Reliable Multicast

Seminararbeit

vorgelegt am

Lehrstuhl für Praktische Informatik IV
Prof. Dr. W. Effelsberg
Universität Mannheim

im
Dezember 1999

von
Mirko Friedrich
Mannheim

Inhaltsverzeichnis

1 Einleitung

Der Daten- und Informationsaustausch zwischen verschiedenen Parteien ist in der heutigen Zeit zu einer zentralen Frage geworden. Das Internet bietet hier eine Möglichkeit, Informationen und Daten auf schnellstmöglichem Weg zu erlangen. Bisher waren Daten- und Informationsaustausch auf Punkt zu Punkt Verbindungen beschränkt (also zwischen zwei unabhängigen Rechnern). Um diese Verbindung, auch Unicast genannt, herzustellen wurden Protokolle entwickelt, die diesen Datenaustausch ermöglichen. Wichtige Aufgabe dieser Protokolle ist mitunter, den „richtigen Weg“ durch das Internet für die zu übermittelnden Daten zu finden. Dies wird auch als Routing oder Leitwegebestimmung bezeichnet [Gru96]. Die *International Organization of Standardization* (ISO) versuchte mit Hilfe der gebildeten Gruppe *Open System Interconnection* (OSI) eine Standardisierung der Protokolle zu erreichen [Gru96]. Die heute bekanntesten Unicast - Protokolle, die dieses Routing übernehmen, sind das *Transmission Control Protocol* (TCP) und das *User Data Protocol* (UDP). Ersteres steht für einen verbindungsorientierten und UDP für einen verbindungslosen Dienst.

Neuere Anwendungen beteiligen nun auch mehrere Endsysteme am Datenaustausch. Hierzu zählen Anwendungen wie Videoconferencing, bei denen auch mehrere Sender vorhanden sein können. Durch Verwendung der bestehenden Unicast – Protokolle kann eine unnötige Belastung des Netzes erfolgen (z. B. durch mehrfaches Versenden von gleichen Daten auf dem selben Weg). Deswegen wurden neue Protokolle, sog. Multicast – Protokolle, entwickelt, die diesen neuen Anforderungen gerecht werden. Manche Anwendungen benötigen einen „schnellen“ Dienst (z.B. Audio- und Videoübetragungen auf dem *Multicast Backbone* (Mbone)) ohne Rücksicht auf Vollständigkeit der gesendeten Daten (auch Pakete genannt). Andere benötigen hingegen einen zuverlässigen Dienst wie die Benutzung der *Digital Lecture Board* (DLB) - Anwendungen.

Die Frage nach zuverlässigen Multicast-Protokollen kam erst in den letzten Jahren auf. Hierbei ist zu beachten, daß sich die Definition von Zuverlässigkeit in Multicast mitunter stark von der Zuverlässigkeit in Unicast unterscheidet. Z. B. kann eine Übertragung im Multicast als zuverlässig definiert werden, wenn ein Paket mindestens k Empfänger (k sei vor der Übertragung festgelegt worden) erreicht hat. Diese und andere Fragen sollen in dieser Arbeit erläutert und ausgeführt werden. Dafür wird zuerst die Definition von Zuverlässigkeit erörtert und danach eine Klassifikation und Vorstellung von Multicast–Protokollen erfolgen.

2 „Reliable Multicast“-Kommunikation

2.1 Grundlagen der Multicast–Übertragungen

Die Klassifizierung von Multicast–Protokollen erfordert zunächst die Klärung von Grundbegriffen der Multicast–Übertragungen. In der Einleitung wurde bereits erwähnt, daß es verschiedene Arten der Kommunikation gibt. Neben Unicast und Multicast existiert auch noch die Broadcast- Übertragung. Bei einer Multicast–Übertragung wird noch zwischen einer „reinen“ Multicast–Übertragung und einer Multipeer–Übertragung unterschieden. Letztere zeichnet sich dadurch aus, daß mehrere Sender in einer Gruppe vorhanden sind. In einer „reinen“ Multicast–Übertragung ist nur ein Sender vorhanden. Bei Broadcast–Übertragungen sendet ein Sender an alle möglichen Empfänger in einem Netz. Diese Art der Übertragung ist hier nicht von Interesse.

2.1.1 Gruppenspezifikation

Weitere Eigenschaften der Gruppenkommunikation sind die Lebensdauer, Dynamik und Bekanntheit einer Gruppe.

Die Lebensdauer einer Gruppenkommunikation kann permanent oder temporär sein. Eine permanente Gruppe besteht immer und muß nicht eingerichtet werden [Ata98]. Dieser Gruppe wird eine permanente IP–Adresse zugeordnet. Im Gegensatz hierzu stehen die temporären oder transienten Gruppen mit einer temporären IP–Adresse, die nach Ablauf einer Übertragung wieder freigegeben wird. Die Verwaltung solcher IP–Adressen ist mit hohem Aufwand verbunden und wird noch immer manuell von autorisierten Personen vorgenommen.

Bei einer dynamischen Gruppe können während der Übertragung Mitglieder beitreten oder austreten. Dies sollte keinen Einfluß auf den Sender haben. Neu hinzukommende Empfänger müssen damit rechnen, daß sie nicht mehr alle bisher gesendeten Pakete anfordern können. Bei einer statischen Gruppe ändert sich die Zusammensetzung der Gruppe über die Übertragungszeit nicht [Gru96]. Jedoch kann der Austritt eines Mitgliedes in dynamischen und statischen Gruppen zu Problemen führen, da hier der Empfänger evtl. auch Pakete duplizieren oder einfach nur weiterleiten muß an weitere Empfänger.

In Multicast–Protokollen ist ein wichtiger Aspekt die Bekanntheit der Gruppe. Man unterscheidet zwischen anonymen und bekannten Gruppen. Bei bekannten Gruppen kennt der Sender alle Gruppenmitglieder zu jedem Sendezeitpunkt. Die Gruppe kann sich durchaus während einer Übertragung verändern, aber eine statische Übertragung impliziert immer eine bekannte Gruppe.

2.1.2 Gruppentopologien

In Multicast–Protokollen können Gruppen zu verschiedenen Topologien angeordnet werden. Z. B. kann eine Gruppe nach einem Baum (z. B. einem „Spanning Tree“) oder einem Ring aufgebaut werden. In diesem Fall spricht man von einer *strukturierten Gruppentopologie*. Diese ermöglicht z. B. die Organisation einer dynamischen Gruppe. Dagegen nennt man eine Gruppe *flach*, wenn keine Topologie vorhanden ist. Die Gruppenmitglieder haben hier keine Beziehungen zueinander.

2.2 Definition von Zuverlässigkeit

Wie bereits erwähnt, unterscheidet sich der Begriff Zuverlässigkeit von Unicast– zu Multicast– Übertragungen mitunter stark. So schreiben Mankin, Romanow, Bradner und Paxson im RFC 2357:

> „For unicast transport, the requirements for reliable, sequenced data delivery are fairly general. ...
>
> In contrast, different multicast applications have widely different requirements for reliability.“

Anhand dieser Aussage ist zu erkennen, daß die Definition von „Reliability“ in Multicast schwierig ist. Hier soll nun versucht werden, einige Kernpunkte von Zuverlässigkeit darzustellen, die von fast allen zuverlässigen Multicast–Protokollen beachtet werden.

2.2.1 Ordnung

Der Erhalt der Ordnung fordert, daß die in der richtigen Reihenfolge gesendeten Pakete auch in der richtigen Reihenfolge bei den Empfängern ankommen. Geschieht dies nicht, so haben je nach Protokoll Sender oder Empfänger dafür zu sorgen, daß diese Reihenfolge wiederhergestellt wird.

In den heutigen Multicast–Protokollen gibt es folgende Stufen der Ordnung: Ungeordnet, Sender–geordnet und total-geordnet. Bei einem Protokoll mit ungeordnetem Dienst wird keine Ordnung garantiert. Bei Sender-geordneten Diensten wird eine geordnete Reihenfolge aller zu sendenden Pakete garantiert. Eine Ordnung der Pakete zwischen verschiedenen Sendern wird nicht zugesagt. Dies wird bei einem total–geordneten Dienst angeboten. Die Sendereihenfolge der Dateneinheiten aller Quellen bei allen Empfängern stimmt hier zusätzlich überein.

Die Herstellung dieser Ordnung kann in verschiedenen Schichten erfolgen. Hierzu wird ein sog. Synchronisationsprotokoll benötigt. Dieses kann in die Transportschicht integriert werden, als eigene Schicht auf der zuverlässigen Transportschicht aufsetzen, oder die Anwendungsschicht übernimmt selbst die Herstellung der geforderten Ordnung. Man sollte nur beachten, daß nicht jede Anwendung die volle (totale) Ordnung benötigt. Deswegen ist es fraglich, ob jedes Protokoll die vollen Ordnungsdienste anbieten sollte.

2.2.2 Fehlererkennung und -signalisierung

Grundsätzlich unterscheidet man bei zuverlässigen Multicast–Protokollen zwischen senderinitiierten und empfängerinitiierten Protokollen.

Bei senderinitiierten Protokollen liegt die Verantwortung der Fehlererkennung beim Sender. Der Sender hat dabei dafür zu sorgen, daß der Empfängern jedes Paket korrekt erhalten hat. Hier erwartet der Sender von den Empfängern eine Bestätigung über den Erhalt und die Korrektheit der versendeten Pakete mittels sog. *Acknowledgements* (Ack).

Bei empfängerinitiierten Protokollen hat der Empfänger dem Sender mitzuteilen, daß er ein Paket oder mehrere Pakete nicht oder unkorrekt erhalten hat. Dies geschieht durch versenden sog. *Negative Acknowledgements* (NAck)

Es sollte noch erwähnt werden, daß sowohl bei senderinitiierten als auch bei empfängerinitiierten Protokollen eine mitunter erhebliche Belastung des Netzes entstehen kann. Falls viele Acks oder NAcks versendet werden müssen und diese das Netz erheblich überfluten, spricht man von einer *Ack*- oder *NAck–Implosion*.

Auf Einzelheiten bez. der Vor- und Nachteile der senderinitiierten und empfängerinitiierten Multicast - Protokolle sowie der Vermeidung von Ack- und NAck– Implosions wird in Kapitel 3 näher eingegangen.

Auch muß unterschieden werden, ab wann eine Übertragung eines Pakets als zuverlässig eingestuft werden kann. Würde man in Anlehnung an Unicast–Protokolle vorgehen, so würde eine Übertragung abgebrochen werden, wenn ein Empfänger ein Paket nicht erhalten hat. Deswegen wurde der Begriff des halbzuverlässigen Dienstes in dynamischen Gruppen eingeführt. Dieser besagt, daß die korrekte Auslieferung an alle Empfänger nicht garantiert wird. Diese Dienste unterstützen dennoch Fehlerkennung und –behebung [Gru96]. Der halbzuverlässige Dienst läßt sich noch in den k–zuverlässigen Dienst und den statisch zuverlässigen Dienst unterteilen.

Der k–zuverlässigen Dienstes besagt, daß eine Übertragung erfolgreich war, wenn mindestens k Empfänger ein Paket korrekt erhalten haben. Wie k zu wählen ist, hängt von verschiedenen Parametern wie Art der Anwendung, Netzbelastung, etc. ab.
Bei einem statisch zuverlässigen Dienst werden die gesendeten Daten über einen bestimmten Zeitraum gepuffert. Kommen in dieser Zeit Anforderungen für eine Sendewiederholung, so kann der Sender diese erfüllen. Kommen nach Leerung des Puffers Anfragen, so können diese Forderungen nicht erfüllt werden. Die Größe des Zeitintervalls der Datenpufferung und die Größe des Speichers bestimmen die Zuverlässigkeit der Übertragung.

2.2.3 Retransmission

Nach Erkennen und Signalisieren eines Fehlers in einer Übertragung muß eine erneute Übertragung der fehlerhaften Pakete erfolgen. Die Fragen, die sich hier stellen, sind, wie und von wem diese Pakete erneut übertragen werden sollen. Einige Ansätze übertragen die fehlenden Pakete per Unicast an den entsprechenden Empfänger, andere per Multicast an die gesamte Gruppe. Empfänger, die diese per Multicast gesendeten Pakete nicht brauchen, verwerfen diese.
Die Retransmission kann vom Sender erfolgen oder von benachbarten Empfängern, um die Überlastung des Netzes und des Senders zu vermeiden. Im RMTP gibt es sog. *Designated Receiver*, welche die Quittungsbearbeitung und Retransmissions teilweise übernehmen. Erst wenn ein Designated Receiver selbst die benötigten Datenpakete nicht erhalten hat, wendet dieser sich an den Sender. Auf diese Art der Übertragungswiederholung wird in Kapitel 3 und Kapitel 4 (RMTP) näher eingegangen.

2.3 Congestion Control und Quality of Service (QoS)

In diesem Abschnitt soll nur knapp auf die Flußkontrolle und QoS in Multicast–Protokollen eingegangen werden, da diese ein eigenes Forschungsgebiet bilden und den Rahmen dieser Schrift überschreiten würden.
„Congestion Control“ verhindert, daß ein schneller Sender langsame Empfänger überschwemmt. Bei senderinitiierten Protokollen können, ähnlich wie bei Unicast-Protokollen, das *Sliding-Window* Verfahren eingesetzt werden. Der Sender muß dabei auf alle Quittungen (Acks) der Empfänger warten, bevor er weitersendet. Der langsamste Empfänger bestimmt also die Übertragungsgeschwindigkeit, was nicht immer gewünscht ist.

Bei empfängerorientierten Protokollen (also mit NAcks) können verschiedene Methoden angewandt werden. Hier sind die *Ratenkontrolle* (z. B. in *Scalable Reliable Multicast* (SRM)), die *Layered Groups* und die *dynamische Ratenkontrolle* (z. B. in *Reliable Adaptive Multicast Protocol* (RAMP)) zu nennen, die in verschiedenster Form implementiert worden sind.
Es soll noch kurz erwähnt werden, daß Flußkontrolle nötig ist, da Multicast i.a. größeren „Schaden“ als Unicast im Netz anrichten kann. Hier sind die zusätzlichen Belastungen des Netzes durch Kontrolleinheiten (Acks, NAcks, Statusmeldungen, etc.) zu nennen. Auch haben zuverlässige Multicast Anwendungen keine „harten“ Zeitgrenzen (wie Audio- und Video). So kann eine Multicast File Transfer Anwendung die Daten solange senden, bis alle Empfänger die Daten erhalten haben [RFC98]. Dies kann endlos sein, wenn sich ein Mitglied unkontrolliert abmeldet.
Auch für den Bereich QoS trifft die Aussage zu, daß dies ein eigener Forschungsbereich ist und deshalb ebenfalls nur knapp erwähnt werden kann. Nach der ISO sind in verbindungsorientierten Protokollen die Dienstgütemerkmale durch die Parameter Durchsatz, Übertragungsverzögerung, Restfehlerrate und Übertragungswahrscheinlichkeit bestimmt. Durch die „Reliability“ kann keine max. Übertragungsverzögerung garantiert werden. Der einzige Parameter, der zuverlässige Transportdienste charakterisiert, ist der Durchsatz. Diesen gilt es bei einer Übertragung auszuhandeln [Gru96].

3 Klassifizierung von „Reliable Multicast“-Protokollen

Neben allgemeinen Kriterien wie *Fragmentation/Reassembly*, *Flow Control*, *Skalierbarkeit* etc. [Rmp99], gibt es bei zuverlässigen Multicast–Protokollen die Hauptunterscheidungsmerkmale senderinitiiert, empfängerinitiiert und Baum–basiert.

3.1 Senderinitiierte Protokolle

Liegt die Fehlererkennung beim Sender, so spricht man von senderinitiierten Protokollen. Die Empfänger senden nach Erhalt der Pakete dem Sender eine Bestätigung (Acknowledgement). Dieses Verfahren ist angelehnt an das traditionelle Verfahren im Unicast. Erhält der Sender in einem gewissen Zeitraum nicht alle Acks von den Empfängern, so hat er eine Übertragungswiederholung für die betreffenden Empfänger vorzunehmen. Nachteil dieses Verfahrens ist, daß bei anwachsender Zahl der Empfänger die Bearbeitung der Acks zu einem erheblichem Aufwand wird. Man spricht hier von einer *Ack-Implosion*. Die Skalierbarkeit dieser Protokolle ist sehr schlecht. Daher können senderinitiierte Protokolle nur in kleinen Gruppen eingesetzt werden und sind in der Praxis kaum noch von Bedeutung.

3.2 Empfängerinitiierte Protokolle

Die aktuellen Multicast-Protokolle sind fast alle empfängerinitiierte Protokolle. Die Fehlererkennung liegt hier bei den Empfängern. Diese haben durch versenden von NAcks für die Übertragungswiederholung zu sorgen. Empfänger erkennen den Verlust eines Pakets anhand der Sequenznummern der Pakete. Vorteil dieses Verfahrens ist, daß der Sender nicht mit der Fehlererkennung belastet wird. Außerdem ist die Belastung des Netzes geringer, da i. a. weniger NAcks im Vergleich zu Acks gesendet werden. Falls aber ein Datenpaket viele Empfänger nicht erreicht hat, kann es auch hier zu einer *NAck-Implosion* kommen. Auch kann es zu unnötigen Übertragungswiederholungen kommen.
Hierzu wurden von Ramakrishnan et al. Verfahren entwickelt, die als *NAck-Avoidance* bezeichnet werden. Auf die Vorgehensweise wird in Kapitel 4 näher eingegangen.
Es soll noch erwähnt werden, daß NAcks auch zwei grundlegende Probleme mit sich bringen. Zum einen weiß ein Sender nie genau, ob auch alle Empfänger ein Paket erhalten haben, da die NAcks auch verloren gehen könnten. Der Sendepuffer kann demnach erst sehr spät geleert werden, um diese spät eintreffenden Übertragungswiederholungen befriedigen zu können.

Zum anderen ist der Verlust eines einzigen gesendeten Pakets oder des letzten Pakets nicht feststellbar. Der Empfänger hat keine Möglichkeit den Verlust zu realisieren. Zur Lösung dieses Problems werden periodisch sog. Statuspakete versendet (*heartbeats*), welche die zuletzt gesendeten bzw. empfangenen Sequenznummern enthalten. Jedoch tritt dadurch eine weitere Belastung des Netzes auf. Die Sendeintervalle der heartbeats wird mit der Größe der Gruppe vergrößert, um die Skalierbarkeit der Protokolle zu gewährleisten [Gey99]. Eine weitere Möglichkeit bietet die *Sättigung.* Hierbei muß eine Nachricht eine Mindestanzahl von Paketen haben, die über eine Mindestzeitspanne gesendet werden. Diese Methode löst aber auch nicht Probleme wie Partitionierung des Netzes. Für weitere Informationen wird auf die Literatur verwiesen.

Wie oben geschrieben, kann sich ein Sender nie sicher sein kann, daß alle Empfänger ein gesendetes Paket erhalten haben, da NAcks auch mehrfach verloren gehen können. Durch die lange Pufferung der Daten muß der Sender einen großen Speicherplatz zu Verfügung haben. Eine Lösung wäre es, die Aufbehaltungsfristen zu verkürzen. Dann müßte gefolgert werden, daß solche Protokolle nur einen statistisch zuverlässigen Dienst anbieten (wie im *Multicast Transport Protocol 2* (MTP2)), da nicht mehr alle Übertragungswiederholungen erfüllt werden können. Nach Garcia – Luna - Aceves ist ein zuverlässiger empfängerinitiierter Transportdienst nur in der Anwendungsschicht zu realisieren [Gru96].

3.3 Baum-basierte Protokolle

Die Mitglieder bei dieser Art von Protokollen werden in einem Baum angeordnet (was bei größerer Zahl aufwendig werden kann). Die Fehlerkontrolle wird mittels positiver Acknowledgements vorgenommen. Um die Überlastung des Senders zu vermeiden, wird die Fehlerüberwachung verteilt. Empfänger versenden die Bestätigungen nur an ihren direkten Vater im Baum. Bei Bedarf versendet dieser die Daten erneut an seine Kinder. Ein Vater kann nur eine max. Anzahl an Kindern haben. Somit ist die Skalierbarkeit solcher Protokolle sehr gut. Eines der ersten Protokolle dieser Art war das *Reliable Multicast Transport Protocol* (RMTP) von Paul et al. , welches in Kapitel 4 noch vorgestellt wird.

Probleme ergeben sich bei einer Multipeer Kommunikation. In den bisher beschrieben Baum–basierten Protokollen gibt es nur einen Sender (i.e. die Wurzel). Eine Lösung wäre die Einrichtung mehrere Bäume für mehrere Sender. Allerdings ist die Verwaltung dieser Bäume u. U. sehr aufwendig, da z. B. ein Empfänger mehrere Väter haben kann und dieser die Acks ihrem richtigen Vater zusenden müssen. Das Lorax Protokoll schlägt als Lösung einen ge-

meinsamen Baum vor, den sog. *Shared Ack Tree*. Hierbei ist es notwendig, daß der Empfänger den sog. Leitweg zu seinem Sender kennt [Gru96]. Dies kann ein Vater oder ein Kind sein. Für die Bestimmung dieses Leitweges wird auf die Literatur verwiesen.

Die Baumverwaltung ist einer der wichtigsten Aspekte dieser Protokolle. Man muß dabei unterscheiden zwischen dem Beitritt eines neuen Mitglieds, dem Austritt eines Mitglieds und dem Überschreiten der max. Anzahl der Kinder.

Der Beitritt eines neuen Mitglieds ist einfach. Das neue Mitglied kann sich passiv oder aktiv verhalten. Bei passivem Verhalten wartet das neue Mitglied, bis es eine Einladung erhält. Bei aktivem Verhalten sucht das neue Mitglied einen Vater. Aber auch eine Kombination ist möglich.

Der Austritt eines Mitglieds mit Kindern ist mit Schwierigkeiten verbunden, da dieser Vater die Verantwortung dafür trägt die Kinder mit Daten zu versorgen. Die Kinder müssen sich einen neuen Vater suchen. Dieser neue Vater war bis zu dem neuen Zeitpunkt nicht für die fehlenden Daten zuständig. Seine Kinder hatten ihm im ungünstigen Fall schon diese Daten quittiert. Eine Lösung wäre es, den Austritt (einen sog. kontrollierten Austritt) dem Sender mitzuteilen, damit dieser den Sendevorgang unterbricht. Der Nachteil ist, daß die Senderate damit sinkt.

Bei einem unkontrolliertem Austritt trägt der Vater keine Verantwortung für seine Kinder. Es werden keine Maßnahmen zur Umstrukturierung des Baumes getroffen [Gru96]. Hier kann nicht garantiert werden, daß die Kinder die Daten vollständig erhalten werden. Lösungsmöglichkeiten wären, daß diese Kinder die Gruppenkommunikation verlassen müssen oder diese Übertragung als gescheitert erklärt wird. Im SRM und RMTP werden die Daten in der Anwendungsebene gespeichert, welche dann für die Übertragungswiederholung zuständig ist. Von Levine et al. [Lev96a] wird vorgeschlagen, sog. *aggregierte Bestätigungen* zu benutzen. Diese Bestätigungen quittieren den Erhalt der eigenen Daten und den Erhalt der Daten der Kinder. Der Sender löscht erst die Daten, wenn alle aggregierten Bestätigungen eingetroffen sind.

Nach Austritt eines Mitglieds muß der (Teil-)Baum neu strukturiert werden. In Protokollen, die einen Shared Ack Tree benutzen, muß der neue Teilbaum bekannt gemacht werden. Hierzu werden per Multicast Nachrichten versendet, die den alten und den neuen Teilbaum enthalten, damit die Bestätigungen wieder korrekt versendet werden können. In dieser Zeit können keine Nutzdaten versendet werden. Hierbei ist zu beachten, daß mit dem Anwachsen der Mitglieder die Zeit für die Umstrukturierung des Baumes steigen wird. Dies beeinträchtigt erheblich die Skalierbarkeit dieser Protokollart.

4 Ausgewählte Protokolle

Die Anzahl zuverlässiger Multicast–Protokolle ist so groß, daß nur einige Protokolle stellvertretend und nicht im Detail vorgestellt werden sollen, da dies den Rahmen dieser Schrift überschreiten würde.

4.1 Local-Based Receiver-Reliable Multicast (LBRM)

Das Protokoll LBRM baut auf dem unzuverlässigen Multicast–Netzdienst (IP und UDP) auf. LBRM soll für große Gruppen anwendbar sein, ist empfängerorientiert und sieht kein Gruppenmanagement vor (der Sender kennt die Gruppe nicht).
Die Datenpakete sind mit Sequenznummern und Prüfsumme versehen, um verlorengegangene und verfälschte Pakete zu erkennen. Übertragungswiederholungen werden per NAck angefordert. Hier wird das Heartbeat–Verfahren (siehe Abschnitt 3.2) angewendet, um nicht erhaltene Pakete zu erkennen. Das Heartbeat–Intervall wird der Netzsituation angepaßt, damit die Kontrolldateneinheiten eine nicht zu große Netzlasten erzeugen. Die Größe des Heartbeat–Intervalls bestimmt, wie schnell ein Paketverlust erkannt werden kann. Falls nun ein Datenpaket versendet wird, wird der Zeitgeber des Heartbeat–Intervalls zurückgesetzt. Wird keine Dateneinheit gesendet, so wird nach Ablauf des Zeitgebers eine Kontrolleinheit gesendet. Werden längere Zeit keine Dateneinheiten gesendet, so wird nach Ablauf jedes Zeitgebers das Heartbeat–Intervall verdoppelt.
Im LBRM–Protokoll werden Übertragungswiederholungen nicht zwingend gefordert. Die Anwendung entscheidet, ob eine Wiederholung nötig ist.
Negative Quittungen werden nicht an die Quelle versendet, sondern an einen Logging–Server. Die Empfänger bauen dann eine Punkt–zu–Punkt Verbindung (wie im Unicast) zu diesem Server auf und fordern die fehlenden Pakete an. Die Daten werden dort solange gespeichert, bis alle Empfänger diese bestätigt haben. LBRM bietet auch keine Ordnungserhaltung an, da eine korrekte Übertragung vom Protokoll nicht gefordert wird. Dies könnte eher Aufgabe der Anwendung sein. Um eine Quittungsimplosion zu vermeiden, wurden sog. sekundäre Logging–Server vorgeschlagen. Empfänger wenden sich zuerst an ihren sekundären Logging Server. Falls dieser sekundäre Server die Daten nicht hat, so wendet er sich dieser an den primären Server. Durch geschickte Wahl der sekundären Server kann die Netzlast begrenzt und können evtl. Übertragungswiederholungen beschleunigt werden. Diese sekundären Server

sollten dann aber auch außerhalb der belasteten Netzteile liegen. Die Lokalisierung der Server ist allerdings ungeklärt [Wit99].
Ein weiterer Vorschlag, die Netzlast zu verringern ist es die Übertragungswiederholungen per Multicast zu senden. Dies lohnt sich aber nur bei einer gewissen Menge von Empfängern, die das gleiche Paket anfordern. Um das festzustellen, wählt der Sender zufällig ein paar Empfänger aus und fordert von ihnen während der Übertragung positive Acks an. Diese zufällig ausgewählten Empfänger sollen charakteristisch für die Gruppe sein. Falls einige der Empfänger die Pakete nicht erhalten haben, sendet der Server die Pakete erneut per Multicast. Wie hoch die Zahl der fehlenden Acks sein muß, ist nicht festgelegt.

4.2 Reliable Multicast Transport Protocol (RMTP)

RMP ist ein zuverlässiges Multicast-Protokoll für Verteildienste wie Software. Die Datenpakete haben daher feste Länge, nur das letzte Paket kann kürzer sein. Bei der Entwicklung des Protokolls wurde auf seine Skalierbarkeit geachtet. Es verwendet lokale Übertragungswiederholungen, um eine Quittungs–Implosion zu vermeiden. RMTP ist ein Protokoll, das in die Kategorie der hierarchischen oder Baum–basierten Protokolle einzuordnen ist. Um die lokalen Wiederholungen durchzuführen, werden sog. *Designated-Receiver* bestimmt, welche die Quittungsverarbeitung für einen Teilbaum übernehmen. RMTP beinhaltet Funktionen zur Fluß-, Raten- und Staukontrolle. Außerdem werden während einer Verbindung Zustandsinformationen ausgetauscht, die aber nicht mit der Gruppenzahl steigen. Dies spiegelt sich in der Skalierbarkeit wider.
Für den Verbindungsaufbau stellt ein *Session Manager* vorab Verbindungsparameter wie Größe des Empfangsfensters, Größe des Sendefensters, Sendeintervall, Länge der Nutzdateneinheit, Schwellwert für Staukontrolle etc. bereit. Sender und Empfänger erhalten die notwendigen Parameter für den Verbindungsaufbau. Bei RMTP erfolgt der Verbindungsaufbau implizit, d. h. der Sender kennt seine Empfänger nicht und kann daher keinen vollständigen zuverlässigen Dienst anbieten.
Ein Empfänger kann auch einer Verbindung später beitreten. Er fordert die bisher gesendeten Pakete per NAcks an.
Der Sender schickt als letztes ein Paket vom Typ DATA–EOF. Die Empfänger wissen nun, daß die Übertragung beendet ist, wenn sie bis dahin alle Pakete korrekt empfangen haben. Der Sender beendet die Session, wenn ein Zeitgeber abläuft, den er nach Abschicken des letzten Pakets gestartet hat. Trifft aber währenddessen eine Quittung ein, so setzt der Sender diesen

Zeitgeber neu. Auch die Designated–Receiver starten diesen Zeitgeber, da sie an der Übertragung ebenfalls beteiligt sind. Einen expliziten Verbindungsabbruch kann man mittels Versenden einer RESET-Dateneinheit erreichen.

Zur Fehlerbehandlung versenden die Empfänger periodisch Quittungen mit einer Sequenznummer und einem Bitvektor. Die Sequenznummer gibt an, bis zu welcher Nummer exklusive Pakete korrekt empfangen wurden. Der Bitvektor beschreibt die korrekt und nicht korrekt empfangenen Pakete (eine 0 bedeutet ein fehlendes Paket). Diese Quittungen werden an den o. g. Designated–Receiver gesendet. Dieser überträgt die fehlenden Pakete entweder per Unicast oder Multicast. Das ist abhängig von dem Verbindungsparameter $MCAST_{tresh}$. Übersteigt die Anzahl der anfordernden Empfänger diesen Wert, so sendet der Designated–Receiver per Multicast. Eine Übertragungswiederholung kann auf Wunsch auch direkt erfolgen. Hierbei wird nicht gewartet, ob noch weitere Quittungen von anderen Empfängern bei dem Designated–Receiver eintreffen.

Falls ein Designated–Receiver die Daten selbst nicht erhalten hat, fordert er ebenfalls eine Übertragungswiederholung an. Dies kann an einen im Baum höhergelegen Designated–Receiver oder an den Sender erfolgen. Die Designated–Receiver können hierarchisch im Baum angeordnet sein. Ein Designated–Receiver versorgt nur einen Teilbaum mit den angeforderten Daten, um die Netzlast gering zu halten. Ein Designated–Receiver versendet die angeforderten Daten immer an den für ihn zuständigen Bereich im Baum. Je näher ein Designated–Receiver am Sender ist, desto größer ist der Teilbaum, den er zu versorgen hat.

Die Technik bei Übertragungswiederholungen, nur zuständige Teilbäume zu versorgen, wird *Subcasting* genannt. Diese Technik ist im IP – Multicast noch nicht implementiert. Um dies trotzdem zu ermöglichen, wird eine Tunneltechnik verwendet. Die Daten werden in ein Paket des Typs SUBTREE–MCAST gepackt und an den zuständigen Router gesendet, der erkennt dieses Format, entpackt die Daten und sendet sie per Multicast an die Teilgruppe. RMTP kann also nicht ohne Modifikationen an Routern eingesetzt werden.

Die Abstände (Perioden) nach denen eine Quittung versendet wird, wird im Protokoll durch einen Parameter gesetzt. Diese Perioden sind abhängig von der Umlaufzeit. Die Umlaufzeit kann in RMTP mittels verschiedener Mechanismen bestimmt werden, um so die Periodenlänge zu bestimmen.

Die Designated–Receiver sind einfache Empfänger. Die Zuordnung der weiteren Empfänger zu einem Receiver erfolgt dynamisch. Der Sender verschickt dazu ein spezielles Paket an alle Mitglieder mit einem gleichen *Time–To–Live* (TTL) Wert. Die Empfänger können nun anhand des Rest-TTL Wertes erkennen, welcher Designated–Receiver der nächste ist. Dieser

Vorgang wird während der Übertragung wiederholt, damit kann sich die Struktur zur Laufzeit ändern. Dadurch wird eine schnelle Übertragungswiederholung erreicht, da die Umlaufzeiten zum nächsten Designated–Receiver so klein wie möglich gehalten werden. Allerdings werden keine anderen Parameter, wie z. B. verfügbare Bandbreite berücksichtigt. Auch muß die laufende Übertragung komplett gespeichert werden, um Übertragungswiederholungen zu ermöglichen. Dies bedeutet eine mitunter extreme Beanspruchung an Pufferplatz bei Sender und Designated–Receivern.

RMTP bietet auch Mechanismen zu Flußkontrolle. Die Parameter T_{send}, *Packet_Size* und W_s (Fenstergröße) steuern die Senderate. Hierbei wird ein fensterbasiertes Verfahren verwendet. Der langsamste Empfänger bestimmt dabei die Senderate.

Zur Staukontrolle bietet RMTP auch eine Möglichkeit an. RMTP zählt während einer Übertragung die Anzahl der NAcks. Überschreitet diese einen bestimmten Wert ($CONG_{thresh}$), so ist dies ein Indiz für „*Congestion*“ und die Senderate wird gedrosselt. Diese kann dann langsam wieder gesteigert werden [Wit99].

4.3 Scalable Reliable Multicast (SRM)

SRM gehört nicht zu den klassischen Multicast–Protokollen. Dieses Protokoll orientiert sich am *Application Level Framing* (ALF) Konzept. Es Konzept besagt, daß verschiedene Anwendungen nicht die gleichen Anforderungen an den Transportdienst haben. Daher sollten Protokolle nach dem ALF–Konzept nur Grundfunktionen von Zuverlässigkeit anbieten. Die Anwendung kann diese dann nach eigenem Ermessen erweitern. Dies trifft im speziellen auf die Ordnungserhaltung zu. SRM bietet nur einen halbzuverlässigen Dienst an [Wit99]. Es wurde für das *Whiteboard* (wb), eine kooperative Anwendung, entwickelt. Daher sind Seiten die Grundlage für die Strukturierung der Datenpakete [Wit99].

Im SRM – Protokoll gibt es keinen explizitem Verbindungsaufbau, da es für Anwendungen mit loser Kopplung entwickelt wurde. Da SRM eine empfängerorientiertes Protokoll ist, sind die Empfänger für die Fehlerentdeckung zuständig. Die Datenpakete sind mit Prüfsumme und Sequenznummer versehen, um verlorengegangene Pakete zu entdecken. Das bekannte Problem von empfängerorientierten Protokollen (i.e. der Verlust eines einzelnen Pakets oder Verlust sämtlicher Pakete) wird mittels einem dem Heartbeat (siehe Kapitel 3) ähnlichen Prinzip beseitigt. Diese im SRM eingesetzte Version ist umfangreicher, da die heartbeats genutzt werden, um die Umlaufzeiten der Gruppe abzuschätzen und die Mitglieder der Gruppe kennenzulernen. In SRM werden diese Statusmeldungen *Session Reports* genannt und ent-

halten neben der Sequenznummern noch einen Zeitstempel für die schon genannte Bestimmung der Umlaufzeit. Die Statusmeldungen werden von jedem Empfänger periodisch per Multicast an die gesamte Gruppe gesendet. Jedes Mitglied kann nun an den Statusmeldungen erkennen, inwieweit seine empfangenen Daten korrekt sind. Die Wahrscheinlichkeit, daß ein Mitglied mindestens eine Statusmeldung erhält ist ziemlich hoch. Es sei denn, es liegt eine Netzpartition vor. Eine Fehlerbehebung ist dann aber auch nur möglich, wenn die Netzpartition beseitigt wurde.

Hat ein Empfänger den Verlust eines oder mehrere Pakete festgestellt, sendet er ein NAck per Multicast an die Gruppe. Im SRM–Protokoll kann nun potentiell jedes Mitglied die Übertragungswiederholung vornehmen. Da auch hier die Gefahr der NAck–Implosion besteht, wird NAck–Avoidance mittels eines Zeitgeber benutzt. Ein Empfänger, der ein Paket nicht erhalten hat, wartet eine zufällige Zeit und versendet dann ein NAck per Multicast an die Gruppe. Andere Empfänger, die das Paket ebenfalls nicht erhalten haben und die Statusmeldung (i.e. das NAck) empfangen, unterdrücken ihre negative Quittung (daher der Name NAck–Avoidance). Nach einer gewissen Zeitspanne wird aber auch diese Station eine negative Quittung versenden, falls das betreffende Paket nicht eingetroffen ist. SRM modifiziert dieses Prinzip, indem es die zufällige Zeitspanne abhängig davon macht, wie weit der Empfänger vom ursprünglichen Sender entfernt ist. Dadurch werden Übertragungen von nahegelegenen Stationen wahrscheinlicher als von weiter entfernten Stationen.

Um zu verhindern, daß alle Stationen, welche die negative Quittung erhalten haben, anfangen, das angeforderte Paket zu senden, wird auch ein Zeitgeber verwendet. Dieser Zeitgeber ist wieder in Abhängigkeit der Laufzeitabschätzung. Läuft der Zeitgeber ab, ohne daß das Paket gesendet worden wäre, sendet die Station selber das Paket.

Die Intervallänge der Statusberichte ist abhängig von der Gruppenstärke, um die Skalierbarkeit zu gewährleisten. Je größer die Gruppe ist, desto größer ist das Intervall. Für die Übertragungsverzögerung ist dies kein Problem, da durch die größere Anzahl von Empfängern die Wahrscheinlichkeit steigt einen Statusbericht zu erhalten.

An SRM wird kritisiert, daß das Protokoll in Weitverkehrsnetzen schlechter skaliert, da der NAck–Avoidance Mechanismus nicht mehr so gut funktioniert. Außerdem kann bei ungünstiger Konstellation (i.e. langsames Teilnetz) die Prozessorlast stark ansteigen. Die mangelnde Anwendungsunterstützung ist ebenfalls ein Kritikpunkt, wie der hohe Entwicklungsaufwand beim Einsatz von SRM, da das Protokoll keine wohldefinierte Schnittstelle besitzt wie beispielsweise TCP.

5 Zusammenfassung

Abschließend ist zu sagen, daß aufgrund neuer Anwendungen im Multicast Bereich auch neue Protokolle benötigt wurden. Die bisher entwickelten Protokolle wie *Distance Vector Multicast Routing Protocol* (DVMRP) oder *Multicast Open Shortest Path First* (MOSPF) waren nicht in der Lage, diesen neuen Anforderungen gerecht zu werden. Diese Anforderungen sind ein zuverlässiger Multicast Dienst. Leider ist die Definition von Zuverlässigkeit im Multicast-Bereich sehr schwammig, da verschiedene Anwendungen unterschiedliche Definitionen von Zuverlässigkeit haben. Anders ausgedrückt, es gibt verschiedene Zuverlässigkeitsstufen bei Multicast-Anwendungen. Die Ergebnisse der Forschungen spiegelten dies eindeutig in der Vielzahl und der Unterschiedlichkeit der entwickelten Protokolle wider.

Wesentliche Probleme bei der Entwicklung eines zuverlässigen Multicast–Protokolls ist neben der Implementierung eines zuverlässigen Dienstes auch die Skalierbarkeit der Protokolle. Eine Lösung scheinen die Baum–basierten Protokolle zu liefern. Jedoch wird hier übersehen, daß für die Verwaltung des Baumes mitunter extreme Kosten entstehen, die meist bei der Bewertung solcher Protokolle zu wenig Beachtung finden.

Kritisch anzumerken ist, daß Multicast – Protokolle allgemein schlecht skalieren und fehlerhaft arbeiten, sobald die Größe des Netzes zunimmt. Gemeint sind Netzgrößen im Bereich der Weitverkehrsnetze. Auch Baum–basierte Protokolle haben diesen Schwachpunkt.

Es besteht also durchaus noch Entwicklungsbedarf an zuverlässigen Multicast–Protokollen, die in Weiterverkehrsnetzen gut skalieren und dennoch einen zuverlässigen Dienst anbieten. Außerdem sollte vorher die Definition von Zuverlässigkeit genau geklärt werden oder zumindest die einzelnen Zuverlässigkeitsstufen, die implementiert werden können bzw. müssen.

Literaturverzeichnis

[Ata98] Tanenbaum, A. S.: „Computer Netzwerke“, 3., revidierte Auflage, Prentice Hall, 1998

[Gey99] Geyer, W.: „Das digitale lecture board – Konzeption, Design und Entwicklung eines Whiteboards für synchrones Teleteaching“, Dissertation, Lehrstuhl für Praktische Informatik IV, Universität Mannheim, 1999

[Gru96] Grumann, M.: “Entwurf und Implementierung eines zuverlässigen Multicast – Protokolls zur Unterstützung sicherer Gruppenkommunikation in einer Teleteaching – Umgebung“, Diplomarbeit, Lehrstuhl für Praktische Informatik IV, Universität Mannheim, 1996

[Lev96a] Levine, B. N., Lavo, D., Garcia-Luna-Aceves, J.: „The Case of Concurrent Reliable Multicast Using Shared Ack Trees“. In: Proc. ACM Multimedia, Boston, November 1996, verfügbar unter URL: http://www.cse.ucsc.edu/research/ccrg/publications.html.

[Rmp99] TASC, Inc. „Reliable Multicast Protocols“, URL: http://www.tascnets.com/mist/doc/mcpCompare.html

[RFC98] Mankin, A., Romanow, A., Bradner, S., Paxson, V.: „Request for Comments: 2357“, RFC 2357

[Wit99] Wittmann, R. , Zitterbart, M.: „Multicast: Protokolle und Anwendungen“, dpunkt - Verlag, 1999