

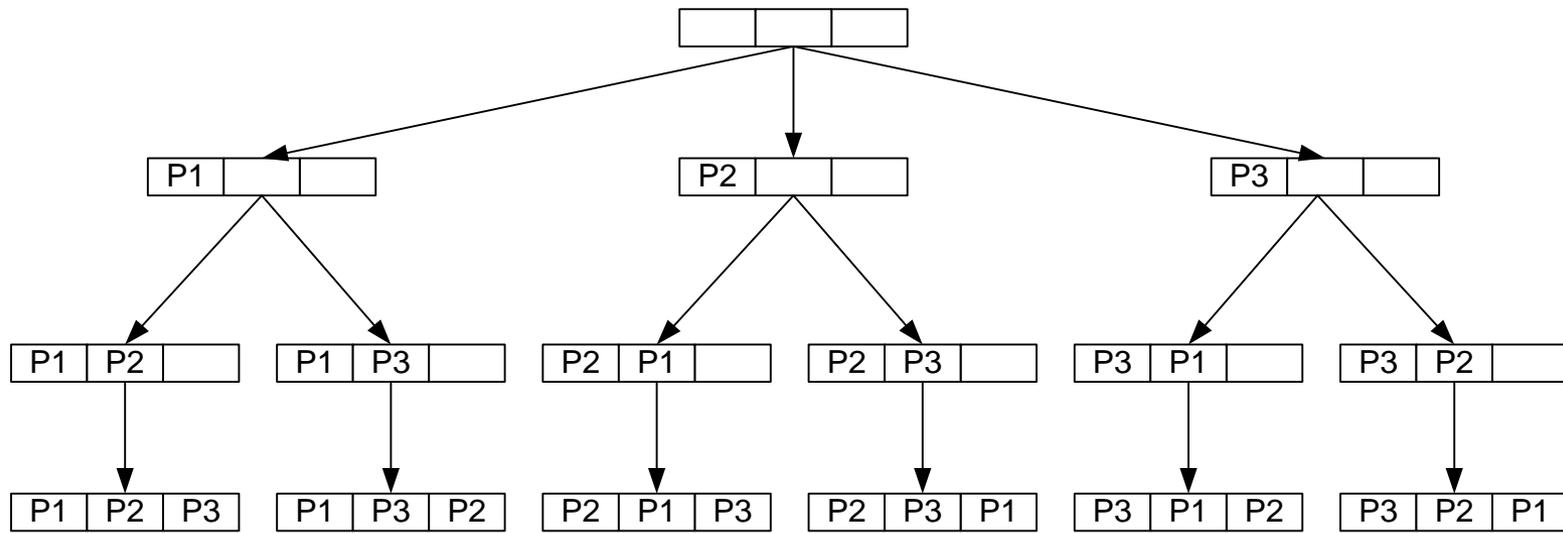


Scheduling

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 Bereitstellungszeiten r_i
- Prinzipiell gilt jedoch: **Bei komplexen Systemen ist Planen durch Suchen nicht möglich.**

Bewertung

- Vorteile:
 - Sehr geringer Overhead zur Laufzeit
 - Sehr gut zu analysieren wegen Determinismus
 - Einfache Lösung um Race Conditions zu vermeiden
- Nachteil:
 - Alle Schedulingparameter (Startzeiten, Ausführungszeiten, Fristen) müssen bekannt sein oder abgeschätzt werden
 - Falls Schätzung erfolgt: häufig ineffizient wegen schlechter Ressourcenausnutzung
 - Komplexität der Berechnung
- Anwendung in der Praxis:
 - Einsatz eher selten und dann bei hochkritischen Systemen als zeitgesteuertes Scheduling
 - Zeitvergabe durch den Entwickler
 - Häufig in Kombination mit anderen Verfahren (z.B. prioritätenbasiertes Scheduling): Eine Teilmenge der Prozesse bekommt einen gemeinsamen Zeitabschnitt zugewiesen, innerhalb des Abschnittes erfolgt das Scheduling nach anderen Kriterien

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.

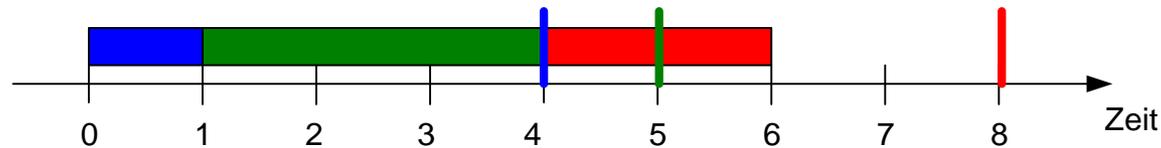
Beispiel

- 3 Prozesse:

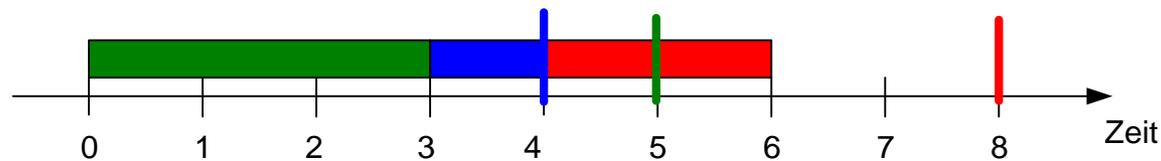
$P_1: r_1=0; e_1=2; d_1=8;$

$P_2: r_2=0; e_2=3; d_2=5;$

$P_3: r_3=0; e_3=1; d_3=4;$



Earliest Deadline First



Least Slack Time

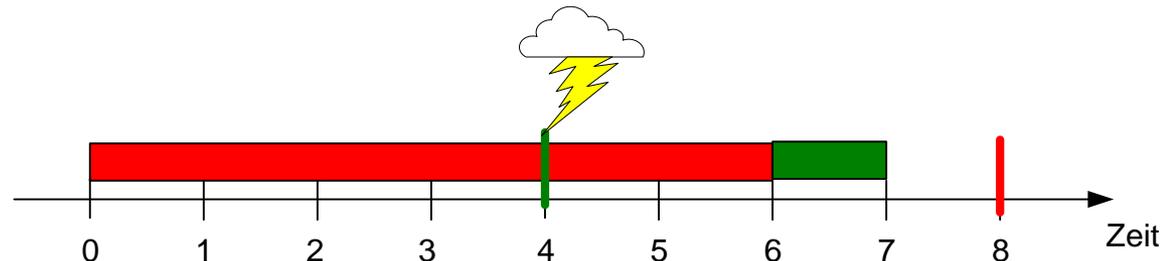
Versagen von LST

- LST kann selbst bei gleichen Bereitzeiten im nicht-präemptiven Fall versagen.

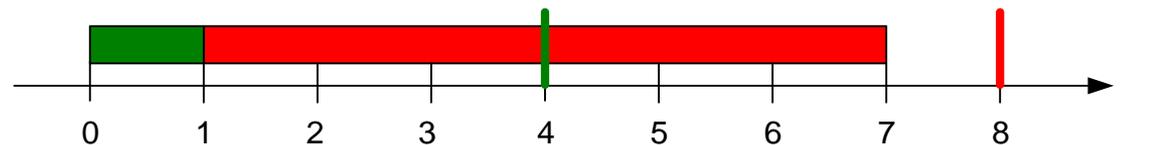
- 2 Prozesse:

$P_1: r_1=0; e_1=6; d_1=8;$

$P_2: r_2=0; e_2=1; d_2=4;$



LST: P2 verpasst Deadline



EDF liefert optimalen Plan

- 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
 - Sei 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}
 - $P(\text{Plan}_x, t)$ sei der Prozess, der von Plan_x zum Zeitpunkt t ausgeführt wird.
 - $\text{Plan}_x(t)$ ist der bis zum Zeitpunkt t in Plan_{EDF} überführte Plan mit $\text{Plan}_x(0) = \text{Plan}_x$.

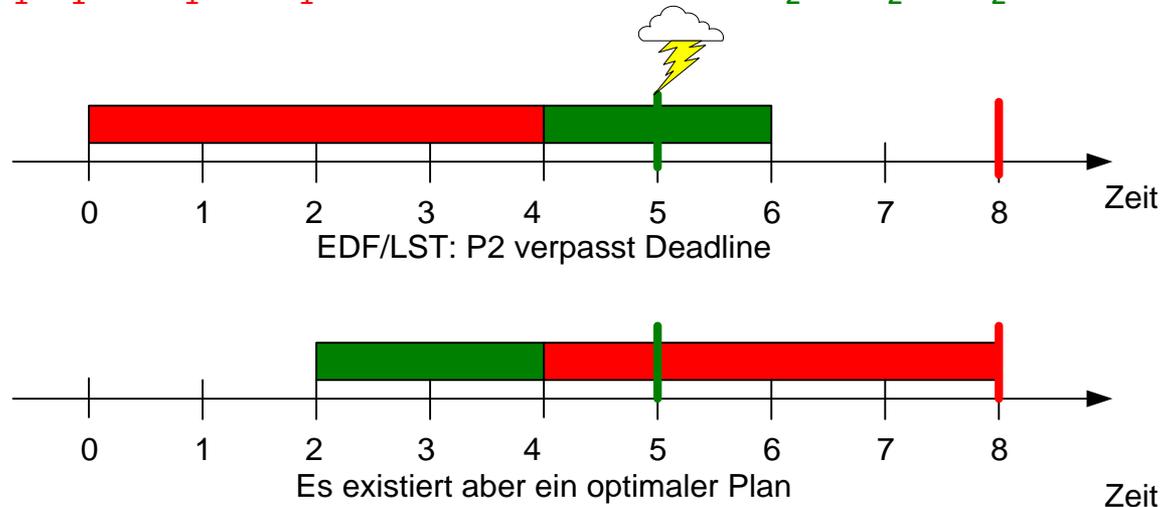
Fortsetzung des Beweises

- Wir betrachten ein beliebig kleines Zeitintervall Δ_t .
- Zum Zeitpunkt t gilt:
 $i = P(\text{Plan}_{\text{EDF}}, t)$
 $j = P(\text{Plan}_x, t)$
- 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 wird, kann der Prozess P_i im Plan_x noch nicht beendet sein.
→ EXISTS $t' > t + \Delta_t$: ($i = P(\text{Plan}_x, t') = P(\text{Plan}_x, t' + \Delta_t) \wedge t' + \Delta_t \leq d_i \leq d_j$)
→ 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 → Übergang von $\text{Plan}_x(t)$ zu $\text{Plan}_x(t + \Delta_t)$

Versagen von EDF bei unterschiedlichen Bereitzeiten

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

$P_2: r_2=2; e_2=2; d_2=5$



- Anmerkung:** Jedes prioritätsgesteuerte, **nicht präemptive** Verfahren versagt bei diesem Beispiel, da ein solches Verfahren nie eine Zuweisung des Prozessors an einen lafbereiten 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)



Scheduling

Scheduling-Verfahren für Mehrprozessor-Systeme

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.

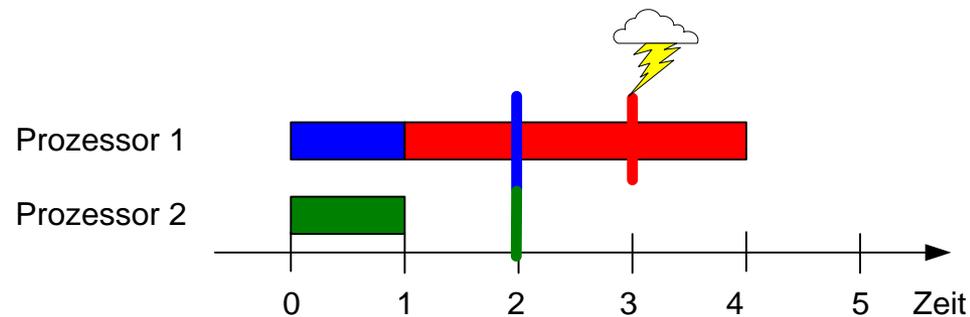
Beispiel: Versagen von EDF

- 2 Prozessoren, 3 Prozesse:

$P_1: r_1=0; e_1=3; d_1=3;$

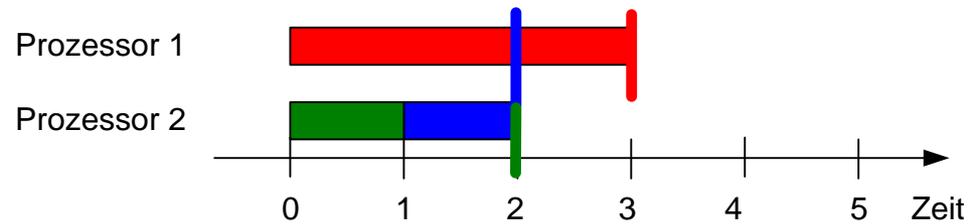
$P_2: r_2=0; e_2=1; d_2=2;$

$P_3: r_3=0; e_3=1; d_3=2;$

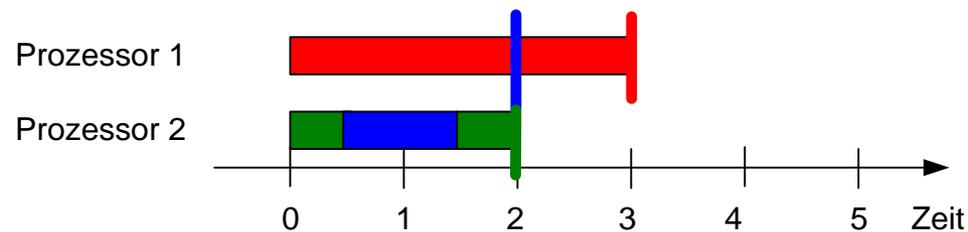


EDF-Verfahren: Deadline d_1 wird verpasst

Beispiel: Optimaler Plan und LST-Verfahren



Optimaler Plan



LST-Verfahren mit $\Delta t = 0.5$

Beispiel: Versagen von LST

- 2 Prozessoren, 5 Prozesse, $\Delta_t=0,5$:

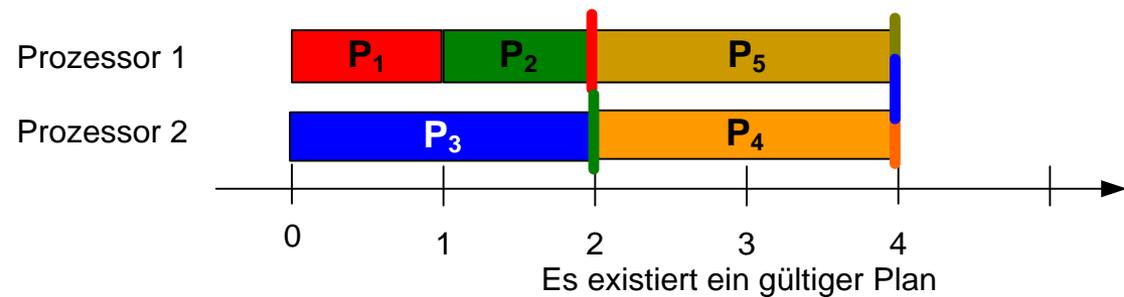
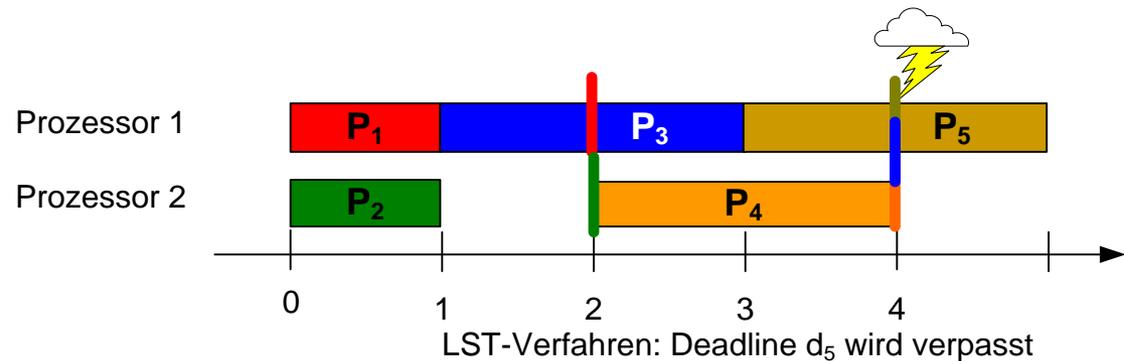
P_1 : $r_1=0$; $e_1=1$; $d_1=2$;

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 Bereitstellungszeiten 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) → 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:

$P_1: r_1=0; e_1=1; d_1=1;$

$P_2: r_2=0; e_2=2; d_2=4;$

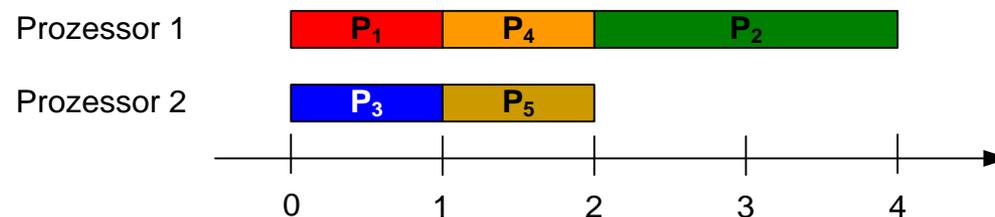
$P_3: r_3=0; e_3=1; d_3=2;$

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

→ Es gibt je nach Strategie zwei Fälle zu betrachten: P_2 oder P_3 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.
- Anstelle der zum Zeitpunkt 2 noch notwendigen 5 Ausführungseinheiten sind nur 4 vorhanden.
- Aber es gibt einen gültigen Ausführungsplan:

