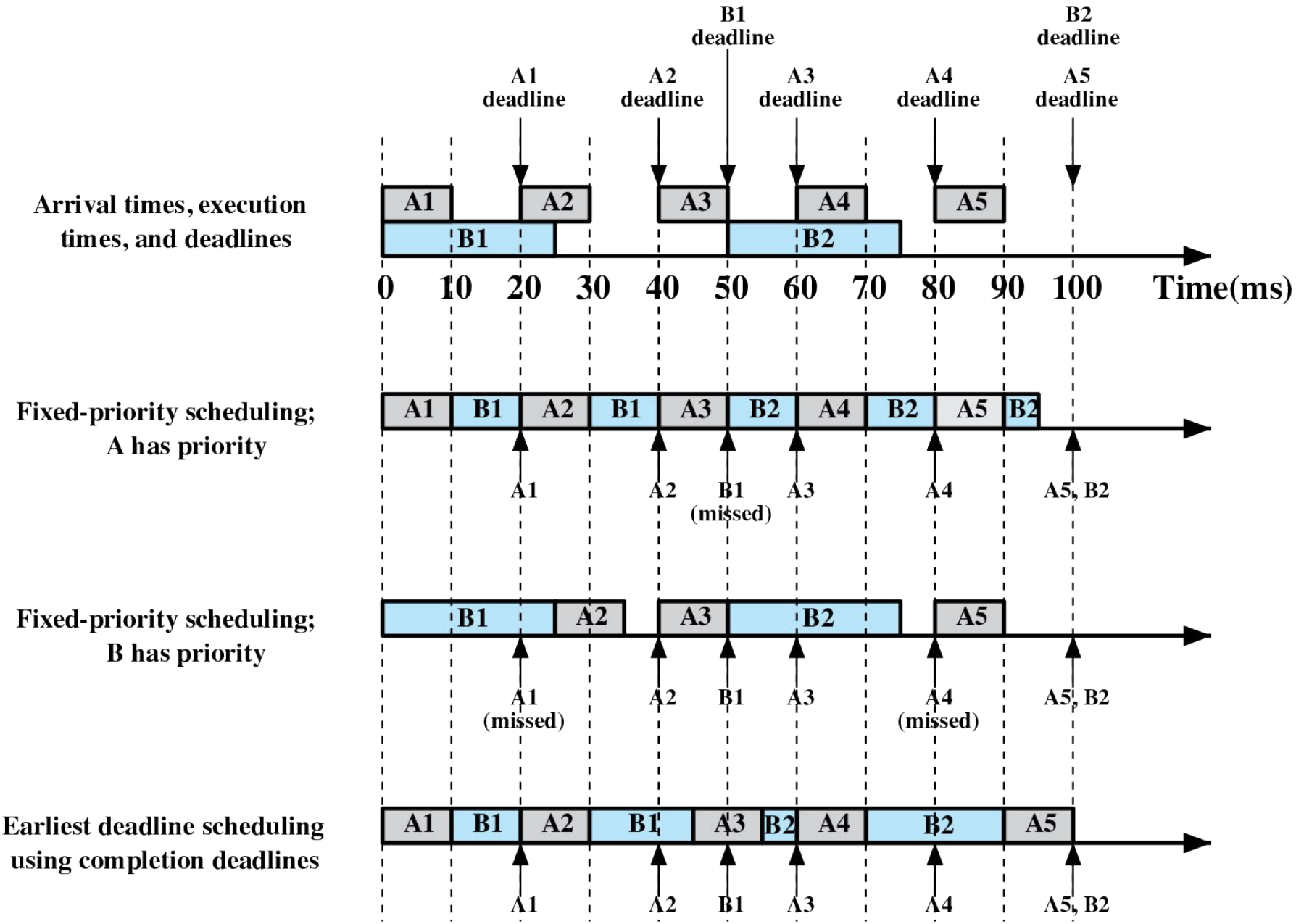
Algorytmy szeregowania dla systemów RT

Algorytm **RMS** (ang. *rate-monotonic scheduling*),
czyli **szeregowanie monotoniczne w częstotliwości**.
Stosuje wywłaszczanie. Zakłada, że $p_i = d_i$. Przypisuje zadaniom statyczny priorytet zależny od czasu terminu.

Algorytm **EDF** (ang. earliest deadline first),
czyli **najpierw wcześniejszy termin zakończenia**.
Stosuje wywłaszczanie. Wybiera zadanie o najwcześniejszym terminie (dynamiczny priorytet).

Okazuje się, że zestaw zadań jest **szeregowalny**:

- dla **RMS** jeśli $\sum t_i / p_i \leq n(2^{1/n} - 1)$, które zbiega do $\ln 2$
- dla **EDF** jeśli $\sum t_i / p_i \leq 1$


B1 deadline
B2 deadline
A1 deadline
A2 deadline
A3 deadline
A4 deadline
A5 deadline
Arrival times, execution times, and deadlines
A1
A2
A3
A4
A5
B1
B2
0
10
20
30
40
50
60
70
80
90
100
Time(ms)
Fixed-priority scheduling; A has priority
A1
B1
A2
B1
A3
B2
A4
B2
A5
B2
A1
A2
B1 (missed)
A3
A4
A5, B2
Fixed-priority scheduling; B has priority
B1
A2
A3
B2
A5
A1 (missed)
A2
B1
A3
A4 (missed)
A5, B2
Earliest deadline scheduling using completion deadlines
A1
B1
A2
B1
A3
B2
A4
B2
A5
A1
A2
B1
A3
A4
A5, B2

Pytania?