Anwendungsgewahre Ressourcenverwaltung in Echtzeitregelungssystemen

Daniel Schiener
FAU Erlangen-Nürnberg
daniel.schiener@fau.de

ZUSAMMENFASSUNG

Überlastsituationen in Echtzeitsystemen sind typischerweise das Resultat von Rekonfigurationsmaßnahmen zur Laufzeit. Für die Rückkehr in den Normalzustand des Systems ist eine spezielle Behandlungsroutine verantwortlich. Deren Ausführung beansprucht zusätzliche Rechenzeit zu den regulär eingeplanten Aufgaben. Die Seltenheit der Rekonfigurationssituationen erlaubt allerdings bei der strikten Planung eines Echtzeitsystems keine Reservierung eines entsprechenden zeitlichen Anteils. Zugunsten der Behandlungsroutine wird somit regulären Aufgaben Rechenzeit entzogen. Terminverletzungen, gleichzusetzen mit verspäteten bzw. veralteten Berechnungsergebnissen, oder gar vollständige Berechnungsausfälle sind unter Umständen die Folge. Abhängig von der Art des Echtzeitsystems können derartige Effekte unter Umständen bis zu einem gewissen Grad toleriert werden. So erlaubt die inhärente Robustheit eines Echtzeitregelungssystems beispielsweise von Haus aus eine graduelle Abnahme der Regelgüte. Dieser Vorteil in Form von Flexibilität ermöglicht letztlich den Einsatz neuartiger Konzepte zur Toleranz der Überlastsituationen.

Schlüsselwörter

Dynamische Reallokation, Echtzeitregelungssysteme, Moduswechsel, Überlast, Garantie der Regelgüte

1. EINLEITUNG

Bei Echtzeitregelungssystemen bedarf es einer grundsätzlichen Unterscheidung von konventionellen Softwaresystemen. Neben funktionalen Anforderungen wie Speicherplatzbegrenzung oder limitierter Rechenleistung stehen bei Echtzeitregelungssystemen vor allem nicht-funktionale Anforderungen wie Vorhersagbarkeit und Rechtzeitigkeit im Vordergrund. Diese beiden Anforderungen grenzen den Einsatz bereits bekannter Möglichkeiten zur dynamischen Anpassung des Systemverhaltens sehr stark ein. Methoden wie das dynamische Austauschen von Funktionen auf Basis von Indirektionen beim Funktionsaufruf [2] sind bedingt durch deren nicht-deterministisches Verhalten kaum anwendbar. Ebenso Methoden wie das Einspielen von Aktualisierungen für Module des Linuxkernels durch KSplice [3], deren Einsatz eine Stillstandzeit des gesamten Systems erfordern, sind im Sinne der Rechtzeitigkeit nur schwer einplanbar. Die genannten Eigenschaften sind ausschlaggebend dafür, dass bei Echtzeitsystemen der Fokus in erster Linie nicht auf der dynamischen Modifikation des Betriebssystems, sondern eher auf Modifikationen in der Anwendungsebene liegt.

Regelungsanwendungen sind im Allgemeinen die weit verbreitetste Art von Anwendungen im Bereich von Echtzeitsystemen. Zusammen mit anderen Anwendungen (z.B. Kommunikationsstapel) werden sie als Aufgaben (engl. task) eingeplant und ausgeführt. Regelungsaufgaben folgen üblicherweise einem einheitlichen, periodischen Schema: Messen, rechnen, regeln. Diese Periodizität hat maßgeblich Einfluss auf die Regel- (engl. quality of control, QoC) bzw. Dienstgüte (engl. quality of service, QoS) des Systems.

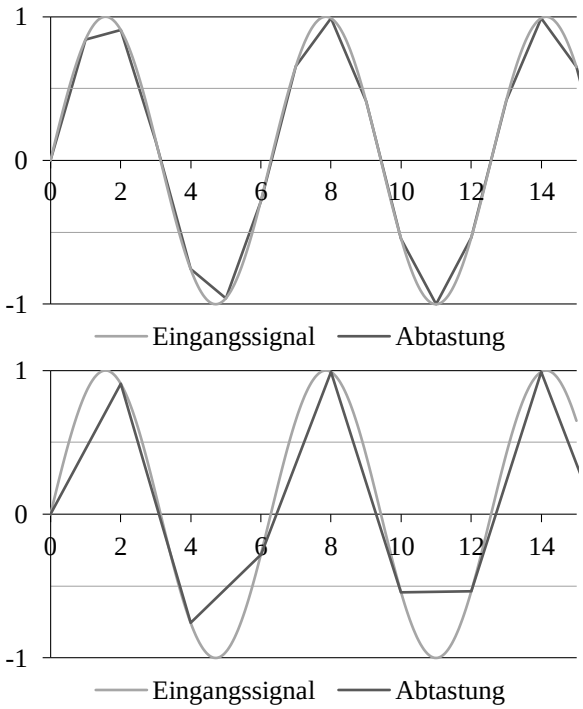

Abbildung 1: Abtastung eines Sinussignal durch eine Aufgabe mit der Periode 1 (oben) und mit der Periode 2 (unten)

Die Abbildung 1 zeigt die beispielhafte Abtastung eines Sinussignals durch eine Aufgabe mit der Periode 1 (oben) sowie mit der Periode 2 (unten). Ziel der Regelanwendung ist es die Rekonstruktion (dunkel) des Eingangssignals basierend auf den Abtastwerten vorzunehmen. Es ist ersichtlich, dass die Qualität des rekonstruierten Signals stark von der Abtastungsrate (engl. sampling rate) abhängt. Somit gilt

die Aussage: Je kürzer die Abtastzeitpunkte (engl. sampling time) aufeinander folgen, sprich je häufiger die Aufgabe ausgeführt wird, desto höher ist die Regelgüte.

Weiterhin zeichnen sich Regelungsanwendungen durch eine inhärente Robustheit aus, sprich sie verfügen über eine Art Toleranzbereich. So ist eine Regelungsanwendung beispielsweise in der Lage eine gewisse Anzahl sowohl von verpassten Abtast- oder Regelzeitpunkten als auch von vollständigen Berechnungsausfällen zu kompensieren. Problematisch hingegen sind üblicherweise Schwankungen (engl. jitter) der Abtastzeitpunkte. Sofern die Regelungsanwendung auf einem kontinuierlichen Prozess basiert, können Schwankungen die Signalrekonstruktion erheblich beeinflussen und zu einem Fehlverhalten oder gar zu einem Komplettausfall des Systems führen. Im Gegensatz zum diskreten Fall, bei der die Signalwiederherstellung zu jedem Abtastwert auch den zugehörigen diskreten Abtastzeitpunkt erhält, erwartet eine kontinuierliche Signalwiederherstellung die Abtastwerte in unveränderlichen, relativen Zeitintervallen.

Um den Anforderungen eines Echtzeitregelungssystems gerecht zu werden, müssen alle zeitlichen Parameter (CPU-Taktung, Sensorfrequenzen, Ausführungszeiten, etc.) bereits zur Entwurfszeit bekannt sein. Dies ermöglicht die Erstellung eines exakten Ablaufplans sowie verlässliche Aussagen über das zeitliche Verhalten des Systems bereits vor dessen Inbetriebnahme. Der Ablaufplan fungiert quasi als Fahrplan für das Echtzeitsystem. Unter Annahme der Ausführungszeit im schlechtesten Fall (engl. worst-case execution time, WCET) wird die zur Verfügung stehende Rechenzeit auf alle Aufgaben aufgeteilt. Aufgaben untergliedern sich wiederum in Arbeitsaufträge (engl. job) und definieren sich über ihren Auslösezeitpunkt (engl. release time), ihren Einlast- (engl. start time) und Endzeitpunkt (engl. finish time), ihre Periode (engl. period), ihre Ausführungszeit (engl. execution time) sowie über ihren Termin (engl. deadline). Letzterem kommt eine besondere Bedeutung zu. Handelt es sich um harte Termine, dürfen diese im Gegensatz zu weichen Terminen unter keinen Umständen verletzt bzw. überschritten werden. Anderenfalls ist eine Ausnahmebehandlung vonnöten mit der die Terminverletzung eventuell toleriert werden kann. Bei Regelungsanwendungen kann mit einem Termin beispielsweise die Freigabe der berechneten Stellwerte für die Aktuatoren assoziiert werden. Um die Wichtigkeit einer Aufgabe hervorzuheben, lassen sich den Aufgaben in Abhängigkeit von dem verwendeten Ablaufplanungsverfahren auch Prioritäten zuordnen. Die Verwendung von Prioritäten impliziert üblicherweise auch die Eigenschaft der Verdrängung. Mit dem raten-monotonen Ablaufplanungsverfahren (engl. rate monotonic, RM) erhalten beispielsweise Aufgaben ihre Prioritäten in Abhängigkeit ihrer Perioden: Je kürzer die Periode desto höher die Priorität. Bei dem RM-Verfahren handelt es sich um ein etabliertes Verfahren zur Ablaufplanung in der Domäne der Echtzeitsysteme.

Wie bereits angedeutet existieren trotz exakter Planung situationsbedingte Ausnahmefälle, die zur Verletzung der Termine und folglich zu einer Abnahme der Regelgüte des Systems führen. Als Ausnahmefälle gelten beispielsweise der Wechsel des Betriebsmodus oder in seltenen Fällen die Wiederherstellung der Anwendung nach dem Ausfall eines Prozessors. Die mit solchen Situationen einhergehende Behandlungsroutine muss zusätzlich zu den regulär eingeplanten Aufgaben ausgeführt werden. Die Tatsache, dass im Ablaufplan für derartige Situationen aufgrund ihrer Seltenheit üblicherweise kein zeitliches Kontingent reserviert wird, überführt das System in einem Zustand der Überlast. An diesem Punkt angelangt gilt es diesen Zustand zu tolerieren, um die Regelgüte des Systems gewährleisten und einen Totalausfall vermeiden zu können.

Im Folgenden wird nun zunächst ein naiver Ansatz zur dynamischen Rekonfiguration in Echtzeitsystemen vorgestellt. Im Anschluss daran folgen zwei Konzepte aus dem Bereich der Forschung, deren Ziel es ist eine Gewährleistung der Regelgüte in Überlastsituationen zu ermöglichen. Zu diesem Zweck verfolgt der erste Ansatz zur Ausführung der Regelaufgabe die Verwendung eines bandweite-bewahrenden Zustellers (engl. constant bandwith server, CBS) kombiniert mit einem speziellen Berechnungsmodell. Demgegenüber betrachtet der zweite Ansatz eine Anpassung der Periode der Regelaufgabe basierend auf der stetigen Überwachung der Regelgüte des Systems.

2. ANSATZ ZUR DYNAMISCHEN REKONFIGURATION: „ONE-SHOT“

Eine Möglichkeit um eine Rekonfiguration eines Echtzeitregelungssystems zur Laufzeit durchzuführen ist der Wechsel des Betriebsmodus. Grund hierfür sind üblicherweise sich verändernde externe Einflüsse auf das System. Eine gängige Methode um den Wechsel durchzuführen ist der Tausch des Ablaufplans. So wird beispielsweise bei einsetzendem Schneefall im Abstandsregelungssystem eines Automobils von Radar- auf Lasersensorik gewechselt. Als Umschaltzeitpunkt bietet sich der Beginn einer neuen Hyperperiode an. Eine Hyperperiode bezeichnet die Zeitspanne, die vonnöten ist um jede Aufgabe mindestens ein Mal abzuarbeiten. Die Länge einer Hyperperiode entspricht deshalb dem kleinsten gemeinsamen Vielfachen aller Aufgabenperioden.

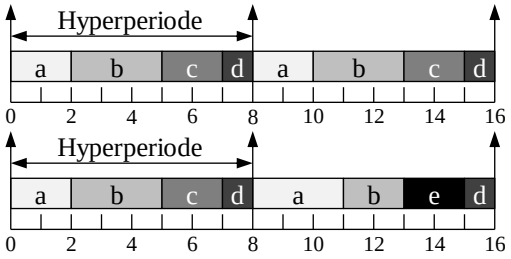

Abbildung 2: Hyperperiode (oben) sowie Moduswechsel zum Beginn einer neuen Hyperperiode (unten)

Abbildung 2 veranschaulicht einen Tausch von Ablaufplänen zum Beginn einer neuen Hyperperiode. Der obere Ablaufplan aus Abbildung 2 zeigt die Wiederholung aller Aufgaben und folglich den Beginn einer neuen Hyperperiode zum Zeitpunkt 8. Im unteren Ablaufplan ist ein Wechsel des Betriebsmodus erkennbar. Einerseits wird Aufgabe c aus dem alten Ablaufplan durch eine neue Aufgabe e ersetzt, andererseits verändert sich durch den Moduswechsel auch der Auslösezeitpunkt sowie die Ausführungszeit der Regelaufgabe b.

Der Umschaltvorgang selbst unterliegt einer wichtigen zeitlichen Anforderung, die direkt mit der Regelgüte des Systems korreliert. Um eine Abnahme der Regelgüte ausschließen zu können, müssen die Start- und Endzeitpunkte bzw. Termine der Regelaufgabe eingehalten werden. Im Fall des Umschaltvorgangs bedeutet das ein Zeitintervall ausgehend vom Endzeitpunkt der Regelaufgabe des alten Ablaufplans bis zum Startzeitpunkt der Regelaufgabe des neuen Ablaufplans. Im Beispiel aus Abbildung 2 entspricht das dem Zeitintervall von 5 bis 11 im unteren Ablaufplan. Verletzt der Umschaltvorgang diese Bedingung, so verpasst die neue Regelaufgabe mindestens einen Startzeitpunkt. Im Erfolgsfall spricht man bei umschaltbasierten Moduswechseln dieser Art von sogenannten „one-shot"-Lösungen. Unvorteilhaft ist auch die Dauer einer Hyperperiode. Enthält ein Ablaufplan viele Aufgaben mit unterschiedlicher Periodizität, so ist eine lange Hyperperiode unvermeidbar. Infolgedessen können der Anforderungszeitpunkt eines Moduswechsels und der effektive Umschaltzeitpunkt zeitlich weit auseinander liegen, worunter letztlich die Reaktionsfähigkeit des Systems leidet. Am Beispiel von sicherheitskritischen Systemen, wo eine minimale Reaktionslatenz gefordert wird, ist dieses Verhalten unvereinbar. Der Umschaltzeitpunkt zum Beginn einer neuen Hyperperiode bringt allerdings einen wichtigen Vorteil mit sich. Da die Allokation von Betriebsmitteln (z.B. Sperrvariablen) von Aufgaben über Hyperperioden hinweg nicht zulässig ist, können Ressourcenkonflikte, transitive Blockierungen oder Verklemmungen mit dieser Methode ausgeschlossen werden. Zum Beginn einer neuen Hyperperiode befindet sich das System quasi in einem Grundzustand (engl. quiescence state).

3. TOLERANZ DER ÜBERLAST

3.1 Anpassung der Rechenzeit

In [1] beschreiben die Autoren Fontanelli, Greco und Palopoli eine zur Laufzeit anwendbare Methode um die Regelgüte eines Echtzeitregelungssystems in Überlastsituationen gewährleisten zu können. In ihrem Szenario betrachten sie ausschließlich den kontinuierlichen Fall. Als die wichtigsten Ziele lassen sich demnach die Reduzierung von Schwankungen in der Abtastung und Regelung, sprich die Vermeidung von Terminverletzungen sowie vollständige Berechnungsausfälle der Regelaufgabe identifizieren. Der Eckpfeiler für die Realisierung ihres Konzepts ist der Einsatz eines bandweitebewahrenden Zustellers in Kombination mit einem speziellen Berechnungsmodell.

3.1.1 Zusteller

Mit einem Zusteller wird üblicherweise die Abarbeitung nichtperiodischer Aufgaben in Verbindung gebracht. Zu diesem Zweck wird ein Zusteller im Allgemeinen als eine spezielle periodische Aufgabe eingeplant. In deren Kontext werden anschließend zur Verfügung stehende nicht-periodische Aufgaben ausgeführt. Neben den bekannten Parametern einer regulären periodischen Aufgabe zeichnet sich ein Zusteller durch zwei weitere Eigenschaften aus: Das Ausführungsbudget Q sowie die Auffüllperiode R. Die Auffüllperiode ist in etwa gleichzusetzen mit der Periode und das Ausführungsbudget mit der Ausführungszeit einer regulären Aufgabe. In Relation gesetzt ergibt

$$B = \frac{Q}{R}$$

die Bandweite B (engl. bandwidth) eines Zustellers. Die Bandweite definiert somit den zugewiesenen Anteil an Rechenzeit eines Zustellers. Zum Beginn einer neuen Auffüllperiode wird das Ausführungsbudget des Zustellers erneuert. Zu diesem Zeitpunkt hat der Zusteller das Ausführungsbudget entweder für die Abarbeitung verfügbarer Aufgaben verbraucht oder verworfen. Letzterer Fall tritt ein, wenn bei Ausführungsbeginn des Zustellers keine ausführbereiten Aufgaben zur Verfügung stehen, sprich die Auslösung der Aufgabe später als die des Zustellers erfolgt. Später ausgelöste Aufgaben können durch den Zusteller erst in der nächsten Auffüllperiode abgearbeitet werden. Im Unterschied dazu garantiert ein bandweite-bewahrender Zusteller den ihm zugeteilten Aufgaben in jeder Auffüllperiode das Ausführungsbudget unabhängig von ihrem Auslösezeitpunkt. Ein bandweite-bewahrender Zusteller unterliegt vor allem einer Bedingung:

$$\sum_i = B_i \leq U, mit\ U = 1$$

Daraus folgt, dass die Summe der zugewiesenen Bandweiten B_i aller eingeplanten Zusteller bzw. Aufgaben eine Schranke U nicht überschreiten darf. U kennzeichnet in diesem Fall die verfügbare Prozessorauslastung von maximal 100%.

Mit den zusätzlichen Parametern Q und R verfügt man potentiell über zwei Möglichkeiten, die eine feste oder variable Anpassung der Regelgüte des Systems zulassen. Unter einer festen Anpassung der Regelgüte verstehen die Autoren die einmalige Ermittlung der Bandweite für die Ausführung der Regelaufgabe zur Entwurfszeit. Jeder Aufgabe, insbesondere jenen mit harten Terminen, kann anhand dieser Variante eine Garantie für die Einhaltung ihres Termins gegeben werden, sofern für deren Prozessorauslastung im schlechtesten Fall U_i gilt:

$$B_i \geq U_i, mit\ U_i = \sup_j \frac{c_{ji}}{T_i}$$

Die maximale Prozessorauslastung einer Aufgabe entspricht demnach der höchsten Prozessorauslastung durch den zeitintensivsten Arbeitsauftrag der Aufgabe. Demgegenüber existiert auch die Möglichkeit einer variablen Anpassung der Regelgüte. Hierzu wird der Wert der Bandweite einer Aufgabe zum einen zur Laufzeit und zum anderen für jeden Arbeitsauftrag separat ermittelt.

3.1.2 Konzept

In dem Szenario aus [1] wird eine implizit dreigeteilte, periodische Regelaufgabe mit weichem Termin betrachtet. Die Regelaufgabe unterteilt sich in die Arbeitsaufträge „Abtastung", „Berechnung" und „Regelung". Eine implizite Dreiteilung bedeutet an dieser Stelle, dass in [1] faktisch nur die Berechnung eingeplant und ausgeführt wird. Abtastung und Regelung sind quasi zeitlos. Im Kontext eines bandweitebewahrenden Zustellers wird die Regelaufgabe sie zusammen mit gleich- bzw. niederwertigen Aufgaben auf einem Ein-Prozessor-System ausgeführt. Für die Periode T der Regelaufgabe bzw. des Zustellers soll aus Gründen der Performanz gelten:

$$T = nR$$

Die Auffüllperiode ist folglich ein ganzzahliger Teiler der Aufgabenperiode T. In Abhängigkeit von der Systemauslastung wird die Bandweite des Zustellers variabel durch eine

Veränderung des Ausführungsbudgets angepasst. Infolgedessen steht für die Regelaufgabe in Überlastsituationen unter Umständen nicht ausreichend Ausführungsbudget zur Verfügung, weshalb eine höhere Anzahl an Auffüllperioden für deren Abarbeitung vonnöten ist. Das führt letztlich zu Terminverletzungen bzw. Verzögerungen bei der Regelung, sprich Schwankungen auf der Ausgabeseite. Das nachfolgende Berechnungsmodell erlaubt allerdings in Kombination mit dem bandweite-bewahrenden Zusteller eine frühzeitige Abschätzung des schlechtesten Falls für derartige Situationen. Für das Berechnungsmodell gilt im Allgemeinen:

- Regel 1: Ein Arbeitsauftrag endet vor seinem Termin. Die Freigabe der Stellwerte erfolgt am Ende der Periode.
- Regel 2: Ein Arbeitsauftrag endet nach seinem Termin. Die Freigabe der Stellwerte erfolgt am Ende der aktuellen Auffüllperiode.
- Regel 3: Ein Arbeitsauftrag verletzt seinen Termin und überschreitet einen Grenzwert (eine Periode). Der Arbeitsauftrag wird zugunsten eines neuen abgebrochen.
- Regel 4: Die Abtastung der Eingabewerte der aktuellen sowie die Regelung der Stellwerte der vorhergehenden Aufgabe erfolgen zum gleichen Zeitpunkt.

Das Ziel von Regel 1 ist die Vermeidung von Schwankungen auf der Ausgabeseite, hervorgerufen durch das frühzeitige Ende einer Regelaufgabe. In Abbildung 3 endet die Regelaufgabe 2 bereits zum Zeitpunkt 15 und könnte die Regelung durchführen. Regel 1 bewirkt allerdings die Verzögerung der Regelung bis zu Zeitpunkt 18, dem Periodenende. Absicht von Regel 2 ist die Vermeidung von variablen und unbekannten Zeitpunkten der Regelung im Fall verletzter Termine. Für Regelaufgabe 1 in Abbildung 3 ist eine Terminverletzung zum Zeitpunkt 9 erkennbar. Angesichts der Regel 2 erfolgt die Regelung allerdings zum Zeitpunkt 12 statt zum Zeitpunkt 10. In Kombination mit Regel 1 und unter der Annahme, dass $T = nR$ gilt, bewirkt die Anwendung von Regel 2 die Ausrichtung der Regelung an den Auffüllperioden. Resultat von Regel 3 ist die Vermeidung sehr hoher und nicht tolerierbarer Verzögerungen einer Regelaufgabe. Mit Regel 4 erfolgt im Fall von Terminverletzungen eine Aktualisierung der Eingabewerte. Im Normalfall bzw. nun auch unter Betrachtung von Regel 1 und Regel 4 erfolgt die Abtastung der Eingabewerte immer zum Periodenende der Regelaufgabe. Für Regelaufgabe 2 in Abbildung 3 stünden folglich veraltete Eingabewerte von der Abtastung zum Zeitpunkt 9 zur Verfügung. Durch Regel 4 erhält die Regelaufgabe 2 aktualisierte Eingabewerte zum Zeitpunkt 12.

3.1.3 Fazit

Das Konzept des bandweite-bewahrenden Zustellers kombiniert mit dem vorgestellten Berechnungsmodell erlaubt eine weitaus flexiblere Reaktion auf Überlastsituationen als herkömmliche Ansätze. Mithilfe des Berechnungsmodells können Terminverletzungen oder Berechnungsausfälle und somit der schlimmste Fall bereits zur Entwurfszeit abgeschätzt und toleriert werden. Weiterhin eliminiert das Berechnungsmodell Schwankungen unbekannter Dauer sowohl auf Ein-

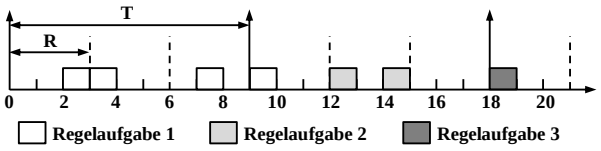

Abbildung 3: Beispielhafter Ablaufplan für das Berechnungsmodell

als auch auf Ausgabeseite und stellt der Berechnung immer aktuelle Eingabedaten zur Verfügung. Die implizite Dreiteilung der Regelaufgabe und die Folge, dass Abtastung und Regelung de facto zeitlos sind, gibt allerdings Grund zur Skepsis. Ebenso wird exklusiv die Behandlung unabhängiger Aufgaben betrachtet. Angesichts des weit verbreiteten Erzeuger-Verbraucher-Szenarios (engl. producer-consumer-scenario) ist das eine starke Annahme. Darüber hinaus ist das Verfahren für die Ermittlung der Systemauslastung insofern fragwürdig, als dass hierzu keine weiteren Aussagen getroffen werden. Dieses Referenzmaß ist jedoch ausschlaggebend für die dynamische Anpassung der Bandweite des Zustellers und somit die resultierende Anpassung der Regelgüte des Systems.

3.2 Anpassung der Perioden

Ein weiteres Konzept zur Toleranz von Überlastsituationen in Echtzeiregelungssystemen ist in [4] beschrieben. Die Autoren Shin und Meissner betrachten hierzu den diskreten Fall. Für ihre Untersuchungen stehen Rekonfigurationssituationen wie ein Wechsel des Betriebsmodus oder insbesondere der Ausfall eines Prozessors im Vordergrund. In diesem Sinne liegt in [4] der Fokus auch in erster Linie auf einer wenig zeitintensiven Wiederherstellung der Regelungsanwendung im Fehlerfall. Hierzu betrachten die Autoren beispielsweise die Verwendung von sogenannten Schattenaufgaben (engl. shadow task). Im Allgemeinen basiert ihr Konzept allerdings auf der Anpassung der Aufgabenperioden im Fehler- bzw. Reallokationsfall. Hierzu präsentieren sie ein spezielles Belohnungssystems mit dessen Hilfe die Periodenanpassungen unter Einhaltung der minimalen Regelgüte vorgenommen werden.

3.2.1 Konzept

Für das Szenario in [4] wird eine periodische, hochpriore Regelaufgabe betrachtet. Zusammen mit einer Menge von weiteren Aufgaben unterschiedlicher Prioritäten und Perioden wird die Regelaufgabe auf einem Multiprozessorsystem ausgeführt. Im Kontrast zum vorhergehenden Konzept, behandelt man in [4] sowohl abhängige als auch unabhängige Aufgaben. Kern des Konzepts ist ein spezielles Belohnungssystem. Unter diesem Aspekt erhält jede Aufgabe zur Entwurfszeit neben ihren bekannten Merkmalen eine Belohnung (engl. reward). Darüber hinaus existiert für jede Aufgabe eine Menge von Perioden unter denen sie potentiell ausgeführt werden kann. Die Belohnung einer Aufgabe ist in etwa gleichzusetzen mit dem Nutzen, den die Ausführung der Aufgabe unter einer bestimmten Periode für das Gesamtsystem bedeutet. Im Allgemeinen richtet sich das Belohnungssystem nach folgendem Schema:

- je wichtiger eine Aufgabe, desto höher ihr Nutzen

- je höher ihr Nutzen, desto höher ihre Belohnung
- je höher ihre Belohnung, desto höher ihr Anteil an Rechenzeit

Es existiert somit eine direkte Abhängigkeit zwischen der zugewiesenen Rechenzeit einer Aufgabe und ihrem Nutzen bzw. ihrer Belohnung. Das Ziel in einer Rekonfigurationssituation ist es, den Gesamtnutzen des Systems trotz Überlast zu optimieren. Zu diesem Zweck verfügt das Konzept über zwei Reallokationsalgorithmen, die in diesen Situationen zum Einsatz kommen.

Ein weiterer Bestandteil des Konzepts ist ein sogenannter Performanzindex (engl. performance index, PI). Bei dem PI handelt es sich prinzipiell um ein spezielles Referenzmaß. Mit dessen Hilfe lässt sich unter anderem die resultierende Regelgüte einer Aufgabe bestimmen, sofern diese unter einer ihrer Perioden ausgeführt wird. Der PI ermöglicht folglich eine Abschätzung der Kosten für das System, um eine Aufgabe oder Schattenaufgabe mit einer bestimmten Periode auszuführen. Zu diesem Zweck erfolgt die Ermittlung des jeweiligen PI simulationsbasiert zur Entwurfszeit. Unter einer Schattenaufgabe verstehen die Autoren eine Art Sicherungskopie der Basisaufgabe. Insbesondere bei einem Ausfall der Basisaufgabe, beispielsweise durch den Verlust des entsprechenden Prozessors, wird im weiteren Verlauf auf die Ergebnisse der Schattenaufgabe zurückgegriffen. Da die Berechnungsergebnisse einer Schattenaufgabe somit im Normalzustand irrelevant sind, erfolgt deren Ausführung immer unter der maximalen Periode ihrer Basisaufgabe um Ressourcen zu sparen. Eine Begrenzung der Anzahl von Schattenaufgaben für eine Basisaufgabe existiert im Allgemeinen nicht. Das Konzept der Sicherungskopie bedingt jedoch, dass sich eine Schattenaufgabe nie den Prozessor mit ihrer Basisaufgabe oder mit typgleichen Schattenaufgaben teilen darf.

Fällt ein Prozessor aus, so werden zunächst alle vom Ausfall betroffenen Aufgaben für die Reallokation markiert. Sofern Schattenaufgaben existieren, werden deren Ergebnisse für den Zeitraum der Reallokation verwendet. Nachdem die Überprüfung und Markierung aller Aufgaben abgeschlossen ist, beginnt einer der Reallokationsalgorithmen mit der Verteilung der Aufgaben auf die restlichen Prozessoren. Die Wahl des Algorithmus richtet sich schlussendlich nach dem Abhängigkeitsszenario des Anwendungsfalls. Für unabhängige Aufgaben ist es AIT (engl. algorithm for independent tasks) und für abhängige Aufgaben ist es ART (engl. algorithm for related tasks). Der Verteilungs- oder Reallokationsprozess ist ein Optimierungsproblem und kann gut mit dem bekannten Rucksackproblem (engl. knapsack problem) assoziiert werden. Der Rucksack entspricht beispielsweise der gesamten, verfügbaren Rechenzeit aller fehlerfreien Prozessoren. Anstelle von Objekten mit Kosten und Nutzwert gilt es nun den Rucksack durch Aufgaben mit Rechenzeit und Nutzen optimal zu befüllen. Die Zielfunktion entspricht demnach:

Maximiere den Gesamtnutzen

Als Nebenbedingungen für das Optimierungsproblem gelten:

- die maximale Prozessorauslastung beträgt 100%
- Basisaufgaben und Schattenaufgaben dürfen nicht auf demselben Prozessor allokiert werden

Als zusätzliche Nebenbedingung für abhängige Aufgaben gilt:

- die Zuteilung der Belohnung bzw. der Rechenzeit einer Aufgabe erfolgt unter Beachtung ihrer Abhängigkeiten (wird beispielsweise die Belohnung einer Vorgängeraufgabe erhöht, so erhöhen sich auch die Belohnungen der nachfolgenden Aufgaben)

Enden AIT oder ART steht eine vorläufige Allokation zur Verfügung. Die vorläufige Allokation dient wiederum als Ausgangspunkt für die Verteilung der jeweiligen Aufgaben mit ihren ermittelten Belohnungen und Rechenzeiten auf die einzelnen Prozessoren. Im Erfolgsfall entspricht die vorläufige gleich der endgültigen Allokation und die jeweiligen Ablaufpläne werden an die verfügbaren Prozessoren verteilt. Im Fehlerfall erfolgt eine iterative Wiederholung des Optimierungsprozesses solange bis eine zulässige und endgültige Allokation ermittelt wird.

3.2.2 Evaluierung

Zur Evaluierung des AIT wie auch des ART wird von den Autoren ein spezielles Testszenario betrachtet, in dem eine zufällig erzeugte Aufgabenmenge von 100 Aufgaben im Fall des Verlusts zweier Prozessoren reallokiert werden muss. Die nachfolgenden Abbildungen 4 und 5 zeigen sowohl für AIT als auch für ART die Ergebnisse.

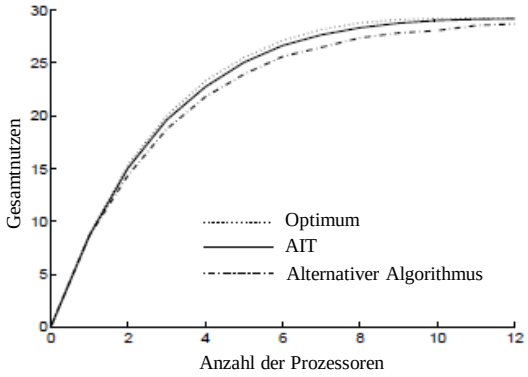

Abbildung 4: Reallokation durch AIT im Fall des Verlusts zweier Prozessoren

Auffällig in beiden Abbildungen ist, dass beide Algorithmen offenbar sehr vielversprechende Resultate liefern. In beiden Abbildungen verläuft die Kurve des jeweiligen Algorithmus nahe der des Optimum. Folglich erreichen beide Reallokationsalgorithmen im Fehlerfall einen Gesamtnutzen, der sich nur geringfügig vom optimalen Gesamtnutzen des Systems unterscheidet. Für den besseren Vergleich des AIT wird zusätzlich ein alternativer Algorithmus evaluiert. Dieser Algorithmus ist bei der Reallokation der Aufgaben eingeschränkt insofern, als dass er zufällig nur einen fehlerfreien Prozessor

nutzt und nicht alle. Demgegenüber wird für den ART zusätzlich eine initiale Allokation im Fall eines Neustart betrachtet.

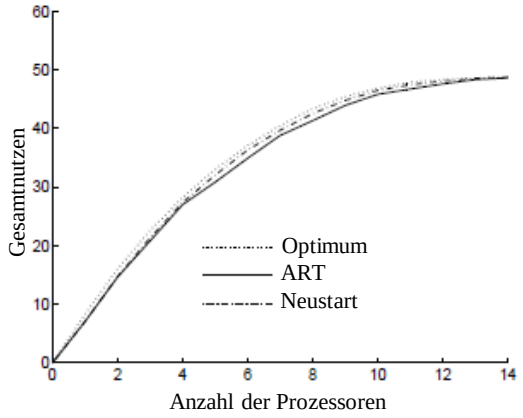

Abbildung 5: Reallokation durch ART im Fall des Verlusts zweier Prozessoren

3.2.3 Fazit

Offenbar liefern beide Algorithmen in einer Rekonfigurationssituation vielversprechende Resultate nahe dem Optimum. Die explizite Betrachtung von Abhängigkeitsbeziehungen unter Aufgaben ist ebenfalls positiv zu bewerten, da sie in der Praxis typischerweise einen gängigen Anwendungsfall repräsentieren. Diese Tatsache fördert allerdings auch den Grund zur Skepsis, da entsprechende Abhängigkeitsbeziehungen in regelungstechnischen Anwendungen nicht selten einen hohen Komplexitätsgrad erreichen können. Das Konzept der Sicherungskopien in Form von Schattenaufgaben ist, wenn auch nur für den speziellen Anwendungsfall eines Multiprozessorsystems einsetzbar, in jedem Fall zukunftsträchtig. Es arbeitet nicht nur ressourcenschonend sondern erlaubt in Rekonfigurationssituationen eine bestmögliche Reaktionslatenz sowie eine minimale Regelgüte gewährleisten zu können. Analog dem Konzept aus [1] ist allerdings auch bei diesem Konzept die Ermittlung des PI und damit des Referenzmaßes zur Kostenabschätzung im Zuge einer Reallokation fragwürdig. Ohne Einschränkung handelt es sich bei dem PI um ein anwendungsspezifisches Referenzmaß, was die Wiederverwendbarkeit und Übertragbarkeit im Allgemeinen verhindert.

4. FAZIT UND AUSBLICK

Beide Konzepte erreichen eine weitaus höhere Flexibilität bei der dynamischen Rekonfiguration in Echtzeitsystemen als herkömmliche Methoden wie der „one-shot“-Ansatz. Das begründet sich darin, dass in beide Konzepte der Vorteil der inhärenten Robustheit einer Regelungsanwendung gezielt einfließt. Insgesamt werden interessante Resultate erzielt, die Hoffnung für die Zukunft bergen. Über die Nachweisbarkeit der Gültigkeit oder Aussagekraft dieser Resultate lässt sich hingegen streiten. Beide Konzepte nutzen im Allgemeinen eine stetige Kontrolle der Regelgüte des Systems, um zur Laufzeit auf Veränderungen reagieren zu können. Die Ermittlung dieses Referenzmaßes sowie dessen eingeschränkte Portabilität bzw. Wiederverwendbarkeit für andere Anwendungen schränkt den Einsatz beider Konzepte in der Praxis aber wohl stark ein.

5. LITERATURVERZEICHNIS

[1] D. Fontanelli, L. Greco, and L. Palopoli. Soft real-time scheduling for embedded control systems. *Automatica*, 49:2330–2338, May 2013.

[2] T. Lang. Dynamic updates for object-oriented operating-system kernels. May 2014.

[3] C. Ünver. Rebootless security patches for the linux kernel. June 2014.

[4] K. G. Shin and C. L. Meissner. Adaption and graceful degradation of control system performance by task reallocation and period adjustment. *Proceedings of the 11th Euromicro Conference on Real-Time Systems*, pages 29–36, June 1999.