

Problem Beschreibung

Bereich

Arbeitsplatz-Rechner (PCs oder Workstations) werden schon seit längerem nicht nur zur Datenverarbeitung eingesetzt. Solche Geräte dienen immer häufiger auch als Kommunikationsgerät, so kann man z.B. heute mit einem PC und geeigneter Software über das Internet telefonieren. Solche Anwendungen verarbeiten Video- und Audiodaten (sogenannte kontinuierliche Daten), diese erfordern jedoch die Berücksichtigung von Realzeit-Bedingungen. Es muß gewährleistet sein, das solche Daten zeitlich gleichmäßig und mit einer möglichst geringen Verzögerung verarbeitet werden. Letzteres ist insbesondere für die interaktive Kommunikation von Mensch zu Mensch erforderlich (z.B. beim telefonieren). Die Architektur heutiger Arbeitsplatz-Rechner und die hier eingesetzten Betriebssysteme sind jedoch darauf ausgelegt diskrete Daten (ohne bezug zur Realzeit) zu verarbeiten. Kontinuierliche Daten werden auf solchen Systemen verarbeitet indem sie in eine Folge von diskreten Dateneinheiten zerlegt werden (logical data units: LDU). Realzeitanforderungen für die Verarbeitung einzelner LDUs werden nicht berücksichtigen oder gar garantiert. Beispiele von Realzeitanforderungen sind: „die Verarbeitung einer LDU muß bis zu einem bestimmten Zeitpunkt (deadline) abgeschlossen sein“ oder „die Verarbeitungszeit für eine LDU darf ein gegebenes Zeitintervall nicht überschreiten“.

Technisches Problem

Auf einem Arbeitsplatz-Rechner konkurrieren üblicherweise mehrere Prozesse um den Zugriff auf die CPU Ressource(n). Neben den von Nutzern gestarteten Anwendungen sind in der Regel noch eine Vielzahl von Service-Prozessen aktiv. Ein Prozess darf eine CPU für ein festgelegtes Zeitintervall¹ belegen, spätestens nach Ablauf dieses Zeitintervalls wird die CPU einem anderen Prozess zugeordnet. Das Umschalten von einem Prozess auf einen anderen (Kontextwechsel) übernimmt der sogenannte Dispatcher. Die Reihenfolge in der die Prozesse der/den CPU(s) zugeordnet werden legt ein Prozess-Scheduler nach bestimmten Verfahren fest (Schedulingstrategie). Heute übliche Betriebssysteme für Arbeitsplatz-Rechner verwenden Schedulingstrategien, die auf veränderbaren Prozess-Prioritäten beruhen. Diese Strategien berücksichtigen keinerlei Realzeit-Anforderungen der Prozesse.

Bisherige Lösung des Problems

Allgemein

Es werden spezielle Prozess-Scheduler verwendet, welche aufgrund ihrer Schedulingstrategie die Realzeit-Anforderungen medienverarbeitender Prozesse berücksichtigen können. D.h. es wird versucht die Verteilung der verfügbaren CPU-Zeit explizit zu steuern. Da der Prozess-Scheduler eine zentrale und essentielle Komponente jedes Betriebssystems darstellt, ist es in der Regel schwierig diese Komponente auszutauschen bzw. zu verändern².

¹ Die Größe dieses Zeitintervalls ist Systemabhängig, üblicherweise ist dieses Zeitintervall für alle Prozesse gleich kann unter Umständen aber auch variable sein. Ein typischer Wert ist 10 ms.

² Einige Betriebssysteme unterstützen die Erweiterung des Prozess-Scheduler, z.B. das Unix Betriebssystem Solaris, die weit verbreitetend Windows Betriebssysteme bieten diese Möglichkeit jedoch nicht an.

Realisierungen

Idealerweise sollten Hersteller von Betriebssystemen bereits Prozess-Scheduler vorsehen, die Realzeit-Anforderungen von Prozessen unterstützen können. Entsprechende Scheduler werden in der Literatur beschrieben.

In der Praxis hat sich jedoch gezeigt, dass auch moderne Varianten der weit verbreiteten Betriebssysteme für Arbeitsplatz-Rechner solche Scheduler nicht bereitstellen. Um trotzdem Realzeit-Anforderungen berücksichtigen zu können, können sogenannte Meta-Scheduler verwendet werden. Diese steuern durch die Veränderung von Prozessprioritäten das Verhalten des Betriebssystemeigenen Prozess-Schedulers. Die Nachteile dieser Vorgehensweise sind:

- Es ist notwendig die Prioritäten der Prozesse sehr häufig zu ändern. Dies ist mit einem hohen Rechenaufwand verbunden.
- Manche Betriebssystemeigene Scheduler verändern ebenfalls die Prozess-Prioritäten, jedoch mit einer anderen Zielsetzung als die Unterstützung von Realzeit-Prozessen (z.B. die Scheduler der Windows Betriebssysteme). Ein Meta-Scheduler würde daher kontraproduktiv zum Betriebssystemeigenen Scheduler arbeiten, dies erschwert die Realisierung eines Meta-Schedulers.

Bei der Verwendung eines Prozess-Schedulers, der Realzeit-Anforderungen berücksichtigen soll, stellt sich allgemein das Problem, dass eine genaue Beschreibung der Realzeit-Anforderungen in der Regel nicht bekannt ist. So ist beispielsweise die Bearbeitungszeit für eine LDU meist nicht bekannt, da dieser Wert von der Verarbeitungsleistung des jeweiligen Systems abhängt. Selbst wenn dieser Wert statistisch ermittelt worden ist, kann dennoch die Bearbeitungszeit für einzelne LDUs stark abweichen.

In der Praxis ist es daher heute üblich keine speziellen Scheduler zu verwenden. Es wird darauf vertraut, dass die Performanz der üblichen Arbeitsplatz-Rechner ausreichend ist für die Verarbeitung von kontinuierlichen Medien. Falls ein Arbeitsplatz-Rechner ausschließlich für diesen Zweck eingesetzt wird ist dies meistens auch der Fall. Problematisch ist aber weiterhin der Fall wenn auf einem Arbeitsplatz-Rechner neben der Medienverarbeitung weitere aktive Prozesse existieren. In diesem Fall werden die Medienverarbeitenden Prozesse aufgrund der Schedulingverfahren immer wieder von anderen Prozessen ohne Realzeitbezug unterbrochen³. Dadurch vergrößert sich die Bearbeitungszeit einer LDU und damit die Verzögerung des gesamten Datenstromes. Insbesondere für die Interaktive Kommunikation von Mensch zu Mensch ist es jedoch wichtig möglichst kurze Verzögerungszeiten zu erreichen, damit eine für den Menschen angenehme Kommunikation möglich ist. Abhängig von der Anzahl und Art der Anwendungen, die ein Nutzer neben den Medienverarbeitenden Prozessen nutzt, kann es zu einer mehr oder weniger starken „Störung“ durch diese non-realtime Prozesse kommen. Zu starke „Störungen“ führen dazu, dass Teile des Medienstromes nicht rechtzeitig ausgegeben werden können. Die Folge sind hörbare Unterbrechungen der Audioausgabe oder das fehlen einzelner Videobilder. Letzteres wird in der Regel erst wahrgenommen, falls mehrere Bilder fehlen oder häufiger einzelne Bilder fehlen. Dagegen kann bereits der Ausfall einer Audio LDU vom Nutzer als Störend wahrgenommen werden.

Weitere Details zu diesem Thema werden u.A. in dem Buch „Multimedia Technologie“ von Ralf Steinmetz beschrieben.

³ Man sagt die Prozesse arbeiten quasi-parallel, da der Wechsel zwischen den Prozessen derart schnell geschieht (z.B. alle 10 ms) das ein Benutzer den Eindruck hat die Prozesse würden gleichzeitig auf einem Rechner ablaufen.

Der neue Lösungsansatz, Merkmale der Erfindung

Konzeptionelle Beschreibung des Lösungsansatzes

Die hier vorgestellte Idee verfolgt einen grundsätzlich anderen Ansatz als die bisherigen Lösungen. Diese Unterschiede können bereits durch unterschiedliche technische Zielsetzungen beschrieben werden. Bisher wurde mit Hilfe spezieller Scheduler versucht bestimmte zeitliche Vorgaben für medienverarbeitende Prozesse einzuhalten. Bei dem hier beschriebenen Ansatz soll lediglich versucht werden medienverarbeitenden Prozessen die gleiche Performanz eines ansonsten unbelasteten Rechners zur Verfügung zu stellen. Das Ziel orientiert sich also nicht an der Einhaltung von Zeitvorgaben, sondern daran die maximale Performanz zur Verfügung zu stellen.

Für die bisher bekannten Lösungen war ein spezieller Scheduler notwendig, der den Zugriff auf die Ressource CPU steuert. Bei diesem Ansatz wird der Zugriff auf die Ressource CPU nur durch bereits im System vorhandene Komponenten gesteuert, es ist jedoch zusätzlich eine neue Komponente notwendig, welche die Ressourcennutzung überwacht und bei Bedarf gezielt einschränken kann. **Die Komponente zur gezielten Einschränkung der CPU-Last ist neben dem neuen Konzept die hier vorgestellte neue Erfindung.** Das neue Konzept wird durch drei Mechanismen realisiert:

- 1) Medienverarbeitende Prozesse werden gegenüber allen anderen Prozesse bevorzugt. Diesen stehen damit die gleichen CPU-Ressourcen zur Verfügung wie auf einem unbelasteten System. Diese Bevorzugung geschieht durch die Verwendung sehr hoher Prioritäten. Am besten geeignet ist der sogenannte Realtime-Modus⁴, den die meisten Betriebssysteme anbieten.
- 2) Überwachung der CPU-Last medienverarbeitender Prozesse. Dies kann durch einfache Messverfahren geschehen, welche meist bereits von den Betriebssystemen realisiert werden und lediglich geeignet abgerufen werden müssen.
- 3) Die CPU-Last medienverarbeitender Prozesse muss gezielt eingeschränkt werden können, falls ein gegebenes Limit überschritten wird. Dieser Mechanismus ist neu entwickelt worden und funktioniert ohne den Zugriff auf die Ressource CPU direkt zu steuern.

Das technische Problem bei der Realisierung dieses Mechanismus liegt darin, dass eine Einschränkung der Nutzung von CPU-Zeit im allgemeinen nur durch den Prozess-Scheduler erfolgen kann. Hier wird jedoch eine Eigenschaft medienverarbeitender Prozesse ausgenutzt, so dass eine spezielle Lösung für solche Prozesse gefunden werden kann. Diese haben die Eigenschaft, dass sie in mehr oder weniger gleichmäßigen zeitlichen Abständen LDUs verarbeiten müssen. Falls keine LDUs zur Verarbeitung vorhanden sind, zeigen solche Prozesse (unter bestimmten Voraussetzungen) keine Aktivität, d.h. sie benötigen die CPU Ressourcen nicht. Daher ist es grundsätzlich möglich den Bedarf an CPU Ressourcen medienverarbeitender Prozesse zu senken, indem LDUs aus dem Datenstrom entfernt und verworfen werden. Diesen Effekt unter Verwendung geeigneter Steuerungsmechanismen auszunutzen, stellt eine neue Lösung des technischen Problems (Steuerung des CPU-Zeit Bedarfs) für den Spezialfall medienverarbeitenden Prozesse dar.

⁴ Der Realtime-Modus wird von den meisten gängigen Betriebssystemen angeboten (z.B. Windows NT, Windows 2000, Solaris, AIX, ...). Durch die Verwendung der höchsten Priorität, die für Standard-Applikationen vorgesehen ist, kann nicht das gleiche Maß an Bevorzugung erreicht werden, wie durch die Verwendung des Realtime-Modus.

Technische Beschreibung des Lösungsansatzes

Die Erfindung besteht zum einem aus dem oben beschriebenen neuen Konzept: die von Medienverarbeitenden Prozessen erzeugte CPU-Last indirekt über den Datenstrom zu steuern. Zum anderen besteht die Erfindung aus neuen Verfahren, welche die Regelung der CPU-Last durch die Steuerung des Datenflusses ermöglichen:

- Bei Regelungsausgaben wird üblicherweise unterschieden zwischen einem Regler und einer (Regel-)Strecke (siehe Anhang: Der Regelkreis). Die Abbildung des Stellwertes der Strecke auf die Regelung des Datenflusses medienverarbeitender Prozesse stellt ein neues Verfahren dar. Dieses ist zudem aufgrund vieler Randbedingungen, die für den praktischen Einsatz notwendig oder zumindest hilfreich sind nicht trivial. Folgende Randbedingungen werden von der Abbildung berücksichtigt:
 - Berücksichtigung von Prozessprioritäten. Verschiedene medienverarbeitende Prozesse sind für den Nutzer unterschiedlich wichtig, daher wird der Ressourcenverbrauch der einzelnen Prozesse unterschiedlich geregelt. Mit Hilfe einer Hauptpriorität wird eine feste Vorrangrelationen zwischen Prozessen definiert. D.h. der Ressourcenverbrauch eines Prozesse mit hoher Priorität wird erst dann eingeschränkt⁵, wenn der Ressourcenverbrauch aller Prozesse mit einer niedrigeren Priorität bereits maximal reduziert wurde. Beispiel: Bei einer Videokonferenz ist die Sprachübertragung deutlich wichtiger als die Bildübertragung, es ist also sinnvoll zunächst nur den Ressourcenverbrauch der Videoübertragung zu beschränken, bevor der Ressourcenverbrauch der Audioübertragung reduziert wird. Weiterhin kann mit Hilfe einer Sub-Priorität eine relative Gewichtung zwischen Prozessen gleicher Hauptpriorität definiert werden, so dass der Ressourcenverbrauch eines Prozesses stärker beschränkt wird als der eines anderen.
 - Die Abbildung berücksichtigt, dass der Datenfluß verschiedener medienverarbeitender Prozesse an unterschiedlichen Stellen des Verarbeitungsprozesse gesteuert werden muss.
 - Der Regler des Regelkreise erzielt i.d.R. die besten Ergebnisse, wenn das zu steuernde System linear auf Veränderungen des Actuatorwertes reagiert. Dies ist jedoch im vorliegende Anwendungsfall eher unwahrscheinlich. Die Abbildung verwendet daher weitere Funktionen (Anpassungsfunktionen) als Parameter, die es bei Kenntnis des Prozessverhaltens ermöglichen, ein lineares Verhalten anzunähern bzw. zu erreichen.
- Ein weiteres neues Verfahren ist der Notfall-Mechanismus, welcher in Fehler-situationen einem Prozess die Real-Time Priorität entziehen kann und somit eine Blockade des gesamten Systems verhindert. Durch die Steuerung des Datenflusses kann die von einem Prozess erzeugte CPU-Last gesteuert werden. Es sind jedoch Ausnahmesituationen denkbar, in denen dieser Steuerungsmechanismus nicht mehr funktioniert. So kann z.B. ein Prozess aufgrund eines Programmierfehlers eine endloslange Berechnung durchführen („abstürzen“). Der Verbrauch der Ressource CPU-Zeit ist dann nicht mehr abhängig von der Datenrate. Der Notfall-Mechanismus erkennt, einen extrem hohen Ressourcenverbrauch und setzt daraufhin die Priorität des entsprechenden Prozesses auf ein normales Niveau zurück. Dadurch wird eine dauerhafte Blockade des gesamten Systems vermieden und der Nutzer kann eine geeignete Massnahme zur Beseitigung des Fehlers einleiten.

⁵ Eine Beschränkung des Ressourcenverbrauches führt zwnagsweise zu einer Verschlechterung der Darstellungsqualität.

Was ist die „Erfindung“

Erfinderische Leistung

Der neue Ansatz (siehe „Konzeptionelle Beschreibung des Lösungsansatzes“) zur Lösung eines lange bekannten Problems stellt die wesentliche Neuerung dar. Damit dieser Ansatz gefunden werden konnte, wurden die üblichen herangehensweisen an dieses Problem neu überdacht. Bisher wurden die Nutzung der Ressource CPU (CPU-Last) durch eine direkte Steuerung des Zugriffs (Scheduler) kontrolliert. Hier wird nun die Nutzung der Ressource CPU indirekt über die Steuerung des Datenflusses geregelt. Auf diese Weise können die Schwierigkeiten der direkten Kontrolle vermieden werden. Da jedoch die CPU-Last nicht in einem linearen Verhältniss zur Menge des Datenflusses steht und meist sogar dieses Verhältniss unbekannt ist, entsteht ein komplexeres Steuerungsproblem. Dieses in geeigneter Weise zu Lösen ist die hier beschriebenen technische Erfindung (siehe „Technische Beschreibung des Lösungsansatzes“).

Vorteile der Erfindung

Dieser Lösungsansatz besitzt folgende Vorteile gegenüber der Verwendung von Schemulern oder Meta-Schemulern zur Berücksichtigung von Realzeit-Anforderungen:

- Falls kein Regulierungsbedarf besteht – dies dürfte in vielen praktischen Fällen der Normalfall sein – wird nur ein geringer zusätzlicher Overhead (bzgl. Der benötigten CPU-Zeit) erzeugt. Es muss lediglich der CPU-Zeit Bedarf einiger Prozesse überwacht werden anstatt den Zugriff auf die CPU ständig zu steuern.
- Es wird nicht versucht den Zugriff auf eine Ressource (die CPU) zu steuern, auf die Prozesse (normalerweise⁶) keinen Zugriff haben. Kernelmodifikationen oder die Steuerung des systemeigenen Prozess-Schemuler über modifikation von Prioritäten ist nicht notwendig.
- Falls Aufgrund zu knapper Ressourcen Daten des Medienstromes verworfen werden müssen, kann dies an definierten und sinnvollen Stellen bzw. Zeitpunkten geschehen. So können z.B. bevorzugt Daten verworfen werden, die bisher nur wenige Verarbeitungsschritte durchlaufen haben. Daten, in die bereits viel CPU-Zeit „investiert“ wurde können soweit wie möglich noch verarbeitet werden. Eine solche Unterscheidung ist bei der Verwendung von Schemulern nicht möglich, da die Schemuler keine Informationen über den Status der Datenverarbeitung eines Prozesses besitzen. Normalerweise werden Daten verworfen, sobald eine gegebene Deadline überschritten wurde. Falls Daten nach einer bestimmten Strategie verworfen werden sollen, muss dies durch zusätzlich Mechanismen geregelt werden.
- Die Kenntniss von konkreten Realzeitanforderungen ist nicht erforderlich. Das Ziel ist lediglich die gleiche Performanz für die Medienverarbeitung zu erreichen, die auch auf einem System ohne zusätzliche non-realtime Prozesse erreicht wird. Scheduling Verfahren verfolgen dagegen das Ziel Daten bis zu einer gegebenen Deadline verarbeitet zu müssen, so dass zur Planung des Scheduling Zeitangaben wie Verarbeitungsdauer oder die Deadline notwendig sind. Die (unter Umständen schwierige) Feststellung solcher Parameter entfällt bei dem hier vorgestellten Ansatz.

⁶ Nur wenige Betriebssysteme erlauben eine Erweiterung des systemeigenen Prozess-Schemuler, z.B. Solaris.

Beispiel der Realisierung

Technische Beschreibung: Abbildung eines globalen Stellwertes auf die Steuerung des Datenflusses

Die Abbildung des globalen Stellwertes auf die Steuerung des Datenflusses wird in mehreren Schritten durchgeführt (siehe auch Abbildung 1):

- 1) Abbildung des globalen Actuators auf den Actuator einer Prozessklasse. Hierfür wird der Wertebereich des globalen Actuators entsprechend der vorhandenen Hauptprioritäten der Prozesse in Klassen unterteilt. Anschließend wird anhand des aktuellen Wertes des globalen Actuators eine Klasse bestimmt. Nur die Prozesse, die dieser Klasse angehören werden geregelt. Prozesse, die einer Klasse niedrigerer Priorität angehören werden vollständig angehalten. Prozesse, die einer Klasse höherer Priorität angehören laufen ohne Beschränkung des Ressourcenverbrauchs.
- 2) Abbildung des Actuators einer Prozessklasse auf Actuatoren für einzelne Prozesse (=Streams⁷). Mit Hilfe einer Adaptionfunktion wird für jeden Stream der Klasse der Actuator der Prozessklasse auf den jeweiligen Stream-Actuator abgebildet. Die Adaptionfunktion ist dabei abhängig von der Subpriorität des jeweiligen Streams.
- 3) Abbildung des Stream-Actuators auf die einzelnen Connection-Actuatoren. Jedes Verbindungsobjekt besitzt eine Priorität. Es wird das gleiche Verfahren wie unter 2 beschrieben verwendet, um den Stream-Actuator auf die jeweiligen Connection-Actuatoren abzubilden
- 4) Verwerfen von Daten in Abhängigkeit des Connection-Actuators. Da immernur ganze LDUs verworfen werden können, wird ein Zähler bestimmt der festlegt, wie oft eine LDU aus dem Datenstrom entfernt wird. Bei der Bestimmung des Zählers werden nicht nur ganzzahlige Werte berücksichtigt. Es ist z.B. möglich abwechselnd jede 3'te bzw. jede 4'te LDU zu verwerfen, dies entspricht dem Verwerfen jeder 3,5'ten LDU. Weiterhin werden Änderungen des Connection-Actuators so auf den Zählerwert übertragen, dass eine ein neuer Actuatorwert nicht einem Neustart des Zählvorganges erfordert. Damit ist es möglich regelmäßig neue Actuatorwerte zu bestimmen ohne dadurch die Regelung zu beeinflussen.

Im Anhang (Abbildung eines globalen Stellwertes auf die Steuerung des Datenflusses) wird eine detaillierte Beschreibung dieser Schritte gegeben. Die dort beschriebenen Verfahren wurden bereits in einem Prototyp implementiert und sind unter Windows NT lauffähig.

⁷ Ein Prozess entspricht einem Stream siehe Beschreibung unter Voraussetzungen.

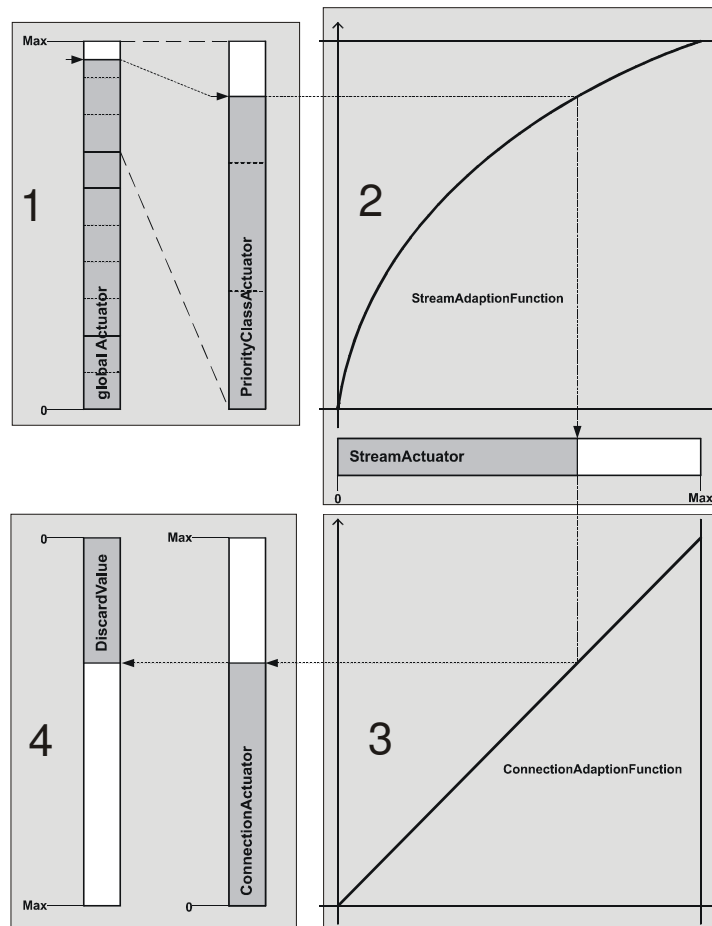


Abbildung 1: Abbildung des globalen Actuators auf die Datenflußsteuerung

Technische Beschreibung: Notfall-Mechanismus

Falls ein Prozess nicht mehr über den Datenfluß gesteuert werden kann, z.B. weil dieser Prozess „abgestürzt“ ist, muss ein Notfall-Mechanismus greifen, der einen solchen Prozess auf eine normale Prioritätsstufe zurückversetzt. Andernfalls könnte ein solcher Prozess das gesamte System blockieren. Dazu wird überwacht, ob die Gesamtheit der medienverarbeitenden Prozesse einen bestimmten Schwellwert für die CPU-Last für einen „längeren“ Zeitraum (z.B. 1 Sekunde) überschreitet. Dieser Schwellwert muss entweder deutlich höher liegen als der Schwellwert, bei dem der oben beschriebene Regelungsprozess einsetzt. Alternativ kann auch ein wesentlich längerer Zeitraum betrachtet werden. Im dem Fall, dass der Notfall-Mechanismus greift wird folgendes Verfahren angewendet:

- 1) Wähle den medienverarbeitende Prozesse aus, welcher die meiste CPU-Last erzeugt
- 2) Setze diesen Prozesse auf eine normale Priorität zurück
- 3) Verfahre weiter mit Schritt 1, wenn die CPU-Last der verbleibenden medienverarbeitenden Prozesse (im Realtime Modus) noch oberhalb des Schwellwertes für das Eingreifen des Regeleungsmechanismus liegt.

Dieser Mechanismus ist generell dazu geeignet überlast Situationen zu beheben, welche durch den Regelungsmechanismus nicht oder nur schlecht kontrolliert werden

können. In speziellen Situtation ist es auch denkbar ausschließlich diesen Mechanismus zu verwenden, wenn davon ausgegangen werden kann, dass eine Regelung des Datenflusses niemals nötig ist aufgrund der niedrigen erwarteten CPU-Last. Soll z.B. lediglich ein Audiostrom verarbeitet werden, so liegt die dafür erforderliche CPU-Last auf heutigen Desktop-System üblicherweise deutlich unter 10%. Eine Regelung des der CPU-Last über den Datenfluss wird also voraussichtlich niemals nötig sein. Bei einer dauerhaften Überschreitung der CPU-Last von z.B. 40% kann von einer Fehlfunktion ausgegangen werden und der Prozess zur Verarbeitung des Audiostromes auf eine normale Priorität zurückgestuft werden.

Technische Beschreibungen von bekannten Verfahren

Der Regelkreis

Der Regelkreis (siehe Abbildung 2) besteht im wesentlichen aus dem Regler, einer (Regel-)Strecke und einem Messverfahren. Mit Hilfe eines geeigneten Messverfahrens wird regelmäßig der Zustand des zu steuernden System festgestellt und als Ist-Wert $x(t)$ für den Regelkreis zur Verfügung gestellt. Dem Regelkreis wird ein Sollwert $w(t)$ vorgegeben und mit dem Ist-Wert $x(t)$ verglichen, deren Differenz $x_d(t)$ dient periodisch als Eingabewert für den Regler. Der Regler berechnet daraufhin einen neuen Stellwert $y(t)$ für die Strecke. Dieses Prinzip ist bekannt und stammt aus der Regelungstechnik, ebenso wie Berechnungsverfahren für den Regler. Ebenfalls bekannt ist das hier verwendete Messverfahren. Neu ist hier das Anwendungsgebiet dieses Regelungsverfahrens, sowie die Abbildung des Stellwertes der Strecke (auch Actuatorwert) auf die Steuerung des Datenflusses medienverarbeitender Prozesse.

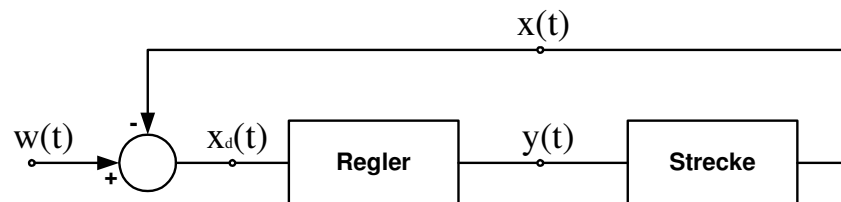


Abbildung 2: Der Regelkreis

Das Stream-Konzept

Die Verarbeitung von Medienströmen (also Ströme von Logical Data Units = LDUs) wird üblicherweise nach dem sogenannten Stream-Konzept durchgeführt (siehe Abbildung 3). Die Verarbeitungsschritte werden von einzelnen Stream-Filtern realisiert. Diese werden über Verbindungsobjekte je nach Bedarf miteinander verbunden, so dass ein Verarbeitungsprozess entsteht, welcher dann auch als Stream bezeichnet wird. Dieser kann eine Folge sequentieller Abläufe sein, aber auch komplexere Formen annehmen. Die Arbeit des Streams wird von einem Stream-Manager gesteuert. Eine Anwendung kommuniziert lediglich mit diesem Manager und steuert über Kommandos (z.B. Start, Stop, Pause, ...) den Stream. Ein solcher Stream ist dann nicht mehr notwendigerweise ein Teil der Applikation, sondern kann als eigenständiger Prozess laufen.

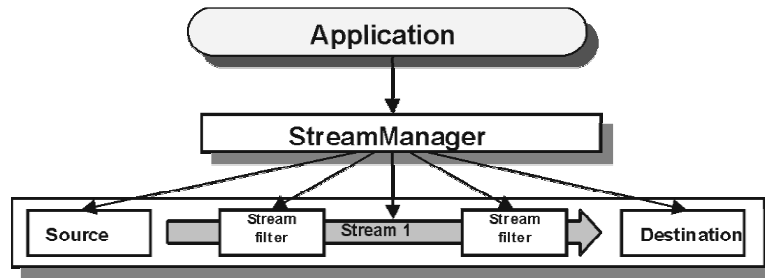


Abbildung 3: Stream-Konzept Prinzip

Detaillierte technische Beschreibungen der Erfindung

Abbildung eines globalen Stellwertes auf die Steuerung des Datenflusses

Das Ziel dieser Abbildung ist es an Hand eines globalen Stellwertes den Ressourcenverbrauch verschiedener Prozesse zu regulieren. Die Regulierung geschieht durch das Verwerfen von sogenannten Logical Data Units (LDU).

Der globale Stellwert kann Werte innerhalb eines bestimmten Intervalls (0..10000) annehmen. Wird das Maximum erreicht, werden im gesamten System keine LDUs verworfen. Wird 0 erreicht, sind alle Streams des Systems abgeschaltet.

Streams erhalten zwei Prioritäten - die Hauptpriorität (0..1000) und die Subpriorität (0..1000). Je kleiner der Zahlenwert ist, desto höher ist die Priorität. Die Hauptpriorität unterteilt die Streams in Klassen, während die Subpriorität innerhalb dieser Klassen für eine Rangfolge sorgt. Die Prioritätsklasse mit der niedrigsten Hauptpriorität (höchster Wert) wird als erste zur Regelung herangezogen. Dazu wird das Intervall des Hauptstellwertes analog zu der Zahl der Prioritätsklassen partitioniert, gewichtet nach der Anzahl der Streams in jeder Klasse.

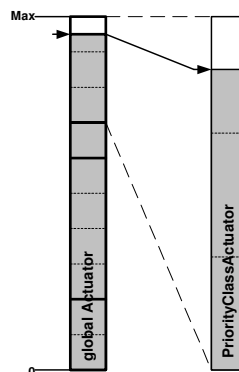


Abbildung 4

Abbildung 4 zeigt beispielhaft einen partitionierten Hauptstellwert, der in 4 Partitionen aufgeteilt wurde. Im Beispiel existieren 4 Prioritätsklassen mit 4 unterschiedliche Hauptprioritäten bei insgesamt 10 Streams. Innerhalb der Prioritätsklasse mit der niedrigsten Priorität laufen 3 Streams – die Prioritätsklasse befindet sich am oberen Ende der linken Säule. Die Position des globalen Stellwertes wird auf die aktuelle Partition umgerechnet. So ergibt sich der Stellwert der Prioritätsklasse (0..1000). Wechselt der globale Stellwert in eine Prioritätsklasse mit höherer Priorität, so sind alle Streams der jeweils niedrigeren komplett abgeschaltet.

Innerhalb einer Prioritätsklasse können mehrere Streams mit jeweils unterschiedlichen Subprioritäten laufen. Aus dem Stellwert der Prioritätsklasse wird ein Stellwert für den jeweiligen Stream generiert (0..1000). Dazu werden verschiedene Anpassungsfunktionen eingeführt, die diese Umsetzung unterschiedlich vornehmen können. Gemeinsam ist allen Anpassungsfunktionen der Kurvenscharparameter P, der die Subpriorität darstellt.

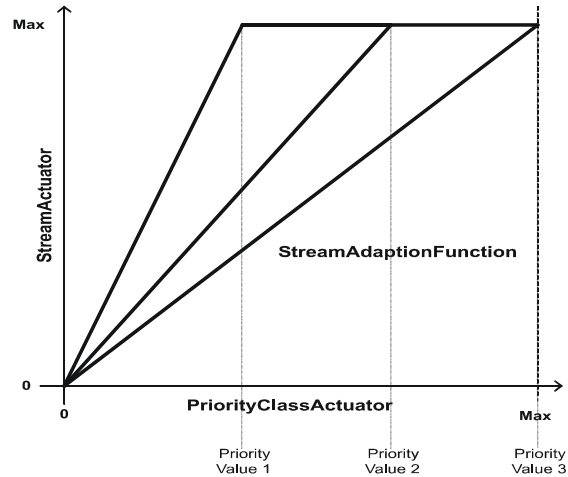


Abbildung 5

Abbildung 5 zeigt die einfachste Anpassungsfunktionsschar basierend auf einer linearen Funktion. P steuert, ab welchem Wert der Stream-Stellwert (SA), ausgehend vom Maximalwert, beginnt, mit linearem Verlauf abzusinken, bis mit dem 0-Durchgang des Prioritätsklassen-Stellwerts (PCA) für alle Anpassungsfunktionen ebenfalls der Wert 0 erreicht wird. Bei einem Prioritätswert von 0 wird an diesen Stream immer nur der Maximalwert (1000) für den Stream-Stellwert weitergegeben:

$$SA = \begin{cases} 1000 & PCA > P \\ 1000 \frac{PCA}{P} & sonst \end{cases}$$

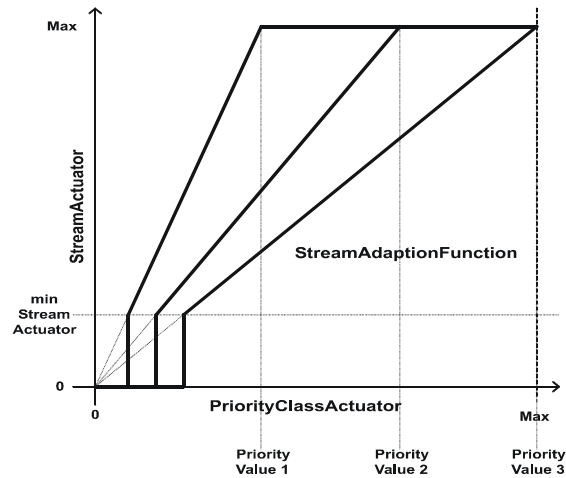


Abbildung 6

In Abbildung 6 wird ein weiterer sinnvoller Parameter verdeutlicht, nämlich ein Schwellwert für den Stream-Stellwert, der nicht unterschritten werden darf. Wird der Prioritätsklassen-Stellwert so klein, dass nach der Adaptionfunktion der Schwellwert für den Stream unterschritten würde, schaltet der Stream ab. Durch die frühzeitige Abschaltung eines Streams wird vermieden, dass ein Medium mit extrem schlechter Qualität dargestellt wird. Dies dürfte i.d.R. als störender empfunden werden als ein nicht vorhandener Datenstrom.

Sinnvoll ist in diesem Zusammenhang auch die Anwendung einer Hystereseeigenschaft, damit nach dem Abschalten eines Streams dieser nicht sofort wieder aktiviert wird und auf diese Weise ein Schwingen des Regelkreises erzeugt.

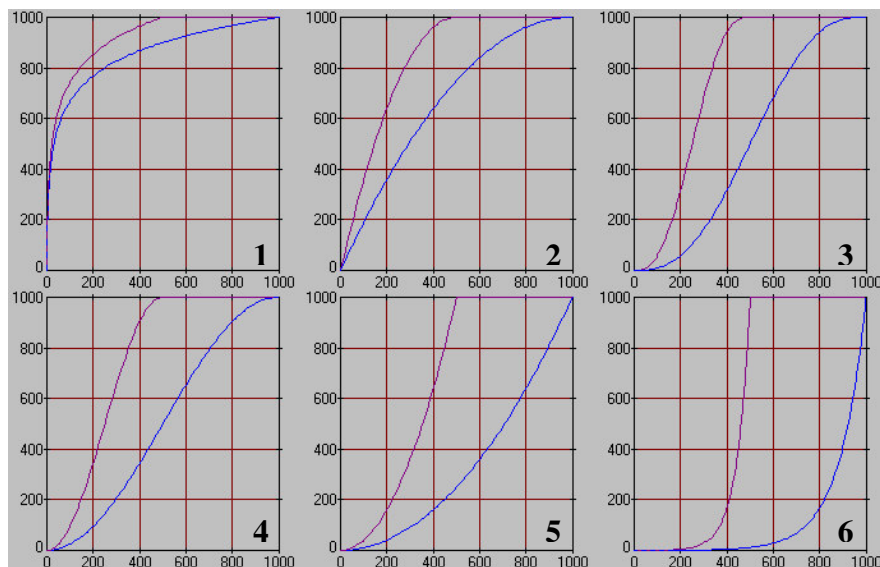


Abbildung 7

Abbildung 7 zeigt sechs weitere Anpassungsfunktionen, die in der ersten Realisierung zur Verfügung stehen. jeweils mit dem Verlauf der Anpassungsfunktion für zwei Prioritäten und einen Schwellwert von 0. Die Möglichkeit unterschiedlicher Funktionen

Adaptionsfunktionen zu Verwenden, soll es erleichtern eine lineares Verhalten der Regelstrecke zu erreichen. Im Folgenden sind die Funktionsgleichungen aufgeführt:

1) Logarithmische Anpassungsfunktion:

$$SA = \begin{cases} 1000 & PCA > P \\ 1000 \frac{\ln(PCA+1)}{\ln(P+1)} & sonst \end{cases}$$

2) Quadratische Anpassungsfunktion:

$$SA = \begin{cases} 1000 & PCA > P \\ -\frac{1000}{P^2} PCA^2 + \frac{2000}{P} PCA & sonst \end{cases}$$

3) Sinusförmige Anpassungsfunktion:

$$SA = \begin{cases} 1000 & PCA > P \\ 500 \left(\sin\left(\frac{\pi * PCA}{P} - \frac{\pi}{2}\right) + 1 \right) & sonst \end{cases}$$

4) Polynomiale Anpassungsfunktion:

$$SA = \begin{cases} 1000 & PCA > P \\ \frac{6000}{P^5} PCA^5 - \frac{15000}{P^4} PCA^4 + \frac{10000}{P^3} PCA^3 & sonst \end{cases}$$

5) Quadratische Anpassungsfunktion:

$$SA = \begin{cases} 1000 & PCA > P \\ \frac{1000}{P^2} PCA^2 & sonst \end{cases}$$

6) Exponentielle Anpassungsfunktion:

$$SA = \begin{cases} 1000 & PCA > P \\ e^{\frac{PCA * \ln(1000 + e^{-2}) + 2}{P} - 2} - e^{-2} & sonst \end{cases}$$

Ist ein Schwellwert definiert, so wird das Ergebnis für SA auf 0 gesetzt, wenn der berechnete Wert den Schwellwert unterschreitet. Ist der Prioritätswert 0 (höchste Priorität), so wird SA immer der maximale Stellwert (1000) zugeordnet.

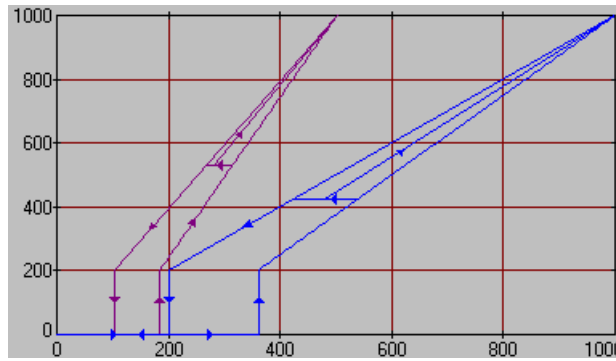


Abbildung 8

Damit weitere Nichtlinearitäten der Strecke untersucht werden können gibt es für Streams eine weitere Anpassungsfunktion, die der Strecke eine Hysterese-Eigenschaft verleiht, wenn ein Schwellwert definiert wird. Abbildung 8 zeigt den Verlauf der Anpassungsfunktion für zwei Prioritäten und einen Schwellwert. Hier ist es von der Vorgeschichte abhängig, welcher Wert als Stream-Stellwert zurückgegeben wird. Der nichtlineare Sprung findet bei unterschiedlichen Prioritätsklassen-Stellwerten statt, je nachdem, ob der Stream ein- oder abgeschaltet wird und auch im eingeschalteten Zustand wird der zurückgelieferte Wert von zwei unterschiedlichen linearen Funktionen nach unten und oben begrenzt.

Der auf die Streams verteilte globale Stellwert wird nun innerhalb des Streams auf die Verbindungs-Objekte umgebrochen werden. Die Verbindungen erhalten bei ihrer Erzeugung eine Priorität (0..1000), die sie gegenüber anderen Verbindungen desselben Streams gewichtet. Die für Streams verwendeten Anpassungsfunktionen werden auf Verbindungsebene erneut genutzt, um aus dem Stream-Stellwert einen prioritätsabhängigen Verbindungsstellwert zu berechnen. Jedoch wird hier auf den Mindeststellwert verzichtet.

Somit erhält man aus dem globalen Stellwert einen Verbindungsstellwert, aus dem direkt die Wegwerfrate für LDUs abgeleitet werden kann. Abbildung 9 visualisiert noch einmal diesen Weg jeweils für die niedrigsten Prioritäten (1000), eine quadratische StreamAnpassungsfunktion, eine lineare Verbindungs-Anpassungsfunktion und einen Schwellwert von 0:

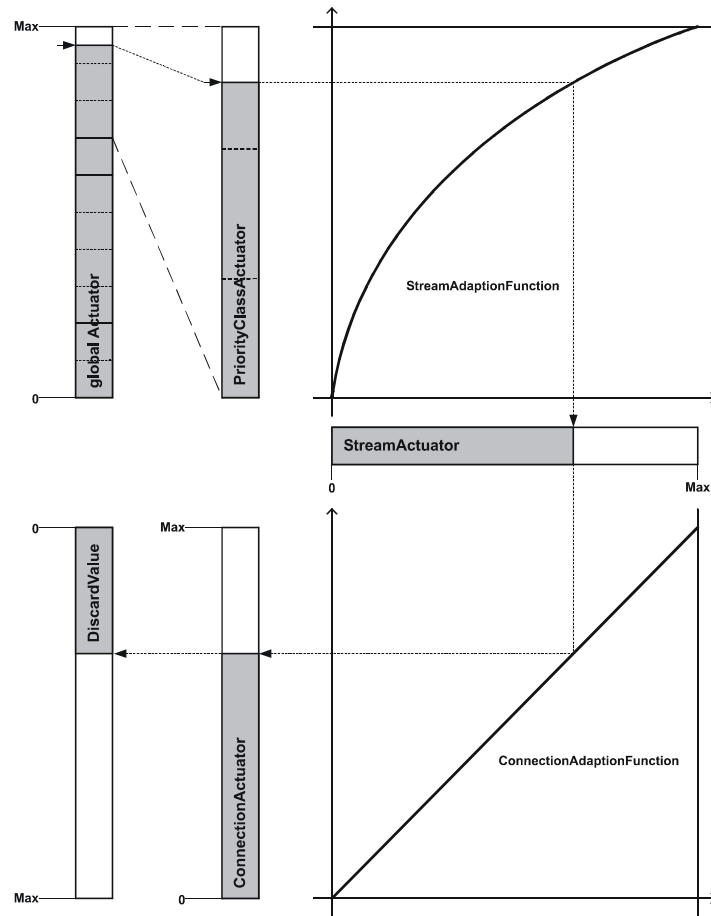


Abbildung 9

Fällt der globale Stellwert so weit ab, dass er in den Bereich der zweiten Prioritätsklasse fällt, so garantiert das System, dass alle Streams der darüberliegenden Partition abgeschaltet werden und nun innerhalb der zweiten Partition derselbe Mechanismus greifen kann wie schon vorher für die erste (Abbildung 10).

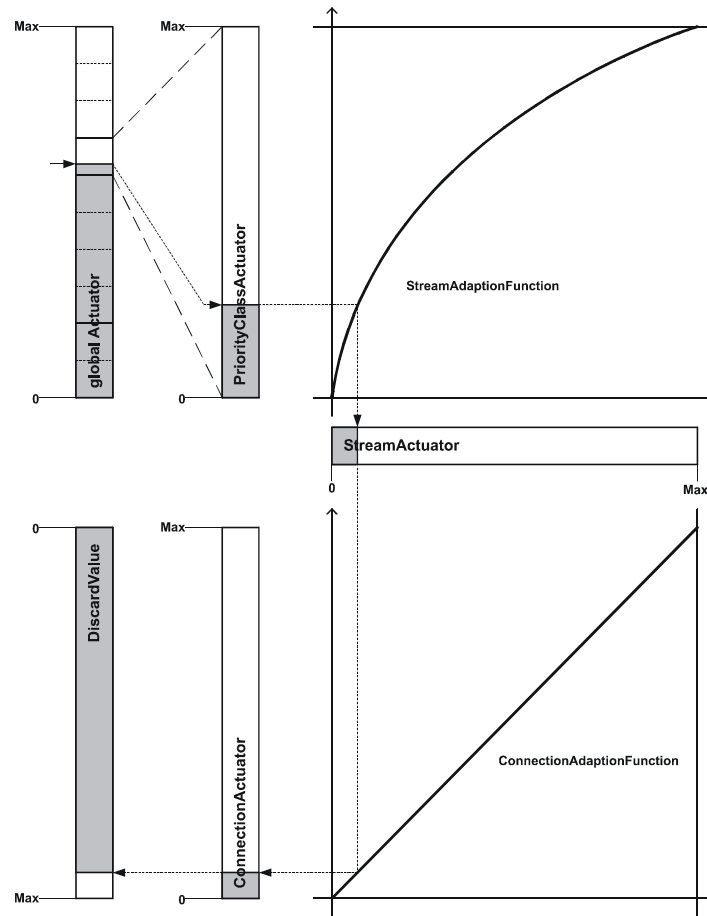


Abbildung 10

Schließlich müssen noch Daten in Abhängigkeit des Connection-Actuators verworfen werden. Aus dem Wert des Connection-Actuators wird ein reeller Zähler bestimmt. Für jede Dateneinheit, die ein Verbindungsobjekt durchläuft wird dieser Zähler um 1 verringert, fällt dabei der Zählerwert unter den ganzzahligen Wert 1 so wird die aktuelle Dateneinheit verworfen. Anschließend wird der Rest des Zählers auf den Initialen Zählerwert addiert. Auf diese Weise können nicht ganzzahlige Zählerwerte berücksichtigt werden. Wenn der Connection-Actuator verändert wird, dann wird ein neuer initialer Zählerwert berechnet, der aktuelle Zählerwert bleibt erhalten, falls dieser niedriger ist als der neue initiale Wert. Andernfalls wird der aktuelle Zähler auf den neuen initialen Zählerwert gesetzt.