Schedulingkriterien

- Kriterien in Standardsystemen sind:
  - Fairness: gerechte Verteilung der Prozessorzeit
  - Effizienz: vollständige Auslastung der CPU
  - Antwortzeit: interaktive Prozesse sollen schnell reagieren
  - Verweilzeit: Aufgaben im Batchbetrieb (sequentielle Abarbeitung von Aufträgen) sollen möglichst schnell ein Ergebnis liefern
  - Durchsatz: Maximierung der Anzahl der Aufträge, die innerhalb einer bestimmten Zeitspanne ausgeführt werden
- In Echtzeitsystemen:
  - Einhaltung der Fristen: d.h. $\forall i\ c_i<d_i$ unter Berücksichtigung von Kausalzusammenhängen (Synchronisation, Vorranggraphen, Präzedenzsystemen)
  - Determinismus des Verfahrens
  - Zusätzliche Kriterien können anwendungsabhängig hinzugenommen werden, solange sie der Einhaltung der Fristen untergeordnet sind.

Scheduling

Verfahren

Allgemeines Verfahren

- Gesucht: Plan mit aktueller Start und Endzeit für jeden Prozess P_i.
- Darstellung zum Beispiel als nach der Zeit geordnete Liste von Tupeln (P_i, s_i, c_i)
- Falls Prozesse unterbrochen werden können, so kann jedem Prozess P_i auch eine Menge von Tupeln zugeordnet werden.
- Phasen der Planung:
  - Test auf Einplanbarkeit (feasibility check)
  - Planberechnung (schedule construction)
  - Umsetzung auf Zuteilung im Betriebssystem (dispatching)
- Bei Online-Verfahren können die einzelnen Phasen überlappend zur Laufzeit ausgeführt werden.
- Zum Vergleich von Scheduling-Verfahren können einzelne Szenarien durchgespielt werden.

Definitionen

- **Zulässiger Plan**: Ein Plan ist zulässig, falls alle Prozesse einer Prozessmenge eingeplant sind und dabei keine Präzedenzrestriktionen und keine Zeitanforderungen verletzt werden.
- **Optimales Planungsverfahren**: Ein Verfahren ist optimal, falls es für jede Prozessmenge unter gegebenen Randbedingung einen zulässigen Plan findet, falls ein solcher existiert.

Test auf Einplanbarkeit

- Zum Test auf Einplanbarkeit können zwei Bedingungen angegeben werden, die für die Existenz eines zulässigen Plans notwendig sind (Achtung: häufig nicht ausreichend):
  1. $r_i + e_i \leq d_i$, d.h. jeder Prozess muss in dem Intervall zwischen Bereitzeit und Frist ausgeführt werden können.
  2. Für jeden Zeitraum $[t_i, t_j]$ muss die Summe der Ausführungszeiten e_x der Prozesse P_x mit $r_x \geq t_i \wedge d_x \leq$ tj kleiner als der Zeitraum sein.
- Durch weitere Rahmenbedingungen (z.B. Abhängigkeiten der einzelnen Prozesse) können weitere Bedingungen hinzukommen.

Schedulingverfahren

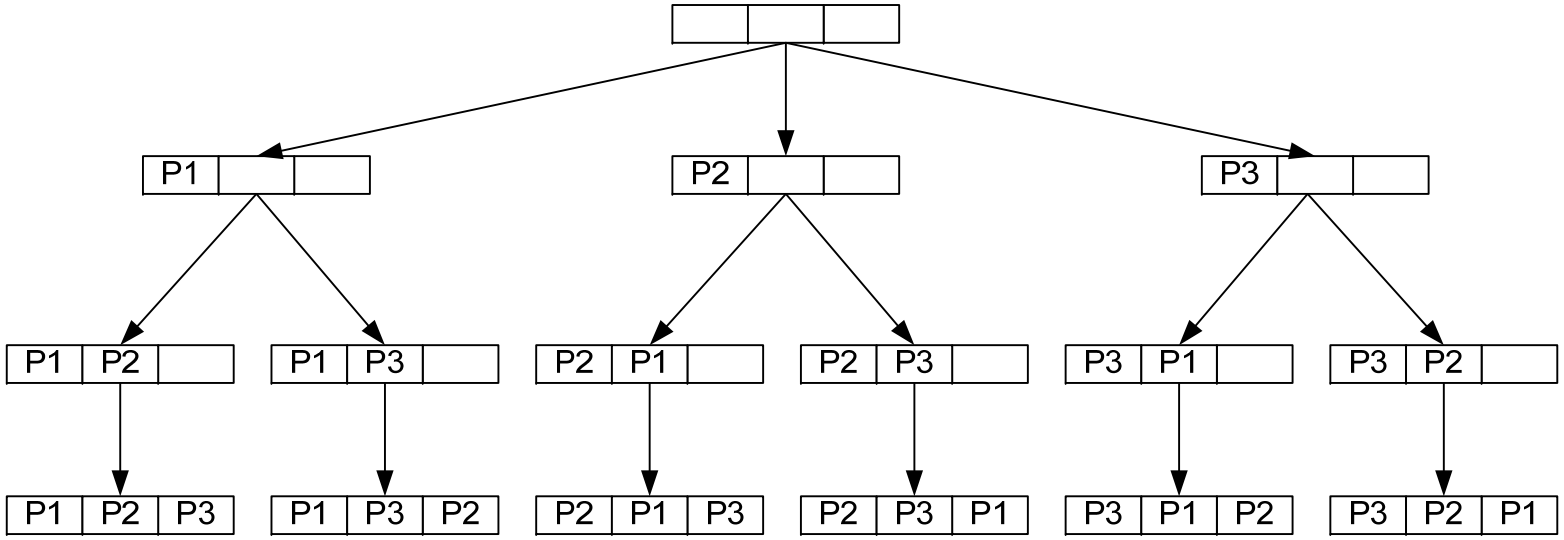
- Planen aperiodischer Prozesse
  - Planen durch Suchen
  - Planen nach Fristen
  - Planen nach Spielräumen
- Planen periodischer Prozesse
  - Planen nach Fristen
  - Planen nach Raten
- Planen abhängiger Prozesse

Scheduling

Scheduling-Verfahren für 1-Prozessor-Systeme

Planen durch Suchen

- Betrachtung: ununterbrechbare Aktionen/Prozesse vorausgesetzt
- Lösungsansatz: exakte Planung durch Durchsuchen des Lösungsraums
- Beispiel:
  - n=3 Prozesse P_1,P_2,P_3 und 1 Prozessor
  - Suchbaum:



Problem: Komplexität

- n! Permutationen müssen bewertet werden, bei Mehrprozessorsystemen ist das Problem der Planung NP-vollständig
- Durch präemptives Scheduling bzw. durch unterschiedliche Bereitzeiten kann das Problem weiter verkompliziert werden.
- Die Komplexität kann durch verschiedene Maßnahmen leicht reduziert werden:
  - Abbrechen von Pfaden bei Verletzung von Fristen
  - Verwendung von Heuristiken: z.B. Sortierung nach Bereitstellzeiten r_i
- Prinzipiell gilt jedoch: **Bei komplexen Systemen ist Planen durch Suchen nicht möglich**.

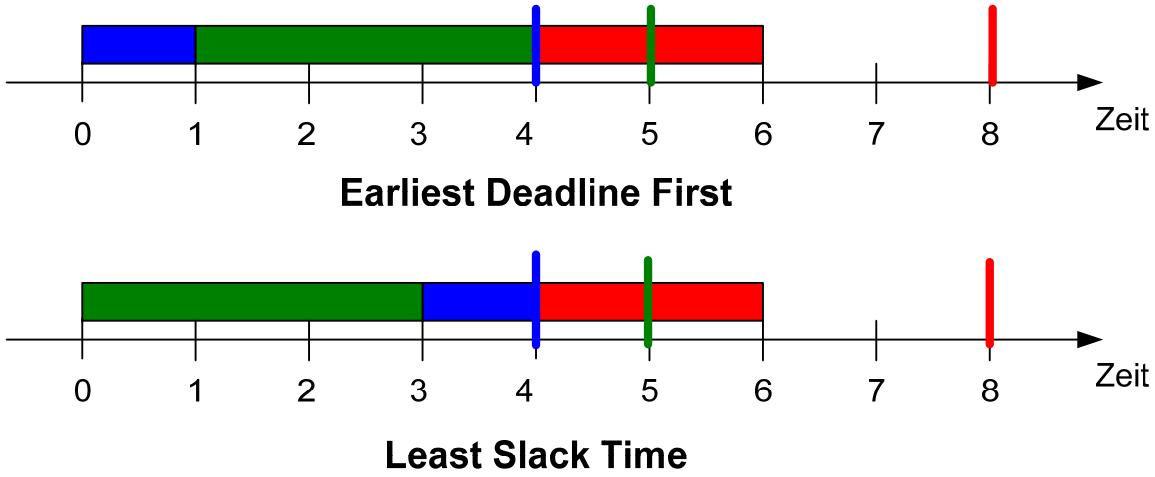
Scheduling-Strategien (online, nicht-präemptiv) für Einprozessorsysteme

1. EDF: Einplanen nach Fristen (Earliest Deadline First): Der Prozess, dessen Frist als nächstes endet, erhält den Prozessor.
2. LST: Planen nach Spielraum (Least Slack Time): Der Prozess mit dem kleinsten Spielraum erhält den Prozessor.
   - Der Spielraum berechnet sich wie folgt:
     Deadline-(aktuelle Zeit + verbleibende Berechnungszeit)
   - Der Spielraum für den aktuell ausgeführten Prozess ist konstant.
   - Die Spielräume aller anderen Prozesse nehmen ab.

- Vorteil und Nachteile:
  - LST erkennt Fristverletzungen früher als EDF.
  - Für LST müssen die Ausführungszeiten der Prozesse bekannt sein.

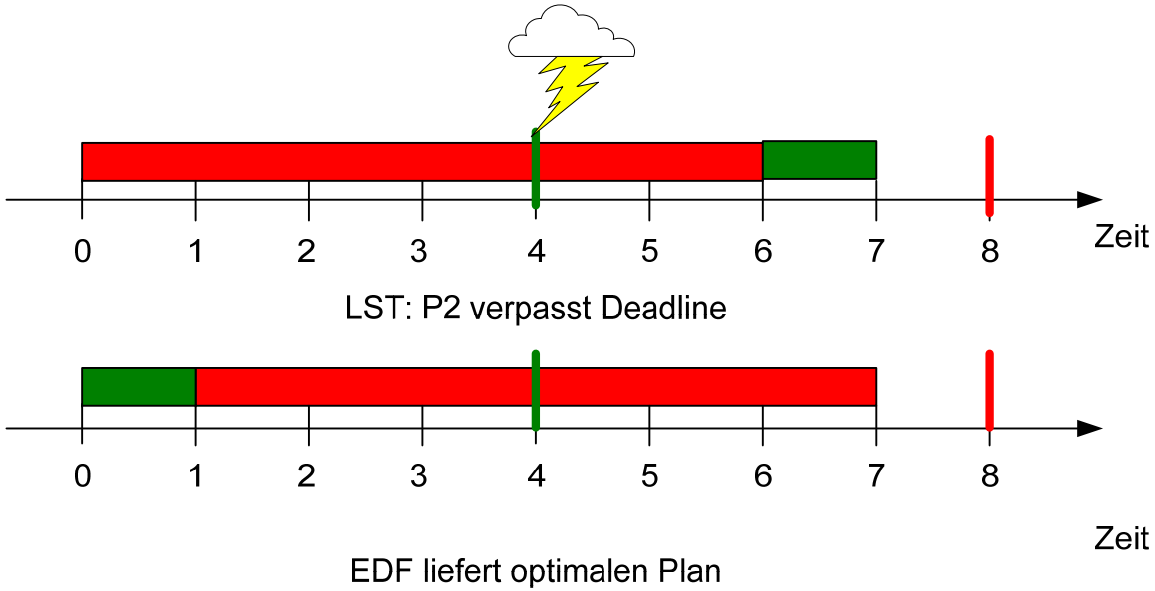
Beispiel

- 3 Prozesse:
  P_1: r_1=0; e_1=2; d_1=8;
  P_2: r_2=0; e_2=3; d_2=5;
  P_3: r_3=0; e_3=1; d_3=4;



Versagen von LST

- LST kann selbst bei gleichen Bereitzeiten im nicht-präemptiven Fall versagen.
- 2 Prozesse:
  P_1: r_1=0; e_1=6; d_1=8;
  P_2: r_2=0; e_2=1; d_2=4;



- Anmerkung: Aus diesem Grund wird LST nur in präemptiven Systemen eingesetzt. Bei Prozessen mit gleichen Spielräumen wird einem Prozess Δ eine Mindestausführungszeit garantiert.

Optimalität von EDF

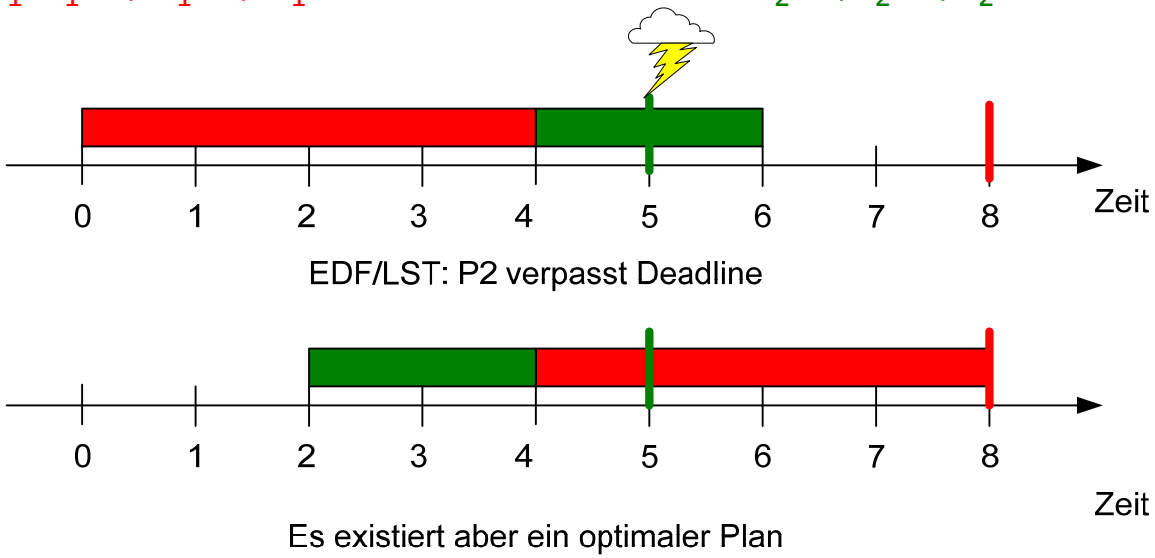
- Unter der Voraussetzung, dass alle Prozesse P_i eine Bereitzeit $r_i=0$ besitzen und das ausführende System ein Einprozessorsystem ist, ist EDF optimal, d.h. ein zulässiger Plan wird gefunden, falls ein solcher existiert.
- Beweisidee für EDF: Tausch in existierendem Plan
  - Sie $Plan_x$ ein zulässiger Plan.
  - Sei $Plan_{EDF}$ der Plan, der durch die EDF-Strategie erstellt wurde.
  - Ohne Einschränkung der Allgemeinheit: die Prozessmenge sei nach Fristen sortiert, d.h. $d_i \leq d_j$ für i<j.
  - Idee: Schrittweise Überführung des Planes $Plan_x$ in $Plan_{EDF}$
    - $P(Plan_x,t)$ sei der Prozess, der von $Plan_x$ zum Zeitpunkt t ausgeführt wird.
    - $Plan_x(t)$ ist der bis zum Zeitpunkt t in $Plan_{EDF}$ überführte Plan ($\Rightarrow Plan_x(0)=Plan_x$).

Fortsetzung des Beweises

- Wir betrachten ein Zeitinterval Δ_t.
- Zum Zeitpunkt t gilt:
  $i=P(Plan_{EDF},t)$
  $j=P(Plan_x,t)$
- Nur der Fall j>i ist interessant. Es gilt:
  - $d_i \leq d_j$
  - $t+\Delta_t \leq d_i$ (ansonsten wäre der $Plan_x$ nicht zulässig)
  - Da die Pläne bis zum Zeitpunkt t identisch sind und P_i im $Plan_{EDF}$ zum Zeitpunkt t ausgeführt sind, kann der Prozess P_i im $Plan_x$ noch nicht beendet sein.
    $\Rightarrow \exists\, t'>t+\Delta_t: (i=P(Plan_x,t')=P(Plan_x,t'+\Delta_t) \wedge t'+\Delta_t \leq d_i \leq d_j$
    $\Rightarrow$ Die Aktivitätsphase von P_i im Zeitintervall $t'+\Delta_t$ und P_j im Zeitintervall $t+\Delta_t$ können ohne Verletzung der Zeitbedingungen getauscht werden $\Rightarrow$ Übergang von $Plan_x(t)$ zu $Plan_x(t+\Delta_t)$

Versagen von EDF bei unterschiedlichen Bereitzeiten

- Haben die Prozesse unterschiedliche Bereitzeiten, so kann EDF versagen.
- Beispiel: P_1: $r_1=0$; $e_1=4$; $d_1=8$ P2:$r_2=2$;$e_2=2$;$d_2=5$



- **Anmerkung**: Jedes prioritätsgesteuerte, **nicht präemptive** Verfahren versagt bei diesem Beispiel, da ein solches Verfahren nie eine Zuweisung des Prozessors an einen laufbereiten Prozess , falls ein solcher vorhanden ist, unterlässt.

Modifikationen

- Die Optimalität der Verfahren kann durch folgende Änderungen sichergestellt werden:
  - Präemptive Strategie
  - Neuplanung beim Erreichen einer neuen Bereitzeit
  - Einplanung nur derjenigen Prozesse, deren Bereitzeit erreicht ist

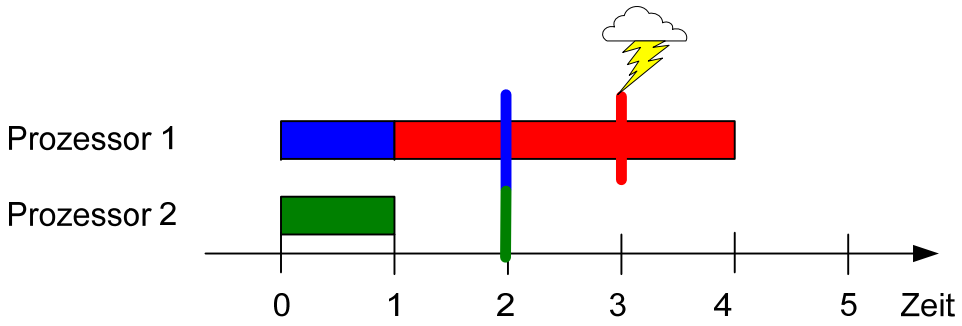
  → Entspricht einer Neuplanung, falls ein Prozess aktiv wird.
- Bei Least Slack Time müssen zusätzlich Zeitscheiben für Prozesse mit gleichem Spielraum eingeführt werden, um ein ständiges Hin- und Her Schalten zwischen Prozessen zu verhindern.
- Generell kann gezeigt werden, dass die Verwendung von EDF die Anzahl der Kontextwechsel in Bezug auf Online-Scheduling-Verfahren minimiert (siehe Paper von Buttazzo)

Zeitplanung auf Mehrprozessorsystemen

- Fakten zum Scheduling auf Mehrprozessorsystemen (Beispiele folgen):
  - EDF nicht optimal, egal ob präemptiv oder nicht präemptive Strategie
  - LST ist nur dann optimal, falls alle Bereitzeitpunkte r_i gleich und präemptive Strategie gewählt wird
  - korrekte Zuteilungsalgorithmen erfordern das Abarbeiten von Suchbäumen mit NP-Aufwand oder geeignete Heuristiken
  - Beweisidee zur Optimalität von LST bei gleichen Bereitzeitpunkten: Der Prozessor wird immer dem Prozess mit geringstem Spielraum zugewiesen , d.h. wenn bei LST eine Zeitüberschreitung auftritt, dann auch, falls die CPU einem Prozess mit größerem Spielraum zugewiesen worden wäre.

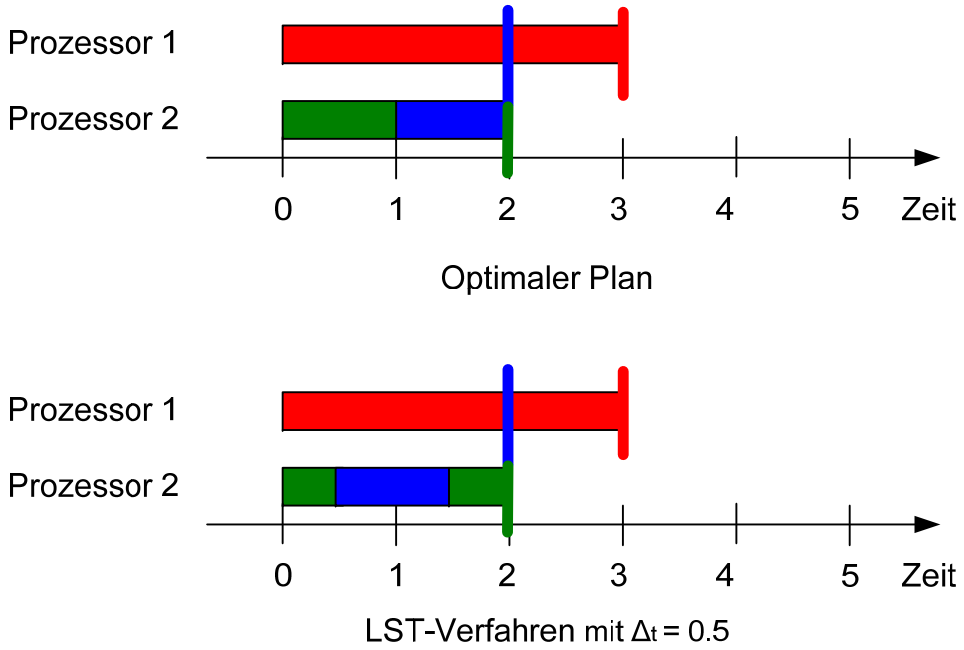
Beispiel: Versagen von EDF

- 2 Prozessoren, 3 Prozesse:
  P_1: r_1=0; e_1=3; d_1=3;
  P_2: r_2=0; e_2=1; d_2=2;
  P_3: r_3=0; e_3=1; d_3=2;



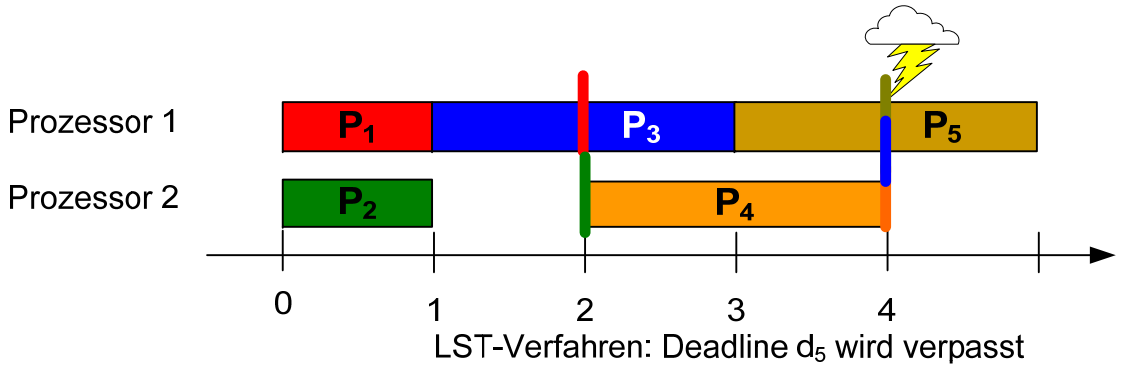
EDF-Verfahren: Deadline d1 wird verpasst

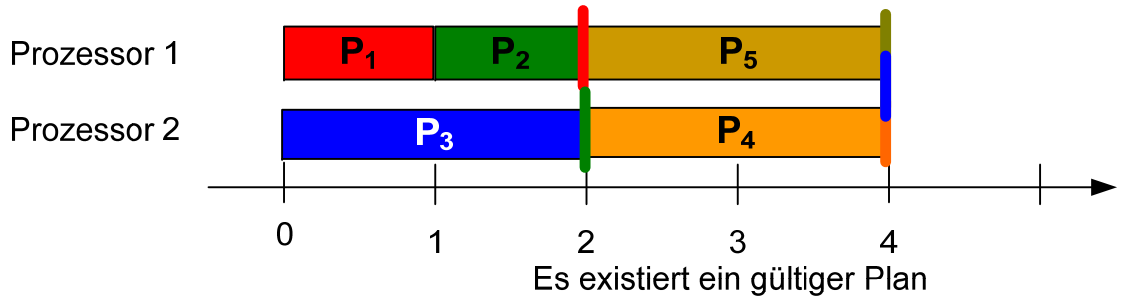
Beispiel: Optimaler Plan und LST-Verfahren



Beispiel: Versagen von LST

- 2 Prozessoren, 5 Prozesse, Δ_t=0,5:
  P_1: r_1=0; e_1=1; d_1=2;
  P_2: r_2=0; e_2=1; d_2=2;
  P_3: r_3=0; e_3=2; d_3=4;
  P_4: r_4=2; e_4=2;d_4=4;
  P_5: r_5=2; e_5=2;d_5=4;





Versagen von präemptiven Schedulingverfahren

- Jeder präemptiver Algorithmus versagt, wenn die Bereitstellzeiten unterschiedlich sind und nicht im Voraus bekannt sind.
  Beweis:
  - n CPUs und n-2 Prozesse ohne Spielraum (n-2 Prozesse müssen sofort auf n-2 Prozessoren ausgeführt werden) $\Rightarrow$ Reduzierung des Problems auf 2-Prozessor-Problem
  - Drei weitere Prozesse sind vorhanden und müssen eingeplant werden.
  - Die Reihenfolge der Abarbeitung ist von der Strategie abhängig, in jedem Fall kann aber folgender Fall konstruiert werden, so dass:
    - es zu einer Fristverletzung kommt,
    - aber ein gültiger Plan existiert.

Fortsetzung Beweis

- Szenario:
  P_1: r_1=0; e_1=1; d_1=1;
  P_2: r_2=0; e_2=2; d_2=4;
  P_3: r_3=0; e_3=1; d_3=2;

  ⟶ Prozess P_1 (kein Spielraum) muss sofort auf CPU1 ausgeführt werden.

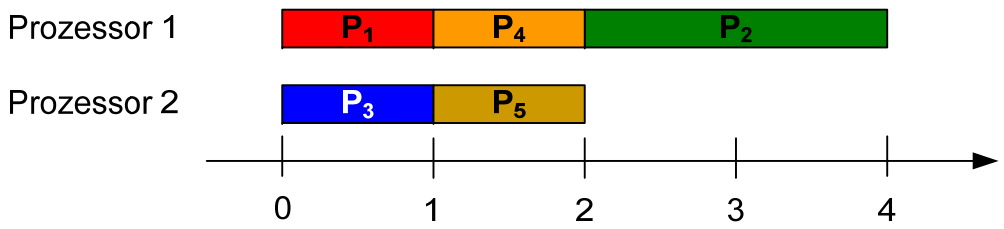
  ⟶ Es gibt je nach Strategie zwei Fälle zu betrachten: P2 oder P3 wird zunächst auf CPU2 ausgeführt.

1. Fall

- P_2 wird zum Zeitpunkt 0 auf CPU2 ausgeführt.
  - Zum Zeitpunkt 1 muss dann P_3 (ohne Spielraum) ausgeführt werden.
  - Zum Zeitpunkt 1 treffen aber zwei weitere Prozesse P_4 und P_5 mit Frist 2 und Ausführungsdauer 1 ein.

  → Es gibt drei Prozesse ohne Spielraum, aber nur zwei Prozessoren.

- Aber es gibt einen gültigen Ausführungsplan:

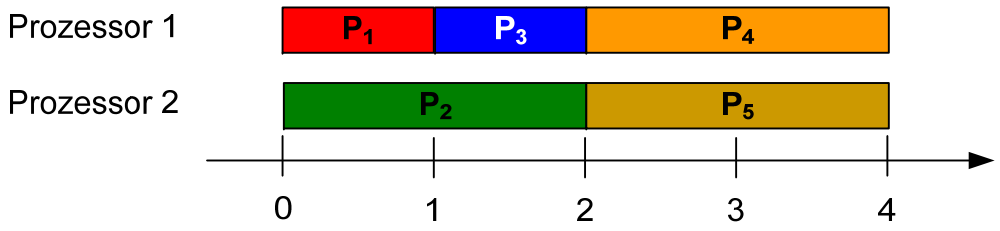


2. Fall

- P_3 wird zum Zeitpunkt 0 auf CPU2 ausgeführt.
  - Zum Zeitpunkt 1 sind P_1 und P_3 beendet.
  - Zum Zeitpunkt 1 beginnt P_2 seine Ausführung.
  - Zum Zeitpunkt 2 treffen aber zwei weitere Prozesse P_4 und P_5 mit Deadline 4 und Ausführungsdauer 2 ein.

  $\Rightarrow$ Anstelle der zum Zeitpunkt 2 noch notwendigen 5 Ausführungseinheiten sind nur 4 vorhanden.

- Aber es gibt einen gültigen Ausführungsplan:



Strategien in der Praxis

- Die Strategien EDF und LST werden in der Praxis selten angewandt. Gründe:
  - In der Realität sind keine abgeschlossenen Systeme vorhanden (Alarme, Unterbrechungen erfordern eine dynamische Planung)
  - Bereitzeiten sind nur bei zyklischen Prozessen oder Terminprozessen bekannt.
  - Die Abschätzung der Laufzeit sehr schwierig ist (siehe Exkurs).
  - Synchronisation, Kommunikation und gemeinsame Betriebsmittel verletzen die Forderung nach Unabhängigkeit der Prozesse.

Ansatz in der Praxis

- Zumeist basiert das Scheduling auf der Zuweisung von statischen Prioritäten.
- Prioritäten werden zumeist durch natürliche Zahlen zwischen 0 und 255 ausgedrückt. Die höchste Priorität kann dabei sowohl 0 (z.B. in VxWorks) als auch 255 (z.B. in POSIX) sein.
- Die Priorität ergibt sich aus der Wichtigkeit des technischen Prozesses und der Abschätzung der Laufzeiten und Spielräume. Die Festlegung erfolgt dabei durch den Entwickler.
- Bei gleicher Priorität wird zumeist eine FIFO-Strategie (d.h. ein Prozess läuft solange, bis er entweder beendet ist oder aber ein Prozess höherer Priorität eintrifft) angewandt.
  Alternative Round Robin: Alle laufbereiten Prozesse mit der höchsten Priorität erhalten jeweils für eine im Voraus festgelegte Zeitdauer die CPU.

Scheduling

Zeitplanen periodischer Prozesse

Zeitplanung periodischer Prozesse

- Annahmen für präemptives Scheduling
  - Alle Prozesse treten periodisch mit einer Frequenz f_i auf.
  - Die Frist eines Prozesses entspricht dem nächsten Startpunkt.
  - Sind die maximalen Ausführungszeiten e_i bekannt, so kann leicht errechnet werden, ob ein ausführbarer Plan existiert.
  - Die für einen Prozesswechsel benötigten Zeiten sind vernachlässigbar.
  - Alle Prozesse sind unabhängig.
- Eine sehr gute Zusammenfassung zu dem Thema Zeitplanung periodischer Prozesse liefert Giorgio C. Buttazzo in seinem Paper „Rate Monotonic vs. EDF: Judgement Day“ (http://www.cas.mcmaster.ca/~downd/rtsj05-rmedf.pdf).

Einplanbarkeit

- Eine notwendige Bedingung zur Einplanbarkeit ist die Last:
  - Last eines einzelnen Prozesses: $\rho_i = e_i * f_i$
  - Gesamte Auslastung bei n Prozessen:

$$\rho = \sum_{i=0}^{n} \rho_i$$

  - Bei m Prozessoren ist $\rho < m$ eine notwendige aber nicht ausreichende Bedingung.

Zeitplanen nach Fristen

- **Ausgangspunkt**: Wir betrachten Systeme mit einem Prozessor und Fristen der Prozesse, die relativ zum Bereitzeitpunkt deren Perioden entsprechen, also $d_i=1/f_i$.
- **Aussage**: Die Einplanung nach Fristen ist optimal.
- **Beweisidee**: Vor dem Verletzen einer Frist ist die CPU nie unbeschäftigt $\Rightarrow$ die maximale Auslastung liegt bei 100%.
- Leider wird aufgrund von diversen Vorurteilen EDF selten benutzt.
- Betriebssysteme unterstützen selten ein EDF-Scheduling
  $\Rightarrow$ Die Implementierung eines EDF-Scheduler auf der Basis von einem prioritätsbasierten Scheduler ist nicht effizient zu implementieren (Ausnahme: zeitgesteuerte Systeme)

Zeitplanung nach Raten

- Rate Monotonic bezeichnet ein Scheduling-Verfahren mit festen Prioritäten Prio(i), die sich proportional zu den Frequenzen verhalten.

  → Prozesse mit hohen Raten werden bevorzugt. Das Verfahren ist optimal, falls eine Lösung mit statischen Prioritäten existiert. Verfahren mit dynamischen Prioritäten können allerdings eventuell bessere Ergebnisse liefern.

- Liu und Layland haben 1973 in einer Worst-Case-Analyse gezeigt, dass Ratenplanung sicher erfolgreich ist, falls bei n Prozessen auf einem Prozessor gilt:

$$\rho \leq \rho_{max} = n*(2^{1/n} - 1)$$

$$\lim_{n \to \infty} \rho_{max} = \ln 2 \approx 0{,}69$$

- Derzeit zumeist verwendetes Scheduling-Verfahren im Bereich von periodischen Prozessen.