Großer Beleg

Erstellung eines Myrinettreibers für DROPS

Sven Reigl
Technische Universität Dresden

13. Oktober 2000

Inhaltsverzeichnis

Kapitel 1

Einleitung

Heutige Netzwerke sind sehr leisungsfähig und bieten hohe Bandbreiten. Die möglichen Übertragungsraten übersteigen die moderner Festplatten deutlich. Das macht Netzwerke für eine zentrale Datenhaltung interessant, da auf diese Weise ein besseres Sichern von Daten möglich ist und sich das Installieren und Warten von Softwarepaketen deutlich vereinfachen läßt. Dies wird dadurch gefördert, daß selbst Arbeitsplatzrechner über schnelle I/O-Busse verfügen.

Moderne Arbeitsplatzrechner bieten eine relativ hohe Rechenleistung und werden immer öfter für sehr rechenintensive Anwendungen eingesetzt. Dazu werden die Rechner zu so genannten Rechnerfarmen organisiert und mit einem Netzwerk verbunden. Der Hauptgrund für diesen Weg ist, daß die Kosten für einen Supercomputer ein Vielfaches von denen mehrere Arbeitsplatzrechner bei gleicher Rechenleistung betragen. Rechenintensive Anwendungen erfordern meist einen hohen Kommunikationsaufwand. Moderne Netzwerke bieten eine ausreichende Leistungsfähigkeit, um dem Nachrichtenaufkommen in einer Rechnerfarm gerecht zu werden.

Diese Arbeit beschäftigt sich mit dem von der Firma Myricom entwickelten Hochgeschwindigkeitsnetzwerk Myrinet. Dabei wurde die Funktionsweise des Myrinets untersucht. Es sollte festgestellt werden, ob Myrinet für die Echtzeitkommunikation tauglich ist. Gleiches gilt für das von Myricom entwickelte Kommunikationsprotokoll GM. Es sollten mögliche Verfahren für die Echtzeitkommunikation diskutiert werden.

Der letzter Teil der Arbeit bestand aus der Portierung des Treibers und GM's für das DROPS-Systems. DROPS ist ein Echtzeitsystem basierend auf L4. Das DROPS-System ist ein Projekt der Technischen Universität Dresden und befindet sich unter Leitung von Hermann Härtig. Bei der Portierung wurde sich stark an dem vorhandenen Linuxtreiber orientiert.

1.1 Über dieses Dokument

Im folgenden Kapitel werden einige Grundlagen vermittelt, auf die sich in den darauf folgenden Kapiteln bezogen wird. Hier sind eine Beschreibung Myrinets und des GM-Protokolls enthalten. Details zur Portierung befinden sich im Kapitel 3. Das Kapitel 4 enthält Ergebnisse durchgeführter Messungen. Eine Bewertung und mögliche Verfahren für die Echtzeitkommunikation mittels Myrinet und GM sind im Kapitel 5 enthalten.

Dieses Dokument wendet sich an Leser mit fundiertem Wissen im Bereich der Informatik und speziell von Betriebssystemen. Zur Vertiefung der Kenntnisse über Betriebssysteme kann z.B das Buch [Cro97] beitragen. Des weiteren ist es von Vorteil, wenn Grundkenntnisse über den Aufbau von Netzwerkarchitekturen und des OSI-Referenzmodells vorliegen. Hierzu kann das Buch [Tan92] von Tanenbaum empfohlen werden. Zur Klärung verwendeter Begriffe befindet sich im Anhang A ein Glossar.

1.2 Erklärung

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

Kapitel 2

Grundlagen

In diesem Kapitel werden Grundlagen für die weiteren Kapitel dargebracht. Im ersten Abschnitt wird kurz auf L4 eingegangen. Ein weiterer Abschnitt stellt die Arbeitsweise Myrinets vor. Der letzte Abschnitt des Kapitels beschäftigt sich mit dem Übertragungsprotokoll GM. GM wurde ebenfalls von der Firma Myricom entwickelt. Es ist im Quelltext verfügbar und darf für nichtkommerzielle Zwecke erweitert und verbreitet werden.

2.1 L4

L4 ist ein von der Gesellschaft für Mathematik und Datenverarbeitung entwickelter Mikrokern. Er entstand unter der Leitung von Jochen Liedke [Lie96]. Die Entwicklung erfolgte für die intelbasierte 80x86-Architektur. Es existieren Portierungen auf MIPS-, DEC-Alpha- und StrongARM-Prozessoren und einigen Digitale Signalprozessoren (DSP's).

L4 bietet folgende Funktionalität an:

- **Adreßraum:** Ein Adreßraum ist eine Abbildung von virtuellen Seiten auf die Kacheln des Hauptspeichers. Diese Umsetzung wird mittels Seitentabellen realisiert. Seiten, die keiner Kachel zugewiesen sind, werden in der Seitentabelle als nicht zugreifbar markiert. Über zusätzliche Attribute können die Seiten vor unberechtigten Zugriff geschützt werden. Greift eine Anwendung auf geschützte oder als nicht zugreifbar markiere Seiten zu, wird vom Prozessor ein Pagefault generiert, der dem Pager der Task gesendet wird.

  Mittels Versenden von Flexpages ist es den Anwendungen möglich, Adreßräume zu manipulieren.

- **Tasks:** Eine Task ist eine Hülle für alle Systemressourcen, in erster Linie für den Adreßraum. In jeder Task können mehrere Threads laufen, die sich die Systemressourcen teilen.

- **Threads:** Ein Thread ist eine aktive Programmausführungseinheit. Jeder Thread hat seine eigenen Statusinformationen, die zu seiner Ausführung notwendig sind, wie Priorität, Register, Stackpointer und Befehlszähler. Es können mehrere Threads in einem Adreßraum agieren. Unter diesen herrscht kein Speicherschutz. Beim Zugriff auf gemeinsame Speicherobjekte ist eine Synchronisation der Threads notwendig.

- **IPC:** Threads können unter L4 mit zwei Nachrichtentypen Daten austauschen. Mittels Short-IPC's können kurze Nachrichten sehr schnell und effizient übertragen werden. Hierzu werden die Daten über Registerinhalte transferiert. Long-IPC's bieten eine mächtigere Kommunikationsmöglichkeit an. In einer Long-IPC können mehrere Strings, 32 Bit Werte und Flexpages gleichzeitig versendet werden.

  Eine Interprozeßkommunikation erfolgt immer zwischen zwei Threads und stellt einen synchronisierten Nachrichtenaustausch mit optionalen Timeouts dar. Ein synchroner Nachrichtenaustausch erfolgt ungepuffert.

- **Flexpages:** Eine Flexpage ist eine Region im virtuellen Adreßraum und besteht aus allen Seiten, die in dieser Region eingeblendet sind. Beim Versenden von Flexpages werden diese Seiten in eine Region des Zieladreßraums eingeblendet. Der Sender bestimmt, welche Region gesendet werden soll, und der Empfänger gibt an, an welche Stelle diese gelangen soll.

- **Hardwareunterbrechungen:** In L4 werden Unterbrechungen durch die Hardware in IPC-Nachrichten übersetzt und an Threads gesendet, die diese behandeln. Diese Threads müssen sich um die Behandlung von Hardwareunterbrechung bewerben.

Für L4 existieren eine Anzahl von Servern und Werkzeuge in Form von Bibliotheken, die von anderen Anwendungen benutzt werden können. Für die Portierung wurden eine Reihe dieser Werkzeuge und Server verwendet:

- VM

- Names

- Omega_0

- OSkit_support
- LibPCI
- DMI

2.2 Myrinet

2.2.1 Allgemeine Beschreibung

Myrinet ist ein Hochgeschwindigkeitsnetzwerk, das von der Firma Myricom entwicket wurde. Dieses Netzwerk erlaubt full-duplex Punkt-zu-Punkt Verbindungen mit einer Übertragungsleistung von bis zu 2560 MBit/s pro Kanal. Gleichzeitig wurde das Netzwerk auf geringe Latenzzeiten optimiert. Der Full-Duplexbetrieb wird durch zwei physische Kanäle gewährleistet.

An jedem Endpunkt (Port) einer Verbindung muß sich ein Switch oder eine Netzwerkkarte befinden. Daten können nur über einen Port versendet bzw. empfangen werden. Eine Netzwerkkarte oder auch Adapterkarte bildet eine Netzwerkkomponente mit einem einzigen Port. Es ist aber möglich, einen Rechner mit mehreren Karten auszustatten. Diese arbeiten, aus der Sicht des Netzwerkes, autonom.

Ein Switch bildet eine Netzwerkkomponente mit mehreren Ports und hat die Aufgabe, Daten von einem Port zu empfangen und an einen anderen Port senden. Es ist jede Netzwerktopologie erlaubt, d.h. Ringe, redundante Verbindungen und direkte Verbindungen zwischen zwei Myrinetkarten dürfen vorkommen.

In diesem Abschnitt werden die Aspekte der Bitübertragungsschicht aus dem OSI-Referenzmodell diskutiert.

2.2.2 Der LANai4-Prozessor

Der LANai-Prozessor ist der Kommunikationsprozessor einer Myrinetadapterkarte [MYR96]. Er ist frei programmierbar. Die Firmware, die den Prozessor steuert, wird bei der Initialisierung in den Speicher der Karte transferiert und gestartet. Es ist möglich, eine eigene Firmware zu programmieren. Dieses Programm kann so angepaßt werden, daß grundlegende Funktionen der Netzwerkarchitektur direkt von der Netzwerkkarte realisiert werden können.

Der LANai-Prozessor hat eine Verarbeitungsbreite von 32 bit, besitzt 24 Mehrzweckregister und 30 Spezialregister. Die ersteren dienen zur temporären Aufnahme von Werten während der Programmausführung. Die Spezialregister sind für besondere Aufgaben vorgesehen, z.B. die Steuerung der DMA-Transfers oder die Steuerung von Hardwareunterbrechungen.

Die Myrinetkarte blendet ihren gesamten Speicher und die meisten ihrer Spezialregister in den I/O-Adreßraum des Hosts ein und kann von dort aus wahlfrei zugegriffen werden. Das Betriebssystem muß nun dafür sorgen, daß nur berechtigte Prozesse auf diesen Bereich zugreifen dürfen.

Der LANai-Prozessor kennt zwei Systemmodi. Der Nutzermodus ist für die Abarbeitung der Firmware verantwortlich und ist unterbrechbar. Der Systemmodus kann nicht unterbrochen werden. Er wird zur Behandlung der eingetroffenen Hardwareunterbrechungen benutzt. Weiterhin unterstützt der Prozessor einen rudimentären Speicherschutzmechanismus. Es ist möglich, Speicherbereiche vor Schreibzugriffen vom externen Bus zu schützen. Hierzu gibt es ein Spezialregister, das angibt, welche Bereiche beschrieben werden können.

Der LANai 4 Chip kann maximal 1 MByte Speicher adressieren. Der aktuellste LANai-Prozessor trägt die Versionsnummer 9 und unterstützt einen Kartenspeicher bis 4MB.

2.2.3 Physische Übertragung

Der Myrinetstandard [VIT98] spezifiziert Bitraten von 640 MBit/s, 1280 MBit/s und 2560 MBit/s. Weiterhin sind 3 Myrinetinterfaces spezifiziert: Backplane (BP), System Area Network (SAN), Local Area Network (LAN). Jedes Interface ist für bestimmte Anwendungsbereiche vorgesehen und unterstützt nur bestimmte Datenraten.

- **System Area Network (SAN):** Das SAN-Interface ist für die Kommunikation innerhalb eines Rechnerclusters vorgesehen. Die maximale Kabellänge für eine Punkt-zu-Punkt Verbindung darf bei Kupferkabel 3 Meter nicht übersteigen. Sollen größere Entfernungen überbrückt werden, muß man auf Glasfaserkabel ausweichen oder einen bzw. mehrere Switches zwischenschalten. Es sind Datenraten von 1280 MBit/s und 2560 MBit/s vorgesehen.

  Ein SAN-Verbindung besitzt 20 Leitungen zur Übertragung von Daten und Steuersignalen, getrennt auf zwei Datenkanäle verbleiben 10 Leitungen pro Kanal. Hiervon sind 8 Leitungen zur Übertragung von Datenwörtern reserviert. Die verbleibenen Signalleitungen dienen zur Steuerung der Übertragung.

  Die Daten/Kontroll-Steuerleitung zeigt an, ob es sich bei dem empfangenen Datenwort um eine Steuerinformation oder um ein Datum handelt. Daten werden einfach an den Sendeport transferiert bzw. als Routinginformation interpretiert. Zusätzlich zu den Datenwörten sind 4 Kontrollwörter reserviert, die Steuerungsaufgaben während der Übertragung wahrnehmen. Das Gap-Symbol wird zur Anzeige von Paketgrenzen benutzt. Ist diese Symbol gesetzt, handelt es sich bei dem letzten Datenbyte um den Paketanhang, und das nächste Byte gehört zu einem neuen Paket und stellt somit eine Route dar.

Das Beat-Symbol dient Überwachungzwecken. Dieses Steuerwort wird in bestimmten Abständen gesendet. Bleibt das Symbol in der erwarteten Periode aus, wird ein interner Alarm ausgelöst. Zwei weitere Symbole sind für eventuelle Erweiterungen reserviert.

Um eine effiziente Datenflußkontrolle zu gewährleisten, gibt es eine spezielle Flußkontrolleitung. Ist die Signalleitung „1“, muß der Sender den Datenausstoß einstellen, bis die Signalleitung auf „0“ gesetzt wurde.

- **Backplane (BP):** Die Eigenschaften des Backplaneinterfaces entsprechen weitestgehend denen des System Area Network. Es wird aber ein Backplane-Connector verwendet.

- **Local Area Network (LAN):** Die Kommunikation im LAN ist für Entfernungen von einigen Metern vorgesehen. Die Datenraten erreichen bei Kabellängen von unter 10 Metern 1280 MBit/s. Sollen größere Strecken überwunden werden, können Bandbreiten von nur 640 MBit/s erreicht werden. Es sind hierbei bis zu 25 Metern Kabellänge erlaubt.

  Im LAN stehen nur noch 18 Leitungen zur Verfügung, d.h. acht Datenleitungen und eine Steuerleitung je Kanal. Dies wird erreicht, indem man die separate Flußkontrolleitung einspart und zwei weitere Steuersymbole einfügt. Das Stop-Symbol läßt den Sendefluß erlöschen, und das Go-Symbol signalisiert dem Sender, daß er die Übertragung fortsetzten kann.

2.2.4 Switching von Myrinetpaketen

Um die Durchlaufzeit eines Datenpakets durch einen Switch zu minimieren, wird das „cut through“-Routingverfahren verwendet. Hierbei muß die Portelektronik ermitteln, ob der Port, auf dem gesendet werden soll, frei ist. Ist dies der Fall, werden die Daten direkt auf diesen Port weitergeleitet. Im anderen Fall wird der Sender aufgefordert, den Datentransfer einzustellen. Hierzu werden die o.g. Flußkontrollmechanismen verwendet. Alle bis dahin aufgelaufenen Daten werden in einem Puffer zwischengespeichert.

Ist der Empfänger zu einem späteren Zeitpunkt bereit, Daten entgegenzunehmen, wird zuerst der Pufferinhalt gesendet und gleichzeitig wird über die Flußkontrollmechanismen dem Sender signalisiert, daß er weitere Daten senden kann. Durch die bekannte Senderate (R) und durch die maximal erlaubte Kabellänge(L) läßt sich die Puffergröße (N) ermitteln, die nötig ist, die Signallaufzeiten (c) des Übertragungsmediums auszugleichen: $N = \frac{2LR}{c} + K$. K ist die Anzahl der Daten, die bereits empfangen wurden, bevor der Trans-

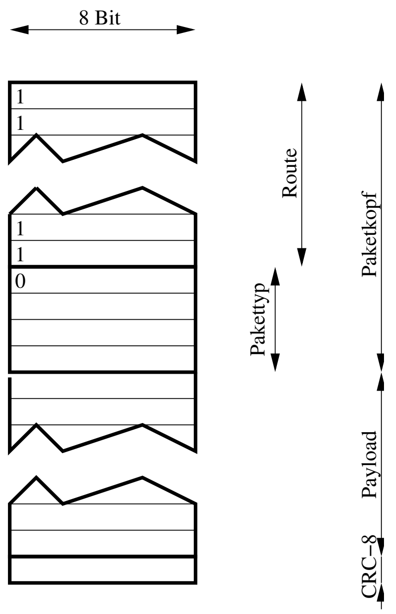

Abbildung 2.1: *Aufbau eines Myrinetpaketes*

ferstop ausgelöst wurde. Sollte sich ein Sender nicht an einen Transferstop halten, darf der Switch das Datenpaket verwerfen.

Ein Transferstop kann auch von einer anderen Instanz in der Route ausgelöst werden, z.B. wenn der Empfänger einen neuen Pufferbereich bereitstellen muß. Dann wird das Stopsignal weitergereicht, und der obige Mechanismus wird angewandt.

2.2.5 Myrinetpakete

Ungewöhnlicherweise ist der Aufbau eines Myrinetpakets auf der Schicht 1 des OSI-Referenzmodels definiert. Dies ist aber notwendig, um ein effizientes Routing zu gewährleisten.

Ein Datenpaket besteht aus 3 Teilen: dem Paketkopf, den Daten und dem Paketanhang in Form einer CRC-Berechnung. Der Paketkopf ist aus der Route und dem Pakettyp zusammengesetzt.

Der Pakettransfer in Myrinet arbeitet „Source-routed“, d.h. der Sender muß in jedem Myrinetpaket die Auflistung der Ports, durch die das Paket gesendet werden soll, angeben. Jeder Switch, den ein Paket durchläuft, entfernt seinen Anteil von der Route, d.h. das erste Byte. Eine Bedingung für eine gültige Route ist die Existenz dieses Ports bzw. die Erreichbarkeit eines initialisierten Gerätes. Ist die Liste der Route leer, ist der Empfänger gefunden. Der Paketempfänger muß ein Hostadapter sein. Ein Switch wird ein an ihn gerichtetes Paket verwerfen.

Der Pakettyp ist 4 Bytes lang. Die ersten 16 Bit werden als Primärtyp verwendet und der Rest als Subtyp innerhalb des Primären. Die primären Typen werden von Myricom verwaltet. Der Typ eines Datenpakets wird auf seinem Weg durch das Netz nicht ausgewertet. Erst die Firmware einer Netzwerkkarte entscheidet, ob sie diesen Pakettyp verarbeiten kann. Das von Myricom entwickelte Nachrichtenprotokoll und dessen Pakettyp heißen GM (siehe nachfolgendes Kapitel).

Um die Route vom Pakettyp unterscheiden zu können, ist das höchstwertigste Bit jedes Bytes der Route „1“ und das höchtwertigste Bit des Pakettypes „0“. Da ein Datenwort 8 Bit lang ist, bleiben 7 Bits zur Codierung des nächsten Ports übrig.

Die Daten oder der Payload können theoretisch $2^{31}-1$ Byte lang sein. Um Laufzeit- und Pufferprobleme zu vermeiden, sollte die Paketgröße maximal 4MByte betragen.

Der Paketanhang ist ein Byte lang. Er wird durch eine CRC-8 Berechnung ermittelt. Ein Problem ergibt sich dadurch, daß jeder Switch ein Byte aus dem Paketkopf entfernt. Folglich stimmt der CRC-Code des Pakets mit den in Wirklichkeit versendet Daten nicht mehr überein. Um dieses Problem zu umgehen, wird für jedes empfangene Paket eine CRC-Berechnung durchgeführt. Ist das Ergebnis 0, so war die Übertragung korrekt, anderenfalls trat ein Kommunikationsfehler auf. Ein Switch reagiert aber nicht auf Übertragungsfehler, da das Paket längst weitergesendet wurde. Er muß aber dafür sorgen, daß der Empfänger der Daten diesen Übertragungsfehler registriert. Dazu wird bei jedem Senden ein neuer CRC berechnet und mit dem zuvor berechneten Ergebnis einer XOR-Operation unterworfen. Bei Kommunikationsfehlern wird der gerade ermittelte CRC verfälscht. Die nächste Instanz wird nun ebenfalls einen Übertragungsfehler feststellen und das selbe Verfahren anwenden. Erst der Empfänger des Paketes wird das Paket verwerfen. Bedauerlicherweise kann bei diesem Verfahren nicht festgestellt werden, an welcher Stelle im Netzwerk der Fehler auftrat.

Aufgrund des Paketaufbaus braucht ein Myrinetswitch keine Sicherungsschicht und kann so sehr einfach gehalten werden. Ein Switch muß für seine Arbeit lediglich das erste und das letzte Byte eines Paketes verändern und die anderen für die CRC-Berechnungen lesen.

2.3 Das Message Passing System GM

2.3.1 Allgemeine Beschreibung

GM ist ein nachrichtenbasiertes Kommunikationsprotokoll für Myrinet. Es setzt sich aus einer Firmware, Treibern und einer Anwendungsschnittstelle zusammen. Beim Entwurf wurde auf hohe Portabilität, eine geringe Prozessorbelastung und eine sehr geringe Latenzzeit bei gleichzeitig hoher Bandbreite geachtet. GM übernimmt die Aufgaben der Sicherungsschicht aus dem OSI-Referenzmodell.

Die Grundaufgabe besteht darin, die Daten in ein gültiges Myrinetpaket zu packen, eine Route zum Empfängerhost zu finden und das Paket zu versenden bzw. ankommende Pakete entgegenzunehmen und die enthaltenen Daten der zuständigen Anwendung bzw. der Vermittlungsschicht zukommen zu lassen. Der Datenaustausch zwischen dem Hauptspeicher des Hosts und des Speichers der Netzwerkkarte erfolgt mittels DMA-Transfers. Bei PCI-Geräte spricht man in diesem Zusammenhang vom Busmastering.

GM ermöglicht eine gesicherte Übertragung zwischen Rechnern. Werden falsch übertragene oder verlorengegangene Pakete festgestellt, sendet GM diese automatisch neu. Treten Netzwerkfehler auf, ermittelt GM selbständig eine neue Route, wenn es eine gibt. Sollten Fehler nicht bearbeitet werden können, werden Fehlermeldungen generiert und an die Anwendung weitergeleitet, die dann adäquat darauf reagieren kann. Praktisch alle Aufgaben der Sicherungsschicht werden durch die Firmware gelöst.

Das GM-Übertragungsprotokoll erlaubt eine theoretische Nachrichtenlängen von $2^{31}-1$ Bytes. Beschränkt wird die Nachrichtenlänge durch den DMA-fähigen Speicherbereich, den das Betriebssystem zur Verfügung stellen kann. In GM werden Nachrichten mit 2 Prioritäten unterstützt. Eine Einhaltung der Sendereihenfolge ist nur innerhalb einer Prioritätsstufe gewährleistet. Weiterhin sagt die Priorität nur aus, in welcher Reihenfolge die Nachrichten vom Host gesendet werden. Ein Switch kann nicht zwischen hochprioren und niederprioren Daten unterscheiden, da er das GM-Protokoll nicht kennt.

2.3.2 Der Mapper und das Erstellen der Topologiekarte

Eine Myrinetkomponente kennt zur Kommunikation nur Ports. Für das Weiterleiten von Paketen, wird aber die komplette Route vom Sender zum Empfänger benötigt. Es stellt sich nun die Frage, wie ein Rechner zu dieser Route gelangt. In GM hält jeder Host eine komplete Topologiekarte des Netzwerks. Soll mit einem Host Daten ausgetauscht werden, kann GM eine Route zu diesem berechnen und als Route in den Paketkopf eintragen. Trifft bis zu einer gewissen Zeit keine Bestätigung ein, berechnet GM eine neue Route zum Zielrechner und sendet das Datenpaket erneut.

Zur Erstellung der Topologiekarte benötigt GM mindestens einen Rechner, auf dem eine Mapper-Prozeß läuft. Es darf aber nur ein Mapper aktiv sein. Aktiver Mapper ist der Prozeß mit der höchsten Priorität bzw. der mit der höchsten Hardwareadresse (MAC-Adresse).

Jeder Rechner im Netzwerk erhält eine eindeutige Identifikationsnummer (ID). Der aktive Mapper verwaltet diese ID´s und vergibt freie an hinzukommende Rechner.

Um Änderungen im Netz festzustellen, sendet der Mapper periodisch spezielle Datenpakete, die das Netzwerk erforschen sollen. Hierfür ist ein primärer Pakettyp reserviert. Der Mapper sendet an alle ihm bekannten bzw. von ihm vermuteten Ports sogenannte Scout-Nachrichten. Erreicht eine solche Botschaft einen Host, sendet dieser eine Antwortnachricht zurück. Dazu ist in der Scout-Botschaft die Route vom Host zum Mapper enthalten, auf der die Antwort gesendet werden soll. Der Mapper vermerkt diese Hosts.

Trifft eine Scout-Botschaft auf einen Switch, wird sie von diesem verworfen. Wird in einer gewissen Zeit keine Rückmeldung empfangen, sendet der Mapper die Nachricht erneut. Nach mehreren Versuchen sendet der Mapper sogenannte Switch-Nachrichten. Bei diesen wird davon ausgegangen, daß es sich bei dem Zielknoten um einen Switch handelt und dieser nicht antworten kann. Dazu baut der Mapper ein Paket, dessen Route durch den Switch verläuft und dessen Empfänger er selber ist. Empfängt er dieses Paket, ist ein weiterer Switch gefunden und wird vermerkt. Im anderen Fall wurde ein nicht existenter oder leerer Port gefunden.

Wurde bei der Suche ein neuer Host gefunden, berechnet der Mapper alle Routen zu diesem Host und versendet diese an schon bekannte Netzteilnehmer. Zusätzlich informiert er den hinzukommenden Host über die bestehende Topologiekarte.

Die erstellte Topologiekarte des Netzwerkes enthält weiterhin den Rechnernamen und die Hardwareadresse der Netzwerkkarte (MAC-Adresse). Zur Kommunikation mit einem Host ist aber die Identifikationsnummer zwingend erforderlich.

2.3.3 GM-Ports

Der Nachrichtenaustausch erfolgt in GM über Kommunikationsendpunkte, Ports genannt. Ein Port kann immer nur einem Prozeß „gehören“, allerdings darf ein Prozeß mehrere Ports „besitzen“. GM unterstützt momentan 8 Ports pro Adapterkarte. Ein GM-Port ist kein Myrinetport. Der Hauptunter-

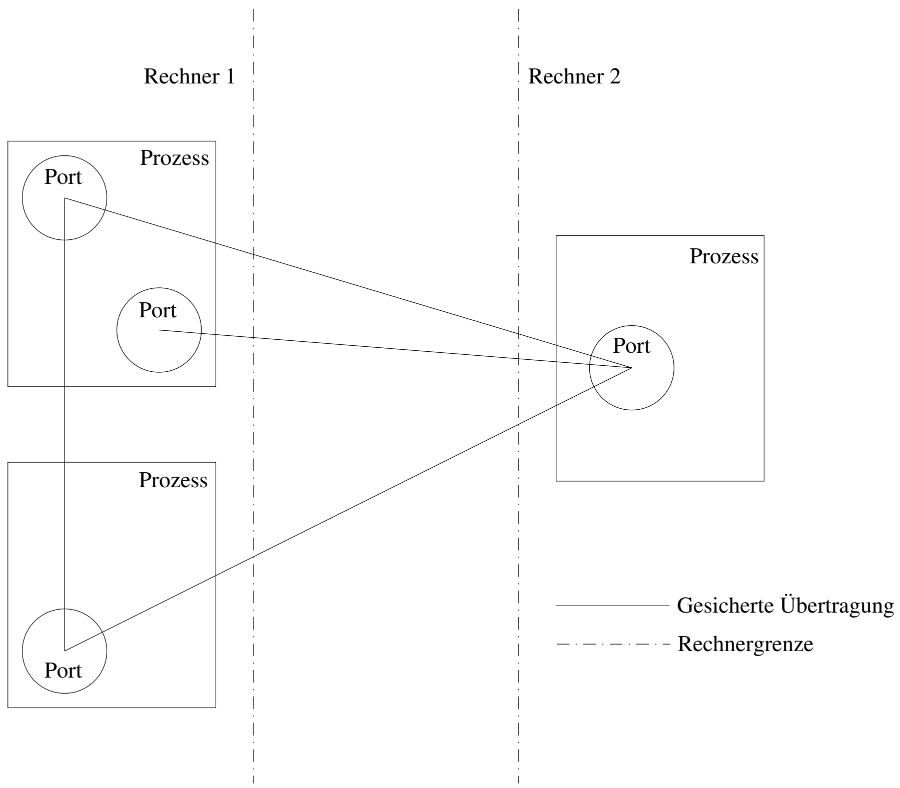

Abbildung 2.2: *Die Kommunikation erfolgt in GM über Ports*

schied der Portarten besteht darin, daß ein Myrinetport mit einem physischen Connektor assoziiert wird und zum Routen im Netz zwingend erforderlich ist. Ein GM-Port stellt einen logischen Kommunikationspunkt dar und ist eine spezifische Konstruktion im GM-Protokoll. In diesem Abschnitt ist mit dem Begriff Port ein GM-Port gemeint.

Um einen möglichst effizienten Datentransfer zu gewährleisten, ist der Speicher der Netzwerkkarte statisch aufgeteilt. Für jeden GM-Port ist ein seperater Bereich reserviert. Jeder Bereich enthält eine Sende- und eine Empfangstokenwarteschlange. Diese werden les- und schreibbar in den Adreßraum des Portbesitzers eingeblendet. Um zu verhindern, daß der Besitzer beliebige physische Speicherbereiche versenden kann, werden in den Warteschlangen virtuelle Adressen verwendet. Dazu ist es notwendig, für jeden Port eine eigene Übersetzungtabelle von virtuellen auf physische Adressen zu halten. Diese Tabellen werden vor nichtpriviligierten Clients verborgen gehalten. Weiterhin darf nur der Treiber diese Umrechnungstabelle verändern. Das Betriebssystem muß dafür sorgen, daß die Speicherseiten für den Datentransfer nicht ausgelagert werden und sich die physischen Seiten nicht ändern (pinnen von Speicherseiten). Der Vorteil ist nun, daß der Client seine Warteschlangen selbständig verwalten kann. Der Client kann ohne Kontextwechsel eine Sendeoperation durchführen oder einen Empfangsbereich bereitstellen.

Für den Datenempfang wird für jeden Port eine Eventwarteschlange angelegt. Diese befindet sich im Betriebssystemkern und wird einem Portbesitzer ebenfalls eingeblendet.

Das Verfahren funktioniert im Multiclientbetrieb, weil alle Clientanwendungen auf unterschiedlichen Speicherbereichen arbeiten und somit keine Synchronisation notwendig ist. In diesem Lösungsansatz ist es keinem Client möglich, physische Speicherseiten anzugeben. Sollte eine Anwendung für Puffer virtuelle Adressen benennen, die keine gültige physische Seiten besitzen, kann die Firmware diese nicht auflösen und gibt Fehlermeldungen zurück.

Da alle Pufferseiten gepinnt und in der Übersetzungstabelle vermerkt werden müssen, sind das Anfordern und das Freilassen von Speicherbereichen teuer. Eine Anwendung sollte sich daher während ihrer Initialisierungsphase alle Puffer besorgen, die sie für ihre Arbeit benötigt und diese immer wieder verwenden.

Gibt eine Anwendung einen Port frei oder wird diese beendet, werden alle eingeblendeten Seiten entfernt und die Speicherseiten als auslagerbar markiert. Der Port und damit alle Ressource können von einer anderen Anwendung angefordert werden.

Nachteilig an diesem Verfahren ist die Möglichkeit einer Clientanwendung, beliebig viele Speicherseiten pinnen zu können. Auf diese Weise lassen sich leicht „denial of service"-Angriffe konstruieren.

2.3.4 Das GM-Programmiermodell

2.3.4.1 Nachrichtenübertragung

Eine Anwendung muß, um Nachrichten versenden oder empfangen zu können, einen Port öffnen und kann dann über diesen Port Nachrichten an einen beliebigen Port eines Netzwerkhosts senden bzw. über diesen Nachrichten von einem beliebigen anderen Port empfangen.

Der Nachrichtenaustausch kann nur in bzw. aus DMA-fähigen Speicher heraus geschehen. Hierzu kann vom Treiber ein DMA-fähiger Speicherbereich angefordert werden, der zu einem späteren Zeitpunkt wieder freigegeben werden kann.

In GM wird streng zwischen der Größe (size) und der Länge (length) einer Nachricht unterschieden. Mit der Länge einer Nachricht ist die Anzahl der Bytes der zu übermittelnen Nachricht gemeint. Ferner sagt die Länge

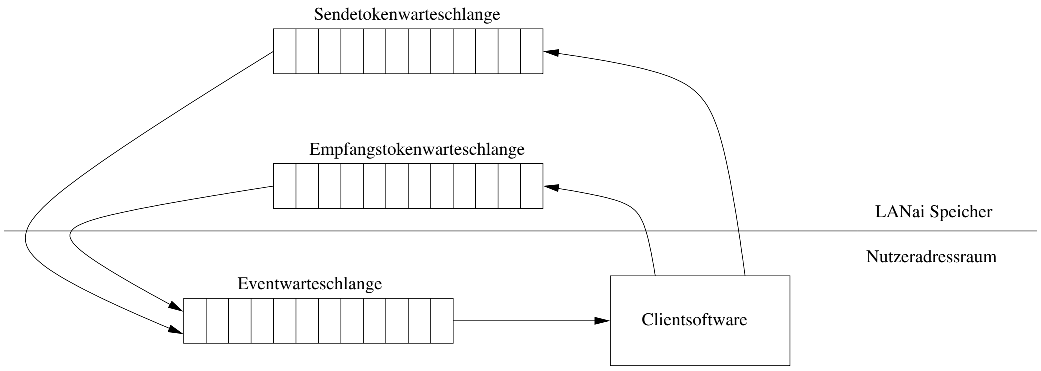

Abbildung 2.3: *Schematisch: Tokenfluß im GM*

aus, wieviele Datenwörter über das Medium gesendet werden. Die Größe einer Nachricht charakterisiert die Puffergröße, aus der die Nachricht gesendet wurde und gleichzeitig die Puffergröße, die empfängerseitig bereit stehen muß, um diese Nachricht zu empfangen. Dabei muß die Puffergröße 2^n Byte entsprechen. Ohne diese Verfahren wäre es aufwendig zu ermitteln, ob der Empfänger genügend Puffer bereitstellen kann, um eine beliebig lange Nachricht zu empfangen. Eine Anwendung kann bei diesem Lösungsansatz auch ausdrücklich angeben, welche Nachrichtengrößen verarbeitet werden können. Dazu muß lediglich eine Maske mit den akzeptierten Größen registriert werden.

2.3.4.2 Die Nachrichtenkoordinierung mittels Token

Sowohl das Senden von Nachrichten als auch das Empfangen erfolgt über ein implizites Tokenverfahren. Ein Token repräsentiert eine Datenstruktur, die in die Sende- bzw. Empfangstokenwarteschlange der Myrinetkarte kopiert wird. Jeder GM-Port hat seperate Warteschlangen.

Ein Sendetoken enthält die Empfänger-ID, den Empfängerport, die virtuelle Adresse des Nachrichtenpuffers, die Größe und Länge der Nachricht und die Priorität der Botschaft. Um Manipulationen zu verhindern, werden erst auf der Netzwerkkarte die Senderinformationen zur Nachricht hinzugefügt. Auf diese Art und Weise stellt GM sicher, daß ein Empfänger korrekte Informationen über die Herkunft eines Datenpaketes erhält. Ein Empfangstoken enthält einen Zeiger auf den bereitgestellten Puffer, dessen Größe und für welche Priorität er vorgesehen ist.

Jede Anwendung erhält beim Öffnen eines Ports sämtlichen Sende- und Empfangstoken. Die Applikation kann die Token beliebig verwenden.

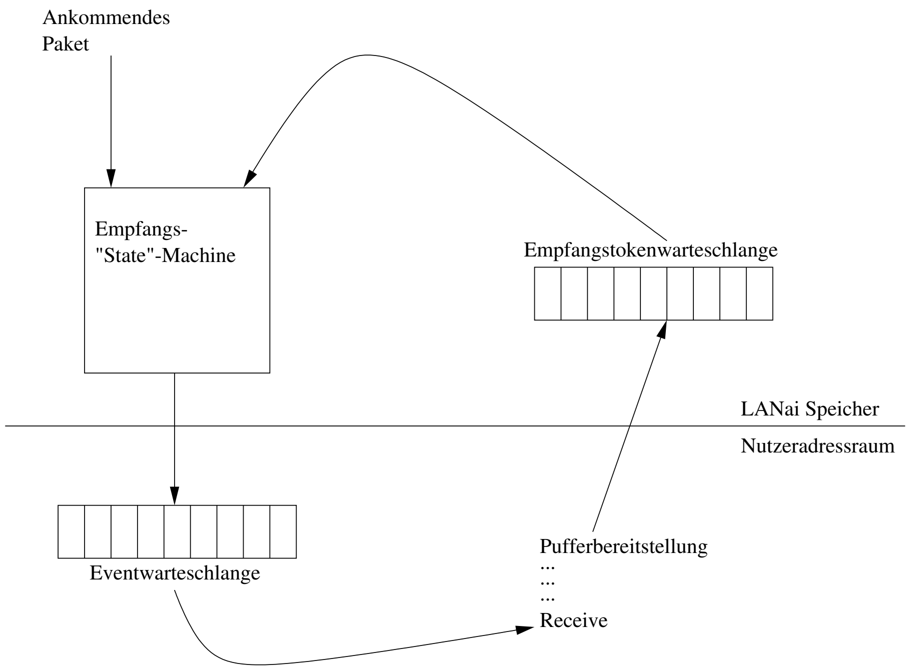

Abbildung 2.4: *Das Bereitstellen von Empfangspuffer und das Empfangen von Nachrichten bzw. von Events in GM. In den erhaltenen Ereignissen können sich auch Netzwerkfehler verbergen, die von der Clientanwendung nicht verarbeitet werden müssen.*

Eine Anwendung hat vor dem Empfangen die Aufgabe, Nachrichtenpuffer für alle akzeptierten Größen bereitzustellen und hat auch dafür zu sorgen, daß immer Pufferbereiche vorhanden sind. GM benutzt nicht automatisch die nächst größeren Puffer, wenn für andere Größen keine Puffer bereitstehen. Dabei ist zu beachten, daß für jede Priorität separate Puffer bereitzustellen sind.

Bei der Bereitstellung eines Puffers wird ein Empfangstoken an GM übergeben. Wenn eine Nachricht eingetroffen ist, wird von GM ein Empfangsereignis generiert und in die Ereigniswarteschlange des Ports eingereiht. Führt die Anwendung eine Empfangsfunktion aus, wird der Token an die Anwendung zurückgegeben, die diesen wieder vergeben kann.

Das Senden erfolgt ähnlich. Nachdem ein Sendepuffer angefordert und gefüllt wurde, wird bei einem Sendeaufruf ein Sendetoken an GM übergeben. Ist die Bestätigung des Empfängers angekommen, wird die Applikation mittels eines Ereignisses benachrichtigt und ihr der Token zurückgegeben.

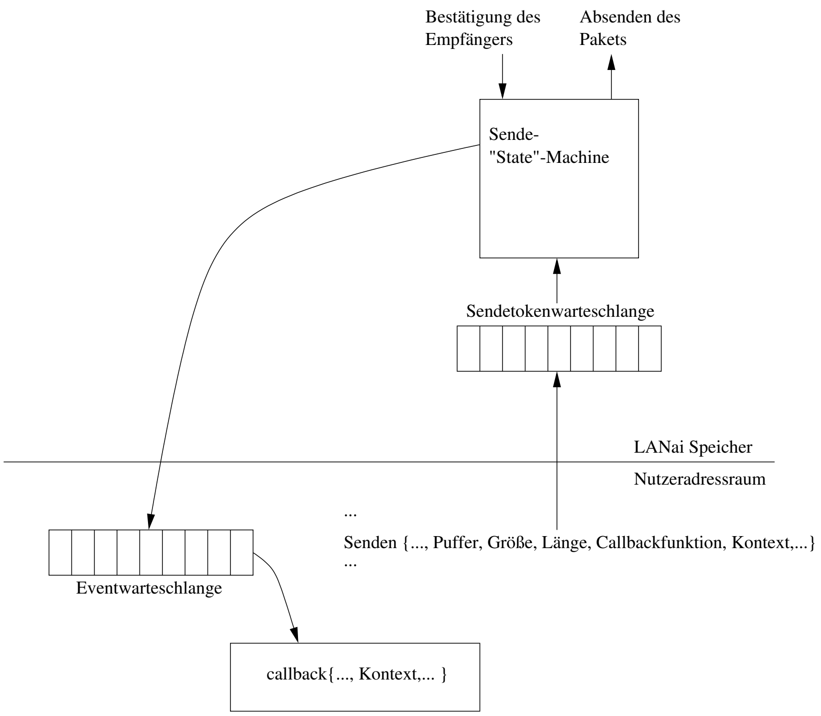

Abbildung 2.5: *Der Sendevorgang, das Empfangen der Bestätigungsnachricht und das Ausführen der Callbackfunktion. Der Aufruf der Callbackfunktion erfolgt implizit über das Auswerten von Ereignissen (Events).*

2.3.4.3 Events und Callbackfunktionen

Die Benachrichtigung einer Anwendung über Ereignisse erfolgt über eine Eventwarteschlange und mittels Callbackfunktionen. Bei der Ausführung einer Empfangsfunktion wird diese Warteschlage abgefragt, und es wird von GM eine Ereignisstruktur zurückgeliefert. Die Abfrage der Warteschlange kann mittels einer blockierenden oder einer nichtblockierenden Funktion geschehen. Bei einer nichtblockierenden Abfrage kann die Ereigniswarteschlange leer sein. GM generiert dann ein leeres Event, das an die Anwendung zurückgeliefert wird. Sollte sich beim Aufruf einer blockierende Empfangsoperation kein Ereignis in der Warteschlange befinden, blockiert der rufende Prozeß. Trifft zu einem späteren Zeitpunkt ein Ereignis ein, reaktiviert GM den Prozeß auf und übergibt ihm die Ereignisstruktur.

Es gibt in GM 30 verschiedene Ereignisse. Ein Prozeß kann diese auswerten, ist aber nicht dazu gezwungen. Ereignisse, die die Anwendung nicht behandeln möchte, können einfach an GM zurückgegeben werden. GM führt

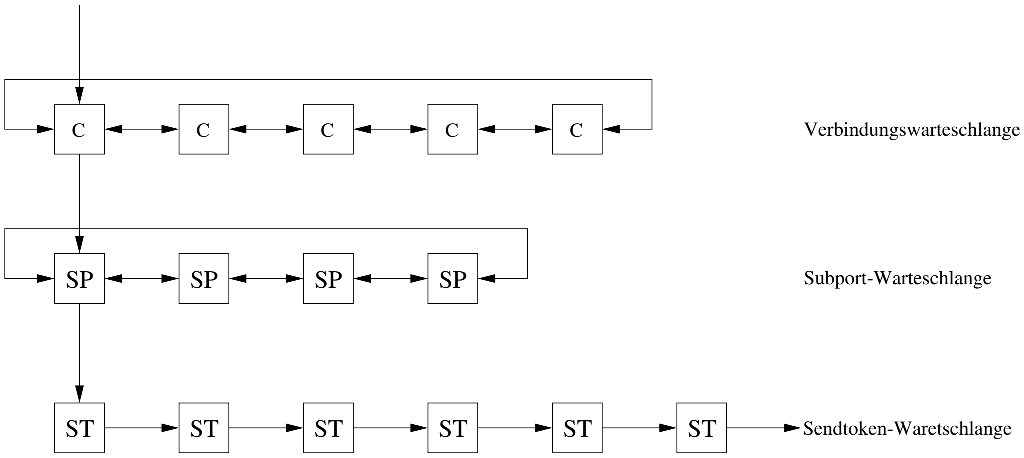

Abbildung 2.6: *Aufbau der Sendewarteschlange in der GM-Firmware*

dann Standardfunktionen aus oder ruft eine Callbackfunktion des Clients auf. Jede sinnvolle Anwendung sollte aber wenigstens auf den Empfang von Datenpaketen reagieren und GM einen neuen Pufferbereich zur Verfügung stellen.

Callbackfunktionen müssen nicht implementiert werden. In einem solchen Fall verwendet GM eigene Funktionen, die ein ungestörtes Weiterarbeiten der Anwendung erlauben. Die Sende-Callbackfunktion wird von GM nach einer erfolgreichen Sendeoperation aufgerufen. Hier erhält der Client seinen Sendetoken zurück und kann eine neue Sendeoperation auslösen. Über das Alarm-Callback informiert GM die Anwendung über aufgetretene Störungen. Eine Anwendung kann so auf Fehlersituationen reagieren.

2.3.5 Die Firmware

Ein großer Vorteil der Myrinetkarten ist, daß ein Nutzer eine eigene Firmware schreiben kann. Diese wird bei der Initialisierung auf die Karte transferiert und kann dazu benutzt werden, Protokolle direkt auf der Karte zu realisieren. Die Firmware kann an die eigenen Erfordernisse angepaßt werden.

Unter GM ist der Speicherbereich der Netzwerkkarte statisch aufgeteilt. Zur Initialisierungszeit werden die Speicherbereiche für die Sende- und Empfangstoken der Ports und deren Übersetzungtabellen reserviert.

Zum Versenden aller Nachrichten wird eine Sendewarteschlage verwendet. Diese Sendewarteschlage ist eine doppelt verkettete Liste von Verbindungen (siehe Abb. 2.6). Verbindungen sind in diesem Zusammenhang Nachrichten, die an denselben Zielknoten gesendet werden sollen. Die Nachrichten an einen

Host können natürlich von unterschiedlichen Ports stammen. Um hochpriore Daten zu bevorzugen, werden sog. Subports gebildet. Ein Subport setzt sich aus einem Port/Prioritätspaar zusammen. Über jeden Subport werden die einzelnen Sendetoken verwaltet. Jeder Sendetoken repräsentiert einen verbleibenden Sendetransfer.

Ein Sendetoken wird erst entfernt, wenn die zugehörige Empfangsbestätigung eingetroffen ist und nicht, wenn die Sendelänge der Wert Null erreicht hat. Sollte der Empfänger einen Sendefehler anzeigen oder ist ein Sendetimeout eingetreten, muß die Sendelänge nur auf den Originalwert gesetzt werden, und das Kontollprogramm wird den Sendeauftrag erneut senden.

Um ein gerechtes Senden zu realisieren, wählt das Kontollprogramm nach einer Sendeoperation einen neuen Subport und anschließend eine neue Verbindung. Hierbei werden hochpriore Sendeaufträge bevorzugt.

Bei einer Empfangsoperation wird als erstes der Empfangsport ermittelt. Es wird festgestellt, ob die Anwendung Nachrichten der empfangenen Größe verarbeiten möchte, gegebenfalls wird eine Fehlernachricht an den Sender geschickt. Es wird ein Empfangstoken der nötigen Größe gesucht und die Daten werden mittels eines DMA-Transfers in den Nachrichtenpuffer des Tokens transferiert.

Blockiert ein Client beim Empfangen einer Nachricht, wird von der Firmware eine Hardwareunterbrechung ausgelöst. GM bestätigt die Hardwareunterbrechung, generiert ein Empfangsereignis und reiht dieses in die Ereigniswarteschlange des Clients ein. Der Client wird reaktiviert und ihm ein Zeiger auf das neue Event übergeben. Für nichtblockierendes Empfangen wird keine Hardwareunterbrechnung erzeugt.

Kapitel 3

Portierung von GM auf L4

3.1 Der Treiber

Der GM Treiber besteht aus einer Menge von systemunabhängigen Funktionen. In diesen werden die allgemeinen Verwaltungsaufgaben erledigt, die Eventwarteschlangen der Clientanwendungen aktualisiert, die Hardwareressourcen verwaltet und auf Hardwareunterbrechungen reagiert. Die systemunabhängigen Funktionen benötigen aber systemabhängige, um mit den Besonderheiten des Betriebssystems umgehen zu können. Zu einer Portierung müssen lediglich die systemabhängigen Funktionen angepaßt werden. Diese sind von Myricom dokumentiert.

In einem monolithischen Betriebssystem arbeitet der Treiber im Systemkontext und hat in diesem Zugriff auf alle Ressourcen. Unter L4 muß der Treiber als Server im Nutzermodus arbeiten. Dieser Prozeß muß anderen Prozessen, den Clients, Services bereitstellen, die die gleiche Funktionalität besitzen, wie die Betriebssystemfunktionen in einem monolithischen System (siehe Client-Server Kommunikation).

Um GM portieren zu können, werden einige Fähigkeiten vom Betriebssystem gefordert bzw. die Bereitstellung äquivalenter Funktionalität:

1. Um alle Sicherheitsmöglichkeiten von GM nutzen zu können, muß das Betriebssystem virtuellen Speicher bereitstellen können. Dies ist aber keine notwendige Forderung, d.h. in Systemen ohne virtuellen Speicher muß lediglich auf diese Sicherheitsmechanismen verzichtet werden.

2. Speicher, der im Kern angefordert wird, muß gepinnt sein. Es muß sich dabei um einen kontinuierlichen Speicherbereich handeln, und dieser Bereich muß für DMA tauglich sein.

3. Das System muß zum Ausführen eines DMA-Transfers aus oder in Speicherseiten eines Nutzerprozesses ein unbegrenzt langes Pinnen dieser Speicherseiten erlauben. Anderenfalls müssen Speicherseiten im Kern angefordert werden, die dann in den Nutzeradreßraum eingeblendet werden müssen.
4. Es müssen Mechanismen zur Einblendung von PCI-I/O-Adressen in den Speicherbereich des Kerns und der Clients vorhanden sein.
5. Zur Synchronisierung von Zugriffen auf Speicherobjekte müssen Kernmutexe und interruptsichere Semaphore bereitstehen.
6. Zur Ermittlung von Timeouts muß möglich sein, einen Timer zu setzen.
7. Es werden Möglichkeiten zur Bestimmung einer Seitengröße und des Rechnernames benötigt.
8. Zur Kommunikation zwischen Treiber und Client werden Unixsystemrufe verwendet. Benötigt werden open, close, mmap und ioctl.
9. Zur Behandlung der Hardwareunterbrechungen der Myrinetkarte ist es notwendig, einen Interrupthandler zu installieren.

In den folgende Abschnitten werden mögliche Lösungen für die obenstehende Forderungen diskutiert. Es sollen hierbei die Vor- und Nachteile der Designentscheidungen dargelegt und eventuell andere Lösungsansätze gezeigt werden.

3.1.1 Die Speicherverwaltung

Unter L4 arbeiten alle Tasks in separaten Adreßräumen, und L4 sorgt für alle notwendigen Speicherschutzmechanismen, so daß alle sicherheitsrelevanten Fähigkeiten vom GM zum Tragen kommen können. Eine Ausnahme sind Hauptspeicherseiten, die in Adreßräume mehrerer Tasks eingeblendet sind.

Die Aufgabe der Speicherverwaltung übernimmt unter L4 ein Userlevel-Pager. Dieser Pager muß alle o.g. Eigenschaften erfüllen, d.h. der Pager muß kontinuierliche Speicherbereiche beschaffen können, und dieser hat auch dafür zu sorgen, daß diese nicht ausgelagert werden oder sich die Zuordnung von virtuellen zu physischen Adressen ändert. Da ein Auslagern von Hauptspeicherkacheln auf eine Festplatte momentan nicht implementiert oder nicht verwendet wird, treten solche Ereignisse nicht auf. Es steht aber einem Pager

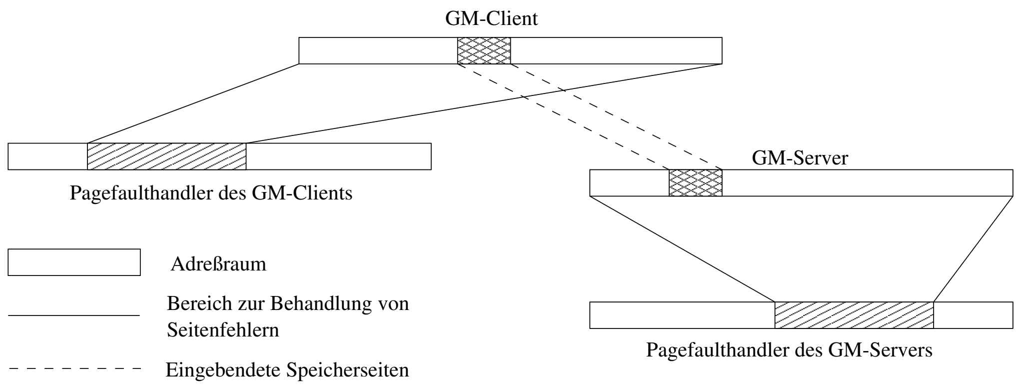

Abbildung 3.1: *Schematische Darstellung der Adreßräume eines GM-Clients und des GM-Servers*

frei, einem Client jederzeit Speicherseiten zu entziehen, somit bleibt das Kernproblem der gepinnten Speicherseiten erhalten. Im Augenblick gibt es keinen Pager, der Speicherseiten pinnen kann.

Bei der Implementierung des GM-Servers wurde als Pager VM [Gru99] verwendet, da dieser Servicefunktionen zum Anfordern von zusammenhängenden Speicherbereichen und zur Ermittlung von physischen Adressen bereitstellt. Um eine korrekte Arbeit der letzteren Funktion zu gewährleisten, ist es aber notwendig, das „OSkit-malloc“ anzupassen. Die Originalversion fordert den Pager einer Anwendung auf, eine Hauptspeicherseite zu besorgen und diese als Flexpage an die Anwendung zu senden. Der Pager kennt bei dieser Lösung aber die virtuelle Adresse nicht, an der die Seite eingeblendet wird, so daß ein Aufruf der Übersetzungsfunktion VM's fehlschlägt. Wird stattdessen an der Stelle, an die die Seite gelangen soll, ein Pagefault generiert, kann VM eine gültige Abbildung schaffen.

Jeder Client kann den Pager seiner Wahl verwenden. Dieser Ansatz erlaubt die größte Flexibilität und die geringsten Einschränkungen für die beteiligten Clients. Leider birgt diese Ansatz einige Probleme. Da die Servertask keine direkte Kontrolle über ihre Clients hat, besitzt sie auch keinen Zugriff auf deren Seitentabellen. Der Server kann somit nicht garantieren, daß sich auch physische Seiten an viruellen Adressen befinden, die ein Client angibt. Ebensowenig kann festgestellt werden, ob die Seiten gepinnt sind oder sich diese mit der Zeit ändern. Zweitens kann der Server die Abbildung von virtuellen auf physische Adressen nicht direkt bestimmen, er muß immer seine Clients bitten, diese Abbildung für ihn zu erwirken.

Eine Möglichkeit, diese Mängel zu beseitigen, ist, den GM-Server gleichzeitig zum Pager des Clients bzw. eines bestimmten Adreßbereiches zu machen. Zu letzterem ist ein Regionenverwalter nötig, der zum Standardpager des Clients ernannt wird. Nachdem der Regionenverwalter einen Seitenfehler erhalten hat, sendet er diesen dem Pager der Region zu. Es muß zusätzlich ein Protokoll existieren, über das sich der Client und der Server über Regionen einigen.

Ein anderer Lösungsansatz ist, mittels Versenden von Flexpages Speicherseiten in den Adreßraum eines Clients einzublenden. Es ist hierfür nur ein geringer Kommunikationsaufwand nötig, und der GM-Server muß keine Seitenfehler behandeln. Der Client muß dazu den Adreßbereich angeben, in den er Seiten eingeblendet haben möchte. Der Server kann diesen Bereich nicht kontollieren. Da aber nur er die Übersetzungtabelle eines Clients schreiben darf, würde es einer Anwendung nichts nützen, beim Senden oder bei der Bereitstellung von Pufferbereichen andere virtuelle Seiten anzugeben. Die Firmware kann unbekannte virtuelle Adressen nicht auflösen.

Unter der 80x86-Architektur von Intel ist ein DMA-Transfer lediglich von jeder auf 32 Bit ausgerichteten Adresse möglich, d.h. die letzten beiden Adreßbits müssen Null seien.

3.1.2 Das Einblenden von IO-Adreßbereichen

Die libPCI enthält Servicefunktionen, mit denen alle PCI-Geräte ermittelt und konfiguriert werden können. Mit der Funktion „pci_find_device" kann nach Geräten eines bestimmten Typs und eines Herstellers gesucht werden. Alle Gerätearten und Gerätehersteller sind im PCI-Standard aufgeführt. Als Ergebnis einer Suche erhält man eine Struktur, in der alle nötigen Informationen des Gerätes enthalten sind. Bei erneuten Aufrufen dieser Testfunktion werden weitere Strukturen zurückgeliefert, wenn die Suchanforderungen auf mehrere PCI-Geräte zutrifft. Aus einer solchen Struktur können die benötigten I/O-Bereiche ermittelt werden.

Um unter L4 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 L4 nicht, seitengroße Bereiche aus einer Superpage als Flexpage zu versenden. Es ist nur möglich, die gesamte Superpage als Flexpage zu versenden. Da GM aber ein schreib-/lesbares Weiterleiten von Teilbereichen des eingeblendeten Adreßbereichs erfordert, kommt es hier zu einem Sicherheitsproblem. Nach dem Einblenden der Superpage ist es einem Client 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.

3.1.3 Synchronisation

Zur Sychronisation von Threads werden L4-IPC's verwendet. Da IPC's unter L4 synchron ablaufen, verlaufen auch die Clientanfragen geordnet und synchron. Der GM-Server kann dann die geordneten Anfragen bearbeitet.

3.1.4 ioctl

ioctl ist eine andere Möglichkeit eines Clients, Daten über der Treiber mit der Myrinetkarte auszutauschen. Dabei handelt es sich um beliebig lange Daten, meistens sind sie aber nur einige Bytes lang. Um ein Versenden beliebig langer Daten zu gewähren, wird zuerst eine Short-IPC an den Server gesendet, in der die Länge der zu sendenden bzw. zu empfangenen Daten angegeben wird. Der Server kann jetzt einen Puffer reservieren, in den mittels einer Long-IPC die Daten transferiert werden. Die Ergebnisdaten werden in denselben Puffer kopiert und an den Client zurückgesendet.

Um auf Nachrichtenaustausch mittels Long-IPC zu verzichten, wäre es aber möglich, einen speziellen ioctl-Bereich für jede Clienttask zu halten. Hierzu könnten ein oder zwei Speicherseiten in den Adreßraum des Clients eingeblendet werden. Mittels kurzer IPC-Botschaften kann die Synchronisation von Server und Clientanwendung erfolgen. Sollte dieser Bereich zur Aufnahme der Daten nicht genügen, kann noch auf die alte Variante zurückgegriffen werden.

3.1.5 Interruptbehandlung

Omega$_0$ ist ein Interruptbehandlungsserver für L4 [LoHo2000]. Er bietet Treibern Funktionen zur Bestätigung von Hardwareunterbrechungen und erlaubt ein Interruptsharing, d.h. mehrere Geräte können sich einen Hardwareinterrupt teilen. Interruptsharing ist für PCI-Geräte üblich. Ein Thread kann sich bei Omega$_0$ um einen oder mehrere Hardwareinterrupts bewerben. Tritt eine bestimmte Hardwareunterbrechung ein, sendet Omega$_0$ jedem Treiberserver für diesen Interrupt eine IPC.

Hardwareunterbrechungen treten nur bei Fehlern im Netzbetrieb auf oder wenn ein Client ein blockierendes Empfangen durchführt. Soll ein Client reaktiviert werden, ruft der unabhängige Teil des Treibers eine systemabhängige Funktion auf. Aus den Aufrufparametern ist ersichtlich, um welchen Port es sich handelt. Es wird der zugehörige Client ermittelt und ihm eine Short-IPC gesendet.

Um möglichst schnell auf eine Meldung von Omega$_0$ reagieren zu können, gibt es einen eigenen Interruptbehandlungsthread, der im selben Adreßraum arbeitet wie der Server. Der Interruptthread bestätigt ankommende Hardwareunterbrechungen beim Omega_0-Server. Soll ein Client reaktiviert werden, sendet er ihm die Short-IPC. Trat ein Fehler auf, ruft er eine spezielle GM-Funktionen auf.

3.1.6 Restliche Funktionen

Zu Zeitmessungen können die interne Uhr des Rechners oder das Eventregister des Prozessors ausgelesen werden. Soll ein Thread für eine bestimmte Zeit blockiert werden, kann man „l4_sleep“ verwenden.

Zur Bestimmung einer Seitengröße verwendet der GM-Server das Makro „L4_PAGESIZE“.

Um den Rechnernamen ermitteln zu können, gibt es in L4 momentan keine Möglichkeit. Es ist aber denkbar, den Namensserver „Names“ um diese Funktionalität zu erweitern.

3.2 Client-Server Kommunikation

Die notwendigen Verwaltungsfunktionen und Datentypen sind in einer Bibliothek und einer Includedatei (libgm.a und gm.h) enthalten. GM fordert für die Client-Server-Kommunikation nur 4 Systemrufe: open, close, mmap und ioctl. Read und write werden nicht unterstützt. Unter L4 müssen diese Betriebssystemfunktionen nachgebildet werden. Hierzu muß zusätzlich die „gm_clientlib“ eingebunden werden. In ihr sind alle benötigten Systemfunktionen durch IPC-Rufe zum GM-Server ersetzt.

Um ein effektives Versenden von Daten und Botschaften an den GM-Server zu gewährleisten und möglichst wenige Long-IPC's versenden zu müssen, wurde das DMI-Nachrichtenprotokoll verwendet und erweitert. DMI steht für „Drops Message Interface“. In DMI werden eine Protokollnummer, eine Funktionsnummer innerhalb des Protokolls und ein 16 bit langer Wert zu einem 32 bit Wert verknüpft. Da bei einer Short-IPC zwei 32 bit Werte versendet werden, kann noch zusätzlich ein 32 Bit langes Datum versendet werden. Über diesen Konstrukt werden alle Servicefunktionen des GM-Servers aufgerufen.

3.2.1 open

Die GM-Bibliothek ruft diese Funktion auf, um dem Treiber mitzuteilen, daß eine Anwendung eine Myrinetkarte benutzen möchte. Die libgm übergibt dazu einen Pfad und die gewünschte Zugriffsart. Im Augenblick wird unter L4 kein Dateisystem benutzt. Die Pfadangabe hat somit keine direkte Bedeutung. Sie kann aber trotzdem dazu benutzt werden, dem Server mitzuteilen, um welches Gerät es sich handelt. Die Zugriffsart wird ignoriert, da immer Les- und Schreibfähigkeit erforderlich sind.

Der Server vergibt für jedes open eine interne Nummer, diese wird als Rückgabewert des open-Befehls verwendet. Auf diese Weise kann ein Client mehrere GM-Ports benutzen. Diese Nummer wird vom Client als Filedescriptor behandelt und von ihm bei allen weiteren Operationen verwendet. Der Server kann mittels des Wertes und der L4_Task_ID des Rufenden entscheiden, ob ein Zugriff gültig ist und eine Zuordung zu seinen internen Strukturen machen.

3.2.2 close

Wenn der Server eine Botschaft mit dieser Anweisung erhält, werden die internen Verwaltungsstrukturen freigegeben. Die Verwaltungsnummer kann neu vergeben werden. Dem Client werden alle gemappten Speicherbereiche entzogen, wenn dies nicht schon durch vorherige ioctl-Aufrufe erledigt wurde.

3.2.3 ioctl

Siehe Abschnitt 3.1

3.2.4 mmap

Das Einblenden von Speicherseiten erfolgt mittels des Versendens von Flexpages. Unter L4 kann jeder Client seinen eigenen Adreßraum verwalten. Bereiche, an die Seiten eingeblendet werden sollen, muß er selber organisieren.

3.2.5 Blockierendes Empfangen

Verwendet ein Client eine blockierende Empfangsoperation, ermittelt GM, ob die Eventwarteschlange leer ist. Ist dies der Fall, wird von der GM-Bibiothek eine spezielle systemabhängige Funktion aufgerufen, um durch diese den Client zu blockieren. Zuvor wird eine Struktur in den Speicher der Myrinetkarte kopiert, mit der der Firmware mitgeteilt wird, daß sie nach dem Empfang einer Botschaft für diesen Client eine Hardwareunterbrechung durchführen soll.

Unter L4 bietet es sich an, auf eine Short-IPC vom GM-Server zu warten. Trifft diese ein, wird in die Bibiothek zurückgesprungen und GM übergibt dem Client die empfangene Ereignisstruktur. War die Eventwarteschlange bei einer blockierenden Empfangsoperation nicht leer, wird dem Client sofort eine Ereignisstruktur zurückgeliefert.

Kapitel 4

Leistungsmessung

Für die Leistungsanalysen standen ein Rechner mit einem Intel P90-Prozessor und ein Rechner mit einem AMD K2 mit 400 MHz zur Verfügung. Wegen der ungleichen Leistungsfähigkeit der Rechner wurde in den Tests der Durchsatz des P90-bestückten Systems gemessen. Beide Systeme waren mit je einer Myrinetkarte ausgestattet.

Für die Bestimmung der Zeitdifferenzen wurde die Echtzeituhr der Myrinetkarte des P90-Systems ausgelesen. Diese läuft mit einer Genauigkeit von einer Mikrosekunde (μs).

4.1 Senden

Bei der Sende-Leistungsmessung wurden für einen Testlauf vom P90-System aus 1000 Pakete gleicher Größe versendet. Es wurde mit einer Paketgröße von 4000 Bytes begonnen. Für jeden weiteren Meßdurchlauf wurde die Anzahl der gesendeten Daten verdoppelt. Zusätzlich wurde der Durchsatz bei der kleinstmöglichen Paketgröße gemessen. Zum Vergleich wurde die gleichen Messungen unter Linux durchgeführt. Bei der Auswertung muß berücksichtigt werden, daß unter Linux mehrere Prozesse parallel arbeiteten und viele Geräte gleichzeitig aktiv waren. Die zusätzliche Belastung des Systems durch vorhandene Prozesse war aber verschwindend gering.

Bei dieser Messung wurde von der Annahme ausgegangen, daß der schnellere Rechner die Daten zügiger verarbeitet, als diese vom P90-System versendet werden und er so nicht als Flaschenhals auftritt. Es wurde erwartet, daß der Durchsatz mit größer werdenen Paketen steigt, da der Mehraufwand durch das GM-Portokoll im Verhältnis zur übertragenden Datenmenge sinkt.

Um den Einfluß durch die Verzögerung beim Senden und Empfangen in der Messung zu beseitigen, wurden alle Sendetoken mit Puffern versehen und

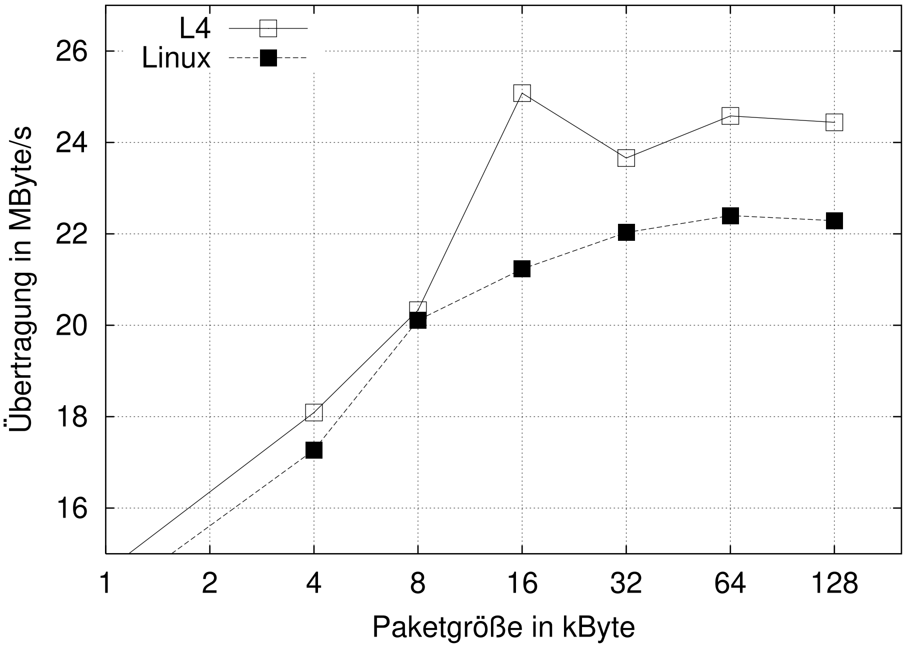

Abbildung 4.1: *Leistungsmessung: Durchsatz beim Senden von Datenpaketen in Abhängigkeit von der Paketgröße*

unverzüglich in den Speicherbereich der Myrinetkarte kopiert.

Auswertung

Wie aus dem Diagramm 4.1 zu entnehmen ist, verhält sich das System unter Linux wie erwartet. Der Durchsatz steigt mit wachsender Paketgröße. Der Durchsatz des L4-Systems war immer größer als unter Linux. Dies kann zum Teil auf die Mehrbelastung des Linuxsystems zurückgeführt werden. Warum sich einer Paketgröße von 16 kByte das Maximum des erreichten Übertragungsleistung befindet, kann zur Zeit nicht erklärt werden.

4.2 Empfangen

Bei dieser Leistungmessung wurden vom schnelleren System Datenpakete gesendet. Die Paketgrößen entsprechen denen im Sendetest. Das P90-System wurde dabei mit Daten überflutet und mußte somit zeigen, wie schnell es die Daten abnehmen kann. Für den Empfang wurden die blockierenden Funktionen verwendet.

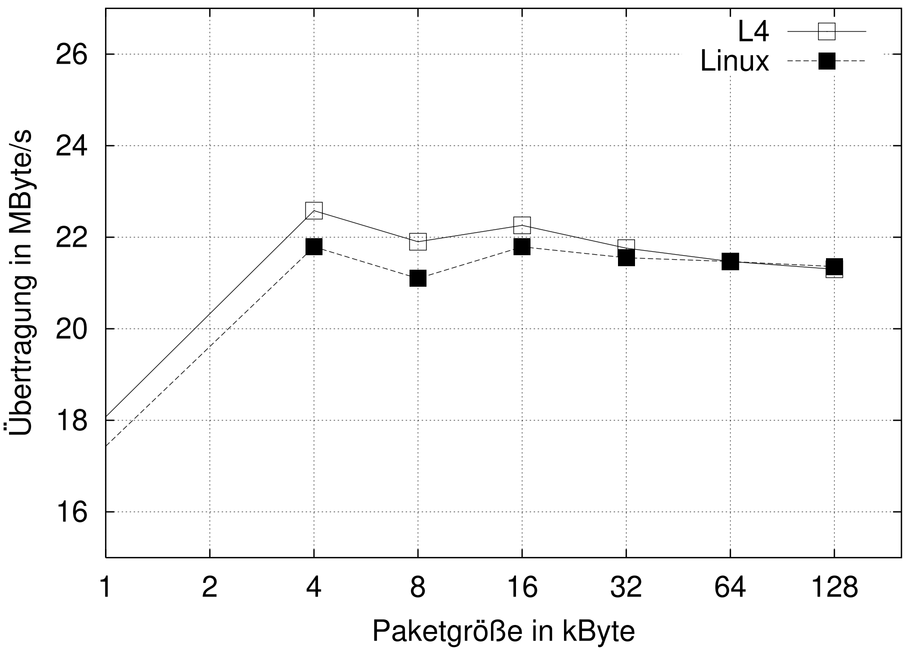

Abbildung 4.2: *Leistungsmessung: Durchsatz beim Empfangen von Datenpaketen in Abhängigkeit von der Paketgröße*

Es wurden vor dem Sendebeginn alle Empfangstoken mit Puffern der erwarteten Datengröße bereitgestellt. Da für einen Port 120 Empfangstoken bereit standen, ergab sich, bei einer Paketgröße von 128 kByte, ein Pufferbedarf von 15,3 MByte. Daher erschien es nicht sinnvoll, die Messung mit einer weiteren Verdopplung der Paketgröße fortzusetzen. Aus diesem Grund wurden die Messungen nur bis Puffergrößen von 128 kByte durchgeführt.

Erwartet wurde ein Ergebnis, in dem die Empfangsleistung an einer bestimmten Paketgröße am höchsten ist. Zu den Seiten würde die Kurve abfallen, da bei kleineren Paketen der Mehraufwand durch das GM-Protokoll hoch ist. Bei großen Paketen hingegen treten vermehrt Hardwareunterbrechungen auf. Diese werden nur ausgelöst, wenn die Ereigniswarteschlage eines Clients leer ist und schlafen gelegt wurde.

Auswertung

Die Annahmen wurden weitestgehend bestätigt (siehe Abb. 4.2). Die leicht höhere Empfangsrate unter L4 wird wieder auf die Mehrbelastung des Systems unter Linux zurückgeführt. Es wird aber ersichtlich, daß Linux sehr viel besser mit Hardwareunterbrechungen umgehen kann, als daß unter L4 mögliche Modell. Beim Auftreten einer Unterbrechung sendet der Mikrokern Omega_0 eine Short-IPC. Omega_0 benachrichtigt den Interruptthread von GM, der wiederum eine Short-IPC an den wartenden Client sendet.

Es wurde die Messung auch mit nichtblockierenden Empfangsoperationen durchgeführt. Das Ergebnis wurde der Übersichtlichkeit wegen nicht mit in das Diagramm aufgenommen. Die Messung bestätigte, daß der Durchsatz nicht so stark zurückgeht, wie bei der blockierenden Variante. Die Übertragungsleistung ist mit der der blockierenden Funktion vergleichbar.

4.3 Verzögerungsmessung

Für die Messung der Verzögerung wurde vom Meßprogramm ein Datenpaket an sich selber gesendet und wieder empfangen. Dabei wurde es zu einem Switch geschickt, der es zum Ursprungsort leitet. Für diesen Test wurde der Mittelwert aus 1000 Paketen berechnet.

Für den Empfang wurde die nichtblockierende Funktion verwendet, da bei dieser Messung kein paralleles Senden möglich ist und somit die Wahrscheinlichkeit, daß bei einem blockierenden Warten eine Hardwareunterbrechung auftritt, sehr hoch ist. Hardwareunterbrechungen sind unter L4 sehr teuer und verzerren somit die Meßergebnisse.

Für diese Messung wurde erwartet, daß die Verzögerungszeit mit der Paketgröße steigt. Vermutet wurde, daß bei einer Verdoppelung der Paketgröße sich die Änderung der Verzögerungszeit weniger als das Doppelte beträgt, da sich der Mehraufwand durch das GM-Protokoll bei gleicher Datenmenge verringert.

Für dieser Meßreihe wurde zusätzlich Messungen bei 128 Byte und 1 kByte großen Paketen durchgeführt.

Auswertung

Die Annahmen wurden nur teilweise bestätigt. Bis zu einer Paketgröße von 4 kByte ist die Vermutung richtig. Über die Grenze wächst die Verzögerung sogar um etwas mehr als das Doppelte.

Daß das Meßprogramm unter L4 unter der 4 kByte Paketgröße eine geringere Verzögerung erreicht, ist wieder auf die Mehrbelastung des Systems unter Linux zurückzuführen. Der Grund, warum das Meßprogramm unter Linux bei Paketgrößen über 4 kByte eine geringere Verzögerung hat, ist im Augenblick unbekannt.

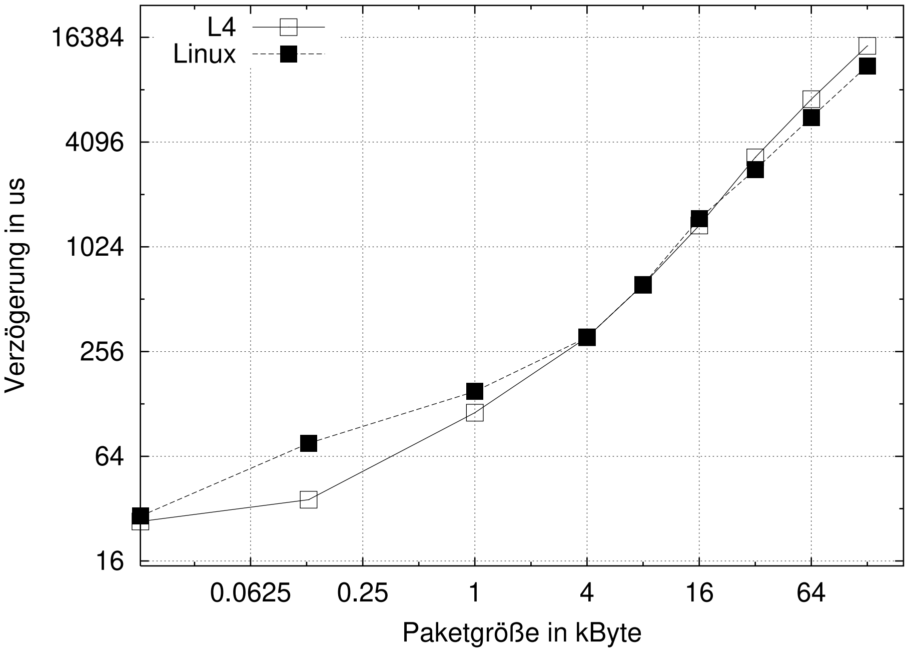

Abbildung 4.3: *Leistungsmessung: Verzögerungszeit für einen Roundtrip in Abhängigkeit von der Paketgröße*

Kapitel 5

Echtzeitbetrachtung

5.1 Myrinet

Durch sein Design ist Myrinet nur bedingt für die Echtzeitkommunikation geeignet. Es stehen, im Gegensatz zu ATM, keine Bandbreitenreservierungsmechanismen zur Verfügung. Wie schon erwähnt, ist Myrinet auf hohe Bandbreiten und geringe Latenzzeiten optimiert. Hier sei noch einmal auf die geringe Funktionalität eines Myrinetswitches und die aufwendige Erstellung der Topologiekarte hingewiesen.

Weiterhin ist es Anwendungen möglich, den Datenaustausch von Teilnehmern zu stören, indem große Pakete an Rechner bzw. Switches gesendet werden, die diese nicht verarbeiten können oder wollen.

Da Myrinet als gegeben angesehen werden muß, lassen sich diese Mengel ohne zusätzliche Hardwareunterstützung nicht beseitigen. Ein schon genannter Vorteil ist aber, eine eigene Firmware programmieren zu können, die auf die Echtzeitkommunikation optimiert ist.

5.2 GM

GM ist ebenfalls nicht für die Echtzeitkommunikation entworfen. Probleme ergeben sich dadurch, daß jeder Client seine Sende- und Empfangstokenwarteschlangen selbst verwaltet. Ein Client kann beliebig lange Datenpakete versenden und kann hierzu große Speicherbereiche Pinnen.

GM bietet aber schon Möglichkeiten, die für die Echtzeitkommunikation ausgenutzt werden kann. So kann die Größe der zu verarbeitenden Pakete beschränkt werden. Hierzu muß lediglich eine Maske in der Firmware registriert werden. GM erlaubt den Nachrichtenaustausch mittels zweier Prioritätsstufen. Die Firmware benutzt ein Kontrollprogramm, welches eine Sendetoken für ein Senden auswählt. Dabei werden die hochprioren Nachrichten bevorzugt.

Zur Verbesserung der Reaktionsfähigkeit sollte die Anzahl der Puffer und deren Größe beschränkt werden, die ein Client benutzen darf. Die Verwaltung der Sende- und Empfangstokenwarteschlangen der Clients sollte durch eine zentrale vertrauenswürdige Instanz vorgenommen werden, die gleichzeitig über die Einhaltung der Zusagen wacht, z.B. der GM-Server bzw. Treiber. Eine andere Möglichkeit ist es, die Überwachung der Zusagen in die Firmware zu verlagern.

5.3 Mögliche Verfahren zu Echtzeitkommunikation über Myrinet

Mögliche Verfahren werden in den nächsten beiden Unterabschnitten beschrieben.

5.3.1 Tokenringähnliches Verfahren

In einem Switch werden Pakete, die sich nicht behindern, gleichzeitig weitergeleitet. Unter Umständen läßt sich so ein tokenringähnliches Netz erstellen. Dazu darf jeder Rechner nur mit einem seiner unmittelbaren Nachbarn kommunizieren und dieser wiederum mit dem nächsten usw. Jedes Paket muß den Empfänger enthalten und wird zum jeweiligen Nachbarn gesendet. Dieser puffert das Paket und ermittelt, ob es für ihn bestimmt ist und übermittelt es gegebenenfalls der zuständigen Anwendung. War er nicht der Empfänger, sendet er das Paket zu einem späteren Zeitpunkt weiter. Ein Scheduler kann nun anhand von gewissen Prioritäten festlegen, wann ein weiterzuleitendes und wann ein eigenes Paket versendet wird.

Wenn jedem Host 1/n-tel (n ist die Anzahl der vorhandenen Rechner im Netzwerk) der Kommunikationsbandbreite zusteht, besitzt jeder Host eine gewisse Anzahl an Zeitschächten (Slots), die er verwenden kann. Es ist aber zu klären, wie eine Taktung der Slots erfolgen kann.

Das Verfahren hat einige Schwachstellen. Es darf nur eine feste Anzahl von Rechnern im Netz geben, um eine feste und damit vorhersagbare Anzahl Slots pro Rechner zu gewährleisten. Fällt ein Rechner aus, ist der Ring unterbrochen. Zur Ermittlung dieser Art Fehler kann die o.g. Taktung verwendet werden. Es müssen zusätzliche Nachrichtenpakete versendet werden, um allen Mitgliedern des Netzwerkes diesen Fehler mitzuteilen. Noch problematischer ist aber das Hinzufügen eines Rechners, da hierbei die ungestörte

Kommunikation gewährleistet bleiben muß. Um den Kommunikationsmehraufwand für das Erforschen des Netzwerkes einzusparen, sollten die Routen statisch vergeben werden.
Der Vorteil des Verfahrens:

- Es wird eine hohe Bandbreite erreicht.
- Die Firmware kann autonom alle entscheidenen Entscheidungen treffen. Es ist kein Einfluß des Hostes nötig.
- Das Verfahren ist deterministisch.
- Das Verfahren arbeitet dezentral.

Nachteile:

- Es ist relativ schwer zu verwirklichen, vor allem die Taktung eines Slots.
- Es gibt eine feste Anzahl von Rechnern im System bzw. eine obere Grenze.
- Es ist aufwendig, Rechner hinzuzufügen bzw. aus dem Verband zu lösen.
- Es können auf der Karten Ressourcenprobleme auftreten, die behandelt werden müssen.

5.3.2 Erweiterung des GM-Mappers

Die im GM-Protokoll erforderlichen Mapper kennen alle Routen und alle Teilnehmer im Netz. Es ist denkbar, die Mapper um die Funktionalität der Echtzeitkommunikation zu erweitern. Möchte ein Teilnehmer in Echtzeit kommunizieren, muß er sich beim Mapper dafür anmelden. Dieser entscheidet anhand der ihm vorliegenden Informationen, ob dies möglich ist. Ist dies der Fall, bekommt die Anwendung die Erlaubnis zum Senden. Die Kommunikationspartner müssen sich vor oder nach der Bestätigung austauschen, ob diese Verbindung zu stande kommen kann.

Die Firmware oder das Protokoll, welches zur Kommunikation verwendet wird, muß die Zusagen der Anwendungen kontrollieren. Hält sich eine Anwendung nicht an diese, wird die Verbindung abgebaut.

Das Verfahren funktioniert mit der Annahme, daß die Aufträge im Mittel bearbeitet werden können. Es kann micht garantiert werden, daß Daten deterministische ankommen oder versendet werden können [Ham97].
Vorteile:

- Dieses Verfahren ist einfach zu verwirklichen, da nur Anpassungen am Mapper und in der Verarbeitung von GM nötig sind.
- Es gibt keine Beschränkung in der Anzahl der Teilnehmer.
- Die Ressourcen und deren Verwaltung entspricht der der Originalversion.
- Der Aufwand zur Erstellung einer Topologiekarte bleibt der gleiche.

Nachteile:

- Der Mapper kann zum Flaschenhals werden. Enthält der Mapper Fehler, werden diese nicht registriert. Fällt ein Mapper aus, kann nach dem Verfahren aus dem Abschnitt 2.3.2 ein neuer Mapper aktive werden.
- Bei diesem Verfahren ist wahrscheinlich keine hohe Bandbreite zu erwarten, da ein Mapper eine Nutzertask ist. Diese muß immer aufgeweckt werden und kann dann erst arbeiten. Ein Lösung in der Firmware ist deutlich schneller und von der Geschwindigkeit des Hostprozessors unabhängig.

Kapitel 6

Schlußbemerkungen und Zusammenfassung

Diese Arbeit gibt einen Einstieg in die Arbeitsweise von Myrinet und das Kommunikationsprotokoll GM. Sie gewährt Einblicke in die Details der L4-Portierung, und es werden ihre Schwächen und Stärken aufgezeigt.

Für die Zukunft sind noch einige Verbesserungsmöglichkeiten offen. Zum ersten müssen die genannten Sicherheitslöcher, die geschlossen werden müssen.

1. Es wird ein Möglichkeit gebraucht, Speicherbereiche mit Adressen über einem GByte, seitenweise, d.h. mit einer Größe von 4kByte, als Flexpage versenden zu können. Hierzu ist notwendig, den L4-Kern zu verändern.

2. Der GM-Server muß um die Fähigkeit zum Einblenden von Speicherseiten an seine Clients erweitert werden (siehe Abschnitt 3.1.1).

3. Um ein ungestörtes Arbeiten von GM zu gewährleisten, sind Pager nötig, die Speicherseiten auf unbegrenzte Zeit pinnen können.

Des weiteren bietet der Originalquellcode GM's viele Möglichkeiten zur Verbesserung der zu erreichenden Bandbreite und zur Verringerung des Mehraufwandes zu dessen Ausführung an, da er allgemein geschrieben wurde und sich nicht an ein bestimmtes Betriebssystem wendet.

Der Treiber ist im Augenblick nicht echtzeitfähig. Er kann aber durch die beschriebenen Verfahren zusagefähig gemacht werden. Weiterhin muß der Treiber noch an das Streaming-Interface von DROPS angepaßt werden, um ihn im DROPS-Projekt nutzen zu können, dies war nicht Teil der Aufgabenstellung.

Anhang A

Glossar

Adreßraum: Menge gültiger Speicheradressen

DMA: „Direct Memory Access“; Zugriff auf den Hauptspeicher ohne Prozessor

Frame: siehe Kachel

I/O: „Input Output“; Ein- und Ausgabeoperationen auf Hardwaregeräte

ioctl: „control device“; Eine Möglichkeit einer Anwendung, Daten mit einem Gerät auszutauschen

ISO: „International Standards Organisation“

Kachel: Der Hauptspeicher ist in gleich große Kacheln unterteilt. Unter der intelbasierten x86-Architektur beträgt die Kachelgröße 4096 Byte.

OSI: „Open Systems Interconnection“

Page: siehe Seite

Page fault: siehe Seitenfehler

Payload: In kommerziellen Netzwerken erfolgt die Abrechnung einer Dienstleistung über die Anzahl der versendeten Daten. Ein Datenpaket besteht i.A. aus einem Paketkopf, Daten und einem Paketanhang. Der Payload besagt, welcher Teil eines Datenpaketes in die Abrechnung eingeht.

PCI: „Peripheral Component Interconnect“; ein verbreitetes I/O-Bussystem

pinnen von Speicherseiten: Das Betriebssystem garantiert, daß Speicherseiten nicht ausgelagert werden und sich die Abbildung von virtuellen auf physischen Adressen nicht ändert.

Seite: Der virtuelle Adreßraum ist in gleich große Seiten geteilt. Die Seitengröße unter der intelbasierten x86-Architektur ist 4096 Byte.

Seitenfehler: Unerlauberter Zugriff auf eine Speicheradresse. Entweder gehört diese Adresse nicht zum Adreßraum einer Task oder es wurde schreibend auf eine Nur-Lesbar markierte Seite zugegriffen.

Superpage: Speicherseite mit einer Größe von 4 MB

timeout: Es wird eine bestimmte Zeit auf ein Ereignis gewartet. Tritt dieses Ereignis nicht in der erwarteten Zeit ein, tritt der timeout auf und die Anwendung kann diesen behandeln.

Literaturverzeichnis

[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."

[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"

[Gru99] L. Grützmacher: *Dokumentation VM **Oktober 99***;

Verfügbar unter URL:

„http://www.os.inf.tu-dresden.de/~lg2/doc/vm.ps"

[Ham97] Cl.-J. Hamann: *Jitter Constrained Periodic Streams **January 1997***

Verfügbar unter URL:

„http://www.os.inf.tu-dresden.de/papers_ps/stroeme.ps"

[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“

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

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