Pufferdimensionierung für schwankungsbeschränkte Ströme in DROPS

Dr. rer. nat. Claude-Joachim Hamann, Dipl.-Inf. Lars Reuther, Technische Universität Dresden

Kurzfassung

Hohe Dienstgüteanforderungen heutiger Echtzeitapplikationen einerseits und leistungsfähige Hardware-Architekturen andererseits bedingen einen effizienten Systementwurf bereits auf der Ebene des Betriebssystems. Dem dient u.a. die vorausschauende Reservierung von Betriebsmitteln in einem Umfang, der den Anforderungen der Applikationen möglichst genau angepaßt ist. In dem Beitrag wird an einem ausgewählten Gegenstand – der Behandlung von Echtzeitdatenströmen – der Weg skizziert, mit dem im Rahmen des DROPS-Projekts (***D**resden **R**eal-Time **OP**erating **S**ystem*) dieses Ziel erreicht werden soll. Dabei erfolgt die Bestimmung der zur verlustlosen Datenübertragung erforderlichen Pufferkapazität mittels eines einfachen, aber wirkungsvollen deterministischen Modells „schwankungsbeschränkter Strom".

1 Einleitung

Zu den charakteristischen Anforderungen für verteilte Multimedia-Applikationen gehört die Koexistenz von dynamischen Echtzeitanwendungen und von „klassischen" Timesharing-Anwendungen sowohl auf Host-Rechnern als auch im Netzwerk, verbunden mit hohen Anforderungen an Dienstgüte (Quality of Service, QoS), wie Ende-zu-Ende-Verzögerung, Übertragungsbandbreite, Pufferbedarf und garantierte Diensterbringung über einen längeren Zeitraum. Während moderne Netzwerktechnologien (vorrangig ATM) entsprechende QoS-Leistungen wie etwa das Reservieren von Bandbreite anbieten, ist dies bei den Host-Rechnern zumindest auf Betriebssystem-Ebene üblicherweise nicht der Fall. Das in Entwicklung befindliche Betriebssystem DROPS soll diese Situation verbessern [1]. Im Mittelpunkt des Systementwurfs stehen die folgenden Gesichtspunkte:

- Während es sich bei traditionellen Echtzeitsystemen weitgehend um dedizierte Systeme handelt, denen die gesamte Rechnerleistung zur Verfügung steht, wurde mit dem Aufkommen von Multimedia-Applikationen die geteilte Nutzung von Systemressourcen wie Rechnerkern, Externspeicher, Video-Subsystem und Netzwerk erforderlich. Dabei sind die Echtzeitanwendungen dynamisch in zweierlei Hinsicht: einerseits werden „ständig" Applikationen gestartet und andere beendet, andererseits kann eine einzelne Anwendung sowohl stark schwankende als auch adaptierbare Ressourcenanforderungen haben. Traditionell werden diese Probleme entweder ignoriert oder durch enormes Verschwenden von Ressourcen gelöst. Mit DROPS wird versucht, die Koexistenz von Timesharing- und Echtzeitanwendungen unterschiedlichster Anforderungen durch einen entsprechenden Architekturentwurf – zu dessen wesentlichen Bestandtteilen sog. Ressourcen-Manager auf Betriebssystem-Ebene gehören – zu organisieren. Ein Ressourcen-Manager reserviert für eine Echtzeitanwendung das jeweilige Betriebsmittel, er kann es dynamisch anfordern oder freigeben, und die verbleibende Kapazität des Betriebsmittels wird den Timesharing-Anwendungen zur Verfügung gestellt [2].
- Die Fortschritte in der Entwicklung der Rechnerarchitektur besonders des letzten Jahrzehnts ermöglichten erhebliche Leistungssteigerungen für Timesharing-Systeme durch Ausnutzen des beobachteten lokalen Verhaltens derartiger Systeme. Allerdings sind etliche der Techniken nicht sinnvoll nutzbar für Echtzeitsysteme, die nicht auf den durchschnittlichen Fall, sondern für worst-case-Situationen ausgelegt sein müssen (z.B. Caches). Daher nutzen viele dedizierte Echtzeitsysteme teure, aber weniger mächtige Architekturen. DROPS versucht, einfache leistungsfähige Standardtechnologie vorhersagbar zu machen [3].
- Ein komfortables System sollte sowohl Systementwicklern als auch Anwendern eine weit verfügbare, leistungsfähige Timesharing-Schnittstelle anbeiten. Dies erfolgt in DROPS mit einem Linux-API als standardmäßigem Timesharing-Interface.

Im Gegensatz zu weit verbreiteten Auffassungen wird mit DROPS das Ziel verfolgt, bereits auf Betriebssystem-Ebene diesen Gesichtspunkten gerecht zu werden, da andernfalls weder Betriebsmittelreservierung noch Ausnutzung moderner Hardware-Architektur effizient möglich sind. Diesem Gedanken trägt auch eine den Entwurf begleitende mathematische Modellierung Rechnung. Anliegen und Vorgehensweise sollen mit dem folgenden Beitrag verdeutlicht werden, wozu ein ausgewählter Problemkreis betrachtet wird, nämlich der Transport von Echtzeitdaten zwischen einzelnen DROPS-Komponenten. Dazu werden zunächst anhand eines Anwendungsbeispiels Datenströme und deren Einplanung in DROPS beschreiben, gefolgt von der Darlegung eines einfachen, aber leistungsfähigen Modells, mit dem die Arbeitslast der Systemkomponenten einheitlich beschrieben werden kann und das im letzten Teil die Ermittlung des Pufferbedarfs ermöglicht, der für das fristgerechte Weitergeben von Datenpaketen erforderlich ist.

2 Datenströme

2.1 Anwendungsmodell

In dem DROPS zugrundeliegenden Modell für Echtzeitanwendungen werden Datenströme durch eine Kette von Einzelkomponenten bearbeitet, wobei die Bearbeitung der Daten in der Regel periodisch erfolgt (s. **Bild 1**). Als ein typisches Beispiel sei an eine Video-Präsentation erinnert, bei der ein auf einem Plattengerät gespeichertes MPEG-Video (unter Mitwirkung der Komponenten SCSI-Treiber und Dateisystem) durch einen Dekoder einerseits einer Präsentations-, andererseits einer Transportkomponente (z.B. ATM-Treiber) bereitgestellt werden.

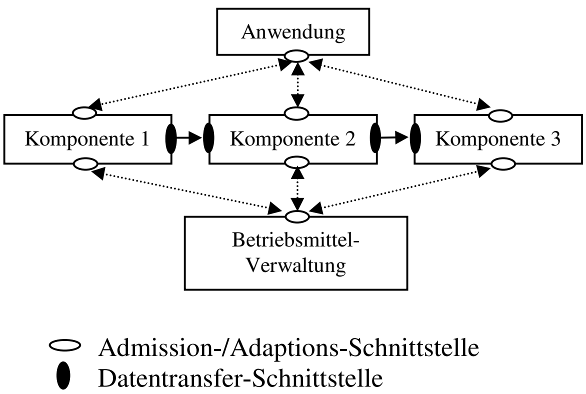

Bild 1 DROPS Anwendungsmodell

Die Übertragung der Daten erfolgt durch das zeitgesteuerte Einblenden von Speicherseiten (im vorliegenden Kontext als „Pufferelemente“ bezeichnet) in den Adreßraum der Empfängerkomponente als einer speziellen Form eines shared memory. Für das Einblenden der Pufferelemente wird ein Adreßbereich reserviert, der in der Regel weit kleiner ist als der Gesamtumfang des Datenstroms. Daher wird dieser Bereich zyklisch verwendet, wobei die in dem Puffer enthaltenen Daten eine gewisse Gültigkeitsdauer haben; die Komponenten prüfen vor dem Zugriff auf den Adreßbereich die Gültigkeit der Daten. Aufgabe des Systementwurfs ist es, durch geeignetes Scheduling für die rechtzeitige Bereitstellung der Daten zu sorgen sowie durch gute Pufferdimensionierung eine hohe Auslastung der Systemressourcen bzw. einen hohen Parallelitätsgrad von Applikationen zu ermöglichen.

2.2 Aufbau eines Datenstroms und Einplanung (Admission Control)

Der Aufbau des Datenstroms erfolgt in mehreren Schritten [2]. Zunächst wird durch eine spezielle Systemkomponente eine Beschreibung des Datenstroms erstellt, die alle erforderlichen Informationen enthält (vor allem die Werte der in Abschnitt 3 beschriebenen Parameter, ggf. aber auch Bandbreiten oder Video-Auflösung). Die Art und Weise der Erstellung dieser Beschreibung ist von der jeweiligen Anwendung abhängig, sie kann beispielsweise durch das Lesen einer Parameterdatei erfolgen. Im nächsten Schritt wird die Beschreibung als Parameter der Anforderung zum Öffnen des Datenstroms bei den einzelnen Komponenten verwendet. Dadurch wird eine lokale Admission Control in den Komponenten bewirkt, d.h., jede Komponente reserviert die zur Bedienung des Datenstroms erforderlichen Betriebsmittel; die Art und der Umfang der Betriebsmittel werden aus der Datenstrombeschreibung ermittelt. Lassen sich für alle Komponenten die benötigten Ressourcen reservieren, so werden im letzten Schritt die einzelnen Komponenten zu der Komponentenkette verknüpft, und die Bearbeitung des Datenstroms wird gestartet. Andernfalls kann entweder der Datenstrom mit neuen Parameterwerten ausgehandelt werden, oder es kann durch Adaption anderer Datenströme versucht werden, die fehlenden Ressourcen zu erhalten.

3 Schwankungsbeschränkte Ströme

In einer Reihe von Echtzeitanwendungen (und insbesondere im Multimedia-Bereich) hat die Last folgenden Charakter: An einer bestimmten Schnittstelle des Systems treten von einem Startzeitpunkt t_0 an Ereignisse nacheinander eigentlich in konstantem Abstand T ein, können aber in festen Grenzen

schwanken, nämlich um eine Zeit τ verfrüht oder um τ' verspätet sein; die Ereignisse müssen dabei einen Mindestabstand $D < T$ einhalten.
Derartige als „schwankungsbeschränkte periodische Ströme“ bezeichneten Ereignisfolgen werden in unterschiedlichen Projekten durch eine Vielzahl von – zum Teil voneinander abhängigen – Parametern beschrieben; dazu gehören Spitzenrate, durchschnittlicher Nachrichtenabstand, maximale Burstlänge und minimale Puffergröße zum Vermeiden von Datenverlust. Das folgende Modell [4] ermöglicht eine einheitliche Behandlung heterogener Komponenten eines Echtzeitsystems.

Definition. Gegeben seien

$$D, T, \tau, \tau' \in \mathbb{R} \quad \text{mit} \quad T > D > 0, \quad \tau, \tau' \geq 0,$$

sowie $i \in \mathbb{N}$. Ein (τ, τ')-schwankungsbeschränkter Strom mit konstanter Periode T und Mindestabstand D (kurz: schwankungsbeschränkter Strom) ist dann eine Folge $(E_i)_{i=0,1,...}$ von Ereignissen, die in den Zeitpunkten

$$a_i \in [iT - \tau, iT + \tau'] \subseteq \mathbb{R}$$

eintreten und deren Abstände der Bedingung

$$a_{i+1} - a_i \geq D \quad \forall i \in \mathbb{N}$$

genügen.

Der in [4] angegebene Kalkül gestattet es, wichtige Bewertungsgrößen zu berechnen (maximale Burstlänge, frühester und spätester Startzeitpunkt eines Bursts, Abstände zwischen Bursts und minimale Puffergröße zur verlustlosen Datenübertragung) und Transformationen zwischen diesen Parametern vorzunehmen (beispielsweise die Bestimmung von τ, τ' bei gegebener maximaler Burstlänge). Mit Blick auf Abschnitt 4.3 sei erwähnt, daß der Grenzfall $D = 0$ in das Modell eingeschlossen werden kann. Im Rahmen des DROPS-Projekts liefert das Modell einen einheitlichen Parametersatz zur Beschreibung von Daten- bzw. Ereignisströmen zwischen Systemkomponenten (wobei statt der zwei Parameter τ, τ' ein gemeinsamer Parameter verwendet wird und dafür der Startzeitpunkt t_0 eines Stromes hinzukommt).

4 Pufferdimensionierung in DROPS

4.1 Problemstellung

Zur Illustration des Gesagten werden im folgenden zwei Modelle beschrieben, um die Größe des für den Datentransfer zwischen zwei Komponenten (die den Erzeuger bzw. Verbraucher der Daten darstellen) erforderlichen Puffers zu berechnen. Das Poblem besteht darin, daß der Verarbeitungsprozeß VP nicht „verhungern“ darf und daß gleichzeitig der dazu erforderliche Puffer möglichst klein sein soll. Dies erfordert einen Vorlauf V des Erzeugerprozesses EP bzw. einen Anfangsfüllstand P_0 des Puffers und eine Vorperiode T_0 von EP. Der Vorlauf stellt den gegenüber VP vorgezogenen Startzeitpunkt von EP dar; die Vorperiode ist die Phasenverschiebung innerhalb einer Periode von EP, die daraus resultiert, daß V nicht notwendig ein Vielfaches der Periodenlänge von EP ist. Die Lösung erfolgt mit zwei Modellansätzen entsprechend der Detailliertheit der Strombeschreibung. In der Regel liegt die Beschreibung als schwankungsbeschränkter Strom gemäß Abschnitt 3 vor. Oftmals aber (z.B. beim Abspielen eines MPEG-Videos) enthält die in Abschnitt 2 erwähnte Objektbeschreibung eine wesentlich genauere Zeitspur-Information, d.h. zu welchem Zeitpunkt welche Bytes dem Verbraucher zur Verfügung stehen und damit sich im Puffer befinden müssen. Zusätzlich ist dabei der Fall von besonderem Interesse, daß einer der beiden Prozesse (hier der Erzeugerprozeß) schwankungsfrei arbeitet, wie es etwa bei der Bereitstellung von Daten durch den SCSI-Treiber (Plattenblöcke konstanter Größe) für das Dateisystem der Fall ist. Im Blick darauf und das in Abschnitt 2 beschriebene Modell wird von folgenden Annahmen ausgegangen (vgl. auch **Bild 2a, 2b**):

– Der Erzeugerprozeß EP füllt periodisch (Periodenlänge T_A) ab der Zeit $t_A = 0$ einen Puffer der Gesamtgröße $P_{\min}$ mit Datenmengen konstanter Größe Q (Quantum). Die Bereitstellung der Daten erfolgt mit Sicherheit jeweils zu Beginn einer Periode, die Füllzeit wird vernachlässigt („Pufferbereich wird der Anwendung übergeben“).
– Der Verbraucherprozeß VP entnimmt periodisch (Periodenlänge T_B) ab der Zeit $t_B = 0$ Datenpakete variabler Größe D_j, $j = 1,...,n$, $n \geq 1$ Datenpaket-Anzahl. Das Entnehmen (Leeren) dauere die Zeit $L < T_B$, am Ende jeder Periode ist der zugehörige Pufferbereich garantiert frei, die angeforderten Daten sind mit Sicherheit „verbraucht“ („Pufferbereich wird der Anwendung entzogen“).
– Die mittleren Raten r_A und r_B von EP und VP sollen gleich sein.
– Weiter sei

$$D = \sum_{j=1}^{n} D_j > 0$$

Gesamtanzahl der angeforderten Daten.
Die Daten werden mit natürlichen Zahlen $1,...,D$ identifiziert (Bytezähler, kontinuierliche äquidistante Speicheradressen o.dgl.).

Damit sind bei gegebenem T_B, Q, $(D_j)_{j=1,...,n}$ und n die Größen T_A, V, $P_{\min}$, P_0, T_0 zu bestimmen.

4.2 Zeitspur-Modell

Aus der Forderung gleicher durchschnittlicher Raten $r_A = r_B$ von EP und VP folgt sofort

$$T_A = \frac{nQ}{D} \cdot T_B = \frac{Q}{r_B}. \qquad (1)$$

Zur Berechnung des Vorlaufs V sei $m = \dfrac{D}{Q}$ und damit $\lceil m \rceil$ die Anzahl von Dateneinheiten der Größe Q, in denen D eingelesen wird; weiter sei

$$S_j := \sum_{k=1}^{j} D_k\,, \quad j = 1,\ldots,n.$$

Nun sei (s. Bild 2a,b)

$$B(t) = \begin{cases} 0 & \text{für} \quad t < 0 \\ S_j & \text{"} \quad t \in [(j-1)T_B,\, jT_B), \quad j = 1,\ldots,n \\ D & \text{"} \quad t \geq nT_B \end{cases}$$

die „kumulative Bedarfsfunktion“ (kumulative Entnahme der Daten) und entsprechend

$$A(t) = \begin{cases} 0 & \text{für} \quad t < 0 \\ iQ & \text{"} \quad t \in [(i-1)T_A,\, iT_A), \quad i = 1,\ldots,\lceil m-1 \rceil \\ D & \text{"} \quad t \geq \lceil m-1 \rceil \cdot T_A \end{cases}$$

die „vorläufige kumulative Angebotsfunktion“ (Füllfunktion); beide Funktionen sind linksstetige Treppenfunktionen.

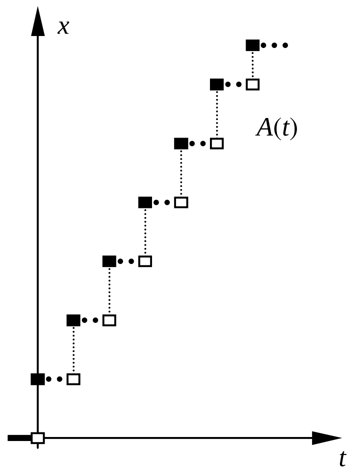

Bild 2a Erzeugerprozeß

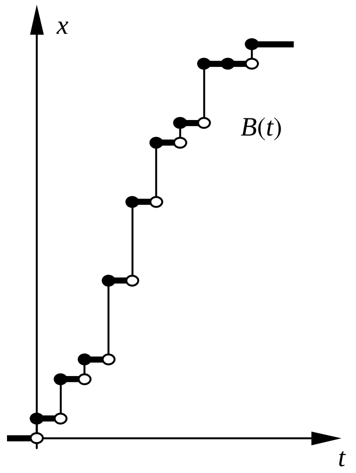

Bild 2b Verbraucherprozeß

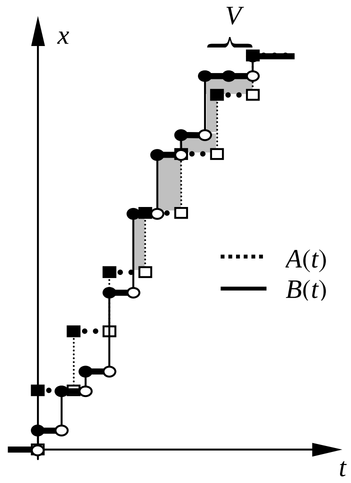

Bild 2c Vorlauf V

Die für eine (eineindeutige) Funktion F übliche Bezeichnung F^{-1} der Inversen versteht sich im folgenden als Bezeichnung der Pseudo-Inversen einer Treppenfunktion, d.h. konkret für die folgendermaßen auf $[0, D]$ definierten Funktionen:

$$A^{-1}(x) = \begin{cases} iT_A & \text{für} \quad x \in (iQ,\, (i+1)Q] \\ \lceil m-1 \rceil T_A & \text{"} \quad x \in (\lceil m-1 \rceil Q,\, D] \end{cases}$$

$$B^{-1}(x) = jT_B \quad \text{für} \quad x \in (S_j,\, S_{j+1}], \quad j = 0,\ldots,n-1,$$
$$\text{wobei} \quad S_0 := 0, \quad (x, x] := \varnothing.$$

A^{-1} und B^{-1} sind rechtsstetige Treppenfunktionen mit den Vielfachen von Q bzw. den Werten D_j als Sprungstellen und den Perioden T_A bzw. T_B als Sprunghöhen. $A^{-1}(x)$ drückt aus, wann das Datum x eingelesen ist (Pufferplatz belegt), entsprechend $B^{-1}(x)$, wann es bereitstehen muß.
Schließlich sei

$$\Delta := A^{-1} - B^{-1}.$$

Ist die Differenz $\Delta(x)$ positiv, so bedeutet dies, daß der Zeitpunkt des Einlesens (Füllens) von Datum x größer ist (das Einlesen also später erfolgt) als der des Entnehmens von x. Die Differenz dieser Ordinaten für den Wert x ist gleich dem Abstand der Strecken der Funktionen A und B in Höhe von x, die deren Sprungstellen senkrecht (also in x-Richtung) miteinander verbinden. In den Bereichen der t-Achse, in denen A rechts von B liegt, treffen die Daten zu spät ein, und die Verspätung ist dort am größten, wo dieser Abstand am größten ist (s. **Bild 2c**). Wird daher die Funktion A um diesen Abstand V nach links – in Richtung negativer t-Werte – verschoben, so treffen nunmehr *alle* Daten um die Zeit V früher und damit rechtzeitig ein; ist die Verschiebung geringer als V, so wird zumindest ein zu V gehöriges Datum x mit $\Delta(x) = V$ zu spät eingelesen. Damit ergibt sich zusammengefaßt als Mindestvorlauf für EP:

$$V = \max_{x \in [0,D]} \Delta(x) \tag{2}$$

und es ist

$$\hat{A}(t) := A(t+V), \quad t \in \mathbb{R},$$

die endgültige Angebotsfunktion.
Durch ähnliche Überlegungen ergibt sich für Anfangsfüllung P_0 und Vorperiode T_0

$$P_0 = \left(\left\lfloor \frac{V}{T_A} \right\rfloor + 1 \right) \cdot Q \tag{3}$$

$$T_0 = \left\lceil \frac{V}{T_A} \right\rceil \cdot T_A - V \tag{4}$$

und schließlich als kleinster mindestens erforderlicher Puffer $P_{\min}$

$$P_{\min} = \max_{t \in \mathbb{R}} \left(A(t+V) - B(t - T_B) \right). \tag{5}$$

Wie eingangs erwähnt, füllt EP „irgendwann" Pufferbereiche der Größe Q mit Daten und erklärt diese Bereiche nacheinander im konstanten Abstand T_A als gültig für VP. Nach einer bestimmten Zeit (Gültigkeitsdauer T_Q) werden diese Bereiche VP wieder entzogen, wenn die Daten nicht mehr benötigt werden. Außerdem benutzt EP in jedem Fall vollständige Pufferelemente der Größe Q, auch wenn $P_{\min}$ dies nicht erfordern würde; es sei n_Q deren kleinste Anzahl, die gewährleistet, daß $P_{\min}$ darin gespeichert werden kann. Dann gilt:

$$n_Q = \left\lfloor \frac{P_{\min} - 2}{Q} \right\rfloor + 2, \qquad T_Q = n_Q \cdot T_B. \tag{6}$$

4.3 Modellierung als schwankungsbeschränkter Srom

In diesem Fall sollen die wesentlichen Ergebnisse nur genannt werden, für Einzelheiten wird auf [5] verwiesen.
Der Verbraucherprozeß VP benötige in Intervallen der Länge T_B Daten in Form eines (τ, τ')-schwankungsbeschränkten Stroms mit der Rate R und Mindestabstand 0. Die Daten werden durch EP in Einheiten konstanter Größe Q und in konstantem Abstand $T_A = \dfrac{Q}{R}$ bereitgestellt. Der Verbrauch der von VP zu einem Zeitpunkt angeforderten Daten ist mit Sicherheit bis zum Beginn der nächsten Anforderung abgeschlossen. Dann gilt für Vorlauf V und Mindestpuffergröße $P_{\min}$:

$$V = \tau \tag{7}$$

$$P_{\min} \le \lceil R(\tau + \tau' + T_B) + Q \rceil. \tag{8}$$

5 Zusammenfassung und Ausblick

Ausgehend von einem Modell zur Beschreibung schwankungsbeschränkter Ströme wurde das Problem betrachtet, eine möglichst genaue Schranke für die Größe des mindestens erforderlichen Puffers zu ermitteln, der benötigt wird, um die verlustfreie Übertragung von Daten zwischen zwei Prozessen zu gewährleisten. Im vorliegenden Fall arbeitet der Erzeugerprozeß schwankungsfrei, die Loslösung von dieser Voraussetzung ist jedoch problemlos möglich. Die Ergebnisse schlagen sich im Systementwurf des DROPS-Projekts nieder und ermöglichen dort eine effiziente Reservierung der Ressource Speicher im Hinblick auf Pufferung. Werkzeuge zur Erfassung der erforderlichen Informationen über die Datenströme (Zeitspur, Schwankungsparameter) und zur Auswertung im beschriebenen Sinne stehen zur Verfügung. Gegenwärtig erfolgt die Integration der Methoen in DROPS. Zu den nächsten Aufgaben gehört Festlegung und Implementation der zugehörigen Schedulingverfahren.

Literatur

[1] Härtig, Hermann, et al.: DROPS – OS Support for Distributed Multimedia Applications. In: Proceedings of the Eigth ACM SIGOPS European Workshop, Sintra (Portugal), September 1998

[2] Härtig, H.; Reuther, L.; Wolter, J.; Borriss, M.; Paul, T.: Cooperating Resource Managers. In: Proceedings of Workshop on QoS Support for Real-Time Internet Applications. Vancouver, June 1999

[3] Liedtke, J.; Härtig, H.; Hohmuth, M.: OS-Controlled Cache Predictability for Real-Time Systems. In: Proceedings of the Third IEEE Realtime Technology and Applications Symposium, Montreal, June 1997

[4] Hamann, Claude-J.: On the Quantitative Specification of Jitter Constrained Periodic Streams. In: Proceedings of MASCOTS '97, Haifa January 1997

[5] Hamann, Claude-Joachim: Pufferberechnung für schwankungsbeschränkte Ströme. Technischer

Bericht TUD-FI99-06 - Juli-99, Technische
Universität Dresden, 1999