Hochleistungsfähige Implementierung von Protokollen mit zellenbasierten Fehlerkontrollmechanismen für ATM-Netze

Georg Carle, Institut für Telematik, Universität Karlsruhe
Jochen Schiller, Institutionen för Datorteknik, Uppsala Universitet, Schweden
E-Mail: [g.carle, j.schiller]@ieee.org

Kurzfassung

Der Asynchrone Transfermodus (ATM) beginnt sich mehr und mehr als wichtige Netzwerktechnologie sowohl in privaten als auch öffentlichen Netzen durchzusetzen. Es besteht aber immer noch ein großer Mangel an Erfahrung mit Implementierungen leistungsfähiger, an das ATM-Umfeld angepaßter Protokolle auf unterschiedlichen Implementierungsplattformen. Wichtig ist hierbei, daß der Nutzen neuartiger Protokolle durch reale Implementierungen oder praxisnahe Simulationen nachgewiesen werden kann. Der vorliegende Artikel beschreibt anhand eines Protokolls mit zellenbasierten Vorwärtsfehlerkorrekturmechanismen [CaEG95a, CaEG95b] in der ATM-Adaptionsschicht sowie eines Protokolls mit ATM-spezifischen Übertragungswiederholungsmechanismen [CaZi95] unser Vorgehen bei der Gewinnung von Leistungsaussagen. Zur Untersuchung der Protokollimplementierungen werden preiswerte, PC-basierte Systeme, die Emulation von ATM-Netzwerken auf Arbeitsplatzrechnern und schließlich die Implementierung auf hochleistungsfähiger Hardware aufgeführt. Als ein Resultat der Arbeiten kann der Nutzen des gewählten Protokolls und der Implementierungen anhand konkreter Leistungskenndaten aufgezeigt werden.

1 Einleitung

Die Einführung von ATM-Netzen stellt in mehrerer Hinsicht ein Paradigmenwechsel dar. So wird mit ATM in Zukunft ein System zur Verfügung stehen, das mit einem einheitlichen Übertragungsverfahren, dem Asynchronen Transfermodus, eine Vielfalt unterschiedlicher Endsysteme miteinander verbinden wird. Zusätzlich bietet ATM die Möglichkeit, vielerlei unterschiedlicher Dienste, wie Audio, Video und Massendatentransfer mit garantierter Dienstgüte unter Zuhilfenahme von sogenannten Adaptionsschichten anzubieten. Für herkömmliche Netze entwickelte Kommunikationsprotokolle höherer Schichten sind nun aufgrund dieser Veränderungen oft nicht mehr adäquat für den Einsatz oberhalb der ATM-Schicht. Zusätzlich sind neue Dienste wie die Gruppenkommunikation gefordert, die von gängigen Protokollen nicht oder nur unzureichend unterstützt werden. Aus diesen Gründen ist eine Neuentwicklung angepaßter Protokolle unabdingbar für die vollständige Ausnutzung der Leistungsfähigkeit von ATM.
Ein wichtiger Schritt im Rahmen der Protokollentwicklung ist der Praxistest. Ein Protokoll sollte nicht nur in der Theorie fehlerfrei funktionieren, sondern vor allem im realen Einsatz seine Leistungsfähigkeit und Funktionalität unter Beweis stellen. Eine weitere wesentliche Eigenschaft eines Protokolls ist die Möglichkeit der einfachen Integration in verschiedene Implementierungsarchitekturen. Beispiele für solche Architekturen sind preiswerte Standard-PCs als Endgeräte, leistungsfähige Arbeitsplatzrechner für den technisch-wissenschaftlichen Einsatzbereich sowie dedizierte, hochleistungsfähige Zwischensysteme.

Im folgenden Abschnitt erfolgt eine Einführung in das als Protokollbeispiel gewählte Verfahren zur Vorwärtsfehlerkorrektur. Im 3. Abschnitt wird dann die Integration dieses Protokolls in die ATM-Adaptionsschicht aufgezeigt. Anschließend wird im 4. Abschnitt anhand einer Protokolltestumgebung auf Arbeitsplatzrechnern aufgezeigt, wie sich das Verhalten des Protokolls unter verschiedenen Umgebungsbedingungen wirklichkeitsgetreu überprüfen läßt. Als leistungsfähigste Implementierung des Protokolls wird im 5. Abschnitt eine dedizierte Hardware-Einheit zur Unterstützung der Protokollabarbeitung vorgestellt. Anschließend erfolgt eine Übersicht über die im Rahmen dieser Arbeiten gewonnenen Ergebnisse.

2 Fehlerkontrollmechanismen in ATM-Netzwerken

ATM-Netze erlauben es, bei Diensten mit variabler Bitrate die Bandbreite dynamisch zwischen mehreren Verbindungen aufzuteilen, wodurch sich ein statistischer Multiplexgewinn erzielen läßt. Bei der Überlagerung stoßartiger Datenströme kann es dabei zum Verlust von Zellen durch Pufferüberläufe kommen. In Breitbandnetzen mit optischer Übertragung, die sehr kleine Bitfehlerwahrscheinlichkeiten besitzen, sind Zellverluste durch Pufferüberläufe weitaus häufiger als Zellverluste durch Übertragungsfehler. Pufferüberläufe sind keine unabhängigen Ereignisse, sondern zeigen ein stark korreliertes Verhalten [OhKi91]. Die Wahrscheinlichkeit für Zellverluste kann für verschiedene Dienste über einen großen Bereich variieren [StGr94]. Für Quellen mit stark stoßartigem Sendeverhalten kann eine akzeptable Netzauslastung häufig nur erreicht werden, wenn höhere Zellverlustwahrscheinlichkeit in Kauf genommen werden [WoFD93]. Für den Sender einer Multicastverbindung steigt mit wachsender Anzahl von Empfängern auch die Wahrscheinlichkeit für das Auftreten von Fehlern.
Zellverluste können entweder durch Übertragungswiederholungen (Automatic Repeat ReQuest, ARQ) oder durch Vorwärtsfehlerkorrektur (Forward Error Correction, FEC) behoben werden. ARQ-Verfahren besitzen schlechte Skalierungseigenschaften für große Pfadkapazitäten sowie für Gruppenkommunikation mit vielen Empfängern. Die Leistungsfähigkeit von FEC-Verfahren ist unabhängig von Entfernung und Pfadkapazität. FEC-Verfahren besitzen außerdem für Multicastverbindungen sehr gute Skalierungseigenschaften, weil ihre Leistungsfähigkeit unabhängig von der Gruppengröße ist. Als Nachteil ist der permanent erforderliche Mehraufwand für die Redundanzdaten zu sehen.

2.1 Motivation für die Entwicklung leistungsfähiger Adaptionsschichtprotokolle

Die Schwächen der für heterogene Netze entwickelten Kommunikationsprotokolle beim Einsatz in einem homogenen ATM-Netz sind darauf zurückzuführen, daß ATM-Netze gegenüber den bisherigen Netzen in mehrerer Hinsicht einen Paradigmenwechsel darstellen. Die für heterogene Netze entwickelten Kommunikationsprotokolle für zuverlässige Gruppenkommunikation haben in der neuen Umgebung funktionale und leistungsbezogene Defizite.
Durch die hohe vermittelbare Bandbreite in ATM-Netzen wurde die Protokollverarbeitung, die zur Sicherstellung eines zuverlässigen Ende-zu-Ende-Dienstes erforderlich ist, zum potentiellen Leistungsengpaß.
Neben unzureichender Verarbeitungsleistung kann ein Leistungsengpaß auch darauf zurückzuführen sein, daß Protokollmechanismen unzureichend an die Eigenschaften des darunterliegenden Netzwerks angepaßt sind. Während die konventionellen Internetzwerke in der Regel von einer verbindungslosen Netzwerkschicht geprägt sind, ist ATM ein verbindungsorientier-

tes Protokoll. Aus diesem Grund eignen sich die für eine verbindungslose Netzwerkschicht entwickelten Transportprotokolle nur bedingt für ATM-Netze.
In bestehenden paketvermittelten Netzwerken treten Pakete variabler Länge auf, die häufig mehrere Kilobyte groß sind. Im Überlastfall treten in den Netzknoten Pufferüberläufe auf, bei denen Pakete verworfen werden. In den Netzknoten bieten diese Pakete variabler Länge die Basiseinheit des Multiplexens zur Zuteilung von Netzressourcen. Im Gegensatz dazu bilden in ATM-Netzen kurze Zellen mit einer Länge von 53 byte die Basiseinheit des Multiplexens. Überlastete ATM-Knoten verwerfen einzelne Zellen. Bei Verwendung der Adaptionstypen AAL3/4 und AAL5 hat schon ein einzelner Zellverlust zur Folge, daß ein aus bis zu mehreren hundert Zellen bestehendes Paket komplett verloren geht [StGr94]. Daher ist eine Fehlerbehebung in der Transportschicht, wo immer der Verlust ganzer Pakete behoben werden muß, nicht geeignet, Fehler durch beschädigte oder verlorene ATM-Zellen effizient zu beheben. Bei einer Behebung von Zellverlusten in der Transportschicht führt häufig schon eine verhältnismäßig geringe Zellverlustwahrscheinlichkeit zu einem dramatischen Einbruch der Dienstqualität [Roma93].
Um zuverlässige Gruppenkommunikationsdienste in ATM-Netzen auf effizientere Weise als mit den bisher vorhandenen Protokollen erbringen zu können, ist die Entwicklung neuer Protokolle erforderlich, die an die spezifischen Randbedingungen in ATM-Netzen besser angepaßt sind. Dabei ist es wichtig, Fehlerkontrollmechanismen zur Verfügung zu haben, die beim Auftreten von Zellverlusten eine große Leistungsfähigkeit besitzen. Da bei der Gruppenkommunikation eine Vielzahl unterschiedlicher Randbedingungen möglich sind, ist es auch wichtig, Fehlerkontrollmechanismen einsetzen zu können, die an die charakteristischen Eigenschaften eines Kommunikationsszenarios angepaßt sind. Gleichzeitig müssen diese Mechanismen mit geringem Implementierungsaufwand realisierbar sein, um Leistungsengpässe durch die Protokollverarbeitung zu vermeiden.

2.2 FEC-SSCS: Dienstspezifische Konvergenzteilschicht mit FEC für AAL5

Ein Schwerpunkt unserer Untersuchungen bildete ein Adaptionsschichtprotokoll mit Vorwärtsfehlerkorrektur, dessen Schichtmodell in Abbildung 1 gezeigt ist. Bei dem innerhalb des ATM-Forums diskutierten Protokolls FEC-SSCS (FEC Service Specific Convergence Sublayer, [CaEG95a, CaEG95b]) handelt es sich um eine dienstspezifische Konvergenzteilschicht für AAL5, die zellenbasierte Vorwärtsfehlerkorrektur mit einer vom Sender dynamisch anpaßbaren Anzahl von Redundanzzellen ermöglicht. FEC-SSCS verwendet pro 46 byte großem Datensegment einen Protokollkopf von 2 byte Länge, der eine Zellsequenznummer sowie die Prüfsumme CRC-10 enthält. Damit können Bitfehler und Zellverluste erkannt werden. Zur Fehlerkorrektur werden innerhalb der AAL5-Nutzlast zusätzliche 46 byte große Redundanzsegmente hinzugefügt, die zusammen mit dem Protokollkopf die sogenannten Redundanzzellen bilden. Die Protokollverarbeitung beim Empfänger ist im verlustlosen Fall sehr gering, da lediglich die ursprüngliche Nutzdaten reassembliert werden müssen und die Redundanzzellen einfach verworfen werden können. Während die ursprüngliche Spezifikation von FEC-SSCS [CaEG95b] ausschließlich Reed-Solomon-Codes zur Generierung der Redundanzzellen einsetzt, untersuchten wir zusätzlich eine Generierung von Redundanzzellen mit einem Matrix-basierten XOR-Verfahren.
Wie in [EsCa95] gezeigt, läßt sich FEC-SSCS auch sehr vorteilhaft in Verbindung mit dem Protokoll SSCOP [Q2110] als höhere Schicht einsetzen. Unsere Implementierung von FEC-SSCS erlaubt, wie in Abschnitt 6 gezeigt, den Einsatz von SSCOP über FEC-SSCS.

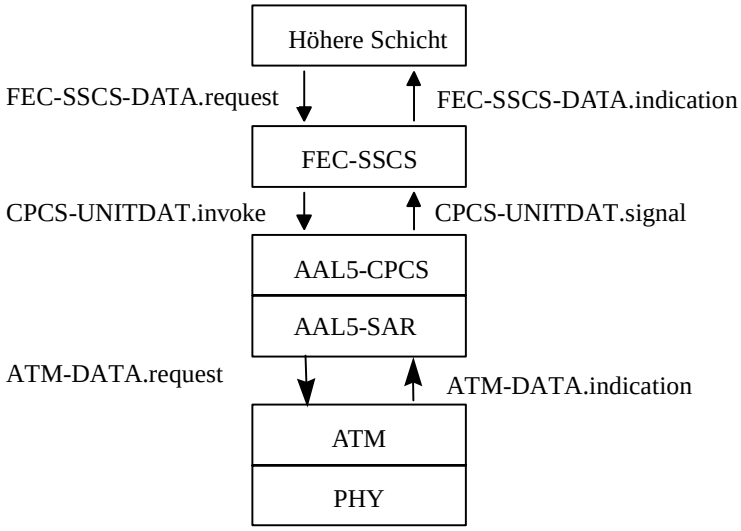

Abbildung 1: Dienstspezifische Konvergenzteilschicht mit FEC für AAL5

2.3 RMC-SSCS: Dienstspezifische Konvergenzteilschicht mit zellenbasierter Vorwärtsfehlerkorrektur und Übertragungswiederholung für AAL5

Das RMC-AAL-Protokoll (Reliable Multicast ATM Adaptation Layer, [CaZi95]) wurde für die Dienstspezifische Konvergenzteilschicht (SSCS) von AAL5 entwickelt und daher auch als RMC-SSCS bezeichnet. Es verfügt neben zellenbasierter FEC über die Möglichkeit, rahmenbasierte oder zellenbasierte Übertragungswiederholungen durchzuführen. RMC-SSCS kann sowohl in Endsystemen als auch in speziellen Zwischensystemen, sogenannten Gruppenkommunikationsservern, eingesetzt werden, siehe Abbildung 3 [CaZi95].

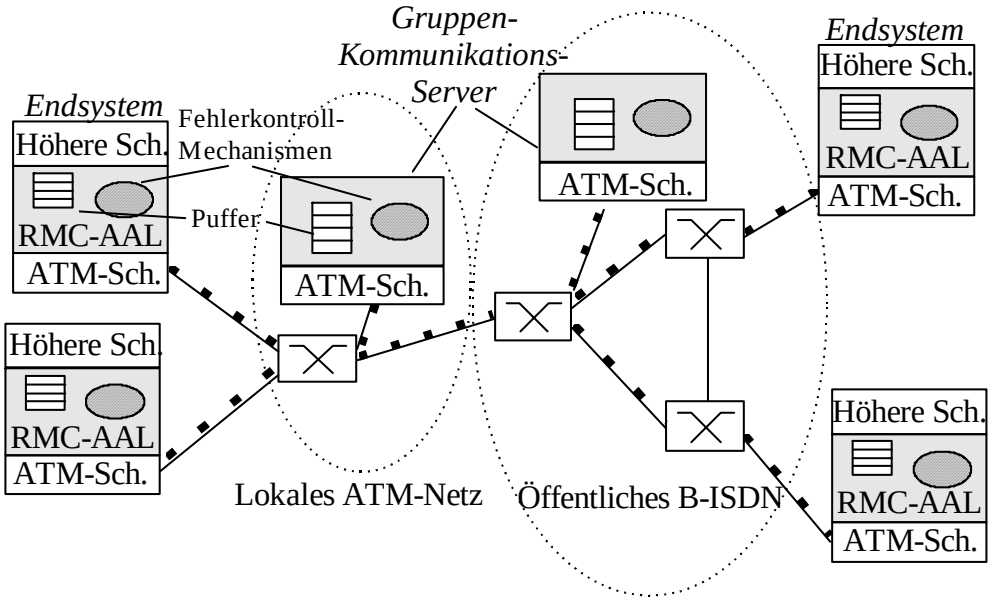

Abbildung 2: Adaptionsschichtprotokoll RMC-AAL mit Mechanismen zur Unterstützung zuverlässiger Multicastkommunikation

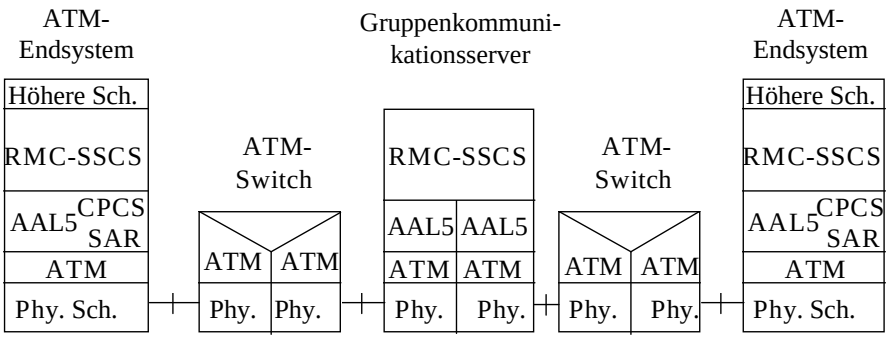

Abbildung 3: Schichtmodell für RMC-AAL

RMC-SSCS verwendet einen Rahmenkopf von 10 byte und kann dadurch zuverlässige Dienste mit einer höheren Effizienz erbringen, als dies bei der Verwendung eines Transportprotokolls wie TCP oder XTP möglich wäre, die einen deutlich größeren Paketkopf verwenden. Durch ein spezielles Quittungsformat mit Binärfeldern stellt die Quittungsverarbeitung von RMC-SSCS auch für eine große Empfängerzahl nur geringe Anforderungen an die Verarbeitungsleistung, wodurch sich das Problem einer Quittungsimplosion abschwächen läßt.
Für höhere Zellverlustwahrscheinlichkeiten sowie für Anwendungen, die besonders niedrige Verzögerungszeiten fordern, verfügt RMC-SSCS über einen Fehlerkontrollmechanismus mit zellenbasierter Übertragungswiederholung. Durch die Unterstützung des Strommodus kann mit RMC-SSCS im Vergleich zu konventionellen Protokollen eine besonders niedrige Verzögerung erzielt werden, da beim Sender auch von unvollständig vorliegende Rahmen schon einzelne Zellen gesendet werden können, und beim Empfänger Fehler auch schon bei unvollständig empfangenen Rahmen erkannt werden können.
RMC-SSCS verwendet hierarchische Sequenznummern, bestehend aus einer Rahmensequenznummer von 24 bit Länge und einer Zellsequenznummer von 6 bit Länge. Gegenüber existierenden Vorschlägen für eine zellenbasierte Fehlerkontrolle [BoLa93] konnte der zusätzliche Bandbreitenbedarf deutlich gesenkt werden. Durch die Einführung von sogenannten Rahmenfragmenten für die zellenbasierte Übertragungswiederholung lassen sich Zellverluste auch für Verbindungen mit großer Pfadkapazität effizient beheben. Rahmenfragmente bestehen aus einer Kopfzelle zur Identifikation der Wiederholung, gefolgt von einer Sequenz zu wiederholender Zellen.
Das FEC-Verfahren von RMC-SSCS verwendet XOR-Operationen zur Generierung von Redundanzzellen und zur Wiederherstellung verlorener Zellen. Dieses Verfahren zeichnet sich durch einen geringen Verarbeitungsaufwand sowie durch einer große Robustheit gegenüber den in ATM-Netzen typischen korrelierten Zellenverlusten aus und ermöglicht außerdem eine dynamische Änderung der Redundanz.
Der Einsatz von Gruppenkommunikationsservern bietet insbesondere im Weitverkehrsbereich und bei großen Gruppen eine wirkungsvolle Unterstützung für Quittungsverarbeitung, Übertragungswiederholung und Vorwärtsfehlerkorrektur. Mit Gruppenkommunikationsservern ist außerdem eine Leistungssteigerung bei heterogenen Gruppen möglich, bei denen innerhalb der Gruppe große Unterschiede bezüglich der Verbindungseigenschaften sowie bezüglich Funktionalität und Leistungsfähigkeit der Gruppenteilnehmer bestehen. Außerdem unterstützt ein Gruppenkommunikationsserver das Multiplexen mehrerer Sender über eine einzige ATM-Verbindung, womit die Skalierbarkeit bei Gruppen mit mehreren Sendern verbessert werden kann.

3 Implementierung von Protokollmechanismen in die ATM-Adaptionsschicht

Die vollständig funktionsfähige Implementierung der Fehlerkorrekturmechanismen wurde auf PCs unter dem Betriebssystem Linux [Linux] ausgestattet mit 155 Mbps ATM-Adaptern [ENI96] durchgeführt. Abbildung 4 zeigt dabei die Lage des FEC-Moduls innerhalb des Schichtenmodells an. Das FEC-Modul befindet sich auf der Hardware-unabhängigen Seite, so daß es prinzipiell für beliebige ATM-Adapterkarten einsetzbar ist. Das Modul ATM-sockets stellt die Treiberschnittstelle für Anwendungsprogramme dar. Hierüber können außer den üblichen Befehlen zur Datenübertragung alle FEC-Parameter beinflußt werden. Ein Beispiel hierfür ist der folgende Befehl, der die FEC-Parameter auf 2 Redundanzzellen pro AAL-Rahmen, Reed-Solomon-Algorithmus und aktivierte Bitkorrektur setzt.

```
ioctl(socket, FEC_SETMODE, FEC_MODE(2, FEC_REEDSOLOMON,
      FEC_BITCORRECTION));
```

Die aktuell gültigen FEC-Parameter können mittels

```
result = ioctl(socket, FEC_GETMODE, 0);
```

abgefragt werden. Die Anwenderschnittstelle wurde also in die bereits vorhandene Schnittstelle zur Steuerung der Gerätetreiberoptionen (ioctl, input/ouput control) eingebunden. Der Vorteil dieser Schnittstelle liegt in der einfachen Handhabung und Aufrufbarkeit. Für die FEC-Parameter selbst wurde ein bisher ungenutztes Feld innerhalb der socket-Struktur verwendet. Ein weiterer Vorteil dieser Lösung liegt darin, daß bereits existierende Anwendungen, die kein Gebrauch der FEC-Erweiterung machen, weiterhin problemlos betrieben werden können, da für sie kein Unterschied in der Anwenderschnittstelle sichtbar ist.

Das Modul ATM-Koordination ist für die gesamte Steuerung des Hardware-unabhängigen Teils des Treibers zuständig, darunter auch für Senden und Empfang der Daten. Die Steuerung des Zugriffs auf die Hardware der Adapterkarte führt das Modul PHY-Treiber durch, die Segmentierung und Reassemblierung der Daten übernimmt das Modul SAR-Treiber.

Das implementierte FEC-Modul (siehe Abbildung 4) erhält von der ATM-Koordination die zu sendende Nutzdateneinheit. Nun wird die für die zugehörige Verbindung gültige FEC-Einstellung verwendet, um entweder im Reed-Solomon- oder im XOR-Kodierer Redundanzsymbole für die Nutzdateneinheit zu berechnen. Die Einstellungen für die Verbindungen sind dabei jeweils unabhängig voneinander. Die Nutzdateneinheit wird nun um einen SSCS-PDU-Kopf erweitert und zusammen mit den Redundanzdaten in Segmente von 46 byte Länge zerlegt, von denen jedes mit einem Erkennungskopf versehen wird. Diese Datensegmente werden dem SAR-Treiber zum Versand übergeben. Ist der Bitfehler-Erkennungsmodus aktiviert, so wird jedem Erkennungskopf eines Datensegments noch eine CRC-10-Prüfsumme hinzugefügt.

Beim Empfang von Daten werden diese vom SAR-Treiber an das FEC-Modul weitergeleitet. Anhand des SSCS-PDU-Kopfes wird die ursprüngliche Größe des Datenblocks und die verwendeten FEC-Parameter ermittelt. Verlorengegangene Zellen können leicht mittels des Erkennungskopfes ermittelt werden. Bei aktivierter Bitfehlererkennung werden zusätzlich noch die Prüfsummen der einzelnen Zellen überprüft. Bei einem Prüfsummenfehler wird eine Zelle als verloren markiert. Sind keine Übertragungsfehler aufgetreten, so werden die Nutzdaten extrahiert und an die ATM-Koordination weitergeleitet. Sind jedoch Fehler aufgetreten, so wird die beschädigte SSCS-PDU mit einer Liste der verlorenen Zellen je nach FEC-Parameter an die Reed-Solomon- bzw. XOR-Einheit weitergeleitet. Hier wird mit Hilfe der Redundanz versucht, die ursprünglichen Daten wieder herzustellen. Ist dies möglich, so wird das Paket wie im fehlerfreien Fall weitergeleitet ansonsten verworfen.

Die Implementierung des FEC-Moduls geschah vollständig im Kern des Betriebssystems Linux, da hier ATM-Gerätetreiber und das Betriebssystem als Quellcode vorhanden sind.

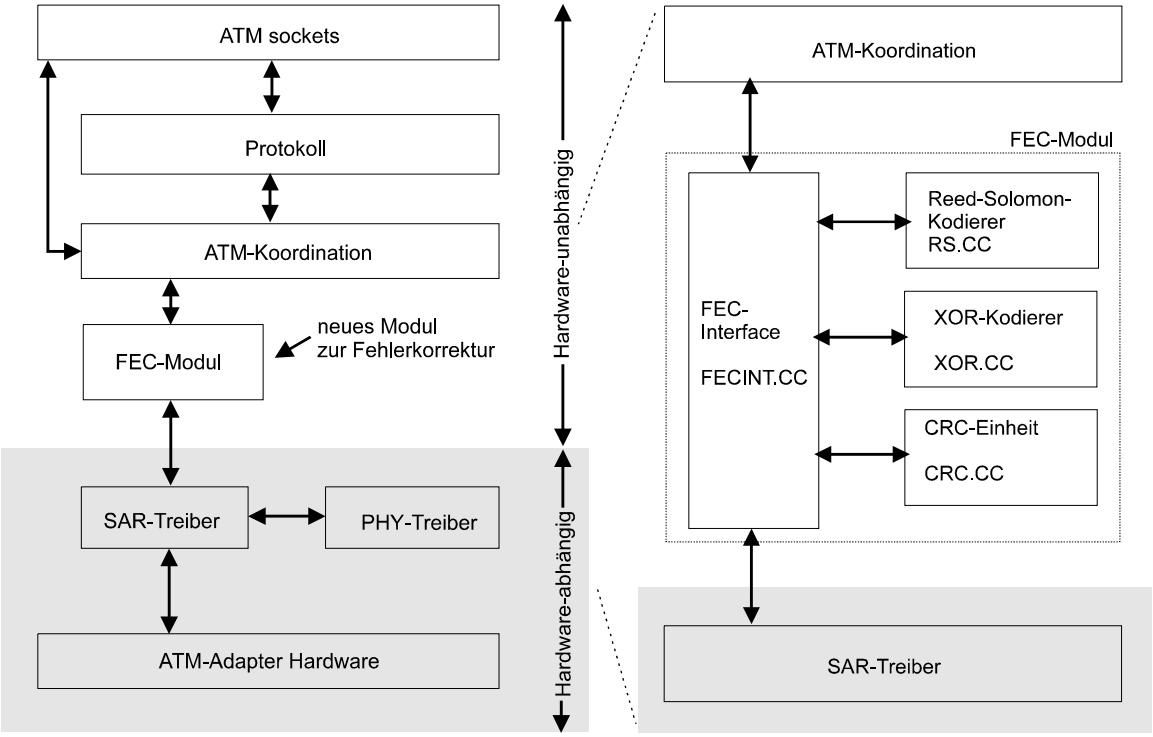

Abbildung 4: Modularer Aufbau des FEC-Moduls

4 Protokolltestumgebung

Mehr noch als bei konventionellen Protokollen gilt es bei Protokollen für die Gruppenkommunikation das Verhalten für eine große Anzahl an Teilnehmern unter realistischen Bedingungen zu testen. Für das hier gewählte Beispiel bedeutet dies unter anderem, daß die Leistungsfähigkeit der Fehlerkorrektur für verschiedene Fehlerwahrscheinlichkeiten nachgewiesen werden muß. Basierend auf realer ATM-Hardware lassen sich solche Untersuchungen nur sehr schwer durchführen, da sich beispielsweise Parameter wie Zellverlustrate oder Puffergröße innerhalb eines ATM-switches im allgemeinen nicht beeinflussen lassen. Aus diesem Grund wurde im Rahmen dieser Arbeiten auf das ATM-Softwarepaket VINCE (Vendor Independent Network Control Entity, [MaHP94]) zurückgegriffen, das eine Emulation von ATM-Netzwerken und ATM-Vermittlungseinrichtungen basierend auf konventionellen Netzen (mit TCP/IP) und Rechnern gestattet. Zwar liegt die absolute Leistungsfähigkeit dieser Software-Emulation um eine Größenordnung unter der realer ATM-Hardware, doch können nun relativ einfach alle benötigten Parameter beeinflußt werden.
Abbildung 5 gibt den Aufbau des Protokollstapels von VINCE mit der Erweiterung zur Fehlererzeugung und -behebung basierend auf TCP/IP an.

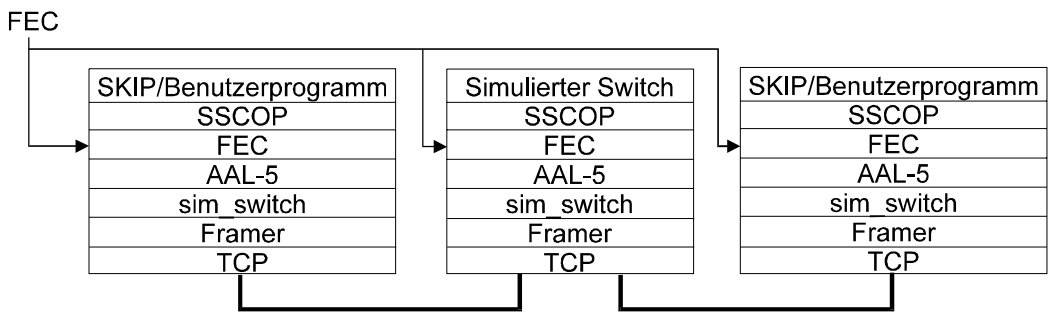

Abbildung 5: Schichtenaufbau der Protokolltestumgebung

Die Nachbildung der ATM-Hardware geschieht in den Schichten sim_switch und Framer. Darunter wird über eine Socket-Schnittstelle auf TCP/IP und die reale Hardware zugegriffen. Die Fehlererzeugung für die Testumgebung wurde in die Schicht sim_switch integriert, da sie Fehler im ATM-Netz nachbilden soll, d.h. die Möglichkeit zum Manipulieren bzw. Verwerfen von ATM-Zellen besitzen muß. Oberhalb von AAL5 folgen die Schicht zur Fehlerbehebung (FEC), darüber SSCOP (Service Specific Convergence Protocol) und schließlich die Benutzerprogramme/SKIP zur Steuerung von VINCE. Hier können nun zum Programmstart oder während der Laufzeit Funktionen innerhalb von VINCE (z.B. Erzeugen eines Switches, Verbindungsverwaltung) ausgeführt werden.
Als Anwendung wurde ein Programm zum Darstellen von MPEG-codierten Videosequenzen gewählt. Auf diese Weise ist es für einen Betrachter sehr einfach möglich, die Auswirkungen von Fehlern und die Fähigkeiten der Fehlerkorrektur zu bewerten. Abbildung 6 zeigt die Oberfläche des Programms XVINCE zur Eingabe der benötigten Parameter für die Fehlererzeugung und -behebung. Neben der Eingabe von Netzadressen und Befehlen können im oberen Teil des Fensters verschiedene Makros ausgeführt werden (Verbindungsaufbau etc.). Im mittleren Teil können für beliebige emulierte ATM-Switches Parameter wie Zellverlustrate, Bitfehlerwahrscheinlichkeit oder Anzahl der FEC-Redundanzzellen eingestellt werden. Sobald mit Hilfe dieser Software Verbindungen aufgebaut wurden, können beliebige Daten übertragen werden. Während der Übertragung können nun die Parameter geändert werden, um so direkt die Auswirkungen zu beobachten.

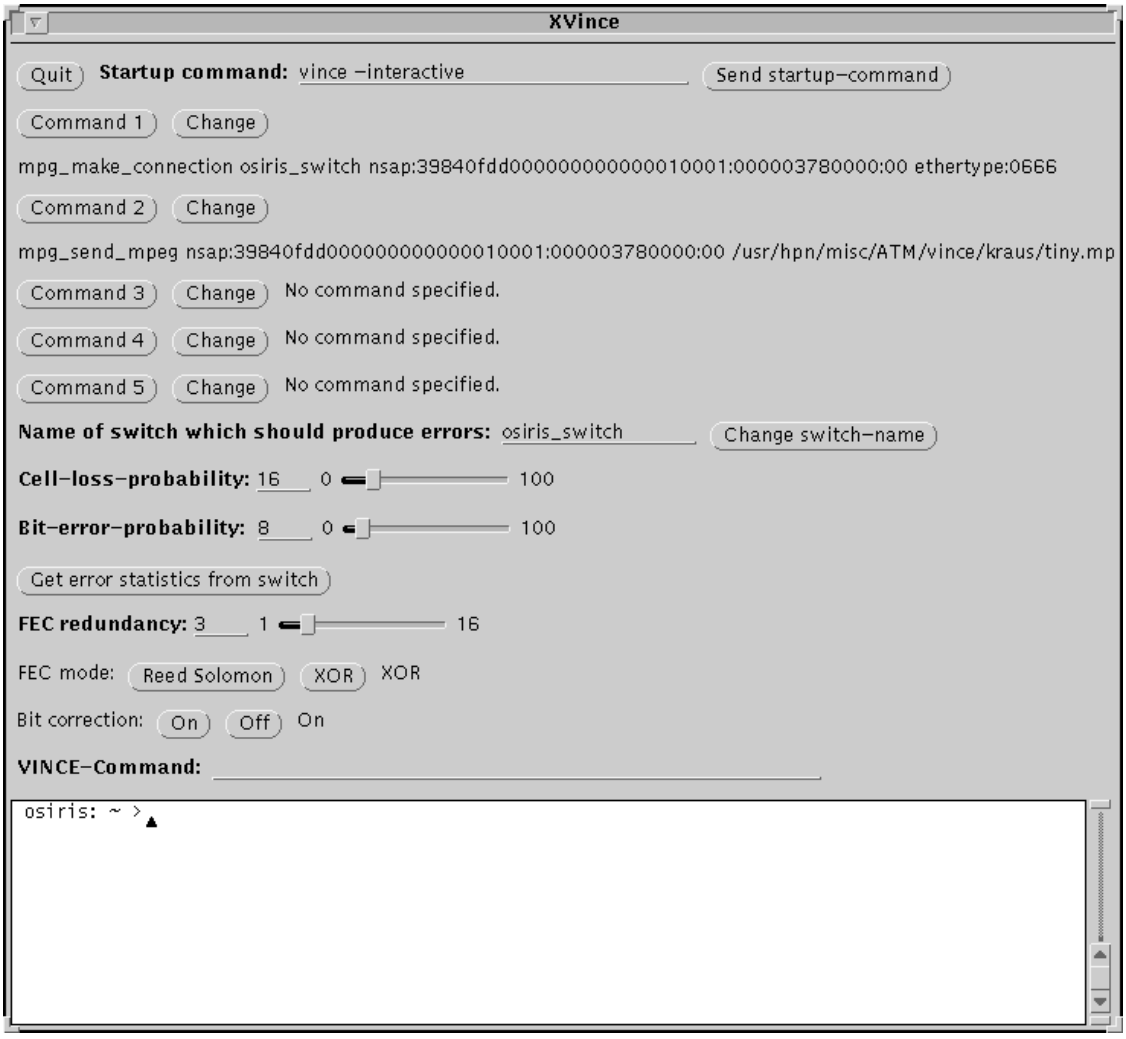

Abbildung 6: Oberfläche von XVINCE zur Steuerung der Parameter eines Software-switches

5 GAPPU: Generic ATM Protocol Processing Unit

Die in den vorigen Abschnitten vorgestellten Mechanismen zur Fehlerkorrektur können eine relativ große Rechenleistung erfordern. Dies gilt insbesondere beim Einsatz dieser Mechanismen in Zwischensystemen, die eine sehr große Gruppe von Empfängern zu bedienen haben. Hinzu kommt, daß es wesentlich wirtschaftlicher ist, in einigen wenigen Zwischensystemen eine sehr hohe Rechenleistung zur Verfügung zu stellen, um dann im Gegenzug Endsysteme relativ einfach gestalten zu können. So sollten beispielsweise Zwischensysteme mit adäquaten Fehlerkorrekturmechnismen für mobile, leistungsschwächere Endsysteme und hochleistungsfähige Arbeitsplatzrechner ausgestattet sein.
Um die für diese Heterogenität erforderliche Rechenleistung erbringen zu können, wurde das Konzept der Protokollverarbeitungseinheit GAPPU (Generic ATM Protocol Processing Unit) entwickelt. GAPPU stellt ein Rahmenwerk dar, das mehrere RISC-Prozessoren, dedizierte Hardware und Speicher umfaßt. Wesentlich ist hierbei, daß alle Komponenten durch eine hochgradig parallele Kopplungseinheit (Kreuzschienenverteiler) verbunden werden, die zusätzlich für einen vorhersagbaren und steuerbaren Datenaustausch zwischen den Komponenten sorgt. Dies bedeutet, daß im Rahmen der Datenübertragung innerhalb der Architektur im Gegensatz zu gängigen Systemen eine vorher festgelegte Dienstgüte garantiert werden kann. Abbildung 7 zeigt die Architektur von GAPPU, die vom Konzept her vergleichbar mit dem Multimedia-Prozessor TMS320C80 [Texa95] ist.

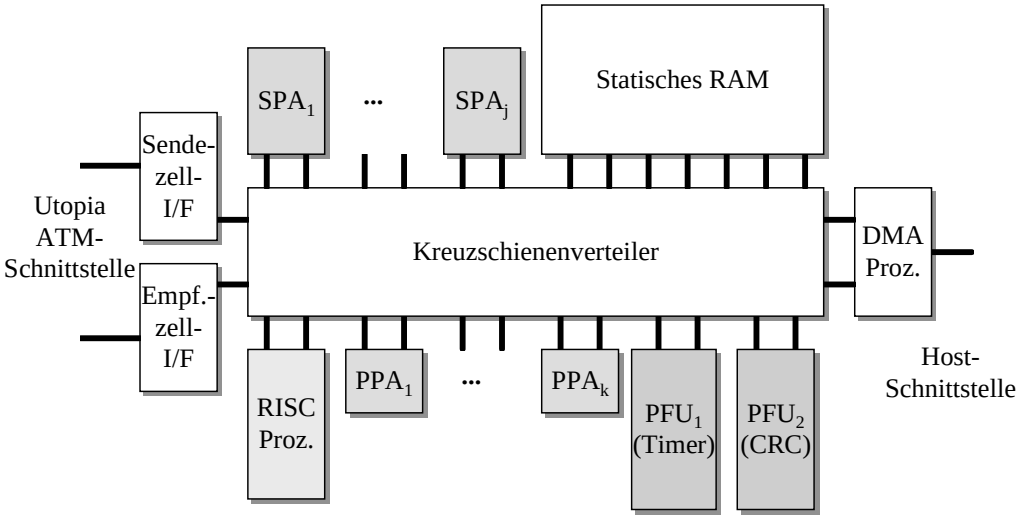

Abbildung 7: Architektur von GAPPU

Für das Senden und Empfangen von ATM-Zellen verfügt GAPPU über eine ATM-PHY-Schnittstelle gemäß der UTOPIA-Spezifikation [AFUT94]. Für leistungsunkritische Teile eines Kommunikationsprotokolls (z.B. Verbindungsverwaltung) können RISC-Prozessoren integriert werden. Sind leistungsfähige Protokollautomaten zu implementieren, so kann zwischen programmierbaren Protokollautomaten (PPA) und synthetisierten Protokollautomaten (SPA) gewählt werden. In PPAs wird ein Mikroprogramm abgearbeitet, das den entsprechenden Protokollautomaten repräsentiert. PPAs sind von ihrem Aufbau identisch und bieten spezielle Mechanismen zum schnellen Kontextwechsel und zur Verwaltung von Verbindungsdaten. SPAs werden anhand einer Beschreibung eines Protokollautomaten individuell generiert und stellen somit eine exakt an den jeweiligen Automaten angepaßte Hardware dar. Weiterhin können spezielle Funktionen zur Unterstützung eines Kommunikationsprotokolls in sogenannten Protocol Function Units (PFU) implementiert werden. Diese Einheiten werden von

Hand entworfen, da hier eine außerordentlich hohe Leistungsfähigkeit erforderlich ist. Beispiele hierfür sind CRC- und Zeitgebereinheiten [Schi96]. Allen Komponenten steht außer lokalem Speicher auf der Komponenten selbst noch ein gemeinsames statisches RAM zur Verfügung. Die Anbindung an zusätzliches dynamisches RAM geschieht über eine DMA-Einheit. Alle Komponenten innerhalb GAPPU besitzen identische Schnittstellen zur Ankopplung an den Kreuzschienenverteiler und können via Nachrichtenaustausch oder gemeinsamen Speicher miteinander kommunizieren.
In Abhängigkeit von der zu implementierenden Funktionalität und der gewünschten Leistungsfähigkeit können nun unterschiedliche Komponenten integriert werden. Die Architektur zielt auf eine Implementierung als leistungsfähiger ASIC (Application Specific Integrated Circuit) ab. Dafür wurden die Rahmenarchitektur und wesentliche Komponenten mit Hilfe der Hardware-Beschreibungssprache VHDL [IEEE87, LeWS94] beschrieben, simuliert und auf verschiedene Zieltechnologien synthetisiert. Anhand der Simulationsergebnisse läßt sich die Funktionalität des Entwurfs überprüfen. Die Synthese dient sowohl der Leistungsabschätzung als auch der Ermittlung des Flächenbedarfs als Chip.
Als Vielzweckprozessor zur Abarbeitung von Protokollautomaten wird im Rahmen von GAPPU auf den RISC-Prozessor DLX [HePa94, SaKa96, FeRe94] zurückgegriffen, der eine gemeinsame Untermenge der wesentlichen Eigenschaften gängiger RISC-Prozessoren umfaßt und sowohl als Simulationsmodell als auch in einer vollständig synthetisierbaren Version vorliegt. Wie in [Gumm95] gezeigt, läßt sich ein DLX-Prozessor mit lediglich 60.000 Transistoren realisieren. Als dedizierte Komponenten zur Unterstützung der Protokollabarbeitung (PFU) wurden seither eine Einheit zur Verwaltung dynamischer Datenstrukturen, eine Zeitgeberverwaltung sowie Komponenten zur Berechnung von Redundanzdaten für die Vorwärtsfehlerkorrektur entworfen, simuliert und auf programmierbare Bausteine (FPGAs [Virt95, Xili94]) und 0,7 µm CMOS-ASICs synthetisiert [Euro94, Syno94, Cade94]. Bei den entworfenen Einheiten handelt es sich jeweils um Komponenten, welche die RISC-Prozessoren von besonders zeitintensiven Rechenoperationen entlasten. Dieses Konzept ist vergleichbar mit dem Einsatz von speziellen Gleitkommaarithmetikeinheiten in gängigen PCs [Schi95]. Bei Gruppenkommunikation mit hohen Datenraten würde die Abarbeitung dieser Funktionen in Software aufgrund der großen Anzahl von zu unterstützenden Verbindungen und Empfängern zu Leistungsengpässen und zu einer Verschlechterung der Dienstgüte führen. Geforderte Garantien für die Dienstgüte könnten bei der angestrebten Leistungsfähigkeit somit nicht mehr eingehalten werden.
Im Rahmen des Entwurfs eines Kommunikationssystems ist es ebenfalls von großer Wichtigkeit, daß Kommunikationsprotokolle ohne Abänderung der Protokolle selbst auf den bisher vorgestellten unterschiedlichen Plattformen realisiert werden können. Besonders wichtig ist die effiziente und einfache Implementierung auf einer Hardware-Realisierung wie es das System GAPPU darstellt. Ausgangspunkt unserer Protokollimplementierung stellt die Spezifikation des Kommunikationssystems in der standardisierten und im Telekommunikationsbereich weit verbreiteten Sprache SDL [CCIT89] dar. Basierend auf dieser Spezifikation kann nun C-Code für eine Software-Implementierung [Veri95] oder VHDL-Code für eine Hardware-Implementierung als SPA automatisch generiert werden [Schi96]. Falls flexible Protokollautomaten höchster Leistungsfähigkeit benötigt werden, so müssen PPAs eingesetzt werden. Hier wird ebenfalls mit Compiler-Unterstützung ein für den Menschen einfach lesbarer Assemblercode direkt in Microcode für den Protokollautomaten abgebildet.
Besonders elegant ist hierbei die Möglichkeit der Übersetzung eines C-Programms in den Maschinencode für den DLX-RISC-Prozessor mit Hilfe eines GNU-C-Compilers. Dies bedeutet, daß C-Code für reine Software-Implementierungen übernommen und nun auf den RISC-Prozessoren innerhalb der GAPPU abgearbeitet werden kann. Eine natürliche Aufteilung des Programmes auf die verschiedenen Prozessoren ergibt sich im allgemeinen aus der Tatsache,

daß RMC-SSCS wie auch zahlreiche andere Kommunikationsprotokolle aus einer Menge miteinander kommunizierender Automaten besteht. So können nun, in Abhängigkeit von der benötigten Leistungsfähigkeit ein/mehrere Protokollautomat(en) auf eine RISC-CPU abgebildet werden.
Die Automatisierung der Abbildung einer Protokollspezifikation auf leistungsfähige Implementierungen stellt eine wesentliche Erhöhung der Korrektheit der Systeme dar, da viele Fehler der Handimplementierung ausgeschlossen werden können.

6 Ergebnisse

Für die Leistungsmessungen der Implementierung der FEC-Mechanismen unter Linux wurde ein 90 MHz Pentium-PC mit 16 Mbyte RAM und 155 Mbps ATM-Adapter eingesetzt. Das Testprogramm übertrug pro Kombination der FEC-Parameter fünf Mal eine 3 Mbyte große Datei und ermittelte dabei den mittleren Datendurchsatz und die mittlere Kodierzeit. Hierbei wurden die Zellverluste so eingestellt, daß für das FEC-Modul jeweils der maximale Dekodieraufwand entstand. In Abbildung 8 macht sich, wie zu erwarten, der Geschwindigkeitsgewinn des einfacheren XOR-Verfahrens deutlich bemerkbar.

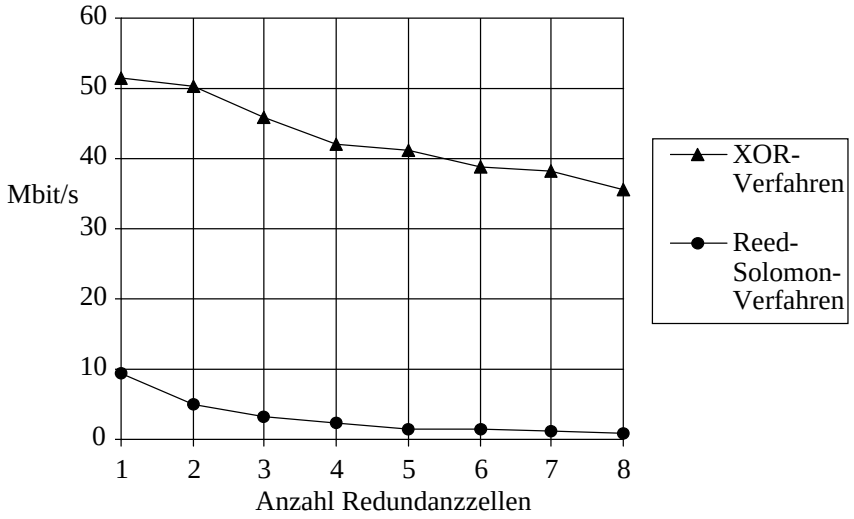

Abbildung 8: Leistungsvergleich der FEC-Verfahren bei Software-Implementierung (Intel)

Anhand der Ergebnisse läßt sich feststellen, daß das Reed-Solomon-Verfahren für eine reine Software-Implementierung zu rechenintensiv ist. Mit dem XOR-Verfahren lassen sich deutlich höhere Übertragungsraten erzielen, jedoch muß auch hierbei beachtet werden, daß bei den in Abbildung 8 angegebenen Datenraten die CPU des Rechners vollständig mit der Kodierung/Dekodierung ausgelastet ist und keine weiteren Kapazitäten für Anwendungen verfügbar sind. Hinzu kommt, daß das XOR-Verfahren außerordentlich einfach in Hardware zu implementieren ist, auch für das RS-Verfahren existieren Hardware-Ansätze. Insgesamt wäre also eine Unterstützung dieser Funktionalität in Hardware auf dem ATM-Adapter wünschenswert.
Da noch nicht alle benötigten Komponenten vollständig simuliert und synthetisiert wurden, mußten sich bisherige Leistungsmessungen der GAPPU auf Teilkomponenten beschränken. Die dabei erzielten Ergebnisse zeigen, daß gegenüber einer Software-Lösung eine deutliche Leistungssteigerung erreicht werden kann, so daß der Ansatz insgesamt als sehr vielversprechend bezeichnet werden kann. Zusammengefaßt konnten seither folgende Kenngrößen ermittelt werden. Bei der Analyse der Verarbeitungszeit in Abbildung 9 wurde von einer Verar-

beitung des RMC-SSCS-Protokolls auf einer GAPPU mit 6 RISC-Prozessoren sowie weiteren dedizierten Hardware-Komponenten ausgegangen. Hierbei wurden 32-bit RISC-Prozessoren mit einer mittleren Verarbeitungsleistung von 100 MIPS vorausgesetzt, was beim heutigen Stand der Technik eine konservative Annahme darstellt. Als PFUs wurden Hardware-Komponenten für Segmentierung und Reassemblierung, für die Berechnung der CRC-Prüfsummen und für die Vorwärtsfehlerkorrektur vorausgesetzt. Die Verarbeitung von Nutzzellen des RMC-SSCS-Protokolls erfolgt in den Komponenten Empfangsprozessor, Frame Manager Receive, Frame Manager Send, Send Manager und Sendprozessor. Abbildung 9 zeigt die Verarbeitungszeiten der einzelnen Komponenten für die unterschiedlichen Fehlerkontrollverfahren des RMC-SSCS-Protokolls. Hierbei wird die Verarbeitungszeit der Komponente Frame Manager Receive in drei Funktionsbereiche unterteilt: ARQ rahmenbasiert, ARQ zellenbasiert, und FEC. Für das rahmenbasierte ARQ-Verfahren von RMC-SSCS muß lediglich der Funktionsbereich ARQ rahmenbasiert verarbeitet werden, wobei sich die geringste Verarbeitungszeit ergibt. Für das zellenbasierte ARQ-Verfahren müssen die Funktionsbereiche ARQ rahmenbasiert und ARQ zellenbasiert verarbeitet werden. Für rahmenbasiertes ARQ mit FEC müssen die Funktionsbereiche ARQ rahmenbasiert und FEC verarbeitet werden, während für zellbasiertes ARQ mit FEC alle drei Funktionsbereiche verarbeitet werden müssen. Da im untersuchten Fall von einer dedizierten PFU für FEC ausgegangen wird, fallen für den Funktionsbereich FEC pro verarbeiteter Zelle lediglich 0,11 µs zur Kommunikation mit der FEC-Komponente an.

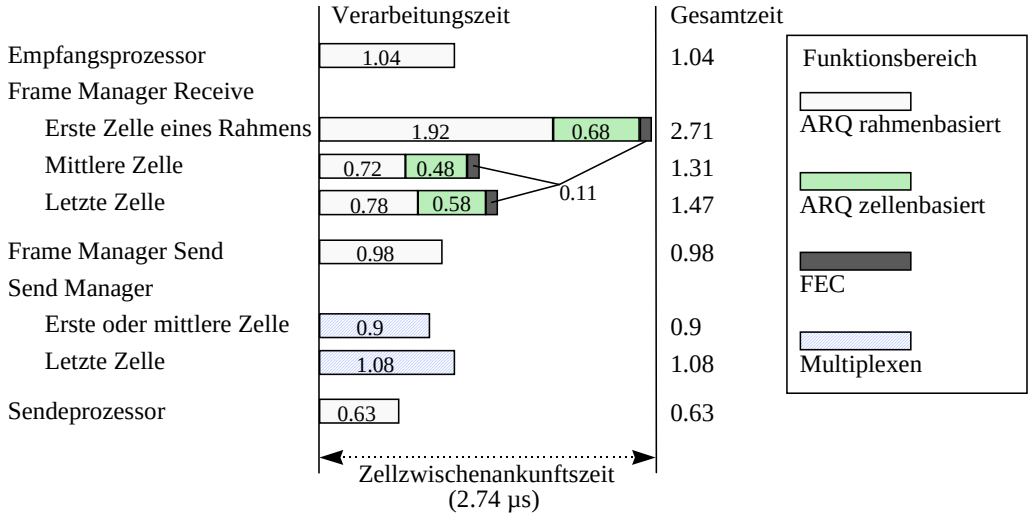

Abbildung 9: Verarbeitungszeit für Nutzzellen in den einzelnen Prozessoren

Eine Integration von 6 DLX-RISC-Prozessoren, dedizierten Komponenten für die Verwaltung dynamischer Datenstrukturen und Zeitgebern, plus ein 16 bit breiter Kreuzschienenverteiler für 14 Sender-/Empfänger resultiert in einem Flächenbedarf von unter 200000 Gattern. Dabei nehmen die RISC-Prozessoren 60%, der Kreuzschienenverteiler 18%, die dedizierten Hardware-Einheiten zur Unterstützung der Prozessoren 11% und schließlich die restliche Logik (Schnittstellen, Treiber, Testpfad etc.) ebenfalls 11% der Chipfläche ein. Hinzu kommt der Flächenbedarf für lokalen Speicher. Im Vergleich dazu besitzt der Prozessor TMS320C80 ca. 4 Millionen Transistoren, was in etwa 1 Million Gatter entspricht. Diese Ergebnisse sprechen also für die Realisierbarkeit des Entwurfs. Als maximale Taktfrequenz des bisherigen Entwurfs wurde unter konservativen Annahmen 50 MHz ermittelt. Diese im Vergleich zu RISC-Prozessoren niedrige Frequenz liegt darin begründet, daß pro Takt innerhalb der GAPPU eine wesentlich höhere Funktionalität abgearbeitet wird. Als Beispiel hierfür dient die Listenver-

waltung, deren Leistung in Abbildung 10 mit einem gängigen RISC-Prozessor verglichen wird. Dazu wurde die mittlere Verarbeitungszeit für das Einfügen eines neuen Eintrags in eine listenbasierte Zeitgeberverwaltung untersucht. Hier zeigt sich deutlich, wie trotz einer niedrigeren Taktfrequenz von 20 MHz bei der FPGA-Lösung eine höhere Leistungsfähigkeit als bei der Verwendung eines Alpha-AXP-Prozessors mit einer Taktfrequenz von 175 MHz erreicht werden kann. Bei der Software-Implementierung auf dem RISC-Prozessor wurden einmal die Meßwerte ohne die gleichzeitige Überprüfung auf einen Alarm (AXP21064, ohne) und einmal mit dieser Überprüfung ermittelt. Ein positiver Effekt der niedrigen Frequenz der FPGA-Lösung ist eine wesentliche Vereinfachung des Systementwurfs und die Möglichkeit, billigere Teilkomponenten (z.B. Speicher) zu verwenden.

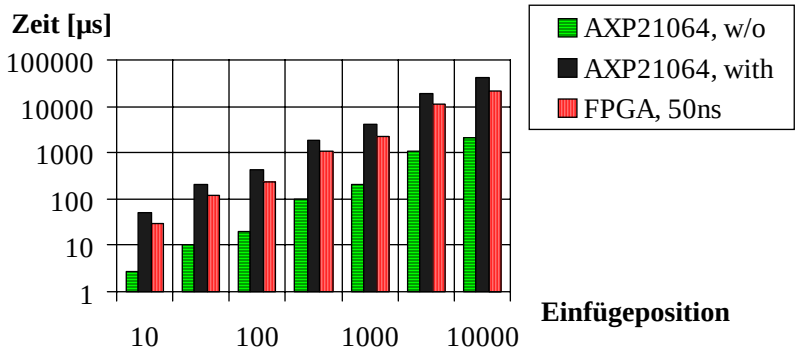

Abbildung 10: Leistungsvergleich zwischen Software- und Hardware-Implementierung von Zeitgebern

7 Zusammenfassung

Mit der Einführung von ATM-Netzen steht ein leistungsfähiges Verfahren zur Datenübertragung zur Verfügung, womit eine Vielzahl unterschiedlicher Dienste realisiert werden kann. An diese Netze werden in Zukunft Endsysteme mit unterschiedlicher Leistungsfähigkeit angeschlossen. Auch auf Seiten der Übertragungstechnologie stehen Alternativen von Glasfaser bis Funk mit stark unterschiedlichen Charakteristiken zur Verfügung. Sollen nun Protokolle für ATM entworfen werden, so müssen sie ihre Tauglichkeit für viele Implementierungsalternativen unter Beweis stellen. In diesem Beitrag wurde anhand von Protokollen mit ATM-spezifischen Fehlerkontrollmechanismen für FEC und ARQ aufgezeigt, wie Implementierungsalternativen realisiert und getestet werden können. Die betrachteten Protokolle eigenen sich nicht nur für den Einsatz in Endsystemen, sondern auch für spezielle Gruppenkommunikationsserver. Wie in einer ausführlichen Motivation dargestellt, ist diese Art von Protokollen von besonderem Interesse, sollen in Zukunft beispielsweise Konferenzsysteme mit einer Vielzahl von Teilnehmern oder andere verteilte Anwendungen kostengünstig und dennoch leistungsfähig aufgebaut werden. Traditionelle Protokolle sind nur schlecht mit der Teilnehmerzahl skalierbar und schwer an die ATM-Technologie anpaßbar. Insbesondere für mobile Anwendungen stellt die Verlagerung der Fehlerkorrektur in Zwischensysteme den einzig sinnvollen Weg dar, da hier höhere Fehlerwahrscheinlichkeiten zwischen Feststation und Mobilteilnehmer auftreten, und ohne eine Fehlerkontrolle in den Zwischensystemen die Dienstqualität aller Empfänger einer Gruppe in Mitleidenschaft gezogen würde.
Die FEC-Mechanismen wurden in Software und Hardware implementiert und vermessen. Hierbei konnte mit Hilfe einer ATM-Emulation das Verhalten der Mechanismen unter verschiedenen Fehlerwahrscheinlichkeiten überprüft werden. Mit gängigen PCs reicht dabei die

Leistungsfähigkeit aus, um die Auswirkungen anhand eines MPEG-Videodatenstroms optisch nachzuvollziehen. Um jedoch das Konzept der Vorwärtsfehlerkorrektur sinnvoll für große Gruppen einsetzen zu können, ist eine Hardware-Komponente wie die entworfene GAPPU unabdingbar. GAPPU stellt eine hochleistungsfähige Architektur für Zwischensysteme dar, die eine Vielzahl von Protokollfunktionen übernehmen kann. Es wurde anhand von Leistungsmessungen gezeigt, daß diese Hardware leistungsfähig genug ist, um selbst komplexe Operationen innerhalb von Zwischenankunftszeiten von ATM-Zellen durchzuführen.
Aktuelle Arbeiten umfassen vor allem die Erweiterung der Funktionseinheiten für GAPPU und deren vollständige Synthese als ASIC. Hierzu wurden bereits detaillierte Beschreibungen in VHDL vorgenommen. Zum Test der einzelnen Funktionen werden diese zusätzlich auf ein in einen Arbeitsplatzrechner integriertes FPGA-Board abgebildet.

8 Literatur

[AFUT94] ATM Forum: UTOPIA, An ATM-PHY Interface Specification, Level 1, Version 2, März 1994

[BoLa93] Bondi, A.; Lai, W.-S.: The influence of cell loss patterns and overheads on retransmission choices in broadband ISDN, Computer Networks and ISDN Systems 26, S. 585-598, 1994

[Cade94] Cadence Design Systems, Inc.: Dokumentation zur Entwurfsumgebung DFW II, Cadence Design Systems, Inc., San Jose, U.S.A., 1994

[CaSc95] Carle, G.; Schiller, J.: Enabling High Bandwidth Applications by High-Performance Multicast Transfer Protocol Processing, 6th IFIP Conference on Performance of Computer Networks, PCN95, Istanbul, Türkei, Oktober 1995

[CaEG95a] Carle, G.; Esaki, H.; Guha, A.; Tsunoda, K.; Kanai, K.: Necessity of an FEC Scheme for ATM Networks, Contribution ATMF/95-0325; ATM Forum Technical Committee 'Service Aspects and Applications SA&A', Denver, Colorado, U.S.A., April 1995

[CaEG95b] Carle, G.; Esaki, H.; Guha, A.; Tsunoda, K.; Kanai, K.: Proposal for Specification of FEC-SSCS for AAL Type 5, Contribution ATMF/95-0326; ATM Forum Technical Committee 'Service Aspects and Applications SA&A', Denver, Colorado, U.S.A., April 1995

[CaZi95] Carle, G.; Zitterbart, M.: ATM Adaptation Layer and Group Communication Servers for High-Performance Multipoint Services, 7th IEEE Workshop on Local and Metropolitan Area Networks, Marathon, Florida, März 1995

[CCIT89] CCITT: Functional Specification and Description Language (SDL), Recommendations Z.100-Z.104, Blue Book, Oktober 1989

[EsCa95] Esaki, H.; Carle, G.: Combination of SSCOP and an AAL-Level FEC Scheme, Contribution ATMF/95-1560; ATM Forum Plenary, London, U.K., Dezember 1995

[ENI96] Efficient Networks, Inc.: ATM Adapter ENI 155p for PCI Bus Product Information, Hardware Installation Guide, Hardware Device Interface User’s Guide, 1996

[Euro94] European Silicon Structures: Dokumentation zur 0,7 µm-Bibliothek, European Silicon Structures, Rousset, Frankreich, 1994

[Gumm95] Gumm, M.: VHDL-Modelling and Synthesis of the DLXS-RISC-Prozessor; VLSI Design Course, Institut für Parallele und Verteilte Höchstleistungsrechner (IPVR), Universität Stuttgart, 1995

[FeRe94] Feldman, J.M.; Retter, C.T.: Computer architecture: a designer’s text based on a generic RISC, McGraw-Hill, 1994

[HePa94] Hennessy, J.L.; Patterson, D.A.: Rechnerarchitektur: Analyse, Entwurf, Implementierung und Bewertung, Vieweg Verlag, Braunschweig, 1994

[IEEE87] IEEE Std 1076-1987, VHDL: VHSIC Hardware Description Language

[KrKS93] Krishnakumar, A.S.; Kneuer, J.G.; Shaw, A.J.: HIPOD: An Architecture for High-Speed Protocol Implementations, in: Danthine, A.; Spaniol, O. (Eds.): High Performance Networking, IV, IFIP, North-Holland, 1993, S. 383-396

[KiFa95] Kim, H.; Farber, D.: The Failure of Conservative Congestion Control in High-Speed Networks, Proceedings of Second Gigabit Networking Workshop (GBN'95), Boston, MA, U.S.A., April 1995

[LeWS94] Lehmann, Wunder, Selz: Schaltungsdesign mit VHDL, Franzis-Verlag, 1994

[Linux] Linux Homepage, http://www.linux.org/

[MaHP94] Mankin, A.; Hoffman, E.; Perez, M.: Vendor Independent (and Architecture Flexible) Network Control Entity, Proceedings of INET'94, Internet Society, Prag, Juni 1994

[OhKi91] Ohta, H., Kitami, T.: A Cell Loss Recovery Method Using FEC in ATM Networks, IEEE Journal on Selected Areas in Communications, Vol. 9, No. 9, Dezember 1991, S.1471-1483

[Q2110] ITU-T Draft Recommendation Q.2110: B-ISDN Adaptation Layer - Service Specific Connection Oriented Protocol (SSCOP), International Telecommunication Union, Genf, 1994

[Roma93] Romanov, A.: Some Results on the Performance of TCP over ATM, Second IEEE Workshop on the Architecture and Implementation of High Performance Communication Subsystems HPCS'93, Williamsburg, Virginia, U.S.A., September 1993

[SaKa96] Sailer, P.M.; Kaeli, D.R.: The DLX Instruction Set Architecture Handbook, Morgan Kaufmann Publishers Inc., San Francisco, USA, 1996

[Schi95] Schiller, J.: A Flexible Co-Processor for High-Performance Communication Support, IEEE Globecom'95, Singapore, November 1995

[Schi96] Schiller, J.: Teilautomatisierter Entwurf modularer Prozessorsysteme für die Hochleistungskommunikation, Fortschrittsberichte VDI, Reihe 10, Nr. 426, VDI-Verlag Düsseldorf, 1996

[ScZi95] Schiller, J.; Zitterbart, M.: Modular VLSI Implementation Architecture for High-Performance Communication Support, 7th IEEE Workshop on Local and Metropolitan Area Networks, Marathon, Florida, März 1995

[StGr94] Stock, T; Grünenfelder, R.: Frame Loss vs. Cell Loss in ATM Concentrators and Policing Units, Proceedings of 12th Annual Conference on European Fibre Optic Communications an Networks, Heidelberg, Juni 1994

[Stee94] Steenkiste, Peter A.: A Systematic Approach to Host Interface Design for High-Speed Networks, IEEE Computer, März 1994, S. 47-57

[Syno94] Synopsys Inc.: Dokumentation zu Simulator, Design Compiler, Design Analyzer, Version 3.0b, Synopsys Inc., Mountain View, USA, 1994

[Texa95] Texas Instruments Inc.: TMS320C80 Multimedia Video Processor: Technical Reference, Texas Instruments Inc., http://www.ti.com/

[Veri95] Verilog SA: Dokumentation zu Geode, Verilog SA, Toulouse, Frankreich

[Virt95] Virtual Computer Corporation: EVC1s technical reference, Virtual Computer Corporation, Reseda, U.S.A., 1995

[WoFD93] Worster, T.; Fischer, W.; Davis, S.P.: Resource Allocation for Packet Data Traffic on ATM: Problems and Solutions, in Gerner, N.; Hegering, H.-G.; Swoboda, J. (Hrsg.): Proceedings of „Kommunikation in Verteilten Systemen"; München, März 1993, Springer-Verlag, S. 100-113

[Xili94] Xilinx: The programmable logic data handbook, Xilinx, San Jose, U.S.A., 1994