Multimedia-Präsentationen auf persönlichen digitalen Assistenten mit geringer Übertragungsrate

Wolfgang Leister, Håvard Hegna, Thor Kristoffersen,
Lars Aarhus, Anders Moen, Bjarte M. Østvold
*Norwegian Computing Center**

Zusammenfassung

Es wird eine einfache und robuste Client-Server-Architektur zur Präsentation von Multimedia-Inhalt auf mobilen Terminals vorgestellt. Dabei werden die Daten auf einem Server vorbereitet und in einem Streaming-Verfahren auf den Client übertragen. Die Multimedia-Präsentation entsteht durch das Abspielen von Multimedia-Befehlen auf einer abstrakten Maschine. Unter anderem wird der Ansatz des *dehnbaren Streaming* eingesetzt, um variierenden Empfangsverhältnissen gerecht zu werden. Dieses Konzept wurde für sehr geringe Datenrate entwickelt (z.B. über GSM) und wurde für eine Nachrichtenanwendung vorgeführt.

Anwendungen zur Präsentation von multimedialen Inhalten für persönliche digitale Assistenten (PDA) sind bisher nicht allgemein verfügbar. Die Übertragungskapazität ist für allgemeine Videoanwendungen nach dem Streaming-Prinzip bei der heute verfügbaren Datenrate bei der Übertragung mit dem GSM System zu gering. Bei der Einführung von paket-orientierten Netzwerken mit höherer Übertragungskapazität (z.B. UMTS) wird eine Änderung erwartet. Jedoch werden auch diese Netzwerke sehr unterschiedliche Übertragungskapazitäten aufweisen, die unter anderem von der Position und den Bewegungen des Benutzers abhängen.
Andere Eigenschaften, die solche Multimedia-Anwendungen erschweren, sind die zur Zeit noch relativ geringe Rechenkapazität im Verhältnis zu stationären Rechnern.
Die hier vorgestellte Architektur beruht auf dem Prinzip des *dehnbaren* Streaming von synchronisierten multimedialen Inhalten auf mobilen Terminals. Der Begriff *dehnbar* wird so realisiert, dass Pausen und Füllmaterial an geeigneten Stellen der Präsentation bei Bedarf eingefügt werden können, um eine flüssige Präsentation auch bei zu geringer Übertragungsrate oder wechselnden Übertragungsverhältnissen zu ermöglichen. Dabei werden auf Anwendungsebene keine zusätzlichen Rückmeldungen an den Server gegeben, was das Konzept auch für Anwendung geeignet macht, die auf dem Rundfunk-Prinzip (broadcast) beruhen.
Die Architektur beruht auf der Idee einer abstrakten Multimedia-Maschine (AMM). Dabei übernimmt der Server die Funktion eines Compilers, der eine Beschreibung

*Norsk Regnesentral, Postboks 114 Blindern, NO-0314 Oslo, Norway. email: wolfgang@nr.no

des Medieninhalts in der Beschreibungssprache SMIL in Befehle der Multimedia-Maschine übersetzt. Diese werden auf die Client-Maschine übertragen und schließlich ausgeführt, was in der Präsentation des Inhaltes resultiert. Die Architektur wird in Abb. 1 illustriert.
Wichtige Anforderungen an die Anwendung sind dabei geringe Antwortzeiten von Bestellung des Inhaltes bis zum Beginn der Präsentation, sowie flüssiges Abspielen, wobei die Unzulänglichkeiten der Übertragung so weit als möglich überdeckt werden sollen.
Die Multimedia-Architektur wird anhand einer Nachrichtenanwendung erläutert, bei der der Inhalt in Text, Bildern und Ton an den Benutzer auf einen PDA übermittelt werden. Eine animierte Nachrichtensprecherin mit lippensynchroner Mundbewegung und Mimik vermittelt den Eindruck einer 'echten' Nachrichtensendung wie im Fernsehen.
Der Benutzer ruft anhand einer Menuauswahl aktuelle Nachrichten ab. Die Nachrichten werden von einem Server mittels einer GSM-Datenverbindung übertragen. Jede Nachricht besteht aus kontinuierlichen (Audio, Animationen) und statischen (Bilder, Text) Datentypen, die auf dem PDA synchronisiert abgespielt werden.
Alle Daten werden komprimiert übertragen. Ebenso stellt die Verwendung einer Animation anstatt eines Videos eine Form der Komprimierung dar: Die Animationskommados kommen mit einer geringen Datenrate im Vergleich zur Videoübertragung aus.
Die Benutzeroberfläche ist einfach gestaltet und erlaubt das Auswählen von Nachrichten aus einen dargebotenen Listen, die verschiedenen Nachrichtenkanälen zugeordnet werden. Die Präsentation einer Nachricht kann bei Bedarf durch den Benutzer angehalten oder abgebrochen werden. Die verfügbaren Nachrichten werden auf einem Server zur Verfügung gestellt und dynamisch auf den neuesten Stand gebracht. Weitere Interaktionsmöglichkeiten bestehen in der Auswahl verschiedener Texte (z.B. Untertexte) und alternativer Bilder. Jeder Client hat in unserem Beispiel eine individuelle Verbindung zum Server.
Der Inhalt ist in einer Datenbank gespeichert, auf die der Server zugreifen kann. Der Inhalt ist auf vorgelesenen (Radio-)Nachrichten aufgebaut. Daraus werden die Lippenbewegungen für die Animation automatisch extrahiert, während passende Kopfbewegungen und Zwinkern automatisch dazugeneriert werden. Danach werden Texttafeln, Überschriften und Bilder der Präsentation hinzugefügt.
Nachrichten werden in der Synchronized Multimedia Integration Language (SMIL) [Wor, Hen01] repräsentiert. Dabei werden parallele, sequentielle und geschachtelte Teile eines Medienstroms definiert. Weiterhin können relative Zeitangaben und räumliche Angaben hinzugefügt werden. Die elementaren Bausteine verweisen auf Dateien, die die Medien enthalten.

1 Streaming

Die Technik des Streaming wird eingesetzt, damit ein Benutzer nicht warten muss, bis ein großes Multimediaobjekt vollständig geladen ist, bevor es abgespielt werden kann. Streaming ist ein wichtiges Konzept in dieser Arbeit. Jedoch wird dieser Begriff in der Literatur unterschiedlich definiert, weshalb er hier nochmals klargelegt wird.
Gegeben sei zeitkontinuierlicher Multimediainhalt, z.B. Video oder Audio, der am Ort S gespeichert ist. Sei weiterhin C ein Ort verschieden von S, der aber mit S kommunizieren kann, so dass der Multimediainhalt in C in einer Anwendung präsentiert werden kann. Streaming ist in diesem Zusammenhang eine Technik, die es erlaubt die Präsentation des Inhaltes zu beginnen bevor der gesamte Inhalt von S nach C übertragen worden ist [McC98].
Steinmetz und Nahrstedt [SN95] schlagen ein Referenzmodell zur Synchronisierung von Multimedia vor, das aus vier Ebenen besteht: (1) Die Medium-Ebene beschreibt die Abfolge logischer Dateneinheiten (z.B. Videobilder oder Tonwerte) eines Stroms. (2) Die Strom-Ebene beschreibt die Synchronisierung zwischen Strömen bzw. Gruppen von Strömen. (3) Die Objekt-Ebene beschreibt die Integration zwischen kontinuierlichen Strömen und diskreten Daten, die nicht in einem Strom definiert werden, wie Bilder und Text. (4) Die Spezifikationsebene beschreibt Anwendungen und Werkzeuge, die eine Erzeugung einer Synchronisationsspezifikation ermöglicht.
Im Vergleich zum Referenzmodell von Steinmetz und Nahrstedt enthält unser Ansatz eine zusätzliche Ebene, die Element-Ebene (Unit-level), die über der Objekt-Ebene liegt. Diese Ebene enthält Elemente für dehnbare Pausen mit einer Mindestlänge (diese kann auch 0 sein), die die nicht-dehnbaren Medien-Elemente verbinden.
Ein Multimedia-Übersetzer auf der Server-Seite produziert aus der SMIL-Beschreibung eine Serie von Befehlen, die Synchronisationsinformation für die drei darunterliegenden Ebenen enthalten. Die Synchronisation auf der Element-Ebene wird durch den Medienstrom implizit vorgegeben: Innerhalb eines Medienstroms darf keine Dehnung auftreten, sondern nur zwischen aufeinanderfolgenden Strömen. Durch diesen Mechanismus wird der Begriff der *Dehnbarkeit* in unserer Architektur definiert.

2 Andere Systeme und Ansätze

Im allgemeinen wird für streamed Multimedia in Netzwerkumgebungen mit geringer und variabler Datenrate eine Ebenentechnik zur Kodierung von kontinuierlichen und statischen Medientypen verwendet. Dieses so genannte *Layered Coding* wird für kontinuierlichen Medien sowohl beim RealSystem iQ streaming als auch bei der MPEG-4-kodierung verwendet. In vielen Fällen wird dabei auch die Datenrate des Medienstroms an die Gegebenheiten im Netzwerk angepasst, was erfordert, dass vom Netzwerk Parameter wie *delay* und *jitter* gemessen werden, um Über- bzw. Unterläufe zu vermeiden [SMZ99].

Turner und Ross [TR00] ermitteln die optimale Anzahl von Ebenen für *statische* Medientypen, um deren Präsentation in Netzwerken mit variabler Datenrate zu maximieren. Als Nebenbedingung gilt bei ihnen, dass der Bedarf bezüglich der Datenrate des Medientyps konstant, und die Zeitachse der Präsentation starr sind.
Georganas et al. [DG97, JG97] zeigen eine Anwendung für die Übermittlung von Nachrichten auf Abruf. Dieses System baut auf Rückmeldungen vom Netzwerk auf. Es arbeitet mit einer starren Präsentationszeitachse, und muss daher Bilder verzögern bzw. überspringen falls ein Unterlauf vom Netzwerk verursacht wird.
Für das Internet gibt es mehrere Architekturen für Medien-Streaming. Zwei wichtige kommerzielle Systeme stellen dabei das RealSystem iQ [Rea] von RealNetworks und Windows Media Technologies [Win] von Microsoft dar. Beide sind traditionelle Client-Server Architekturen mit Punkt-zu-Punkt Verbindungen, die proprietäre Formate und Protokolle verwenden, in diesem Falle RealMedia Format bzw. Advanced Streaming Format (ASF). Diese Protokolle erlauben untere Datenraten von typisch 28.8 kb/s oder 56.6 kb/s wie sie für Modemverbindungen bzw. ISDN verwendet werden.
Das RealSystem iQ ist von besonderem Interesse, da Inhalt formuliert im Standard SMIL durch Streamining vermittelt werden kann. Weiterhin wird dort das Real Time Streaming Protocol (RTSP) [SRL98] zur vom Client-getriebenen Steuerung des Servers auf Anwendungsebene eingesetzt. Obwohl das RealSystem iQ eine erweiterbare und flexible Architektur ist, die an verschiedene Systeme und Datenraten angepasst werden kann, ist sie zur Zeit (noch) nicht für mobile Plattformen verfügbar.
Ein Rahmen für das Kodieren, Senden, Abspielen und Synchronisieren von Multimedia-Daten über Netzwerke wird im Java Media Framework (JMF) [Jav99] vorgeschlagen. Dabei wird ein API bereitgestellt, das die Synchronisierung mehrere Abspieleinheiten für eine Multimedia-Präsentation erlaubt.
Eine weitere Architektur stellt die Familie des MPEG Standards dar. Da der MPEG-2 Transportstrom (MPEG-TS) [Int93] für das Aussenden von Daten nach dem Rundfunkprinzip ausgelegt ist, haben wir in unserer Arbeit einige Eigenschaften davon übernommen, da wir ebenfalls auf die Möglichkeit von Rückmeldungen über das Netzwerk weitgehend verzichten. MPEG-2 ist für Anwendungen des digitalen Fernsehens entwickelt, wobei die Datenraten um Größenordnungen höher sind. Multimediaströme dabei in einem Transport-Multiplex gemeinsam übertragen.
MPEG-4 [Int99] wird zur Aussendung von 'layered' Multimedia-Strömen verwendet. Der Fokus wird dabei auf niedrige Datenrate gelegt mit Bitraten ab 10 kb/s. Einige Eigenschaften machen MPEG-4 geeignet zur Aussendung von Multimedia-Daten für Mobile Terminals [PE98]. Jedoch versuchen sowohl JMF als auch die MPEG Architekturen viele Gebrauchsbereiche zu decken, so dass nicht alle Anforderungen für Client-Maschinen mit sehr beschränkten Ressourcen optimal gedeckt werden können.

3 Entwurf des Systems

In diesem Abschnitt erläutern wir die Problemstellungen beim Aufbau der Anwendung und gehen insbesondere auf einige Designentscheidungen ein. Die Erstellung der Nachrichtenanwendung und die physikalischen Beschränkungen, die durch die Hardware eines mobilen Terminals auferlegt werden, sowie die Funkverbindung zum Terminal stellen besondere Anforderungen an das Design [Sat96].

1. Die Präsentation des Inhaltes sollte so früh wie möglich nach dem Absenden der Anforderung gestartet werden. Weiterhin sollte die Präsentation nicht ins Stocken kommen. Im Falle der Nachrichtenanwendung sollte sich die Anwendung an Fernsehnachrichten anlehnen.
2. Die Ressourcen auf der ausgewählten Hardware sind begrenzt, besonders bezüglich Geschwindigkeit und Speicherplatz. Daher sollte die Präsentation möglichst einfach gestaltet sein.
3. Das Funknetzwerk hat eine begrenzte Datenrate für die Datenübertragung. Weiterhin kann die Qualität der Verbindung unvorhergesehen schwanken. Die Wahrscheinlichkeit des Verlustes von Datenpaketen, und die Bitfehlerrate sind höher als beim stationären Internet, vor allem hervorgerufen durch Interferenzen (physikalische Effekte durch Regen, Bewegung und Handoffs).

Bezüglich *Anforderung 1* gibt es generell zwei Lösungsstrategien, die wir beide verwenden: Die *präventive*, d.h. die Strategie vermeidet die Probleme, und die *korrektive*, d.h. die entstehenden Probleme werden beim Auftreten gelöst [JG97].
In Bezug auf das Referenzmodell zur Synchronisation wird multimedialer Inhalt auf dem Stream- und Objekt-Layer übertragen. Dadurch kann der Client die Multimediapräsentation starten auch wenn noch nicht der gesamte, mit Timestamps versehene Inhalt, übertragen worden ist. Dieses Schema erfordert jedoch, dass der Client ermitteln kann, wann die Präsentation gestartet werden kann ohne dass die Gefahr eines Unterlaufs besteht. Wir benutzen einen solchen Mechanismus, bei dem auf dem Server Berechnungen vorgenommen werden so dass bei bekannter Datenrate des Kanals die Übertragung durchgeführt werden kann.
In unserer Anwendung machen die Audiodaten die größte Datenmenge aus. Die Audiodaten werden in Segmente unterteilt an den Stellen, an denen natürliche Pausen im Redefluss der Nachrichtensprecherin auftauchen. Das Ergebnis ist eine Folge von nicht-dehnbaren Segmenten, die durch dehnbare Pausen unterbrochen werden. Dadurch kann der Client die korrektive Strategie verfolgen, indem diese Pausen bei Bedarf verlängert werden können, z.B. wenn noch nicht genügend Daten zum weiteren Abspielen bereitgestellt wurden.
Um Anforderung 2 zu befriedigen werden auf dem Server Präsentationen in SMIL in kleinere Einheiten mit Timestamps umgeformt, bevor diese zum Client gesendet werden. Diese Einheiten sind in einer 'maschinennäheren' Präsentationsstufe. Daher kann der Client einfacheren Regeln bei der Analyse und Präsentation des Inhaltes folgen, was auch einfachere Datenstrukturen zulässt als dies mit SMIL möglich wäre.

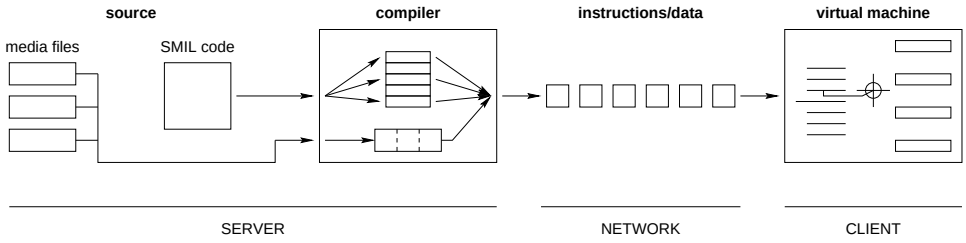

Abbildung 1: Systemarchitektur des Gesamtsystems.

Bezüglich *Anforderung 3* wird für den Datentransport das Transmission Control Protocol (TCP) mit einer Verbindung zum Client eingesetzt, wobei verschiedene Medientypen komprimiert und im Multiplex über diese eine Verbindung übertragen werden. Diese Wahl erfolgte aus pragmatischen Gründen um schnell einen Prototyp implementieren zu können. Die Architektur ist jedoch auch für Übertragungen nach dem Rundfunkprinzip (broadcast) und UDP geeignet.

3.1 Multimedia Compiler und Virtuelle Maschine

In Abb. 1 werden die Hauptkomponenten unseres Ansatzes auf zwei Ebenen illustriert:

- Der Server und der Multimediainhalt, das Netzwerk und den Client, auf dem der Inhalt abgespielt wird.
- Ein Compiler und seine Quelldateien, den Befehlscode und die Daten, sowie eine virtuelle Maschine, die diese Befehle abarbeitet.

Multimedialer Inhalt wird in Form von Inhaltsbefehlen und Dateifragmenten an den Client übertragen. Die Dateifragmente enthalten Bild- und Audioinhalte, die im Client wieder entkomprimiert und zu vollständigen Dateien zusammengesetzt werden. Die Inhaltsbefehle enthalten Anweisungen zum Anzeigen des Inhalts mit Verweisen auf die Inhaltsdaten.
Der Server überträgt eine Sequenz von Multimedia-Befehlen an den Client. Diese Befehle entsprechen einer übersetzten Version der SMIL Quelle und den Dateien mit dem Multimedia-Inhalt. Die übersetzte Version ist zum direkten Abspielen auf der virtuellen Maschine geeignet. Die Befehle enthalten Informationen darüber welcher Inhalt wie, wohin und wann der Inhalt zu rendern ist. Außerdem werden auf dem Server die Befehle in die Übertragungsreihenfolge gebracht, um minimale Antwortzeiten zu erreichen.

3.2 Funktionsweise der Abstrakten Multimedia-Maschine

Die Befehle enthalten Zeit-, Ort- und Opcode-Informationen, sowie Parameter, die für die verschiedenen Opcodes unterschiedlich sind. Die Befehle sind in folgender

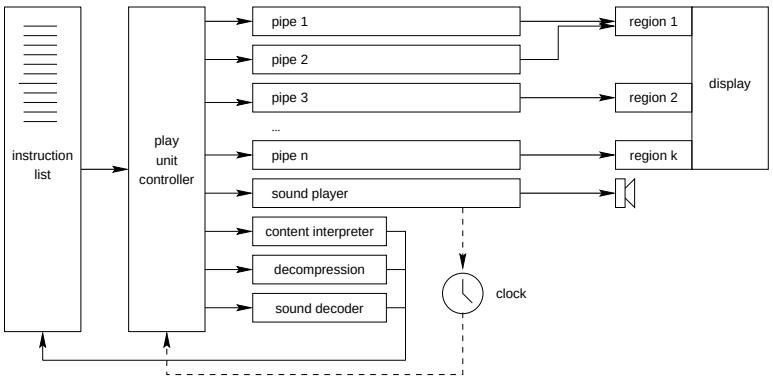

Abbildung 2: Abstrakte Multimedia-Maschine.

allgemeinen Form abgelegt:

seqno:timestamp	region, level	opcode	params

Der Inhalt wird in *Sequenzen* organisiert, die jeweils zum Zeitpunkt 0 starten und eine Zeitachse haben, die zeitlich nicht begrenzt ist. Für die hier besprochene Nachrichten-Anwendung wird eine zusammenhängende Meldung normalerweise in einer Sequenz abgelegt. Alle Inhaltsbefehle enthalten einen Zeitstempel, der relativ zum Anfang einer Sequenz angegeben wird, sowie die Sequenznummer.
Jeder Befehl enthält Parameter mit Verweisen auf den Inhalt, z.B. Bilder oder Ton. Der Bildschirm des Terminals ist in nicht-überlappende Regionen aufgeteilt. Die Regionen können in mehreren Ebenen adressiert werden, um alternativen Inhalt zu ermöglichen (z.B. für mehrsprachige Untertitel).
Die AMM arbeitet diese Befehle gemäß gewisser Regeln und in Bezug auf eine innere Uhr ab, wobei der Multimedia-Inhalt auf dem Bildschirm des PDA angezeigt, bzw. der Ton abgespielt wird. Der interne Aufbau der AMM wird in Abb. 2 dargestellt. Die abstrakte Maschine übersetzt die Inhaltsbefehle in Mikrobefehle und steuert das Bereitstellen und Abarbeiten der Mikrobefehle.
Das Befehlsformat für Inhalts- und Mikrobefehle ist identisch. Die Befehle des Client sind in mehrere Typen unterteilt, die in Tabelle 1 gezeigt werden. Bei Mikrobefehlen sind Opcode und Parameter so aufgebaut, dass ein weiteres Parsen des Inhaltes nicht mehr notwendig ist.
Nach der Ankunft auf dem Client werden die Befehle in Mikro-Befehle umgewandelt und in den Sequenzspeicher nach Zeitstempel sortiert abgelegt.
Der so genannte *Play Unit Controller* (PUC) ist verantwortlich für das Einsortieren und Abarbeiten der Befehle und Daten. Außerdem sorgt er für die korrekte Adressierung der Arbeitseinheiten, die *Pipes* genannt werden. Für jede Region, Opcode und Ebene gibt es maximal eine Pipe zum Rendern des Inhaltes. Durch sie werden Rasterbilder und Texte dargestellt, sowie der Ton abgespielt und Konvertierungen vorgenommen. Animationen werden durch so genannte *periodische Pipes* implemen-

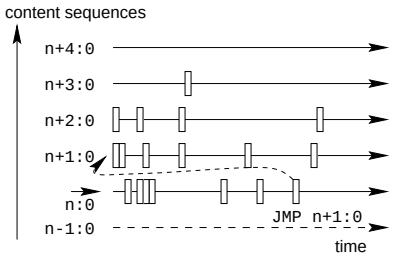

Abbildung 3: Befehle im Sequenzspeicher mit einem Sprungbefehl.

Befehlsart	*Beispiele*	*Semantik*
`atomic`	`MSG, TXT`	Inhaltsbefehle für Meldungen und formatierten Text.
`state`	`ACT, SUS`	Befehle zur Zustandssteuerung.
`control`	`JMP, JIN`	Sprungbefehle, Jingle-Unterprogramme.
`resource`	`EXC, RCV`	Befehle zur Ressourcenhandtierung.
`render`	`RND, VIS`	Rendern von Medienobjekten, z.B. Bilder (`RND RAS`), Audio (`RND AUD`) und Animationsbefehle.

Tabelle 1: Beispiele für Befehle der AMM.

tiert. Diese werden in regelmäßigen Abständen vom Betriebssystem aufgerufen und berechnen jedes Mal einen neuen Schnappschuss, der sofort auf dem Bildschirm dargestellt wird. Die periodischen Pipes nehmen Befehle zu Zustandsänderungen entgegen.

Die AMM arbeitet alle Befehle in einer Inhaltssequenz in ihrer zeitlich korrekten Reihenfolge (Sequenznummer, Zeitstempel) ab. Ein `JMP`-Befehl lässt Sprünge beim Abspielen zu. Dies ist in Abb. 3 illustriert, wobei jede Zeile eine Inhaltssequenz darstellt. Die Inhaltsbefehle sind als weiße Rechtecke dargestellt. Im Beispiel wird in Sequenz n ein Sprungbefehl zu Sequenz $n + 1$ durchgeführt.

Alle eingehenden Befehle werden im Befehlsspeicher in nach Zeitstempel sortierter Reihenfolge gespeichert. Die Befehle werden in Sequenzen und Abspieleinheiten (Play Units) eingeteilt. Play Units können abgespielt werden, ohne dass nach ihrem Start auf weitere Ressourcen (z.B. noch fehlende Bilder) gewartet werden muss.

Die Befehle werden dann abgearbeitet, wenn die Zeit des Zeitstempels erreicht wird. Allerdings werden Befehle mit Zeitstempel 0 sofort ausgeführt, wenn sie den Play Unit Controller erreichen. Dadurch kann eine Unterbrechung von außen hervorgerufen werden. Dies wird unter anderem ausgenutzt, um Sprungbefehle einzufügen, wenn mit dem Abspielen von neuen Inhalten begonnen werden soll.

Die AMM arbeitet in Zyklen, die durch einen Timer gesteuert werden. In jedem Zyklus wird untersucht ob der erste Befehl der Befehlsliste ausgeführt werden soll. Die Befehle werden dadurch ausgeführt, indem sie an die entsprechende Pipe geschickt werden, die durch Opcode, Region und Ebene adressiert sind. Nachdem ein Befehl sich in einer Pipe befindet, wird der weitere Prozess dieser Pipe überlassen.

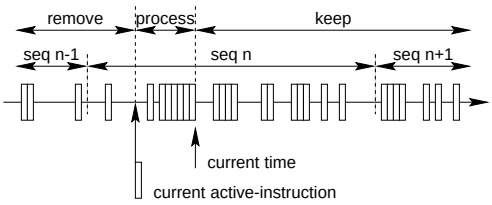

Abbildung 4: Abarbeiten der Befehle im *aktiv*-Zustand.

Die AMM befindet sich immer in einem von zwei Zuständen: *aktiv* (active) und *warten* (suspended). Diese Zustände sind identisch mit Befehlen gleichen Namens, die einen Übergang in genau diesen Zustand veranlassen können. Der momentane Zustand ist in der AMM in einem Register (in Form des Befehls) gespeichert bis er durch einen neuen Befehl in einen neuen Zustand gebracht wird.
Der Zustand *aktiv* zeigt an, dass die AMM Inhalt abspielt. Ein *aktiv*-Befehl gibt den Start einer neuen Abspieleinheit (Play Unit) an, was auch angibt, wann die Abspieleinheit gestartet wird. Die Play Units werden intern im Client gemäß einer Heuristik ausgeführt[1]
Alle Inhaltsbefehle mit Zeitstempel vor dem aktuellen *aktiv*-Befehl werden gelöscht, während Inhalt zwischen dem Start des *aktiv*-Befehls und der aktuellen Zeit sofort in Sortierreihenfolge abgearbeitet werden. Dieser Mechanismus zur Abarbeitung der Befehle wird in Abb. 4 illustriert. Dadurch wird sichergestellt, dass auch verspätete Befehle noch abgespielt werden, wenn das noch Sinn macht.
Im Status *warten* wartet die AMM auf eine Ressource, z.B. die Ankunft von Medieninhalt. Dann wird die Zeitachse nicht mehr fortgeschaltet. Alle Befehle vor dem laufenden *warten*-Befehl werden von der Befehlsliste gelöscht, während alle anderen Befehle nicht angerührt werden, d.h. es werden keine Befehle an die Pipes geschickt. Die Pipes werden intern jedoch nicht beeinflusst, was ermöglicht, dass z.B. Animationen weiterhin abgespielt werden. Ein plötzliches Einfrieren der Bewegung ist unerwünscht.
Wenn eine Ressource benötigt wird, die noch nicht vorhanden ist, wird ein *warten*-Befehl erzeugt, während die Ankunft einer Ressource einen entsprechenden *aktiv*-Befehl erzeugt. Hierzu werden die Ressourcen-Befehle verwendet. Intern werden diese Befehle, die zur gleichen Ressource gehören, bzw. denselben Zeitstempel aufweisen, gegeneinander aufgerechnet. Ein *aktiv*-Befehl im Befehlsspeicher löscht zugehörige *warten*-Befehle.
Der *warten*-Befehl enthält optional einen Timeout-Wert und eine Sprungadresse in eine Sequenz. Befindet sich die AMM länger als dieser Wert im Wartezustand, so wird ein Sprung zu dieser Adresse ausgelöst. Während dieses Sprungs wird vordefinierter Inhalt in diese Sequenz generiert, sowie ein Rücksprung zur aktuellen Adresse.

[1]Für unsere Anwendung erachteten wir es als sinnvoll dass der Start von Audio-Einheiten den Start einer neuen Abspieleinheit anzeigt.

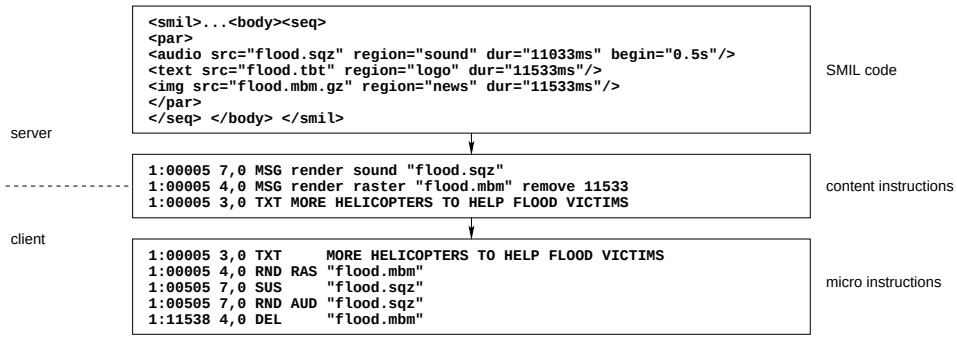

Abbildung 5: Übersetzung des Multimedia-Codes.

4 Code-Übersetzung für Multimedia

Im Gesamtsystem werden mehrere Übersetzungsvorgänge vorgenommen, um jeweils die optimale Repräsentation zu erhalten: Im Server wird SMIL verwendet, um Mediendateien zusammenzubinden, während beim Transport Inhaltsbefehle und Dateifragmente sich besser eignen. Nach Ankunft auf dem Client werden die Inhaltsbefehle in Mikrobefehle umgewandelt, da diese direkt an die Pipes geschickt werden können. Zwischen diesen Repräsentationen findet ein Übersetzungsprozess statt, der in Abb. 5 an einem Beispiel gezeigt wird. Die AMM hat folgende Mechanismen zur Befehlserzeugung:

- Die Ankunft von Mediendateien und Inhaltsbefehlen (speziell Befehle zum Anzeigen von Inhalt) stoßen die Erzeugung von `Status`-Befehlen an.
- Die Nutzinformation der Inhaltsbefehle wird geparst und in Mikrobefehle übersetzt.
- Der Client kann vorgefertigten Inhalt (z.B. Jingles) abspielen. Dieser Inhalt wird bei Bedarf in den Befehlsspeicher kopiert.
- Animationsbefehle für den Kopf der Nachrichtensprecherin werden in einer komprimierten Datei in Dateifragmenten vom Server übertragen. Aus diesen wird nach deren Ankunft auf dem Client eine große Anzahl von Mikro-Befehlen erzeugt. Durch diese Vorgehensweise wird Datenrate eingespart.

5 Diskussion

Unser Ansatz stellt einen SMIL-Player mit einer verteilten Client-Server-Architektur dar. Da der Inhalt auf einer separaten Servermaschine vorbereitet wird,

ist die Software auf dem Client geeignet zur Implementierung auf PDAs und für geringe Datenraten bei der Übertragung.
Die Architektur eignet sich für Multimedia-Anwendungen, die keine absoluten Real-Time-Anforderungen haben, d.h. Anwendungen bei denen in manchen Fällen auf die Ankunft von Medieninhalt innerhalb vernünftiger Zeiten gewartet werden kann. Außer den Wartezeiten besitzt unser Ansatz keinen Mechanismus zur Verringerung der Qualität falls die Wartezeiten zu groß werden.
In unserer Architektur kann der Client die Präsentationszeit strecken. Dadurch können sehr unterschiedliche Netzwerk-Eigenschaften kompensiert werden. Als Folge daraus ist die Dauer einer Multimedia-Präsentation nicht mehr allein durch den Inhalt vorgegeben. Zwischen Client und Server ist kein spezieller Feedback-Mechanismus notwendig. Wegen der relativ hohen Round-Trip-Zeiten wäre ein solcher nicht unbedingt sinnvoll. Daher werden alle Zeitstempel im Voraus berechnet, und der Client arbeitet nach einem eigenen Zeitbegriff.
Während der Pausen in denen der Client auf Inhalt wartet kann spezieller Inhalt abgespielt werden, um ein plötzliches Einfrieren der Bewegung zu vermeiden. Zum Beispiel kann die Animation der Nachrichtensprecherin bei Bedarf plausible Kopfbewegungen durchführen, ein Jingle kann abgespielt werden, oder ein Räuspern kann eingelegt werden.
Im Vergleich zu anderen besprochenen Architekturen ist unser Ansatz dem von Turner und Ross [TR00] am nächsten. Beide Ansätze wurden für Anwendungen mit niedriger Datenrate entwickelt. Einer der größten Unterschiede zwischen beiden Systemen ist, dass in deren Ansatz kein dehnbares Streaming eingesetzt wird.
Die Nachrichtenanwendung von Georganas et al. [DG97, JG97] arbeitet mit MPEG-Videoobjekten zusätzlich zu anderen Medien. Jedoch ist dieses System nicht für Netzwerke mit geringer Übertragungsrate ausgelegt. Deren Architektur setzt einen Rückkoppelungsmechanismus für die Streaming-Steuerung voraus. Die meisten anderen Architekturen sind nur für Festnetze bzw. Funkübertragung im wireless LAN und nicht für Telekommunikationsnetzwerke mit geringer Übertragungsrate gebaut. Einer der großen Vorteile unserer Architektur ist dass nur ein kleiner “Footprint” auf dem mobilen Terminal benötigt wird. Dagegen sind Architekturen wie MPEG bzw. JMF nicht genügend für solche Plattformen ausgelegt bezüglich Speicherplatzbedarf und Rechenkapazität, insbesondere Fließpunktarithmetik.
Neben dem Client-Server-Betrieb kann unser Ansatz auch für Rundfunk-Betrieb eingesetzt werden. Dabei werden die Dateien in einem Karussell bereitgestellt, wie z.B. in MPEG-2 DSM-CC [Int93]. Allerdings müssen für diesen Fall einige Heuristiken bei der Erzeugung von Synchronisationsbefehlen geändert werden. Außerdem muss ein Filter eingebaut werden, das die im Augenblick nicht benötigten, aber vom Karussell gelieferten Datenpakete wegwirft. Des weiteren muss eine Umnummerierung der Sequenznummern erfolgen. Sollte ein Datenverlust im Karussell-Betrieb vorkommen, muss eine weitere Runde gewartet werden. Alternativ kann bei mehreren erfolglosen Runden der Inhalt trotz der fehlenden Befehle durchgeführt werden, dann aber mit Abstrichen bei der Qualität. Die meisten dieser Änderungen betreffen lediglich die Anwendungsebene.

6 Der Prototyp

Unser Ansatz wurde auf der Ericsson *Communicator* Plattform implementiert[2]. Dieses Gerät bietet GSM-Kommunikation bei 9.6 kb/s, besitzt einen 190 MHz StrongARM Prozessor mit 16 MB RAM, und wird unter dem Betriebssystem EPOC betrieben, was speziell für Anwendungen mit geringem Speicherplatzbedarf geeignet ist.
Der Server basiert die Planung der Befehls- und Datenübertragung auf einem optimistischen Konzept, das eine konstante Datenrate von 7 kb/s voraussetzt. Mit dieser Datenrate kann der Client gesteuert werden. Gleichzeitig liegt dieser Wert genügend unter der maximalen Datenrate, so dass kleinere Einbrüche in der Kommunikation nicht bemerkt werden. Messungen bestätigten einen Wert von 8.2–8.6 kb/s als realistisch. Das Audio-Streaming wurde so geplant, dass half-rate GSM audio coding bei 5.6 kb/s übertragen wurde, was 2.6–3.0 kb/s für die anderen Daten und Befehle übrig ließ. Damit war ein flüssiges Abspielen des Inhalts nach einer angemessenen Wartezeit möglich.

Literatur

[DG97] M. Daami and N. Georganas. Client based synchronization control of coded data streams. In *Proc. Intl. Conf. on Multimedia Computing and Systems, (ICMCS'97)*, pages 387–394. IEEE, 1997.

[Hen01] Peter A. Henning. *Taschenbuch Multimedia*. Fachbuchverlag Leipzig, 2. auflage edition, 2001.

[Int93] International Organization for Standardization. *MPEG-2 – Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s*, 1993. ISO/IEC 11172-1:5.

[Int99] International Organization for Standardization. *MPEG-4 – Coding of audio-visual objects*, 1999. ISO/IEC 14496-1:6.

[Jav99] Java Media Framework API Guide, 1999. Web document, `http://java.sun.com/jmf/`.

[JG97] J. Jarmasz and N. Georganas. Designing a distributed multimedia synchronization scheduler. In *Proc. Intl. Conf. on Multimedia Computing and Systems, (ICMCS'97)*, pages 451–457. IEEE, 1997.

[McC98] Mark McCarthy. What is streaming? Web document, `http://www.unc.edu/courses/ssp/streaming/streaming.html`, Center for Instructional Technology, University of North Carolina, 1998.

[2] Siehe `http://www.ericsson.com/cebit2000/pressroom/`

[PE98] A. Puri and A. Eleftheradis. MPEG-4: An object-based multimedia coding standard supporting mobile applications. *Mobile Networks and Applications*, 3:5–32, 1998.

[Rea] RealSystem iQ. Web document, `http://www.realnetworks.com/`.

[Sat96] M. Satyanarayanan. Fundamental challenges in mobile computing. In *Proceedings of the fifteenth annual ACM symposium on Principles of distributed computing*, pages 1–7. ACM, May 1996.

[SMZ99] Y. Song, M. Mielke, and A. Zhang. Netmedia: Synchronized streaming of multimedia presentations in distributed environments. In *Proceedings of IEEE Multimedia*, 1999.

[SN95] Ralf Steinmetz and Klara Nahrstedt. *Multimedia: Computing Communications and Applications*. Prentice Hall, 1995.

[SRL98] H. Schulzrinne, A. Rao, and R. Lanphier. *Real Time Streaming Protocol (RTSP)*. IETF, 1998. RFC 2326.

[TR00] D. A. Turner and K. W. Ross. Optimal streaming of synchronized multimedia presentations with layered objects. In *Proceedings of IEEE International Conference on Multimedia and Expo*, 2000.

[Win] Windows Media Technologies. Web document, `http://www.microsoft.com/windows/windowsmedia/en/default.asp`.

[Wor] World Wide Web Consortium. *Synchronized Multimedia Integration Language (SMIL) 1.0 Specification*. REC-smil-19980615.