Diplomarbeit

Ein echtzeitfähiger IP-Stack für die Myrinet-Architektur

Sven Reigl

22. August 2001

Inhaltsverzeichnis

Kapitel 1

Einleitung

Die rasante Entwicklung des Internets hat eine große Anzahl neuer Anwendung entstehen lassen. Bestärkt wird diese Entwicklung durch sehr leistungsfähige Netzwerke. Diese neuen Anwendungsgebiete stellen hohe Anforderung an die Qualität und die Sicherheit der Übertragung. Die Nutzerakzeptanz hängt dabei stark vom Preis-Leistungverhältnis und der Verfügbarkeit der Lösung ab. Der Netzwerkdienst soll Beschränkungen der Verzögerung und des Jitters zulassen und gleichzeitig ein gleichbleibenden Datendurchsatz und eine hohe Zuverlässigkeit gewährleisten. Die Haupteinsatzgebiete sind Multimedia-Anwendungen.

Auf dem Gebiet der zusagefähigen Netzwerke wurde bereits eine große Anzahl an Forschungen betrieben. Diese Dokument stellt eine kleine Auswahl dieser Arbeiten vor. Viele dieser Forschungsprojekte beschäftigten sich mit ausgewählten Netzwerken und liefern keine allgemein gültigen Verfahren zur Realisierung der Vorhersagbarkeit bzw. der Beschränkung von Leistungsparametern. Dies liegt zum Teil an der großen Anzahl der verfügbaren Netze. Diese unterscheiden sich in der Art des Mediumzugriffs, der möglichen Leistungsfähigkeit und der zulässigen Netztopologie.

Das Ziel dieser Arbeit liegt darin, ein Netzwerkprotokoll zu entwickeln, mit dem eine Zusagefähigkeit für möglichst alle Netzwerk erreicht wird. Weiterhin soll ein einheitlicher Zugriff von Anwendungen auf die Netzwerkschnittstellen möglich sein.

Der zweite Teil dieser Arbeit beschäftigt sich mit der Echtzeit-Fähigkeit der TCP/IP-Protokollfamilie. Der letzte Teil der Aufgabe bestand darin, das entwickelte Protokoll für das DROPS-Projekt (Dresdener Real-Time Operating System) zu programmieren. Hierzu wurde als Referenznetzwerk das Myrinet ausgewählt.

1.1 Über dieses Dokument

Im folgenden Kapitel wird eine Echtzeit-Betrachtung von Netzwerken vorgenommen. Es werden allgemeine Gesichtspunkte bzgl. vorhersagbare Kommunikationssysteme diskutiert. Ein großer Teil des Kapitels widmet sich der Analyse vorhandener vorhersagbarer Kommunikationsprotokolle. Im Kapitel 3 werden Entwurfsmöglichkeiten vorgestellt und diskutiert. Dem schließt sich ein Kapitel mit Implementierungsdetails an. Das Kapitel 5 enthält Ergeb-

nisse durchgeführter Messungen. Im abschließenden Kapitel erfolgt eine Zusammenfassung und eine Bewertung der Arbeit.

Dieses Dokument wendet sich an Leser mit fundiertem Wissen im Bereich der Informatik und speziell von Rechnernetzen. Es sind Kenntnisse über den Aufbau von Netzwerkarchitekturen und des OSI-Referenzmodells erforderlich. Hierzu kann das Buch [Tan92] von Tanenbaum empfohlen werden.

1.2 Erklärung

Hiermit erkläre ich, daß ich diese Arbeit selbstständig erstellt und keine anderen als die angegebenen Hilfsmittel verwendet habe.

Kapitel 2

Echtzeit-Betrachtung in Netzwerken

Große Netzwerke können eine beliebige Topologie aufweisen und bestehen meist aus einer Anzahl kleiner Netzwerke (Teilnetze). Diese Teilnetze können von unterschiedlichem Typ sein und werden durch Netzwerkknoten miteinander verbunden. Die Nachrichten können auf ihrem Weg von der Quelle zur Senke eine Anzahl von Knoten passieren. Die Informationen werden in einigen Knoten zwischengespeichert und anschließend zu einem benachbarten Knoten weitergesendet („Store-and-forward“). Die Verbindung zwischen benachbarten „Store-and-forward“-Knoten kann über Teilnetze erfolgen, welche die Daten nicht zwischenspeichern.

Mögliche Netzwerkknoten sind Switches, Hosts oder spezielle Kommunikationsknoten (Gateways bzw. Router). Hosts und Router werden durch Adapterkarten in die Teilnetze integriert und sind frei programmierbar. Auf ihnen können beliebige Protokolle zum Einsatz kommen, und es ist durch die verwendete Software möglich, eine zusagefähige Verarbeitung zu realisieren.

Switches sind speziell an das jeweilige Netzwerk angepaßt und bieten oftmals keine oder nur beschränkte Einflußmöglichkeiten durch den Netzwerkbetreiber. Ihre Arbeitsweise ist auf hohen Gesamtdurchsatz und auf geringen Ressourcenbedarf optimiert. Ein Echtzeitsystem benötigt Zusagen über den Nachrichtenaustausch zwischen Routern. Daher ist es notwendig die zwischenliegenden Teilnetze zu untersuchen.

Im folgenden Abschnitt werden verschiedene Switchingtechniken hinsichtlich ihrer Zusagefähigkeit betrachtet. In einem anderen Abschnitt werden Möglichkeiten zu Verbesserung der Vorhersagbarkeit von Kommunikationssystemen beschrieben. Der sich daran anschließende Abschnitt stellt vorhandene Echtzeit-Systeme vor. Im letzten Abschnitt dieses Kapitels wird die TCP/IP-Protokollfamilie auf ihre Zusagefähigkeit untersucht.

2.1 Switching-Techniken

Es gibt eine Vielzahl verschiedener Netzwerke, die sich nach ihrer Arbeitsweise gegliedert, in solche mit Punkt-zu-Punkt Verbindungen und Netzwerke mit Broadcast-Kanälen unterteilen lassen.

Um durch ein Teilnetz größere Strecken überbrücken zu können, Netzwerkfehler besser zu isolieren oder eine größere Anzahl von Knoten zu unterstützen, werden Switches eingesetzt. Diese und die verwendeten Switching-Techniken müssen von Echtzeit-Systemen beachtet werden, da es hier zu zusätzlichen Verzögerungen oder Engpässen kommen kann. Oft wird durch Switches die Topologie des Netzwerkes festgelegt.

In modernen Netzwerken sind drei Arten von Switchingtechniken gebräuchlich: geschaltetes und paketorientiertes Switchen sowie Switches mit virtuellen Kanälen. Es sind aber auch Netzwerke im Einsatz, die ohne Switches auskommen. Im Verlauf dieser Arbeit werden nur Tokenringnetzwerke betrachtet.

Im Bereich der Rechnernetze hat der Begriff Port mehrere Bedeutungen. In diesem Abschnitt ist mit einem Port eine feste, physische Verbindungsschnittstelle eines Gerätes gemeint. Desweiteren werden Abläufe immer aus der Sicht der jeweiligen Instanz erläutert. Folglich ist ein Empfangsport ein Port, über den Daten eintreffen, und ein Sendeport der, über den Daten versendet werden.

2.1.1 Geschaltetes Switchen

Bei dieser Switching-Technik wird für jeden Datentransfer eine feste Verbindung vom Sender zum Empfänger geschaltet. Diese Verbindung bleibt bis zu deren Abbau erhalten. Die Daten folgen in dieser Zeit einer festen Route durch das Netzwerk. Der Hauptanwendungsbereich dieser Technik ist das Telefonnetz.

Vorteile:

- Diese Technik garantiert eine ständig gleichbleibende Bandbreite. Sie wird durch die Kapazität der geschalteten Verbindung bestimmt.
- Es treten gleichbleibende und geringe Verzögerungszeiten auf.
- Durch diese Art von Netzwerken können sehr große Entfernungen überwunden werden. Hierzu sind sie hierarchisch organisiert.
- Durch die genannten Eigenschaften eignen sich diese Netzwerke gut für die zusagefähige Kommunikation.

Nachteile:

- Durch den expliziten Auf- bzw. Abbau der Verbindung entsteht ein Mehraufwand. Die Verbindungsaufbauzeiten können sehr groß sein, was das System unattraktiv für die Nicht-Echtzeit-Kommunikation macht.
- Das System ist nicht sehr flexibel, da die Bandbreiten oft durch das System festgelegt sind. Das Switching-Schema ist für sporadisches und lawinenartiges Botschaftenaufkommen ungeeignet.

- Wird ein Teil der reservierten Bandbreite nicht benötigt, geht diese verloren.
- Es ist immer nur eine Verbindung möglich. Werden mehrere Verbindungen benötigt, müssen mehrere physische Ports installiert werden. Aus diesem Grund wird diese Netzwerkart lediglich zur Verbindung von entfernten Teilnetzen eingesetzt.

2.1.2 Paketorientiertes Switchen

Die Daten werden mittels kleiner, voneinander unabhängiger Netzwerkübertragungseinheiten (Pakete) versendet. Jedes Paket wird von Switch zu Switch transferiert, bis es seinen Bestimmungsort erreicht hat. Während der Übertragung des Paketes steht dem Sender die gesamte Bandbreite des Mediums zur Verfügung.

Die paketorientierte Switching-Technik ist in Computernetzwerken am weitesten verbreitet. Aber auch bei der Übertragung von digitaler Sprache und von Rundfunk wird diese Technik zunehmend eingesetzt.

- **„Store-and-forward“ Switches:**

  Ein „Store-and-forward“ Switch kann eine kleine Anzahl von Paketen zwischenspeichern. Wenn ein Paket vollständig und korrekt empfangen wurde, wird ermittelt, ob der Port, auf den gesendet werden soll, frei ist. Ist dies der Fall, werden die Daten aus dem Puffer heraus versendet. Anderenfalls verbleiben die Daten im Puffer, bis der Sendeport frei ist.

  Kommt es bei der Übertragung eines Pakets zu einer Kollision, kann der Switch die Daten aus dem Puffer heraus erneut versenden. Erst wenn das Paket ohne Kollisionen übermittelt wurde, wird der Puffer für den Empfang anderer Paketen verwendet.

  Das Routen kann prinzipiell auf mehrere Arten erfolgen. Ein Switch kann intern eine Tabelle aufbauen, in der er vermerkt, über welchen Port eine Station Pakete versendet hat. Trifft ein Paket für einen bestimmten Empfänger ein, sendet der Switch das Paket auf den Port hinaus, von dem das letzte Paket des entsprechenden Host eintraf. Das Netzwerkprotokoll muß einen Mechanismus bereitstellen, um einen Port zu einem unbekannten Empfänger zu ermitteln.

  Es ist aber auch denkbar, daß ein „Store-and-forward“-Switch „source-routed“ arbeitet, d.h. der Sender gibt in jedem Paket, das er verschickt, die komplette Route durch das Netzwerk an.

  Vorteile:

  - Bei dieser Art des Switchens wird eine hohe Ressourceneffizenz erreicht, da potentiell sehr viele Pakete im Netzwerk unterwegs sein können. Ein Auf- bzw. Abbau von Verbindungen ist nicht notwendig.
  - Paketorientierte Netzwerke lassen sich sehr flexibel einsetzen. Es gibt keine Beschränkungen in der Topologie und der Verwendung. Ein Switch muß aber mit der Möglichkeit umgehen können, daß es mehrere Wege zum Empfänger gibt.

- Da „Store-and-forward“-Switches in der Lage sind, einige Pakete zu speichern, können sie gut mit lawinenartig auftretenden Nachrichten umgehen.
- Ein Switch kann selbständig auf Netzwerkfehler reagieren, indem er Pakete auf einem alternativen Weg zum Empfänger sendet.

Nachteile:

- Durch das Zwischenspeichern tritt eine relativ hohe Verzögerung der Pakete auf. Ein Zwischenspeichern erfolgt auch, wenn der Sendeport frei ist.
- Bei hoher Auslastung des Netzwerkes kann es zu einem Puffermangel im Switch kommen, und eintreffende Pakete können nicht gespeichert werden. Der Switch kann diese Pakete nur verwerfen. Es besteht somit keine Garantie über die Auslieferung eines Paketes.
- Es sind zusätzliche Mechanismen nötig, die ein unendliches „Kreisen“ von Paketen verhindern.
- Da in einem „Store-and-forward“-Switch viele aufwendige Prozesse ablaufen, ist die Hardware ebenso aufwendig und oft teuer.
- Um möglichst viele Pakete zu speichern und die Kosten nicht übermäßig ansteigen zu lassen, können im Netzwerk nur relativ kleine Pakete verwendet werden.
- In reiner Form sind „Store-and-forward“-Switches nur schlecht für Echtzeit-Kommunikation geeignet.

- **„Cut-through“-Switches:**

  Bei dieser Art des Switchens werden die Pakete nicht zwischengespeichert. Die Elektronik ermittelt, ob der Sendeport frei ist und legt die Daten direkt auf diesen Port. Ist der Port belegt wird dem Sender signalisiert, den Nachrichtentransfer einzustellen. Der Switch speichert nur die bis dahin aufgelaufenen Daten. Spezielle Mechanismen verhindern ein unendlich langes Blockieren von Ports. „Cut-through“-Switches arbeiten im Allgemeinen „source-routed“.

  Da die ankommenden Daten direkt weitergeleitet werden, kann ein Switch nicht auf Übertragungsfehler reagieren. Erst der Empfänger des Paketes entscheidet, wie mit diesem zu verfahren ist.

  Vorteile:

  - „Cut-through“-Switches besitzen kleinere und besser vorhersagbare Verzögerungszeiten als „Store-and-forward“-Switches.
  - Der Vorgang des Switchens ist relativ einfach. Die Anforderungen an die Hardware sind gering, so daß die Kosten für einen Switch geringer ausfallen als für einen „Store-and-Forward“-Switch.
  - Es bestehen kaum Einschränkungen bzgl. der maximalen Paketgröße.

Nachteile:

- Diese Art von Switches kann weniger gut mit lawinenartigem Nachrichtenaufkommen umgehen.
- Durch das Blockieren von Ports kann Bandbreite verloren gehen, da andere Pakete, die den Port benutzen müssen, nun ebenfalls „warten“ müssen.
- In reiner Form sind „Cut-through“-Switches nicht für die Echtzeit-Kommunikation geeignet. Sie bieten aber besser vorhersagbare Mechanismen als „Store-and-forward“-Switches. Es werden keine Pakete verworfen, und nicht blockierte Pakete haben eine geringe Verzögerungszeit.

2.1.3 Switches mit virtuellen Kanälen

In modernen Punkt-zu-Punkt Netzwerken sind über eine physische Verbindung mehrere virtuelle Kanäle möglich. Dies wird durch Zeitmultiplexing und -demultiplexing erreicht. Diese Switching-Schemen werden den Anforderungen von Anwendungen besser gerecht, da mehrere Kanäle gleichzeitig eine physische Verbindung nutzen können. Die virtuellen Verbindungen besitzen bestimmte Leistungseigenschaften, die von den Switches berücksichtigt werden. Vor dem Senden müssen die Kanäle aufgebaut werden.

Es werden im wesentlichen zwei Techniken verwendet: „Wormhole“- und virtuelles „Cut-through“-Switching. Es sind aber auch hybride Switching-Techniken bekannt [SD95], auf die hier nicht weiter eingegangen werden soll. Ist der Sendeport nicht belegt, arbeiten beide Verfahren nach dem „Cut-through“-Schema. Sie unterscheiden sich im Umgang mit belegten Sendeports.

- **Virtuelles „Cut-through“:**

  Ist der Sendeport belegt, wird bei virtuellem „Cut-through“ das Paket im Switch gespeichert („store-and-forward“).

  Vorteile:

  - Gegenüber „Store-and-forward“-Switches verringert sich die Latenzzeit der einzelnen Pakete bei einer großen verfügbaren Bandbreite.
  - Bei einer hoher Netzwerkauslastung ist eine sehr gute Ressourceneffizienz möglich, auch wenn nur wenige virtuelle Kanäle verwendet werden.
  - Das Verfahren ist gut für Echtzeit-Kommunikation einsetzbar, da geringe Latenzzeiten und eine Bandbreitenreservierung möglich sind.

  Nachteile:

  - Das Switching-Schema stellt sehr hohe Anforderung an die Hardware. Es fallen sehr hohe Kosten an.
  - Es wird eine große Anzahl von Puffern benötigt.

- **Switchen mittels „Wormhole“:**

  Bei einem belegten Sendeport wird der virtuelle Kanal, über den das Paket empfangen wurde, blockiert. Der Sender des Paketes wird aufgefordert, den Datenfluß einzustellen.

  Vorteile:

  – Gegenüber „Store-and-forward“-Switches verringert sich die Latenzzeit der einzelnen Pakete bei einer großen verfügbaren Bandbreite.

  – Es werden nur wenige Puffer benötigt.

  – Das Verfahren ist gut für Echtzeit-Kommunikation einsetzbar, da geringe Latenzzeiten und eine Bandbreitenreservierung möglich sind.

  Nachteile:

  – Das Switching-Schema stellt sehr hohe Anforderung an die Hardware. Es fallen sehr hohe Kosten an.

  – Es ist eine weniger gute Ressourcenauslastung bei hoher Netzwerkauslastung und geringer Anzahl virtuellen Kanälen zu erwarten. Dies kann zum Teil durch eine Erhöhung der Anzahl von virtuellen Kanälen ausgeglichen werden.

2.1.4 Tokenringnetze

In diesen Netzwerken sind die beteiligten Hosts und Gateways zu einem Ring organisiert. Der Datenfluß erfolgt in eine festgelegte Richtung. Um Kollisionen von Paketen zu vermeiden, darf immer nur eine Station zu einem Zeitpunkt senden. Das Senden wird durch Kontrollpakete (Token) gesteuert. Alle Pakete werden von Station zu Station weitergeleitet und sind folglich für alle Netzteilnehmer sichtbar.

Empfängt eine Station ein Kontrollpaket und ist sendebereit, nimmt sie dieses vom Netz und sendet stattdessen ein Datenpaket. Das Datenpaket umkreist den Ring und der Empfänger kann die Daten lesen. Er entfernt die Nachricht jedoch nicht. Stattdessen setzt er in dem Paket zur Empfangsbestätigung ein Flag. Erst der Sender entfernt das Paket und generiert eine neue Steuernachricht.

Eine detaillierte Beschreibung von Token-Ring Netzwerken kann z.B. in [Tan92] nachgeschlagen werden.

Vorteile:

- Da jede Station nur ein Paket sendet und anschließend das Übertragungsmedium freigibt, werden alle Stationen fair behandelt und niemand kann das Medium unendlich lange besetzen.

- Es ist möglich Prioritäten festzulegen. Es ist aber nun möglich Stationen „verhungern“ zu lassen.

- Für Stationen der höchsten Priorität ergibt sich ein deterministischer und vorhersagbarer Mediumzugriff. Daher sind solche Netzwerke gut für Echtzeit-Systeme geeignet.

Nachteile:

- Diese Art von Netzwerken benötigt eine zentrale Instanz, die über Fehlerzustände entscheidet, duplizierte Steuerpakete entfernt oder verlorengegangene erneuert.
- Der Ring kann durch ausfallende Stationen unterbrochen werden. Es gibt aber Verfahren, um diese Fehlerzustände zu beseitigen.
- Wird der Ring durch die Beschädigung einer Verbindungsleitung unterbrochen, kann der Netztransfer oft nicht aufrecht erhalten werden.

2.2 Echtzeit-Betrachtung von Kommunikationssystemen

Nachdem einige Aspekte des Nachrichtenaustausches in Netzwerken betrachtet wurden, sollen in diesem Abschnitt die Aufgaben von Systemen diskutiert werden, die auf diese Netze aufbauen. Im Folgenden werden nur paketorientierte Netzwerke betrachtet, da diese Form der Datenübertragung zwischen Rechnern am verbreitetesten ist.

Ein Teilabschnitt stellt allgemeingültige Methoden zur Verbesserung der Vorhersagbarkeit vor. Ein weiterer Teilabschnitt beschäftigt sich mit wichtigen Begrifflichkeiten in Kommunikationssystemen und deren Einfluß in Echtzeit-Systemen. Gleichzeitig sollen Möglichkeiten zur Verbesserung der beschriebenen Systemeigenschaften erörtert werden.

2.2.1 Annahmen und Beschränkungen

Echtzeit-Anwendungen kennen ihren Bedarf an Betriebsmitteln, der sich aus den erwarteten Leistungseigenschaften ergibt. Das System kann anhand der bestehenden Zusagen und noch vorhandenen Ressourcen ermitteln, ob es neue Verbindlichkeiten eingehen kann. Nachdem Zusagen getroffen wurden, hat das System dafür zu sorgen, daß diese eingehalten werden. Die ausgehandelten Leistungsparameter stellen aber nur eine obere Schranke dar, Unterschreitungen sind zulässig.

Beschränkung der Paketgröße

Viele Netzwerke schreiben einen festen Aufbau bzw. eine feste Paketgröße (MTU - Maximum Transmission Unit) vor, z.B. ATM und DQDB. In einem solchen Netzwerk ist es keinem Knoten möglich, die vereinbarte maximale Paketgröße zu überschreiten. Andere Netzen besitzen keine oder wenige Beschränkungen. Insbesondere im Myrinet sind sehr große Paket erlaubt. Ein langes Belegen des Übertragungsmediums führt zu schlechten Antwort- und hohen Verzögerungszeiten. Durch das verwendete Protokoll sollte eine

Beschränkung der Paketgröße erfolgen. Große Nachrichten müssen vor dem Versenden in mehrere kleine Pakete aufgeteilt werden (Fragmentierung).

Andererseits ist bekannt, daß der höchste Gesamtdurchsatz bei großen Paketen erreicht wird. Das Verhältnis der Anzahl der Nutzdaten zu der Anzahl an Daten, die für das Übertragungsprotokoll benötigt werden, ist günstiger. Zusätzlich ist der Mehraufwand durch das Übertragungsprotokoll geringer. Es muß ein Kompromiß zwischen geringer Antwortzeit und hohem Durchsatz gefunden werden.

Wohlverhalten aller Kommunikationspartner

Die meisten Netzwerke, insbesondere lokale Netzwerke (LAN), bieten keine Mechanismen zur Bandbreitenreservierung oder zur Beschränkung des Verkehrsaufkommen an. Einem nicht kooperativen Teilnehmer ist es in solchen Netzwerken zu jeder Zeit möglich, beliebige Daten zu senden. Nicht eingeplantes Datenaufkommen kann den Datentransfer des gesamten Netzes stören.

Ein Netzwerkteilnehmer kann in einer nicht zusagefähigen Umgebung den Datenaustausch zwischen Rechnern beeinflussen. Überflutet er einen Host mit Paketen, muß der Protokollstapel des Empfängers diese bearbeiten, auch wenn er sie verwirft. Durch sinnlose Serviceanfragen an die Serveranwendungen eines Hosts kann er einen Teil der Betriebsmittel binden und somit die Servicefähigkeit es Systems herabsetzen („Denial of service“-Angriff).

Da Fehler bzw. Angriffe ohne Hardwareunterstützung nicht zu vermeiden sind, muß angenommen werden, daß sich alle Hosts an die ausgehandelten Serviceeigenschaften halten und niemand versucht den Nachrichtenaustausch zu stören. Dies kann dadurch erreicht werden, daß alle Rechner im Netzwerk mit dem selben Echtzeit-Protokoll kommunizieren. Jeder einzelne Rechner setzt die Einhaltung der Serviceeigenschaften der Anwendungen durch. Es darf keiner Anwendung möglich sein, unüberwacht auf das Übertragungsmedium zuzugreifen.

2.2.2 Anforderungen an echtzeitfähige Kommunikationssysteme

Bandbreite

Das wohl wichtigste Kriterium an die Qualität einer Verbindung ist eine ständig gleichbleibende zur Verfügung stehende Bandbreite. Der Datenfluß wird in einen Strom von Paketen umgewandelt, d.h. es kann von einer festen bzw. maximalen Anzahl von Paketen je Zeiteinheit ausgegangen werden. Diese Eigenschaft läßt sich sehr gut durch ein Kontrollprogramm überwachen.

Paketverlustrate

Zusagefähige Systeme können nicht verhindern, daß Pakete auf dem Weg durch das Netzwerk verfälscht werden. Verfälschungen kommen durch physische Einflüsse auf das Übertragungsmedium zu Stande. Die Häufigkeit von Verfälschungen hängt von dem verwendeten

Medium ab. Eine Verbesserung der Situation kann durch die Verwendung von fehlerkorrigierenden Kodes erzielt werden. Aber auch mit diesen ist nur eine gewisse Anzahl von Fehlern zu korrigieren. Da Netzwerkfehler relativ selten sind und oft sporadisch auftreten, werden fehlerkorrigierende Kodes so gut wie nie verwendet.

Paketverluste treten meist bei Pufferüberläufen auf. Diese Fehlerart wird durch ein echtzeitfähiges Systeme ausgeschlossen, auch wenn das System stark ausgelastet ist.

Periode

Echtzeit-Prozesse besitzen einen festen Zeitpunkt („deadline“), bis zu dem sie eine Aufgabe erfüllen müssen. Viele Echtzeitsysteme arbeiten periodisch, d.h. Prozesse werden in festen Zeitabständen vom System aktiviert und konsumieren in einer Periode ein Quantum an Rechenzeit. Aperiodische Echtzeitsystem werden im weiteren Verlauf dieses Dokumentes nicht betrachtet. Wenn diese eine Beschränkung der Kommunikation gestatten, können die im Verlauf dieses Dokumentes vorgestellten Kommunikationsverfahren verwendet werden.

Bei der Echtzeit-Kommunikation gibt die Periode an, in welchen Zeitabständen eine Anwendung bereit ist, Daten zu senden oder zu empfangen. Das System muß ausreichend Datenpuffer zur Verfügung stellen, um alle ankommenden Daten solange zu speichern bis die Anwendung vom Scheduler aktiviert wurde und die Daten verarbeitet hat.

Verzögerungszeit

Der Datenfluß unterliegt einer ständigen Verzögerung, z.B. der Signallaufzeit eines Übertragungsmediums oder der Verarbeitungszeit einer aktiven Netzwerkkomponente. Um Audio- oder Videoströme in gleichmäßig hoher Qualität darstellen zu können, muß die Verzögerung der Datenpakete gleichmäßig erfolgen.

Die Gesamtverzögerung darf in vielen Anwendungsfällen nicht zu groß sein. Vor allem in Audio- und Videokonferenzen oder bei Telefonverbindungen stört ein solches Verhalten. Echtzeit-Systeme sollten eine Beschränkung der maximale Verzögerungszeit zulassen. Das System kann durch die Wahl der Route durch das Netzwerk Einfluß auf diese Größe nehmen.

Die Gesamtverzögerung setzt sich aus den Verzögerungen durch die zwischenliegenden Routern zusammen: $D = D_1 + D_2 + D_3 ....$ Wobei D_i die Verarbeitungszeit des Paketes durch den Router i darstellt. Wird jedem dieser Router eine maximale Verzögerung zugewiesen, beträgt die höchstmögliche Gesamtverzögerung eines Echtzeit-Kanals: $D_{max} = D_{max1} + D_{max2} + D_{max3} ...$(Abb. 2.1). Die maximale Verzögerung eines Pakets in einem Router kann z.B. eine „deadline“ sein, die der Scheduler des Routers einhalten muß. Für die Verzögerung durch das Teilnetzwerk zwischen den Routern wird der „worst case“ angenommen.

Jitter und minimale Verzögerung

In einem idealen System treffen die Datenpakete zu äquidistanten Zeitpunkten beim Empfänger ein. Dies kann in einem realen System nicht gewährleistet werden. Die Schwankungen zwischen der vereinbarten Ankunftsrate und dem tatsächlichen Eintreffen der Da-

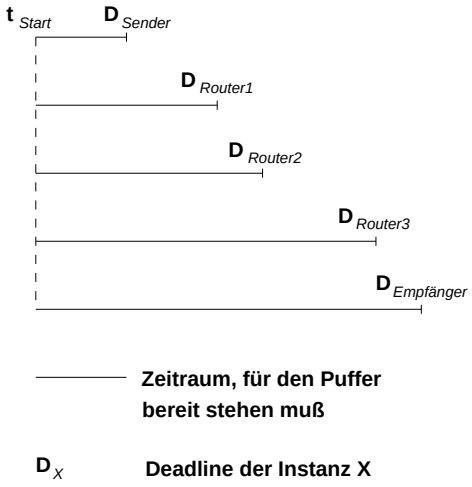

Abbildung 2.1: *Beschränkung der maximalen Verzögerung eines Paketes*

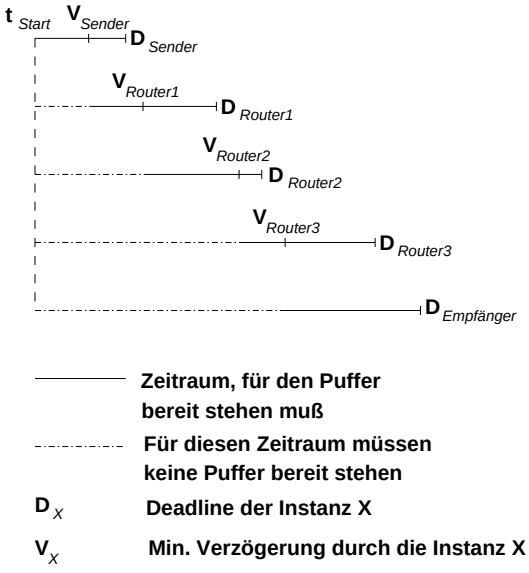

Abbildung 2.2: *Einführung einer minimalen Verzögerung eines Paketes*

tenpakete wird Jitter genannt. Der Jitter kann beim Empfänger durch Zwischenspeichern ausgeglichen werden. Hierzu sind zusätzliche Puffer nötig. In Routern kann die benötigte Speicherkapazität schnell die vorhandene übersteigen. Folglich sollte der Jitter so gering wie möglich gehalten werden. Dies kann durch die Einführung einer minimale Verzögerung [VZF91] von Paketen erreicht werden. Hierzu hält ein Router ankommende Pakete so lange zurück, bis die minimale Verzögerungszeit erreicht ist (Abb. 2.2).

2.3 Vorhandene Echtzeit-Systeme

Es existiert eine ganze Reihe von Arbeiten, die sich mit der Zusagefähigkeit von Netzwerken beschäftigen, da Multimedia- und Echtzeit-Anwendung und moderne Hochgeschwindigkeitsnetzwerke ein enormes Potential an neuen Möglichkeit bieten.

Ein Teil der vorhandenen Systeme erreicht die Zusagefähigkeit auf Bitübertragungsschicht und der andere Teil auf Netzwerkschicht. Vertreter aus der ersten Gruppe werden in den folgenden zwei Unterabschnitten vorgestellt. Die Vertreter der anderen Gruppe bilden den Abschluß dieses Abschnittes.

2.3.1 Echtzeit über Ethernet mittels eines virtuellen Tokenring

Die Verwendung eines virtuellen Tokenring zur Verwirklichung von Echtzeit-Fähigkeit ist ein relativ häufig verwendetes Schema. Tokenringnetzwerke sind wohlverstanden und erprobt. Es sind Bandbreitenreservierungen möglich, und es kann eine maximale Verzöge-

rungszeit berechnet werden. Stellvertretend soll hier eine Arbeit vorgestellt werden, die mittels dieses Schemas eine Zusagefähigkeit über Ethernet realisiert [VC]. Die Arbeitsweise des Tokenrings[1] wurde bereits unter Abschnitt 2.1.4 erläutert. An dieser Stelle wird nur auf Unterschiede und Besonderheiten eingegangen.

Das Ethernet ist auf Grund seines Mediumzugriffsprotokoll nur sehr schlecht für die Echtzeit-Kommunikation geeignet. Bei der Übertragung eines Paketes kann es zu Kollisionen kommen, die von allen Stationen des Netzwerkes beobachtet werden können. Nach einer Kollision warten die sendebereiten Stationen einen zufällig gewählten Zeitraum und senden das Paket erneut. Dabei können erneut Kollisionen auftreten. Bei einem ausgelasteten Netzwerk treten sehr viele Kollisionen auf. Der Gesamtdurchsatz nimmt stark ab.

Im Bereich der lokalen Netzwerke nimmt das Ethernet eine führende Rolle ein. Dies liegt zum großen Teil an den geringen Kosten für die Adapterkarten und das Übertragungsmedium. Die weite Verbreitung von Ethernet läßt die Notwendigkeit von zusagefähigen Mediumzugriffsprotokollen über Ethernet aufkommen.

RETHER - Real-Time Ethernet

Im Grundzustand befinden sich die Hosts im üblichen CSMA-Modus (Carrier Sense Multiple Access) des Ethernets. Möchte ein Klient in Echtzeit kommunizieren, werden alle Stationen in den RETHER-Modus umgeschaltet. In diesem Modus gibt es zwei Arten von Token, eines für die Nicht-Echtzeit- und eines für die Echtzeit-Kommunikation. Jeder Host kennt das komplette Netzwerk, die anderen Station und deren Bandbreitenreservierungen. Er kann anhand dieser Informationen selbständig entscheiden, ob eine ausreichende Bandbreite zur Verfügung steht, um eine neue Echtzeit-Verbindung zu etablieren.

Eine Echtzeit-Reservierung erfolgt mittels zweier Parameter, der maximalen Tokenumlaufzeit (MTRT - Maximum Token Rotation Time) und der maximalen Tokenhaltezeit (MTHT - Maximum Token Holding Time). Letzterer Parameter ist für jede Station spezifisch. Jeder Netzwerkteilnehmer besitzt nur eine Haltezeit. Diese kann von ihm geändert werden. Die Haltezeit wird auf alle Anwendungen des Hosts verteilt, die in Echtzeit kommunizieren wollen.

Eine Station des Netzwerkes wird zum Koordinator ernannt. Diese generiert und wartet die Token. Sie versendet zuerst ihre eigenen Echtzeit-Daten. Ist dieser Vorgang abgeschlossen, generiert sie ein Echtzeit-Token und sendet diesen um den Ring. Hierbei werden nur die Stationen mit einer Tokenhaltezeit berücksichtigt. Hat das Token den Ring umlaufen und ist noch ausreichend Zeit bis zur MTRT, wird diese für die Nicht-Echtzeit-Botschaften verwendet. Der Koordinator erzeugt hierzu ein Nicht-Echtzeit-Token und sendet dieses um den Ring. Hierbei werden alle Stationen des Netzwerkes berücksichtigt. Möchte eine Station eine neue Echtzeit-Verbindung etablieren, wartet sie auf das Nicht-Echtzeit-Token und setzt alle Stationen über ihren neue MTHT in Kenntnis.

Der Ausfall einer Station wird dadurch erkannt, daß das Token nach der maximalen Haltezeit der Station nicht weitergereicht wurde. Möchte eine Station in das System auf-

[1] Eine detaillierte Beschreibung von Tokenringnetzwerken kann in [Tan92] nachschlagen werden.

genommen werden, wartet sie, bis das Übertragungsmedium unbenutzt ist, und sendet ein spezielles Datenpaket.

Nachteile

- Es ist keine Beschränkung der Verzögerung und des Jitters möglich. Eine Beschränkung erfolgt nur durch die maximale Tokenumlaufzeit.
- Das RETHER beschränkt sich momentan auf ein Ethernetsegment.
- Bei der Anmeldung eines Knotens im System kann es zu Kollisionen kommen.
- Bei diesem Verfahren wird ein Teil der Botschaften mittels Broadcast versendet. Diese Art des Botschaftenaustausches steht nicht in allen Netzwerken zur Verfügung.

2.3.2 Statistische Echtzeit über Ethernet

Als ein zweites Verfahren wurde ein statistisches Echtzeit-Modell für das Ethernet [KSZ] ausgewählt. Es basiert auf der Analyse des verwendeten 1-persistent CSMA/CD.[2] Hierzu wird angenommen, daß es sich bei dem CSMA/CD um einen Semi-Markov Prozeß handelt.

Statistische Echtzeit bedeutet: die Wahrscheinlichkeit, daß ein Paket auf dem Weg von der Quelle zur Senke verloren geht und die Wahrscheinlichkeit, daß die ausgehandelte Verzögerungszeit des Paketes nicht eingehalten wird, ist kleiner als eine definierte Toleranz. Die Verzögerungszeit eines Paketes hängt davon ab, wie oft es zu Kollisionen mit anderen Paketen kam, d.h. die Anzahl der Versuche zu senden. Folglich muß das System statistisch sicherstellen, daß die Anzahl der Versuche kleiner oder gleich einer maximalen Anzahl von Sendeversuchen bleibt.

Um dieses Verhalten zu erzeugen, wird die Paketankunftsrate im Netzwerk unter einem Schwellwert gehalten. Dieser Wert wird „network-wide input limit“ genannt. Jedem Netzwerkteilnehmer wird ein Teil der gesamten Netzwerkpaketrate zugewiesen. Dieser Wert wird „station input limit“ genannt und kann dynamisch vergrößert bzw. verkleinert werden. Das Ankunftsverhalten einer Station wird als Poissonprozeß betrachtet. Die Berechnung des „network-wide input limit“ erfolgt anhand der Prozeßmodelle und der Anzahl Knoten. Der Wert ist jedem Rechner bekannt.

Um möglichst wenige Änderungen an bestehenden Protokollen vorzunehmen und trotzdem das Ankunftsverhalten der Pakete zu beeinflussen, wird zwischen der TCP/IP- und der Ethernetschicht eine zusätzliche Protokollschicht eingefügt. Diese reguliert das Nachrichtenaufkommen der Klientanwendungen, so daß der Paketausstoß unter dem „station input limit“ bleibt. Diese Glättung des Paketstromes ist besonders bei lawinenartigem Nachrichtenaufkommen notwendig, da in diesem Zusammenhang häufig Kollisionen auftreten.

[2] p-persistentes CSMA bedeutet, daß eine Station vor dem Senden das Übertragungsmedium überprüft. Ist das Medium frei sendet die Station mit der Wahrscheinlichkeit p. Im anderen Fall wartet sie eine Zeiteinheit lang.

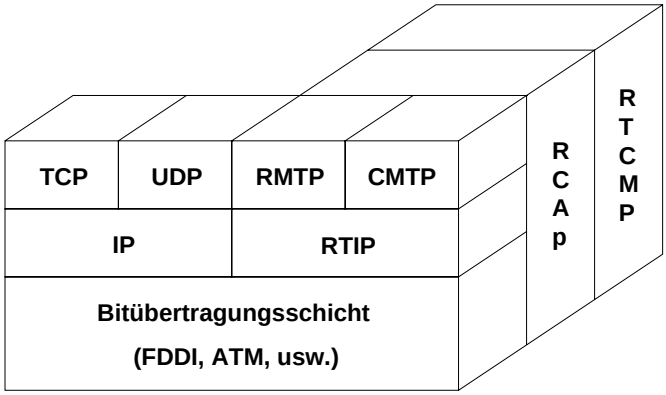

Abbildung 2.3: *Die Tenet Echtzeit-Protokollfamilie*

Vorteile:

- Es sind nur sehr geringe Änderungen an den bestehenden Protokollstapeln notwendig.

Nachteile:

- Das Verfahren erlaubt keine absolute Vorhersagbarkeit bzgl. der Übertragungszeiten von Datenpaketen.
- Es kann nur eine feste Anzahl von Netzwerkteilnehmern geben. Jeder Knoten muß das gesamte Netz kennen.

2.3.3 Tenet und Vorarbeiten

Das Tenet-Projekt [BFMM94] basiert auf dem Modell der Echtzeit-Kanäle [FV90]. Dieses Modell hat eine weite Verbreitung gefunden und wurde im Laufe der Zeit erweitert und verbessert ([MIS96] und [VZF91]).

Die Architektur der Tenet-Protokollsuite ist in der Abbildung 2.3 skizziert und stellt einen eigenen Protokollstapel neben dem TCP/IP dar. Es können beide Protokollsuites gleichzeitig verwendet werden. Zum Erstellungszeitpunkt der Arbeit [BFMM94] waren die Teilprotokolle RTCMP (Real-Time Control Message Protocol) und CMTP (Continues Media Transport Protocol) noch nicht fertig gestellt. Daher wird im folgenden auch nicht näher darauf eingegangen.

Echtzeit-Kanäle

Ein Echtzeit-Kanal [FV90] ist eine virtuelle Verbindung zwischen einem Sender und einem Empfänger. Die Daten werden mittels voneinander unabhängiger Pakete versendet (Datagram Semantik). Der Datenfluß unterliegt bestimmten Leistungseigenschaften und folgt einer festen Route durch das Netzwerk. Die Nachrichtenreihenfolge bleibt auf dem Weg

von der Quelle zur Senke erhalten. Es wird angenommen, daß Netzwerkfehler sehr selten und Verzögerungen bei der Netzübertragung klein und bekannt sind. Da der gesamte Datenaustausch kontinuierlich und zeitbezogen erfolgt, ist eine Wiederherstellung gesendeter Daten nicht möglich. Der Transfer ist verbindungsorientiert, erfolgt aber ungesichert, d.h. beim Auftreten eines Netzwerkfehlers erfolgt keinerlei Signalisierung und folglich auch kein erneuter Datentransfer.

RCAP - Real-Time Channel Admission Protocol

Das RCAP stellt Servicefunktionen für die Signalisierung und Überwachung von Echtzeit-Kanälen bereit. Die Hauptaufgaben sind der Auf- bzw. Abbau von Verbindungen, es sind aber auch Funktionen zur Wartung von Echtzeit-Kanälen enthalten.

Beim Aufbau einer Verbindung wird der Bedarf an Ressourcen im Sendersystem berechnet und mit den noch zur Verfügung stehenden Betriebsmitteln verglichen. Kann der Bedarf gedeckt werden, darf der Kanal erstellt werden. Der Senderechner schickt zum Verbindungsaufbau ein Paket zu einem Router seiner Wahl. Dieser ermittelt seinerseits, ob er den Ressourcenbedarf befriedigen kann, ohne bestehende Zusagen zu verändern. Die nötigen Betriebsmittel werden angefordert und für den Kanal bereitgestellt. Der Router ermittelt den nächsten Host und sendet diesem das Paket usw. Als letzter in der Reihe ermittelt der Empfänger, ob er die benötigten Betriebsmittel bereitstellen kann und sendet eine Bestätigungsbotschaft. Das Antwortpaket nimmt die gleiche Route durch das Netz, wie das Paket für den Verbindungsaufbau, nur in umgekehrter Reihenfolge.

Kann eine Instanz den Anforderungen nicht entsprechen, sendet sie eine Botschaft mit einer Ablehnung zum Initiator. Da dieses auf der Route erfolgt, auf der das Paket zum Verbindungsaufbau eintraf, können alle zwischenliegende Instanzen die reservierten Betriebsmittel freigeben.

Ein Verbindungsabbau kann zu jeder Zeit sowohl von Empfänger als auch vom Sender ausgelöst werden. Das Paket folgt der festen Route. Jeder zwischenliegende Router gibt die Betriebsmittel, die er für den Kanal reserviert hat, frei.

RTIP - Real-Time Internet Protocol

Das Real-Time Internet Protocol ist die Netzwerkschicht der Tenet-Protokollsuite. Die Hauptaufgabe liegt in der echtzeitfähigen Versendung von Datenpaketen, so daß die getroffenen Vereinbarungen eingehalten werden. Im Gegensatz zum Internet Protokoll (IP) arbeitet das RTIP verbindungsorientiert. Das RTIP stellt einen ungesicherten, unidirektionalen Paketübermittlungsdienst bereit. Da die Pakete einem festen Weg durch das Netzwerk folgen, bleibt die Paketreihenfolge erhalten.

RMTP - Real-Time Message Transportprotocol

Das RMTP ist für den Botschaftenaustausch mittels des Sende-Empfangsparadigmas konzipiert worden. Das CMTP hingegen orientiert sich an periodischer und zeitgetriebender Kommunikation. Das Real-Time Message Transportprotocol stellt ein ungesichertes und

nachrichtenbasiertes Transportprotokoll dar. Wird eine gesicherte Übertragung benötigt, muß diese durch höherliegende Protokolle realisiert werden.

Die Hauptaufgabe des RMTP ist das Aufteilen von Nachrichten in eine Sequenz von Paketen, die auf der Empfängerseite zu der Originalbotschaft zusammensetzen werden. Auf diese Art und Weise können Anwendungen größere Nachrichten versenden, als es das RTIP zuläßt.

Nachteile des Systems:

- Das Modell des Echtzeit-Kanals berücksichtigt bzw. erlaubt keine Multicast-Verbindungen. Dies kann sich für verschiedene Anwendungen als ungünstig erweisen, z.B. Videokonferenzen. Anwendungen mit einer sehr großen Anzahl von Teilnehmern sind ohne Multi- bzw. Broadcast kaum vorstellbar, z.B. Fernseh- und Rundfunkübertragung im Internet.

  Die Multicast-Semantik kann durch den Aufbau mehrerer Kanäle nachgebildet werden. Hierfür sind erheblich mehr Ressourcen nötig.

- Leider wird in keiner der verwendeten Arbeiten beschrieben, wie eine Reservierung des Mediums erfolgen kann. [MIS96] und [VZF91] beschränken sich auf ein hypothetisches Netzwerk und verwenden Simulationen zum Korrektheitsnachweis. Im Tenetsystem ([BFMM94]) wird ein FDDI-Netzwerk verwendet, auf dem eine Bandbreitenreservierung relativ einfach erfolgen kann.

- Ein Verbindungsabbau kann nur durch den Sender oder dem Empfänger geschehen.

2.3.4 RSVP - Resource Reservation Protocol

Das RSVP ([BKMS]) ist ein von der Internet Engineering Task Force (IETF) entwickeltes Protokoll. Die momentan verwendete Servicearchitektur im Internet arbeitet klassenlos. Sie ist auf eine möglichst hohe Leistungsfähigkeit bei einem geringen Betriebsmittelbedarf optimiert. Die neu zu entwerfende Architektur soll mehrere Klassen von Diensten unterstützen, die unterschiedliche Qualitätsanforderungen an ein System stellen.

Trotz der neuen Dienste sollen existierende TCP/IP-Protokollstapel und Socketschnittstellen weiter verwendet werden können. Anwendungen, die keine besonderen Servicequalitäten beanspruchen, bleiben ohne Änderungen lauffähig. Für den Daten- und Kontrollfluß werden verschiedene Datenpfade verwendet, welche dieselbe Route durch das Netz nehmen. Die Kontrollinformationen werden mittels asynchroner Botschaften übermittelt. Es ist möglich, bestehende Reservierungen zu ändern und für einen schon bestehenden Nachrichtenstrom nachträglich eine Dienstgüte zu vereinbaren.

Zur Realisierung der Dienstqualität wird ein QoS-Manager (quality of service) verwendet. Er reserviert und überprüft die für die Kommunikation notwendigen Betriebsmittel und führt die Zulassungskontrolle für neue Qualitätszusagen durch. Hierzu muß er mit dem Netzwerktreiber, der Zugriffsschnittstelle zum Netzwerk und dem Speichermanagment des Betriebssystems zusammenarbeiten.

Der Aufbau einer Verbindung

Zum Aufbau einer Verbindung mit einer festen Dienstgüte werden zwei Nachrichtentypen verwendet. Eine „PATH“-Botschaft wird vom Initiator ausgesendet. Sie enthält die geforderten Leistungsmerkmale. Empfängt ein Router das Paket, vermerkt er die Parameter in einem sogenannten „Soft state“. Anschließend versendet er die Nachricht zum nächsten Knoten. Erhält der Empfänger das Paket und ist er bereit die Verbindung aufzubauen, sendet er ein „RESV“-Botschaft zu Sender. Diese Botschaft folgt der gleichen Route durch das Netz wie das „PATH“-Paket. Die dazwischen liegenden Router fordern die Betriebsmittel an und reservieren diese für die Verbindung. Schlägt eine Reservierung fehl, wird eine „RESV ERROR“-Nachricht generiert und zum Empfänger gesendet. Die Router, die auf diesem Weg liegen, nehmen die schon getroffenen Verbindlichkeiten zurück. Der Empfänger signalisiert dem Initiator, daß ein zusagefähige Verbindung nicht möglich ist.

Eine Reservierung kann auch für Multicastverbindungen durchgeführt werden. In diesem Fall werden „PATH“-Botschaften zu alle Teilnehmern der Gruppe gesendet. Es ist möglich, für jeden Teilnehmer der Gruppen unterschiedliche Leistungparameter festzulegen. Weiterhin können zu jeder Zeit Teilnehmer die Multicastgruppe verlassen oder zu dieser Gruppe hinzugefügt werden.

Zur eindeutigen Identifizierung einer Verbindungen wird ein Fünf-Tupel verwendet: (Protokoll, Quelladresse, Quellport, Zieladresse, Zielport).

Es soll an dieser Stelle ausdrücklich darauf hingewiesen werden, daß das RSVP nur ein Hilfsmittel zur Erstellung der „Soft state“ ist und keine Umsetzungsdetails festlegt. Die Umsetzung kann prinzipiell für jedes System anders realisiert werden.

Serviceklassen

Die Betriebsmittelbereitstellung und die nötigen Mechanismen zur Verwaltung des Netzwerktransfers sind stark von der geforderten Dienstgüte abhängig. Im Standardisierungsprozeß[3] werden zwei wichtige Dienstklassen festgelegt: garantierte Dienste und solche mit einer überwachten Übertragungsleistung.

Zu der ersten Gruppe zählen Anwendungen, die Zusagen über die zu erwartende Paketverlustrate und -verzögerung benötigen, wie Audio- und Videoübertragung. In der zweiten Gruppe sind Anwendungen zu finden, die sich damit zufrieden geben, wenn ein sehr hoher Prozentsatz der Pakete erfolgreich eintrifft und die gewünschte Verzögerung aufweist.

Nachteile

- Zur Berechnung der benötigten Betriebsmittel werden lediglich der Pufferspeicher und die Bandbreitenauslastung der Netzwerkschnittstelle berücksichtigt. Weiterhin sind die meisten heute eingesetzten Betriebssystem nicht zusagefähig, so daß eine wirkliche Echtzeit-Fähigkeit nicht möglich ist.

[3]Zum Erstellungszeitpunkt des Dokuments. Es ist möglich, andere Klassen festzulegen.

- Die Router lehnen einen Verbindungsaufbau erst bei der „RESV“-Nachricht ab. Dies hätte eventuell schon bei der „PATH“-Botschaft geschehen können.

2.4 Internet-Protokolle

Für die Kommunikation im Internet hat das TCP/IP Protokoll die weiteste Verbreitung gefunden. Es hat sich praktisch zum Standard für den Botschaftenaustausch zwischen Rechnern etabliert. Beim TCP/IP handelt es sich um eine Protokollfamilie bestehend aus dem IP-, TCP-, UDP- und einer Reihe anderer Protokolle, die hier nicht weiter betrachtet werden. Detaillierte Beschreibungen der beteiligten Protokolle sind in [Tan92] und [Ste94] nachzulesen. Weiterhin soll auf die RFCs verwiesen werden, die die Protokolle beschreiben oder Teilaspekte behandeln.

Mittels Sockets und der darunterliegenden Transportprotokolle ist es relativ einfach, Klientanwendungen zu programmieren. Diese Einfachheit soll auch für Echtzeit-Anwendungen zur Verfügung stehen. Leider bieten TCP, UDP und das darunterliegende IP keine oder nur geringe Funktionalität bzgl. der Echtzeit-Fähigkeit.

In den folgenden Unterabschnitten sollen die drei Protokolle kurz vorgestellt werden. Es werden nur die Funktion beschrieben, ohne in die Tiefe zu gehen. Gleichzeitig wird eine Einschätzung bzgl. der Vorhersagbarkeit vorgenommen.

2.4.1 Das IP - RFC 791

Das Internetprotokoll stellt einen verbindungslosen und ungesicherten Botschaftenvermittlungdienst bereit (Datagram Semantik). Verbindungslos bedeutet, daß alle Datagramme unabhängig voneinander behandelt werden. Die Datagramme können in einer anderen Reihenfolge bei einem Empfänger eintreffen, als sie versendet wurden.

Soll eine Nachricht übermittelt werden, wählt die Netzwerkschicht, falls keine direkte Verbindung besteht, einen Router aus, dem sie die Botschaft zusendet. Hierzu ermittelt das IP die zugehörige Netzwerkschnittstelle. Bevor sie dieser die Nachricht übergibt, kann es nötig sein, die Daten in kleinere Stücke aufzuteilen und als getrennte Datagramme zu übergeben. Die IP-Schicht des Empfängers setzt die Teile (Fragmente) zur Originalbotschaft zusammen.

Ein Router ist ein Netzwerkknoten, der mit mindestens zwei Teilnetzen verbunden ist. Er kann Pakete von einem Teilnetz empfangen und an ein anderes weiterleiten. Hierzu kann es nötig sein eingetroffene Datagramme noch weiter zu fragmentieren. Ein Router benötigt für jedes Teilnetz eine eigene IP-Adresse.

Das Internetprotokoll besitzt in seinem Protokollkopf zwar ein Feld für den Servicetyp, der gewisse Eigenschaften vom IP erwartet. Es hängt aber sehr stark von der jeweiligen Implementation ab, ob dieses Feld überhaupt beachtet wird. Aufgrund seiner verbindungslosen Arbeitsweise, läßt sich unter gewissen Annahmen ein obere Schranke für die Bearbeitungszeit ermitteln. Die notwendigen Annahmen sind eine genügend große Anzahl

von Puffern und eine ausreichende Rechenleistung. Gleichzeitig muß die Ankunftsrate von Paketen bekannt sein.

2.4.2 Das TCP - RFC 793

Das TCP Protokoll dient dem gesicherten Datenaustausch zwischen Anwendungen. Hierzu wird eine feste Verbindung zwischen den beteiligten Rechnern aufgebaut, über welche die Klienten kommunizieren können. Das Protokoll erzeugt aus dem Eingangsdatenstrom Sequenzen von Botschaften. Zusätzlich werden Nachrichten zur Steuerung der bestehenden Verbindung generiert und versendet. TCP garantiert die Datenintegrität und -reihenfolge, so daß sich die Anwendungen um diese nicht kümmern müssen. Verlorengegangene oder durch Übertragungsfehler verfälschte Pakete werden erkannt und erneut versendet.

In einem streng zeitgesteuerten System macht es keinen Sinn verfälschte bzw. verlorene Datenpakete erneut zu versenden. Diese Daten haben oftmals keinen „Wert“ mehr, sie beeinträchtigen den Datenaustausch anderer Verbindungen und sind wegen ihres sporadischen Auftretens schlecht oder gar nicht einplanbar. Ein zusagefähiges TCP ist wahrscheinlich nicht möglich und wird im weiterem Verlauf nicht betrachtet.

2.4.3 Das UDP - RFC 768

Im Gegensatz zu TCP stellt UDP ein einfaches, datagramorietiertes und ungesichertes Transportprotokoll dar. Es arbeitet nicht stromorientiert und bietet den Anwendungen nur wenig Komfort. Die Datagramme werden lediglich in ein IP-Datagram eingekapselt.

Ein Echtzeit-Datenaustausch mittels der TCP/IP Protokollfamilie ist nur mit dem UDP sinnvoll. Um die nötigen Ressourcen auf allen Rechnern auf dem Weg zum Empfänger bereitzustellen, muß im Gegensatz zum gewöhnlichen UDP eine Verbindung aufgebaut werden. Erst wenn eine solche Verbindung etabliert wurde, darf der Sender Daten senden.

Kapitel 3

Entwurf

In diesem Kapitel werden mögliche Modelle für ein Echtzeit-System vorgestellt. Dabei wird sich auf das Schema der Echtzeit-Kanäle (Abschnitt 2.3.3) gestützt. Es beschreibt sehr gut auf welche Art und Weise zusagefähige Kommunikation stattfinden soll und welche Leistungseigenschaften zu optimieren sind. Das Schema ist relative unabhängig von dem konkreten Netzwerk. Leider erweist sich das Tenet ([BFMM94]) als nicht universell einsetzbar, so daß die Suche nach einer anderen Lösung begründet ist.

Am erfolgversprechendsten sind Systeme, welche die Echtzeit auf der Bitübertragungsschicht realisieren. Auf dieser erfolgt der Mediumzugriff, folglich kann nur hier eine Reservierung der Bandbreite geschehen. Durch diese Schicht kann auch die Beschränkung der Verzögerung und des Jitters erfolgen. Die Mediumzugriffsschicht kann von potentiell unsicheren Anwendungen zugegriffen werden, so daß es notwendig sein kann, die Betriebsmittelreservierung und -überwachung auf dieser Schicht zu verwirklichen.

In einem weiteren Abschnitt werden Möglichkeiten beschrieben, das IP und das UDP zusagefähig zu machen.

3.1 Zusagefähige Mediumzugriffsverfahren

Für Echtzeit-Systeme ist die Vorhersagbarkeit der Kommunikation entscheidend. Es muß zu jedem Zeitpunkt feststehen, wie hoch das Nachrichtenaufkommen im Netzwerk ist, wer zu welchem Zeitpunkt senden darf und welche Route ein Paket durch das Netzwerk folgt.

Ein Problem stellt die Notwendigkeit dar, gleichzeitig Nicht-Echtzeit- und Echtzeit-Kommunikation zuzulassen. Nicht-Echtzeit-Kommunikation tritt sporadisch und oft lawinenartig auf. Die von der Echtzeit-Kommunikation nicht verwendete Bandbreite soll fair an alle anderen Anwendungen verteilt werden. Trotzdem soll das System dem burstartigem Auftreten des Nicht-Echtzeit-Nachrichtenaufkommens Rechnung tragen. Diese Entscheidungen können dezentral oder zentral gelöst werden.

In diesem Abschnitt werden mögliche Modelle für die Echtzeit-Kommunikation aufgezeigt und bzgl. der Einhaltung der Zielestellungen und zuvor beschriebenen Kriterien untersucht.

Bewertungskriterien von Systemen:

Leistungsfähigkeit: Durch die Leistungsfähigkeit eines Systems wird ausgedrückt, wieviel der vorhandenen Bandbreite eines Netzwerkes verwendet werden kann. Wichtige Kriterien sind der notwendige Mehraufwand durch das verwendete Echtzeit-Protokoll und die Möglichkeit des parallelen Versendens von Datenpaketen.

Skalierbarkeit: Die Skalierbarkeit beschreibt die Veränderung des Systemverhaltens bei steigender Anzahl von Netzwerkteilnehmern. Ein Gesichtspunkt stellt das dynamische Erweitern bzw. Verkleinern des Netzwerkes dar.

Ausfallsicherheit und Fehlertoleranz: Wie verhält sich das System beim Ausfall eines Knotens? Was passiert bei Netzwerkpartitionierung? Kann bei Netzwerkfehlern die Echtzeit-Fähigkeit des Systems aufrecht erhalten werden? Werden beim Ausfall eines Rechners vorhandene Echtzeit-Kanäle abgebaut und reservierte Betriebsmittel freigegeben?

3.1.1 Dezentrale Netzzugriffsentscheidung mittels „worst case“

In weiten Bereichen eines Computersystems ist der genaue Ablauf von Ereignissen bzw. der genaue innere Zustand nicht vorhersagbar. Es kann aber fast immer eine obere Schranke angegeben werden, bis zu der ein Betriebsmittel benutzt wird. Bei der Einplanung der Betriebsmittelnutzung wird diese obere Schranke verwendet. Das Netzwerk kann als eine weitere Ressource betrachtet werden.

Im nächsten Schritt muß ermittelt werden, was der „worst case“ ist. Bei Netzwerken ist das der Fall, wenn sämtliche Kommunikation über ein und die selbe physische Verbindung, zur gleichen Zeit und in die gleiche Richtung erfolgen soll. An dieser Stelle sollte erwähnt werden, daß dieses Schema bei kollisionsbehafteten Netzwerken nicht funktionieren kann.

Jeder Rechner erhält eine gewisse Bandbreite, die zu seiner freien Verfügung steht und die er nach Belieben verwendet. Er hat dafür zu sorgen, daß dieser Wert nicht überschritten wird.

Variante 1:

Im einfachsten Fall wird die Bandbreite B, die einem Rechner zur Verfügung steht, auf $B = \frac{1}{(n-1)}$ beschränkt. n ist die maximale Anzahl von Rechnern im Netz.

Vorteile:

- Das Verfahren arbeitet dezentral und ist sehr einfach.
- Jeder Host kann sofort entscheiden, ob eine Echtzeit-Kommunikation möglich ist oder nicht.
- Das Schema ist fehlertolerant gegenüber ausfallenden oder hinzukommenden Netzwerkknoten.

Nachteile:

- Eine extrem schlechte Leistungsfähigkeit, da der „worst case“ so gut wie nie eintreffen wird. Die unbenutzte Bandbreite eines Hosts geht verloren. Nachrichten, die sich nicht behindern, werden in diesem System nicht betrachtet und können keine höhere Bandbreite erhalten. Sind weniger Rechner im System als durch den Maximalwert festgelegt, bleibt auch diese Bandbreite ungenutzt.

- Ein solches System skaliert sehr schlecht oder gar nicht, da es eine feste maximale Anzahl von Rechnern im System gibt. Wird diese Anzahl heraufgesetzt, wird gleichzeitig die zur Verfügung stehende Bandbreite eines Hosts herabgesetzt.

- Das Schema ist nicht auf kollisionsbehaftete Netzwerke anwendbar.

- Ein solches System erlaubt keine Beschränkung der Verzögerungzeiten. Es kommt zu einem hohem Jitter.

Variante 2:

Da das Grundkonzept nur auf Punkt-zu-Punkt Netzwerke anwendbar ist, können unter bestimmten Annahmen Verbesserungen erzielt werden. Durch einen Switch und den unmittelbar verbundenen Rechnern wird ein Teilnetz aufgespannt. Wird nun angenommen, daß der größte Teil der Kommunikation in diesem Teilnetz stattfindet, kann der „worst case“ auf das Nachrichtenaufkommen innerhalb des Teilnetz optimiert werden. Dazu wird die Kommunikation nach außerhalb des Teilnetzes benachteiligt. Die zur Verfügung stehende Bandbreite eines Host für den Botschaftenaustausch im Teilnetz beträgt: $B = \frac{1}{(n-1)}$. Hierbei ist n die Anzahl der Ports[1], die der Switch besitzt. Möchte ein Host mit einem Rechner außerhalb des Teilnetzes kommunizieren, sinkt die verfügbare Bandbreite um eine Potenz je dazwischen liegenden Switch, d.h. $B = \frac{1}{(n_1-1)} * \frac{1}{(n_2-1)} * \frac{1}{(n_3-1)}...$, n_i ist die Anzahl der Ports des Switches i auf der Route zum Empfänger. Ist n_i für alle Switches konstant, beträgt $B = (\frac{1}{(n-1)})^z$ wobei z die Anzahl der zu überwindenden Switches darstellt.

Verbesserungen:

- Die mögliche Leistungsfähigkeit des Systems steigt erheblich an. War im Ausgangssystem ein Datenaufkommen von höchstens der Bandbreite einer physischen Verbindung möglich, beträgt sie im verbessertem System das j-fache dieser Bandbreite. j stellt die Anzahl der Switches im Netzwerk dar.

- Das System skaliert gleichmäßig und vorhersagbar.

[1] Mit einem Port ist hier eine physische Verbindungsstelle eines Gerätes gemeint.

Neue Nachteile:

- Für Kommunikation über mehrere Switches hinweg steht nur noch eine eingeschränkte Bandbreite zur Verfügung.
- Eine leicht erhöhter Aufwand zu Berechnung und Verwaltung der Bandbreiten.
- Es hängt stark vom Verwendungszweck ab, ob die getroffenen Annahmen gut zu den jeweiligen Erfordernissen passen.

3.1.2 Verwaltung der Eingangsbotschaften durch die Klienten

Jeder Knoten kennt die Anzahl der Verbindungen zu ihm und kann die verwendete Bandbreite ermitteln. Er entscheidet selbständig ob weitere Verbindungen zu ihm aufgebaut werden können.

Möchte ein Klient eine neue Echtzeit-Verbindung etablieren, ermitteln die Kommunikationssysteme der beteiligten Hosts, ob hinreichend Netzwerk-, Puffer- und Rechenkapazität vorhanden sind und akzeptiert die neue Verbindung oder lehnt diese ab.

Fehlersituationen sind durch das Ausbleiben von Botschaften zu erkennen. Der Host kann hierzu Timer setzen und nach Ablauf dieser alle getroffenen Verbindlichkeiten zurücknehmen.

Vorteile:

- Das Verfahren arbeitet dezentral.
- Es sind eine hohe Leistungsfähigkeit und Skalierbarkeit möglich, solange es zu keinen Überlastsituationen innerhalb des Netzwerkes kommt.
- Das System bietet eine hohe Fehlertoleranz und Ausfallsicherheit.

Nachteile:

- Bei diesem Verfahren hat niemand Informationen über das Verhalten im Netzwerk. Es kann nur angenommen werden, daß keine Überlastsituationen auftreten.
- Da bei diesem Schema kein vorhersagbarer und geordneter Mediumzugriff erfolgt, sind kollisionbehaftete Netzwerke ausgeschlossen.
- In einem solchen System wird die Nicht-Echtzeit-Kommunikation nicht berücksichtigt. Es lassen sich nur schwer Mechanismen vorstellen, die Nicht-Echtzeit-Kommunikation beachtet und nicht gleichzeitig auf einem völlig anderen System beruhen.
- Es ist nicht vorhersagbar, wie stark ein Paket durch das Netzwerk verzögert wird. Es tritt ein hoher Jitter auf.

3.1.3 Verteilung sämtlicher Informationen im Netzwerk

Der entgegengesetzte Ansatz zum vorherigen Schema ist, alle Informationen eines Netzwerkknotens jedem anderen zugänglich zu machen[2]. Soll ein neuer Echtzeit-Kanal aufgebaut werden, überprüft der Klient anhand der verteilten Informationen, ob dies möglich ist. Ist das der Fall, sendet er an jeden Rechner im Netzwerk die Echtzeit-Eigenschaft der neuen Verbindung. Er kann der Anwendung erst seine Bestätigung geben, wenn alle Netzwerkknoten den Eigenschaften zugestimmt haben. Dies kann in großen Netzwerken sehr lange dauern. Es ergeben sich dadurch keine Probleme, da der Auf- bzw. Abbau von Echtzeit-Verbindungen keinen Echtzeit-Eigenschaften genügen muß.

Probleme ergeben sich erst, wenn mehrere Klienten versuchen Verbindungen aufzubauen und die aufsummierten Eigenschaften nicht mit den verfügbaren Ressourcen zu vereinbaren sind. In diesem Fall stimmen wenigstens die jeweiligen „gegnerischen“ Parteien, den Parametern nicht zu. Jede Partei muß die schon versendeten Echtzeit-Eigenschaften zurücknehmen. Zur Schlichtung des Konflikts können die „konkurrierenden“ Parteien versuchen, den Konflikt zu lösen, z.B. durch die Wahl einer anderen Route.

Ein weiteres Problem stellt der Umgang mit Nicht-Echtzeit-Botschaften dar. Diese müssen ebenfalls im System angemeldet werden. Um den Protokollmehraufwand nicht zu stark ansteigen zu lassen, wird nicht jedes Paket, sondern vielmehr eine gewisse Anzahl von Paketen, angemeldet. Werden Ströme mittels Nicht-Echtzeit-Botschaften versendet, muß zyklisch mit jedem Host eine neue Entscheidung getroffen werden.

Um neue Netzwerkteilnehmer aufzunehmen, müssen sämtliche Rechner des Netzwerkes bekannt sein. Von Zeit zu Zeit werden den nicht aktiven Knoten Steuerpakete zugesandt. Das Ausscheiden eines Hosts ist einfach. Wird der Rechner ordnungsgemäß heruntergefahren, meldet sich das Kommunikationssystem ab. Ist er durch einen Fehler ausgefallen, treten im Laufe der Zeit „timeouts“ auf.

Vorteile:

- Auch dieses Schema kommt ohne eine zentrale Instanz aus.
- Es kann zu keiner Überlastung im Netzwerk kommen.
- Die Fehlertoleranz bleibt erhalten.

Nachteile:

- Es sind sehr viele Kontroll- und Statusnachrichten notwendig. Stehen Multicast bzw. Broadcast zu Verfügung, läßt sich der Aufwand verringern. Ansonsten sind sehr viele Einzelbotschaften nötig.
- Für den Nicht-Echtzeit-Datentransfer ist der Verwaltungsaufwand noch erheblich höher. Prinzipiell muß jedes Datenpaket angemeldet werden.
- Das System ist groß und die Verwaltung sehr aufwendig. Es gibt zahlreiche innere Zustände.

[2] Ähnlich der in den Teilabschnitten 2.3.1 und 2.3.2 beschriebenen Systeme

3.1.4 Virtueller Tokenring

Wie bereits erwähnt sind virtuelle Tokenringe ein häufig verwendeter Ansatz, nicht echtzeitfähige Netzwerke zusagefähig zu machen. Das unter Abschnitt 2.3.1 vorgestellte System verwendet für seine Arbeit Broadcast-Nachrichten, diese Art zu kommunizieren steht nicht unter allen Netzwerken zur Verfügung. In diesem Teilabschnitt werden nur Verbesserungsvorschläge diskutiert. Hierbei wird sich auf die Arbeitsweise des Tokenringnetzwerkes bezogen. Zuvor werden die Vor- und Nachteile eines virtuellen Tokenrings beschrieben.

Vorteile:

- Da alle Nachrichten den Ring umlaufen, ist auch in Punkt-zu-Punkt Netzwerken ein Nachrichtenaustausch mittels Broadcast möglich.
- Das Verfahren ist einfach und erprobt.
- Die Entscheidungen über das Senden kann eine Station autonom treffen.

Nachteile:

- Es wird eine zentrale Instanz benötigt, die über Fehlerzustände entscheidet.
- Das System skaliert schlecht.
- Das Verfahren besitzt eine schlechte Fehlertoleranz. Der Ring kann durch ausfallende Stationen unterbrochen werden. Es ist aufwendig, neue Knoten zu erkennen und in der Ring zu integrieren. Das kann dazu führen, daß getroffene Verbindlichkeiten nicht eingehalten werden können.
- Die Leistungsfähigkeit ist gering. Die meisten Netzwerkkarten sind dazu ausgelegt, Pakete vom Übertragungsmedium zu empfangen und diese in einem Puffer zu speichern. Sie können eintreffende Pakete nicht direkt verändern. Da die Pakete den gesamten Ring umlaufen müssen, werden diese sehr stark verzögert. Aufgrund der deterministischen Arbeitsweise bleibt der Jitter klein.

Verbesserungsmöglichkeiten:

- Um den Datenaustausch zu beschleunigen, kann eine Station, die ein Token erhalten hat, die Daten auf direktem Wege zum Empfänger senden. Nur Pakete, die von globalem Interesse sind, umlaufen den Ring. Im Besonderen sind dies die Bekanntmachung von Echtzeit-Eigenschaften und die Token.

  Verbesserungen:

  - Es ergibt sich eine signifikante Verbesserung der Leistungsfähigkeit des Systems.
  - Die Verzögerungszeit von Datenpaketen wird deutlich herabgesetzt.
  - Das System skaliert besser als das originale Schema.

Verschlechterungen:

- Leicht erhöhter Aufwand für den einzelnen Netzwerkknoten.
- Jede Station muß die Routen zu allen Stationen kennen.

- Da die Pakete in jedem Host zwischengespeichert werden, kann die Leistungsfähigkeit durch zusätzliche Token erhöht werden. Das kann so weit getrieben werden, bis für jede Verbindung von einer Station zu ihrem Nachfolger ein Token vorhanden ist.

  Verbesserungen:

  - Bei einer hohen Anzahl von Token kann die Leistungsfähigkeit sehr stark ansteigen.
  - Das verbesserte Schema skaliert besser.

  Verschlechterungen:

  - Es ist ein erhöhter Aufwand vonnöten.
  - Es wird aufwendiger Fehlerzustände zu erkennen. Die beteiligten Stationen müssen synchronisiert werden.
  - Kann es im Netzwerk zu Kollisionen von Paketen kommen, ist das Verfahren nicht anwendbar.

3.1.5 Netzwerkscheduler

In diesem Modell gibt es gleichberechtigte Netzwerkknoten und einen Netzwerkscheduler. Jeder Host besitzt einen Serverprozeß, der die Hardware verwaltet und den Zugriff auf diese überwacht. Dieser Prozeß ist für den Auf- bzw. Abbau der Echtzeit-Kanäle verantwortlich. Kommen Anfragen zum Auf- oder Abbau einer Verbindung aus dem Netzwerk, nimmt er diese entgegen, ermittelt den Ressourcenbedarf bzw. gibt die entsprechenden Ressourcen frei und leitet die Anfragen an die jeweilige Anwendung weiter.

Der Netzwerkscheduler entscheidet, welcher Knoten zu welchem Zeitpunkt das Übertragungsmedium verwenden darf. Hierzu sendet er Kontrollnachrichten, die die Hosts darüber informieren, wieviele Datenpakete für welchen Kanal verwendet werden dürfen. Nicht verwendete Bandbreite wird an den Scheduler zurückgegeben. Es können mehrere Echtzeit-Datenpakete gleichzeitig im Netz unterwegs sein, wenn sich deren Wege nicht kreuzen.

Die Bandbreite, die nicht für Echtzeit-Daten verwendet wurde, wird auf alle Rechner im Netz aufgeteilt. Jeder Rechner kann diese Bandbreite beliebig benutzen oder, wenn er sie nicht benötigt, an den Scheduler zurückgeben. Da für den Nicht-Echtzeit-Datentransfer die Routen nicht bekannt sind, kann immer nur ein Rechner das Netzwerk benutzen.

3.1.5.1 Aufbau eines Echtzeit-Kanals

Möchte ein Klient einen neuen Echtzeit-Kanal etablieren, wendet er sich mit seinen Echtzeit-Anforderungen an den Serverprozeß. Dieser ermittelt den Ressourcenbedarf und informiert seinerseits den Netzwerkscheduler.

Der Netzwerkscheduler hat alle notwendigen Informationen über das Netzwerk. Er berechnet mit Hilfe dieser Informationen, den Eigenschaften der schon vorhandenen Echtzeit-Kanälen und den neuen Anforderungen einen Schedule, der den Serviceeigenschaften der Anwendungen genügt. Ist dies nicht möglich wird die Anfrage zurückgewiesen. Der Scheduler entscheidet, auf welcher Route die Daten gesendet werden sollen und sendet diese im Antwortpaket mit. In den folgenden Kontrollpaketen wird sich immer auf die berechnete Route bezogen.

Im nächsten Schritt wendet sich der Serverprozeß mit den Serviceparametern an seinen Kollegen auf der Empfängerseite. Diese prüft wiederum die benötigten Ressourcen und setzt sich mit der jeweiligen Anwendung in Verbindung. Tritt während dieser Prozedur ein Fehler auf oder kann eine Instanz die Serviceparameter nicht gewährleisten, muß der Serverprozeß des Senders den Abbau schon bestehender Verbindlichkeiten veranlassen.

3.1.5.2 Abbau eines Echtzeit-Kanals

Der Abbau einer Echtzeit-Verbindung kann von beiden Seiten erfolgen. Hierzu sollte der Initiator der Operation zuerst seinen Partner hiervon in Kenntnis setzen. Alle weiteren Schritte erfolgen durch den Initiator. Er wendet sich an den Serverprozeß auf der Empfängerseite und dann an den Netzwerkscheduler. Jede Instanz gibt die reservierten Betriebsmittel der Verbindung frei, die für neue Verbindungen zur Verfügung stehen.

3.1.5.3 Dynamisches Aufnehmen bzw. Ausschließen von Netzwerkknoten

Hierzu sind mehrere Wege denkbar.

- Im ersten Ansatz kennt jeder Host eine gültige Route zum Netzwerkscheduler. Wird der Rechner gestartet, setzt sich der Serverprozeß mit dem Netzwerkscheduler in Verbindung und meldet sich somit bei diesem an. Der Scheduler berechnet einen neuen Schedule, der den hinzugekommenen Knoten für den Nicht-Echtzeit-Datenaustausch berücksichtigt.

  Kehren von einem Host in einem gewissen Zeitraum keine Antwortnachrichten ein, wird angenommen, daß der Rechner ausgefallen ist. Der Netzwerkscheduler kann die Partner von Echtzeit-Kanälen mit diesem Rechner darüber informieren, so daß die zugehörigen Betriebsmittel freigeben werden können. Der Scheduler entfernt die betroffenen Kanäle aus seinen internen Strukturen und berechnet einen neuen Schedule.

  Vorteile:

  - Jeder Rechner kann sich autonom in das System einbringen.

Nachteile:

- Dieses Nachrichtenaufkommen ist nicht eingeplant und es kann zu Kollisionen bzw. Behinderungen mit Echtzeit-Nachrichten kommen.
- Der Netzwerkscheduler muß immer auf der selben Route erreichbar sein bzw. der Host muß immer auf anderem Wege über dessen Route informiert werden.

- Um den zweiten Nachteil des vorherigen Ansatzes zu beseitigen, kann der Host mittels eines geeigneten Verfahrens versuchen, eine Route zum Netzwerkscheduler zu finden. Hierzu wählt er eine beliebige, möglichst kurze Route und sendet auf dieser ein spezielles Paket.

  Trifft dieses Paket auf einen Knoten, der den Weg zum Scheduler kennt, sendet dieser ein Antwortpaket mit der Route zurück. Nun kann der Host eine Route zum Scheduler berechnen und setzt sich mit diesem in Verbindung. Trifft bis zu einem definierten Zeit keine Antwort ein, versucht der Host es erneut. Nach mehreren erfolglosen Versuchen wählt er eine neue Route.

  Vorteile:

  - Jeder Rechner kann sich autonom in das System einbringen.
  - Die Route zum Netzwerscheduler kann beliebig sein.

  Nachteile:

  - Die Nachrichten sind nicht eingeplant und können mit anderen Nachrichten kollidieren bzw. es kann zu Behinderungen von Echtzeit-Nachrichten kommen. Da es potentiell sehr viele Versuche geben kann, sind starke Beeinträchtigungen möglich.
  - Der Aufwand des einzelnen Hosts steigt an.

- Ein anderer Ansatz ist, die Auffindung neuer Rechner durch den Netzwerkscheduler zu veranlassen. Wenn er alle Rechner des Netzwerkes und eine gültige Route zu diesen kennt, braucht er diesen nur periodisch Pakete zusenden. Erhält er eine Antwort von einem noch nicht angemeldeten Rechner, wird dieser im nächsten Schedule berücksichtigt.

  Vorteile:

  - Die zusätzlichen Nachrichten sind einplanbar. Es werden keine anderen Nachrichten beeinträchtigt. Dies bedeutet allerdings einen Mehraufwand für den Netzwerkscheduler.
  - Die Route zum Netzwerscheduler kann beliebig sein.

Nachteile:

- Es wird eine zentrale Instanz benötigt.
- Es werden sehr viele sinnlose bzw. unbeantwortete Anfragen benötigt. In großen Systemen kann dadurch sehr viel Bandbreite verloren gehen.
- Die gesamte Netzwerktopologie muß dem Netzwerkscheduler bekannt sein.

- Um eine völlig dynamische Netzwerktopologie zuzulassen, kann der Netzwerkscheduler das vom Mapper des GM-Protokoll verwendete Verfahren einsetzen, um das Netzwerk zu erkunden (siehe [MYR99] und [Rei2000] Abschnitt. 2.3.2).

  Vorteile:

  - Die zusätzlichen Nachrichten sind einplanbar. Es werden keine anderen Nachrichten beeinträchtigt. Dies bedeutet einen erhöhten Aufwand für den Netzwerkscheduler.
  - Die Route zum Netzwerscheduler kann beliebig sein.

  Nachteile:

  - Es wird eine zentrale Instanz benötigt.
  - Das erstellen der Topologiekarte kann sehr aufwendig sein.

3.1.5.4 Vorteile des Modells

- Das vorgestellte Modell zeichnet sich durch seine universelle Einsetzbarkeit aus. Es stellt nur sehr geringe Anforderungen an einen einzelnen Rechnerknoten. Dieser muß lediglich seine Ressourcen verwalten und hat dafür zu sorgen, daß sich alle Klientenanwendungen an die vereinbarten Bedingungen halten. Die eigentliche Arbeit bzgl. der Entscheidung, wer wann senden darf, wird durch den Netzwerkscheduler erledigt.

- Zum Auf- und Abbau von Echtzeit-Kanälen, zur Abstimmung der Hosts, wann gesendet werden darf, usw. werden nur wenige Kontroll- bzw. Steuernachrichten benötigt. Häufig sind diese sehr kurz.

- Das Modell erlaubt auf den speziellen Systemen Optimierungen.

3.1.5.5 Nachteile und mögliche Verbesserungen

1. Der Netzwerkscheduler stellt einen „single-point-of-failure“ dar. Fällt er aus, ist das gesamte System nicht mehr arbeitsfähig. Alle getroffenen Verbindlichkeiten sind nichtig.

2. Das Modell erlaubt nur eine geringe Bandbreitenauslastung bei der Nicht-Echtzeit-Kommunikation.

Lösungsmöglichkeiten:

Um die Ausfallsicherheit des Systems zu erhöhen, kann ein Netzwerkscheduler potentiell auf jedem Rechner gehalten werden. Günstig ist es, hierfür einen eigenen Thread oder Prozeß zu erzeugen. Es muß sichergestellt werden, daß nur ein Netzwerkscheduler aktiv ist. Fällt der aktive Scheduler aus, kann ein anderer aktiv werden. Hierzu ist es notwendig, alle Statusinformationen zwischen den Netzwerkschedulern auszutauschen. Zusätzlich muß die Arbeit des aktiven Schedulers überwacht und ein Protokoll entwickelt werden, über welches ein neuer Netzwerkscheduler zum aktiven Scheduler ernannt wird („Election"-Verfahren). Dieses Verfahren hilf nur, wenn der aktive Netzwerkscheduler ausfällt, nicht aber, wenn er einen internen Fehler enthält und Echtzeit-Zusagen nicht mehr beachtet.

Da der aktive Scheduler den einzelnen Rechner belastet, kann dieser Rechner nur noch begrenzt kommunizieren. Wenn jeder Host die notwendigen Informationen hält, kann der aktive Netzwerkrechner, wenn es notwendig werden sollte, auf einen anderen Knoten migrieren. Diese Erweiterung bedeutet einen weiteren Mehraufwand im Protokoll.

3.2 Zusagefähiges IP und UDP

In diesem Abschnitt werden Möglichkeiten diskutiert, das IP und das UDP für das DROPS-System in eine zusagefähige Form zu bringen. Zuvor wird auf Probleme mit dem Modell der Echtzeitkanäle aufmerksam gemacht.

3.2.1 Probleme von IP, UDP und dem Modell der Echtzeit-Kanäle

Um die Kommunikation mittels IP und UDP auch über die Grenzen eines Teilnetzes zu gewährleisten, dürfen keine Änderungen an Protokollen vorgenommen werden. Ein Problem ergibt sich dadurch, daß das Modell der Echtzeit-Kanäle verbindungsorientiert arbeitet und dies beim IP und beim UDP nicht der Fall ist. Es bestehen keine direkten Möglichkeiten mittels dieser Protokolle einen Echtzeit-Kanal zu etablieren.

Erweiterung der Protokolle

Das IP und das UDP werden um die fehlenden Funktionalitäten erweitert.

Nachteile:

- Ein Datenaustausch ist nur zwischen Hosts möglich, die diese Erweiterungen kennen und unterstützen. Sollen Botschaften zu Knoten gesendet werden, die diese Erweiterungen nicht kennen, muß auf einem zwischenliegenden Router eine Art „Proxy"-Service geben, der den Strom von zusagefähigen Datenpaketen in einen nicht-zusagefähigen wandelt.
- Die Erweiterungen können nicht transparent erfolgen, d.h. Klientanwendungen müssen an die neue Protokollschnittstelle angepaßt werden.

Einführung eines Systemservices zur Erstellung eines Echtzeit-Kanals

Um eine Änderung der bestehenden Protokolle zu umgehen, kann eine neue Servicefunktion eingeführt werden. Beim Aufruf des Services erfolgt der Kanalaufbau. Ist dies erfolgreich geschehen, erfolgt eine Anbindung an den UDP/IP-Protokollstapel und die Klientanwendung erhält ein gültiges Socket, über welches sie im Weiteren kommuniziert. Schließt die Anwendung das Socket, wird der Kanal abgebaut.

Die Nachteile der ersten Variante bleiben erhalten. Lediglich die Änderung der Protokolle kann entfallen.

3.2.2 Verwendung des L4_TCP/IP Server

Im Rahmen einer Diplomarbeit hat Marco Rudolph [Rud2000] einen TCP/IP-Protokollstapel für das DROPS-System programmiert. Dieser verwendet als Kodebasis den TCP/IP-Protokollstapel vom FreeBSD.

Vorteile:

- Durch diese Arbeit gibt es einen lauffähigen TCP/IP Stack, der durch gewisse Änderungen an das Myrinet angepaßt werden kann.
- Die Speicherverwaltung erfolgt im FreeBSD-Kerns mittels mbuf's. Diese können gut an die Mechanismen von FIASCO angepaßt werden.

Nachteile:

- Leider realisiert der vorhandene Server lediglich eine Priorisierung beim Zugriff auf das Übertragungsmedium. Für den Echtzeit-Nachrichtentransfer ist aber ein funktionierendes Reservierungsschema und eine Ressourcenüberwachung notwendig.
- Aufgrund der verwendete Threadstruktur im Server können nur wenige Klientanwendungen gleichzeitig kommunizieren.
- Es ist Nicht-Echtzeit-Anwendungen möglich, den Nachrichtenaustausch von Echtzeit-Anwendungen zu beeinflussen.

3.2.3 Neuimplementierung

Die Protokolle können neu implementiert werden. Dies ist bei einer vollständigen Umsetzung aufwendig. Da für die Echtzeit-Fähigkeit viele Funktionen oder notwendige Informationen nicht benötigt werden oder nach der Erstellung des Echtzeit-Kanals schon fest stehen und oft nicht mehr verändert werden dürfen, kann die Option einer Neuprogrammierung interessant werden. Bestärkt wird dies dadurch, daß die Einhaltung der Verbindlichkeiten durch die Bitübertragungsschicht erfolgt und keine Veränderungen bzw. Erweiterungen an den Protokollen vorgenommen werden müssen.

Probleme ergeben sich dadurch, daß für die Echtzeit-Verarbeitung keine spezielle Protokollnummer für den IP-Paketkopf bereit steht. Es kann nur die Protokollnummer des UDP's verwendet werden. Es muß nun sichergestellt werden, daß UDP-Portnummern nicht gleichzeitig für den Nicht-Echtzeit- und Echtzeit-Datenaustausch verwendet werden. Dies kann durch einen Abgleich beider Protokollstapel erfolgen.

3.2.3.1 Ein zusagefähiges IP und UDP mittels eines Servers

Wenn eine Anwendung mittels IP/UDP zusagefähig kommunizieren möchte, wendet sie sich an einen Serverprozeß und übermittelt diesem ihre erwünschten Leistungsparameter. Der Server etabliert einen neuen Echtzeitkanal, der den Leistunganforderungen entspricht. Ist dies nicht möglich, weist er die Anfrage ab. Im Erfolgsfall erstellt er einen neuen Thread, der die folgenden Anfragen des Klienten beantwortet.

Der Thread hat die Aufgabe, die Daten des Klienten in gültige UDP- und IP-Datagramme zu packen. Das IP-Protokoll muß eventuell die Daten in mehrere Fragmente zerlegen, um die maximale Paketgröße der Bitübertragungsschicht nicht zu überschreiten. Anschließend übergibt der Servicethread die Pakete an die Bitübertragungsschicht.

Treffen Pakete für den Klienten ein, setzt der Thread die Fragmente zur originalen Nachricht zusammen, überprüft im UDP-Protokoll die korrekte Übertragung der Nachricht und übermittelt diese der Anwendung. Da die Reihenfolge der IP-Datagramme auf dem Weg durch das Netzwerk nicht verändert wird, kann das Zusammensetzen der Botschaft relativ einfach erfolgen.

Der Datentransfer zwischen einem Klient und einem Thread erfolgt gewöhnlich durch das Kopieren der Daten. Um es Anwendungen zu gestatten sehr leistungsfähig mit dem Thread zu kommunizieren, kann der Datenaustausch über gemeinsam genutzte Speicherseiten erfolgen. Leider müssen die Daten bei der Berechnung der Prüfsumme gelesen werden, so daß dieser Mehraufwand weiter bestehen bleibt. Viele Netzwerkadapter bieten die Möglichkeit, die Prüfsummenberechnung in Hardware zu erledigen. Diese Fähigkeit kann den beschriebenen Mehraufwand beseitigen.

Vorteile:

- Dieser Ansatz erlaubt es, einen leistungsfähigen Dienst anzubieten, der die Vorzüge und Besonderheiten des DROPS-Systems ausnutzt.
- Wie bereits erwähnt, muß nur ein Teil des UDP/IP-Protokollstapels zur Verfügung stehen. Der resultierende Service ist dadurch einfacher.

Nachteile:

- Trotz der eingesparten Funktionalität kann eine Neuprogrammierung langwierig, aufwendig und fehleranfällig sein.
- Unter FIASCO sind im Augenblick lediglich 128 Threads pro Prozeß erlaubt. Es kann folglich nur eine gewisse Anzahl von Echtzeit-Kanälen etabliert werden.

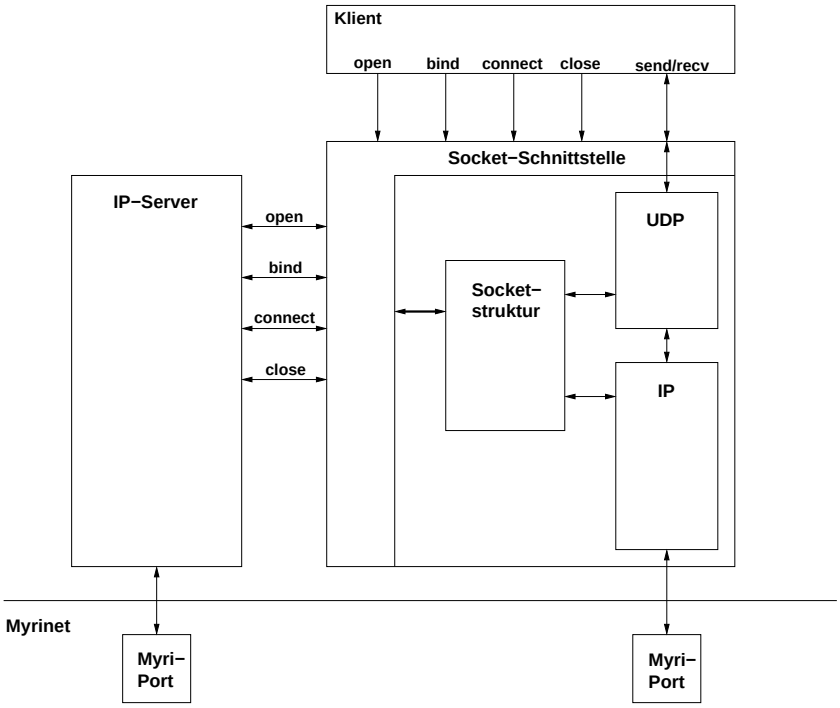

Abbildung 3.1: *Modell für einen zusagefähigen IP-Stack*

- Es ist nicht einfach den Bedarf an Betriebsmitteln zu bestimmen, der zusätzlich nötig ist, um mit den Thread zu kommunizieren. Er kann durch den nötigen Kontextwechsel groß sein, da z.B. „Cachemisses“ berücksichtigt werden müssen. Die Verzögerungszeit und der mögliche Jitter eines Pakets steigen an.

- Der Scheduler des Systems muß die Arbeit des Threads einplanen, oder der Serverprozeß muß einen eigenen Scheduler mitliefern.

3.2.3.2 Ein zusagefähiges IP und UDP durch eine Bibliothek

Da die Betriebsmittelüberwachung durch die Bitübertragungsschicht erfolgt und es eine feste Zuordnung von UDP-Port zu Echtzeit-Kanal gibt, kann kein Klient Daten an beliebige Rechner oder UDP-Ports auf der Empfängerseite senden. Es ist somit möglich, die Funktionen des UDP's und des IP's in Bibliotheken zu verlagern, die von den Klientanwendungen eingebunden werden müssen. Da das DROPS-System in naher Zukunft verteilte Bibliotheken unterstützt, ergeben sich kaum Nachteile durch einen erhöhten Speicherbedarf.

Möchte eine Klientanwendung in Echtzeit kommunizieren, muß sie bei der Anforderung ihrer Betriebsmittel den Mehraufwand durch die Bibliothek berücksichtigen. Die Berechnung der IP- bzw UDP-Datagramme auf der Senderseite bzw. die Rückwandlung der Daten auf der Empfängerseite erfolgt im Kontext der Klienten.

Weitere Verbesserungen:

- Dieser Ansatz erlaubt einen noch leistungsfähigeren Dienst als die Servervariante.
- Es ist einfacher den zusätzlichen Ressourcenbedarf zu berechnen, da kein Wechsel des Kontextes notwendig ist.
- Die Verzögerungszeit und der Jitter eines Pakets wird nicht größer.

Sich ergebende Probleme:

- Für diese Schema ist ein neuer Routingmechanismus nötig. Es wird ein Server benötigt, der diese Funktionalität bietet. Der Server kann für seine Arbeit die beschriebenen IP-Bibliothek verwenden. Wird beim Aufbau eines neuen Echtzeitkanals ein Routing notwendig, setzt der IP-Server den Routing-Server davon in Kenntnis.

Kapitel 4

Realisierung für das Myrinet

Im letzten Kapitel wurden mögliche Verfahren für den Zugriff und die Reservierung des Übertragungsmediums vorgestellt. Zur Erstellung des zusagefähigen Netzwerksystems wurde das Modell des Netzwerkschedulers gewählt. Das Verfahren ist universell einsetzbar, d.h. es kann auf jedes paketorientierte Netzwerk angewendet werden und erlaubt viele Optimierungen für das jeweilige Netzwerk. Für diese Arbeit wurde als Referenznetzwerk das Myrinet ausgewählt. Die Verarbeitung der Daten mittels IP und UDP erfolgt durch eine Bibliothek, welche von den Klientanwendungen eingebunden wird (siehe 3.2.3.2).

Im folgenden Abschnitt werden die Fähigkeiten der Myrinetadapter beschrieben. Ein weiterer Abschnitt erläutert Umsetzungsziele für die Myrinetarchitektur. In den darauf folgenden Abschnitten werden Teilaspekte der Umsetzung beschrieben, wie Konzepte und Arbeitsweise der beteiligten Instanzen. Der abschließende Abschnitt erläutert einige Details aus der Implementierung des IP- und des UDP-Protokolls und der Socketschnittstelle.

4.1 Fähigkeiten einer Myrinetadapterkarte

Um eine hohe Leistungsfähigkeit zu erreichen, sollen die gebotenen Fähigkeiten der Myrinetadapterkarten effizient genutzt werden. Dieser Abschnitt soll aber nicht dazu dienen, das Myrinet vorzustellen. Dies wurde bereits detailliert unter [Rei2000] getan.

- Das Myrinet ist ein Punkt-zu-Punkt-Netzwerk. Die Switches verwenden die „Cut-through“-Switchingtechnik (siehe 2.1.2).
- Jede Myrinetkarte besitzt einen Prozessor, der die anderen Hardwarekomponenten der Adapterkarte steuert. Dies erfolgt über Spezialregister. Der Prozessor ist frei programmierbar und kann mittels einer Firmware für Steuerungs- und Verwaltungszwecken der Netzwerkprotokolle herangezogen werden. Es ist vorstellbar einen Teil des TCP/IP-Protokollstapels auf der Karte zu realisieren. Zu diesem Zweck werden von der Firma Myricom ein angepaßter C-Compiler und andere Programmierwerkzeuge angeboten.

Die Prozessoren der vorliegenden Netzwerkkarten (LanAI 4.01 bzw. 4.03) arbeiten mit einer Verarbeitungsbreite von 32 Bit.

- Der Prozessor kennt zwei Betriebsmodi, einen nicht unterbrechbaren Systemmodus und einen unterbrechbaren Nutzermodus.

- Wie im „Peripheral Component Interconnect“-Standard (PCI) spezifiziert, kann die Adapterkarte mittels Busmastering (DMA-Transfer) selbständig Hauptspeichertransfers durchführen. Während des DMA-Transfers berechnet die zuständige Steuereinheit die Internetprüfsumme und speichert diese in einem Spezialregister. Ein Transfer kann nur Adressen erfolgen, die auf 32 Bit ausgerichtet sind.

- Für den Zugriff auf das Übertragungsmedium steht eine eigene Steuereinheit zur Verfügung. Es ist möglich Daten in 8, 16 und 32 Bit Granularität direkt zu versenden bzw. zu empfangen. Weiterhin kann ein Bereich des Kartenspeichers angegeben werden, der durch die Steuereinheit mittels DMA gefüllt bzw. gesendet wird.

- Die Steuereinheiten können den Prozessor mittels Unterbrechungen über Ereignisse informieren. Der Prozessor wird hierbei in den Systemmodus geschaltet. Arbeitet er bereits in diesem Modus, kann er über ein Statusregister der Zustand der Steuereinheiten erfragen.

- Eine Myrinetkarte besitzt zwei Möglichkeiten zur Zeitmessung. Erstens kann die „Echtzeit-Uhr“ ausgelesen werden. Diese arbeitet mit einer Genauigkeit von $0.5\mu s$. Im zweiten Fall kann nach einem bestimmten Intervall eine Unterbrechung ausgelöst werden. Hierzu muß lediglich der Zeitraum (ebenfalls in $0.5\mu s$ großen Zeitschritten) in eines der Spezialregister eingetragen werden.

- Der Kartenspeicher, der Inhalt des EEPROM's und ein Teil der Spezialregister des Prozessors werden in den Adreßraum des Hauptprozessors eingeblendet und sind von dort aus direkt zugreifbar.

4.2 Umsetzungziele

- Um ein unnötiges Kopieren von Daten zu vermeiden, soll die „Scatter-Gather“-Technik verwendet werden. Hierbei wird ein Datenpaket aus einer Anzahl von Teilen zusammen gesetzt, die sich an beliebigen Stellen im Speicher befinden. Diese Teile sind die Daten und die Protokollinformationen für die Empfängerstation.

  Motiviert wird dieses durch den schichtenweisen Aufbau der Netzwerkprotokolle. Eine Schicht nimmt von der darüberliegenden Daten entgegen und berechnet daraus ihren Protokollkopf. Der Protokollkopf und die entgegengenommenen Daten bilden die Daten der darunterliegenden Schicht. Um die Daten nicht in jeder Schicht kopieren zu müssen, kann ein Datenaustausch über gemeinsamen Speicher erfolgen. Hierzu übergibt eine Schicht der darunterliegenden eine Liste. Darin sind die Startadresse

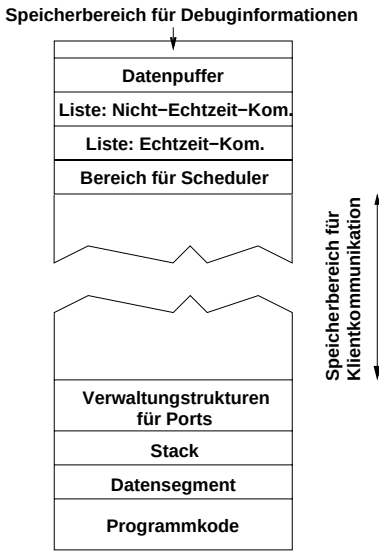

Abbildung 4.1: *Die Speicheraufteilung für die Firmware*

und die Größe der Datenblöcke verzeichnet. Anhand dieser kann die Schicht ihren Protokollkopf berechnen und fügt in der Liste einen Eintrag mit einem Verweis auf ihren Datenblock hinzu.

- Die Berechnung der Internet-Prüfsumme ist eine relativ aufwendige Operation. Es müssen alle Daten einer Botschaft addiert werden. Die Summe wird an einer bestimmten Stelle eines Protokollkopfes abgelegt. Diese Operation soll von den Adapterkarten durchgeführt werden.

- Die Umsetzung soll Anwendungen eine hohe Bandbreite bereitstellen. Gleichzeitig darf sie nur einen geringen Mehraufwand für den Hostprozessor darstellen.

4.3 Die Firmware

Aufgrund der relativ hohen Rechenkapazität des Prozessors einer Adapterkarte übernimmt die Firmware bei der Umsetzung des Modells eine zentrale Rolle. Sie allein erledigt die Aufgaben der Bitübertragungsschicht. Weiterhin stellt sie Hilfsmittel zum Aufbau der Echtzeitkanälen bereit und überwacht die Klientkommunikation. Hierzu benötigt sie die Hilfe eines Serverprozesses. Die genauen Zusammenhänge und Arbeitsweisen werden in den späteren Abschnitten beschrieben.

Für den Rest dieser Arbeit wird sich stark an den von Myricom verwendeten Begriffen orientiert. Im Besonderen soll hier der Begriff Port hervorgehoben werden. Ein Port ist in diesem Zusammenhang ein logischer Kommunikationsendpunkt. Sämtlicher Botschaftenaustausch zwischen den Hosts erfolgt über Ports. Einem Port wird ein fester Bereich im

Kartenspeicher zugewiesen, über den die Firmware und die Klientanwendungen kommunizieren. Der Anwendung wird hierzu dieser Bereich in den Adreßraum eingeblendet. Je mehr Ports bereitstehen, um so mehr Verbindungen zu einem Rechner sind möglich.

Ein Port kann entweder für den Echtzeit- oder den Nicht-Echtzeit-Datenaustausch verwendet werden, nie für beide Übertragungsarten gleichzeitig. Er kann immer nur einem Prozeß „gehören“, allerdings darf ein Prozeß mehrere Ports „besitzen“. Die Kommunikation erfolgt bei Echtzeit-Ports unidirektional und bei Nicht-Echtzeit-Ports bidirektional.

Um eine an das System angepaßte Berechnung der Sende- und Empfangszeiten vorzunehmen, wird nach der Initialisierung der Firmware eine Leistungmessung durchgeführt. Hierbei werden DMA-Transfer- und Roundtrip-Zeiten bei verschiedenen Datengrößen ermittelt. Die Ergebnisse werden sowohl dem Server als auch dem Scheduler zugänglich gemacht und als „worst case“-Transferzeiten interpretiert. Die Meßergebnisse werden im Kapitel 5 vorgestellt. Die Firmware unterstützt eine maximale Paketgröße von 8192 Bytes. Dabei darf für keines der „Scatter-Gather“-Teile größer als 4096 Bytes sein.

4.4 Das Arbeitskonzept

Um ein sicheres und störungsfreies Arbeiten zu gewährleisten, gibt es für jeden Port fünf Verwaltungsstrukturen. Jede Struktur erfüllt bestimmte Aufgaben.

Um Wettlaufbedingungen beim Zugriff auf gemeinsam genutzte Strukturen zu vermeiden, sind drei davon zu Warteschlangen in Form von Ringpuffern organisiert. Jeder Partner erhält einen Zeiger auf die Struktur, die er als nächstes bearbeiten möchte. Nur er darf diesen Zeiger verändern. Alle anderen beteiligten Partner dürfen diesen nur lesen. Auf diese Weise können Fehlerzustände und Überläufe der Ringpuffer festgestellt werden.

Informationen für den Klient: Diese Struktur enthält die Zeiger des Klienten. Anhand dieser ermittelt die Firmware mögliche Fehlerzustände. Weiterhin sind hier Flags zur Steuerung der Unterbrechungsbehandlung vorgesehen. Möchte ein Klient ein blockierendes Empfangen durchführen, setzt er eines dieser Flags. Empfängt die Firmware ein Paket für diesen Port, transferiert sie die Daten in einen bereitgestellten Puffer und löst eine Hardwareunterbrechung aus.

Sendetoken-Warteschlange: Ein Sendetoken ist eine Struktur zum Senden eines Datenpaketes. Ein Klient füllt diese Struktur, kopiert sie in einen freien Schacht des Ringpuffers und setzt seinen Zeiger ein Warteschlangen-Element weiter. Wurde das Paket versendet, wird das Token zurückgegeben, d.h. die Firmware setzt ihren Zeiger auf das nächste Element der Warteschlange.

Der Datentransfer in den Kartenspeicher erfolgt mittels eines DMA-Transfers. Die Firmware fügt die Daten zu einem gültigen Myrinetpaket zusammen und sendet dieses über das Medium.

Die Sendetoken-Warteschlange befindet sich zusammen mit der Empfangstoken-Warteschlange und der Informationsstruktur für den Klienten in der physischen Seite des

Kartenspeichers, welche in den Klientadreßraum eingeblendet ist. Sie ist somit direkt zugreifbar. Die Anzahl der Elemente des Ringpuffers wird durch den Serverprozeß bestimmt.

Empfangstoken-Warteschlange: Mittels eines Empfangstoken stellt der Klient Puffer bereit, in denen ankommende Pakete gespeichert werden sollen. Die Arbeitsweise erfolgt ähnlich wie bei der Sendetoken-Warteschlange.

Ereignis-Warteschlange: Ein Ereignis ist im vorliegenden System der Empfang eines Datenpakets. Es wird durch die Firmware generiert und in die Warteschlange eingereiht. Die Ereignis-Warteschlange wird zum Initialzeitpunkt eines Ports vom Server angelegt und in den Adreßraum des Klienten eingeblendet. Im Gegensatz zu den anderen Warteschlangen befindet sich diese im Hauptspeicher des Hosts.

Informationen über den Port: Diese Struktur enthält die Informationen, welche die Firmware für ihre Arbeit benötigt, wie Art des Ports (Echtzeit/Nicht-Echtzeit), wieviele „Scatter-Gather"-Elemente versendet bzw. empfangen werden können, die Startadresse des Datenpuffers und dessen Größe. Diese Struktur enthält auch die Verwaltungsinformationen für die Warteschlangen, so daß die Firmware die Verbindlichkeiten der Klienten überprüfen kann.

Diese Struktur wird zum Aktivierungzeitpunkt durch den Serverprozeß beschrieben, da dieser die Betriebsmittel beschafft und deren Eigenschaften kennt. Diese Port-Informationen werden vor den Klienten verborgen gehalten.

4.5 Der Serverprozeß

Die Hauptaufgabe des Serverprozesses ist die Verwaltung der Ports und deren Initialisierung. Weiterhin berechnet er den Speicherbedarf der Klientanwendungen und beschafft die berechnete Anzahl an Puffer von seinem Pager. Eine weite Aufgabe besteht darin, den Aufbau von Echtzeit-Kanälen vorzunehmen.

Für die Behandlung von Unterbrechungen ist ein spezieller Thread vorhanden. Hardwareunterbrechungen treten nur bei blockierendem Empfangen auf. Der Thread bestätigt diese und weckt die entsprechende Klientanwendung auf.

4.6 Der Zugriff auf die Portwarteschlangen

Der Zugriff auf die Warteschlangen erfolgt durch die Klientanwendungen. Sie haben dafür zu sorgen, daß für den Empfang ausreichen Puffer zur Verfügung stehen. Empfängt die Firmware ein Paket und es stehen keine Puffer bereit, verwirft sie das Paket.

Um Fehler zu vermeiden, sollten Anwendungen die vorgefertigte Bibliothek verwenden. Diese enthält alle wichtigen Servicefunktionen.

Vorteile:

- Jede Klientanwendung kann ihre Strukturen selbständig verwalten und direkt mit der Firmware kommunizieren. Auf diese Weise werden viele Prozeßumschaltungen vermieden.
- Es können mehrere Klienten gleichzeitig auf ihren Verwaltungsstrukturen arbeiten, ohne daß sich diese behindern oder daß Wettlaufbedingungen auftreten.
- Durch die direkte Arbeitsweise wird die Verzögerung von Paketen durch das System herabgesetzt.
- Die zentrale Instanz wird nur noch zum Auf- bzw. Abbau eines Echtzeit-Kanals benötigt. Sie verwaltet die Puffer, fordert diese an und blendet sie in den Adreßraum der Klienten ein.

Nachteile:

- Es ist ein höherer Aufwand für das Anfordern bzw. Abkoppeln von Ports nötig.
- Ein neues Sicherheitskonzept wird benötigt. Daraus resultiert ein weiterer Mehraufwand.

4.7 Das Speicherkonzept

Um potentiell unsicheren Anwendungen Hardwareseiten einblenden zu können und trotzdem ein sicheres Arbeiten zu gewährleisten, sind einige weiterführende Überlegungen notwendig. Die Anwendungen können durch die Hardware DMA-Transfers ausführen und haben somit Zugriff auf den gesamten physischen Speicher des Hosts. Es muß folglich ein direkter Umgang mit physische Adressen verhindert werden. Anderseits benötigt die Hardware für einen DMA-Transfer physische Adressen.

Echtzeit-Anwendungen kennen ihren Speicherbedarf bereits zur Anforderungszeit des Ports. Es ist zu diesem Zeitpunkt möglich, die benötigten Puffer anzufordern und der Karte bekannt zu machen. Arbeitet ein Klient mit Offsets, genügen zwei Parameter, um den Bereich klar abzugrenzen. Diese Parameter sind die physische Startadresse und die Größe des Puffers. Außerdem muß der Serverprozeß nicht wissen, an welche Stelle im Adreßraum des Klienten der Puffer eingeblendet wird.

Schwerer ist dies bei Nicht-Echtzeit-Anwendungen. Diese kennen ihren Pufferbedarf oftmals nicht oder er variiert über die Zeit sehr stark, z.B. ein TCP/IP-Protokollstapel, der wiederum von Anwendungen benutzt wird. Durch das „Scatter-Gather“-Verfahren verschärft sich das Problem noch mehr, da Anwendungen angeforderte Speicherbereiche ihren Klienten weitermappen wollen.

Lösungsmöglichkeiten:

- Ein Ansatz zur Lösung des Problems ist, für Nicht-Echtzeit-Anwendungen das Verfahren zu verwenden, das GM [MYR99] benutzt. In diesem System arbeiten die Klienten mit virtuellen Adressen. In der Firmware wird für jeden Port eine Umrechnungstabelle von virtuellen auf physische Adressen gehalten. Diese Tabelle wird durch den Serverprozeß verwaltet und bleibt den Klienten verborgen. Die Klienten müssen Speicherbereiche beim Server anfordern.

  Vorteile:

  - Dynamisches Anfordern von benötigten Speicherbereichen ist möglich.

  Nachteile:

  - Das Verfahren benötigt für seine Arbeit sehr viel Speicher auf der Myrinetkarte. Um große Tabellen zu unterstützen, wird unter GM ein Teil des Hauptspeichers verwendet. Wird ein Teil der Tabelle benötigt, liest die Firmware die entsprechenden Seiten in den Kartenspeicher. Es ist zusätzliche Zeit für diesen Transfer einzuplanen. Eine Nicht-Echtzeit-Anwendung kann dafür sorgen, daß Speicherseiten mit den Übersetzungstabellen von Echtzeit-Anwendungen überschrieben werden. Dadurch wird die Vorhersagbarkeit verschlechtert bzw. es muß der „worst case“ ermittelt und beachtet werden.

- In einem restriktiven Ansatz wird einer Anwendung ein Speicherbereich fester Größe zugewiesen, mit dem diese zurechtkommen muß.

  Vorteile:

  - Das Verfahren ist sehr einfach.

  Nachteile:

  - Benötigt eine Anwendung für ihre Arbeit sehr wenig Speicher, wird der überschüssige Bereich verschwendet. Benötigt eine Anwendung sehr viel Speicher, kann sie eventuell nicht effizient oder gar nicht arbeiten oder sie kann nicht so viele Klienten bedienen.

- Ein Zwischenweg kann darin liegen, daß Anwendungen bei der Anforderung eines Ports Hinweise auf deren Ressourcenbedarf liefern. Mittels dieses Hinweises wird der Anwendung ein fester Bereich zu gewiesen.

  Verbesserungen:

  - Die Pufferbereitstellung orientiert sich besser an den Bedürfnisse der Klientanwendungen.

  Für die Umsetzung des Modells wird diese Vorgehensweise angewendet.

4.8 Blockierendes und nicht blockierendes Empfangen

4.8.1 Nicht blockierendes Empfangen

Beim nicht blockierenden Empfangen ermittelt der Klient ob neue Ereignisse in seiner Ereigniswarteschlage stehen. Ist dies der Fall, kann er die Daten verarbeiten. Anderenfalls kehrt die Bibliotheksfunktion unverzüglich zurück und die Anwendung kann sich einer andere Aufgaben widmen. Sie muß die Warteschlange zyklisch prüfen.

4.8.2 Blockierendes Empfangen

Für das blockierende Empfangen überprüft die Bibliothek, ob sich noch Ereignisse in der Warteschlage befinden. Ist diese leer, ermittelt die Bibliothekfunktion, ob die Firmware ein Paket für diesen Portbesitzer verarbeitet. Hierzu gibt es ein Flag in der eingeblendeten Speicherseite. Erst wenn auch diese Bedingung nicht erfüllt ist, wird die Zustellung einer Unterbrechung erbeten. Hierzu setzt die Bibliothek ein Flag. Anschließend führt sie eine Empfangsoperation (Short-IPC) durch, bei der sie nur Botschaften des Unterbrechungsbehandlungsthreads akzeptiert.

Wird eine IPC empfangen, sendet die Bibliothek eine Antwortnachricht zurück. Dies ist notwendig, da FIASCO die Zeitscheibe des Senders an den Empfänger übergibt. Um schnell auf ankommende Unterbrechungen reagieren zu können, ist diese Konstruktion notwendig.

4.8.3 Hardwareunterbrechungen

Wie schon erwähnt, treten Unterbrechungen nur im Zusammenhang mit blockierendem Empfangen auf. FIASCO generiert beim Auftreten einer Hardwareunterbrechung eine IPC und sendet diese dem Thread, der sich für die Unterbrechungsbehandlung registriert hat. Im konkreten Fall ist dies der Omega_0-Server [LoHo2000]. Er bewirbt sich beim Start des Systems um alle verfügbaren Unterbrechungen. Auf diese Weise wird die fehleranfällige Unterbrechungsbehandlung von einer zentralen Einheit durchgeführt. Gleichzeitig ist es möglich, daß sich mehrere Geräte einen Interrupt teilen.

Ein Thread kann sich nun um die Zustellung der Unterbrechung beim Omega_0-Server bewerben. In diesem System ist das der Interrupt-Thread des Myrinetservers. Er ermittelt den Klient, den er benachrichtigen soll und bestätigt den Erhalt der Unterbrechung bei der Firmware. Im nächsten Schritt sendet er dem Klient eine Short-IPC, durch die dieser aufgeweckt wird.

4.9 Netzwerkscheduler und Firmware

Jeder Host im Netzwerk erhält vom Netzwerkscheduler eine eindeutige Host-ID zugewiesen. Die Kommunikation mit anderen Netzwerkteilnehmern erfolgt über diese Identifikations-

nummer. Die Firmware löst diese Nummer in eine Route auf, auf der das Paket versendet wird.

Um ein effizientes Scheduling zu gewährleisten, ist eine enge Zusammenarbeit zwischen dem Scheduler und der Firmware des Hosts, auf dem der Scheduler läuft, nötig. Im weiteren Verlauf dieses Abschnittes wird sich immer auf diese Firmware bezogen.

Der Scheduler wird auf dem Hostprozessor ausgeführt und arbeitet als Nicht-Echtzeit-Anwendung. Er muß sich mit den anderen Nicht-Echtzeit-Anwendungen die verbleibende Rechenkapazität teilen. Es kann somit sehr lange dauern, bis ein gültiger Schedule berechnet wurde. Aus diesem Grund erstellt der Scheduler zwei Listen, eine für die Nicht-Echtzeit- und eine für die Echtzeit-Kommunikation. Diese Listen werden in den Speicher der Myrinetkarte transferiert. Um Störungen mit den bereits vorhandene Listen zu vermeiden und die Verzögerung klein zu halten, wird der Firmware mitgeteilt, daß neue Listen bereit stehen. Diese wartet so lange, bis Zeit für einen Nicht-Echtzeit-Transfer vorhanden ist und liest die Listen mittels eines DMA-Transfers in den Kartenspeicher.

Die Generierung von Steuerpaketen bleibt der Firmware überlassen. Sie analysiert hierzu die Echtzeit- und Nicht-Echtzeit-Listen. Um die Echtzeit-Liste nicht zu groß werden zu lassen, wird durch den Scheduler eine Periode festgelegt. Während einer Periode wird diese Liste komplett abgearbeitet. Ein Listeneintrag besteht aus einem Startzeitpunkt, der Host-ID des Empfängers, der Information, für welchen Port Pakete versendet werden dürfen, der Anzahl an Paketen und einem „timeout“. Der Startzeitpunkt legt fest, wann, relativ zum Periodenbeginn, ein Steuerpaket versendet werden muß. Der Wert für den „timeout“ wird nicht versendet. Er wird vielmehr von der Firmware benötigt, um festzustellen, ob ein Knoten ausgefallen ist. Dies wird durch ein Ausbleiben einer Rückantwort erkannt.

Die Zeit zwischen der Rückmeldung und dem Startzeitpunkt des nächsten Listeneintrages kann für die Nicht-Echtzeit-Kommunikation verwendet werden. In der Nicht-Echtzeit-Liste werden alle verfügbaren Knoten des Netzwerkes, deren Host-ID's und einer Route zu diesen aufgeführt. Die Firmware arbeitet die Liste von oben nach unten durch. Nach deren Abarbeitung wird auf den Listenanfang zurückgesetzt. Der Scheduler legt nur fest, wieviel Zeit pro Host verwendet werden darf. Kann einem Knoten nur ein Teil dieser Zeit gewährt werden, vermerkt dies die Firmware. Er bekommt die fehlende Zeit bei der nächsten Gelegenheit zugewiesen. Benötigt ein Host nur einen Teil der ihm zugewiesenden der Zeit, gibt er diese der Firmware des Schedulers zurück.

4.10 Die Berechnung eines Schedules

Die Berechnung eines Schedules erfolgt im Augenblick aus folgenden Parametern: die Periode der Anwendung, der maximalen Anzahl von Daten pro Periode, der maximalen Paketgröße und den gemessenen Leistung des Systems. Es wird angenommen, daß die Pakete maximal gefüllt werden, so daß eine maximalen Anzahl von Pakete errechnet werden kann. Die Echtzeitkanäle werden nach der Periode sortiert und der Kanal mit der kleinsten Periode bestimmt die Periode der Echtzeit-Liste. Der Scheduler ermittelt anhand der Leistung der Systeme und der maximalen Paketgröße den Bedarf für einen Kanal. Zuvor muß er

die geforderten Leistungsparameter auf die minimale Periode umrechnen und gegebenenfalls auf die maximale Paketgröße aufrunden. Der Scheduler optimiert auf lawinenartiges Nachrichtenaufkommen, so daß der Mehraufwand zum Versenden von Kontrollnachrichten gering bleibt.

Es bestehen Beschränkungen für die möglichen Perioden. Bei der Überschreitung dieser Schranken, errechnet der Scheduler neue Perioden, so daß die geforderten Eigenschaften eingehalten werden. Unterschreitungen sind nicht zulässig.

Durch die Umrechnung und der notwendigen Aufrundungen kann Bandbreite verschwendet werden, obwohl diese nie verwendet wird. Es können weniger Echtzeitkanäle etabliert werden. Kann der Scheduler kein gültiges Schedule erstellen, verwirft er die Berechnung und weist die Anfrage ab.

Nachteile:

- Bei der Berechnung eines Schedules wird die Verzögerung und der Jitter der Pakete nicht berücksichtigt.
- Es kann zu einer Verschwendung von Bandbreite kommen.

4.11 Probleme bei Umsetzung des Entwurfes

- Um unter FIASCO auf physische Speicheradressen über einem GByte zugreifen zu können, muß mittels des sigma$_0$-Protokoll (σ_0) [Lie96] die 4MB Superpage angefordert werden, die den benötigten I/O-Adreßbereich enthält. Leider gestattet es FIASCO im Augenblick nicht, seitengroße Bereiche aus einer Superpage als Flexpage zu versenden. Es ist nur möglich, die gesamte Superpage als Flexpage zu versenden. Da aber ein schreib-/lesbares Weiterleiten von Teilbereichen des eingeblendeten Adreßbereichs erforderlich ist, kommt es hier zu einem Sicherheitsproblem. Nach dem Einblenden der Superpage ist es einem Klient nicht nur möglich, sämtliche Informationen der Karte zu lesen, sondern diese auch zu verändern. Ferner kann er den Speicherbereich der Firmware manipulieren. Es gibt für dieses Problem im Augenblick keine Lösung.
- Unter dem DROPS-System gibt es momentan keinen Pager, der das Pinnen von Speicherseiten gestattet. Für diese Arbeit wird angenommen, daß keine Speicherseiten verdrängt werden.
- Wie schon weiter oben beschrieben wurde, ist es möglich, die Berechnung der Internetprüfsumme durch die Hardware erledigen zu lassen. Aus einem noch nicht geklärtem Grund ließ sich diese Fähigkeit nicht korrekt anwenden. Das gestellte Ziel wurde nicht erfüllt. Die Prüfsummenberechnung erfolgt durch den Hostprozessor.
- Die Arbeitsweise der Myrinetkarten und deren Programmierung sind sehr ausführlich dokumentiert. Leider sind in den Dokumenten und in den Kommentaren des frei

verfügbaren GM-Protokolls Fehler enthalten. Desweiteren arbeitet der mitgelieferte C-Compiler in älteren Versionen fehlerhaft. Die Problemauffindung und -beseitigung erwies sich als sehr aufwendig und zeitintensiv.

- Zur Initialisierung einer Myrinetkarte wird diese in einen Resetzustand versetzt und die Firmware in den Kartenspeicher transferiert. Wird der Resetzusand aufgehoben, beginnt der Prozessor die Firmware abzuarbeiten. Gleichzeitig ist der physische Port sende- und empfangsbereit. Dies kann nicht umgangen werden. Die Steuerung des Empfangs erfolgt aber durch die Firmware. Erst wenn das letzte Datum vom Netz genommen wurde, ist die physische Verbindung zwischen einem Sender und einem Empfänger freigegeben. Führt ein Host gerade seine Leistungsmessungen durch und der Netzwerkscheduler erforscht in dieser Zeit das Netz. Kann die Verbindung zwischen dem Scheduler und dem Host blockieren. Dieser Zustand wird erst aufgehoben, wenn das Paket vom Empfänger entgegennimmt oder der „timeout“ der Empfangseinheit auftritt. Im letzteren Fall wird das Paket durch die Empfangseinheit verworfen. Da der „timeout“ nur sehr grob eingestellt werden kann, ist es möglich, daß „deadlines“ des Schedulers nicht eingehalten werden. Diese Verhalten wurde im Entwurf nicht beachtet.

4.12 IP, UDP und Sockets

4.12.1 Details zum IP-Protokoll

Aufgrund der strengen send/receive-Semantik wird für das Empfangen von Datenpaketen die blockierende Variante verwendet. Diese Lösung ist zwar nicht sehr flexibel, kommt aber dem originalen Empfangen am nächsten. Zusätzlich muß die Bearbeitungszeit für eine Hardwareunterbrechung auf der Empfängerseite eingeplant werden.

4.12.2 Das Routen von IP-Datagrammen

Ein Router erstellt für jeden Echtzeitkanal einen neuen Thread, dieser besitzt jeweils einen Port einer Netzwerkschnittstelle. Da ein Echtzeitkanal unidirektional arbeitet, nimmt der Thread von einem Port Pakete entgegen und sendet diese auf dem anderen Port hinaus. Da die Signalisierung des Verbindungabbaus außerhalb des Datenstromes stattfindet und die Routeninformation feststehen, kann der Vorgang des Routens sehr leistungsfähig erfolgen.

Zum Erstellungszeitpunkt dieser Arbeit ist Routen von IP-Datagrammen noch nicht implementiert.

4.12.3 Die Socketschnittstelle

Die Socketschnittstelle entspricht, außer der socket()-Funktion, der im BSD 4.3 eingeführten API (Application Programming Interface). Es wird allerdings eine Einhaltung der Reihenfolge bei den Funktionsaufrufe erwartet. So ist vor dem Senden bzw. Empfangen

eine socket()-, eine bind()- und eine connect()-Operation durchzuführen. Der Aufbau des Echtzeitkanals erfolgt erst während der connect()-Anweisung.

Bei der socket()-Anweisung muß der Programmierer zusätzlich die erwarteten Leistungsparameter angeben. Bei einer erfolgreichen Bearbeitung liefert die Socketschnittstelle wird eine Socketnummer zurück, auf die in den folgenden Operation verwiesen wird.

Für das Senden steht die send()-Operation und für das Empfangen die recv()-Operation zur Verfügung. Die readv()- und writev()-Funktionen sind zwar vorgesehen, aber noch nicht implementiert.

4.12.4 Die Speicherverwaltung

Das Arbeitsschema des IP-Servers und der IP-, UDP- und der Socketbibliothek wurde bereits in der Abbildung 3.1 vorgestellt. Die Bibliotheksfunktionen sind so ausgelegt, daß sie die bereitgestellte „Scatter-Gather“-Funktionalität verwenden. Zu diesem Zweck wird eine einheitliche Speicherverwaltung verwendet, die auf einem Konzept ähnlich der mbuf's aus dem FreeBSD aufbaut.

Im Gegensatz zu FreeBSD kann ein mbuf im gewähltem Ansatz keine Daten aufnehmen. Sie dienen lediglich der Verwaltung des eingeblendeten Speichers. Hierzu verweist ein mbuf auf einen Bereich dieses Speichers. Ein Bereich kann maximal 4096 Bytes groß sein. Um größere Speicherbereiche zu bilden, können die mbuf's verkettet werden. Eine Verkettung kann auch vorgenommen werden, um die „Scatter-Gather“-Funktionalität zu erreichen.

Die mbuf's werden nach dem Öffnen und dem erfolgreichen Einblenden des gepinnten Speichers angelegt. Die Verweise auf diesen Speicher werden aus der Anzahl der „Scatter-Gather“-Element und deren Größe berechnet.

Das Grundkonzept des zusagefähigen Myrinets sieht ein Mappen des Speichers vom Server zum Klienten vor. Der Klient seinerseits kann Server für andere Anwendungen sein. Das Arbeiten auf den Daten kann sehr lange dauern, des weiteren müssen verwendete Puffer explizit angefordert und freigegeben werden. Ist die Kette sehr lang, sind viele Botschaften notwendig. In dieser Zeit ist der Puffer nicht verfügbar und es entsteht ein hoher Mehraufwand. Aus diesem Grund wurde entschieden, die Daten an der Socketgrenze zu kopieren.

Kapitel 5

Leistungsmessung

In diesem Kapitel werden die Ergebnisse durchgeführter Leistungsmessungen vorgestellt. Diese Ergebnisse werden für die Berechnung des Schedules benötigt.

Die Ausführungsgeschwindigkeit der Firmware ist für alle verwendeten Myrinetkarten gleich. Diese arbeiten mit einem Prozessortakt von 40MHz. Dies gilt auch für die Übertragungsleistung auf das Medium. Es steht eine Bandbreite von 1280 MByte pro Sekunde zur Verfügung. Unterschiede existieren für die Rechenleistung eines Hostes und dessen DMA-Durchsatz. Für die Messungen standen drei Rechner zur Verfügung, ein PPro 200 und zwei P90.

Da die Paketgröße 8192 Bytes nicht übersteigt, wurden die Messungen bis zu dieser Paketgröße durchgeführt. Jede Messungen erfolgt 10 mal. Die größte Zeitdifferenz wird gespeichert und als „worst-case“ verwendet. Die Zeitmessung erfolgt mittels der Echtzeituhr der Adapterkarten.

5.1 Messungen in der Firmware

5.1.1 Messung der DMA-Transfers

Diese Messungen werden zur Initialisierungszeit einer Myrinetkarte durchgeführt. Hierbei werden Datenblöcke unterschiedlicher Größe aus dem Hauptspeicher in den Kartenspeicher transferiert und umgekehrt.

DMA-Transfer aus dem Hauptspeicher

Bei dieser Messung wird erwartet, daß der Durchsatz mit der Größe der Pakte zunimmt (Abb. 5.1). Die Erwartungen haben sich weitestgehend bestätigt. Warum der Durchsatz bei dem PPro200-System und einer Paketgröße von 8192 Bytes leicht zurück geht, ist ist im Augenblick nicht bekannt.

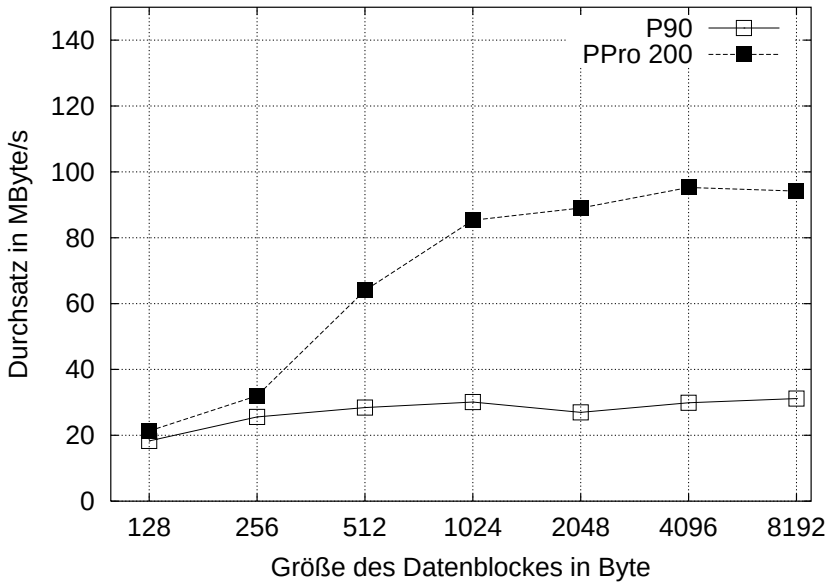

Abbildung 5.1: *Leistungsmessung: Der Durchsatz bei einem DMA-Transfer aus dem Hauptspeicher in Abhängigkeit von der Größe des Datenblockes*

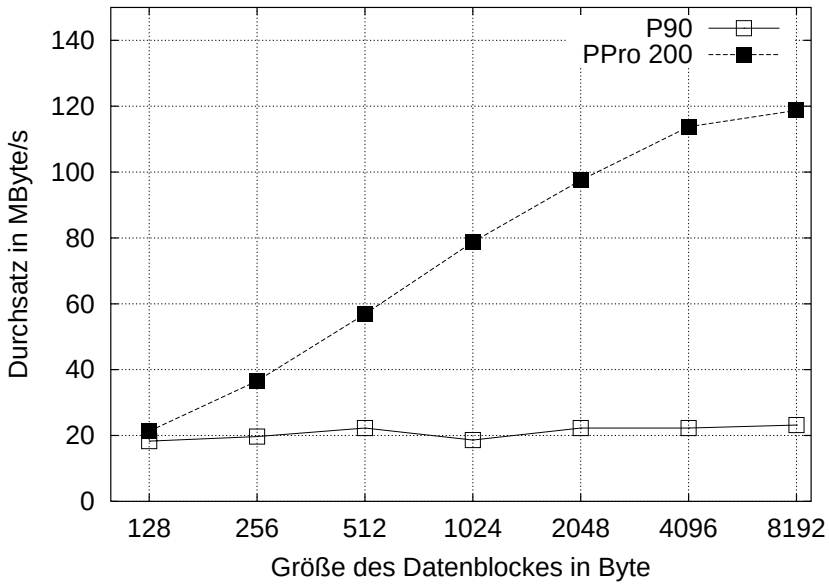

Abbildung 5.2: *Leistungsmessung: Der Durchsatz bei einem DMA-Transfer in den Hauptspeicher in Abhängigkeit von der Größe des Datenblockes*

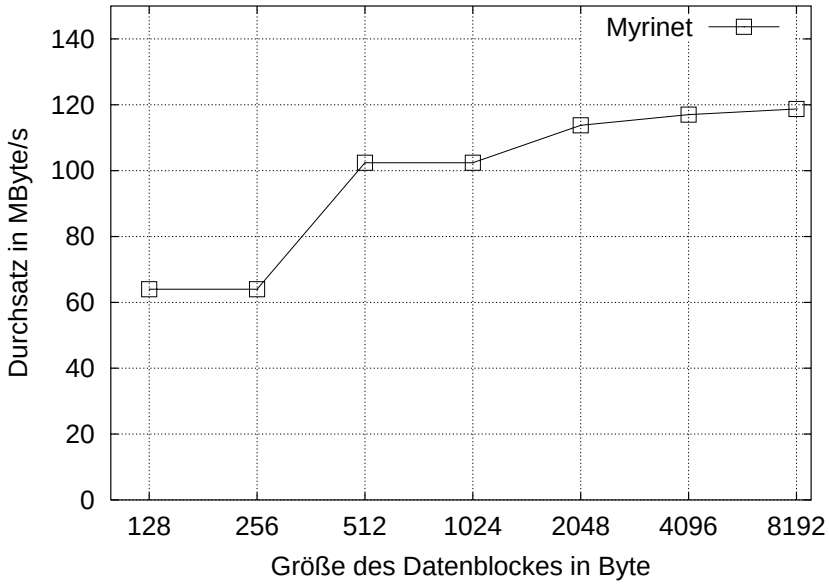

Abbildung 5.3: *Leistungsmessung: Der Durchsatz bei einem Netzwerzugriffs in Abhängigkeit von der Größe des Datenblockes*

DMA-Transfer in den Hauptspeicher

Auch bei dieser Messung (Abb. 5.2) sollte der Durchsatz mit steigender Paketgröße zunehmen. Dies ist auch bei allen Systemen, auf denen die Messungen durchgeführt wurden, der Fall. Es ist aber zu beachten, daß die Daten nicht direkt in den Speicher transferiert werden. Der Schaltkreis, der die Verbindung zum Speicherbus realisiert, kann einen Teil der Daten puffern. Der Durchsatz dürfte bei großen Datenmengen geringer ausfallen. Desweiteren soll an dieser Stelle angemerkt werden, daß es keinerlei Verwaltung des PCI-Busses gibt. Transferieren mehrere Geräte Daten über diesen Bus, kann es zu Einschnitten der verfügbaren Bandbreite kommen. Dieser Umstand wird im Augenblick nicht beachtet.

5.1.2 Messung des Netzwerkdurchsatzes

Die Messung des Netzwerkdurchsatzes muß nicht von jeder Firmware durchgeführt werden, da die Meßumgebung für alle Netzwerkknoten die gleiche ist. Vielmehr wurde sie beispielhaft durchgeführt und die Resultate sind allen Rechner zugänglich.

Zur Durchführung wurden Myrinetpakete unterschiedlicher Größe versendet. Die Route wurde so gewählt, daß die Pakete zu einem Switch gesendet wurden und dieser die Pakete zum Empfänger zurück sendet. Für die Verzögerung durch das Switchen wird der von Myricom angegeben „worst case“ Wert verwendet. Er liegt bei $2\mu s$ pro Switch. Die gemessene Verzögerung lag immer unterhalb der Meßtoleranz.

Bei den Vorüberlegungen zu dieser Messung gelangt man zu den gleichen Annahmen,

Roundtripzeit in [μs]	Verlust an Bandbreite in [MByte/s]	Verlust an Bandbreite in [%]
35	44,5	ca. 3,5

Abbildung 5.4: *Leistungsmessung: Aufwand zur Versendung der Steuerpakete*

Rechner	Zeit für Unterbrechungs-behandlung in [μs]	Max. Anzahl an Unter-brechungen in [s^{-1}]
PPro 200	49	ca. 20400
P90	150	ca. 6600

Abbildung 5.5: *Leistungsmessung: Zeiten für die Interruptbehandlung*

wie bei den zuvor beschriebenen Meßreihen. Diese werden durch die erhaltenen Ergebnisse bestätigt (Abb. 5.3).

5.1.3 Das Versenden von Steuerpaketen

Ein wichtiges Kriterium zur Bewertung des Systems ist der Aufwand zur Versendung der Steuerpakete. Für diese Aufgabe muß die Firmware des Schedulers die Nicht-Echtzeit- und die Echtzeit-Listen auswerten. Ist dies geschehen erzeugt sie ein Steuerpaket und sendet dieses einem Host zu. Der Empfänger muß das Paket auswerten und seine Verwaltungsstrukturen durchsuchen. Je nach Art des Steuerpaketes sendet der Host Nicht-Echtzeit- oder Echtzeit-Daten. Im Anschluß übermittelt er seinerseits ein Steuerpaket zur Firmware des Netzwerkschedulers.

In der Tabelle 5.4 sind die Meßergebnisse für einen beschriebenen Durchlauf angegeben. Der Empfängerhost durchsucht seine Listen, hat aber keine Daten zu transferieren. Er sendet daraufhin ein Steuerpaket zurück. Die Zeit wurde vor dem Absenden und nach dem Empfangen der Pakete ermittelt und die Differenz gebildet.

5.2 Messung auf dem Host

5.2.1 Unterbrechungsbehandlung

Um Daten mittels einer blockierenden Empfangensoperation entgegenzunehmen, muß die Zeit für die Verarbeitung von Hardwareunterbrechungen berücksichtigt werden. Die Arbeitsweise wurde im Abschnitt 4.8.3 vorgestellt. Es sind zur Behandlung einer Unterbrechung vier Short-IPC's notwendig. Für diese Messung wurde die Zeitdifferenz zwischen dem Auslösen der Unterbrechung und deren Bestätigung durch den Unterbrechungsbehandlungsthread ermittelt.

Rechner	Socketschicht [μs]	UDP-Schicht [μs]	IP-Schicht [μs]	Summe [μs]
PPro 200	45	35	15	95
P90	243	181	32	456

Abbildung 5.6: *Bearbeitungszeiten für ein Datenpaket maximaler Größe in der Socket-, UDP- und die IP-Schicht*

Für die Ergebnisse (Tabelle 5.5) dieser Messung ist fast ausschließlich die Rechenleistung der System maßgebend, d.h. wie schnell wird der Prozeßkontext gewechselt. Diese Werte haben einen großen Einfluß auf die mögliche Bandbreite pro System.

5.2.2 Socket, UDP und IP

Für die Betriebsmittelanforderung benötigen die Anwendungen Informationen über den Bedarf, der durch die Verwendung der IP, der UDP und der Socketbibliothek entsteht. Dieser wird durch diese Messung bestimmt. Insbesondere ist der Mehraufwand an Rechenleistung von Interesse. Es wird im Folgenden nur auf diese Ressource eingegangen. Es wurden Datagramme maximaler Größe erstellt (Tabelle 5.6).

Der Hauptaufwand in der Socketschicht entsteht durch das Erstellen einer Kopie von den Daten. Die UDP-Schicht muß diese Kopie durchlaufen und die Prüfsumme berechnen. Zusätzlich ist eine gewisse Zeit für die Erstellung des UPD- und des Pseudoheaders nötig. In der IP-Schicht wird lediglich der IP-Header gebildet, die „Scatter-Gather“-Liste erstellt und ein Sendtoken gefüllt. Der Sendetoken wird in einen frei Schacht des Senderingpuffer kopiert.

Kapitel 6

Zusammenfassung und Ausblick

Diese Arbeit stellt ein Modell zur Realisierung der Zusagefähigkeit von Netzwerken vor, welches auf eine Vielzahl von Netzwerken angewendet werden kann. Es basiert auf dem Schema der Echtzeitmodell. Es werden die Vor- und Nachteile der Lösungen aufgezeigt und mögliche Verbesserungvorschläge unterbreitet. Die Implementation des Modells erlaubt den Klientanwendungen direkt mit der Firmware zu kommunizieren. Ermöglicht wird dies durch die hohe Leistungsfähigkeit der Myrinetkarten. Die Lösung bietet einen Netzwerkservice, der eine sehr geringe Belastung für den Hostprozessor darstellt. Der Netzwerktransfer, die Überwachung der Klientanwendungen und die Aufgaben der Bitübertragungsschicht werden durch die Firmware erledigt. Der Datenaustausch zwischen den Klientanwendungen und der Firmware erfolgt mittels der „Scatter-Gatter“-Technik.

Es wird ein Verfahren vorgestellt, wie ein sehr effizienter Nachrichtenaustausch zwischen unabhängige Instanzen mittels gemeinsamen Speichers erfolgen kann. Obwohl das Verfahren in der vorgestellten Lösung zur sicheren Kommunikation zwischen den Klientanwendungen und der Firmware verwendet wird, kann es auch für das Nachrichtenaustausch zwischen Prozessen angewendet werden. Für Netzwerkadapter ohne leistungsfähiger Hardware realisiert ein Serverprozeß die Funktionalität der Firmware. Die Programmierschnittstelle bleibt für die Klientanwendungen unverändert. Das Verfahren ist unabhängig vom Modell der Echtzeitkanäle.

Desweiteren wird eine Möglichkeit aufgezeigt das IP- und das UDP-Protokoll für das DROPS-Projekt zusagefähig zu machen. Diese Funktionalität wird durch eine Bibliothek realisiert. Diese Lösung ermöglicht einen sehr leistungsfähigen Service, da der Kontext der Anwendung nicht verlassen werden muß. Es ist aber eine Abstimmung mit dem IP-Server notwenig.

6.1 Weitere Betrachtungen

Neben den bereits im Abschnitt 4.11 beschriebenen Problemen, sind für die Zukunft noch eine Anzahl weiterer Aufgaben zu lösen:

1. Die Speicherverwaltung in der Firmware erfolgt im Augenblick statisch. Alle Ver-

waltungsstrukturen sind schon zum Zeitpunkt des Bindens der Programmobjekte vorhanden und zugeordnet. Desweiteren haben die Verwaltungsstrukturen der Warteschlangen eine feste Größe. Aus diesem Grund sind die Token für das Senden und Empfangen immer gleich groß, auch wenn nur wenige „Scatter-Gather“-Elemente benutzt werden. Bei einer dynamischen Tokengröße können mehr Daten pro Zeiteinheit versendet werden. Dies bedeutet allerdings einen erhöhten Verwaltungsaufwand.

2. Die momentane Lösung des Servers erlaubt lediglich eine Myrinetkarte pro Rechner.

3. Die Berechnung des Schedules erfolgt auf sehr einfache Art und Weise. Die Möglichkeit des nebenläufigen Versendens von Datenpaketen, der Jitter und die maximale Verzögerung eines Paketes werden nicht berücksichtigt.

4. Um die Leistungsfähigkeit der IP-, UDP- und Socketschicht zu verbessern, sollte die Kommunikation mit den Klientanwendungen über gemeinsamen Speicher erfolgen.

5. Die aktuelle Realisierung des IP-Protokolls erlaubt keine Fragmentierung von Datenpaketen. Es können nur Datenblöcke von maximal 4096 Bytes versendet werden. Es wird kein Routen der Datagramme unterstützt. Die im Abb. 3.1 skizzierte Struktur ist nur zum Teil verfügbar.

6. Für die Zukunft ist es denkbar, daß die Aufgabe des Netzwerkschedulers in die Firmware integiert wird. Hierzu kann der LanAI-Prozessor in den Nutzermodus versetzt werden und die nicht benötigte Rechenzeit wird zur Berechnung eines neuen Schedules verwendet. Es gibt noch keine Erfahrungen bei der Anwendung dieser Fähigkeit.

7. Die augenblicklichen Lösungen bieten sicher noch eine Reihe von Verbesserungen der Leistungsfähigkeit.

Literaturverzeichnis

[VIT98] VITA 26-199X: *Myrinet-on-VME Protokol Specifcation Draft Standard. Draft 1.1,* ***31 August 1998****;*

Verfügbar unter URL: „http://www.vita.com“

[MYR99] Myricom, Inc. 325 N. Santa Anita Ave, Arcadia, CA 91024: *The GM Message Passing System,* ***16. October 1999****;*

Verfügbar unter URL: „http://www.myri.com“

[MYR96] Myricom, Inc. 325 N. Santa Anita Ave, Arcadia, CA 91024: *LANai4.X.doc,* ***17. January 1996****;*

Verfügbar unter URL: „http://www.myri.com“

[Ste94] W. Richard Stevens: *TCP/IP Illustrated, Volume 1, The Protocols, 4.Auflage, Addison-Wesley Publishing Company* ***1994****, ISBN: 0-201-63346-9*

[Tan92] Andrew S. Tanenbaum: *Computer-Netzwerke. Deutsche Ausgabe 2.Auflage, Wolfram Fachverlag* ***1992****, ISBN: 3-925328-79-3*

[FV90] Domenico Ferrari, Dinesh C. Verma: *A Scheme for Real-Time Channel Establishment in Wide-Area Networks. IEEE J. Selected Areas Communication. vol. SAC-8, n. 3, April 1990*

[MIS96] Ashish Mehra, Atri Indiresan und Kang G. Shin, University of Michigan: *Resource Management for Real-Time Communication: Making Theory meet Practice,* ***1996***;

[VZF91] Dinesh C. Verma, Hui Zhang und Domenico Ferrari, University of California at Berkeley: *Delay Jitter Control for Real-Time Communication in a Packet Switching Network* ***1991***;

[BFMM94] Anindo Banerjea, Domenico Ferrari, Bruce A. Mah, Mark Moran u.a., The Tenet Group, University of California at Berkeley: *The Tenet Real-Time Protocol Suite* ***1994***;

[MSA] Asish Mehra, Anees Shaikh, Tarek Abdelzaher, u.a., University of Michigan and IBM T.J. Watson Reasarch Center Yorktown Heights: *Realizing Services for Guaranteed-QoS Communication on a Microkernel Operation System*;

[SD95] Kang G. Shin und Stuard W. Daniel, University of Michigan: *Analysis and Implementation of Hybrid Switching,* ***1995***;

[VC] Chitra Venkatramani Tzi-cker Chiueh, State University of York at Stony Brook: *Supporting Real-Time Traffic on Ethernet*;

[KSZ] Seok-Kyu Kweon, Kang G. Shin und Qin Zheng, University of Michigan und Argon Networks, Inc: *Statistical Real-Time Communication over Ethernet for Manufacturing Automatation Systemy*;

[BKMS] Tsipora Barzilai, Dilip Kandlur, Ashish Mehra udn Debanjan Saha, IBM T.J. Watson Research Center: *Design and Implementation of an RSVP based Quality of Service Architecture for an Integrated Services Internet*

[Rud2000] Marco Rudolph: *Diplom: Eine reservierungsfähige Version von IP über Ethernet für DROPS*, **2000**;

[OSKit10] Univertity of Utah: *The OSKit Project*;

URL: „http://www.cs.utah.edu/flux/oskit/“

[Lie96] J. Liedtke: *L4 reference manual (486, Pentium, PPro). Arbeitspapiere der GMD No.1021, GMD German National Resarch Center for Information Technology, Sankt Augustin,* ***Sebtember 1996****. Auch Forschungsbericht RC20549, IBM T.J. Watson Research Center, Yorktown Heigts, NY,* ***Sep 1996****;*

Verfügbar unter URL: „ftp://borneo.gmd.de/pub/rs/L4/l4refx86.ps.“

[Cro97] Charles Crowley: *Operating Systems: A design-oriented Approach. IRWIN* ***1997****, ISBN: 0-256-15151-2*

[LoHo2000] J. Löser und M. Hohmuth: *Omega0: A portable interface to interrupt hardware for L4 systems,* ***January 2000***

Verfügbar unter URL:

„http://www.os.inf.tu-dresden.de/∼jork/paper/omega0.ps.gz“

[Rei2000] Sven Reigl: *Großer Beleg: Erstellung eines Myrinettreibers für DROPS*, **2000**