Technischer Report der Universität Stuttgart
Nr. 1997/20

Untersuchung der Eignung derzeit existierender Multicast-Protokolle für ein Eventmanagementsystem mit mobilen Teilnehmern

Andreas Schiffer

23.12.1997

CR-Klassifikation: C.2.1, C.2.2, D.4.4, D.4.5, D.4.7

Kurzfassung

Ein Eventmanagementsystem ist ein verteiltes System, das die Übermittlung von Nachrichten (Events) von einem Teilnehmer des Eventmanagementsystems zu einer Gruppe von anderen Teilnehmern des Systems übernimmt. Es handelt sich also um eine 1-zu-n-Nachrichten-Übermittlung, die vom Eventmanagementsystem transparent für die Teilnehmer durchgeführt wird. Dies entspricht der klassischen Definition einer Multicast-Semantik.
Effizienz und Skalierbarkeit des verwendeten Algorithmus sind für die Wahl eines solchen Eventmanagementsystems ebenso entscheidend wie die Zuverlässigkeit der Nachrichtenübermittlung, die Unterstützung mobiler Teilnehmer und die Fehlertoleranz des Systems.

In dieser Arbeit wird eine Übersicht über die bestehenden Multicast-Algorithmen gegeben. Diese werden beschrieben und auf ihre Verwendbarkeit in einem Eventmanagementsystem untersucht. Bei dieser Untersuchung wird besonderer Wert auf die Erfüllung der soeben genannten Kriterien gelegt.

In Kapitel I dieser Arbeit werden zuerst einmal grundlegende Begriffe und Definitionen aus dem Bereich der verteilten Systeme kurz vorgestellt, um die zugrundeliegenden Konzepte und Technologien darzustellen, sowie Begriffe zu klären.

In Kapitel II sollen die Anforderungen, die an ein gutes Eventmanagementsystem gestellt werden, aufgezählt und nach ihrer Wichtigkeit kategorisiert werden. Die hier aufgezählten Anforderungen definieren die Eigenschaften, die ein Multicast-Protokoll haben sollte, um im Rahmen eines Eventmanagementsystems überhaupt oder gar effizient eingesetzt werden zu können.

In Kapitel III werden verschiedene vorhandene Multicast-Protokolle vorgestellt und anhand der Anforderungen aus Kapitel II auf ihre Eignung für den Einsatz in einem Eventmanagementsystem bewertet. Außerdem werden die Kosten, die jeder Algorithmus mit sich bringt, beschrieben. Am Schluß des Kapitels folgt eine tabellarische Zusammenfassung und Gegenüberstellung aller beschriebenen Multicast-Protokolle.

In Kapitel IV sollen dann Möglichkeiten aufgezeigt werden, wie vorhandenen Multicast-Protokollen noch fehlende Eigenschaften wie die Unterstützung von Mobilität, Zuverlässigkeit bei der Nachrichtenübermittlung, eine Ordnungsrelation bei der Auslieferung der Nachrichten oder Fehlertoleranz hinzugefügt werden können. Die Verwendbarkeit der verschiedenen hier aufgezeigten Möglichkeiten wird dabei bewertet.

In Kapitel V wird diese Arbeit dann noch einmal zusammengefaßt, wobei die wesentlichen Ergebnisse dieser Arbeit beschrieben werden. Außerdem wird ein Ausblick auf die Fortführung dieser Arbeit gegeben.

Schließlich folgt mit Kapitel VI der Anhang, in dem ein Literaturverzeichnis und das Glossar stehen.

Inhaltsverzeichnis

I. Einführung

Im ersten Teil der Arbeit werden nach einer Einleitung grundlegende Begriffe und Definitionen aus dem Bereich der verteilten Systeme kurz vorgestellt

I.1 Einleitung

Stellen Sie sich vor, man wollte eine Telefonkonferenz mit mehreren mobilen Teilnehmern, die dazu jeweils ein Handy besitzen, durchführen. Und überlegen Sie einmal, welche Aufgaben ein System erledigen müßte, das eine solche Telefonkonferenz realisieren wollte ohne das vom Mobilfunkstandard GSM automatisch durchgeführte Weiterreichen von Mobilfunkteilnehmern an andere Empfangsstationen (Handover) zu verwenden.
Wenn man nun noch daran denkt, daß dieses System beliebig viele solche Telefonkonferenzen – teils mit unterschiedlichen, teils mit denselben Teilnehmern – ermöglichen soll, und daß die Gespräche zuverlässig, also ohne zeitweise Gesprächsausfälle, übertragen werden sollen, hat man schon verstanden, was die Aufgabe eines Eventmanagementsystems für mobile Teilnehmer ist:

Ein Eventmanagementsystem verbindet eine Menge von Teilnehmern miteinander und leitet die Nachrichten (Events), die ein Teilnehmer sendet, an alle anderen Teilnehmer weiter. Dabei können diese Teilnehmer, wie im obigen Beispiel angedeutet, auch mobil sein, ihren Standort also wechseln. Das Eventmanagementsystem hat hierbei die Aufgabe eine zuverlässige Übertragung der Nachrichten auch zu dem neuen Standort zu gewährleisten.
Die Teilnehmer können zu (sich überlappenden) Gruppen zusammengefaßt sein, wobei jeder Teilnehmer einer Gruppe alle Nachrichten empfängt, die ein anderer Teilnehmer dieser Gruppe sendet.

Um effizient verwendet werden zu können, muß das Eventmanagementsystem vor allem skalierbar und zuverlässig sein, aber auch geringe Kosten sind ein wichtiger Faktor. All diese Forderungen zu erfüllen ist durchaus keine triviale Aufgabe. Daher ist es das Ziel dieser Arbeit, zu Erforschen, inwieweit die betrachteten Multicast-Algorithmen diese Forderungen erfüllen und sich somit für den Einsatz in einem Eventmanagementsystem eignen.

I.2 Unicast / Broadcast / Multicast

Im Netzwerk kennt man drei prinzipiell unterschiedliche Arten, Nachrichten zu versenden: Unicast, Broadcast und Multicast.

I.2.1 Unicast

Ein Unicast ist das Versenden einer Nachricht von einem Sender zu genau einem Empfänger. Es handelt sich also um eine 1-zu-1-Übermittlung. Dieser Bereich ist in der Literatur bereits sehr ausführlich diskutiert und erforscht worden.

I.2.2 Broadcast

Ein Broadcast ist das Versenden einer Nachricht von einem Sender zu allen anderen Knoten des Netzwerks. Dies ist also eine 1-zu-alle-Übermittlung. Oft wird auch ein sogenannter Hop-Counter eingesetzt, um zu verhindern, daß die Nachricht an wirklich alle Knoten des Netzwerks gesendet wird. Statt dessen wird die Nachricht nur an alle Knoten in einem Teilbereich des Netzwerks gesendet.

In dieser Arbeit soll unter einem Broadcast ein Multicast an alle Router (s.u.) verstanden werden, da die hier betrachtete Netzwerkstruktur, wie sie weiter unten ausgeführt wird, davon ausgeht, daß nur das Multicasting über alle Router betrachtet werden muß, weil die weitere Auslieferung über die LANs mit Hilfe eines LAN-Multicasts, der bereits vorhanden ist, geschieht. Somit ist die kleinste Einheit im Sinne der zu betrachtenden Multicast-Protokolle die der Router.

I.2.3 Multicast

Ein Multicast ist das Versenden einer Nachricht von einem Sender zu einer Gruppe von Empfängern. Hier handelt es sich also um eine 1-zu-n-Übermittlung. Wie man leicht sieht, ist dies eine Verallgemeinerung sowohl des Unicast als auch des Broadcast. Der Sender eines Multicast muß nicht notwendigerweise die Adressen aller Empfänger kennen; er kann auch seine Nachricht an eine für diese Gruppe reservierte Multicast-Adresse senden, und das Netzwerk übernimmt dann die weitere Verteilung der Nachricht.

Im Prinzip wäre es möglich, einen Multicast durch eine Folge von Unicasts zu realisieren. Das bringt allerdings entscheidende Nachteile mit sich:

- Verschwendung von Bandbreite: Wenn der Weg zu mehreren Empfängern des Multicasts streckenweise über die gleichen Kanten des Netzwerks verläuft, ist es unnötiger Verbrauch von Bandbreite, daß mehrere Nachrichtenpakete mit gleicher Information über die gleiche Kante gesendet werden.

- Nicht effizient: Der Sender ist evtl. lange damit beschäftigt, die Nachricht an alle Empfänger zu senden, was ihn viel Rechenzeit kosten kann. Auch die Dauer, bis der letzte Empfänger seine Nachricht erhalten hat, kann sehr groß werden, was gerade bei Echtzeitdaten inakzeptabel ist.

- Keine Anonymität: Jeder Sender muß alle Empfänger einer Gruppe kennen. Wenn ein Rechner zu einer Gruppe dazustoßen oder eine Gruppe verlassen will, müssen alle Sender über diese Änderung benachrichtigt werden.

I.2.4 Internet Group Multicast Protocol (IGMP)

Man sieht also, daß es einfacher und ressourcenschonender ist, einen vom Netzwerk angebotenen Multicast-Mechanismus zu verwenden. Hierzu gibt es verschiedene Protokolle, von denen einige später in dieser Arbeit vorgestellt und untersucht werden. Es gibt eine Standardschnittstelle für die Verwendung solcher Multicast-Mechanismen von der Anwendung aus: Das Internet Group Multicast Protocol (IGMP). Es ist in [RFC 966] beschrieben und unterstützt das Erzeugen einer Multicast-Gruppe (**CreateGroup**), das Beitreten zu einer solchen Gruppe (**JoinGroup**), das Verlassen einer Gruppe (**LeaveGroup**), sowie das Senden (**SendMulticast**) und Empfangen (**ReceiveMulticast**) von Multicast-Nachrichten.

CreateGroup wird oft implizit mit dem ersten JoinGroup an eine neue Gruppe erledigt, und ReceiveMulticast wird selten direkt aufgerufen, da die Multicast-Nachricht meist direkt und ohne Nachfrage an den entsprechenden Prozeß ausgeliefert werden. Somit kommen die meisten Multicast-Protokolle mit drei Befehlen aus: JoinGroup, LeaveGroup und SendMulticast.

Eine Multicast-Nachricht wird als normale UDP-Nachricht an den Multicast-Router des LANs gesendet, der das weitere Versenden an andere Multicast-Router und letztlich an die Empfänger der Multicast-Gruppe entsprechend seinem Multicast-Protokoll übernimmt. Diese Übermittlung geschieht auf Ebene der Netzwerkschicht des OSI-7-Schichten-Modells. Effiziente Ansätze für Multicast auf Ebene der Transportschicht sind derzeit noch Gegenstand der Forschung.

I.3 Klassen von Multicast-Protokollen

Es gibt zwei grundsätzlich verschiedene Verfahrensweisen, wie Multicast-Funktionalität in einem Rechnernetz implementiert wird, und dementsprechend kann man die Multicast-Protokolle in zwei Klassen einteilen:

I.3.1 Sender Based Trees (SBT)

Ein Multicast, der über Sender Based Trees realisiert ist, baut für jeden möglichen Sender von Multicast-Nachrichten einen Auslieferungsbaum aus, d.h. jede Multicast-Nachricht eines Senders wird auf annähernd kürzesten Wegen vom Sender zu den Empfängern gesendet.

Vorteile von SBT gegenüber MST (Minimum Spanning Tree, siehe Kapitel I.3.2):

- Geringe Auslieferungsdauer wegen kürzester Wege vom Sender zu allen Empfängern, während beim MST der Auslieferungsbaum nur in seltenen Fällen den kürzesten Wegen vom Sender zu den Empfängern entspricht.
  Tatsächlich ist der MST nicht viel schlechter als die SBT: David Wall beweist [Wall 80], daß die Auslieferungsdauer bei MST maximal doppelt so groß ist wie die der SBT. Simulationen mit 50-Knoten-Netzwerken [Deering 96] ergeben jedoch, daß der tatsächliche Faktor statt bei 2,0 nur um 1,1 oder 1,2 liegt.

- Relativ gleichmäßige Verteilung der Netzlast über das gesamte Netzwerk, während beim MST eine erhöhte Netzlast in der Nähe der Wurzel des MST entsteht.

- Höhere Fehlertoleranz: Beim MST ist der Baum bei Ausfall nur eines Knotens bzw. einer Kante des Baumes partitioniert und muß neu berechnet werden, während bei SBT möglicherweise die meisten Sender noch alle Empfänger erreichen.

I.3.2 Minimum Spanning Tree (MST)

Ein Multicast, der nach dem Minimum Spanning Tree-Verfahren realisiert ist, baut nur einen einzigen Baum auf: Den minimalen Spannbaum über alle Sender und Empfänger der Multicast-Gruppe. Jede Multicast-Nachricht wird über diesen Baum verteilt, unabhängig davon, von welchem Sender sie stammt.

Vorteile von MST gegenüber SBT:

- Es muß nur ein einziger Baum aufgebaut werden, während bei SBT für jeden neuen Sender ein zusätzlicher, neuer Baum aufgebaut werden muß.

- In den Routern, über die der Baum sich erstreckt, müssen wesentlich weniger zusätzliche Informationen gespeichert werden: Nur die Informationen für einen Baum pro Multicast-Gruppe statt für jeden Sender jeder Multicast-Gruppe einen Baum speichern zu müssen. Somit liegt der Speicheraufwand pro Router in der Größenordnung von N statt von S*N, wobei N die Anzahl der Multicast-Gruppen und S die Anzahl der Sender sei.

I.4 Angenommene Netzwerkstruktur

In diesem Kapitel soll die Netzwerkstruktur beschrieben werden, die in dieser Arbeit als gegeben angenommen und vorausgesetzt wird.

I.4.1 LANs mit Intranetzwerk-Multicastdienst

Als zugrundeliegende Netzwerkstruktur wird die Situation vorausgesetzt, wie sie derzeit im Internet vorzufinden ist. Diese Struktur wird auch in den meisten Artikeln (z.B. [Belkeir 89], [Deering 90]) angenommen. Sie besteht aus LANs, die selbst einen effiziente Intranetzwerk-Multicastdienst bzw. Intranetzwerk-Broadcastdienst bieten. Dies kann bei busbasierten LANs ebenso wie bei ringbasierten LANs leicht implementiert werden und kann deshalb im weiteren vorausgesetzt werden. Die einzelnen Rechner eines LANs haben Adreßfilter in ihren LAN-Interfaces, die Datenpakete, an denen sie kein Interesse haben, erkennen und verwerfen können.

I.4.2 LANs ohne Intranetzwerk-Multicastdienst

In LANs, die aus irgendwelchen Gründen keinen effizienten Intranetzwerk-Multicastdienst bzw. Intranetzwerk-Broadcastdienst bieten, können die einzelnen Rechner ohne Beschränkung der Allgemeinheit im weiteren als einzelne LANs, die jeweils aus nur einem Rechner bestehen, betrachtet werden.

I.4.3 Verbindung zwischen LAN und Netzwerk: Router

LANs haben eine oder mehrere Verbindungen zum übrigen Netzwerk. Diese Verbindung geschieht durch Bridges, Router oder Gateways, im weiteren unter dem Sammelbegriff Router zusammengefaßt.

Router sind durch Punkt-zu-Punkt-Verbindungen, über LANs oder über dazwischen liegende Router miteinander verbunden. Router gehören zu einem oder mehreren LANs, empfangen alle in diesen LANs gesendeten Multicast-Nachrichten und senden diese entsprechend weiter an das übrige Netzwerk. Zudem geben sie alle von ihnen aus dem Netzwerk empfangenen Multicast-Nachrichten, die für ein ihnen benachbartes LAN bestimmt sind, an dieses weiter. Multicasting muß in dieser vorausgesetzten Netzwerkstruktur also nur von Router zu Router betrachtet werden und nicht von Rechner zu Rechner. Router übernehmen nicht nur das Routing zu anderen Routern und LANs, sondern auch Multicast-Aufgaben: Sie müssen also Informationen halten und verwalten, die das Multicast-Protokoll benötigt, um seine Arbeit zu tun. Außerdem müssen sie den Multicast-Algorithmus selbst ausführen. Wenn sie diese Multicast-Aufgaben nicht ausführen können, muß ein Rechner in dem LAN, das mit dem Router verbunden ist, diese Funktionen übernehmen.

Um zu vermeiden, daß Multicast-Nachrichten an LANs, die mehrere Router haben, mehrfach ausgeliefert wird, also von jedem Router, wird meist ein Router als der Multicast-verantwortliche Router für dieses LAN festgelegt. Nur dieser Router ist dafür verantwortlich, daß Multicast-Nachrichten von und zu diesem LAN gelangen. Im folgenden ist eine solche Konstellation gemeint, wenn von dem Multicast-Router eines LANs die Rede ist.

I.4.4 Multicast-Funktionalität mit IGMP

Um die Multicast-Funktionalität in den einzelnen Rechnern zu bieten, läuft auf jedem Rechner ein Multicast-Manager (MM). Dieser bietet auf der einen Seite für die Prozesse auf dem Rechner Multicast-Funktionalität an und handelt auf der anderen Seite das Multicasting mit anderen MMs in seinem LAN und mit dem Netzwerk aus.

Die Prozesse eines Rechners senden ihrem MM mit Hilfe von rechnerinterner Kommunikation Befehle wie SendMulticast, CreateGroup, JoinGroup oder LeaveGroup. Der MM weiß somit immer, welche Multicast-Gruppen auf seinem Rechner vertreten sind und wieviel Teilnehmerprozesse jede dieser Gruppen auf diesem Rechner hat.

Einer **SendMulticast**-Aufforderung kommt er nach, indem er die Multicast-Nachricht mit einem LAN-Multicasts an diese Multicast-Gruppe in diesem LAN sendet. Jeder MM in diesem LAN, sowie der Multicast-Router des LANs, empfangen die Nachricht. Der Multicast-Router braucht sich nun nur noch um die weitere Verteilung der Nachricht im Netzwerk zu kümmern.

Um zu erfahren, welche **Multicast-Gruppen** in seinem LAN vertreten sind, sendet der Multicast-Router periodisch eine „IGMP Host Membership Query“ an alle MMs in seinem LAN. Jeder MM antwortet darauf mit einem „IGMP Host Membership Report“. Um eine hohe Belastung des LANs durch Report-Nachrichten nach einer Query-Nachricht zu verhindern, wartet jeder MM nach dem Empfangen der Query-Nachricht eine Zufallszeit ab und sendet dann seine Report-Nachrichten mit einem LAN-Multicast jeweils an die betreffende Multicast-Gruppe in diesem LAN statt direkt an den Multicast-Router. Wenn ein MM mithört, daß ein anderer MM eine Report-Nachricht für eine Multicast-Gruppe sendet, für die auch er eine Report-Nachricht zu senden hätte, kann er das Senden dieser Nachricht unterdrücken. Der Multicast-Router braucht ja nur zu wissen, daß es Teilnehmer dieser Gruppe in seinem LAN gibt, nicht wieviel es sind. Wenn ein Prozeß bei seinem MM einer Multicast-Gruppe beitreten will, die in diesem LAN noch nicht vertreten ist, braucht dieser keine Query-Nachricht abwarten, um seine Report-Nachricht zu senden.

I.5 Hierarchisches Multicast Routing

Die meisten Unicast-Protokolle und Multicast-Protokolle skalieren schlecht, weil sie Informationen über alle anderen Router im Netzwerk, über deren Zustand und/oder über deren Verbindungen zu anderen Routern speichern müssen, oder weil sie immer wieder Verwaltungsinformationen an alle Router im Netzwerk broadcasten müssen. Daher steigen ihre Speicherkosten bzw. der Verbrauch an Bandbreite im Netzwerk in gleichem Maße wie die Zahl der Rechner im Netzwerk. In einem stark expandierenden Netzwerk wie dem Internet sind schnell die Grenzen an Speicherplatz in den Routern bzw. Bandbreite erreicht.

Die für Unicast-Algorithmen übliche Lösung dieses Problems liegt in der Einführung von hierarchischem Routing [Kleinrock 77]. Diese Lösung besteht darin, ein großes Netzwerk hierarchisch in mehrere, sich nicht überlappende Netzwerkbereiche, sogenannte Domänen, aufzuteilen. Eine solche Domäne wird dann von der darüberliegenden Netzwerkebene wie eine einzige Verbindung, bzw. wie ein einziges LAN, behandelt. Entsprechend werden Nachrichten nur bis zu einem Router, der mit dieser Domäne direkt verbunden ist, geroutet, und das domäneninterne Routing bleibt der Domäne selbst überlassen.

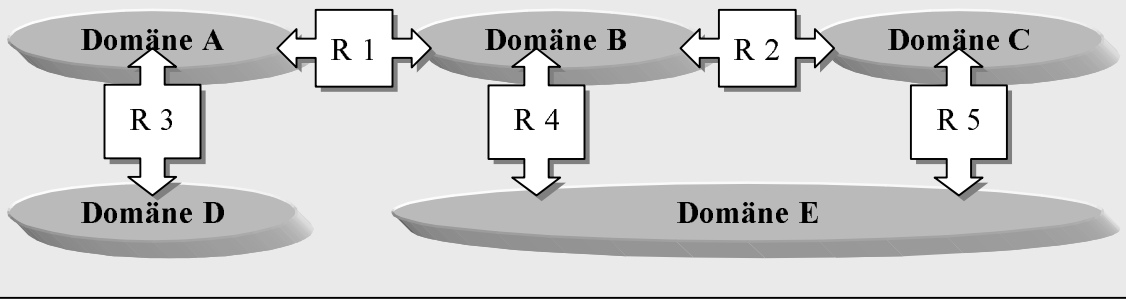

Abbildung I–1: Beispiel zu „Hierarchisches Routing zwischen Domänen“

Die gleiche Lösung kann nach [Deering 90] und [Thyagarajan 95] für Multicast-Algorithmen benutzt werden, um diesen zu ermöglichen, trotz schlechtem Skalierungsverhalten effizient und kostengünstig in großen Netzwerken zu arbeiten. Hierzu wird das Netzwerk in Domänen gegliedert, die wiederum in Subdomänen aufgeteilt werden können, etc.

Eine Domäne wird von der darüberliegenden Netzwerkebene wie ein LAN behandelt und muß (mindestens) einen für sie zuständigen Multicast-Router haben, der Multicast-Nachrichten des Netzwerks an die Domäne weiterleitet, falls es Teilnehmer dieser Multicast-Gruppe in der Domäne gibt, und der Multicast-Nachrichten aus der Domäne weiterleitet an die darüberliegende Netzwerkebene. Der Multicast-Router der Domäne muß die Informationen darüber verwalten, welche Multicast-Gruppen in seiner Domäne vertreten sind.

Domänen müssen, genauso wie LANs in der bisherigen Betrachtungsweise, in der Lage sein, Transitverkehr zu ermöglichen, d.h. Multicast-Nachrichten von einem Router der darüberliegenden Netzwerkebene zu einem anderen, an dieser Domäne angeschlossenen, Router der gleichen Netzwerkebene weiterzuleiten.

Ein weiterer Vorteil dieser Lösung neben der Beseitigung des Skalierungsproblems ist, daß in den Domänen nicht das gleiche Multicast-Protokoll verwendet werden muß, wie auf der darüberliegenden Netzwerkebene oder wie in Subdomänen. Da es klare Schnittstellen gibt, kann jede Domäne ihr eigenes Multicast-Protokoll verwenden, ohne dabei auf Probleme zu stoßen. Jede Domäne muß sich nur um ihr eigenes Multicasting kümmern und nicht um das ihrer Subdomänen. Und das gilt für alle Hierarchieebenen.

I.6 Multicast-Protokolle im Internet

In diesem Kapitel soll die Geschichte und Entwicklung der Multicast-Protokolle im Internet beschrieben werden. Es soll aufgezeigt werden, welche verschiedenen Multicast-Protokolle derzeit im Internet vorzufinden sind.

I.6.1 Distance Vector Multicast Routing Protocol (DVMRP)

Der erste Multicast-Algorithmus, der im Internet eingesetzt wurde, war der DVMRP-Algorithmus (Beschreibung siehe Kapitel III.3). Infolgedessen wurde er auch **IP-Multicast** genannt. Im März 1992 wurde der MBone, also der Multicast Backbone des Internet, mit 40 Subnetzen (LANs) ins Leben gerufen, und der DVMRP war der Multicast-Algorithmus für diesen MBone. Heute besteht der MBone aus mehreren tausend Subnetzen und ist somit weit verbreitet.

Der DVMRP ist ein SBT-Algorithmus, der eng gekoppelt mit dem zugrundeliegenden IP-Unicast-Algorithmus ist. DVMRP kann nur sinnvoll und einigermaßen kostengünstig implementiert werden, wo der DVRA (Distance Vector Routing Algorithm) als IP-Unicast-Algorithmus zugrunde liegt. Das ist im größten Teil des Internet der Fall, da der DVRA der Unicast-Algorithmus ist, der früher der Standard-Unicast-Algorithmus im damaligen Arpanet, dem heutigen Internet war.

I.6.2 Multicast Extensions for „Open Shortest Path First“ (MOSPF)

Da DVMRP wegen seiner starken Abhängigkeit zu DVRA nur in Teilen des Internet eingesetzt werden kann, mußten für andere Unicast-Algorithmen andere Multicast-Lösungen gefunden werden. So wurde MOSPF (Beschreibung siehe Kapitel III.4) entwickelt als Multicast-Protokoll für die Teile des Internet, in denen das OSPF-Verfahren (Open Shortest Path First) als Unicast zugrunde liegt. OSPF ist ein Link-State-Routing-Algorithmus und wird seit geraumer Zeit als der Nachfolger des DVRA, als Standard-Unicast-Algorithmus für das Internet und als „New Arpanet“ propagiert.

Der MOSPF ist, ebenso wie der DVMRP, ein SBT-Algorithmus, der sehr abhängig von dem zugrundeliegenden IP-Unicast-Algorithmus, in diesem Fall OSPF, ist. Das liegt daran, daß beide Protokolle, DVMRP und MOSPF, stark von den Informationen abhängen, die von den zugrundeliegenden IP-Unicast-Algorithmen ausgetauscht werden, und eine zusätzliche Implementierung dieses Informationsaustauschs sehr kostenungünstig und aufwendig wäre.

I.6.3 Core-based Trees (CBT)

Die bisherigen Multicast-Algorithmen, DVMRP und MOSPF, hatten eine Reihe schwerwiegender Nachteile: Starke Abhängigkeit von dem zugrundeliegenden IP-Unicast-Algorithmus, schlechte Skalierbarkeit, hohe Kosten, hoher Verbrauch an Bandbreite und Komplexität.

Als Lösung für diese Probleme wird in [Ballardie 93] wird ein neuer Multicast-Algorithmus beschrieben, der eine bessere Alternative zu DVMRP und MOSPF bieten soll: CBT (Beschreibung siehe Kapitel III.5). Dieser Multicast gehört zur Klasse der MST-Algorithmen, ist unabhängig von dem darunterliegenden Unicast-Algorithmus und bietet Effizienz und bessere Skalierbarkeit. Er wird in Teilen des Internet als Multicast-Algorithmus verwendet.

I.6.4 Protocol Independent Multicast (PIM)

Im Jahr 1995 wurde von Deering und anderen [Deering 95] ein weiterer, „noch besserer" Multicast für das Internet beschrieben: PIM (Beschreibung siehe Kapitel III.6). Er besteht aus zwei Teilen: PIM-DM (PIM-Dense Mode) und PIM-SM (PIM-Sparse Mode).

PIM-DM [Deering 97] entspricht im großen und ganzen dem bisherigen Internet-Multicast-Verfahren DVMRP. Die große Neuigkeit bei **PIM-SM** [RFC 2117] ist, daß dies ein **MST-Algorithmus** ist, der auf Wunsch des Multicast-Empfängers für bestimmte (sehr aktive) Sender in einen **SBT-Modus** gehen kann, so daß für diese Sender ein senderbasierter Auslieferungsbaum aufgebaut wird. Dies verspricht höhere Effizienz, Kostenersparnis und sehr geringe Auslieferungszeiten für bestimmte Sender, außerdem eine bessere Fehlertoleranz der Sender, die im SBT-Modus senden, da ein Knoten- oder Kantenausfall im MST diese Sender nicht betreffen muß. PIM-SM verspricht somit die Nutzung sowohl der Vorteile der MST als auch der SBT.

Da PIM ein Projekt der Internet Engineering Task Force ist (PIM-DM ist derzeit Internet Draft, während PIM-SM bereits RFC ist) und neben der Autorenschaft des „Multicast-Gurus" Stephen Deering auch von der bekannten Router-Firma Cisco unterstützt wird (In der neuesten Generation von Cisco-Routern ist PIM bereits implementiert), ist zu erwarten, daß PIM über kurz oder lang in größeren Teilen des Internet eingesetzt werden wird.

I.7 Events

In diesem Kapitel werden die Konzepte von Events und Eventmanagern vorgestellt, sowie das Modell des Eventmanagementsystems, das dieser Arbeit zugrunde gelegt wird. Hierbei werden dieselben Definitionen verwendet wie in [Beck 97].

I.7.1 Der Event

Event heißt wörtlich übersetzt Ereignis. Analog zu objektorientierten Sprachen und Systemen ist ein Event in unserem Modell die Nachricht, daß ein bestimmtes Ereignis eingetreten ist oder eintreten soll. Eine ereignisorientierte Sicht der Dinge erleichtert den Entwurf einer Anwendung ungemein, da sich Sachverhalte der realen Welt unter Verwendung von Events oft einfacher beschreiben lassen als mit prozeduralen Beschreibungen. Der Event ist ein (persistentes) Objekt, das beliebige Inhalte transportiert. Der Event-Typ beschreibt diese Inhalte, entspricht also der Klasse des Event-Objektes.

I.7.2 Der Eventchannel

Events werden von Event-Erzeugern („Supplier“) erzeugt und von Event-Verbrauchern („Consumer“) empfangen. Prinzipiell darf jedes Objekt, das den Eventmanager verwenden will, als Consumer oder Supplier auftreten. Supplier und Consumer kommunizieren über einen gemeinsamen Kommunikationskanal, den wir „Eventchannel“ nennen. Der Eventchannel garantiert bestimmte Dienstmerkmale der Kommunikation. Supplier und Consumer müssen sich auf einen gemeinsamen Eventchannel geeinigt und dort angemeldet haben, bevor sie mit der Kommunikation beginnen. Pro Eventchannel dürfen beliebig viele Supplier und Consumer auftreten. Es handelt sich also um eine n:m-Kommunikation.

I.7.3 Der Eventmanager

Der Eventchannel ist die logische Sicht zur Verteilung von Events zwischen Suppliern und Consumern und beschreibt die Dienstmerkmale. Der Eventmanager stellt diese Funktionalität zur Verfügung und ist für die korrekte Abarbeitung, Fehlerkorrektur, Verwaltung und die Einhaltung aller gestellten Bedingungen zuständig.

I.7.4 Modell des Eventmanagers

Abbildung I–2 zeigt die Architektur des Eventmanagers. Jede Anwendung, die die Dienste des Eventmanagers verwenden möchte, implementiert einen lokalen Eventmanager. Dieser bildet den Zugangspunkt zum Eventmanagementsystem und kommuniziert mit einem Event-Dämon, der für die optimale, d.h. möglichst ressourcenschonende, Verteilung der Events zuständig ist. In einer optimalen Konfiguration ist pro beteiligtem Rechner ein Dämonprozeß vorhanden, der mit anderen Dämonprozessen kommuniziert. Die Anwendungen kommunizieren über ihren lokalen Eventmanager unter Verwendung lokaler Kommunikationsmechanismen mit dem Event-Dämon.

Man kann sich allerdings leicht vorstellen, bereits für diesen Schritt eine verbindungsorientierte Kommunikation zwischen lokalem Eventmanager und Event-Dämon einzuführen. Dies hätte den Vorteil, daß auch Anwendungen auf Rechnern, auf denen kein Event-Dämon aktiv ist (z.B. „Personal Digital Assistants“, PDAs), an der Event-Kommunikation teilnehmen können. Ihr lokaler Eventmanager kommuniziert dann nicht mit dem lokalen, sondern mit einem entfernten Event-Dämon.

Für die Verteilung der Events zwischen den Event-Dämonen ist der eingesetzte Multicast-Mechanismus zuständig. Das Konzept der lokalen Eventmanager verwenden wir hauptsächlich dazu, um den Eventmanager von den Anwendungen zu trennen und so eine möglichst universelle Nutzung zu ermöglichen. Durch die eindeutige Schnittstelle ist es einfacher, statt einem selbst entwickelten Eventmanager evtl. einen kommerziellen Eventmanager, wie z.B. die Orbix Event Services, für die Verteilung der Events zu verwenden.

Ferner nehmen wir an, daß die Interprozeßkommunikation der Anwendungsprozesse mit dem Event-Dämon im Verhältnis zur Netzwerkkommunikation sehr billig ist. Wenn im folgenden von Eventmanager die Rede ist, so ist damit die Gesamtheit aller Prozesse und Kommunikationspfade gemeint, die für das Management und Verteilen von Events zuständig sind.

Nachdem nun die Struktur des Eventmanagers klar ist, steht auch die Aufgabe fest, die ein Multicast-Mechanismus im Eventmanagementsystem übernehmen muß: Die effiziente und ressourcenschonende Verteilung von Events zwischen den Event-Dämonen.

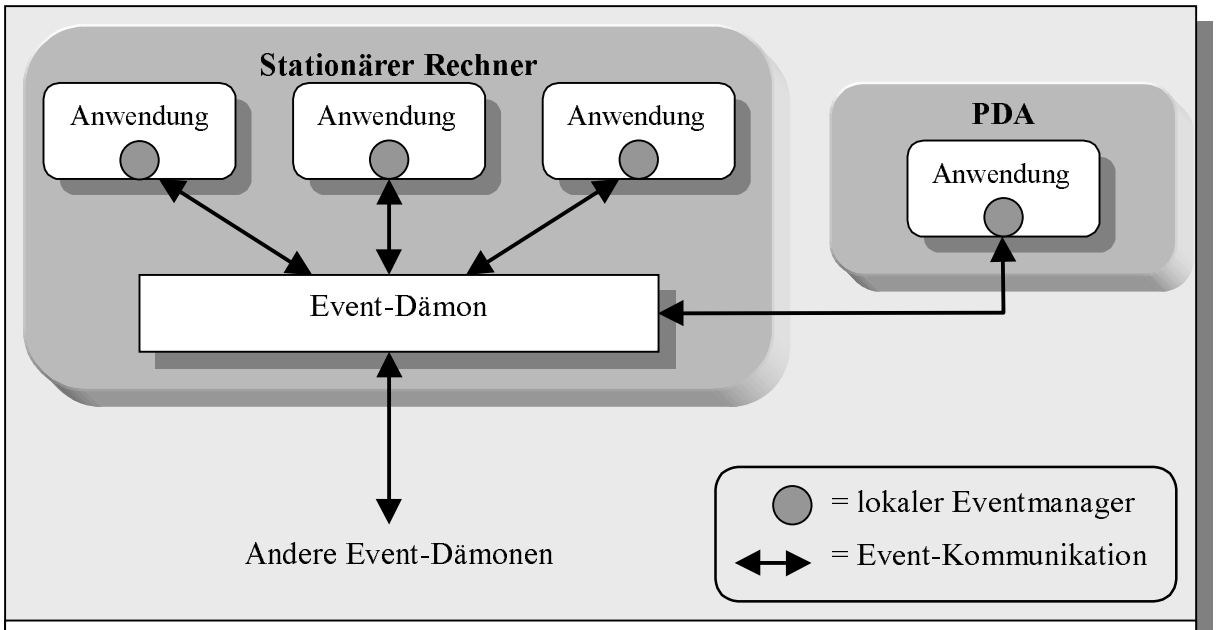

Abbildung I–2: Modell des Eventmanagers und Anbindung an die Anwendung. Zwischen den Event-Dämonen werden Events über einen Multicast-Mechanismus verteilt

II. Anforderungen an Multicast-Protokolle

Um Multicast-Protokolle für die Eignung für ein Eventmanagementsystem bewerten zu können, müssen zuerst die Eigenschaften definiert werden, die ein Multicast-Protokoll haben sollte, um im Rahmen eines Eventmanagementsystems überhaupt oder gar effizient eingesetzt werden zu können. Diese Eigenschaften werden im folgenden in vier Klassen gegliedert.

II.1 Notwendige Eigenschaften

Diese Eigenschaften muß das Protokoll bieten, um den Anforderungen gerecht zu werden.

II.1.1 Skalierbarkeit

Der Eventmanager soll sowohl in kleinen Systemen mit mobilen Teilnehmern sowie auch in möglicherweise sehr großen und weitverteilten System eingesetzt werden. Daher ist Skalierbarkeit, also die Eigenschaft, daß der Aufwand nicht überproportional zur Größe des Systems wächst, ein wichtiger Punkt.

II.1.2 Verteiltheit

Die Verteiltheit des Protokolls spielt eine entscheidende Rolle, denn wenn das Protokoll nur auf einer zentralen Instanz abläuft, wird diese zwangsläufig mit steigender Systemgröße irgendwann zum Flaschenhals. Außerdem würde eine zentrale Instanz auch einen Single-Point-Of-Failure bieten.

II.1.3 Anonymität von Multicast-Teilnehmern

Sender und Empfänger brauchen sich nicht zu kennen. Sie kennen nur eine Multicast-Adresse, bzw. im Fall des Eventmanagements einen Eventchannel, an die sie Nachrichten schicken und von der sie Nachrichten empfangen. Diese Anonymität ermöglicht bzw. vereinfacht dynamisches Hinzufügen und Entfernen von Teilnehmern einer Multicast-Gruppe.

II.1.4 Unterstützung von Mobilität

Eine entscheidende Eigenschaft des gesuchten Protokolls ist die effiziente Unterstützung von mobilen Teilnehmern, da sowohl Sender als auch Empfänger einer Multicast-Gruppe migrieren, also ihren Aufenthaltsort im Rechnernetz beliebig wechseln, können sollten.
Weil die Protokolle oft unterschiedliche Qualität für die Unterstützung von Mobilität für Sender und Empfänger zeigen, ist es sinnvoll, **Sendermobilität** und **Empfängermobilität** getrennt zu untersuchen.
Im weiteren soll nicht die Mobilität innerhalb eines LAN betrachtet werden, da aufgrund der Netzwerkstruktur, die vorausgesetzt wird, Multicast-Bäume über Router aufgebaut werden und nicht über einzelne Rechner in einem LAN. Router, an die ein LAN mit Teilnehmern einer Multicast-Gruppe angeschlossen ist, bekommen somit jede Multicast-Nachricht für diese Gruppe und senden diese weiter über die Multicast-Funktionalität des LANs. Aus diesem Grund liegt die Unterstützung der Mobilität innerhalb eines LANs nicht im Aufgabenbereich der hier besprochenen Protokolle, denn diese Aufgabe wird dem Multicast- (oder Broadcast-) Mechanismus des LANs überlassen, von dem angenommen wird, daß er sie effizient erfüllt.

II.2 Auf höheren Schichten implementierbare, notwendige Eigenschaften

Diese Eigenschaften muß das Protokoll bieten, um den Anforderungen gerecht werden zu können. Wenn das Protokoll diese Eigenschaften jedoch nicht bietet, ist das kein k.o.-Kriterium, da diese Eigenschaften auch von einer weiter oben liegenden Schicht auf ein existierendes Protokoll aufgesetzt werden können.

II.2.1 Zuverlässigkeit

Aus der Aufgabenstellung geht hervor, daß eine zuverlässige Zustellung der Events an alle Teilnehmer garantiert sein muß. Im Gegensatz zu dem, im Internet verwendeten Internet Protocol (IP), das nur eine „Best Effort"-Zuverlässigkeit bietet, wird hier gefordert, daß Events in jedem Fall alle Teilnehmer erreichen. Allerdings hat auch diese Zuverlässigkeit ihre Grenzen, da es keine Möglichkeit gibt, Teilnehmern, die aufgrund von Netzpartitionierungen nicht erreichbar sind, vor der Wiederherstellung der Erreichbarkeit die Events von dem abgeschnittenen Teil des Netzes zuzustellen.

II.2.2 Ordnung von Events

Für manche Zwecke ist es erforderlich, daß eine gewisse Ordnungsrelation für die Auslieferung von Events gefordert werden kann. Hier kann man folgende Ordnungsrelationen in Betracht ziehen [Jalote 94]:

- **FIFO-Ordnung**: Bei allen Empfänger werden die Nachrichten in der Reihenfolge ausgeliefert, in der der Sender sie gesendet hat, d.h. wenn ein Sender Nachricht A vor Nachricht B sendet, liefert jeder Empfänger Nachricht A vor Nachricht B aus.

- **Atomare Ordnung**: Bei allen Empfängern werden die Nachrichten in der gleichen Reihenfolge ausgeliefert, d.h. wenn bei irgendeinem Empfänger Nachricht A vor Nachricht B ausgeliefert wurde, werden beide Nachrichten bei allen anderen Empfängern in der gleichen Reihenfolge ausgeliefert. In welcher Reihenfolge dies aber geschieht, bleibt dem Algorithmus überlassen.

- **Kausale Ordnung**: Diese Ordnung arbeitet wie die atomare Ordnung, allerdings mit der Erweiterung, daß eine Nachricht, die ein Sender S sendet, nachdem er andere Nachrichten empfangen oder gesandt hat, bei allen Empfängern erst ausgeliefert wird, nachdem die von S zuvor empfangenen oder gesandten Nachrichten ausgeliefert wurden.

Totalordnung: Die Nachrichten werden global über alle beteiligten Prozesse hinweg geordnet. Es handelt sich praktisch um eine atomare Ordnung, bei der die Reihenfolge nicht dem Algorithmus überlassen ist, sondern global und eindeutig festgelegt ist.

II.3 Wichtige Eigenschaften

Dies sind Eigenschaften, die das Protokoll nicht in vollem Maße anbieten muß. Schneiden diese Eigenschaften jedoch sehr schlecht ab, ist dies ziemlich nachteilig.

II.3.1 Fehlertoleranz

Das gesuchte Protokoll sollte auftretende Fehler, wie z.B. den Ausfall eines Rechners oder die Unterbrechung einer Leitung zwischen zwei Rechnern, ausgleichen können, wenn es sich nicht um eine Netzpartitionierung handelt. Dies kann beispielsweise durch den Aufbau einer Umleitung für die Events geschehen oder durch den Neuaufbau des Multicast-Baumes. Allerdings kann diese Eigenschaft in vielen, wenn auch nicht allen, Fällen in das existierende Protokoll eingebaut werden, auch wenn diese Eigenschaft bisher darin noch nicht vorgesehen ist.

II.3.2 Geringe Auslieferungsverzögerung

Die Dauer zwischen dem Senden einer Multicast-Nachricht und dem Empfangen der Nachricht bei den einzelnen Teilnehmern sollte möglichst gering sein. Hier sind naturgemäß die SBTs im Vorteil, da sie für jeden Sender das Versenden der Nachrichten auf dem direkten Weg zwischen dem Sender und den einzelnen Empfängern über einen eigenen Auslieferungsbaum anbieten.

II.3.3 Geringe Kosten

Die Kosten, die ein Protokoll erzeugt, können in mehrere Kostenarten gegliedert werden:

- **Rechenzeit**, die benötigt wird, um z.B. einen neuen Auslieferungsbaum für die Multicast-Nachrichten zu berechnen.

- **Bandbreite**, also die Netzlast, die durch das Verschicken von Multicast-Nachrichten erzeugt wird, sowie der Overhead an Nachrichtenverkehr für die Erstellung und Wartung der Auslieferungsbäume.

- **Speicherplatz** im Router für die Speicherung von Verwaltungsinformationen für die Auslieferungsbäume.

II.4 Wünschenswerte Eigenschaften

Dies sind vorteilhafte Eigenschaften, die das Protokoll jedoch nicht unbedingt bieten muß.

II.4.1 Verfügbarkeit als Basisdienst

Für die Effizienz ist es von Vorteil, wenn das Multicast-Protokoll bereits systemnah als Basisdienst implementiert ist, möglichst sogar direkt im Router. Ein Multicast-Protokoll auf einer systemnahen Schicht ist meist effizienter implementiert und bringt somit oft große Geschwindigkeitsvorteile gegenüber Implementierungen auf einer höheren Ebene, z.B. in einer Programmiersprache wie Java.

II.4.2 Einfache API-Schnittstelle

Da der Multicast als Teil eines Eventmanagementsystem verwendet werden soll, ist es wünschenswert, daß er eine einfache API-Schnittstelle bereitstellt, mit deren Hilfe er in das Eventmanagementsystem eingebunden werden kann.
Aber auch für jede andere Anwendung des Multicasts ist eine einfache API-Schnittstelle von großem Vorteil.

II.4.3 Parametrisierbarkeit / Konfigurierbarkeit

Multicast-Protokolle lassen sich für die verschiedensten Anwendungszwecke verwenden und müssen damit unterschiedlichste Anforderungen erfüllen. Dabei ist klar, daß ein Multicast, der beispielsweise optimal für weitverteilte Gruppen mit geringer Teilnehmerzahl ist, nicht die beste Lösung für große Gruppen mit geringer Ausdehnung darstellen muß. Ein Algorithmus, der versucht, alle Anforderung zu erfüllen, bietet somit möglicherweise nur mittelmäßige Ergebnisse für die einzelnen Anforderungen. Hier wäre es wünschenswert, wenn sich das Protokoll vom Anwender speziell für den von ihm vorgesehenen Anwendungszweck konfigurieren lassen könnte, um seine speziellen Anforderungen möglichst gut zu erfüllen.

II.4.4 Entfernungsbegrenzung für Events

Es gibt Situationen, in denen es nicht erforderlich ist, daß die gesamte Multicast-Gruppe einen bestimmten Event empfängt, sondern nur der lokale Teil dieser Gruppe oder der Teil, der sich in den umliegenden LANs befindet. Deshalb wäre es von Vorteil, wenn das Multicast-Protokoll eine Möglichkeit der Entfernungsbegrenzung für Events bieten würde.

II.4.5 Maximale Lebensdauer für Events

Es kann Events geben, deren Auslieferung nach einer bestimmten Zeitspanne überflüssig geworden ist, z.B. sind Nachrichten eines Zeitservers nicht interessant, wenn sie nicht mehr aktuell sind. Mit einer Einschränkung der Lebensdauer für Events kann die Menge der Nachrichten, die für eine eventuelle wiederholte Übertragung (für den Fall, daß Nachrichten verlorengingen) zwischengespeichert werden müssen, weil der Empfänger momentan nicht erreicht werden kann, eingeschränkt werden.

III. Vorstellung und Bewertung verschiedener Multicast-Protokolle

In diesem Teil sollen verschiedene, bereits existierende Multicast-Protokolle vorgestellt und anhand verschiedener Kriterien auf die Eignung für ein Eventmanagementsystem geprüft werden.

III.1 Single Spanning Tree Multicast Routing (SST)

In [Deering 90] wird unter anderen dieser Algorithmus vorgestellt, der zur Klasse der MST gehört und somit Multicast durch einen einzigen spannenden Baum realisiert.

III.1.1 Beschreibung des Algorithmus

Zuerst wird ein Spannbaum über alle Router, die die einzelnen LANs untereinander verbinden oder an den Rest des Netzwerks anbinden, aufgebaut. Dies kann geschehen, indem ein verteilter Algorithmus zur Berechnung eines minimalen Spannbaums (z.B. in [Wall 80] oder in [Gallager 83]) auf allen Routern ausgeführt wird. Der aufgebaute Spannbaum dient nun als Backbone für alle Multicast-Gruppen. Wenn ein Router eine Multicast-Nachricht erhält, sendet er sie einfach auf allen Kanten des Spannbaums weiter außer auf der, auf der er die Nachricht erhalten hat. Dies entspricht einem effizienten Broadcast.

Damit nicht alle Multicast-Nachrichten aller Gruppen unnötigerweise über den gesamten Spannbaum verteilt werden, müssen die Router Informationen darüber haben, auf welche von ihnen ausgehenden Kanten des Spannbaums sie Multicast-Nachrichten einer bestimmten Multicast-Gruppe weitergeben müssen und bei welchen Kanten das nicht nötig ist, da kein Teilnehmer der Gruppe sich in diesem Teil des Spannbaumes befindet. Hierzu werden alle Router zu einer Multicast-Gruppe R zusammengefaßt. Jeder Teilnehmer einer Multicast-Gruppe G sendet nun in periodischen Zeitabständen eine besondere Nachricht namens *membership-report* mit Absender G an die Gruppe R. Diese Nachricht wird über den gesamten Spannbaum an alle Router verbreitet. Jeder Router, der eine solche Nachricht empfängt, weiß nun, daß er in Zukunft Multicast-Nachrichten an die Gruppe G auch über die Kante weitersenden muß, über die er diesen *membership-report* erhalten hat. Wenn ein Router über einen bestimmten Timeout-Zeitraum keinen *membership-report* der Gruppe G über eine Kante, die nach ihren Informationen zu Gruppe G gehört, erhalten hat, nimmt er an, daß über diese Kante kein Teilnehmer der Gruppe G mehr zu erreichen ist und sendet ab sofort keine Nachrichten an Gruppe G mehr über diese Kante.
Die für die Verwaltung der Multicast-Gruppen nötigen Informationen kann der Router also speichern, indem er in einer Tabelle für jede Multicast-Gruppe einen Datensatz anlegt, der die Form (*Adresse*, (*Kante*, *Alter*), (*Kante*, *Alter*), ...) hat. Hierbei steht *Adresse* für die Adresse der Multicast-Gruppe, *Kante* für eine von diesem Router ausgehende Kante des Spannbaums, über die zumindest ein Teilnehmer der Multicast-Gruppe erreicht wird, und *Alter* für das Alter des letzten *membership-report* dieser Gruppe, der über diese Kante kam. *Alter* wird gebraucht, um die (*Kante*, *Alter*) - Information nach Ablauf des Timeouts zu entfernen.

Somit sieht der Algorithmus nun wie folgt aus:
Wenn ein Router nun eine Multicast-Nachricht an eine Gruppe G bekommt, sendet er sie auf allen Kanten, die im Datensatz für G aufgeführt sind, weiter, außer auf der Kante, auf der die Nachricht ankam. Hat diese Nachricht eine Multicast-Gruppe M als Absender, wird sie als *membership-report* betrachtet, und die Kante, auf der die Nachricht ankam, wird in einem Datensatz für die Gruppe M zusammen mit der Altersangabe 0 gespeichert.
Periodisch werden die *Alter*-Felder aller (*Kante*, *Alter*) - Einträge inkrementiert. Wird dabei der Timeout-Wert überschritten, wird der entsprechende (*Kante*, *Alter*) - Eintrag gelöscht.

III.1.2 Bewertung des Algorithmus

III.1.2.1 Notwendige Eigenschaften

Als MST-Algorithmus sollte dieser Algorithmus recht gut **skalieren**, da pro Router nur ein Tabelleneintrag für jede Multicast-Gruppe erforderlich ist. Der große Nachteil ist allerdings, daß jeder Router Informationen über JEDE Gruppe speichern muß, auch wenn es sich um einen Router handelt, die sich in großer Entfernung von einer örtlich stark begrenzten Multicast-Gruppe befindet. Bei einem Netz von der Ausdehnung des Internet ist es inakzeptabel, wenn jeder Router Informationen über jede Multicast-Gruppe des Internet speichern muß.

Der Algorithmus arbeitet sehr **verteilt**. Kein Knoten braucht Informationen von anderen Knoten (abgesehen von *membership-reports* und der Information, welche Knoten in der Baumstruktur mit ihm verbunden sind) oder berechnet Ergebnisse für andere Knoten. Auch der initiale Aufbau des Backbone-Spannbaumes kann sehr verteilt geschehen (vgl. [Gallager 83]).

Die **Anonymität** der Multicast-Teilnehmer ist gewährleistet. JoinGroup und LeaveGroup sind problemlos dynamisch möglich.

Mobilität von Sendern wird sehr gut unterstützt: Ein Sender kann sich an einem beliebigen Ort befinden. Wenn er dort eine Nachricht abschickt, findet sie über den Spannbaum den Weg zu allen Empfängern, da jeder Router Informationen über alle Multicast-Gruppen und entsprechend zu benutzende Ausgangsleitungen hat.

Mobilität von Empfängern wird nicht unterstützt: Der Multicast-Router des LANs, das der Empfänger verläßt, muß sich zwar nicht abmelden (LeaveGroup), da der Timeout-Mechanismus dies automatisch erledigt, aber der Multicast-Router des Ziel-LANs muß sich anmelden (JoinGroup), falls nicht schon ein Empfänger der entsprechenden Multicast-Gruppe in diesem LAN existiert. Ein solches JoinGroup entspricht nach [Deering 90] dem mehrfachen Senden einer *membership-report*-Nachricht an ALLE Router, also einem Broadcasts. Das ist recht teuer !

III.1.2.2 Auf höheren Schichten implementierbare, notwendige Eigenschaften

Die **Zuverlässigkeit** entspricht der des unterliegenden Unicast-Protokolls, bei IP handelt es sich also um „Best Effort"-Zuverlässigkeit.
Über die **Ordnung** von Events kann keine Aussage gemacht werden.

III.1.2.3 Wichtige Eigenschaften

Fehlertoleranz ist keine vorgesehen. Fällt eine Kante im Spannbaum oder ein Router aus, sind alle Multicast-Gruppen betroffen, und es muß ein neuer Spannbaum aufgebaut werden. Besteht dieser neue Spannbaum, werden die Tabellen der Router durch die periodischen *membership-reports* im Lauf der Zeit automatisch wieder aufgebaut. Nachrichten, die während dieser Wiederaufbauzeit gesendet werden, kommen höchstwahrscheinlich - wenn überhaupt - nur bei einem Teil der potentiellen Empfänger an.

Die **Auslieferungsverzögerung** ist ziemlich hoch: Zum einen ist dies natürlich durch das MST-Verfahren bedingt, zum anderen wird ein einziger Spannbaum für alle Multicast-Gruppen verwendet, statt für jede Gruppe einen eigenen optimalen Spannbaum zu verwenden, was eine weitere Verschlechterung mit sich bringt.

Zu den **Kosten** ist zu sagen:

- Es wird wenig Rechenzeit gebraucht, da nur ein einziges Mal ein minimaler Spannbaum über alle Router berechnet werden muß, falls keine Fehler auftreten. Allerdings sind in einem größeren Netzwerk solche Ausfälle wahrscheinlich, und bei jedem solchen Ausfall muß ein neuer Spannbaum berechnet werden.
- Wenn der Zeitabstand zwischen zwei *membership-reports* eines Teilnehmers recht großzügig dimensioniert wird, wird auch nur eine geringe Bandbreite verbraucht. Problematisch ist nur, daß solche *membership-reports* an alle Router im Netzwerk versandt werden.
- Auch der Speicherplatz, der in jedem Router gebraucht wird, ist prinzipiell gering. Da jedoch für jede Multicast-Gruppe im gesamten Netzwerk Informationen gespeichert werden müssen, kann dies bei sehr großen Netzwerken durchaus ein kritischer Punkt sein. Allerdings fallen diese durchaus großen Nachteile eher unter den Punkt Skalierbarkeit.

III.1.2.4 Wünschenswerte Eigenschaften

Eine Implementierung als **Basisdienst** ist nicht vorhanden. Es existiert nur der Algorithmus, der entsprechend programmiert werden müßte. Da es noch keine Implementierung gibt, kann auch keine Aussage über die Komplexität der **API-Schnittstelle** gemacht werden.

Es gibt zwei **Parameter**, die man an seine Bedürfnisse anpassen kann:

1.) Der Zeitabstand zwischen zwei *membership-reports* eines Teilnehmers. Bei der Wahl dieses Parameters muß ein Trade-Off zwischen dem Verbrauch von Bandbreite für Verwaltungs-Overhead und dem unnötigen Weitersenden von Nachrichten an nicht mehr vorhandene Empfänger gefunden werden.
2.) Das Timeout-Intervall, das ein Router nach dem Empfang eines *membership-reports* auf den nächsten *membership-report* wartet. Trifft in diesem Zeitraum kein *membership-report* ein, wird angenommen, daß es den entsprechenden Empfänger nicht mehr gibt. Mit diesem Parameter kann eine Aussage über das zugrundeliegende Netzwerk gemacht werden: Gibt es große Schwankungen zwischen den Nachrichtenverzögerungen oder gehen öfters Nachrichten verloren, muß hier ein höherer Wert angenommen werden, womit in Kauf genommen wird, daß Nachrichten unnötig lange an nicht mehr vorhandene Empfänger weitergesandt werden.

Es gibt weder eine **Entfernungsbegrenzung** noch eine **maximale Lebensdauer** für Events.

III.1.3 Kostenabschätzungen für den SST

- **Rechenzeit in den Routern**: gering
- **Speicherplatz in den Routern**:
  Einen Datensatz der Form (*Adresse*, (*Kante*, *Alter*), (*Kante*, *Alter*), ...) in JEDEM Router für JEDE Multicast-Gruppe.

- **CreateGroup** (=erstes JoinGroup), **JoinGroup**:
  Mehrfaches Senden einer *membership-report*-Nachricht an ALLE Router, also Broadcasts.
- **LeaveGroup**: Keine Nachrichten zwischen den Routern
- **SendMulticast**: Effizientes Verteilen der Nachricht über den Backbone-Spannbaum

- **Verwaltungs-Overhead**:
  In bestimmten Zeitabständen sendet JEDER Teilnehmer einer Multicast-Gruppe eine *membership-report*-Nachricht an ALLE Router, also einen Broadcast.

III.1.4 Zusammenfassung für den SST

Der SST bietet einen interessanten Ansatz eines MST-Algorithmus und ist in kleineren Netzwerken sicher äußerst kostengünstig und gut zu verwenden.
Sein großer Nachteil sind jedoch die *membership-reports*, die jeder Empfänger einer Multicast-Gruppe in periodischen Zeitabständen an ALLE Router im Netzwerk senden muß. Dies entspricht einem Broadcast und ist in einem großen Netzwerk untragbar teuer.
Ein weiterer großer Nachteil ist, daß jeder Router Informationen über jede Multicast-Gruppe halten muß. Das dürfte bei großen Netzwerken mit sehr vielen Multicast-Gruppen die Speicherkapazitäten der Router schnell überfordern.
Zum Einsatz in einem Eventmanagementsystem ist dieser Algorithmus sicher nicht zu empfehlen, da er aus oben genannten Gründen eine sehr schlechte Skalierbarkeit bietet.

III.2 Restricted Broadcast Protocol (RB)

In [Belkeir 89] werden zwei Multicast-Algorithmen aus der Klasse der MST-Algorithmen vorgestellt, die insbesondere die Problematik dynamischer Multicast-Gruppen berücksichtigen sollen: Das bei dynamischen Gruppen mit MST-Algorithmen oft auftretende Problem ist, daß die teure Berechnung des Spannbaumes bei jedem hinzukommenden oder sich abmeldenden Teilnehmer vorgenommen werden muß, was hohe Kosten verursacht. Das RB-Protokoll ist der erste dieser beiden Algorithmen und beschreibt einen MST-Algorithmus, der einen minimalen Spannbaum über das gesamte Netzwerk aufbaut und das Weitersenden von Multicast-Nachrichten auf die Kanten des Spannbaumes beschränkt (to restrict = beschränken), die zu Teilnehmern der Multicast-Gruppe führen.

III.2.1 Beschreibung des Algorithmus

Ebenso wie im SST-Ansatz wird zuerst ein Spannbaum über alle Router, die die einzelnen LANs untereinander verbinden oder an den Rest des Netzwerks anbinden, aufgebaut. Dies kann geschehen, indem ein verteilter Algorithmus zur Berechnung eines minimalen Spannbaums (z.B. in [Wall 80] oder in [Gallager 83]) auf allen Routern ausgeführt wird. Der aufgebaute Spannbaum dient nun ebenfalls, wie beim SST, als Backbone für alle Multicast-Gruppen.
Jeder Router unterhält nun für jede ihm bekannte Multicast-Gruppe (nicht, wie bei SST, für jede Gruppe im Netzwerk) einen Datensatz namens *GroupRec*, der ein Feld für den *status*, sowie eine Liste von Nachbarknoten enthält. *status* kann *member* und *non-member* sein und besagt, ob der Router tatsächlich ein Teilnehmer der Multicast-Gruppe ist (d.h. in dem LAN, dessen Multicast-Router dieser Router ist, existiert ein Teilnehmer der Multicast-Gruppe, an den der Router somit ankommende Multicast-Nachrichten senden muß) oder ob er nur auf einem Ast des Spannbaumes zwischen zwei *member*-Knoten liegt und entsprechend Multicast-Nachrichten über den Spannbaum weiterleiten muß. Die Liste von Nachbarknoten gibt an, welche der Nachbarknoten in der Struktur des Spannbaumes zu Teilnehmern dieser Multicast-Gruppe führen.

Will sich ein Router bei einer Multicast-Gruppe anmelden (**JoinGroup**), überprüft er, ob er diese Gruppe bereits kennt und somit einen *GroupRec* für diese Gruppe hat. Ist dies der Fall, muß er nur seinen *status* von *non-member* zu *member* ändern. Sonst legt er einen *GroupRec* für diese Gruppe an mit *status* = *member* und sendet an alle Spannbaum-Nachbarn eine *join-request*-Nachricht. Schließlich speichert er den Nachbarn, von dem er eine positive Bestätigung (*participant*-Nachricht) bekommt, in der Liste von Nachbarknoten für diese Multicast-Gruppe. Erhält er keine positive Bestätigung, bleibt diese Liste leer, da er offensichtlich der einzige Teilnehmer dieser Multicast-Gruppe im Spannbaum ist.
Jeder Knoten, der eine *join-request*-Nachricht erhält, überprüft, ob er diese Gruppe bereits kennt. In diesem Fall nimmt er den Sender der *join-request*-Nachricht in seine Liste von Nachbarknoten für diese Multicast-Gruppe auf und sendet ihm eine *participant*-Nachricht. Sonst sendet er selbst eine *join-request*-Nachricht an alle seine Spannbaum-Nachbarn außer dem Sender der *join-request*-Nachricht. Erhält er dann eine *participant*-Nachricht, legt er einen *GroupRec* für diese Gruppe an mit *status* = *non-member* und trägt in die Liste von Nachbarknoten für diese Multicast-Gruppe jeden Knoten ein, der ihm eine *participant*-Nachricht schickt einschließlich dem Knoten, der ihm die ursprüngliche *join-request*-Nachricht geschickt

hatte. Erhält er keine *participant*-Nachricht oder ist er ein Blattknoten des Spannbaums, so sendet er eine *non-participant*-Nachricht zurück.
Auf diese Weise wird der sich anmeldende Knoten rekursiv über den Spannbaum an bereits existierende Gruppenteilnehmer angebunden und alle dazwischenliegenden Knoten, die nicht zu der Multicast-Gruppe gehören, sind *non-member*.

Will sich ein Router bei einer Multicast-Gruppe abmelden (**LeaveGroup**), überprüft er, ob er sich auf einem Ast zwischen zwei oder mehreren Teilnehmer der Multicast-Gruppe befindet. In diesem Fall wechselt er einfach seinen *status* von *member* zu *non-member*. Sonst, wenn es also nur einen Knoten in seiner Liste von Nachbarknoten für diese Multicast-Gruppe gibt, sendet er diesem Knoten eine *delete-request*-Nachricht und löscht den GroupRec für diese Gruppe.
Jeder Knoten der eine *delete-request*-Nachricht erhält, löscht den Sender dieser Nachricht aus der entsprechenden Liste von Nachbarknoten und überprüft, ob der Auslieferungsbaum für Multicast-Nachrichten dieser Gruppe weiter beschnitten werden kann. Dies ist der Fall, wenn dieser Knoten kein *member* ist und in der Liste von Nachbarknoten nun nur noch ein Knoten ist. Kann weiter beschnitten werden, wird die *delete-request*-Nachricht verschickt und der *GroupRec* gelöscht.

Ein **SendMulticast** ist einfach, wenn der Router die Gruppe bereits kennt. Dann liefert er die Nachricht an lokale Teilnehmer aus und schickt sie an alle Knoten in seiner Nachbarliste für diese Gruppe, die wiederum die Nachricht an ihre lokale Teilnehmer ausliefern und entsprechend ihrer Nachbarliste weitersenden. Kennt der Router die Gruppe nicht, muß er einen Gruppenteilnehmer suchen und ihm die Nachricht schicken.

III.2.2 Modifikation des Algorithmus

Dieser Algorithmus kann sehr teuer werden, wenn es darum geht, einen Teilnehmer einer bestimmten Gruppe zu finden, also bei einem Anmelde- oder Sendevorgang, wenn der Router die Gruppe noch nicht kennt. Hierfür können die Kosten leicht in Größenordnungen eines Multicasts an den gesamten Spannbaum kommen, da der suchende Knoten von seiner Position in alle Richtungen des Spannbaumes Suchnachrichten absetzt, die sich rekursiv fortpflanzen, bis sie evtl. einen Teilnehmer finden. Aus diesem Grund schlägt [Belkeir 89] eine Modifikation zu seinem Algorithmus vor, die diesem Problem entgegenwirken sollen, indem jeder Knoten des Spannbaumes Informationen über alle Multicast-Gruppen in seinen Unterbäumen trägt. Diese Information speichert er in *Directory*-Datenpaaren der Art (Multicast-Gruppe M, Subtree S). Dies führt allerdings dazu, daß die Router von Blattknoten bis hinauf zur Wurzel des Spannbaumes immer mehr Informationen zu speichern haben. Der Wurzelknoten kennt dann alle Multicast-Gruppen und muß entsprechend viel Information speichern.

Der oben beschriebene Algorithmus ist wie folgt zu modifizieren (Diese Modifikation wird in [Belkeir 89] zwar angesprochen, jedoch nicht weiter ausgeführt. Der folgende Teil gibt demnach nur eine effiziente Möglichkeit der Modifikation im Sinne von [Belkeir 89] an):
Will sich ein Router bei einer Multicast-Gruppe **anmelden** und kennt diese Gruppe noch nicht, legt er einen *GroupRec* für diese Gruppe an mit *status* = *member*. Hat er ein *Directory*-Datenpaar für diese Gruppe, befinden sich alle Teilnehmer dieser Gruppe in einem Unterbaum von ihm, und er sendet eine *join-request*-Nachricht an den entsprechenden Kindknoten des Spannbaumes. Sonst sendet er seine *join-request*-Nachricht an seinen Vaterknoten. Schließlich

speichert er den Nachbarn, von dem er eine positive Bestätigung (*participant*-Nachricht) bekommt, in der Liste von Nachbarknoten für diese Multicast-Gruppe. Erhält er keine positive Bestätigung, ist er offensichtlich der einzige Teilnehmer dieser Multicast-Gruppe im Spannbaum und sendet eine *create-directory-entry*-Nachricht Richtung Wurzel des Spannbaumes und jeder Knoten, über den diese Nachricht geht, legt ein *Directory*-Datenpaar für diese Multicast-Gruppe an.
Jeder Knoten, der eine *join-request*-Nachricht erhält und diese Gruppe noch nicht kennt, überprüft, ob er ein *Directory*-Datenpaar für diese Gruppe hat. In diesem Fall sendet er eine *join-request*-Nachricht an den entsprechenden Kindknoten des Spannbaumes. Sonst sendet er seine *join-request*-Nachricht an seinen Vaterknoten. Die Behandlung von *participant*-Nachrichten geschieht wie im oben beschriebenen Algorithmus, allerdings löscht ein Knoten, der einen neuen *GroupRec* für eine Gruppe anlegen muß, das für diese Gruppe evtl. an diesem Knoten vorhandene *Directory*-Datenpaar.

Will sich ein Router bei einer Multicast-Gruppe **abmelden** oder erhält eine *delete-request*-Nachricht, und es ist nötig, den Auslieferungsbaum für diese Gruppe zu beschneiden, überprüft er, ob sein einziger Nachbarknoten für diese Gruppe ein Kindknoten von ihm ist. In diesem Fall legt er ein Directory-Datenpaar für diese Gruppe an. Ist er der letzte Teilnehmer dieser Gruppe sendet er eine *delete-directory-entry*-Nachricht Richtung Wurzel des Spannbaumes.

Das **Versenden** einer Multicast-Nachricht in dem Fall, daß der Router die Gruppe noch nicht kennt, geschieht, indem die Nachricht so lange an den jeweiligen Vaterknoten weitergereicht wird, bis ein Knoten entweder Teilnehmer dieser Gruppe ist und das Weitersenden übernehmen kann oder ein *Directory*-Datenpaar für diese Gruppe hat und die Nachricht dann in den entsprechenden Unterbaum weiterleitet.

III.2.3 Bewertung des Algorithmus

III.2.3.1 Notwendige Eigenschaften

Als MST-Algorithmus sollte dieser Algorithmus recht gut **skalieren**, da pro Router nur ein Tabelleneintrag für jede Multicast-Gruppe erforderlich ist. Allerdings bergen sowohl die ursprüngliche als auch die modifizierte Version Nachteile:
In der ursprünglichen Version bringt das Suchen von Teilnehmer einer bestimmten Multicast-Gruppe (bei Anmelde- oder Sendevorgang an eine dem Router noch unbekannte Gruppe), das so teuer werden kann wie ein Multicast über den gesamten Spannbaum ein eher schlechtes Skalierungsverhalten.
Dagegen erreicht in der modifizierten Version die Datenhaltung in den Routern teilweise große Ausmaße: Von den Blattknoten des Spannbaumes bis zum Wurzelknoten steigt das Datenvolumen, bis im Wurzelknoten für jede Multicast-Gruppe eine Information bereitgehalten werden muß. Hier stellt sich die Frage, ob das in sehr großen Netzwerken mit vielen Multicast-Gruppen nicht bei weitem die Kapazität der Router übersteigt.

Für die **Verteiltheit** des Algorithmus sowie die **Anonymität** der Teilnehmer gilt ähnliches wie beim SST-Algorithmus: Beides wird sehr gut unterstützt.

Mobilität von Sendern und Empfängern wird gut unterstützt, wenn der Multicast-Router des LAN, zu dem migriert wird, die Multicast-Gruppe bereits kennt. Dann kann der migrierende Empfänger einfach angeschlossen werden, bzw. die gesendete Nachricht entsprechend den Informationen des Routers an alle Empfänger verbreitet werden.

Kennt der Multicast-Router des LANs, zu dem migriert wird, die Multicast-Gruppe nicht, wird Mobilität in der ursprünglichen Version des RB nur schlecht unterstützt: Ist dies der Fall, muß sowohl bei der Migration eines Senders als auch bei der Migration eines Empfängers ein Router im Spannbaum gesucht werden, der die Multicast-Gruppe kennt. Das kann so teuer sein wie ein Multicast über den gesamten Spannbaum.

In der modifizierten Version des RB wird Mobilität auch in diesem Fall sehr gut unterstützt: Der nächste Knoten der Multicast-Gruppe wird schnell und kostengünstig gefunden, indem der Spannbaum in Richtung Wurzel durchgegangen wird, bis ein Knoten ein Directory-Datenpaar über diese Multicast-Gruppe hat, das besagt, in welchem seiner Unterbäume sich der nächste Knoten dieser Gruppe befindet. Hat auch die Wurzel keine solche Information, ist der suchende Knoten der einzige Teilnehmer dieser Gruppe. Daher ist eine solche Suche maximal so teuer wie ein Unicast von der doppelten Tiefe des Spannbaumes.

III.2.3.2 Auf höheren Schichten implementierbare, notwendige Eigenschaften

Die **Zuverlässigkeit** entspricht der des unterliegenden Unicast-Protokolls, bei IP handelt es sich also um „Best Effort"-Zuverlässigkeit.
Über die **Ordnung** von Events kann keine Aussage gemacht werden.

III.2.3.3 Wichtige Eigenschaften

Fehlertoleranz ist keine vorgesehen. Fällt eine Kante im Spannbaum oder ein Router aus, sind alle Multicast-Gruppen betroffen, und es muß ein neuer Spannbaum aufgebaut werden. Besteht dieser neue Spannbaum, werden die Tabellen der Router nicht automatisch wieder aufgebaut. Die Router müßten explizit ihre Multicast-Gruppen durch *join-request*-Nachrichten neu anmelden.

Die **Auslieferungsverzögerung** ist aus dem gleichen Grund wie beim SST-Verfahren ziemlich hoch: Es wird ein einziger Spannbaum für alle Multicast-Gruppen verwendet, statt für jede Gruppe einen eigenen optimalen Spannbaum zu verwenden, was eine weitere Verschlechterung des Nachteils von MST-Verfahren mit sich bringt.

Zu den **Kosten** ist zu sagen:

- Es wird wenig Rechenzeit gebraucht, da nur ein einziges Mal ein minimaler Spannbaum über alle Router berechnet werden muß, falls keine Fehler auftreten. Allerdings sind in einem größeren Netzwerk solche Ausfälle wahrscheinlich, und bei jedem solchen Ausfall muß ein neuer Spannbaum berechnet werden.
- Für Bandbreite und Speicherplatz gilt:
  1.) Die Bandbreite, die für Verwaltungs-Overhead gebraucht wird, ist in der ursprünglichen Version des RB unter Umständen ziemlich hoch, während der Verbrauch an Speicherplatz in jedem Router sehr gering ist (Es wird nur ein Datensatz für jede Multicast-Gruppe, die diesen Router direkt betrifft, gespeichert).

2.) Hingegen ist in modifizierten Version des RB der Verbrauch an Bandbreite sehr gering, während der Verbrauch an Speicherplatz in einigen Routern möglicherweise ziemlich hoch ist. Für eine ausführlichere Diskussion dieser Kosten sei auf den Abschnitt über das Skalierungsverhalten sowie die Beschreibung des Algorithmus verwiesen.

Da diese Nachteile im Bereich aber eher unter den Punkt Skalierbarkeit fallen, kann man sagen, daß die Kosten dieses Algorithmus recht gut ausfallen.

III.2.3.4 Wünschenswerte Eigenschaften

Eine Implementierung als **Basisdienst** ist nicht vorhanden. Es existiert nur der Algorithmus, der entsprechend programmiert werden müßte. Da es noch keine Implementierung gibt, kann auch keine Aussage über die Komplexität der **API-Schnittstelle** gemacht werden.

Der Algorithmus selbst bietet keine **Parameter** an. Man kann nur entscheiden, ob man die ursprüngliche Version des RB implementieren will, was einen optimal niedrigen Speicherplatzbedarf in den Routern mit sich bringt, oder ob man sich für die modifizierte Version des RB entscheidet, wodurch man den Verbrauch an Bandbreite und Nachrichtenverkehr minimiert.

Es gibt weder eine **Entfernungsbegrenzung** noch eine **maximale Lebensdauer** für Events.

III.2.4 Kostenabschätzungen für den RB

- **Rechenzeit in den Routern**: gering
- **Speicherplatz in den Routern**:
  1.) Einen Datensatz *GroupRec* der Form (*Adresse*, *status*, *Kante, Kante*, ...) in jedem Router für jede dem Router bekannte Multicast-Gruppe, also für die Gruppen, deren Teilnehmer er ist, sowie die Gruppen, bei denen er auf einem Ast zwischen zwei Teilnehmern liegt.
  2.) Zusätzlich im modifizierten RB: Einen Datensatz *Directory* der Form ((*Adresse*, *Kante*), (*Adresse*, *Kante*), ...) in jedem Router für mit einem (Adresse, Kante)-Eintrag für jede Multicast-Gruppe in einem seiner Unterbäume.

- **CreateGroup** (=erstes JoinGroup):
  Beim ursprünglichen RB ein Broadcast.
  Beim modifizierten RB ein Unicast mit der Entfernung der maximalen Tiefe des Spannbaumes.
- **JoinGroup**:
  Beim ursprünglichen RB zwischen keiner Nachricht und einem Broadcast.
  Beim modifizierten RB zwischen keiner Nachricht und einem Unicast mit der Entfernung der doppelten maximalen Tiefe des Spannbaumes.
- **LeaveGroup**:
- Keine Nachricht oder Zurückschneiden des Auslieferungsbaumes für diese Multicast.
- **SendMulticast**: Effizientes Verteilen der Nachricht über den Backbone-Spannbaum

- **Verwaltungs-Overhead**: Keiner

III.2.5 Zusammenfassung für den RB

Der RB ist dem SST sehr ähnlich, allerdings hat er ein besseres, wenn auch trotzdem nicht sehr gutes, Skalierungsverhalten. In seiner ursprünglichen Version kann es immer wieder zu teuren Multicasts über einen großen Teil des Spannbaumes kommen. In der modifizierten Version hingegen müssen sehr viele Informationen in den Routern in der Nähe der Wurzel des Spannbaumes gespeichert werden. Erfreulich ist, daß diese Version sehr gute Unterstützung für die Mobilität der Multicast-Teilnehmer bietet. Trotzdem kann man den RB wohl nicht als Grundlage für ein Eventmanagementsystem empfehlen.

III.2.6 Nearest Insertion Protocol (NI)

Das NI-Protokoll ist der zweite der beiden Multicast-Algorithmen, die in [Belkeir 89] vorgestellt werden. Dieser Algorithmus soll laut [Belkeir 89] als Unterstützung des RB für lokale Netze eingesetzt werden.

Der NI berechnet für ein LAN bestehend aus Punkt-zu-Punkt-Verbindungen zwischen den Knoten den minimal spannenden Baum zwischen den beteiligten Knoten. Hierzu bestimmt es für jedes LAN einen Koordinator, der dem Multicast-Router für dieses LAN aus der für diese Arbeit angenommenen Netzwerkstruktur entspricht. Alle Koordinatoren sind untereinander über einen minimalen Spannbaum verknüpft. Multicasting zwischen den Koordinatoren geschieht über das RB-Protokoll. Ein Koordinator meldet sich entsprechend dem RB für eine Multicast-Gruppe an, wenn ein Rechner in seinem LAN sich für diese Gruppe anmelden möchte. Ebenso meldet sich der Koordinator von einer Multicast-Gruppe ab, wenn der letzte Teilnehmer dieser Gruppe in seinem LAN sich von dieser Gruppe abmeldet.

Der NI wird innerhalb des LANs ausgeführt und funktioniert wie folgt: Wenn ein Rechner sich für eine Multicast-Gruppe anmelden möchte, sendet er eine *join-request*-Nachricht an seinen Koordinator. Dieser sendet eine *join-request*-Nachricht an alle Teilnehmer dieser Gruppe in diesem LAN, worauf jeder dieser Teilnehmer seine Entfernung zu dem Sender mit dem Anmeldewunsch zurückmeldet. Der Koordinator gibt nun einen Verweis an den Teilnehmer mit der geringsten Entfernung an diesen Sender zurück. Der Sender baut nun über einen *extend-request*, den er in Richtung dieses nächsten Teilnehmers schickt, eine Verbindung zu diesem auf, und alle dazwischenliegenden Knoten legen ein *GroupRec* für diese Gruppe an mit *status* = *non-member*. Das Abmelden von einer Gruppe sowie das Senden von Multicast-Nachrichten funktioniert beim NI analog zum RB.

Der NI ist allerdings für die Netzwerkstruktur, wie sie in dieser Arbeit angenommen wird, unnötig, da ein bereits vorhandener effizienter Multicast- bzw. Broadcast-Dienst auf LAN-Ebene angenommen wird (wie es ja z.B. bei busbasierten lokalen Netzen sowieso vorhanden ist).
Den NI als Multicast-Algorithmus für das gesamte Netzwerk zu verwenden, ist nicht sinnvoll, da er einen zentralen Koordinator-Knoten erfordert, an den bei jedem Anmeldeversuch eines Knotens eine *join-request*-Nachricht gesendet wird, die dieser dann an alle Teilnehmer der entsprechenden Multicast-Gruppe sendet. Diese Lösung skaliert sehr schlecht, ist sehr teuer und bietet eine sehr schlechte Verteiltheit und Fehlertoleranz.
Aus diesen Gründen wird nicht weiter auf den NI eingegangen; er sei nur der Vollständigkeit halber erwähnt.

III.3 Distance Vector Multicast Routing Protocol (DVMRP)

Dieses Protokoll ist ein im Internet weit verbreitetes Multicast-Protokoll und wird auch **IP-Multicast** genannt. Es liegt dem Multicast Backbone im Internet, dem MBone, zugrunde [Schreiber 95]. Dieses Protokoll gehört zur Klasse der SBT und ist ebenfalls in [Deering 90] beschrieben. Es kann nur dort sinnvoll verwendet werden, wo als Unicast-Algorithmus der Distance Vector Routing Algorithm (DVRA) implementiert ist.

In [Deering 90] wird eine Reihe von Protokollen beschrieben, die das distance-vector-routing als Grundlage haben und jeweils eine Modifikation und Verbesserung des vorhergehenden Protokolls darstellen: Reverse Path Flooding (RPF) → Reverse Path Broadcasting (RPB) → Truncated Reverse Path Broadcasting (TRPB) → Reverse Path Multicasting (RPM). Hier soll nur das letzte und somit am meisten verfeinerte Protokoll vorgestellt werden: Das Reverse Path Multicasting Protocol (RPM).

III.3.1 Beschreibung des Algorithmus

Dem DVMRP liegt als Unicast der Distance Vector Routing Algorithm (DVRA) zugrunde. Die Verwendung des DVMRP ist nur sinnvoll, wenn der DVRA für das Unicast-Routing verwendet wird, da sonst der zusätzliche Aufwand zu groß wird.

Beim **DVRA** speichert jeder Router einen Entfernungsvektor (distance vector) mit Einträgen der Form (*Ziel*, *Distanz*, *Nächste-Hop-Adresse*, *Nächste-Hop-Verbindung*, *Alter*). Für jedes Ziel findet sich in diesem Vektor ein Eintrag, wobei Ziel einzelne Rechner, LANs, Subnetzwerke oder ganze Teilnetzwerke sein können. Distanz ist die Entfernung zu Ziel, Nächste-Hop-Adresse ist die Adresse des nächsten Routers auf dem Weg zu Ziel und Nächste-Hop-Verbindung ist die vom Router ausgehende Kante, die zum nächsten Router auf dem Weg zu Ziel führt. Alter ist ein Zähler, der im Laufe der Zeit hochgezählt wird, und wenn er einen bestimmten Wert erreicht hat, wird diese Information verworfen, weil davon ausgegangen wird, daß sie veraltet ist.
In regelmäßigen Abständen sendet jeder Router diesen Entfernungsvektor an jeden Router, mit dem er direkt oder über ein LAN verbunden ist. Aus den Entfernungsvektoren seiner Nachbar-Router kann jeder Router den kürzesten Weg zu jedem Ziel herausfinden und baut entsprechend seinen eigenen Entfernungsvektor neu auf. Wenn nun eine Nachricht für ein Ziel S ankommt schickt der Router sie über die Nächste-Hop-Verbindung des Entfernungsvektoreintrages für S weiter, was dem kürzesten Weg zu S entspricht.
Für eine ausführliche Beschreibung des DVRA verweise ich auf [RFC 1058].

Das hierauf aufbauende **Multicast-Protokoll** funktioniert nun folgendermaßen:
Da es sich um ein SBT-Verfahren handelt, muß für jeden Sender von Multicast-Nachrichten ein eigener minimaler Auslieferungsbaum aufgebaut werden. Solange ein Sender noch keine Multicast-Nachricht geschickt hat, wird keinerlei Information für ihn in den Routern gespeichert. Wenn ein Multicast-Router S eine Multicast-Nachricht N senden will, schickt er sie über alle von ihm ausgehenden Kanten an seine Nachbar-Router. Jeder Router, der diese Nachricht über die Kante erhält, die für ihn den kürzesten Weg zu S bedeutet, sendet die Nachricht N über alle seine Kind-Kanten bzgl. S weiter.

Einschub zu Kind-Kante: Eine Kante ist Kind-Kante (eine Kante kann auch ein LAN sein) eines Routers bzgl. Sender S, wenn er der Router an dieser Kante mit der geringsten Entfernung zu S ist. Haben zwei Router an einer Kante die gleiche Entfernung zu S, ist sie die Kind-Kante des Routers, der die kleinere Netzwerkadresse hat. Um diese Information zu speichern, muß der Entfernungsvektor erweitert werden auf Einträge der Form (*Ziel*, *Distanz*, *Nächste-Hop-Adresse*, *Nächste-Hop-Verbindung*, *Alter*, *Kinder*). *Kinder* ist eine Bitmap mit soviel Bits, wie der Router ausgehende Kanten hat. Das ist typischerweise eine geringe Zahl. Ein Bit in *Kinder* ist auf 1 gesetzt, wenn die zugehörige Kante ein Kind-Kante für den Sender *Ziel* ist.

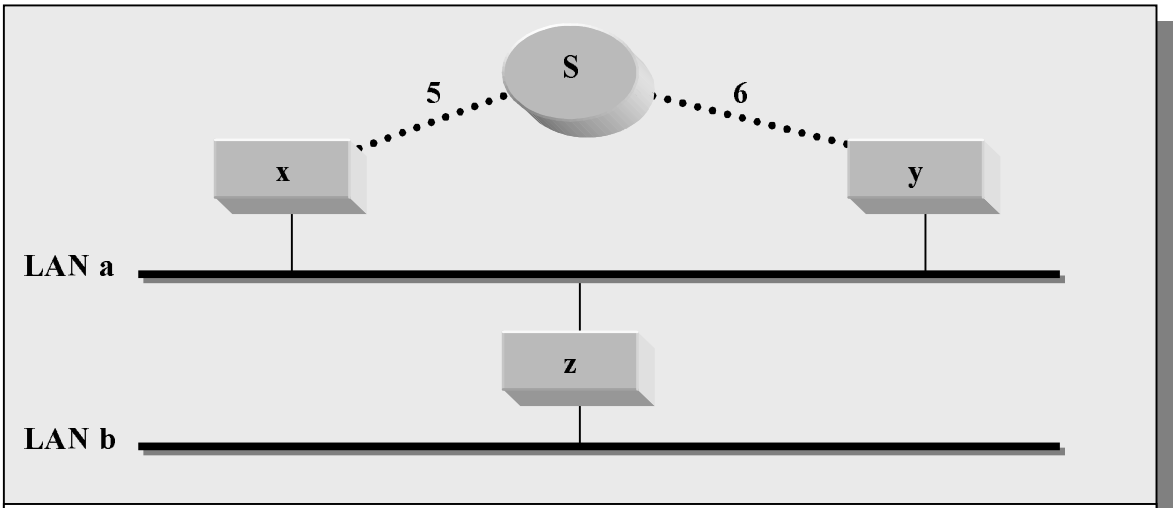

Abbildung III–1: Beispiel zu Kind-Kanten: LAN a ist Kind-Kante für Router x, da seine Entfernung zu S mit 5 kleiner ist als die von Router y. Eine Multicast-Nachricht von S würde also nur von S an LAN a und somit an Router z weitergegeben werden.

Auf diese Weise ein relativ effizienter Broadcast von Sender S realisiert. Um nun den Broadcast auf einen Multicast an die Gruppenteilnehmer der Multicast-Gruppe G, an die S seine Multicast-Nachricht sendet, zu beschränken, senden alle Router, die die Nachricht N bekommen, aber weder Kind-Kanten bzgl. Sender S (sie sind somit Blattknoten bzgl. S) noch Teilnehmer der Gruppe G in ihrem LAN haben, eine *non-membership-report*-Nachricht (NMR) der Form (*Sender* S, *Gruppe* G, *Alter* 0) über die Kante, über die sie die Nachricht N bekamen, zurück.

Ein Router, der eine NMR über eine Kind-Kante bekommt, speichert sie zusammen mit der Information, über welche Kind-Kante diese NMR kam, zählt periodisch den *Alter*-Zähler hoch und sendet keine Multicast-Nachrichten von Gruppe G und Sender S mehr über diese Kind-Kante. Wenn der Alter-Zähler eine Schwelle $T_{MaxAlter}$ erreicht, wird diese NMR verworfen und es werden wieder Multicast-Nachrichten von Sender S an Gruppe G über diese Kind-Kante weitergeleitet.
Bekam ein Router über alle seine Kind-Kanten eine NMR für den gleichen Sender S und die gleiche Gruppe G, sendet er eine NMR über die Kante zurück, über die er die Nachricht N bekam. Als *Alter* setzt er das Maximum der *Alter*-Werte der NMR ein, die er bekam.

Wenn sich bei einem Router ein Teilnehmer für eine Multicast-Gruppe, für die der Router vor weniger als $T_{MaxAlter}$ eine NMR gesendet hat, meldet, oder wenn der Router eine neue Kind-Kante erhält, sendet der Router der NMR eine *cancellation*-Nachricht hinterher, die die NMR rückgängig macht, damit der neue Teilnehmer (bzw. Teilnehmer in der neuen Kind-Kante) nicht bis zum Verfall der NMR warten muß, bis er Nachrichten aus dieser Gruppe bekommt.

Eine *cancellation*-Nachricht muß ebenso wie eine NMR vom Empfänger bestätigt werden, damit sicher angenommen werden kann, daß sie ihr Ziel erreicht hat.

Die NMR sind die einzigen Informationen, die ein Router speichern muß, nachdem ein Sender aktiv geworden ist. Schweigt ein Sender länger als $T_{MaxAlter}$, so wurden alle NMR in den Routern verworfen und der Sender belegt keinen Speicherplatz mehr in den Routern. Allerdings wird eine erneute Nachricht dieses Senders nun wieder an alle Router ausgeliefert, da der Auslieferungsbaum nicht mehr zurückgeschnitten ist.

III.3.2 Bewertung des Algorithmus

III.3.2.1 Notwendige Eigenschaften

Dieser Algorithmus **skaliert** recht gut, wenn die Multicast-Gruppen eine große Ausdehnung über das Netzwerk und viele Teilnehmer haben, also für weitverteilte, dichte Gruppen. Er skaliert ebenso sehr gut für örtlich stark begrenzte Gruppen, da dann die Entfernungsbegrenzung eingesetzt werden kann. Auch für Gruppen mit sehr wenigen Sendern und vielen Empfängern skaliert er gut. Für alle anderen Fälle skaliert er schlecht bis sehr schlecht. Worst Case wäre für ihn eine weitverteilte, dünne Gruppe, in der jeder Teilnehmer immer wieder Nachrichten an die Gruppe sendet.

Der Algorithmus arbeitet sehr **verteilt**. Die Informationen für den Aufbau der einzelnen senderbasierten Spannbäume werden schon in dem zugrunde liegenden Unicast-Protokoll ausgetauscht und können vom Multicast-Protokoll einfach mitverwendet werden.

Die **Anonymität** der Multicast-Teilnehmer ist gewährleistet. JoinGroup und LeaveGroup sind problemlos dynamisch möglich.

Mobilität von Sendern wird nicht unterstützt: Wenn ein Sender das LAN wechselt und von einem anderen LAN, das bisher keine Nachrichten an eine bestimmte Multicast-Gruppe sendete, Multicast-Nachrichten abschickt, muß zwar kein neuer senderbasierter Spannbaum aufgebaut werden, da die Information schon vorhanden ist, aber wenn sich die Gruppe nicht über das ganze Netzwerk erstreckt, entstehen hohe Kosten durch den Broadcast und die anschließende NMR-Flut, die den Spannbaum zurückschneidet.

Mobilität von Empfängern wird besser unterstützt: Der Multicast-Router des LANs, das der Empfänger verläßt, muß sich nicht abmelden (LeaveGroup), nur evtl. bei den nächsten Multicast-Nachrichten NMRs zurückschicken. Der Multicast-Router des Ziel-LANs muß sich auch nicht anmelden (JoinGroup), nur evtl. eine Reihe von *cancellation*-Nachrichten schicken, damit die gewünschten Multicast-Nachrichten wieder an ihn weitergeleitet werden.

III.3.2.2 Auf höheren Schichten implementierbare, notwendige Eigenschaften

Die **Zuverlässigkeit** entspricht der des unterliegenden Unicast-Protokolls, bei IP handelt es sich also um „Best Effort"-Zuverlässigkeit.
Über die **Ordnung** von Events kann keine Aussage gemacht werden.

III.3.2.3 Wichtige Eigenschaften

Fehlertoleranz ist im Algorithmus keine vorgesehen. Fällt eine Kante oder ein Router aus, sind jedoch nur die Multicast-Gruppen betroffen, deren senderbasierter Baum dadurch gestört wird. Allerdings betrifft ein solcher Kanten- oder Routerausfall auch den darunterliegenden DVRA. Dieser behilft sich damit, daß in den nächsten Distanzvektoren die Verschlechterung der Pfade berücksichtigt ist, wodurch für die Nachrichtenauslieferung andere Pfade gewählt werden. Da der DVMRP auf diesen Distanzvektoren aufbaut, wird bei Kanten- oder Routerausfall die Verschlechterung automatisch berücksichtigt, indem manche Router ihre Kind-Kanten an andere Router verlieren, was aber wieder viele NMRs hervorrufen kann. Auf diese Weise wird ein gutes Stück Fehlertoleranz erreicht.

Die **Auslieferungsverzögerung** ist sehr gering, da für jeden Sender ein optimaler Auslieferungsbaum mit geringstmöglicher Verzögerung aufgebaut wird.

Zu den **Kosten** ist zu sagen:

- Es wird sehr wenig Rechenzeit gebraucht, da zum Aufbau der einzelnen senderbasierten Auslieferungsbäume lediglich in Tabellen nachgeschaut wird. Auch das Verschicken und Bearbeiten der NMRs braucht sehr wenig Rechenzeit.

- Für die benötigte Bandbreite gilt:
  1.) Das Senden einer Multicast-Nachricht verbraucht im Prinzip nur sehr wenig Bandbreite. Es ist jedoch teuer, daß die erste Multicast-Nachricht eines Senders, und dann in Zeitabständen von $T_{MaxAlter}$ jeweils mindestens eine weitere Multicast-Nachricht dieses Senders (so er solche Nachrichten sendet), an ALLE Router im Netzwerk gesendet wird.
  2.) Außerdem muß jeder Router in Zeitabständen von $T_{MaxAlter}$ für jede Multicast-Gruppe, die ihn nicht interessiert, und jeden Sender dieser Gruppe, der in diesem Zeitraum gesendet hat, eine NMR senden. Zur Reduktion der benötigten Bandbreite wäre also sehr zu empfehlen, $T_{MaxAlter}$ sehr groß zu wählen, zumal der möglicherweise nachteilige Effekt, daß neue Teilnehmer von Gruppen auf das Verstreichen von $T_{MaxAlter}$ warten müssen, bis sie Nachrichten aus dieser Gruppe empfangen, durch die bestätigten *cancellation*-Nachrichten aufgehoben wird. Somit können die Kosten bei sehr großem $T_{MaxAlter}$ vergleichsweise niedrig gehalten werden.
  3.) Als weiterer Punkt zum Verbrauch der Bandbreite fällt auf, daß Wert darauf gelegt wird, daß jede Kante mit jeder Multicast-Nachricht bedient wird. Dabei wäre es doch nur nötig, daß jeder Knoten (Router) mit jeder Multicast-Nachricht bedient wird. So kann es vorkommen, daß manche Knoten eine Multicast-Nachricht unnötigerweise mehrfach erhalten, z.B. wenn in Abbildung III–1 statt LAN a eine direkte Verbindung zwischen Knoten x und y vorhanden wäre, würde Knoten y die Nachricht sowohl direkt von Sender S als auch über Knoten x erhalten.

- Der Speicherplatz, der in jedem Router gebraucht wird, ist prinzipiell gering: Für die Broadcast-Funktionalität muß jeder Eintrag des vorhandenen Entfernungsvektors um eine kleine Bitmap erweitert werden. Das fällt kaum ins Gewicht. Problematischer könnten allerdings die NMR-Einträge werden, die der Router für das Zurückschneiden der senderbasierten Broadcast-Bäume auf senderbasierte Multicast-Bäume speichern muß. Da die maximale Anzahl der NMR-Einträge der Anzahl der in $T_{MaxAlter}$ erhaltenen Multicast-Nachrichten multipliziert mit der maximalen Anzahl von Kind-Kanten entspricht, wäre es zur Reduktion des benötigten Speicherplatzes in den Routern sinnvoll, ein kleines $T_{MaxAlter}$ zu wählen, damit die NMR-Einträge schneller wieder gelöscht werden können. Dies ist vor allem für Sender, die sehr selten oder gar nicht mehr senden von großem Vorteil. Leider kommt man damit dem Wunsch nach Reduktion der benötigten Bandbreite ins Gehege.

III.3.2.4 Wünschenswerte Eigenschaften

Eine Implementierung als **Basisdienst** ist im Internet vorhanden und mit IGMP existiert auch eine sehr gute **API-Schnittstelle** zum DVMRP.

Es gibt einen **Parameter**, den man an seine Bedürfnisse anpassen kann, der allerdings in der Implementierung als Basisdienst bereits fest voreingestellt und für die Anwendung nicht erreichbar ist: $T_{MaxAlter}$
Wählt man $T_{MaxAlter}$ zu klein, resultiert ein sehr hoher Verbrauch an Bandbreite für das häufigere (unnötige) Senden der Multicast-Nachrichten an ALLE Router, sowie das Zurücksenden von vielen NMRs. Wählt man $T_{MaxAlter}$ zu groß, resultiert ein hoher Verbrauch von Speicherplatz in den Routern, da NMR-Einträge seltener verfallen und gelöscht werden können. (siehe auch Diskussion bei den Kosten des Algorithmus)
Allgemein kann man sagen, daß für weitverteilte, dichte Multicast-Gruppen und viele, selten aktive Sender $T_{MaxAlter}$ eher gering angesetzt werden sollte, während das Gegenteil für dünne Gruppen und wenige, hochaktive Sender gilt.

Es existiert eine **Entfernungsbegrenzung** für Multicast-Nachrichten, aber nicht die Möglichkeit, eine **maximale Lebensdauer** festzulegen.

III.3.3 Kostenabschätzungen für den DVMRP

- **Rechenzeit in den Routern**: sehr gering
- **Speicherplatz in den Routern**:
  1.) Eine Erweiterung des vorhandenen Entfernungsvektors um die Bitmap *Kinder* auf Einträge der Form (*Ziel*, *Distanz*, *Nächste-Hop-Adresse*, *Nächste-Hop-Verbindung*, *Alter*, *Kinder*). *Kinder* ist eine Bitmap mit soviel Bits, wie der Router ausgehende Kanten hat. Das ist typischerweise eine geringe Zahl.
  2.) Einen NMR-Eintrag der Form (*Sender*, *Gruppe*, *Alter*, *Kante*) für jeden benachbarten Router an einer Kind-Kante, der sich über eine unerwünschte Multicast-Nachricht beschwert hat. Wieviele NMR-Einträge muß ein Router maximal speichern ? Die Anzahl der in $T_{MaxAlter}$ erhaltenen Multicast-Nachrichten multipliziert mit der maximalen Anzahl von Kind-Kanten.

- **CreateGroup** (=erstes JoinGroup): Keine Nachrichten zwischen Routern
- **JoinGroup**: Evtl. eine Reihe von *cancellation*-Nachrichten
- **LeaveGroup**: Keine Nachrichten zwischen den Routern

- **SendMulticast**:
  Die erste Multicast-Nachricht eines Senders, und dann in Zeitabständen von $T_{MaxAlter}$ jeweils mindestens eine weitere Multicast-Nachricht dieses Senders (so er solche Nachrichten sendet), wird an ALLE Router im Netzwerk gesendet. Dazwischenliegende Sendungen werden entsprechend den NMR-Einträgen nur an Teile des Netzwerks gesendet.

- **Verwaltungs-Overhead**:
  Jeder Router muß in Zeitabständen von $T_{MaxAlter}$ für jede Multicast-Gruppe, die ihn nicht interessiert, und jeden Sender dieser Gruppe, der in diesem Zeitraum gesendet hat, eine NMR senden.

III.3.4 Zusammenfassung für den DVMRP

Der DVMRP ist sicherlich ein interessanter Multicast für die Verwendung im Eventmanagementsystem. Vor allem sein Vorhandensein als Basisdienst und seine sehr schnelle Auslieferung, seine Fehlertoleranz, sowie seine vergleichsweise gute Skalierbarkeit geben ihm zusammen mit der Entfernungskontrolle Vorteile gegenüber den bisher vorgestellten Multicasts.

Allerdings wäre es bei der Verwendung in einem Eventmanagementsystem mit mobilen Teilnehmern sicher ein sehr teurer Ansatz, jeden sendewilligen Teilnehmer als Sender im Sinne des DVMRP zuzulassen, da solche Teilnehmer sich ja bewegen können und es außerdem potentiell sehr viele Teilnehmer geben kann. Man könnte sich jedoch überlegen, an jedem der Orte, an denen sich Teilnehmer aufhalten können, eine „Sendestation“ anzubringen, über die die Teilnehmer ihre Events senden können. Hiermit wären zumindest die Probleme Mobilität und die große Zahl der Sender reduziert.

III.4 Link-State Multicast Routing (MOSPF)

Dieses Protokoll ist ein im Internet übliches Multicast-Protokoll. Es wird in den Teilen des Internet angewendet, in denen als Unicast-Algorithmus das OSPF-Verfahren (Open Shortest Path First) verwendet wird. Das OSPF-Verfahren ist das bei weitem bekannteste Link-State-Routing-Verfahren und wird in großen Teilen des Internet verwendet. Darum soll hier als zugehöriges Multicast-Verfahren das MOSPF-Verfahren (Multicast Extensions for OSPF) beschrieben werden. Dieses Protokoll gehört ebenfalls zur Klasse der SBT und ist in [Deering 90] und [Schreiber 95] beschrieben.

III.4.1 Beschreibung des Algorithmus

Dem MOSPF liegt als Unicast das OSPF-Verfahren (Open Shortest Path First) zugrunde. Die Verwendung des MOSPF ist nur sinnvoll, wenn das OSPF-Verfahren für das Unicast-Routing verwendet wird, da sonst der zusätzliche Aufwand zu groß wird.

Beim **Link-State-Routing-Verfahren** beobachtet jeder Router den Zustand der Netzwerkverbindungen, die direkt von ihm ausgehen (z.B. deren Funktionsfähigkeit und Belastung). Ändert sich der Zustand einer solchen Verbindung, sendet der Router einen Broadcast an alle anderen Router, in denen er ihnen eine aktuelle Beschreibung des Zustands seiner Verbindungen mitteilt. Dieser spezielle Broadcast wird mit hoher Priorität, schnell und zuverlässig ausgeführt. Jeder Router speichert die jeweils aktuelle Link-State-Nachricht aller anderen Router und kann so mit dem Dijkstra-Algorithmus einen minimal spannenden Baum, dessen Wurzel er selbst ist, errechnen und auf diese Weise entscheiden, welche Verbindung den kürzesten Weg zu einem beliebigen Ziel darstellt. Nachrichten, die er erhält, kann er anhand dieser Informationen über die günstigste Kante weiterschicken.
Für eine ausführliche Beschreibung des OSPF-Verfahrens verweise ich auf [RFC 1131].

Das hierauf aufbauende **Multicast-Protokoll** funktioniert nun folgendermaßen:
Da es sich um ein SBT-Verfahren handelt, muß für jeden Sender von Multicast-Nachrichten ein eigener minimaler Auslieferungsbaum aufgebaut werden. Hierzu erweitert jeder Multicast-Router seine Link-State-Informationen um die bei an einer Verbindung (beispielsweise einem LAN) präsenten Multicast-Gruppen. Ändern sich diese Informationen, muß er - wie bei Änderungen der Verbindungszustände - aktuelle Link-State-Nachrichten broadcasten.
Weil sich die Information über vorhandene Multicast-Gruppen aber unter Umständen schnell und sehr häufig ändern kann, und Broadcasts ziemlich teuer sind, sollte verhindert werden, daß bei jeder Änderung ein Broadcast der aktuellen Link-State-Nachricht vorgenommen wird. Statt dessen sollte man nach dem Auftreten einer Änderung ein Zeitintervall abwarten, z.B. fünf Sekunden, und dann, wenn nicht ohnehin schon zwischendurch eine Link-State-Nachricht gesandt werden mußte, alle in diesem Zeitintervall eingetretenen Änderungen der Multicast-Gruppen akkumuliert in einer Link-State-Nachricht versenden.

Anhand der zusätzlichen Informationen über vorhandene Multicast-Gruppen kann jeder Router für jeden möglichen Sender mit Hilfe des Algorithmus von Dijkstra einen minimalen Multicast-Auslieferungsbaum berechnen. Dies wird allerdings erst bei Bedarf getan: Wenn Sender-Router S eine Multicast-Nachricht für Gruppe G senden will, berechnet er anhand der Informationen über die Verteilung der Gruppenteilnehmer einen minimalen Auslieferungsbaum und sendet die

Multicast-Nachricht über die entsprechenden, von ihm ausgehenden, Verbindungen. Jeder Router, der diese Multicast-Nachricht erhält, berechnet ebenso den minimalen Spannbaum für Sender S und Gruppe G und gibt die Nachricht dementsprechend weiter.

Weil das Berechnen von minimalen Spannbäumen nach Dijkstra relativ rechenaufwendig ist, wird ein Cache eingeführt, der jeweils die Ergebnisse der letzten Spannbaum-Berechnungen speichert. So muß für sehr aktive Sender nicht ständig die Spannbaum-Berechnung durchgeführt werden. Ein solcher Cache-Eintrag hat die Form (*Sender*, *Gruppe*, *min-hops*). *Sender* ist der Sender der Multicast-Nachricht, *Gruppe* ist die Multicast-Gruppe, an die gesendet wird, und *min-hops* ist ein Array, in dem für jede von diesem Router ausgehende Verbindung die minimale Entfernung des nächsten Teilnehmers dieser Multicast-Gruppe entlang dem minimalen Spannbaum gespeichert ist. Ist über eine Verbindung kein Teilnehmer zu erreichen, findet sich hier ein spezieller Wert, der für unendlich steht.
Der gesamte Cache muß geleert werden, wenn der Router eine Link-State-Nachricht erhält und somit annehmen muß, daß sich die Netzwerktopologie geändert hat.

Somit sieht der Algorithmus nun wie folgt aus:
Jeder Router der eine Multicast-Nachricht erhält, sucht in seinem Cache, ob er einen Eintrag für den betreffenden Sender und die betreffende Gruppe hat. Ist dies der Fall, sendet er die Nachricht über alle von ihm ausgehenden Verbindungen weiter, bei denen die minimale Entfernung des nächsten Teilnehmers kleiner oder gleich der Restentfernung ist, die für diese Multicast-Nachricht noch erlaubt ist.
Hat der Router keinen entsprechenden Cache-Eintrag, muß er den minimalen Spannbaum für den Sender und die Gruppe nach Dijkstra berechnen, speichert die Ergebnisse in einem Cache-Eintrag und sendet, wie soeben beschrieben, die Nachricht weiter.

III.4.2 Bewertung des Algorithmus

Bei der Bewertung des Algorithmus soll das teure Broadcasting, das zur Durchführung des zugrundeliegenden Unicast-Algorithmus immer wieder geschieht, außer Acht gelassen werden, da dies ja auch ohne Einführung dieses Multicast-Algorithmus getan wird. Dieser Multicast-Algorithmus nutzt also nur die vorliegenden Informationen.

III.4.2.1 Notwendige Eigenschaften

Dieser Algorithmus **skaliert** sehr gut (Ob das auch für den darunterliegenden Unicast gilt, soll hier nicht betrachtet werden...), einzig die zusätzlichen Link-State-Nachrichten für geänderte Multicast-Gruppenmitgliedschaften können für große Netzwerke mit vielen Multicast-Gruppen und häufigen Änderungen der Teilnehmerschaft das Skalierungsverhalten beeinträchtigen. Dieses Problem wird aber durch über einen Zeitraum akkumulierte Änderungsmeldungen in einer einzigen Link-State-Nachricht (s.o.) entschärft.

Der Algorithmus arbeitet sehr **verteilt**. Die Informationen für den Aufbau der einzelnen senderbasierten Spannbäume werden schon in dem zugrunde liegenden Unicast-Protokoll ausgetauscht und können vom Multicast-Protokoll einfach mitverwendet werden. In jedem Router wird ein eigener Auslieferungsbaum für jede Multicast-Nachricht errechnet.

Die **Anonymität** der Multicast-Teilnehmer ist gewährleistet. JoinGroup und LeaveGroup sind problemlos dynamisch möglich.

Mobilität von Sendern wird gut unterstützt: Wenn ein Sender aus dem LAN, zu dem er gewechselt hat, eine Multicast-Nachricht sendet, ist der einzige Nachteil gegenüber dem Senden aus dem vorigen LAN, das kein Router einen passenden Cache-Eintrag hat, und somit jeder Router einen minimalen Spannbaum errechnen muß. Aber das passiert für unbekannte oder selten sendende Sender sowieso.

Mobilität von Empfängern wird schlechter unterstützt: Der Multicast-Router des LANs, das der Empfänger verläßt, muß sich abmelden (LeaveGroup) und der Multicast-Router des Ziel-LANs muß sich anmelden (JoinGroup). Das bringt jeweils den Broadcast einer Link-State-Nachricht mit sich. Dieses Problem wird aber durch über einen Zeitraum akkumulierte Änderungsmeldungen in einer einzigen Link-State-Nachricht (s.o.) entschärft.

III.4.2.2 Auf höheren Schichten implementierbare, notwendige Eigenschaften

Die **Zuverlässigkeit** entspricht der des unterliegenden Unicast-Protokolls, bei IP handelt es sich also um „Best Effort"-Zuverlässigkeit.
Über die **Ordnung** von Events kann keine Aussage gemacht werden.

III.4.2.3 Wichtige Eigenschaften

Die **Fehlertoleranz** dieses Algorithmus ist sehr gut. Fällt eine Kante oder ein Router aus, wird dies sofort in aktuellen Link-State-Nachrichten vermerkt. Der Cache für die Ergebnisse von früheren Auslieferungsbaum-Berechnungen wird gelöscht, und für alle weiterzusendenden Multicast-Nachrichten wird anhand der aktuellen Link-State-Nachrichten ein neuer, fehlerfreier Auslieferungsbaum berechnet.

Die **Auslieferungsverzögerung** ist im Prinzip sehr gering, da für jeden Sender ein optimaler Auslieferungsbaum mit geringstmöglicher Verzögerung aufgebaut wird. Allerdings können die Verzögerungen, die durch die Berechnungen von Auslieferungsbäumen in jedem Router entstehen, die Auslieferungszeit stark beeinflussen. In [Deering 90] wird geschätzt, daß diese Verzögerungen pro Router in einem typischen Internetzwerk unter fünf Millisekunden liegen.

Zu den **Kosten** ist zu sagen:

- Es wird relativ viel Rechenzeit gebraucht, da in jedem Router für jede Multicast-Nachricht, für die kein Cache-Eintrag existiert, ein minimaler Spannbaum errechnet werden muß.
- Der Algorithmus braucht sehr wenig Bandbreite (was man von dem darunterliegenden Unicast nicht behaupten kann), nur die zusätzlichen Link-State-Nachrichten, die bei Änderungen der in einem LAN vorhandenen Gruppen versendet werden, sind teuer. Aber diese Kosten können durch über einen Zeitraum akkumulierte Änderungsmeldungen in einer einzigen Link-State-Nachricht (s.o.) stark reduziert werden.
- Der Speicherplatz, der in jedem Router gebraucht wird, ist prinzipiell gering: Die Erweiterung der vorhandenen Link-State-Einträge dürfte kaum ins Gewicht fallen und die Größe des Cache für die Ergebnisse aus früheren Spannbaum-Berechnungen kann den Bedürfnissen entsprechend (im Trade-Off mit dem Rechenzeitverbrauch) gewählt werden.

III.4.2.4 Wünschenswerte Eigenschaften

Eine Implementierung als **Basisdienst** ist im Internet vorhanden und mit IGMP existiert auch eine sehr gute **API-Schnittstelle** zum DVMRP.

Es gibt zwei **Parameter**, den man an seine Bedürfnisse anpassen kann, die allerdings in der Implementierung als Basisdienst bereits fest voreingestellt und für die Anwendung nicht erreichbar sind:

1.) Die Größe des Caches in jedem Router, in dem frühere Spannbaum-Berechnungen gespeichert werden. Wird dieser Cache sehr groß angelegt, kostet das viel Speicherplatz im Router, dafür wird aber Rechenzeit gespart, da häufiger Spannbaum-Informationen im Cache gefunden werden und der Router seltener minimale Spannbäume berechnen muß, was eine teure Operation ist. Wird der Cache klein angelegt, spart das Speicherplatz auf Kosten der benötigten Rechenzeit und der damit verbundenen Verzögerung von Multicast-Nachrichten.

2.) Der Zeitraum, in dem Änderungen der in einem LAN vorhandenen Gruppen gesammelt werden, bevor sie als akkumulierte Link-State-Nachricht per Broadcast an alle Router gesendet werden. Wählt man dieses Intervall sehr gering, wird das Netzwerk durch häufige Broadcasts stärker belastet. Wählt man ihn sehr groß, wird die Netzwerkbelastung entsprechend niedriger, zumal die Wahrscheinlichkeit steigt, daß keine zusätzlichen Link-State-Nachrichten wegen solcher Änderungen gesendet werden müssen, weil diese inzwischen durch eine Link-State-Nachricht, die von der Unicast-Ebene initiiert wurde, verbreitet wurden. Nachteilig ist hier nur, daß evtl. Multicast-Nachrichten an LANs geschickt werden, in denen es schon länger keinen Teilnehmer dieser Gruppe mehr gibt, während neue Teilnehmer in anderen LANs länger warten müssen, bis sie erste Nachrichten von dieser Gruppe erhalten.

Es existiert eine **Entfernungsbegrenzung** für Multicast-Nachrichten, aber nicht die Möglichkeit, eine **maximale Lebensdauer** festzulegen.

III.4.3 Kostenabschätzungen für den MOSPF

- **Rechenzeit in den Routern**: In jedem Router muß für jede Multicast-Nachricht, die ihn erreicht, ein minimaler Spannbaum nach Dijkstra berechnet werden, falls es keinen Cache-Eintrag für den Sender und die Gruppe der Multicast-Nachricht gibt. Dies ist eine relativ teure Operation. Die Häufigkeit ihres Auftretens hängt von der Größe des Caches und der Anzahl der Sender und der Gruppen ab.
- **Speicherplatz in den Routern**:
  1.) Eine Erweiterung der vorhandenen Link-State-Einträge um die Information, welche Multicast-Gruppen an dieser Verbindung vertreten sind. Das sollte nicht allzu schwer ins Gewicht fallen.
  2.) Ein Cache, in dem die Ergebnisse der letzten Spannbaum-Berechnungen gespeichert sind, damit für sehr aktive Sender nicht ständig die Spannbaum-Berechnung durchgeführt werden muß. Ein Cache-Eintrag hat die Form (*Sender*, *Gruppe*, *min-hops*). *Sender* ist der Sender der Multicast-Nachricht, *Gruppe* ist die Multicast-Gruppe, an die gesendet wird, und *min-hops* ist ein Array, in dem für jede von diesem Router ausgehende Verbindung die minimale Entfernung des nächsten Teilnehmers dieser Multicast-Gruppe entlang dem minimalen Spannbaum gespeichert ist.

- **CreateGroup** (=erstes JoinGroup), **JoinGroup, LeaveGroup**:
  Broadcast einer zusätzlichen Link-State-Nachricht an ALLE Router. Dies kann aber mit akkumulierten Änderungsmeldungen für ganze Zeitintervalle, z.B. fünf Sekunden, zusammengefaßt werden (s.o.), wodurch die Anzahl der teuren Broadcasts stark reduziert wird.

- **SendMulticast**:
  Optimal im Sinne des Verbrauchs an Bandbreite. Allerdings muß allen Routern, die auf dem senderbasierten Auslieferungsbaum liegen, ein minimaler Spannbaum nach Dijkstra berechnet werden, falls es keinen Cache-Eintrag für den Sender und die Gruppe der Multicast-Nachricht gibt. Dies ist eine relativ teure Operation.

- **Verwaltungs-Overhead**: keiner

III.4.4 Zusammenfassung für den MOSPF

Der MOSPF ist ebenso wie der DVMRP durch sein Vorhandensein als Basisdienst und seine schnelle Auslieferung, seine sehr gute Fehlertoleranz, sowie seine gute Skalierbarkeit sicherlich ein interessanter Multicast für die Verwendung im Eventmanagementsystem. Sein Problem ist vor allem, daß das häufige Broadcasting des darunterliegenden Unicasts eine Skalierbarkeit über größere Netzwerke unmöglich macht.

Würde man den MOSPF für ein Eventmanagementsystem mit mobilen Teilnehmern verwenden, wäre es allerdings teuer, jeden Teilnehmer als Empfänger im Sinne des DVMRP zuzulassen, da solche Teilnehmer sich ja bewegen können und die Mobilität von Empfängern nicht unbedingt kostengünstig beim MOSPF ist. Man könnte sich jedoch - analog zum DVMRP - überlegen, an jedem der Orte, an denen sich Teilnehmer aufhalten können, eine „Empfängerstation“ anzubringen, die den Teilnehmern, die sich in ihrer Nähe befinden, ihre Events weitergeben könnte.

III.5 Core-based Trees (CBT)

In [Ballardie 93] wird ein Multicast-Algorithmus beschrieben, der eine bessere Alternative zum IP-Multicast bieten soll. Verbesserungen sollen vor allem in dem Verbrauch an Bandbreite, in der Einfachheit, der Effizienz, sowie in der besseren Skalierbarkeit und der Unabhängigkeit vom darunterliegenden Unicast liegen. Dieser Multicast gehört zur Klasse der MST-Algorithmen. [Schreiber 95] erwähnt CBT als Alternative zum DVMRP und MOSPF.

III.5.1 Beschreibung des Algorithmus

Der zentrale Punkt in der CBT-Architektur ist ein speziell ausgezeichneter Router, der Core-Router. Für jede Multicast-Gruppe existiert ein solcher Core-Router. Die Adresse des Core-Routers erfährt man, indem man an den Domain Name Server (DNS) eine Anfrage mit dem Gruppennamen (z.B. cbttalk.cs.ucl.ac.uk) sendet. Der DNS gibt auf eine solche Anfrage die Adresse des Core-Routers sowie die ID der Multicast-Gruppe zurück.

Die Baumstruktur einer Multicast-Gruppe wird aufgebaut, indem jeder Router, der zum Auslieferungsbaum gehört, seinen *parent*-Knoten (das ist der von ihm aus nächste Knoten in Richtung Core-Router) sowie seine *child*-Knoten (seine Kindknoten in Bezug auf den Auslieferungsbaum) speichert.

Ein **JoinGroup** wird durchgeführt, indem eine *join-request*-Nachricht zum nächsten Router auf dem Weg zum Core-Router gesendet wird. Dies kann z.B. durch eine Unicast-Nachricht an den Core-Router mit einer Entfernungsbegrenzung von einem Hop geschehen. Die *join-request*-Nachricht wird auf diese Weise immer einen Schritt weiter auf dem Weg zum Core-Router geschickt, bis sie auf einen Router trifft, der Teil des Auslieferungsbaumes für diese Gruppe oder der Core-Router selbst ist. Wenn dieser Router entscheidet, den neuen Teilnehmer zuzulassen, aktualisiert er seine Baumstrukturinformationen und schickt er auf gleiche Weise schrittweise eine *join-ack*-Nachricht zum neuen Teilnehmer. Alle Router, an denen die *join-ack*-Nachricht vorbeikommt, merken sich, daß sie Non-Core-Router für die entsprechende Gruppe sind und speichern *parent*-Knoten und *child*-Knoten. Non-Core-Router sind Router auf dem direkten Weg zwischen dem Core-Router und einem Teilnehmer der Gruppe. Erst durch die *join-ack*-Nachricht wird also der neue Zweig aufgebaut. Wenn ein Router eine *join-request*-Nachricht gesendet hat und auf eine *join-ack*-Nachricht wartet, währenddessen aber selbst *join-request*-Nachrichten bekommt, muß er diese zwischenspeichern, bis er weiß, ob er zum Auslieferungsbaum gehört oder nicht.

Ein **LeaveGroup** geschieht, wenn ein Multicast-Router keinen Teilnehmer der Gruppe in seinem LAN mehr hat, und er in bezug auf diese Gruppe ein Blattknoten im Auslieferungsbaum ist, also keine *child*-Knoten mehr hat. Dann sendet eine *quit-request*-Nachricht an seinen *parent*-Knoten, worauf dieser im eine *quit-ack*-Nachricht zurücksendet, ihn aus seiner *child*-Knoten-Liste löscht und überprüft, ob er selbst ein LeaveGroup durchführen kann.

Ein **SendMulticast** wird ausgeführt, indem die Multicast-Nachricht zum parent-Knoten und allen child-Knoten gesendet wird. Jeder Router, der eine Multicast-Nachricht erhält, sendet sie an parent-Knoten und child-Knoten außer dem Knoten, von dem er sie bekam. Auf diese Weise wird die Multicast-Nachricht über den gesamten Baum verbreitet.

Fehlertoleranz:
Knoten eines Auslieferungsbaumes testen in regelmäßigen Zeitabständen, ob ihre *parent*-Knoten noch erreichbar sind. Dies geschieht durch periodisch gesendete Echonachrichten an diese benachbarten Knoten. Stellt ein Knoten fest, daß sein *parent*-Knoten nicht mehr erreichbar ist, versucht er durch eine *join-request*-Nachricht in Richtung Core-Router, den Auslieferungsbaum über einen anderen *parent*-Knoten wiederherzustellen. Ist dies nicht möglich, wird eine *flush-tree*-Nachricht an den unter ihm liegenden Teil des Baumes gesendet, worauf jeder Knoten selbst versucht, eine Verbindung zum Core-Router herzustellen.

Um dem großen Nachteil des Core-Routers als Single-Point-Of-Failure entgegenzuwirken, wird das Protokoll erweitert: Statt nur einem Core-Router wird eine Liste von Routern als Core-List verwaltet. Diese Router sind nach einer Priorität geordnet und der Listenerste ist der Core-Router. Fällt dieser aus, und kann er auch nicht über andere Wege erreicht werden, wird versucht, einen Spannbaum mit dem nächsten Router in der Core-List als Core-Router aufzubauen.

III.5.2 Bewertung des Algorithmus

III.5.2.1 Notwendige Eigenschaften

Als MST-Algorithmus **skaliert** dieser Algorithmus recht gut, da pro Router nur ein Tabelleneintrag für jede Multicast-Gruppe in deren Auslieferungsbaum er auftritt, erforderlich ist. Es finden auch keine teuren Broadcasts oder anderer teurer Verwaltungs-Overhead statt.

Der Algorithmus arbeitet nicht sehr **verteilt**. Er braucht eine zentrale Instanz, den Core-Router, der als Anlaufpunkt für alle Operationen benötigt wird. Das führt einen Single-Point-Of-Failure ein, der mühsam mit Hilfskonstrukten, wie der Core-List, beseitigt werden soll.

Die **Anonymität** der Multicast-Teilnehmer ist gewährleistet. JoinGroup und LeaveGroup sind problemlos dynamisch möglich.

Mobilität von Sendern wird sehr gut unterstützt: Wenn ein Sender aus dem LAN, zu dem er gewechselt hat, eine Multicast-Nachricht sendet, schickt er sie in Richtung Core-Router, und ab der Stelle, an der sie auf den Auslieferungsbaum trifft, wird sie effizient über diesen Baum verteilt.

Mobilität von Empfängern wird auch gut unterstützt: Der Multicast-Router des LANs, das der Empfänger verläßt, muß sich abmelden (LeaveGroup), was im schlechtesten Fall ein paar *quit-request*-Nachrichten nach sich zieht, und der Multicast-Router des Ziel-LANs muß sich anmelden (JoinGroup), was im schlimmsten Fall einen *join-request*-Unicast bis zum Core-Router sowie einen *join-ack*-Unicast zurück zum Multicast-Router des Ziel-LANs mit sich bringt. Das ist nicht sehr teuer.

III.5.2.2 Auf höheren Schichten implementierbare, notwendige Eigenschaften

Die **Zuverlässigkeit** entspricht der des unterliegenden Unicast-Protokolls, bei IP handelt es sich also um „Best Effort“-Zuverlässigkeit.

Über die **Ordnung** von Events kann keine Aussage gemacht werden. Allerdings könnte man den Algorithmus sehr leicht zu einem Algorithmus, der Totalordnung bietet, umwandeln, indem alle zu sendenden Nachrichten direkt an den Core-Router gesendet und von dort über den Auslieferungsbaum verbreitet werden (unter der Annahme, daß Nachrichten, die bei einem Router ankommen, in FIFO-Ordnung von ihm weitergegeben werden und sich so nicht überholen können).
Auf diese Weise wird auch in dem verteilten Betriebssystem Amoeba eine Gruppenkommunikation mit Totalordnung realisiert [Kaashoek 91]: Der Sender einer Multicast-Nachricht sendet diese an einen Sequencer-Prozeß, der die Nachricht dann als Multicast über einen minimalen Spannbaum an alle Teilnehmer dieser Multicast-Gruppe ausliefert.
Allerdings wird mit diesem Verfahren zur Einführung der Totalordnung die Auslieferungsverzögerung etwas erhöht.

III.5.2.3 Wichtige Eigenschaften

Die **Fehlertoleranz** dieses Algorithmus ist wegen dem Core-Router als Single-Point-Of-Failure sehr schlecht. Fällt eine Kante im Spannbaum oder ein Router aus, ist die ganze Multicast-Gruppe kritisch getroffen. Dieser große Nachteil wird durch die Einführung der Core-List mit Stellvertretern für den Core-Router entschärft. Die betroffenen Router müssen allerdings explizit ihre Multicast-Gruppen durch *join-request*-Nachrichten neu anmelden. Immerhin ist der Ausfall von Kanten und *parent*-Knoten im Algorithmus bereits angedacht, und es werden entsprechende Lösungsmöglichkeiten geboten.

Die **Auslieferungsverzögerung** ist nicht so hoch wie beim SST-Verfahren, da nicht ein einziger Spannbaum für alle Multicast-Gruppen verwendet, sondern für jede Gruppe ein eigener optimaler Spannbaum aufgebaut wird. Dennoch schlägt der prinzipbedingte Nachteils von MST-Verfahren zu Buche, nämlich, daß ein einziger Auslieferungsbaum für alle Sender zwangsläufig längere Wege und somit höhere Auslieferungsverzögerungen mit sich bringt.

Zu den **Kosten** ist zu sagen:

- Es wird sehr wenig Rechenzeit gebraucht, da kein minimaler Spannbaum berechnet werden muß und auch sonst keine Rechenzeit benötigt wird.
- Der Algorithmus braucht wenig Bandbreite. Die einzigen Mehrkosten gegenüber dem absoluten Optimum sind gelegentliche Echonachrichten der Router an ihre parent-Knoten, um zu überprüfen, ob diese noch erreichbar sind. Das ist nicht allzu teuer.
- Der Speicherplatz, der in jedem Router gebraucht wird, ist relativ gering: In jedem Router muß für jede Multicast-Gruppe, zu der er gehört, oder für die er Non-Core-Router ist, ein Datensatz der Form (*parent*-Knoten, *child*-Knoten, *child*-Knoten, ...) gehalten werden. Das ist ein nicht sehr viel.

III.5.2.4 Wünschenswerte Eigenschaften

Eine Implementierung als **Basisdienst** ist nicht vorhanden. Es existiert nur der Algorithmus, der entsprechend programmiert werden müßte. Da es noch keine Implementierung gibt, kann auch keine Aussage über die Komplexität der **API-Schnittstelle** gemacht werden.

Der Algorithmus bietet einen **Parameter** an: Der Zeitabstand, nach dem jeder Router seine *parent*-Knoten mit Hilfe von Echonachrichten fragt, ob sie noch lebendig sind. Wird er zu groß gewählt, fällt ein Rechnerausfall und somit das Ausbleiben von Multicast-Nachrichten erst spät auf; wird er zu klein gewählt, wird Bandbreite für überflüssige Echonachrichten vergeudet.

Es gibt weder eine **Entfernungsbegrenzung** noch eine **maximale Lebensdauer** für Events.

III.5.3 Kostenabschätzungen für den CBT

- **Rechenzeit in den Routern**: sehr gering.
- **Speicherplatz in den Routern**:
  In jedem Router für jede Multicast-Gruppe, zu der er gehört, oder für die er Non-Core-Router ist, ein Datensatz der Form (*parent*-Knoten, *child*-Knoten, *child*-Knoten, ...). Das ist ein geringer Bedarf an Speicherplatz.

- **CreateGroup**:
  Entspricht dem (manuellen) Auswählen der Router für die Core-List, also des Core-Routers und seiner Stellvertreter. Dann muß die Core-List zusammen mit der ID der Gruppe beim DNS unter dem Eintrag für den Gruppennamen angemeldet werden. Die Kosten hierfür entsprechen einem Unicast und sind somit gering.
- **JoinGroup, LeaveGroup**: Unicast von maximal der Länge des Weges zum Core-Router.

- **SendMulticast**:
  Optimal geringer Verbrauch an Bandbreite. Direktes Verteilen über die Baumstruktur.

- **Verwaltungs-Overhead**: Periodische Echonachrichten von jedem Router an alle *parent*-Knoten seiner Multicast-Gruppen. Dies kann in größeren Zeitabständen passieren und ist deshalb nicht allzu teuer.

III.5.4 Zusammenfassung für den CBT

Der CBT ist aufgrund seiner sehr guten Skalierbarkeit, seiner guten Unterstützung von Mobilität, seiner sehr geringen Kosten und der leichten Aufsetzbarkeit einer Totalordnung von Events sicher ein sehr interessanter Ansatz. Was den guten Eindruck aber trübt, sind die schlechte Verteiltheit des Algorithmus, sowie die fehlende Verfügbarkeit als Basisdienst. Aus diesen Gründen ist von einer Verwendung für das Eventmanagementsystem wohl doch abzusehen.

III.6 Protocol Independent Multicast-Sparse Mode (PIM-SM)

In [Deering 95] wird ein Multicast-Algorithmus beschrieben, der die Vorteile des CBT-Multicast weiter verbessern soll: PIM.
PIM besteht aus zwei Teilen: PIM-SM (PIM-Sparse Mode) und PIM-DM (PIM-Dense Mode).

PIM-DM [Deering 97] entspricht im großen und ganzen dem bisherigen Internet-Multicast-Verfahren DVMRP. Der einzige Unterschied ist, daß Multicast-Nachrichten, die über die Kante des kürzesten Weges zum Sender kamen, an alle anderen Kanten des Routers weitergegeben werden, statt, wie bei DVMRP, nur an die Kind-Kanten des Routers. Dieser zusätzliche Overhead wurde in Kauf genommen, um unabhängig vom zugrundeliegenden Unicast-Protokoll zu sein.
Da PIM-DM dem DVMRP entspricht und zudem bisher nur den Status eines Internet Draft hat, soll in diesem Kapitel allein PIM-SM betrachtet werden. Für eine Vorstellung von DVMRP und eine Diskussion seines Aufwandes siehe Kapitel III.3.

Die große Neuigkeit bei **PIM-SM** [RFC 2117] ist, daß dies ein **MST-Algorithmus** ist, der auf Wunsch des Multicast-Empfängers für bestimmte (sehr aktive) Sender in einen **SBT-Modus** gehen kann, so daß für diese Sender ein senderbasierter Auslieferungsbaum aufgebaut wird. Dies verspricht höhere Effizienz, Kostenersparnis und sehr geringe Auslieferungszeiten für bestimmte Sender, außerdem eine bessere Fehlertoleranz der Sender, die im SBT-Modus senden, da ein Knoten- oder Kantenausfall im MST diese Sender nicht betreffen muß. PIM-SM verspricht somit die Nutzung sowohl der Vorteile der MST als auch der SBT. Im Vordergrund stand beim Entwurf von PIM-SM die Unterstützung von dünnen Gruppen, die nur über Teilbereiche des Internet verbreitet sind, da DVMRP und MOSPF gerade hier große Schwächen zeigen.

Da PIM ein Projekt der Internet Engineering Task Force ist (PIM-DM ist derzeit Internet Draft, während PIM-SM bereits RFC ist) und neben der Autorenschaft des „Multicast-Gurus“ Stephen Deering auch von der bekannten Router-Firma Cisco unterstützt wird (In der neuen Generation von Cisco-Routern ist PIM bereits implementiert), ist zu erwarten, daß PIM über kurz oder lang in größeren Teilen des Internet eingesetzt werden wird.

III.6.1 Beschreibung des Algorithmus

Der Algorithmus von PIM-SM besteht aus zwei Teilen:

- Der MST-Modus ist der Default-Zustand und baut einen MST auf, über den alle Multicast-Nachrichten einer Gruppe ausgeliefert werden. Dies funktioniert ähnlich wie bei CBT.
- Der SBT-Modus wird genutzt, wenn die Datenrate eines Senders eine bestimmte Grenze überschreitet oder der Multicast-Router eines LANs anhand anderer Gründe entscheidet, daß er einen senderbasierten Baum von diesem Sender aufbauen will.

Wie in der CBT-Architektur gibt es auch bei PIM-SM einen speziell ausgezeichneter Router, der sich Rendezvous Point (RP) nennt. Für jede Multicast-Gruppe existiert ein solcher RP. Die Adresse des RP jeder Multicast-Gruppe ist jedem Multicast-Router bekannt. Ein Multicast-Router bekommt diese Informationen von einem Bootstrap-Router (BSR), von dem es in jeder Region genau einen gibt. Bei diesem melden sich alle RP-Kandidaten an und senden ihm regelmäßig eine Nachricht, die besagt, daß sie noch lebendig sind. Der BSR baut aus diesen Informationen ein RP-Set auf, das für jede Multicast-Gruppe genau einen (lebendigen) RP enthält, und das er an alle Multicast-Router sendet. So wissen alle Multicast-Router, welcher Router der aktive RP für eine bestimmte Multicast-Gruppe ist.

Wie bei CBT wird auch hier die Baumstruktur einer Multicast-Gruppe aufgebaut, indem jeder Router, der zum Auslieferungsbaum gehört, seinen *parent*-Knoten (das ist der von ihm aus nächste Knoten in Richtung Core-Router bzw. RP) sowie seine *child*-Knoten (seine Kindknoten in Bezug auf den Auslieferungsbaum) speichert.

PIM-SM baut alle seine Auslieferungsbäume empfängerbasiert auf, d.h. ein (senderbasierter) Auslieferungsbaum wird aufgebaut, indem vom Empfänger eine ***join*-Nachricht** schrittweise (also Router für Router, bzw. Hop für Hop) in Richtung Sender gesendet wird, bis sie beim Sender selbst ankommt oder auf einen Router trifft, der bereits Teil des senderbasierten Auslieferungsbaumes dieses Senders für diese Gruppe ist. Hierbei wird automatisch der Auslieferungsbaum auf den neuen Teilnehmer erweitert, d.h. alle dazwischenliegenden Router legen für diesen Sender und diese Gruppe einen Eintrag für *parent*- und *child*-Knoten an.

Ebenso wird das Abbauen bzw. Zurückschneiden eines Auslieferungsbaum durch den Empfänger initiiert: Wenn ein Multicast-Router keinen Teilnehmer der Gruppe in seinem LAN mehr hat, und er in bezug auf diese Gruppe ein Blattknoten im (senderbasierten) Auslieferungsbaum ist, also keine *child*-Knoten mehr hat, dann sendet er eine ***prune*-Nachricht** an seinen *parent*-Knoten, worauf dieser ihn aus seiner *child*-Knoten-Liste löscht und überprüft, ob er selbst eine *prune*-Nachricht weiterschicken muß.

Das Protokoll selbst funktioniert nun folgendermaßen:
Im **MST-Modus**, der die Voreinstellung bei PIM-SM ist, sendet ein Multicast-Router bei einem neuen JoinGroup eine *join*-Nachricht schrittweise in Richtung des RP dieser Gruppe, wird so an den senderbasierten Auslieferungsbaum des RP angeschlossen und erhält in Zukunft alle Multicast-Nachrichten dieser Gruppe über den RP. Bei einem LeaveGroup sendet der Multicast-Router eine *prune*-Nachricht an seinen *parent*-Knoten und erhält nun keine Multicast-Nachrichten dieser Gruppe mehr, weil er vom Auslieferungsbaum des RP abgeschnitten wurde.
Will ein Sender eine Multicast-Nachricht an eine Multicast-Gruppe senden, muß er eine *register*-Nachricht an den RP dieser Gruppe senden. Die *register*-Nachricht ist ein direkter Unicast an den RP (kein schrittweises Senden wie bei *join*- und *prune*-Nachrichten). In der *register*-Nachricht ist die zu sendende Multicast-Nachricht gekapselt. Wenn der RP die *register*-Nachricht erhält, registriert er den Sender als Multicast-Sender dieser Multicast-Gruppe. Falls die Senderate dieses neuen Senders hoch genug ist, baut der RP eine Verbindung zu ihm auf, indem er ihm schrittweise eine *join*-Nachricht sendet. Hierbei ist also der RP der Empfänger eines senderbasierten Baumes. Nun kann der Sender Nachrichten senden, die alle beim RP ankommen, worauf dieser sie über den senderbasierten Baum, dessen Sender er ist, an die Teilnehmer der Gruppe weiterleitet.

Nun kann es vorkommen, daß ein Multicast-Router entscheidet, daß er für einen bestimmten Sender S in den **SBT-Modus** gehen will, d.h. daß er die Multicast-Nachrichten dieses Senders direkt von ihm erhalten will, statt über den RP, z.B. weil das Nachrichtenvolumen dieses Senders eine bestimmte Schwelle überstiegen hat, oder weil er auf eine geringere Auslieferungsverzögerung angewiesen ist. Dann sendet er eine *join*-Nachricht schrittweise an den Sender S und wird somit in dessen senderbasierten Auslieferungsbaum aufgenommen. Wenn der Multicast-Router die erste Nachricht direkt von S erhalten hat, weiß er, daß seine Aufnahme in den Auslieferungsbaum von S vollzogen ist. Dann sendet er eine *prune*-Nachricht an seinen *parent*-Knoten bzgl. dem RP, in der er mitteilt, daß er zwar noch Multicast-Nachrichten dieser Gruppe erhalten will, aber keine Nachrichten, die von Sender S stammen (da er sie ja nun direkt erhält). Will der Multicast-Router für Sender S wieder in den MST-Modus wechseln, sendet er einfach eine *prune*-Nachricht Richtung S und eine *join*-Nachricht an seinen *parent*-Knoten bzgl. dem RP, in der er mitteilt, daß er wieder Nachrichten des Senders S zugestellt bekommen haben will.

Um die Fehlertoleranz zu erhöhen, werden *join*-, *prune*- und *register*-Nachrichten in periodischen Zeitabständen wiederholt. Um nicht zu viel Bandbreite zu verbrauchen, kann der Zeitabstand von der Netzlast abhängig gemacht werden. Mehrere hintereinander ausbleibende Wiederholungsnachrichten führen dazu, daß ihre Wirkung, z.B. ein JoinGroup, rückgängig gemacht wird.

Diese Beschreibung des PIM-SM soll nur einen kurzen Überblick geben, weshalb ich nicht auf die Einzelheiten der verschiedenen Nachrichtentypen und Datenstrukturen eingehen möchte. Das würde viel Platz beanspruchen und ist nicht wesentlich im Rahmen dieser Arbeit. Für ausführliche Informationen verweise ich auf [RFC 2117].

III.6.2 Bewertung des Algorithmus

III.6.2.1 Notwendige Eigenschaften

Dieser Algorithmus ist darauf angelegt, daß er gut **skalieren** soll. Wenn nur vergleichsweise wenige Sender im SBT-Modus betrieben werden, also eigene senderbasierte Bäume aufbauen, trifft dies auch völlig zu.

Der Algorithmus arbeitet nicht sehr **verteilt**. Er braucht eine zentrale Instanz, den RP, der als Anlaufpunkt für alle Operationen im MST-Modus benötigt wird. Das führt einen Single-Point-Of-Failure ein, der mit Hilfskonstrukten, wie dem RP-Set, das der BSR regelmäßig versendet, beseitigt wird. Dies gelingt besser als beim CBT.

Die **Anonymität** der Multicast-Teilnehmer ist gewährleistet. JoinGroup und LeaveGroup sind problemlos dynamisch möglich.

Mobilität von Sendern wird nicht unterstützt: Wenn ein Sender wechseln will, muß er den senderbasierten Baum, dessen Sender er ist, abbauen und alle Teilnehmer, die von ihm Nachrichten im SBT-Modus empfangen, veranlassen, sich wieder an den RP zu wenden. Das Anmelden im Ziel-LAN entspricht dann nur einem Unicast von der doppelten Entfernung zum RP (*register*-Nachricht und *join*-Nachricht).

Mobilität von Empfängern wird relativ gut unterstützt: Der Multicast-Router des LANs, das der Empfänger verläßt, muß sich abmelden (LeaveGroup), was im schlechtesten Fall einige *prune*-Nachrichten nach sich zieht (auch an alle Sender im SBT-Modus), und der Multicast-Router des Ziel-LANs muß sich anmelden (JoinGroup), was im schlimmsten Fall eine *join*-Nachricht bis zum RP mit sich bringt. Das ist nicht allzu teuer.

III.6.2.2 Auf höheren Schichten implementierbare, notwendige Eigenschaften

Die **Zuverlässigkeit** entspricht der des unterliegenden Unicast-Protokolls, bei IP handelt es sich also um „Best Effort“-Zuverlässigkeit.

Über die **Ordnung** von Events kann keine Aussage gemacht werden. Wenn alle Teilnehmer einer Multicast-Gruppe ihre Multicast-Nachrichten im MST-Modus empfangen würden, wäre eine implizite Totalordnung gegeben (unter der Annahme, daß Nachrichten, die bei einem Router ankommen, in FIFO-Ordnung von ihm weitergegeben werden und sich so nicht überholen können). Sobald aber ein Empfänger von einem Sender Nachrichten im SBT-Modus empfängt, gilt dies nicht mehr.

III.6.2.3 Wichtige Eigenschaften

Die **Fehlertoleranz** dieses Algorithmus ist wegen dem RP als Single-Point-Of-Failure sehr schlecht. Dieser große Nachteil wird durch die Einführung des RP-Set, das der BSR regelmäßig versendet, entschärft. Dies gelingt besser als beim CBT, weil der BSR dafür sorgt, daß alle Router den gleichen RP für eine Multicast-Gruppe verwenden. Zudem sorgen die Wiederholungsnachrichten dafür, daß beim Wechsel des RP sich alle Teilnehmer in kurzer Zeit beim neuen RP anmelden.

Die **Auslieferungsverzögerung** kann den Bedürfnissen angepaßt werden: Ist eine geringe Auslieferungsverzögerung nötig, kann man im SBT-Modus arbeiten, was aber einen höheren Speicherbedarf in den Routern mit sich bringt. Sonst werden die Nachrichten über den RP ausgeliefert, was langsamer geht, aber nur sehr wenig Speicherplatz in den Routern kostet.

Zu den **Kosten** ist zu sagen:

- Es wird sehr wenig Rechenzeit gebraucht, da kein minimaler Spannbaum berechnet werden muß und auch sonst keine Rechenzeit benötigt wird.
- Der Algorithmus braucht wenig Bandbreite. Hier sind eigentlich nur die Wiederholungsnachrichten zu erwähnen, deren Häufigkeit aber abhängig von der Netzlast gemacht werden kann.
- Der Speicherplatz, der in jedem Router gebraucht wird, hängt stark ab von der Anzahl der senderbasierten Bäume, die aufgebaut werden, und somit von der Anzahl der Sender, die im SBT-Modus arbeiten und deshalb einen eigenen Auslieferungsbaum aufbauen. Würden alle Sender im MST-Modus arbeiten, wäre der Bedarf an Speicherplatz relativ gering.

III.6.2.4 Wünschenswerte Eigenschaften

Eine Implementierung als **Basisdienst** ist im Internet wahrscheinlich in Kürze vorhanden (die aktuelle Generation der Cisco-Router hat PIM bereits implementiert) und mit IGMP existiert auch eine sehr gute **API-Schnittstelle** zum DVMRP.

Der Algorithmus bietet zwei **Parameter** an, die allerdings in der Implementierung als Basisdienst bereits fest voreingestellt und für die Anwendung nicht erreichbar sind:

- Der Zeitabstand, nach dem jeder Router Wiederholungnachrichten (*join*-, *prune*- und *register*-Nachrichten) sendet. Wird er zu groß gewählt, fällt ein Rechnerausfall und somit das Ausbleiben von Multicast-Nachrichten erst spät auf; wird er zu klein gewählt, wird Bandbreite für überflüssige Wiederholungsnachrichten vergeudet. Dieser Parameter kann aber auch in Abhängigkeit von der Netzlast gewählt werden.
- Man kann wählen, welche Kriterien erfüllt sein müssen, damit ein Multicast-Router von einem Sender Multicast-Nachrichten im SBT-Modus anfordert. Kriterien sind z.B. Aktivität und Datenrate des Senders, gewünschte Auslieferungsverzögerung...

Es gibt weder eine **Entfernungsbegrenzung** noch eine **maximale Lebensdauer** für Events.

III.6.3 Kostenabschätzungen für PIM-SM

- **Rechenzeit in den Routern**: sehr gering.
- **Speicherplatz in den Routern**:
  In jedem Router für jeden senderbasierten Baum, in dem er ein Knoten des Baumes ist: Informationen über den *parent*-Knoten, über die *child*-Knoten, sowie über die Sender einer Gruppe, die über manche *child*-Knoten nicht weitergeleitet werden brauchen. Die Menge an benötigtem Speicherplatz hängt stark von der Anzahl der senderbasierten Bäume ab, bzw. der Anzahl von Sendern im MST-Modus.

- **CreateGroup**:
  Entspricht einem Unicast an den BSR, in dem der Router sich als RP-Kandidat für diese Gruppe meldet, und ist somit günstig.
- **JoinGroup, LeaveGroup**: Unicast (*join*- bzw. *prune*-Nachricht) von maximal der Länge des Weges zum RP bzw. Sender.

- **SendMulticast**:
  Entspricht beim ersten Senden einer *register*-Nachricht an den RP und einer *join*-Nachricht zurück an den Sender, also einem Unicast von der doppelten Länge des Weges von Sender zu RP. Bei weiteren Sendeoperationen einfach direktes Verteilen über die nun aufgebaute Baumstruktur.

- **Verwaltungs-Overhead**: Periodische Wiederholungsnachrichten (*join*-, *prune*-, *register*-Nachricht) von jedem Router an alle *parent*-Knoten seiner senderbasierten Bäume. Dies kann in größeren Zeitabständen passieren oder in Abhängigkeit von der Netzlast, und ist deshalb nicht allzu teuer.

III.6.4 Zusammenfassung für PIM-SM

PIM-SM bietet alle Vorteile des CBT-Algorithmus, da der MST-Modus dem CBT entspricht. Zu diesen Vorteilen zählen vor allem eine gute Skalierbarkeit, eine gute Unterstützung von Mobilität, geringe Kosten und, wenn nur der MST-Modus verwendet wird, eine implizite Totalordnung der Multicast-Nachrichten. Die Fehlertoleranz ist gegenüber CBT verbessert worden.
Die Möglichkeit, mit dem SBT-Modus für manche Sender die geringere Auslieferungsverzögerung von SBT zu nutzen, und die anzunehmende zukünftige Verfügbarkeit als Basisdienst im Internet, machen PIM-SM für ein Eventmanagementsystem sehr interessant.

III.7 Mobile Object Multicast Protocol (MOMP)

MOMP ist ein Eventmanagementsystem, das Bernhard Beck in seiner Diplomarbeit [Beck 97] für das Mole-Projekt der Universität Stuttgart geschrieben hat. Zusätzlich zu den Multicast-Fähigkeiten bietet dieses Protokoll in besonderem Maße Unterstützung für mobile Teilnehmer, im Falle von Mole mobile Agenten.

III.7.1 Beschreibung des Algorithmus

MOMP besteht aus zwei Ebenen: Der LAN-Ebene, auf der das eigentliche MOMP-Protokoll eingesetzt wird, und der Router-Ebene, auf der das RB-Protokoll eingesetzt wird.

Die Router-Ebene:
Jedes LAN, in dem Teile des MOMP-Systems laufen, hat einen speziell ausgezeichneten Rechner: Den **Koordinator**. Dieser Koordinator übernimmt die Aufgaben, die bei IGMP der Multicast-Router des LANs übernimmt: Er überwacht die Gruppenmitgliedschaften in seinem LAN, sorgt dafür, daß er an die entsprechenden Auslieferungsbäume der Multicast-Gruppen angeschlossen ist, und gibt Multicast-Nachrichten vom und an das LAN weiter.

Die einzelnen Koordinatoren bauen unter Verwendung des Gallager-Algorithmus zur verteilten Berechnung eines minimal spannenden Baumes [Gallager 83] einen MST auf, über den sie durch das RB-Protokoll die Multicast-Nachrichten verteilen.

Kurzbeschreibung des RB-Protokolls (ausführliche Beschreibung siehe Kapitel III.2):
Für alle Multicast-Gruppen wird ein einziger minimal spannender Baum als Multicast-Backbone verwendet. Bei einem JoinGroup sucht ein neuer Gruppenteilnehmer den nächsten Teilnehmer dieser Gruppe, worauf der Auslieferungsbaum entlang des Multicast-Backbone bis zu dem neuen Teilnehmer erweitert wird. Bei einem LeaveGroup wird der Auslieferungsbaum dieser Gruppe entsprechend zurückgestutzt.

Hierbei wird das RB-Protokoll in seiner ursprünglichen Fassung verwendet, also ohne die zusätzliche Speicherung von Gruppenmitgliedschaftsinformationen der Subbäume eines Knotens, wobei die höheren Kommunikationskosten für das Suchen des nächsten Knotens einer unbekannten Multicast-Gruppe in Kauf genommen werden.

Die LAN-Ebene:
Auf jedem Rechner des LANs, auf dem das MOMP-System läuft, arbeitet ein Dämon, der die Gruppenmitgliedschaften der Prozesse auf dem Rechner zum LAN hin vertritt. Der Dämon sendet und empfängt Multicast-Nachrichten über den LAN-Multicast und berichtet seinem zuständigen Koordinator über die bei ihm vertretenen Multicast-Gruppen.

Das eigentliche MOMP-Protokoll behandelt den Fall der Migration eines Gruppenteilnehmers und geschieht analog zum Soft-Handover bei Mobilfunknetzen: Es könnte ja passieren, daß ein migrierender Teilnehmer Multicast-Nachrichten verpaßt, die während seiner Migration beim verlassenen oder beim Zielrechner ankamen, oder die bei seinem Zielrechner bereits vor seiner Migration ausgeliefert wurden, während sie bei seinem verlassenen Rechner noch nicht angekommen sind. Um dies zu verhindern, wird vor seiner Migration eine Hilfsverbindung zu sei-

nem Zielrechner aufgebaut, über den alle Nachrichten der Multicast-Gruppen, zu denen der Teilnehmer gehört, direkt mit einer Unicast-Verbindung an den Zielrechner weitergeleitet werden. Nachrichten dieser Art, die während der Migration des Teilnehmers beim verlassenen Rechner ankommen, werden verzögert und nach vollendeter Migration über die Hilfsverbindung an den Teilnehmer am Zielrechner ausgeliefert. Der Zielrechner wird nach der Migration in den Auslieferungsbaum der Multicast-Gruppe(n) des migrierenden Teilnehmers über ein JoinGroup aufgenommen (falls er noch nicht zu diesen Gruppen gehört). Dann sendet der migrierte Teilnehmer eine „Bin-angekommen"-Nachricht an diese Multicast-Gruppen, und wenn diese Nachricht über die Hilfsverbindung wieder zu ihm zurückkommt, weiß er, daß die Hilfsverbindung abgebaut werden kann. Dies ist der Fall, weil zwischen den Koordinatoren TCP-Verbindungen verwendet werden und deshalb Nachrichten sich nicht überholen können. Für eine Diskussion der Korrektheit dieses Ansatzes sei auf [Beck 97] verwiesen.

III.7.2 Bewertung des Algorithmus

III.7.2.1 Notwendige Eigenschaften

Dieser Algorithmus **skaliert** genauso wie der RB-Algorithmus, da der RB-Algorithmus, der auf Koordinatoren-Ebene läuft, das Skalierungsverhalten bestimmt: Pro Router ist nur ein Tabelleneintrag für jede Multicast-Gruppe erforderlich ist, allerdings bringt das Suchen von Teilnehmer einer bestimmten Multicast-Gruppe (bei Anmelde- oder Sendevorgang an eine dem Router noch unbekannte Gruppe), das so teuer werden kann wie ein Multicast über den gesamten Spannbaum ein eher schlechtes Skalierungsverhalten. Die Kosten, die der MOMP-Algorithmus zur Unterstützung der Mobilität mit sich bringt, können bei sehr mobilen Teilnehmer relativ groß werden, dürften im Durchschnitt aber nicht sehr ins Gewicht fallen.

Für die **Verteiltheit** des Algorithmus sowie die **Anonymität** der Teilnehmer gilt das Gleiche wie beim RB-Algorithmus: Beides wird sehr gut unterstützt.

Mobilität von Sendern und Empfängern wird gut unterstützt. Dazu dient ja schließlich die MOMP-Erweiterung des RB-Algorithmus: Wenn der Multicast-Router des LAN, zu dem migriert wird, die Multicast-Gruppe bereits kennt, kann der migrierende Empfänger sofort an den Auslieferungsbaum angeschlossen werden, und die Dauer, bis die Hilfsverbindung abgebaut werden kann, ist geringer, als wenn im gesamten Spannbaum ein Router gesucht werden muß, der die entsprechende Gruppe kennt. Die bei der Migration entstehenden Kosten sind durch den Aufbau, den Betrieb und den Abbau der Hilfsverbindung, sowie durch das Multicasting der „Bin-angekommen"-Nachricht etwas höher als beim RB. Dafür gehen dem migrierenden Teilnehmer keine Multicast-Nachrichten verloren, und er braucht die Zeit, bis er an den Auslieferungsbaum angeschlossen ist, nicht abzuwarten, bevor er weitere Nachrichten der Gruppe erhält.

III.7.2.2 Auf höheren Schichten implementierbare, notwendige Eigenschaften

Die **Zuverlässigkeit** entspricht der des unterliegenden Unicast-Protokolls, bei IP handelt es sich also um „Best Effort"-Zuverlässigkeit.
Über die **Ordnung** von Events kann keine Aussage gemacht werden.

III.7.2.3 Wichtige Eigenschaften

Fehlertoleranz ist wie beim RB nicht vorgesehen, auch die **Auslieferungsverzögerung** entspricht der des RB: Es wird ein einziger Spannbaum für alle Multicast-Gruppen verwendet, statt für jede Gruppe einen eigenen optimalen Spannbaum zu verwenden.

Zu den **Kosten** ist zu sagen:

- Es wird wenig Rechenzeit gebraucht, da nur ein einziges Mal ein minimaler Spannbaum über alle Router berechnet werden muß, falls keine Fehler auftreten. Hinzu kommt allerdings noch die nötige Duplikaterkennung für migrierte Teilnehmer, deren Hilfsverbindung noch nicht abgebaut werden konnte.
- Für Bandbreite und Speicherplatz gilt: Die Bandbreite, die für Verwaltungs-Overhead gebraucht wird, ist unter Umständen ziemlich hoch, während der Verbrauch an Speicherplatz in jedem Router sehr gering ist. Das Verwalten und Verwenden der Hilfsverbindung sollte nicht zu teuer sein, auch der zusätzliche „Bin-angekommen“-Multicast dürfte nicht sehr ins Gewicht fallen.

III.7.2.4 Wünschenswerte Eigenschaften

Eine Implementierung als **Basisdienst** ist nicht vorhanden, wenn man einmal von einer Alpha-Version im Mole-Projekt der Universität Stuttgart absieht. Es existiert nur der Algorithmus, der entsprechend programmiert werden müßte. Da es noch keine Implementierung gibt, kann auch keine Aussage über die Komplexität der **API-Schnittstelle** gemacht werden.

Der Algorithmus selbst bietet keine **Parameter** an.

Es gibt weder eine **Entfernungsbegrenzung** noch eine **maximale Lebensdauer** für Events.

III.7.3 Kostenabschätzungen für MOMP (mit RB)

Hierbei sind die Router mit den Koordinatoren gleichzusetzen.

- **Rechenzeit in den Routern**: gering, evtl. durch Duplikaterkennung etwas höher als RB.
- **Speicherplatz in den Routern bzw. Rechnern**:
  1.) Einen Datensatz *GroupRec* der Form (*Adresse*, *status*, *Kante, Kante*, ...) in jedem Router für jede dem Router bekannte Multicast-Gruppe, also für die Gruppen, deren Teilnehmer er ist, sowie die Gruppen, bei denen er auf einem Ast zwischen zwei Teilnehmern liegt.
  2.) Eine Hilfsverbindung von jedem Rechner, von dem ein Teilnehmer zu einem anderen Rechner migriert ist. Diese Hilfsverbindung kann erst entfernt werden, wenn eine Multicast-Nachricht des migrierten Teilnehmers, die über den Baum gesendet wurde, bei dem Teilnehmer über die Hilfsverbindung wieder ankam. Das kann bei weiten Entfernungen unter Umständen lange dauern.

- **CreateGroup** (=erstes JoinGroup): Ein Broadcast über alle Router.
- **JoinGroup**: Zwischen keiner Nachricht und einem Broadcast.
- **LeaveGroup**: Keine Nachricht oder Zurückschneiden des Auslieferungsbaumes für diese Multicast-Gruppe.
- **SendMulticast**: Effizientes Verteilen der Nachricht über den Backbone-Spannbaum und eventuelles Weiterleiten über Hilfsverbindungen.

- **Verwaltungs-Overhead**: Aufbauen einer Hilfsverbindung bei Migration, sowie ein „Bin angekommen“-Multicast.

III.7.4 Zusammenfassung für MOMP

MOMP entspricht in großen Teilen dem RB: Er hat ein gutes Skalierungsverhalten, aber es kann immer wieder zu teuren Multicasts über einen großen Teil des Spannbaumes kommen. Die Unterstützung für die Mobilität der Multicast-Teilnehmer ist durch die MOMP-Erweiterung zum RB recht gut, vor allem ist es vorteilhaft, daß die Nachrichten, die der Teilnehmer durch die Migration verpassen würde, durch die Hilfsverbindung trotzdem ankommen. Die Nachteile, die man sich dadurch gegenüber dem reinen RB einhandelt, Hilfsverbindungen, „Bin angekommen“-Multicasts und Duplikaterkennung, können im schlimmsten Fall recht teuer sein, sollten aber im Normalfall erträglich sein.

III.8 Tabellarische Bewertungsübersicht

	Notwendige Eigenschaften					**Implementierbare Eigenschaften**		**Wichtige Eigenschaften**			**Wünschenswerte Eigenschaften**				
	Skalierbarkeit	Verteiltheit	Anonymität von Multicast-Teilnehmern	Sender-Mobilität	Empfänger-Mobilität	Zuverlässigkeit	Ordnung von Events	Fehlertoleranz	Geringe Auslieferungsverzögerung	Geringe Kosten	Verfügbarkeit als Basisdienst	Einfache API-Schnittstelle	Anzahl der einstellbaren Parameter	Entfernungsbegrenzung für Events	Maximale Lebensdauer für Events
SST	- -	++	++	++	- -	-	-	-	- -	+	-	-	**2**	-	-
RB	-	++	++	++	/ -[1]	-	-	-	- -	+	-	-	**0**	-	-
DVMRP	**0**	++	++	-	+	-	-	+	++	-	+	+	**1**[4]	+	-
MOSPF[2]	+	++	++	++	**0**	-	-	++	**0**	**0**	+	+	**2**[4]	+	-
CBT	+	-	++	++	+	-	**0**	-	-	++	-	-	**1**	-	-
PIM-SM[3]	+	**0**	++	-	+	-	**0**	**0**	**0**	+	+	+	**2**[4]	-	-
MOMP	-	++	++	+	+	-	-	-	- -	+	-	-	**0**	-	-

++ = sehr gut; + = gut; 0 = Mittelmaß; - = schlecht bzw. nicht vorhanden; - - = sehr schlecht

[1] Sehr gute Unterstützung der Mobilität (Sender und Empfänger sind gleich gut) für die modifizierte Version des RB; bei der ursprünglichen Version ist die Migration zu LANs, deren Router die Multicast-Gruppe nicht kennt, unter Umständen sehr teuer.

[2] Bei der Beurteilung des MOSPF wurde die schlechte Skalierbarkeit des darunterliegenden Unicast (OSPF) und seine hohen Kosten durch teure Broadcasts nicht berücksichtigt.

[3] PIM-SM bietet alle Vorteile des CBT, da der MST-Modus von PIM-SM dem CBT entspricht. Die teilweise schlechteren Ergebnisse sind eine Folge des Miteinbeziehens des SBT-Modus, der höhere Kosten, mehr Speicherbedarf in den Routern, schlechtere Sendermobilität, dafür aber eine geringere Auslieferungsverzögerung mit sich bringt.

[4] Diese Parameter sind bei den verfügbaren Basisdiensten bereits voreingestellt und von der Anwendung nicht beeinflußbar.

IV. Hinzufügen von Eigenschaften

Im diesem Teil sollen verschiedene Möglichkeiten diskutiert werden, Eigenschaften, die ein Multicast-Protokoll nicht bietet, in höheren Schichten zu implementieren.

IV.1 Mobilität von Multicast-Gruppen-Teilnehmern

Nur sehr wenige Multicast-Protokolle bieten eine Unterstützung für die Mobilität von Teilnehmern einer Multicast-Gruppe. Im folgenden soll untersucht werden, welche Möglichkeiten es gibt, diese Unterstützung in höheren Schichten nachzureichen. Allerdings gibt es auch in der Literatur sehr wenig Untersuchungen zu diesem Thema, so daß hier nur wenige Verfahren vorgestellt werden können.

IV.1.1 Ab- und wieder Anmelden

Dies ist die einfachste Möglichkeit: Wenn ein Teilnehmer migrieren will, meldet er sich an seinem derzeitigen Standort (Eventmanager) im Auslieferungsbaum der Multicast-Gruppe ab, migriert und meldet sich an seinem Zielstandort wieder an.
Hierbei kann es aus verschiedenen Gründen passieren, daß er Nachrichten verpaßt. Gründe hierfür können sein: die Verzögerung bei der Migration, die Verzögerung beim Wiederanmelden, oder daß Nachrichten am Zielstandort bereits ausgeliefert wurden, die beim verlassenen Standort noch nicht ausgeliefert wurden.

Um diese unerwünschten Effekte zu beseitigen, kann man analog zum Soft-Handover beim Mobiltelefon (z.B. GSM) eine Hilfsverbindung vom verlassenen Standort zum Zielstandort einrichten. Diese Hilfsverbindung übernimmt das Nachsenden der Nachrichten dieser Multicast-Gruppe, die beim verlassenen Standort während und nach der Migration eintreffen.
Die Hilfsverbindung kann abgebaut werden, sobald

- der mobile Teilnehmer am Zielstandort für die gewünschten Multicast-Gruppe angemeldet ist und somit alle weiteren Nachrichten dieser Multicast-Gruppe erhält, (Sicherstellen der zukünftigen Versorgung) und
- der mobile Teilnehmer alle Nachrichten erhalten hat, die am Zielstandort im Auslieferungsbaum bereits ausgeliefert wurden (Abgleichen der Historie)

Sobald die Hilfsverbindung abgebaut ist, kann sich der verlassene Standort von der entsprechenden Multicast-Gruppe abmelden, falls er nicht noch weitere Empfänger zu versorgen hat.

Hier stellt sich die Frage, wie sichergestellt werden kann, daß die Bedingungen für das Abbauen der Hilfsverbindung eingetreten sind:

Die Erfüllung der ersten Bedingung ist leicht zu erkennen: Sobald die erste Multicast-Nachricht beim mobilen Teilnehmer eintrifft, die nicht über die Hilfsverbindung kommt, ist diese Bedingung erfüllt.

Mit der Erfüllung der zweiten Bedingung ist es schwieriger:
Selbst bei fehlerfreier Funktion des verwendeten Multicast-Protokolls kann nicht angenommen werden, daß jeder Standort zum gleichen Zeitpunkt wie alle anderen Standorte die gesendeten Nachrichten erhält. Daher ist ein Abgleichen der Historie mit dem Zielstandort nötig, um zu garantieren, daß dem mobile Teilnehmer durch die Migration keine Nachricht verlorengeht.
MOMP (siehe Kapitel III.7) beispielsweise benutzt hierzu das Wissen, daß im verwendeten Multicast-Protokoll (auf Basis von TCP-Verbindungen) Nachrichten sich nicht überholen können.
Können solche Annahmen über das darunterliegende Multicast-Protokoll nicht getroffen werden, muß jeder Gruppenteilnehmer eine Liste aller bisher empfangenen Nachrichten führen und darin die sendereindeutige Sequenznummer jeder erhaltenen Nachricht speichern.
Aufgrund dieser Liste kann dann das Abgleichen der Historie erfolgen. In dieser Liste können Sequenznummernbereiche zusammengefaßt werden (z.B. „Bis Sequenznummer 1721 alle erhalten“), so daß der benötigte Speicherplatz pro Sender gering bleibt. Allerdings kann eine beliebige Zahl von Sendern diese Liste beliebig groß machen. Hier bietet sich die Einschränkung auf nur einen Sender pro Multicast-Gruppe an.

Der Vorteil dieses Ansatzes (An- und wieder Abmelden) ist, daß er mit allen Multicast-Algorithmen verwendet werden kann, unabhängig von den verwendeten Basisdiensten. Außerdem ist er sehr kostengünstig, wenn das An- bzw. Abmelden von Teilnehmern im gewählten Multicast-Algorithmus kostengünstig ist.

Der Nachteil liegt vor allem in der evtl. langen Verzögerungszeit bis zum erneuten Anschluß an den Auslieferungsbaum beim Zielstandort, sowie in der Problematik, zu entscheiden, wann die Hilfsverbindung abgebaut werden kann.

IV.1.2 Mobile IP

Eine weitere Möglichkeit ist, die im Internet bereits vorhandene Möglichkeit zur Unterstützung mobiler Internet-Teilnehmer zu benutzen.

Die Idee von Mobile IP [RFC 2002] ist, daß jeder mobile Rechner eine Heimadresse sowie eine entfernte Adresse („Care-of-Adresse“) hat. Die Heimadresse ist die IP-Adresse, an der der mobile Rechner „zuhause“ ist. An der Heimadresse wartet ein Heimagent („home agent“), der alle Nachrichten, die für den mobilen Rechner eingehen, an den mobilen Rechner weiterleitet. Wenn der mobile Rechner nun seine Position geändert hat und an einer neuen Position am Internet angeschlossen ist, bekommt er dort entweder eine eigene, neue IP-Adresse, oder er meldet sich bei einem Gästeagenten („foreign agent“) an, der für die neue Position zuständig ist und die Betreuung der mobilen Rechner (Gäste) an der neuen Position übernimmt. Wenn er sich bei einem Gästeagenten anmeldet, übernimmt dieser in Zukunft das Ausliefern der Nachrichten an den mobilen Rechner. Die entfernte Adresse ist die neue IP-Adresse des mobilen Rechners bzw. die IP-Adresse des Gästeagenten. Der mobile Rechner bzw. der Gästeagent registriert sich nun beim Heimagenten, indem er diesem die entfernte Adresse übermittelt. Der Heimagent leitet ab sofort alle Nachrichten für den mobilen Rechner an die entfernte Adresse weiter. Auf diese Weise erreichen diese den mobilen Rechner. Wenn ein beliebiger Rechner dem mobilen Rechner eine Nachricht schicken will, sendet er sie einfach an dessen Heimagenten und braucht sich nicht um die momentane Position des mobilen Rechners kümmern.

Im Eventmanagementsystem könnte man Mobile IP benutzen, indem jeder Multicast-Teilnehmer einem mobilen Rechner entspricht. Das Multicasting wird dann nach einem beliebigen Multicast-Algorithmus durchgeführt, wobei die Empfänger den Heimagenten entsprechen, die ihrerseits die Multicast-Nachrichten weiterleiten an die mobilen Teilnehmer.

Der Vorteil dieser Lösung ist, daß die Mobilität für den Multicast-Algorithmus völlig transparent ist und dieser sich überhaupt nicht um die Probleme kümmern muß, die auftreten, wenn Teilnehmer sich bewegen.

Der große Nachteil dieser Lösung ist, daß die mobilen Teilnehmer sich im ganzen Internet bewegen können, wodurch das Nachschicken vom Heimagent an den Gästeagenten vergleichsweise sehr teuer werden kann. Außerdem wird der Vorteil des Multicasts gegenüber einer Reihe von Unicasts, nämlich, daß die Auslieferungszeit bzw. der Verbrauch an Bandbreite optimiert wird, völlig aufgegeben. Unter Umständen wird das Multicasting durch Verwendung von Mobile IP noch wesentlich teurer als das Verwenden einer Reihe von Unicasts.
Aus diesen Gründen heraus ist die Verwendung von Mobile IP im Rahmen eines Eventmanagementsystems nicht sinnvoll.

IV.1.3 Multicast-Erweiterung für Mobile IP

In [Acharya 95] wird eine Erweiterung des Mobile IP vorgeschlagen, um mit Mobile IP auch Multicast-Protokolle wie DVMRP benutzen zu können. Das Problem, das in dieser Ausarbeitung angesprochen wird, ist, daß ein (virtuelles) Subnetz von mobilen Rechnern, die durch foreign agents (hier Mobility Support Routers genannt), nicht ein LAN sein muß. Das heißt, zwei mobile Rechner können im gleichen Subnetz sein, aber über verschiedene Gästeagenten an das Internet angeschlossen sein.
Da dieses Problem nicht der Problemstellung dieser Arbeit entspricht, sollen das Problem und das Protokoll, das zur Lösung dieses Problem vorgeschlagen wird, hier nicht ausführlich erläutert werden. Es soll nur betrachtet werden, wie diese Lösung in ein Eventmanagementsystem übertragen aussehen würde:
Alle Eventmanager bilden eine Multicast-Gruppe. Hierzu wird beispielsweise der IP-Multicast verwendet. Jede Multicast-Nachricht wird an die ganze Eventmanager-Multicast-Gruppe gesendet. Jeder Eventmanager, bei dem sich ein Empfänger für die betreffende Multicast-Gruppe befindet, liefert die Nachricht aus, die anderen Eventmanager verwerfen die Nachricht.

Der Vorteil dieser Lösung ist die äußerst geringe Zeit, bis ein migrierender Teilnehmer am Zielstandort an seine Multicast-Gruppe angeschlossen ist, da jede Nachricht ja jeden Standort erreicht.

Der große Nachteil dieser Lösung liegt jedoch in den hohen Kosten, die dadurch entstehen, daß jede Nachricht an ALLE Eventmanager gesendet wird. Dadurch skaliert diese Lösung auch sehr schlecht und ist für die Verwendung in einem Eventmanagementsystem ungeeignet.

IV.1.4 Zusammenfassung

Die einfachste Alternative, nämlich den Multicast-Teilnehmer beim zu verlassenden Standort abzumelden, ihn nach der Migration beim Zielstandort anzumelden und die Kontinuität des Multicast-Nachrichtenstromes durch eine Hilfsverbindung aufrechtzuerhalten, stellt sich als vorteilhaft und kostengünstig im Vergleich zu den anderen Möglichkeiten heraus. Daher sollte diese Alternative gewählt werden, sofern der verwendete Multicast-Algorithmus von sich aus keine Unterstützung für mobile Teilnehmer bietet.

IV.2 Zuverlässigkeit

Zuverlässigkeit als die Eigenschaft, die sicherstellt, daß Nachrichten beim Empfänger ankommen, ist eine sehr wichtige Eigenschaft für ein Eventmanagementsystem. Allerdings bieten viele Multicast-Protokolle nur eine „Best Effort"-Zuverlässigkeit, wie beispielsweise auch IP sie bietet. Für diese Protokolle muß eine Möglichkeit gefunden werden, Zuverlässigkeit auf höherer Ebene zu garantieren. Hierfür gibt es einige Ansätze. An dieser Stelle sollen die wichtigsten davon vorgestellt werden. Sie werden in [Levine 97] ausführlich beschrieben.

IV.2.1 Sender-initiierte Zuverlässigkeit

Protokolle, die diesen Ansatz wählen, legen die Verantwortung für die zuverlässige Zustellung von Nachrichten auf den Sender. Der Sender muß alle Empfänger seiner Multicast-Gruppe kennen. Nachdem er eine Nachricht gesendet hat, muß er diese Nachricht so lange für eine eventuell benötigte erneute Übermittlung (Retransmit) aufbewahren, bis er von allen Empfängern die Bestätigung (ACK) hat, daß die Nachricht bei ihnen angekommen ist. ACKs werden also benötigt, um den für das Aufbewahren von Nachrichten benötigten Retransmit-Speicher, klein zu halten, und entscheiden zu können, wann Nachrichten aus diesem Speicher gelöscht werden können.

Für den Retransmit kann hier zwischen zwei Strategien unterschieden werden:

- **Sender-basierter Retransmit:** Nachdem der Sender eine Nachricht gesendet hat, startet er einen Timeout. Wenn nach Ablauf des Timeouts nicht von allen Empfängern ein ACK vorliegt, sendet der Sender diesen Empfängern die Nachricht erneut (via Unicast), da er annimmt, daß diese die Nachricht nicht empfangen haben.

- **Empfänger-basierter Retransmit:** Nachdem ein Empfänger eine Nachricht erhalten hat, sendet er dem Sender ein ACK und startet einen Timeout. Wenn er nach Ablauf des Timeouts keine weitere Nachricht von dem Sender erhalten hat, nimmt er an, daß eine oder mehrere Nachrichten des Senders nicht bei ihm angekommen sind und bittet den Sender um ein Retransmit, indem er ihm die laufende Nummer der zuletzt korrekt empfangenen Nachricht sendet. Diese Strategie erfordert, daß jeder Sender periodisch Nachrichten sendet (auch wenn sie keinen Inhalt haben).

Der große Nachteil von senderinitiierten Protokollen ist

- zum einen, daß jeder Sender alle Empfänger kennen muß, was diesen Ansatz unhandlich macht, und
- zum anderen, daß das sogenannte „ACK implosion problem" auftritt, was solchen Protokollen eine sehr schlechte Skalierbarkeit einbringt. Das „ACK implosion problem" besteht darin, daß der Sender bei großen Multicast-Gruppen sehr viele Empfänger hat, die ihm alle ein ACK senden, worauf der Sender sehr viel Zeit damit verbringt, alle ACKs entgegenzunehmen und zu bearbeiten.

IV.2.2 Empfänger-initiierte Zuverlässigkeit

Protokolle, die diesen Weg wählen, um Zuverlässigkeit zu implementieren, legen die Verantwortung für die Zuverlässigkeit der Nachrichtenübertragung auf den Empfänger. Daher braucht der Sender seine Empfänger nicht kennen, und es werden auch keine ACKs verwendet. Ein Empfänger stellt fest, daß eine Nachricht nicht bei ihm angekommen ist, wenn ein Fehler bei einer Nachrichtenübertragung aufgetreten ist, wenn ein Sprung in den laufenden Nummern der empfangenen Nachrichten auftritt, oder wenn er über einen Timeout feststellt, daß er über einen bestimmten Zeitraum hinweg keine Nachricht von dem Sender erhalten hat. Stellt der Empfänger auf diese Weise das Fehlen einer Nachricht fest, teilt er dies dem Sender mit einer „Nicht empfangen"-Nachricht (NAK) mit, worauf der Sender ihm die fehlende Nachricht zuschickt.

Die Probleme des senderinitiierten Ansatzes treten hier nicht auf, aber auch dieser Ansatz hat zwei große Probleme bzw. Nachteile:

- Erstens hat der Sender keine Möglichkeit, festzustellen, welche Nachrichten bereits von allen Empfängern empfangen wurden, so daß er sie nicht mehr für einen eventuellen Retransmit aufbewahren muß. Daraus folgt, daß der Sender einen unendlich großen Nachrichtenpuffer haben muß, um für alle möglichen Fälle die Zuverlässigkeit zu garantieren.
- Zweitens kann es analog zum „ACK implosion problem" zu einem „NAK implosion problem" kommen, wenn viele Empfänger eine Nachricht nicht erhalten haben, oder wenn der Timeout bei den Empfängern zu klein gewählt wurde.

Des weiteren muß der Sender regelmäßig Nachrichten senden, auch wenn sie keinen Inhalt enthalten, da sonst die Empfänger NAKs senden, da sie glauben, eine Nachricht verpaßt zu haben.

IV.2.3 Empfänger-initiierte Zuverlässigkeit mit NAK-Vermeidung (RINA; RINA = Receiver-Initiated Protocol with NAK-Avoidance)

Dieser Ansatz funktioniert wie der Ansatz der empfängerinitiierten Zuverlässigkeit, allerdings wird hier das „NAK implosion problem" gelöst: Stellt ein Empfänger das Fehlen einer Nachricht fest, startet er einen Timeout (random timeout) mit einer zufälligen Zeitspanne als Wartezeit. Nachdem der random timeout abgelaufen ist, sendet er sein NAK mit einem Multicast sowohl an den Sender als auch an alle anderen Empfänger, worauf der Sender die fehlende Nachricht erneut an alle Empfänger sendet. Empfängt der Empfänger, der eine Nachricht vermißt, jedoch vor Ablauf des random timeouts ein NAK, das dieselbe Nachricht anfordert wie die, die dieser Empfänger vermißt, sendet er kein NAK, sondern wartet auf den Empfang der vermißten Nachricht (wofür er auch wieder einen Timeout startet). Auf diese Weise hofft man zu erreichen, daß der Sender nur ein NAK empfängt, auch wenn mehrere Empfänger die gleiche Nachricht vermissen.
Dieser Ansatz wird bereits erfolgreich beim Scalable Reliable Multicast (SRM) [Floyd 96] angewandt.

Im Vergleich zum vorigen Ansatz hat dieser Ansatz das „NAK implosion problem" gelöst, jedoch besteht immer noch das Problem, daß der Sender nicht entscheiden kann, welche Nachrichten noch für einen eventuellen Retransmit aufbewahrt werden müssen; somit müßte jeder Sender weiterhin potentiell unendlich viel Speicher zur Verfügung haben. Auch hier muß der Sender wie beim vorigen Ansatz regelmäßig (evtl. leere) Nachrichten senden.

IV.2.4 Ring-basierte Zuverlässigkeit

Ring-basierte Protokolle nutzen eine Ringstruktur, um sowohl Zuverlässigkeit als auch eine globale Ordnung von Events zu garantieren: Alle Empfänger einer Multicast-Gruppe sind in einem Ring organisiert, d.h. jeder Empfänger kennt genau einen Vorgänger und genau einen Nachfolger. Ein Token rotiert in diesem Ring.
Wenn ein Sender eine Nachricht an diese Multicast-Gruppe senden will, versieht er sie mit einer senderbezogenen Sequenznummer und sendet sie mit einem Multicast an alle Empfänger. Der Empfänger, der momentan das Token hat, antwortet mit einem ACK, worauf der Sender weiß, daß die Nachricht im Ring angekommen ist. Der Empfänger mit dem Token sendet diesen ACK, den er mit einer globalen Sequenznummer stempelt auch als Multicast an alle Empfänger des Rings. Jeder Empfänger, der sowohl die Nachricht vom Sender als auch den zugehörigen ACK erhalten hat, kann diese aufgrund der senderbezogenen Sequenznummer einander zuordnen.
Wenn ein Empfänger im Ring merkt, daß ihm eine Nachricht fehlt (z.B. indem er ein ACK bekommt, für den er keinen Nachricht erhielt, oder indem ihm eine Nummer in der Folge der globalen Sequenznummern fehlt), kann er den Empfänger mit dem Token um ein Retransmit bitten.
Wenn der Empfänger mit dem Token sein Token an seinen Ringnachfolger sendet, darf dieser Nachfolger das Token nur akzeptieren, wenn er alle gestempelten Nachrichten erhalten hat. Ist dies nicht der Fall, verweigert er die Annahme des Tokens und bittet um Retransmit der fehlenden Nachrichten. Nachdem er diese fehlenden Nachrichten erhalten hat, wird ihm das Token erneut angeboten. So kann ein Empfänger, wenn er das Token erneut erhält, sicher sein, daß alle Nachrichten, die er bis zu seinem letzten Empfang des Tokens bekommen hat, inzwischen bei allen Empfängern angekommen sind, und er diese Nachrichten aus seinem Speicher löschen kann.
Dieses Protokoll wird beispielsweise beim RMP-Projekt (Reliable Multicast Protocol) [Montgomery 94] der NASA verwendet.

Dieser Ansatz hat einige Vorteile, wie die globale Ordnung, die implizit geliefert wird, die geringe Zahl an ACKs und NAKs, die benötigt werden, oder die sichere Aussage, daß alle Nachrichten, die ein Empfänger erhalten hat, nach einem Token-Umlauf an alle anderen Empfänger ausgeliefert wurden, so daß die Anzahl von Nachrichten, die für ein Retransmit aufbewahrt werden müssen, reduziert werden kann.

Ein gewichtiger Nachteil ist aber vor allem die schlechte Skalierbarkeit: Wenn eine Gruppe viele Empfänger hat, wird der Ring sehr groß, und jeder Empfänger muß viele Nachrichten zwischenspeichern, außerdem dauert es dann lange, bis jeder Empfänger sicher weiß, daß er alle bisherigen Nachrichten erhalten hat. Auch die Kosten der beiden Multicasts (Senden der Nachricht und des ACKs) fallen bei großen Gruppen stark ins Gewicht.
Natürlich kann man einen großen Ring in mehrere kleine Ringe aufteilen. Dadurch steigt jedoch die Komplexität stark und die Zahl der benötigten Multicasts erhöht sich.

IV.2.5 Baum-basierte Zuverlässigkeit

Baum-basierte Ansätze benutzen eine Baumstruktur, um eine zuverlässige Zustellung von Nachrichten zu garantieren. Um eine sehr gute Skalierbarkeit zu erreichen, wird ein k-beschränkter Baum benutzt, d.h. ein Knoten hat maximal k Subknoten. Die Wurzel dieses Baumes sollte (muß aber nicht) der Sender sein. Somit sieht der Baum wie folgt aus: Die Wurzel hat bis zu k Kindknoten, die jeweils wiederum bis zu k Kindknoten haben usw.
Um einen optimalen Durchsatz zu erzielen, sollte der Aufbau dieses Baumes dem Auslieferungsbaum entsprechen, den das verwendete Multicast-Protokoll aufbaut.

Der Sender sendet seine Multicast-Nachrichten als Multicast an alle Empfänger. Sobald ein Empfänger eine Nachricht erhält, sendet er ein HACK (hierarchisches ACK; die Bezeichnung soll klarstellen, daß es sich hier nicht um ein normales ACK handelt, das an den Sender der Nachricht geschickt würde) an seinen Vater im Baum. Sobald ein Knoten von allen seinen Kindern HACKs erhalten hat, braucht er die empfangene Nachricht nicht mehr für einen eventuellen Retransmit bereithalten und kann sie löschen. HACKs werden also benötigt, um den für das Aufbewahren von Nachrichten benötigten Retransmit-Speicher, klein zu halten, und entscheiden zu können, wann Nachrichten aus diesem Speicher gelöscht werden können.

Aus [Levine 97] geht nicht eindeutig hervor, welche Sendestrategie gemeint ist:

- Das direkte Senden an alle Empfänger, indem der Sender einer Multicast-Gruppe seine Nachrichten mit einem Multicast direkt an alle Empfänger sendet, oder

- Das hierarchische Senden an die Empfänger, indem der Sender seine Nachricht an die Wurzel des Baumes sendet, die diese Nachricht mit einem Multicast an alle ihre Kindknoten sendet. Diese wiederum senden die Nachricht via Multicast an ihre Kindknoten usw.

Prinzipiell sind beide Strategien möglich und so sollten beide berücksichtigt werden.

Als Optimierung läßt sich vorschlagen, daß nicht für jede Nachricht ein HACK gesendet wird, sondern daß in bestimmten Zeitabständen ein HACK für alle bis dahin eingetroffenen Nachrichten generiert wird.

Auch hier kann für den Retransmit zwischen verschiedenen Strategien unterschieden werden:

- **Sender-basierter Retransmit:** Erfährt ein Knoten, der Kindknoten hat, von dem Vorhandensein einer neuen Nachricht, startet er einen Timeout. Nach Ablauf des Timeouts sendet er die Nachricht an alle seine Kindknoten, die ihm bis dahin kein HACK gesendet haben, via Unicast und startet für diese Kindknoten erneut einen Timeout.
- **Empfänger-basierter Retransmit:** Nachdem ein Empfänger eine Nachricht erhalten hat, sendet er seinem Vater im Baum ein HACK und startet einen Timeout. Wenn er nach Ablauf des Timeouts keine weitere Nachricht von dem Sender erhalten hat, nimmt er an, daß eine oder mehrere Nachrichten des Senders nicht bei ihm angekommen sind und bittet seinen Vater um ein Retransmit, indem er ihm die laufende Nummer der zuletzt korrekt empfangenen Nachricht in einem NAK sendet. Diese Strategie erfordert, daß der Sender periodisch Nachrichten sendet (auch wenn sie keinen Inhalt haben).
- **Empfänger-basierter Retransmit mit NAK-Vermeidung (tree-NAPP; NAPP = Negative Acknowlegdements with Periodic Polling):** Eine Optimierung des empfängerbasierten Retransmit, indem zusätzlich eine NAK-Vermeidung (vgl. Kapitel IV.2.3) innerhalb einer Subgruppe, also der Gruppe der Kindknoten eines Vaters, durchgeführt wird. Hier stellt sich jedoch die Frage, wozu eine NAK-Vermeidung benötigt werden soll, nachdem die Skalierbarkeit und die Vermeidung eines „NAK implosion problem“ bereits durch die k-Beschränkung des Baumes gewährleistet wird. Wenn man nicht die Strategie des hierarchischen Sendens wählt, müßte nämlich allein für die NAK-Vermeidung auf jeder Hierarchiestufe des Baumes eine eigene Multicast-Gruppe eingerichtet werden.

Die Frage stellt sich nun, welche dieser Strategien am besten ist, oder zumindest am besten für ein Multicast-Protokoll im Rahmen eines Eventmanagementsystems geeignet ist:

Ein empfängerbasierter Retransmit mit NAK-Vermeidung ist sicher nicht die optimale Lösung, da die NAK-Vermeidung eigentlich dazu dient, ein „NAK implosion problem“ zu vermeiden. Dies ist jedoch bei einem k-beschränkten Baum nicht dringend erforderlich, weil bereits die k-Beschränkung dafür eingeführt wurde, ein solches „HACK / NAK implosion problem“ zu vermeiden. Somit ist eine NAK-Vermeidung nicht nötig, bringt aber den großen Nachteil mit sich, daß für jede Subgruppe (inkl. Vaterknoten) eine eigene Multicast-Gruppe eingerichtet werden muß, und daß bei jedem Senden eines NAK ein Multicast gesendet wird. Da dies nicht dringend erforderlich, ist es ein recht teurer, überflüssiger Luxus.

Die Entscheidung zwischen senderbasiertem und empfängerbasiertem Retransmit fällt schwer, da für keine der beiden Lösungen große Vor- oder Nachteile gegenüber der anderen Lösung offensichtlich sind.
Letztlich dürfte ein senderbasierter Retransmit weniger gesendete Nachrichten und somit einen geringeren Bandbreitenverbrauch bedeuten, da die NAKs entfallen. Weil der Vaterknoten sowieso Informationen darüber halten muß, welcher seiner Kindknoten welche Nachrichten erhalten hat, sollte es keinen großen zusätzlichen Aufwand bedeuten, diese Informationen um einen Timeout zu erweitern, wofür der Timeout bei den Kindknoten entfällt. Als weiteren positiver Punkt für den senderbasierten Retransmit kann man verbuchen, daß keine periodischen Nachrichten (mit evtl. leerem Inhalt) notwendig sind.

Aus diesen Gründen heraus sollte der **senderbasierte Retransmit** gewählt werden, wenn man Zuverlässigkeit mit einem baumbasierten Ansatz implementieren will.

Baum-basierte Zuverlässigkeit wird bereits sehr erfolgreich und mit sehr guten Ergebnissen bzgl. der Skalierbarkeit und des Durchsatzes im TMTP-Projekt (Tree-based Multicast Transport Protocol) [Yavatkar 95] der Universität von Kentucky eingesetzt.

Dieser Ansatz skaliert sehr gut und löst die Probleme der vorher besprochenen Ansätze („ACK implosion problem“, „NAK implosion problem“, Sender muß alle Empfänger kennen, Sender muß unendlich großen Speicher haben), außerdem braucht der Sender nicht periodisch Nachrichten senden, wenn der senderbasierte Retransmit gewählt wird.
Der Nachteil dieses Ansatzes liegt in dem Problem, einen optimalen k-beschränkten Baum aufzubauen, der den Auslieferungsbaum des Multicast-Algorithmus nachbildet.

IV.2.6 Zusammenfassung

Die senderinitiierte Zuverlässigkeit hat sehr schwerwiegende Nachteile, vor allem auch, was die Skalierbarkeit betrifft, weshalb sie nicht empfehlenswert ist. Der empfängerinitiierte Ansatz ist schon besser, leidet allerdings auch unter einer schlechten Skalierbarkeit. Diesen Nachteil gleicht der empfängerinitiierte Ansatz mit NAK-Vermeidung aus. Allerdings existiert bei allen empfängerinitiierten Protokollen das Problem, daß der Sender potentiell unendlich viel Speicher für das Aufbewahren von gesendeten Nachrichten bereitstellen muß, da er nie weiß, welche Nachrichten bereits bei allen Empfängern angekommen sind. Die ringbasierten Protokolle schneiden durch ihre schlecht Skalierbarkeit im Vergleich zu den anderen Ansätzen negativ ab. Die baumbasierte Zuverlässigkeit hingegen kennt alle diese Probleme nicht und zeichnet sich durch eine sehr gute Skalierbarkeit aus.

In Simulationen bzgl. Durchsatz und benötigter Rechenleistung, die in [Levine 97] durchgeführt werden, schneidet tree-NAPP (siehe baumbasierte Zuverlässigkeit mit NAK-Vermeidung) durchweg als Bester vor RINA (siehe empfängerinitiierte Zuverlässigkeit mit NAK-Vermeidung) als Zweitbestem und einfachen baumbasierten Verfahren als Drittem ab. Alle anderen Ansätze schneiden wesentlich schlechter ab und zeigen außerdem eine schlechte Skalierbarkeit, indem die benötigte Rechenleistung stark steigt mit zunehmender Empfängerzahl, während gleichzeitig der Durchsatz schnell zurückgeht.

Da RINA das Problem des potentiell unendlich großen Speicherplatzes aufweist, liegt die Entscheidung für tree-NAPP oder ein anderes baumbasiertes Verfahren für den Einsatz in einem Eventmanagementsystem auf der Hand.

IV.3 Ordnung von Events

Für die Verwendung eines Eventmanagementsystems kann es erforderlich sein, daß die Events entsprechend einer Ordnungsrelation ausgeliefert werden. Hier gibt es verschiedene Ordnungsstufen: FIFO-, atomare, kausale und totale Ordnung (siehe Kapitel II.2.2)

Zusätzlich zu der gewünschten Ordnung muß der Bereich festgelegt werden, für den diese Ordnung gilt:

- **Systemweit**: Die Ordnung gilt für alle Multicast-Gruppen eines Systems. Beispielsweise würde eine systemweite FIFO-Ordnung für einen Sender, der Nachrichten für die beiden Multicast-Gruppen G1 und G2 sendet, bedeuten, daß, wenn er eine Nachricht A an G1 und danach eine Nachricht B an G2 sendet, bei allen Empfängern beider Gruppen, immer die Nachricht A vor der Nachricht B ausgeliefert wird.
- **Gruppenweit**: Die Ordnung gilt nur innerhalb einer Multicast-Gruppe. In dem obigen Beispiel wäre also nicht festgelegt, ob bei Empfängern beider Multicast-Gruppen Nachricht A vor Nachricht B ausgeliefert wird oder nicht.

In den allermeisten, wenn nicht gar allen, Fällen reicht sicher eine gruppenweite Ordnung, so daß für die Verwendung in einem Eventmanagementsystem die systemweite Ordnung nicht weiter betrachtet werden soll.

Für die Implementierung einer Ordnung auf einer höheren Schicht gibt es verschiedene Ansätze, die hier vorgestellt werden sollen.

IV.3.1 Vergabe von Sequenznummern (FIFO-Ordnung)

Jeder Sender kennzeichnet die Nachrichten, die er sendet mit einer Sequenznummer, die für diesen Sender eindeutig ist, sowie mit seinem Namen. Die Sequenznummern werden dabei von Nachricht zu Nachricht um eins inkrementiert. Ein Empfänger darf diese Nachrichten nur in der Reihenfolge der Sequenznummern ausliefern.

Der Vorteil dieser Lösung ist, daß sie kaum zusätzliche Kosten bedeutet, denn Sequenznummern sollten schon wegen der dadurch ermöglichten Duplikaterkennung verwendet werden.

Als Nachteil ist nur zu erwähnen, daß die FIFO-Ordnung keine besonders starke Ordnungsrelation ist. In vielen Fällen sollte sie jedoch ausreichend sein.

IV.3.2 Mitsenden aller bisherigen Nachrichten (Kausalordnung)

Dieser Ansatz führt eine Kausalordnung dadurch ein, daß mit jeder Nachricht E die Nachrichten E_i , von denen die Nachricht E kausal abhängig ist, mitgeschickt werden. Es wird also statt der einfachen Nachricht E ein Nachrichtenpaket gesendet, in dem sich zuerst die Nachrichten E_i und schließlich die Nachricht E befinden. Somit ist sichergestellt, daß die Nachrichten E_i vor der Nachricht E ankommen und ausgeliefert werden. Der Empfänger muß eine Duplikaterkennung durchführen, um Nachrichten nicht mehrfach auszuliefern.
[Beck 97] verwendet diesen Ansatz.

Es gibt folgende Optimierungsmöglichkeiten, um Bandbreite und Rechenzeit zu sparen:

- Der Sender kann sich evtl. merken, an welche Empfänger er Nachrichten in E_i bereits sicher ausgeliefert hat, z.B. in einem Auslieferungsbaum, wenn der Sender seine Nachbarn im Baum kennt und von diesen bereits ACKs bezüglich dieser Nachrichten bekommen hat. Dann braucht der Sender die Nachrichten nicht mehr mitschicken.

- Der Sender kann statt die ganzen Nachrichten in E_i mitzuschicken, nur deren eindeutige Kennung senden. Wenn ein Empfänger die Nachricht, auf die die Kennung sich bezieht nicht kennt oder noch nicht erhalten hat, kann dieser um ein Retransmit bitten.

Vorteilhaft ist an dieser Lösung zum einen die Einfachheit und zum anderen der geringe zusätzliche Aufwand.

Nachteilig ist dagegen der zusätzliche Verbrauch an Bandbreite. Dieser sollte jedoch bei Anwendung der Optimierungen nicht zu sehr ins Gewicht fallen.

IV.3.3 Beschränkung auf EINEN Sender (Totalordnung)

Bei diesem Ansatz gibt es eine zentrale Anlaufstelle, die allein das Recht hat, Nachrichten an die Multicast-Gruppe zu senden. Eine ähnliche Technik wird beispielsweise bei PIM-SM (siehe Kapitel III.6) verwendet; dort heißt die zentrale Anlaufstelle dann Rendezvous Point.
Jeder Sender, der vorhat, eine Nachricht an die Multicast-Gruppe zu senden, sendet diese per Unicast an die zentrale Anlaufstelle, die diese Nachricht mit einer eindeutigen Sequenznummer versieht und per Multicast an die Gruppe weitergibt. Jeder Empfänger darf die Nachrichten nur in der Reihenfolge ihrer Sequenznummern ausliefern.

Der große Vorteil dieser Lösung liegt in der Einfachheit und in den global eindeutigen Sequenznummern, die auch das Behandeln sowohl der Zuverlässigkeit als auch der Mobilität wesentlich vereinfachen. Zudem läßt sich dieser Ansatz sehr leicht verwirklichen, wenn als darunterliegendes Multicast-Protokoll ein Protokoll gewählt wird, das bereits eine zentrale Anlaufstelle bietet (wie z.B. CBT und PIM-SM).

Als Nachteil ist die Einschränkung auf die zentrale Anlaufstelle zu sehen, die einerseits als Flaschenhals wirken kann, andererseits Probleme bei der Fehlertoleranz aufwirft und schließlich die Auslieferung der Nachrichten verlangsamt, da diese Nachrichten ja zuerst an diese Anlaufstelle gesandt werden müssen.

IV.3.4 Vergabestelle für Sequenznummern (Totalordnung)

Dieser Ansatz funktioniert ähnlich wie der vorige. Allerdings darf hier jeder Sender seine Nachrichten an die Multicast-Gruppe senden. Nur ein bestimmter Empfänger der Multicast-Gruppe hat das Recht Sequenznummern zu vergeben. Empfängt er eine Nachricht, weist er ihr eine Sequenznummer zu und sendet diese Sequenznummer zusammen mit Charakteristika, die die Nachricht eindeutig bezeichnen, als Multicast an die Multicast-Gruppe. Ein Empfänger darf eine Nachricht nur ausliefern, wenn er sowohl die Nachricht als auch die zugeordnete Sequenznummer erhalten hat. Ist dies der Fall, liefert er die Nachrichten in der Reihenfolge der Sequenznummern aus.
Dieses Verfahren wird z.B. beim RMP-Projekt (Reliable Multicast Protocol) [Montgomery 94] der NASA verwendet (siehe auch Kapitel IV.2.4).

Als Vorteil sind auch hier, wie beim vorigen Ansatz, die global eindeutigen Sequenznummern zu sehen.
Aber auch die Nachteile des vorigen Ansatzes sind hier zu finden: Eine zentrale Sequenznummern-Vergabestelle mit ihrer eventuellen Funktion als Flaschenhals und den Problemen bei der Fehlertoleranz, die sie aufwirft. Hinzu kommen noch die zusätzlichen Kosten für einen zweiten benötigten Multicast pro Nachricht und die durch den zweiten Multicast bedingte weitere Verzögerung bei der Auslieferung der Nachrichten.

IV.3.5 Nutzen von kausalen Blocks (Totalordnung)

In [Macedo 93] wird ein Protokoll namens Newtop vorgestellt, das kausale Blöcke verwendet. Als Grundannahme wird vorausgesetzt, daß jedem Teilnehmer (Sender und Empfänger) einer Multicast-Gruppe die Menge aller Sender dieser Gruppe bekannt ist. Jeder Sender muß auch Empfänger dieser Gruppe sein.
Kausale Blöcke sind in Newtop Zeitabschnitte vergleichbar mit Epochen. Jeder Block enthält Nachrichten (mindestens eine), die zueinander kausal unabhängig sind. Jeder Sender i hat einen eigenen Blockzähler BC_i. Wenn die Multicast-Gruppe gebildet wird, haben alle BC_i den gleichen Wert, z.B. 0.
Will der Sender i eine Nachricht an die Gruppe senden, erhöht er seinen BC_i um eins und stempelt die Nachricht mit BC_i , bevor er sie an die Gruppe sendet. Jeder Teilnehmer der die Nachricht empfängt, vermerkt in einer Tabelle, daß er eine Nachricht von Sender i für den Block BC_i empfangen hat. Ein kausaler Block ist bei einem Teilnehmer vollständig, wenn dieser von allen Sendern eine Nachricht für diesen Block oder für spätere Blocks (dies gilt, da BC_i monoton wachsend ist) erhalten hat. Dann kann er die Nachrichten dieses kausalen Blocks an seine Anwendung ausliefern und seinen eigenen BC gleich diesem kausalen Block setzen, falls sein BC niedriger war als dieser Block.
Damit durch das Schweigen eines Senders nicht alle anderen Teilnehmer am Ausliefern ihrer Nachrichten gehindert werden, verhindert ein Timeout bei jedem Sender, daß dieser länger als eine bestimmte Zeit schweigt. Läuft dieser Timeout ab, muß der Sender eine Nachricht ohne Inhalt (aber mit seinem BC) senden.

Dieses Protokoll ist sicherlich interessant, aber die Einschränkung, daß jedem Teilnehmer einer Multicast-Gruppe jeder Sender seiner Gruppe bekannt sein muß, dürfte nur für sehr wenige Gruppen bzw. Anwendungen realisierbar sein. Für ein Eventmanagementsystem kann man dies sicher nicht voraussetzen.

IV.3.6 Zusammenfassung

Wenn eine FIFO-Ordnung für die Zwecke der Anwendung ausreichend ist, ist die Vergabe von Sequenznummern eine sehr günstige und einfache Möglichkeit, eine Ordnungsrelation einzuführen.
Wenn man jedoch eine kausale oder totale Ordnung für das Eventmanagementsystem fordert, kommen hauptsächlich die ersten beiden der vier hier vorgestellten Ansätze in Betracht: Das Mitsenden aller bisherigen Nachrichten (evtl. mit den beschriebenen Optimierungen) oder die Beschränkung auf EINEN Sender. Zwischen diesen beiden muß man dann je nach verwendetem Multicast-Protokoll entscheiden.
Die anderen beiden Ansätze schneiden schlechter ab, weil sie entweder vergleichsweise hohe Kosten und Verzögerungen (Vergabestelle für Sequenznummern) oder die Einschränkung, daß jedem Teilnehmer einer Multicast-Gruppe jeder Sender seiner Gruppe bekannt sein muß, (Nutzen von kausalen Blocks) mit sich bringen.

IV.4 Fehlertoleranz

In jedem Rechnernetz können Fehler auftreten. Ein Eventmanagementsystem sollte bis zu einem gewissen Grad damit umgehen und trotz aufgetretener Fehler weiterarbeiten können. In diesem Kapitel wird der Umgang des Eventmanagementsystems mit Fehlern diskutiert.

IV.4.1 Fehlerklassen

Zuerst einmal sollen die mögliche Fehlerarten klassifiziert werden. Man kann Fehler in vier Klassen einteilen [Jalote 94], wobei jeweils die vorher genannte eine Teilmenge der nachfolgenden Fehlerklassen darstellt:

- **Crash-Fehler**: Ein Fehler, der die Komponente zum Anhalten bringt. Sie reagiert nicht weiter.
- **Unterlassungsfehler**: Ein Fehler, der die Komponente dazu bringt, auf keine Eingaben oder Nachrichten mehr zu antworten. Sie schweigt einfach.
- **Timing-Fehler**: Ein Fehler, der die Komponente dazu bringt, entweder zu früh oder zu spät zu antworten. Meist ist der problematische Fall der, wenn die Komponente zu spät antwortet. Dann ähnelt dieser Fehler dem Unterlassungsfehler, mit dem Unterschied, daß die Komponente „irgendwann“ wieder Reaktionen zeigt.
- **Byzantinischer Fehler**: Ein Fehler, der die Komponente dazu bringt, völlig willkürlich zu handeln. Es kann unmöglich sein, dieses Verhalten vom Normalverhalten zu unterscheiden.

In dieser Arbeit sollen byzantinische Fehler nicht berücksichtigt werden, da dies den Rahmen der Arbeit sprengen würde. Hier sollen nur Timing-Fehler (und somit natürlich auch Unterlassungsfehler und Crash-Fehler) betrachtet werden.

IV.4.2 Ausfall einer Verbindung zwischen zwei Rechnerknoten

Unter Verbindung zwischen zwei Rechnerknoten soll hier nicht eine verbindungsorientierte Kommunikation verstanden werden, sondern einfach die Datenübertragung zwischen zwei benachbarten Rechnerknoten.
Es sollen bei dem zu entwerfenden Protokoll nur Basisdienste des Netzwerks verwendet werden. Diese sind auf Schicht 3 oder 4 des OSI-7-Schichten-Modells angesiedelt. Das bedeutet, daß diese Basisdienste sich bereits um die Behandlung eines Verbindungsausfalls kümmern. Diese Fehlerbehebung durch die Basisdienste ist transparent für das hier beschriebene Protokoll und muß deshalb nicht berücksichtigt werden.
Die einzigen Fehler bei der Nachrichtenübermittlung, die diese Basisdienste nicht beheben, sind verlorengegangene Nachrichten, Reihenfolgefehler bei der Auslieferung von Nachrichten sowie Duplizieren von Nachrichten. Diese Fehler müssen durch das Gewährleisten von Zuverlässigkeit, durch die Vergabe von Sequenznummern und durch eine Duplikaterkennung bei den Eventmanagern behoben werden.
Ein weiterer Fehler, den die Basisdienste nicht beheben können, besteht, wenn von einem Rechnerknoten auf keinem Wege mehr eine Verbindung zu dem gewünschten Zielknoten hergestellt werden kann. Dieser Fehler entspricht einer Netzpartitionierung.

IV.4.3 Netzpartitionierung

Eine Netzpartitionierung ist die Aufteilung des Netzes in zwei oder mehrere Teilnetze, ohne daß Verbindungsmöglichkeiten zwischen den Teilnetzen bestehen. Dies kann z.B. durch Leitungsausfall hervorgerufen werden.

Das Eventmanagementsystem hat keine Möglichkeit, bei einer Netzpartitionierung die Eventmanager, die sich in einem anderen Teilnetz befinden, zu erreichen. Daher kann das Eventmanagementsystem einen solchen Fehler auch nicht kompensieren. Allerdings sollte es dem Eventmanagementsystem möglich sein, in jedem Teilnetz auf vernünftige Weise weiterzuarbeiten und nach Wiederherstellung des gesamten Netzes, also nach Aufhebung der Netzpartitionierung, wieder zu einem konsistenten Zustand zu finden, so daß das Eventmanagementsystem dann wieder netzweit einsetzbar ist.

IV.4.4 Ausfall eines Rechnerknotens

Der Ausfall eines Rechnerknotens bedeutet für den Rest des Eventmanagementsystems zum einen, daß dieser Rechnerknoten keine Informationen mehr weitergibt, daß also beispielsweise in einer Nachrichten-Auslieferungskette ein Loch entsteht, daß von dem Eventmanagementsystem überbrückt werden muß.
Zum anderen sind beim Ausfall eines Rechnerknotens auch alle Daten, die sich auf diesem Rechner befanden, für das Eventmanagementsystem verloren. Um mit diesem Fehler umgehen zu können, muß das Eventmanagementsystem in der Lage sein, dringend benötigte Daten, die verloren gingen, zu rekonstruieren, um weiterarbeiten zu können.
Als weitere Anforderung gilt, daß ein Rechnerknoten, der ausfiel, aber nun wieder verfügbar ist, wieder in das Eventmanagementsystem eingebunden werden kann.

Die Behandlung des Ausfalls von Rechnerknoten ist die hauptsächliche Aufgabe, der sich das Eventmanagementsystem im Blick auf Fehlertoleranz zu stellen hat. Wie diese Eigenschaft im einzelnen hinzugefügt werden soll, hängt stark vom Protokoll des Eventmanagementsystems ab und kann deshalb hier nicht im einzelnen beschrieben werden. Auf die genaue Behandlung von Fehlertoleranz im Eventmanagementsystem soll deshalb erst später eingegangen werden.

V. Zusammenfassung und Ausblick

V.1 Zusammenfassung

In dieser Arbeit wurden die derzeit existierenden Multicast-Protokolle vorgestellt, kurz beschrieben und auf ihre Verwendbarkeit in einem Eventmanagementsystem geprüft.

Hierzu wurden zuerst einmal die verschieden Anforderungen an ein Multicast-Protokoll, das in einem Eventmanagementsystem eingesetzt werden, zusammengetragen und nach ihrer Wichtigkeit klassifiziert. Anschließend wurden, wie bereits erwähnt, die Multicast-Protokolle beschrieben und anhand dieser Anforderungen bewertet. Nach der Beschreibung und Bewertung der einzelnen Protokolle, wurden die Protokolle in einer Bewertungsübersicht tabellarisch einander gegenübergestellt, so daß die Stärken und Schwächen eines Protokolls im Vergleich zu anderen Protokollen leichter erkennbar sind.

Schließlich wurden noch Möglichkeiten beschrieben und bewertet, Eigenschaften zu Multicast-Protokollen hinzuzufügen, wenn sie diese Eigenschaften nicht von sich selbst aus anbieten. Diese Eigenschaften waren: Mobilität von Teilnehmern, Zuverlässigkeit der Nachrichtenübermittlung, Ordnung von Nachrichten und Fehlertoleranz.

Abschließend bleibt zu sagen, daß jedes Protokoll seine Stärken und Schwächen zeigt, so daß kein Protokoll uneingeschränkt für die Verwendung in einem Eventmanagementsystem empfohlen werden kann. Auch wenn man gewisse Schwächen toleriert, kann man nicht von einem Protokoll sagen, daß es das beste sei. Welches Protokoll das beste ist, hängt zu sehr von den gestellten Anforderungen, von den gesetzten Prioritäten und vom gewünschten Anwendungsbereich ab.

Ebenso gibt es bei den verschiedenen Möglichkeiten, Eigenschaften zu bestehenden Multicast-Protokollen hinzuzufügen, Stärken und Schwächen bei jedem Ansatz. Allerdings fällt es hier leichter, einen Ansatz von den anderen vorgestellten Ansätzen auszuwählen und als empfehlenswert hervorzuheben.

V.2 Ausblick

Nachdem hier die verschiedenen Multicast-Protokolle auf die Verwendbarkeit in einem Eventmanagementsystem geprüft worden sind, bleibt nun noch, tatsächlich ein Eventmanagementsystem zu entwerfen, das den in Kapitel II vorgestellten Anforderungen gerecht wird und evtl. auf einem der beschriebenen Multicast-Protokollen aufbaut.

Dies tue ich in meiner Diplomarbeit [Schiffer 97]. In dieser Diplomarbeit wird ein Eventmanagementsystem diskutiert und beschrieben, das eine verteilte Registerstruktur zur Verwaltung aller Multicast-Gruppen verwendet. Diese Registerstruktur wird sehr ausführlich und gründlich untersucht und sowohl textuell als auch mit einem Algorithmus beschrieben. Aufbauend auf dieser Registerstruktur ist es möglich verschiedene Multicast-Protokolle (auch gleichzeitig) für verschiedene Multicast-Gruppen zu verwenden.

Ein Beispiel für ein Multicast-Protokoll wird gegen Ende der Diplomarbeit beschrieben. Dieses Multicast-Protokoll verwendet den IP-Multicast (also hauptsächlich DVMRP) als Basisdienst und baut auf diesem einen Algorithmus auf, der skalierbar, zuverlässig und fehlertolerant arbeitet und die Mobilität von Teilnehmern unterstützt.

Um die Erstellung verschiedener Multicast-Algorithmen aufbauend auf der Registerstruktur zu ermöglichen wurde insbesondere darauf geachtet, klar definierte Schnittstellen zwischen Registerstruktur und Multicast-Algorithmus zu bieten.

Diese Diplomarbeit führt also die hier beschriebene Betrachtung und Bewertung von Multicast-Protokollen in Richtung auf die tatsächliche Entwicklung eines Eventmanagementsystems fort.

VI. Anhang

VI.1 Literatur

In den Literaturangaben sind Verweise auf „Requests for Comments“ (RFC) des Internet enthalten. RFCs sind unter anderem auf dem FTP-Server des Rechenzentrums der Universität Stuttgart verfügbar (ftp://ftp.rus.uni-stuttgart.de/pub/doc/standards/rfc).

[Acharya 95] Arup Acharya; Ajay Bakre; B.R. Badrinath:
IP Multicast Extensions for Mobile Internetworking
Department of Computer Science, Rutgers University, NJ 08903, USA;
Rutgers DCS TechReport LCSR-TR-243;
http://athos.rutgers.edu/pub/technical-reports/lcsr-tr-243.ps.Z

[Ballardie 93] Tony Ballardie; Paul Francis; Jon Crowcroft:
Core Based Trees (CBT): An Architecture for Scalable Inter-Domain Multicast Routing
Proceedings ACM SIGCOMM, October 1993, S. 85-95

[Beck 97] Bernhard Beck:
Terminierung und Waisenerkennung in einem System mobiler Software-Agenten
Diplomarbeit Nr. 1472, IPVR, Universität Stuttgart, 1997

[Belkeir 89] Nasr E. Belkeir; Mustaque Ahamad:
Low Cost Algorithms for Message Delivery in Dynamic Multicast Groups
Proceedings of IEEE Ninth International Conference on Distributed Computing, June 1989, S. 110-117

[Deering 90] Stephen E. Deering; David R. Cheriton:
Multicast Routing in Datagram Internetworks and Extended LANs
ACM Transactions on Computer Systems, Vol. 8, No. 2, May 1990,
S. 85-110

[Deering 95] S. E. Deering; D. Estrin; D. Farinacci; Van Jacobson; C. Liu; L. Wei:
Protocol Independent Multicast (PIM): Motivation and Architecture
Internet Draft by the IDMR working group of the IETF,
January 11, 1995 (expired)

[Deering 96] S. E. Deering; D. Estrin; D. Farinacci; M. Handley; A. Helmy; Van Jacobson; C. Liu; P. Sharma; D. Thaler; L. Wei:
Protocol Independent Multicast - Sparse Mode (PIM-SM): Motivation and Architecture
Internet Draft by the IDMR working group of the IETF,
October 24, 1996: draft-ietf-idmr-pim-arch-04.txt (expired)

[Deering 97] S. E. Deering; D. Estrin; D. Farinacci; A. Helmy; Van Jacobson; L. Wei:
Protocol Independent Multicast Version 2, Dense Mode Specification
Internet Draft by the IDMR working group of the IETF,
May 21, 1997: draft-ietf-idmr-pim-dm-05.txt

[Floyd 96] Sally Floyd; Van Jacobson; Ching-Gung Liu; Steven McCanne; L. Zhang:
A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing
November 1996; ftp://ftp.ee.lbl.gov/papers/srm.ps.Z

[Gallager 83] R. G. Gallager; P. A. Humblet; P. M. Spira:
A Distributed Algorithm for Minimum-Weight Spanning Trees
ACM Transactions on Programming Languages and Systems, Vol. 5,
No. 1, January 1983, S. 66-77

[Jalote 94] Pankaj Jalote:
Fault Tolerance in Distributed Systems
PTR Prentice Hall, Eaglewood, N.J. 07632, 1994

[Kaashoek 91] M. Frans Kaashoek; Andrew S. Tanenbaum
Group Communication in the Amoeba Distributed Operating System
Proceedings of the 11th Conference on Distributed Computing Systems, 1991

[Kleinrock 77] L. Kleinrock; F. Kamoun:
Hierarchical Routing for large networks; performance evaluation and optimization
Computer Networks, Vol. 1, 1977, S. 155-174

[Levine 97] Brian Neil Levine; J.J. Garcia-Luna-Aceves:
A Comparison of Known Classes of Reliable Multicast Protocols
Computer Engineering Department, University of California, Santa Cruz, CA 95064, USA (unveröffentlicht)

[Macedo 93] Raimundo A. Macêdo; Paul D. Ezhilchelvan; Santosh K. Shrivastava:
Newtop: A Total Order Multicast Protocol Using Causal Blocks
Department of Computer Science, University of Newcastle, UK
http://www.twente.research.ec.org/broadcast/trs/papers/10.ps

[Montgomery 94] Todd Montgomery:
Design, Implementation and Verification of the Reliable Multicast Protocol
Masters of Science Thesis, West Virginia University, 1994
ftp://research.ivv.nasa.gov/pub/doc/RMP/RMPThesis.ps

[RFC 966] Stephen E. Deering; David R. Cheriton:
Host Groups: A Multicast Extension to the Internet Protocol
RFC 966, December 1985

[RFC 1058] C. Hedrick:
Routing Information Protocol
RFC 1058, SRI Network Information Center, June 1988

[RFC 1131] J. Moy:
The OSPF specification
RFC 1131, SRI Network Information Center, October 1989

[RFC 2002] C. Perkins, Editor (Network Working Group):
IP Mobility Support
RFC 2002, October 1996

[RFC 2117] S. E. Deering; D. Estrin; D. Farinacci; M. Handley; A. Helmy; Van Jacobson; C. Liu; P. Sharma; D. Thaler; L. Wei:
Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification
RFC 2117, June 1997

[Schiffer 97] Andreas Schiffer:
Konzeptionierung und Implementierung eines zuverlässigen, verteilten Multicast-Protokolls für mobile Teilnehmer
Diplomarbeit Nr. 1571, IPVR, Universität Stuttgart, 1997

[Schreiber 95] Douglas Schreiber:
Multicast Routing in the Internet and on the MBone: Past, Present, and Future
http://www.nmis.org/AboutNMIS/Team/DougS/MBone.html

[Thyagarajan 95] Ajit S. Thyagarajan; Stephen E. Deering:
Hierarchical Distance-Vector Multicast Routing for the MBone
Proceedings ACM SIGCOMM 95
http://www.eecis.udel.edu/~ajit/sigcomm95.dir/hmcast.html

[Wall 80] David W. Wall:
Mechanisms for Broadcast and Selective Broadcast
Technical Report No. 190, Computer Systems Laboratory, Stanford University, June 1980

[Yavatkar 95] R. Yavatkar; J. Griffioen; M. Sudan:
A Reliable Dissemination Protocol for Interactive Collaborative Applications
In Proceedings of the ACM Multimedia '95 Conference, November 1995.
http://www.dcs.uky.edu/~griff/papers/tmtp-mm95.ps

VI.2 Glossar

Broadcast

Das gleichzeitige Versenden einer Nachricht an alle Knoten eines Netzwerks, 1-zu-alle-Nachrichtenversendung. (siehe Kapitel I.2.2)
Vergleiche auch: Multicast, Broadcast.

CreateGroup

Der Befehl im Sinne von IGMP, eine neue Multicast-Gruppe zu erzeugen. Oft wird dieser Befehl durch den ersten JoinGroup-Befehl an eine neue Gruppe implizit ausgeführt.

IP, Internet Protocol

IP ist ein verbindungsloser, datagrammorientierter Netzwerkdienst zum Versenden von Datenpaketen zwischen Rechnern. Zur Zeit aktuell ist Version 4 (IPv4), die im Laufe der nächsten Jahre durch Version 6 (IPv6 oder IPng, „next generation“) ersetzt werden soll.

IP-Multicast

Der IP-Multicast ist das Multicast-Verfahren, das im Internet verwendet wird. Alle Multicast-fähigen Router im Internet bilden zusammen den MBone, also den Multicast Backbone des Internet. (siehe auch Kapitel I.6 Multicast-Protokolle im Internet)

IGMP, Internet Group Multicast Protocol

Informationsprotokoll für Multicast-Router, um Informationen über die in einem lokalen Netz vorhandenen Host-Gruppen zu erhalten (siehe Kapitel I.2.4). Es beschreibt fünf Befehle: CreateGroup, JoinGroup, LeaveGroup, SendMulticast und ReceiveMulticast.

JoinGroup

Der Befehl im Sinne von IGMP, einer Multicast-Gruppe beizutreten, um in den Auslieferungsbaum dieser Gruppe aufgenommen zu werden und in Zukunft alle Multicast-Nachrichten, die an diese Gruppe gesendet werden, zu empfangen.

LeaveGroup

Der Befehl im Sinne von IGMP, eine Multicast-Gruppe zu verlassen, d.h. keine weiteren Nachrichten mehr von dieser Multicast-Gruppe zu empfangen.

MST, Minimum Spanning Tree

Ein Multicast, der nach dem Minimum Spanning Tree-Verfahren realisiert ist, baut nur einen einzigen Baum auf: Den minimalen Spannbaum über alle Sender und Empfänger der Multicast-Gruppe. Jede Multicast-Nachricht wird über diesen Baum verteilt, unabhängig davon, von welchem Sender sie stammt. (siehe Kapitel I.3.2)

Multicast

Das gleichzeitige Versenden einer Nachricht an eine Gruppe von Netzwerkknoten, 1-zu-n-Nachrichtenversendung. (siehe Kapitel I.2.3)
Vergleiche auch: Broadcast, Unicast.

Multicast-Gruppe

Ein Gruppe von Rechnerknoten, die unter einer Multicast-Adresse zusammengefaßt sind und alle Nachrichten empfangen, die an diese Adresse gesendet werden. Die einzelnen Teilnehmer dieser Gruppe kennen sich untereinander nicht. Ein Sender, der eine Nachricht an eine Multicast-Gruppe sendet, braucht kein Teilnehmer dieser Gruppe zu sein (offene Multicast-Gruppe) und die Teilnehmer der Gruppe nicht zu kennen.

Multicast-Router

Ein Router, der Multicast-Nachrichten effizient weiterleiten kann. Unter dem Multicast-Router eines bestimmten LANs ist der Multicast-verantwortliche Router für dieses LAN zu verstehen. Um zu vermeiden, daß Multicast-Nachrichten an LANs, die mehrere Router haben, mehrfach ausgeliefert wird, also von jedem Router, wird meist einer dieser Router als der Multicast-Router dieses LANs festgelegt. Nur dieser Router ist dafür verantwortlich, daß Multicast-Nachrichten von und zu diesem LAN gelangen.

ReceiveMulticast

Der Befehl im Sinne von IGMP, eine wartenden Multicast-Nachricht einer Multicast-Gruppe zu empfangen. Meist entfällt dieser Befehl, da Multicast-Nachrichten direkt an den Empfänger ausgeliefert werden.

Router

In dieser Arbeit ein Sammelbegriff für alle Bridges, Router und Gateways, die ein LAN an den Rest des Netzwerks anbinden oder andere Vermittlungsfunktionen innerhalb des Netzwerks wahrnehmen. Hier sei insbesondere das Routing und das Ausführen eines Multicast-Protokolls erwähnt. (siehe Kapitel I.4)

SBT, Sender Based Tree

Ein Multicast, der über Sender Based Trees realisiert ist, baut für jeden möglichen Sender von Multicast-Nachrichten einen Auslieferungsbaum aus, d.h. jede Multicast-Nachricht eines Senders wird auf annähernd kürzesten Wegen vom Sender zu den Empfängern gesendet. (siehe Kapitel I.3.1)

SendMulticast

Der Befehl im Sinne von IGMP, eine Multicast-Nachricht an alle Empfänger dieser Gruppe weiterzusenden. Das Senden bedeutet für den Sender dieser Nachricht nur das Schicken einer UDP-Nachricht an den Multicast-Router des LANs. Das Netzwerk übernimmt dann die weitere Vermittlung der Nachricht.

TCP, Transmission Control Protocol

Ein Protokoll, das auf IP aufsetzt und eine gesicherte Datenübertragung zwischen zwei Endsystemen gewährleistet.

UDP, User Datagram Protocol

Sehr einfaches, paketorientiertes Protokoll, das auf IP aufsetzt. Reihenfolgefehler und Duplikate sind möglich.

Unicast

Das Versenden einer Nachricht an genau einen Empfänger im Netzwerk, 1-zu-1-Nachrichtenversendung. (siehe Kapitel I.2.1)
Vergleiche auch: Broadcast, Multicast.