Systemy czasu rzeczywistego

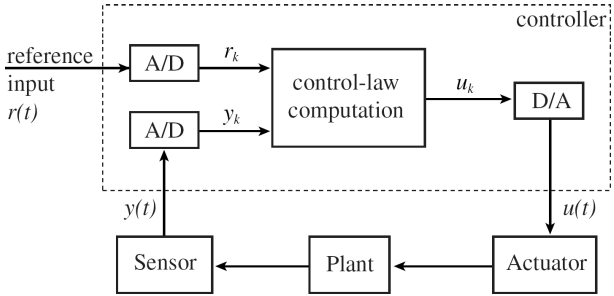
Mariusz Żynel
mariusz@math.uwb.edu.pl
http://math.uwb.edu.pl/~mariusz/

Uniwersytet w Białymstoku

2024/2025

Na poprzednim wykładzie...

Próbkujące regulatory cyfrowe



- Odczyt przez próbkowanie analogowych sensorów (sensor)
- Digitalizacja odczytów (A/D)
- Obliczenie korekty zgodnie z regułami sterowania (control-law)
- Konwersja cyfrowego sygnału sterującego na analogowy (D/A)
- Przesłanie sygnału aktywującego siłowniki (actuator)

Klasyfikacja aplikacji czasu rzeczywistego

Czysto okresowe (cyfrowa kierownica, flight control)

- każde zadanie wykonywane jest okresowo
- operacje wejścia/wyjścia przez odpytywanie (polled I/O)
- zapotrzebowanie na zasoby (obliczeniowe, komunikacyjne i pamięci masowej) nie zmienia się znacząco z okresu na okres

Głównie cykliczne (nawigacja, flight management)

- większość zadań wykonywana jest okresowo
- system musi reagować asynchronicznie na zdarzenia zewnętrzne

Asynchroniczne i przewidywalne (komunikacja multimedialna, systemy radarowe)

- czas wykonania kolejnych zadań może się znacznie różnić
- duże różnice w zapotrzebowaniu na zasoby w różnych okresach
- zakresy zmian są albo ograniczone, albo znane są statystyki

Asynchroniczne i nieprzewidywalne (air traffic control)

- zadania o dużej złożoności w czasie wykonywania
- konieczność reagowania na zdarzenia asynchroniczne

Model referencyjny systemów czasu rzeczywistego

.

A reference model of real-time systems

Prace i zadania

Praca : jednostka działania, która jest zaplanowana i wykonywana przez system (*job*), np.: obliczenie transformaty Fouriera dla danych z sensora, przesłanie pakietu danych po sieci komputerowej, odczytanie pliku z dysku twardego

Zadanie : zestaw powiązanych ze sobą prac, które wspólnie razem zapewniają pewną funkcjonalność systemu (*task*), np.: utrzymywanie stałej wysokości samolotu

Procesory i zasoby

- Praca jest wykonywana na *procesorze* i może wymagać pewnych *zasobów*

Procesor : aktywny komponent, na którym prace są wykonywane (*processor*)

- Na przykład:
  - CPU – wykonuje instrukcje maszynowe
  - łącze danych – przesyła dane z jednego miejsca do drugiego
  - dysk/nośnik – odczytuje i zapisuje pliki
  - baza danych – przetwarza kwerendy
- Procesor posiada atrybut *prędkość*, który oznacza tempo postępu pracy w kierunku jej ukończenia, określany jako np. ilość instrukcji na sekundę, przepustowość sieci
- Dwa procesory są tego samego typu, jeżeli są funkcjonalnie identyczne i można używać ich zamiennie

Zasób : pasywny komponent, wymagany do wykonania pracy (*resource*)

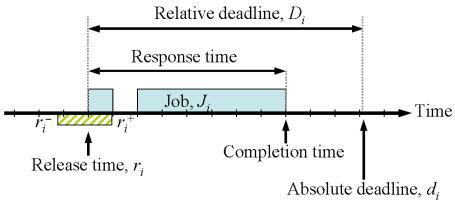
- Na przykład: pamięć, semafory, blokady bazy danych
- Zasoby mają różne typy i rozmiary, ale nie mają atrybutu prędkości
- Zasoby są zazwyczaj wielokrotnego użytku i nie są konsumowane w trakcie używania

Wykorzystanie zasobów

- Jeśli system zawiera ρ typów zasobów, oznacza to, że:
  - istnieje ρ różnych typów zasobów wielokrotnego użytku
  - istnieje przynajmniej jedna jednostka (*unit*) każdego typu zasobu
  - jedna jednostka zasobu może być używana najwyżej przez jedną pracę na raz (dostęp wykluczający się wzajemnie, *mutually exclusive access*)
  - praca musi uzyskać jednostkę wymaganego zasobu, użyć jej, a następnie ją zwolnić
- Zasób jest *obfity* (*plentiful*), jeśli nie zdarza się aby jakakolwiek praca była blokowana z powodu niedostępności tego zasobu, innymi słowy, zasób ten jest zawsze dostępny
  - obfity nie oznacza nieskończony
  - jeśli wiemy, że dany zasób nie wpływa na czas wykonania pracy to możemy go pominąć

Ograniczenia czasowe

Czas uwolnienia : moment, w kórym praca staje się gotowa do wykonania (*release time*)
Czas ostateczny : moment, w którym wykonanie pracy musi zostać ukończone (*deadline*)
Czas wykonania : sumaryczny czas spędzony przez pracę na procesorze (*execution time*)
Czas reakcji : czas od momentu uwolnienia pracy do chwili zakończenia (*response time*)
Względny czas ostateczny : maksymalny dopuszczalny czas reakcji (*relative deadline*)
Bezwzględny czas ostateczny : czas uwolnienia + względny czas ostateczny (*absolute deadline*)

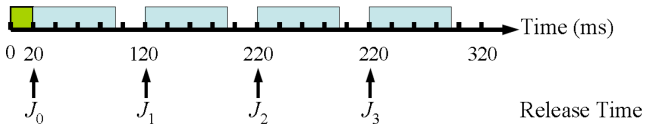


Time constraints

Ograniczenia czasowe - przykład cz. 1

- System do monitorowania i sterowania piecem grzewczym
- System potrzebuje 20ms, aby zainicjować się po załączeniu
- Po inicjalizacji, co 100ms, system:
  - sampluje i odczytuje dane z czujnika temperatury
  - przetwarza odczyty temperatury obliczając regułę sterowania
  - określa prawidłowe natężenia przepływu paliwa, powietrza i chłodziwa
- Fakt, że reguła sterowania jest obliczana okresowo można wyrazić w terminach czasów uwolnienia prac $J_0, J_1, \ldots, J_k, \ldots$ obliczających tę regułę:

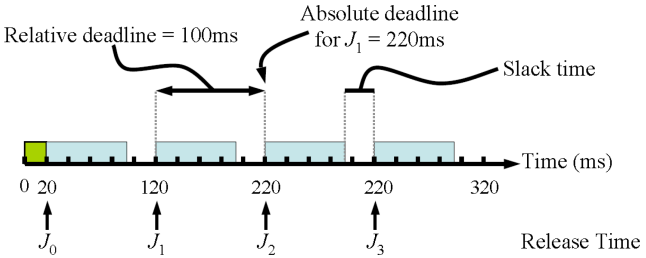
$$r_k = 20 + 100k \quad \text{(ms)}$$



Ograniczenia czasowe - przykład cz. 2

- Każda praca musi zostać ukończona przed uwolnieniem następnej pracy:
  - względny czas ostateczny pracy J_k wynosi: $D_k = 100$ ms
  - bezwzględny czas ostateczny pracy J_k wynosi $d_k = 20 + 100(k + 1)$ ms
- Zwykle wymaga się, aby każde obliczenie reguły sterowania zakończyło się wcześniej, to znaczy względny czas ostateczny powinien być krótszy niż czas między kolejnymi pracami, co daje pewien zapas czasu (*slack time*) dla innych zadań

$$\text{slack time} = \text{relative deadline - execution time}$$



Twarde i miękkie systemy czasu rzeczywistego

.

Hard and soft real-time systems

Klasyfikacja ograniczeń czasowych

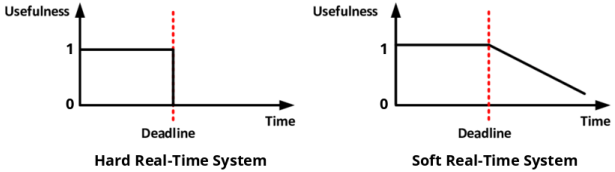
- *System czasu rzeczywistego* to system, w którym poprawność działania zależy od:
  - logicznego wyniku obliczeń
  - czasu, w którym wyniki te są generowane
- System czasu rzeczywistego ma *ograniczenia czasowe* wykonania obliczeń i działań
- Ograniczenia czasowe dzieli się na dwa typy:
  - *twarde* (*hard*)
  - *miękkie* (*soft*)
- Istnieją różne definicje twardych i miękkich ograniczeń czasowych oparte na ocenie:
  - jak bardzo dana praca jest funkcjonalnie krytyczna
  - przydatności wyników
  - prawdopodobieństwa niedotrzymania terminów

Ograniczenia czasowe – funkcjonalna krytyczność

- Ograniczenie czasowe jest twarde, gdy jego niedotrzymanie jest błędem krytycznym
- Spóźniony wynik wyprodukowany przez pracę może mieć katastrofalne konsekwencje:
  - spóźnione polecenie zatrzymania pociągu może spowodować kolizję
  - bomba zrzucona zbyt późno może uderzyć w cywili zamiast w cel wojskowy
  - opóźnienie w wyznaczeniu pozycji statku kosmicznego może doprowadzić do katastrofy
- Późne ukończenie pracy może być niepożądane, ale nie katastrofalne:
  - spóźnione zakodowanie pakietu audio w telefonii VOIP spowoduje zniekształcenia w odbiorze
  - spóźnione odkodowanie ramki video spowoduje chwilowe zamazanie obrazu
  - opóźnienie w wyznaczeniu pozycji samochodu może utrudniać nawigację
- Taka definicja twardych i miękkich ograniczeń czasowych prowadzi do pytania, jak poważne są konsekwencje niedotrzymania terminu ostatecznego
- Pytanie czy ograniczenie czasowe jest twarde czy miękkie, sprowadza się do pytania, na ile poważne jest poważne (ilość cukru w cukrze)

Ograniczenia czasowe – przydatność wyników

- Funkcja *opóźnienia* pracy mierzy jak późno ta praca została ukończona względem jej terminu ostatecznego
- Funkcja opóźnienia przyjmuje wartość 0, gdy praca zostanie ukończona w terminie lub przed nim, w przeciwnym razie jest to różnica między czasem ukończenia a terminem ostatecznym
- Wraz ze wzrostem opóźnienia *przydatność* (*usefulness*) wyniku uzyskanego w ramach pracy z miękkim terminem ostatecznym stopniowo maleje, natomiast użyteczność wyniku uzyskanego w ramach pracy z twardym terminem ostatecznym spada gwałtownie



- Trudno jest zweryfikować, czy miara ogólnej wydajności systemu jako funkcji opóźnienia zachowuje się zgodnie ze specyfikacją funkcji użyteczności
- W konsekwencji tego rodzaju miary ilościowe nie są tak rygorystyczne, jak się wydaje

Ograniczenia czasowe – prawdopodobieństwo

- Ograniczenie czasowe jest:
  - twarde : nigdy nie można przekroczyć terminu ostatecznego (prawdopodobieństwo niedotrzymania terminu $= 0$)
  - miękkie : termin może być czasami przekroczony z pewnym akceptowalnie niskim prawdopodobieństwem (prawdopodobieństwo niedotrzymania terminu > 0)
- Załóżmy, że zadanie polegające na przywróceniu systemu oraz transakcja w punkcie sprzedaży kończą się w ciągu jednej minuty w 99,999% przypadków
- Prawdopodobieństwo niedotrzymania względnego terminu jednej minuty wynosi 10^{-5}
- Taka definicja całkowicie ignoruje konsekwencje niedotrzymania terminu
- W przypadku przywracania systemu, jeśli niedotrzymanie terminu mogłoby spowodować utratę życia albo zniszczenie mienia, wymagalibyśmy dokładnego wykazania, że to prawdopodobieństwo nigdy nie przekracza 10^{-5}
- W przypadku walidacji karty kredytowej nie wymagalibyśmy takiego dowodu

Twarde i miękkie ograniczenia czasowe

- Ograniczenie czasowe jest *twarde* (*hard real-time job*), jeśli użytkownik wymaga potwierdzenia (*validation*), że system zawsze spełnia ograniczenie czasowe
- Przez *potwierdzenie* rozumiemy dowód przy pomocy sprawdzonej, poprawnej i wydajnej procedury lub poprzez wyczerpującą symulację i testowanie
- Ograniczenie czasowe jest *miękkie* (*soft real-time job*), jeśli nie jest wymagane takie potwierdzenie lub wystarczy jedynie dowód, że praca spełnia pewne ograniczenie statystyczne, np.: średnia liczba niedotrzymanych terminów na minutę nie przekracza 2

Twarde ograniczenia czasowe

- Twarde ograniczenia czasowe można określać na wiele sposobów:
  deterministycznie
  - względny termin ostateczny każdego obliczenia reguły sterowania wynosi 50ms
  - czas reakcji co najwyżej 1 z 5 kolejnych obliczeń reguły sterowania przekracza 50ms

  probabilistycznie
  - prawdopodobieństwo przekroczenia czasu reakcji 50ms jest nie większe niż $0,2$
  - prawdopodobieństwo niedotrzymania terminu ostatecznego nie przekracza $0,005$

  użytecznościowo
  - użyteczność każdego obliczenia reguły sterowania wynosi przynajmniej $0,8$
  - użyteczność co najwyżej 1 z 5 kolejnych obliczeń reguły sterowania może spaść do $0,9$
- Najskuteczniej możemy potwierdzić deterministyczne ograniczenia czasowe
- Jeśli znane są wszelkie konsekwencje przekroczenia twardych ograniczeń czasowych i wiadomo, że nie prowadzą one do katastrofy, to można osłabić te ograniczenia
- Gdy nie da się udowodnić ponad wszelką wątpliwość, że naruszenie ograniczeń czasowych nie zagrozi bezpieczeństwu użytkowników lub dostępności infrastruktury krytycznej, stosujemy bezpieczne podejście i wymagamy spełnienia wszystkich ograniczeń czasowych, nawet jeśli wymagania te mogą być niepotrzebnie rygorystyczne

Twarde i miękkie systemy czasu rzeczywistego

Twardy system czasu rzeczywistego : (*hard real-time system*) to system, w którym większość ograniczeń czasowych jest twarda

- kontroler hamulców w pociągu (czas pokonania drogi hamowania)
- kontroler lotu (flight controller)
- układ ABS zapobiegający blokowaniu kół

Miękki system czasu rzeczywistego : (*soft real-time system*) to system, w którym ograniczenia czasowe w większości są miękkie

- system notowań cen akcji (krytyczne wymagania czasowe, kompromis)
- usługa VOD (średnia ilość utraconych klatek na minutę wynosi mniej niż 2)
- sieć telefoniczna (połączenie w 10s w 95% przypadków, w 20s w 99,95% przypadków)

Walidacja systemów czasu rzeczywistego

Walidacja :

- potwierdzenie poprzez dostarczenie obiektywnego dowodu spełnienia określonych wymagań odnośnie do konkretnego użycia lub zastosowania
- ogół czynności mających na celu zbadanie odpowiedniości, trafności lub dokładności

Walidacja systemu czasu rzeczywistego przebiega w trzech krokach:

1. Weryfikacja, czy ograniczenia czasowe są określone poprawnie
2. Weryfikacja, czy każdy komponent systemu spełnia swoje ograniczenia czasowe, gdy wykonuje się samodzielnie i dysponuje wszystkimi wymaganymi zasobami
3. Weryfikacja systemu jako całości – zapewnienie spełnialności wszelkich ograniczeń czasowych wszystkich aplikacji, które kolejkowane są zgodnie z algorytmami używanymi przez system operacyjny i sieci komunikacyjne oraz które konkurują o zasoby

Przewidywalność systemów komputerowych

Przewidywalność systemów komputerowych

- Najważniejszą cechą systemu czasu rzeczywistego jest *przewidywalność*
- System powinien być w stanie:
  - przewidywać jak zadania będą ewoluować
  - z góry zagwarantować spełnialność wszystkich ograniczeń czasowych
- Spełnialność ograniczeń czasowych zależy od wielu czynników sprzętowych i programowych
- Wewnętrzne cechy procesora jak: *instruction prefetch*, *pipelining*, *cache memory* poprawiają wydajność, ale powodują *niedeterminizm* systemu

Przewidywalność: Direct Memory Access - DMA

- Urządzenia peryferyjne przesyłają dane bezpośrednio do pamięci głównej RAM
- Odciążenie procesora CPU od kontroli transferu wejścia/wyjścia
- Procesor i urządzenie DMA współdzielą szynę danych (*cycle stealing*)
- Transfer wejścia/wyjścia oraz program na procesorze wykonywane są równolegle
- Gdy urządzenie DMA i procesor sięgają do pamięci w tym samym czasie, szyna danych przypisywana jest do urządzenia DMA a procesor oczekuje do następnego cyklu
- Nie jesteśmy w stanie przewidzieć jak długo procesor będzie czekać na urządzenie DMA
- Nie jesteśmy w stanie precyzyjnie określić czasu reakcji zadania (*response time*)
- Możliwym rozwiązaniem jest zastosowanie techniki *time-slice*, w której jeden cykl dostępu do pamięci dzielony jest na dwa sąsiednie przedziały czasowe: jeden dla procesora, drugi dla urządzenia DMA

Przewidywalność: pamięć podręczna (cache)

- Szybka pamięć wstawiana jako bufor pomiędzy procesor CPU i pamięć główną RAM
- Pamięć podręczna nie jest zauważalna z poziomu programu uzytkownia
- Po ustaleniu adresu fizycznego lokalizacji pamięci sprawdzane jest, czy żądane informacje są przechowywane w pamięci podręcznej: jeśli tak, dane są odczytywane z pamięci podręcznej, w przeciwnym razie informacje są pobierane z pamięci RAM i kopiowane do pamięci podręcznej wraz z sąsiednimi regionami
- Satystycznie przy 1MB RAM i 8kB cache, dane żądane przez program w 80 przypadkach na 100 znajdują się w pamięci podręcznej (*program locality*, *hit ratio*)
- Obserwacje statystyczne pozwalają jedynie na oszacowanie przeciętnego zachowania aplikacji, ale nie mogą służyć do wyznaczania czasu reakcji zadania (*response time*)

Przewidywalność: przerwania (interrupts)

- W systemie operacyjnym nadejście sygnału przerwania powoduje wykonanie procedury sterownika (*driver*) dedykowanej do zarządzania powiązanym z nim urządzeniem
- Na ogół przerwania są obsługiwane na podstawie ustalonego schematu priorytetów: sterowniki mają statyczny priorytet, wyższy niż priorytety zwykłych procesów, gdyż sterowniki mają ograniczenia czasu rzeczywistego, podczas gdy aplikacje ich nie mają
- Przerwania generowane przez urządzenia peryferyjne, jeśli nie są odpowiednio obsłużone, mogą wprowadzać nieograniczone opóźnienia w trakcie wykonywania programu
- Ilość przerwań oraz opóźnienie wykonania zadania nimi spowodowane są nieprzewidywalne
- Możliwe rozwiązania:
  - Wszystkie zewnętrzne przerwania, poza zagarem (*timer*), są wyłączone, urządzenia perfyferyjne obsługiwane są przez aplikację
  - Wszystkie zewnętrzne przerwania, poza zagarem (*timer*), są wyłączone, urządzenia perfyferyjne obsługiwane są przez dedykowane moduły jądra okresowo aktywowane zegarem
  - Zewnętrzne przerwania są włączone, sterowniki ograniczają się do aktywowania odpowiednich zadań znajdujących się pod kontrolą systemu operacyjnego dzięki czemu priorytety tych zadań mogą być ustwiane w zależności od wymagań innych aplikacji

Przewidywalność: funkcje systemowe (system calls)

- Aby precyzyjnie określić najgorszy czas wykonania zadania, wszystkie wywołania jądra powinny mieć znany czas ich realizacji
- Kod, którego nie można wywłaszczyć (przejąć), może opóźnić aktywację lub wykonanie krytycznych czynności
- Wszystkie funkcje systemowe powinny być wywłaszczalne (*preemptable*)

Przewidywalność: semafory

- Ograniczeniem machanizmu semaforów to odwrócenie priorytetów (*priority inversion*)
- Odrócenie priorytetów polega na tym, że zadanie o wysokim priorytecie czeka na zadanie o niskim priorytecie
- System staje się nieprzewidywalny z powodu niedeterministycznych opóźnień wykonania zadań w wyniku odwrócenia priorytetów
- Rozwiązaniem jest stosowanie protokołów, które modyfikują priorytet zadania w zależności od bieżącego użycia zasobów i kontrolują dysponowanie zasobami za pomocą sprawdzeń wykonywanych przed każdym wejściem do sekcji krytycznych (*critical sections*)

Przewidywalność: zarządzanie pamięcią

- Procesy widzą *pamięć wirtualną* (*virtual memory*), którą zarządza system operacyjny
- Pamięć wirtualna składa się z pamięci fizycznej RAM oraz przestrzeni (*swap*) na nośniku
- Aby szybciej odnajdować dane w pamięci wirtualnej stosuje się *stronicowanie* (*pages*)
- *Page fault* powoduje wczytanie odpowiedniej *ramki* (*page frame*) z nośnika do pamięci RAM — i dla lepszego wykorzystania zasobów — wykonywane jest w tym czasie *przełączanie kontekstowe* (*context switching*)
- Zarządzenia pamięcią jest niedeterministyczne z uwagi na nieprzewidywalne opóźnienia spowodowane przez page faults

Przewidywalność: język programowania

- Real-Time Euclid
- Real-Time Concurrent C

```
> FIN < ACK < FIN > ACK

init 5
```